MSA 설계와 구축
(Microservice Architecture)

2019. 10.

안 종 석
james@jslab.kr
JS Lab

1

james@jslab.kr

목차

1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ **실습교재 (별도)**

JS Lab

2

1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ **실습교재 (별도)**

james@jslab.kr

JS Lab

3

1장. 개요

- **MSA(Microservice Architecture) 개요**
- **레퍼런스**
- **Cloud Native**

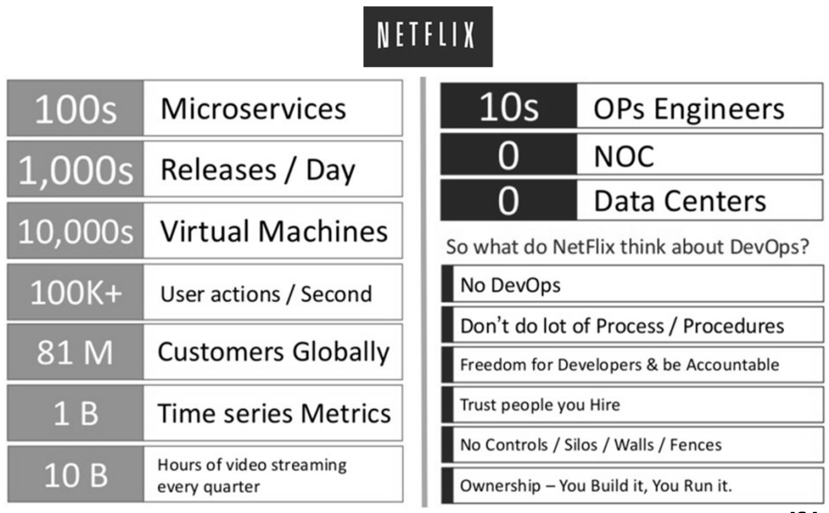
james@jslab.kr

JS Lab

4

1장. 개요

- Netflix (예)



NETFLIX

100s	Microservices
1,000s	Releases / Day
10,000s	Virtual Machines
100K+	User actions / Second
81 M	Customers Globally
1 B	Time series Metrics
10 B	Hours of video streaming every quarter

10s	OPs Engineers
0	NOC
0	Data Centers

So what do NetFlix think about DevOps?

- No DevOps
- Don't do lot of Process / Procedures
- Freedom for Developers & be Accountable
- Trust people you Hire
- No Controls / Silos / Walls / Fences
- Ownership – You Build it, You Run it.

https://www.youtube.com/watch?v=CZ3wIuvmHeM

JS Lab

james@jslab.kr

5

1장. 개요

- Wikidipia: 마이크로서비스 아키텍처
  - 마이크로서비스는 애플리케이션을 느슨히 결합된 서비스의 모임으로 구조화하는 서비스 지향 아키텍처(SOA) 스타일의 일종인 소프트웨어 개발 기법
  - MSA에서 서비스들과 프로토콜은 가벼운 편
  - 애플리케이션을 여러 서비스로 분해할 때의 장점은 모듈성을 개선, 애플리케이션의 이해, 개발, 테스트를 더 쉽게 해주고 애플리케이션 침식에 탄련적으로 만들어 줌
  - 규모가 작은 자율적인 팀들이 팀별 서비스를 독립적으로 개발, 전개, 규모 확장을 할 수 있게 함으로써 병렬로 개발
  - 지속적인 리팩터링을 통해 개개의 서비스 아키텍처가 하나로 병합될 수 있게 허용
  - 마이크로서비스 기반 아키텍처는 지속적 배포와 전개(디플로이)를 가능케 한다.

https://ko.wikipedia.org/wiki/%EB%A7%88%EC%9D%B4%ED%81%AC%EB%A1%9C%EC%84%9C%EB%B9%84%EC%8A%A4

JS Lab

james@jslab.kr

6

1장. 개요

TTA : 마이크로서비스 아키텍처

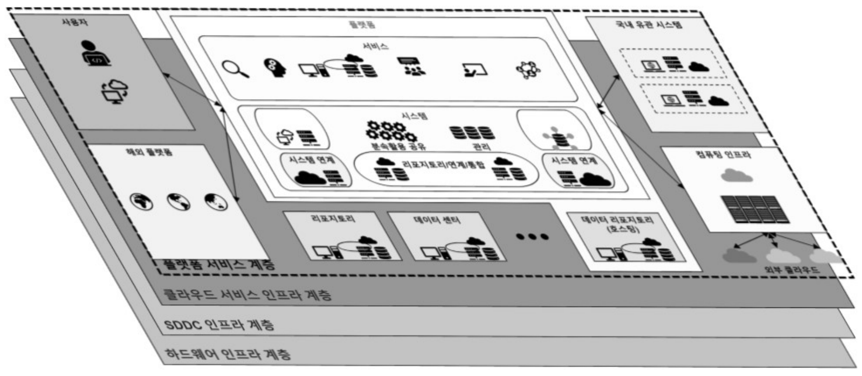
- **MSA 대규모 소프트웨어 개발에 적용하기 위한 것으로 단독으로 실행 가능하고 독립적으로 배치될 수 있는 작은 단위(모듈)로 기능을 분해하여 서비스 하는 아키텍처.**
- 작은 단위로 기능을 분할할 때 수직 방향의 기능별로 절단
- 절단된 독립적인 작은 모듈인 마이크로서비스는 공유나 프로세스 간 통신이 없이도 독립적으로 실행되며 운영 관리
- 마이크로서비스 간 연결은 응용 프로그래밍 인터페이스(API: Application Programming Service)를 이용
- 마이크로서비스는 표현이나 데이터 관리 등에 있어 기능적으로 완전
- 마이크로서비스 아키텍처 사용으로 개발자들이 클라우드 망을 통해 공유하고 협업하여 자유롭게 소프트웨어를 개발
- 개발 및 유지보수에 드는 시간과 비용이 절감
- 기존 모놀리식(monolithic) 방식과 반대되며, 서비스 지향 아키텍처(SOA: Service-Oriented Architecture) 방식보다 더 세분화

http://www.tta.or.kr/data/weeklyNoticeView.jsp?pk_num=5193

1장. 개요

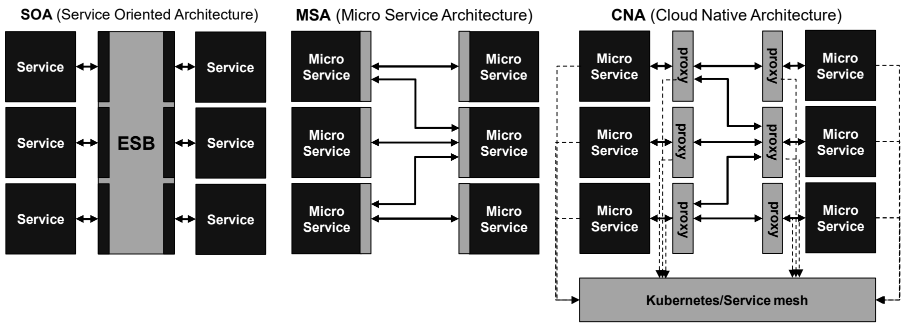
엔터프라이즈를 위한 MSA 필요

- 플랫폼 기반 서비스 제공
- 클라우드 서비스 인프라 기반 서비스
- 클라우드 서비스를 위한 SDDC 인프라 기반 추상화 계층 제공



1장. 개요

- Monolithic
- Service Oriented Architecture (SOA)
- Microservices architectures (MSA)
- Cloud Native Architecture (CSA)


SOA (Service Oriented Architecture)
Service
ESB
MSA (Micro Service Architecture)
Micro Service
CNA (Cloud Native Architecture)
proxy
Kubernetes/Service mesh

james@jslab.kr

JS Lab

9

1장. 개요

- 클라우드 네이티브 (Cloud Native)
  - 클라우드 네이티브 컴퓨팅 재단(CNCF)에서 정의
  - CNCF(Cloud Native Computing Foundation)는 오픈소스와 제조사 중립 프로젝트 생태계 육성/유지하기 위해 리눅스 재단에서 관리
  - 주요 글로벌 서비스 회사와 제조사 등이 CNCF에 참여 하는 등 220개 이상의 멤버가 활동하며 계속 증가 중
  - 자동화와 결합하여 엔지니어가 최소한의 노력으로 빈번하고 예측 가능한 변경 작업을 수행 할 수 있는 기술과 쉬운 관리 뛰어난 복원력을 제공 할 수 있음
  - 클라우드 고유 기술을 사용하며 동적 환경에서 구축하고 실행하여 서비스를 확장하기 위해 컨테이너, 서비스 메쉬, 마이크로서비스, 변경 불가능 인프라(Immutable Infrastructure) 및 선언적 API를 사용하는 접근 방식을 보여줌

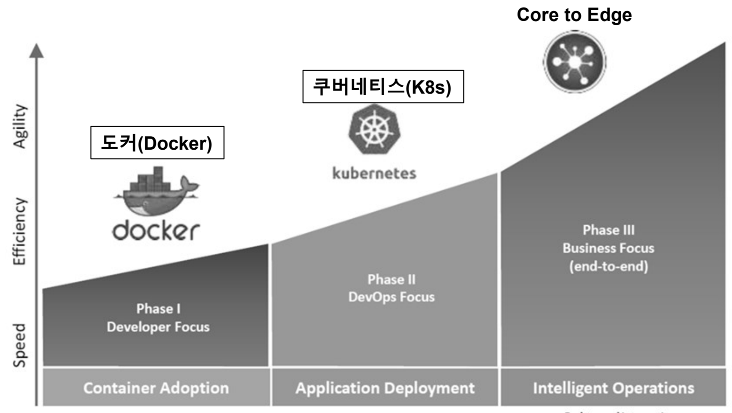
james@jslab.kr

JS Lab

10

1장. 개요

- The Cloud Native Journey



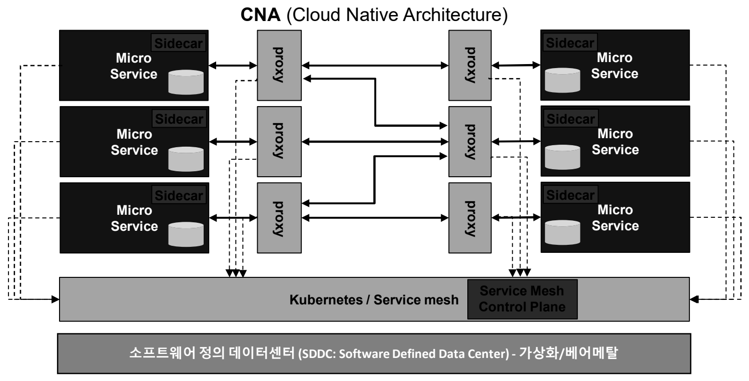
Focus	Developer adoption	DevOps deployment	End-to-end integration
Applications	Dev/Test apps	Production apps	Digital business apps
Automation	Simple orchestration	Advanced orchestration	Serverless, DevSecOps, & ML
Community	Individual developers	Teams & lines of business	Cloud native enterprises

JS Lab

11

1장. 개요

- CSA 의 서비스 메쉬 관리 체계
  - Sidecar Design Pattern: 라우터 내장 CB, LB, SD 내장
  - CNCF의 'Istio'는 정책 강화 Telemetry 제공


CNA (Cloud Native Architecture)
Sidecar
Micro Service
proxy
Kubernetes / Service mesh
Service Mesh Control Plane
소프트웨어 정의 데이터센터 (SDDC: Software Defined Data Center) - 가상화/베어메탈

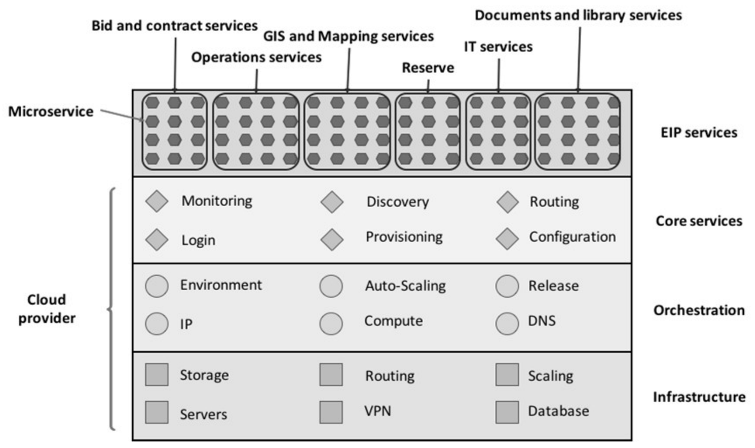
Circuit Breaker(CB), Load Balancer(LB), Service Discovery(SD)

JS Lab

12

1장. 개요

- Microservice Implementation on Cloud Infrastructure
  - IT teams shift from actively managing hardware and software to managing service layers in the microservice environment

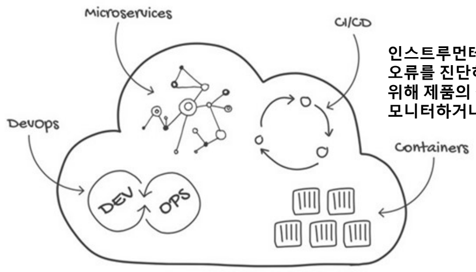


JS Lab

13

1장. 개요

- Cloud native design considerations
  - Instrumentation
  - Security
  - Parallelization
  - Resiliency
  - Event-driven
  - Future-proofed



인스트루먼테이션 (instrumentation)은 오류를 진단하거나 추적 정보를 쓰기 위해 제품의 성능 정도를 모니터하거나 측정하는 기능

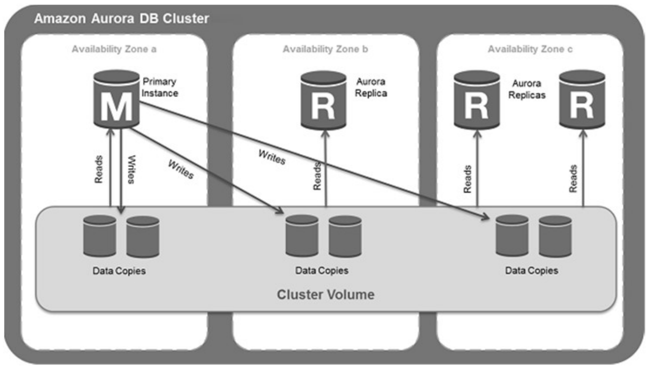
https://medium.com/walmartlabs/cloud-native-application-architecture-a84ddf378f82

JS Lab

14

1장. 개요

- **Cloud native managed services 예 (1 of 2)**
  - **AWS has a managed database service, Amazon Aurora, built on a fully distributed and self-healing storage service deigned to keep data safe and distributed across multiple availability zones.**

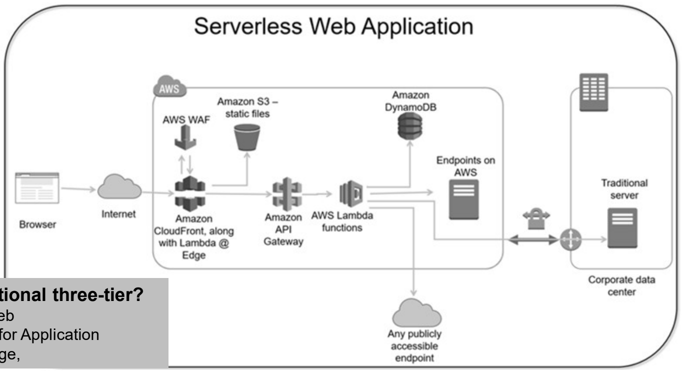


https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Overview.html

15

1장. 개요

- **Cloud native managed services 예 (2 of 2)**
  - **There are many types of serverless services: compute, API proxies, storage, databases, message processing, orchestration, analytics, and developer tools.**
  - **Licensing and usage are priced.**



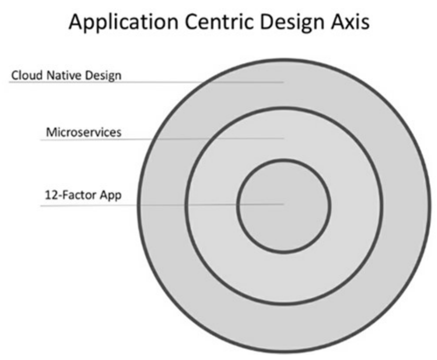
Is that a traditional three-tier?
- Front end web
- Middleware for Application
- DB for storage,

https://aws.amazon.com/ko/blogs/compute/powering-hipaa-compliant-workloads-using-aws-serverless-technologies/

16

1장. 개요

- Application Centric Design Axis
  - **12-Factor App**
  - **Microservices**
  - **Cloud Native Design**



Microservice ≤ SCS SCS (Self-contained system)

- Each SCS is an autonomous web application.
- Each SCS is owned by one team.
- Communication with other SCSs or 3rd party systems is asynchronous wherever possible.
- An SCS can have an optional service API.
- Each SCS must include data and logic.
- An SCS should make its features usable to end-users via its own UI.
- To make SCSs more robust and improve decoupling shared infrastructure can be minimized.

https://scs-architecture.org/#autonomous

JS Lab

james@jslab.kr

17

1장. 개요

- **Twelve-factor app design principles**

1. **Codebase:** One codebase tracked in revision control, many deploys
2. **Dependencies:** Explicitly declare and isolate dependencies
3. **Config:** Store config in the environment
4. **Backing services:** Treat backing services as attached resources
5. **Build, release, run:** Strictly separate build and run stages
6. **Processes:** Execute the app as one or more stateless processes
7. **Port binding:** Export services via port binding
8. **Concurrency:** Scale out via the process model
9. **Disposability:** Maximize robustness with fast startup and graceful shutdown
10. **Dev/prod parity:** Keep development, staging, and production as similar as possible
11. **Logs:** Treat logs as event streams
12. **Admin processes:** Run admin/management tasks as one-off processes

https://12factor.net/

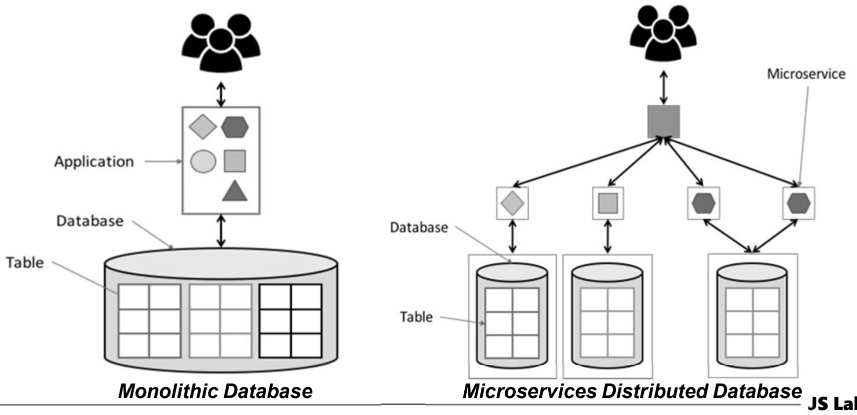
JS Lab

james@jslab.kr

18

1장. 개요

- Illustration of Monolithic Module Refactoring
  - Decentralized business and messaging rules
  - Decentralized governance
  - Decentralized data management



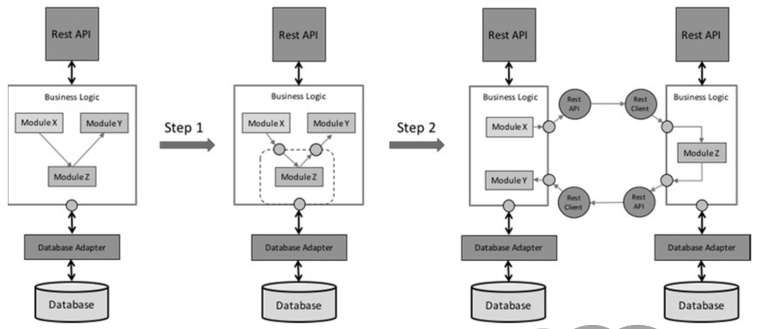
james@jslab.kr

JS Lab

19

1장. 개요

- Illustration of Monolithic Module Refactoring



Refactoring

> 외부동작을 바꾸지 않으면서 내부 구조를 개선하는 방법으로, 소프트웨어 시스템을 변경하는 프로세스
> 소프트웨어의 기능은 바꾸지 않음

james@jslab.kr

https://haloworld.tistory.com/24 [Halo World:)]

JS Lab

20

1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ **실습교재 (별도)**

james@jslab.kr

JS Lab

21

2장. 가상화 (Virtualization)

- **하이퍼바이저**
- **가상화와 클라우드 서비스**
- **통신 시장의 NFV 환경**

james@jslab.kr

JS Lab

22

2장. 가상화 (Virtualization)

- "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

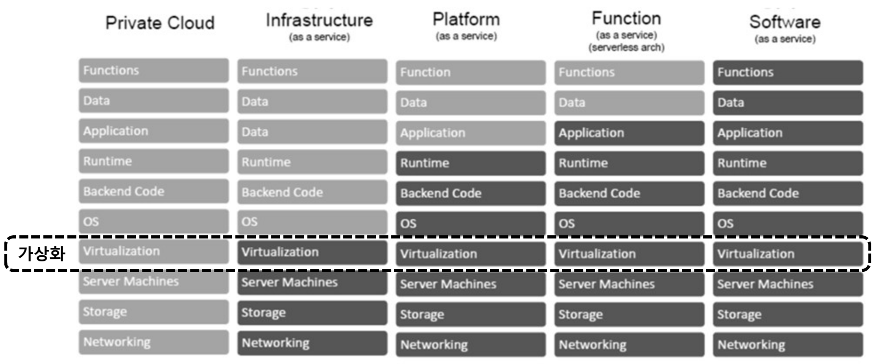
james@jslab.kr

JS Lab

23

2장. 가상화 (Virtualization)

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)



	Private Cloud	Infrastructure (as a service)	Platform (as a service)	Function (as a service) (serverless arch)	Software (as a service)
	Functions	Functions	Function	Functions	Functions
	Data	Data	Data	Data	Data
	Application	Data	Application	Application	Application
	Runtime	Runtime	Runtime	Runtime	Runtime
	Backend Code	Backend Code	Backend Code	Backend Code	Backend Code
	OS	OS	OS	OS	OS
가상화	Virtualization	Virtualization	Virtualization	Virtualization	Virtualization
	Server Machines	Server Machines	Server Machines	Server Machines	Server Machines
	Storage	Storage	Storage	Storage	Storage
	Networking	Networking	Networking	Networking	Networking

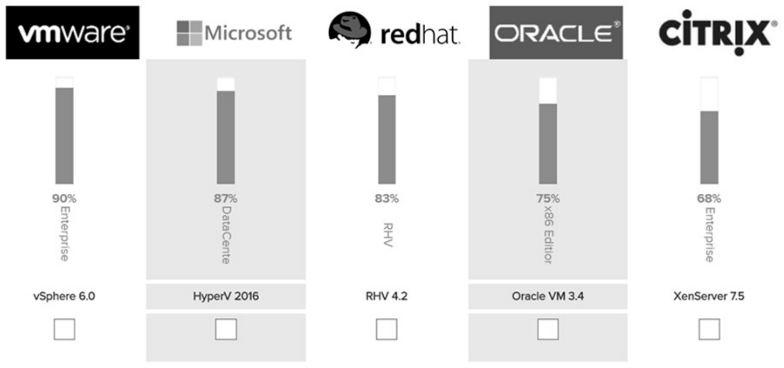
james@jslab.kr

JS Lab

24

2장. 가상화 (Virtualization)

- 기업용 Private Cloud를 위한 가상화
- VMware, Microsoft, redhat, Oracle, Citrix
- 필요 기능에 따라 제조사 평가 결과가 다를 수 있음



https://www.whatmatrix.com/comparison/Virtualization

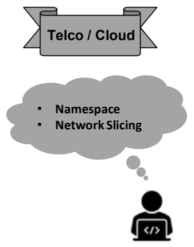
JS Lab

james@jslab.kr

25

2장. 가상화 (Virtualization)

- Key Features of Cloud Computing
  - Speed and Agility
  - Cost
  - Easy Access to Resources
  - Maintenance
  - Multi-tenancy
  - Reliability



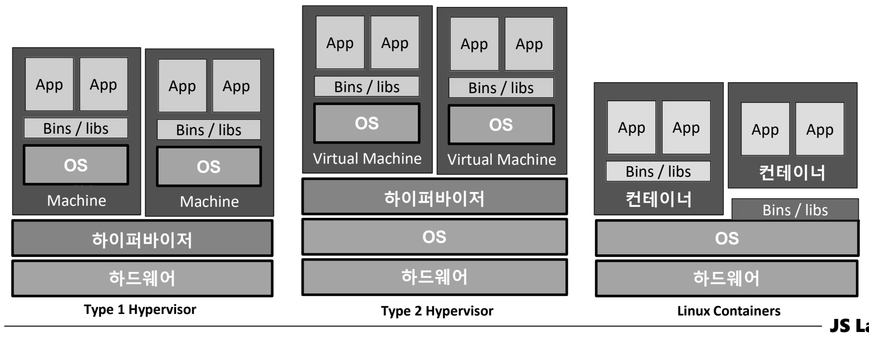
JS Lab

james@jslab.kr

26

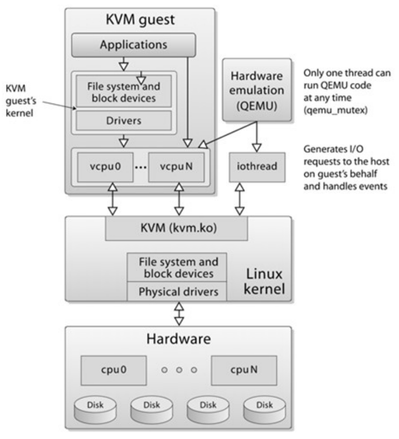
2장. 가상화 (Virtualization)

- KVM
- Xen
- VMware
- VirtualBox
- Hyper-V



2장. 가상화 (Virtualization)

- KVM



KVM guest
Applications
KVM guest's kernel
File system and block devices
Drivers
vcpu0
vcpuN
Hardware emulation (QEMU)
Only one thread can run QEMU code at any time (qemu_mutex)
iothread
Generates I/O requests to the host on guest's behalf and handles events
KVM (kvm.ko)
File system and block devices
Physical drivers
Linux kernel
Hardware
cpu0
cpuN
Disk

A High-Level Overview of the KVM/QEMU Virtualization Environment
(by V4711, licensed under CC BY-SA 4.0, retrieved from Wikipedia)

2장. 가상화 (Virtualization)

- **VirtualBox**
  - **VirtualBox is an x86 and AMD64/Intel64 virtualization product from Oracle, which runs on Windows, Linux, Macintosh, and Solaris hosts and supports guest OSes from Windows, Linux families, and others, like Solaris, FreeBSD, DOS, etc.**
  - **It is an easy-to-use multi-platform hypervisor. It is not part of the mainline kernel. So, to use it on Linux, we have to compile and insert the respective kernel module.**
  - **VirtualBox is distributed under the GNU General Public License (GPL) version 2.**

2장. 가상화 (Virtualization)

- **Benefits of Using VirtualBox**
  - **It is an open source solution.**
  - **It is free to use.**
  - **It runs on Linux, Windows, OS X, and Solaris.**
  - **It provides two virtualization choices: software-based virtualization and hardware-assisted virtualization.**
  - **It is an easy-to-use multi-platform hypervisor.**
  - **It provides the ability to run virtualized applications side-by-side with normal desktop applications.**
  - **It provides teleportation - live migration.**

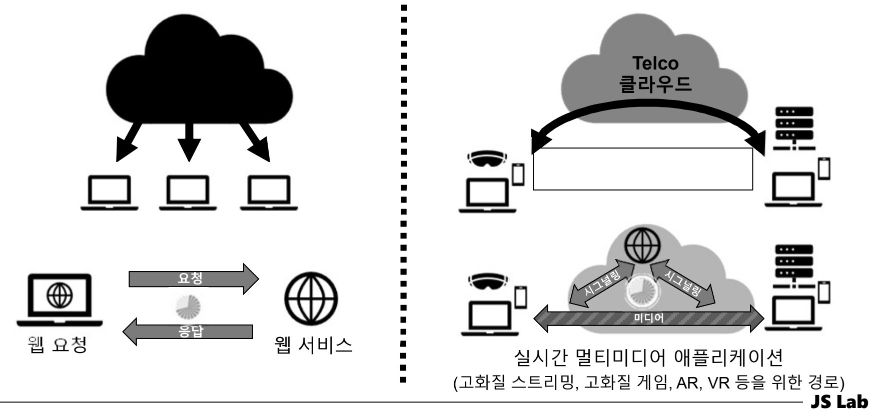
2장. 가상화 (Virtualization)

- **Vagrant**
  - **Reproducible environment**
  - **Management of multiple projects in their restricted environment**
  - **Sharing the environment with other teammates**
  - **Keeping the development and deployment environments in sync**
  - **Running the same VM on different OSes, with a hypervisor like VirtualBox.**

```
$ vagrant init hashicorp/precise64
$ vagrant up
```

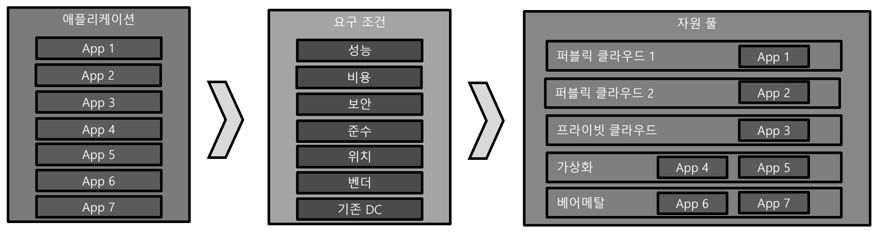
2장. 가상화 (Virtualization)

- **5G 코어 인프라 기반 NFV의 클라우드화**
  - **Public Cloud: 소프트웨어 정의 가상 인프라 기반 서비스 (웹서비스)**
  - **Telco Cloud: 언더레이 인프라 기반 클라우드 서비스 (전송경로 제공)**
  - **Telco Cloud는 하드웨어 인프라 환경 고려**

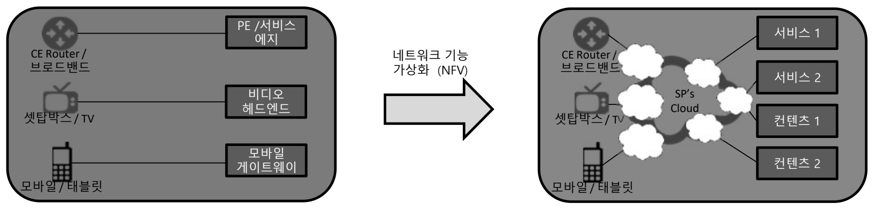

요청
응답
웹 요청
웹 서비스
Telco
클라우드
시그널링
시그널링
미디어
실시간 멀티미디어 애플리케이션
(고화질 스트리밍, 고화질 게임, AR, VR 등을 위한 경로)

2장. 가상화 (Virtualization)

- 엔터프라이즈는 애플리케이션 요구에 적합한 클라우드 자원



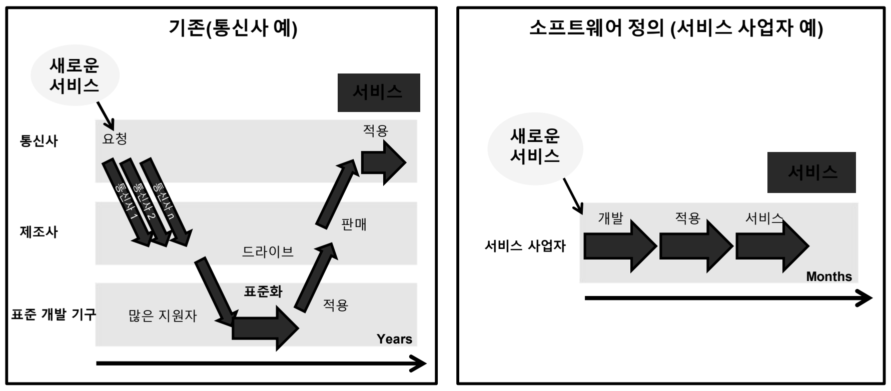
- 텔코 (Telco)는 네트워크 기능 가상화 기반 데이터센터화



33

2장. 가상화 (Virtualization)

- 통신사와 서비스 사업자의 적용 환경
- 개발능력 미보유 엔터프라이즈는 서비스 제공 가능 오픈소스나 제조사의 SDx 솔루션 필요



34

2장. 가상화 (Virtualization)

◻ Edge vs Core @ Telco

	Edge (에지) 클라우드	중앙 클라우드
App의 위치	노드의 물리적 위치에서 중요한 서비스	비교적 위치와 독립적인 서비스
워크로드의 이동성	워크로드가 노드간 이동	클라우드 노드 장애 이외에는 비교적 고정
워크로드의 역동성	다양한 App들이 다양한 시간에 크게 다른 요구를 함	서비스를 적용하면 대부분의 시간에 안정적인 워크로드
아키텍쳐	다른 형태의 많은 수의 노드와 다양한 용량과 기술	대부분 동일하며 차이가 작음 (예: AWS, OpenStack, Azure 등)
지연	지연과 거리는 종단 사용자들을 위한 주요 역할	대부분 지연에 민감하지 않음
자원 가용성	에지노드는 작고, App을 위한 자원의 가용성을 보장하지 않음	가용성 확보는 중요하며 주요 기능 중 1개

35

1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ **실습교재 (별도)**

36

3장. 클라우드 서비스 (Cloud Services)

- IaaS
- PaaS
- Hybrid / Multi-Cloud

37

3장. 클라우드 서비스 (Cloud Services)

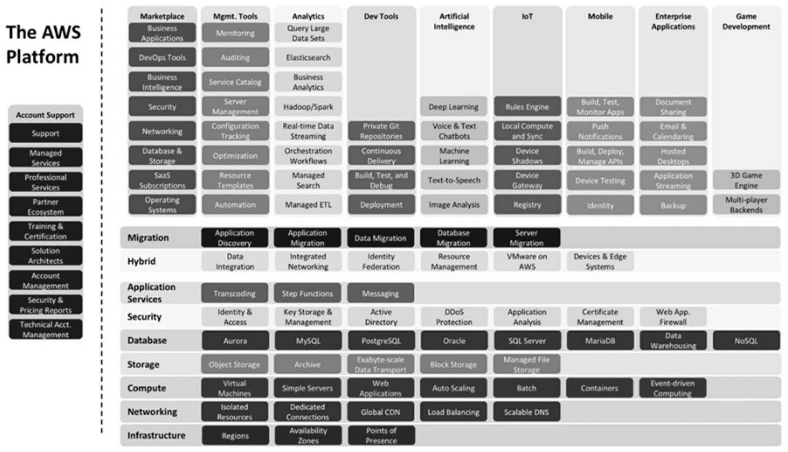
- **Cloud vendor managed service offerings**
  - **Databases**
  - **Hadoop**
  - **Directory services**
  - **Load balancers**
  - **Caching systems**
  - **Data warehouses**
  - **Code repositories**
  - **Automation tools**
  - **Elastic searching tools**

38

3장. 클라우드 서비스 (Cloud Services)

□ Cloud Vendor’s Services

■ **Amazon Web Service(AWS) 클라우드 플랫폼(예)**



39

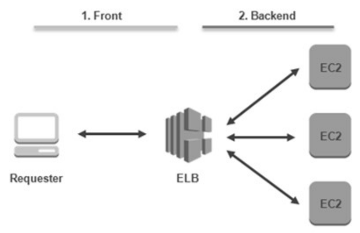
https://cloud.google.com/docs/compare/aws/?utm_source=google&utm_medium=cpc&utm_campaign=japac-KR-all-en-dr-bkws-all-all-trial-e-dr-1003987&utm_content=text-ad-none-none-DEV_c-CRE_343065165075-ADGP_Hybrid+%7C+AW+SEM+%7C+BKWS+~+T3+%7C+EXA+%7C+Multi+Cloud+%7C+M:1+%7C+KR+%7C+en+%7C+anthos-KWID_43700042873424630-kwd-505549821792&userloc_1009871&utm_term=KW_anthos&gclid=Cj0KCQjwuNbsBRC-ARIsAAzITucqoGQ2flQHbO8ctoG312oVY_NCTGpwbLbOmHfJUX88e-ng7zjTXoEaAqpAEALw_wcB

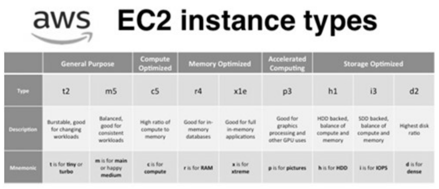
서비스 카테고리	서비스	AWS	Google Cloud Platform
컴퓨팅	IaaS	Amazon Elastic Compute Cloud	Compute Engine
	PaaS	AWS Elastic Beanstalk	App Engine
	컨테이너	Amazon Elastic 컨테이너 서비스	Google Kubernetes Engine
	서버리스 함수	AWS Lambda	Cloud Functions
	관리형 일괄 컴퓨팅	AWS Batch	해당 없음
네트워크	가상 네트워크	Amazon Virtual Private Cloud	가상 사설 클라우드
	부하 분산기	Elastic Load Balancer	Cloud Load Balancing
	전용 상호 연결	Direct Connect	Cloud Interconnect
	도메인 및 DNS	Amazon Route 53	Google Domains, Cloud DNS
	CDN	Amazon CloudFront	Cloud CDN
저장소	개체 저장소	Amazon Simple Storage Service	Cloud Storage
	블록 저장소	Amazon Elastic Block Store	Persistent Disk
	제한 가용성 저장소	Amazon S3 Standard-Infrequent Access, Amazon S3 One Zone-Infrequent Access	Cloud Storage Nearline
	아카이브 저장소	Amazon Glacier	Cloud Storage Coldline
	파일 저장소	Amazon Elastic File System	Cloud Filestore(베타)
데이터베이스	RDBMS	Amazon Relational Database Service, Amazon Aurora	Cloud SQL, Cloud Spanner
	NoSQL: 키-값	Amazon DynamoDB	Cloud Firestore, Cloud Bigtable
	NoSQL: 색인 생성	Amazon SimpleDB	Cloud Firestore
빅데이터 및 분석	일괄 데이터 처리	Amazon Elastic MapReduce, AWS Batch	Cloud Dataproc, Cloud Dataflow
	스트림 데이터 처리	Amazon Kinesis	Cloud Dataflow
	스트림 내부 데이터화	Amazon Kinesis	Cloud Pub/Sub
	분석	Amazon Redshift, Amazon Athena	BigQuery
	워크플로 조정	Amazon Data Pipeline, AWS Glue	Cloud Composer
애플리케이션 서비스	메시지	Amazon Simple Notification Service, Amazon Simple Queueing Service	Cloud Pub/Sub
관리 서비스	모니터링	Amazon CloudWatch	Stackdriver Monitoring
	로깅	Amazon CloudWatch Logs	Stackdriver Logging
	배포	AWS CloudFormation	Cloud Deployment Manager
머신러닝	음성	Amazon Transcribe	Cloud Speech-to-Text
	Vision	Amazon Rekognition	Cloud AutoML Vision
	자연 언어 처리	Amazon Comprehend	Cloud Natural Language
	번역	Amazon Translate	Cloud Translation
	대화 인터페이스	Amazon Lex	Dialogflow Enterprise 버전
	Video Intelligence	Amazon Rekognition Video	Cloud Video Intelligence
	자동 생성 모델	해당 없음	Cloud AutoML(베타)
	완전 관리형 ML	Amazon SageMaker	AI Platform

40

3장. 클라우드 서비스 (Cloud Services)

- Amazon EC2 has many features, allowing you to:
  - Create an Elastic IP for remapping the Static IP address automatically
  - Provision a Virtual Private Cloud for isolation. Amazon Virtual Private Cloud provides secure and robust networking for Amazon EC2 instances
  - Use CloudWatch for monitoring resources and applications
  - Use Auto Scaling to dynamically resize your resources, etc.





aws EC2 instance types

	General Purpose		Compute Optimized	Memory Optimized		Accelerated Computing	Storage Optimized		
Type	t2	m5	c5	r4	x1e	p3	h1	i3	d2
Description	Burstable, good for changing workloads	Balanced, good for consistent workloads	High ratio of compute to memory	Good for in-memory databases	Good for full in-memory applications	Good for graphics processing and other GPU uses	HDD backed, balance of compute and memory	SSD backed, balance of compute and memory	Highest disk ratio
Mnemonic	**t** is for **tiny** or **turbo**	**m** is for **main** or happy **medium**	**c** is for **compute**	**r** is for **RAM**	**x** is for **xtreme**	**p** is for **pictures**	**h** is for **HDD**	**i** is for **IOPS**	**d** is for **dense**

JS Lab

41

3장. 클라우드 서비스 (Cloud Services)

- Benefits of Using Amazon EC2:
  - It is an easy-to-use IaaS solution.
  - It is flexible and scalable.
  - It provides a secure and robust functionality for your compute resources.
  - It enables automation.
  - It is cost-effective: you only pay for the time and resources you use.
  - It is designed to work in conjunction with other AWS components.
  - It promises 99.99% uptime.
  - It provides specialized instances for workloads, such as floating point operations, high graphics capability, high input/output (I/O), High Performance Computing (HPC), etc.

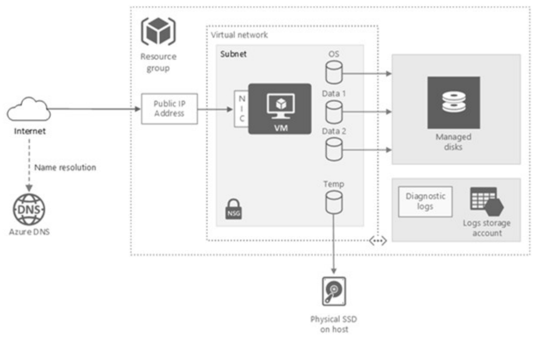
JS Lab

42

3장. 클라우드 서비스 (Cloud Services)

□ **Azure Virtual Machine**

- **We can manage Virtual Machines from Azure's web interface.**
- **Azure also provides a command line utility to manage resources and applications on the Azure cloud.**



43

3장. 클라우드 서비스 (Cloud Services)

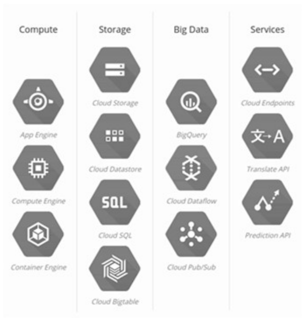
□ **The benefits of using Azure virtual machine are:**

- **It is an easy-to-use IaaS solution.**
- **It is flexible and scalable.**
- **It provides a secure and robust functionality for your compute resources.**
- **It enables automation.**
- **It is cost-effective: you only pay for the time and resources you use.**
- **It is designed to work in conjunction with other Azure services.**

44

3장. 클라우드 서비스 (Cloud Services)

- Google Compute Engine
  - Google Cloud Platform is Google's Cloud offering, which has many products in different domains, like compute, storage, networking, big data, and others. Google Compute Engine provides the compute service. We can manage the instances through GUI, APIs or command line. Access to the individual VM's console is also available



3장. 클라우드 서비스 (Cloud Services)

- GCE supports different machine types, which we can choose from depending on our need. They are categorized in the following types:
  - Standard machine types
  - High-CPU machine types
  - High-memory machine types
  - Shared-core machine types
  - We can also configure custom machine types.

3장. 클라우드 서비스 (Cloud Services)

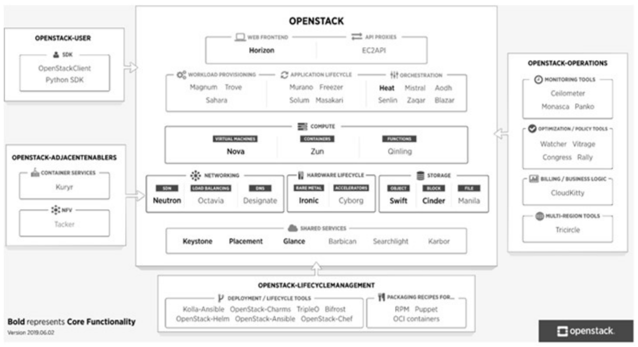
- Benefits of Using Google Compute Engine:
  - It is flexible and allows you to scale your applications easily.
  - Fast boot time.
  - It is very secure, encrypting all data stored.
  - It enables automation.
  - It is cost-effective: you only pay for the time and resources you use.
  - It supports custom machine types.
  - It supports Virtual Private Cloud, Load Balancers, etc.

3장. 클라우드 서비스 (Cloud Services)

- Introduction to OpenStack
- With OpenStack, we can offer a cloud computing platform for public and private clouds. OpenStack was started as a joint project between Rackspace and NASA in 2010. In 2012, a non-profit corporate entity, called the OpenStack Foundation, was formed and it is managing it since then. It is now supported by more than 500 organizations. OpenStack is an open source software platform, which is released under an Apache 2.0 License.

3장. 클라우드 서비스 (Cloud Services)

- OpenStack
- Component/Features
  - Keystone
  - Nova
  - Horizon
  - Neutron
  - Glance
  - Swift
  - Cinder
  - Heat
  - Ceilometer



https://www.openstack.org/software/

JS Lab

james@jslab.kr

49

3장. 클라우드 서비스 (Cloud Services)

- Benefits of Using OpenStack
  - It is an open source solution.
  - It is a cloud computing platform for public and private clouds.
  - It offers a flexible, customizable, vendor-neutral environment.
  - It provides a high level of security.
  - It facilitates automation throughout the stages of the cloud lifecycle.
  - By reducing system management overhead and avoiding vendor lock-in, it can be cost-effective.

JS Lab

james@jslab.kr

50

3장. 클라우드 서비스 (Cloud Services)

◻ Platform as a Service (PaaS)

- **A class of cloud computing services which allows its users to develop, run, and manage applications without worrying about the underlying infrastructure. With PaaS, users can simply focus on building their applications, which is a great help to developers.**
- **We can either use PaaS services offered by different cloud computing providers like Amazon, Google, Azure, etc., or deploy it on-premise, using software like OpenShift Origin.**
- **PaaS can be deployed on top of IaaS, or, independently on VMs, bare metal, and containers.**
- **In this chapter, we will take a closer look at some of the PaaS providers and their features. We will also provide a demo video for each one of them.**

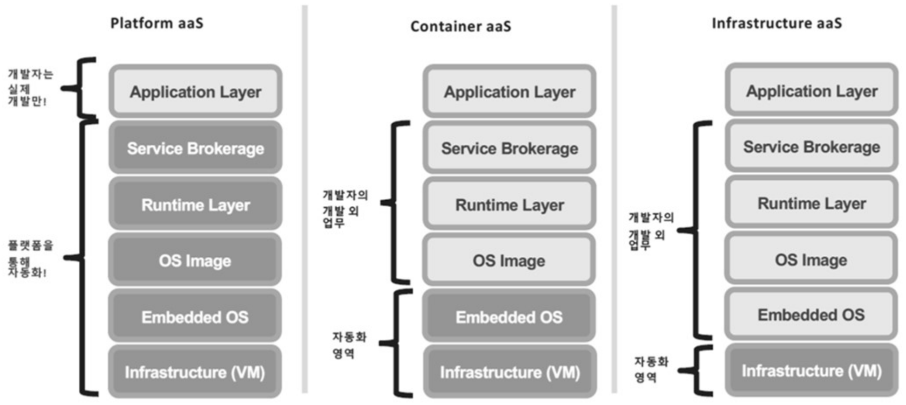
JS Lab

51

3장. 클라우드 서비스 (Cloud Services)

◻ Cloud Foundry

- **PCF(Pivotal Cloud Foundry)**

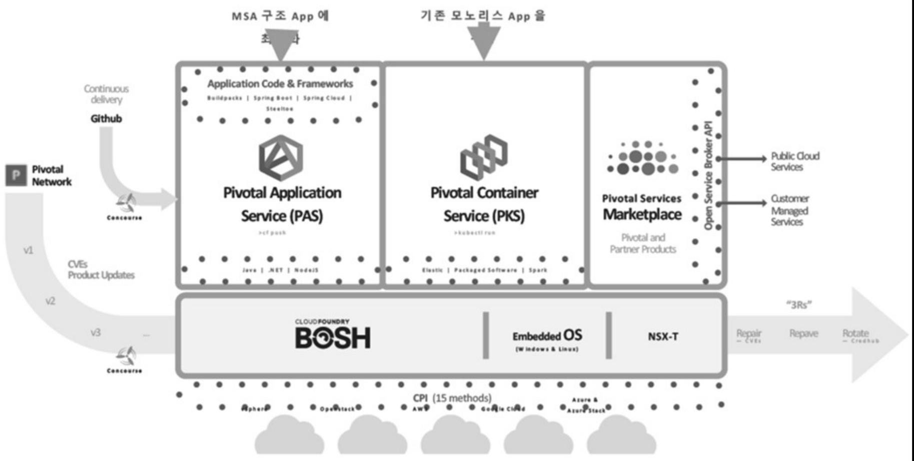


JS Lab

52

3장. 클라우드 서비스 (Cloud Services)

- Cloud Foundry
  - MSA 개발/운영클라우드플랫폼모델



53

3장. 클라우드 서비스 (Cloud Services)

- 멀티 클라우드 Federation w/K8s
  - High Availability
  - Application Migration
  - Policy Enforcement
  - Vendor Lock-in Avoidance
  - Capacity Overflow

Cloud Native Applications

Tools Chain
CI/CD, Provisioning, Automation, Registry, Telemetry, Policy, IAM, Security

Service Mesh

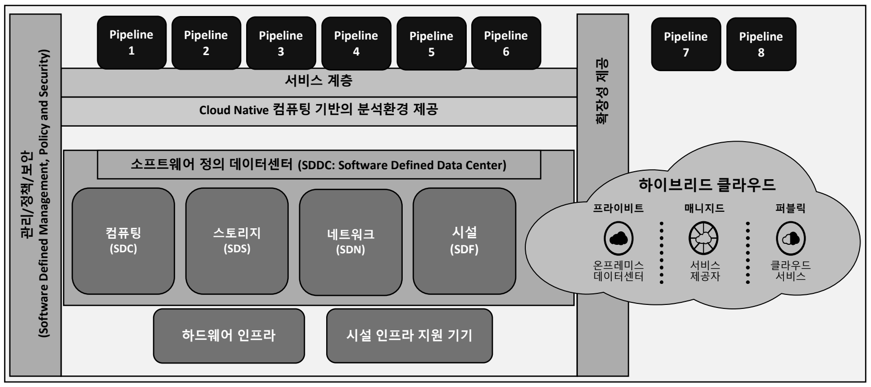
kubernetes
kubernetes
kubernetes

54

3장. 클라우드 서비스 (Cloud Services)

□ **SDDC 기반 인프라**

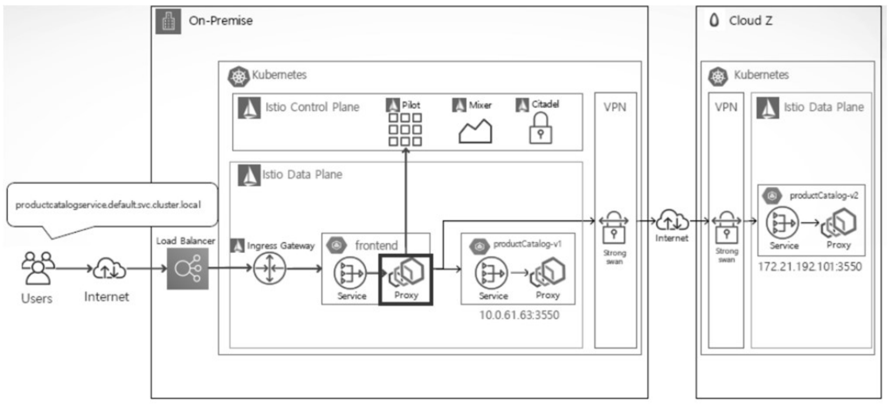
- **Cloud Native등의 기술 발전 수용 SDDC 데이터센터 구축 체계**
- **데이터센터 자원의 추상화 관리 (SDDC)**



3장. 클라우드 서비스 (Cloud Services)

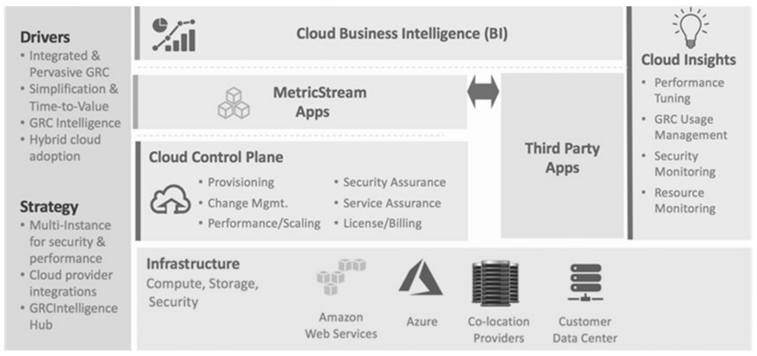
□ **Multi-Cloud Service Mesh Architecture (예)**

- **Multi-Cloud(Cross Cluster)간 Service Mesh 연결/확장**



3장. 클라우드 서비스 (Cloud Services)

- SDDC 기반 인프라 from Public Cloud
  - IBM Cloud Pak
  - AWS Outpost
  - MS Azure Stack
  - Google Anthos



https://www.metricstream.com/technology/grc-cloud.htm

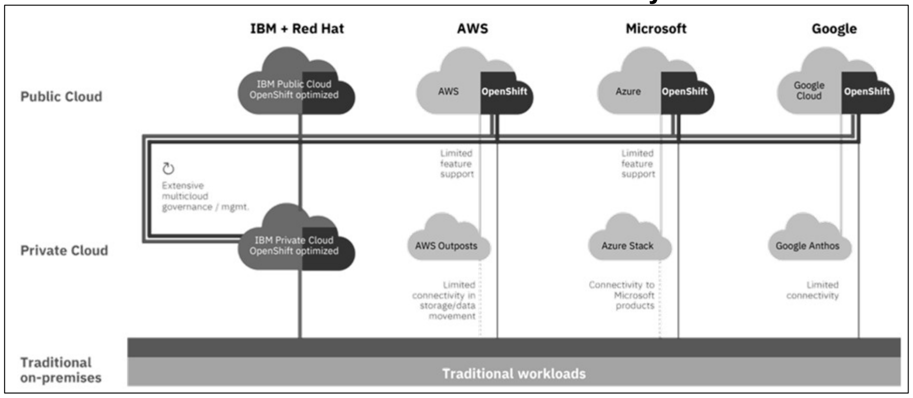
JS Lab

james@jslab.kr

57

3장. 클라우드 서비스 (Cloud Services)

- IBM Cloud Pak
  - Hybrid, Multicloud platform
  - Cloud Pak on OpenShift
  - Cloud native for all clouds - middleware anywhere



https://www.metricstream.com/technology/grc-cloud.htm

JS Lab

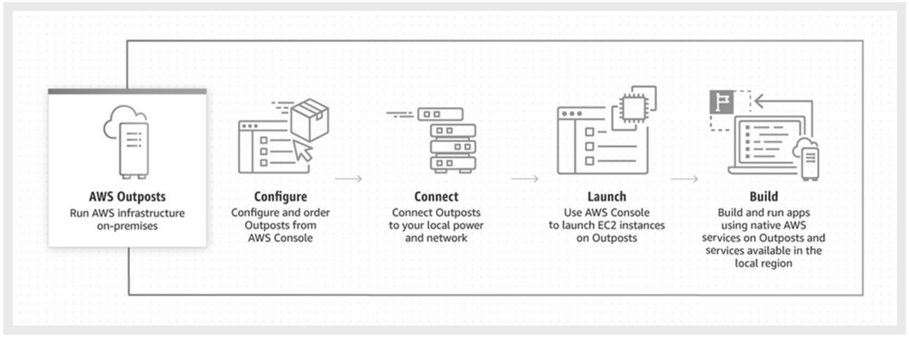
james@jslab.kr

58

3장. 클라우드 서비스 (Cloud Services)

□ **AWS Outpost**

- **AWS 온프레미스 확장**
- **관리 플레인 선택: AWS 네이티브 변형 or VMware Cloud on AWS**
- **완전관리형**


AWS Outposts
Run AWS infrastructure on-premises
Configure
Configure and order Outposts from AWS Console
Connect
Connect Outposts to your local power and network
Launch
Use AWS Console to launch EC2 instances on Outposts
Build
Build and run apps using native AWS services on Outposts and services available in the local region

https://aws.amazon.com/ko/outposts/

JS Lab

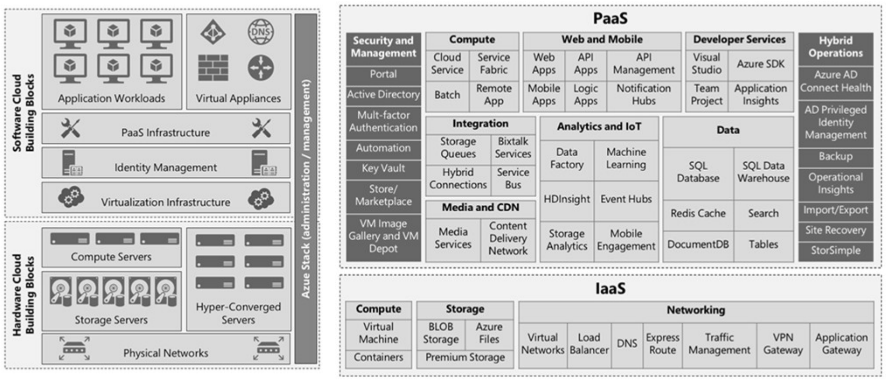
james@jslab.kr

59

3장. 클라우드 서비스 (Cloud Services)

□ **MS Azure Stack**

- **공용 클라우드 서비스 사용**
- **온-프레미스에서 클라우드 서비스 사용**
- **온-프레미스에서 가상화된 애플리케이션 실행**



https://azure.microsoft.com/ko-kr/overview/azure-stack/ https://argonsys.com/solutions/cloud-building-blocks/

JS Lab

james@jslab.kr

60

3장. 클라우드 서비스 (Cloud Services)

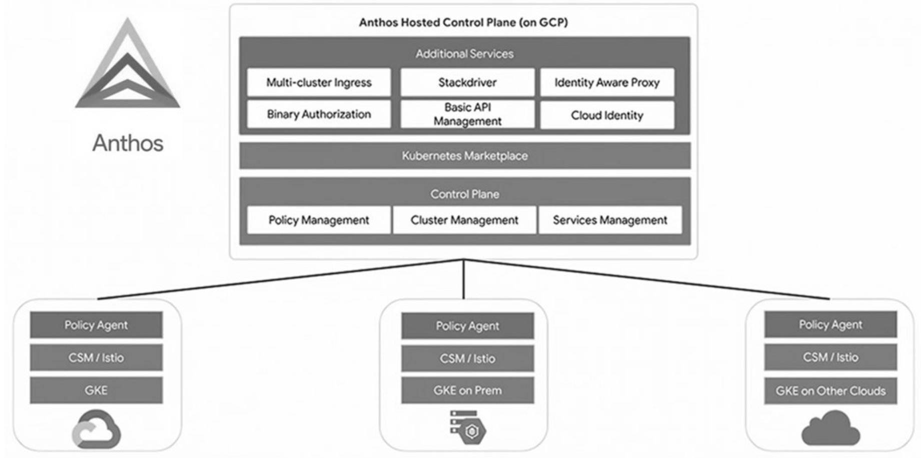
- MS Azure Stack
  - Hybrid Cloud as a Service

JS Lab

61

3장. 클라우드 서비스 (Cloud Services)

- Google Anthos



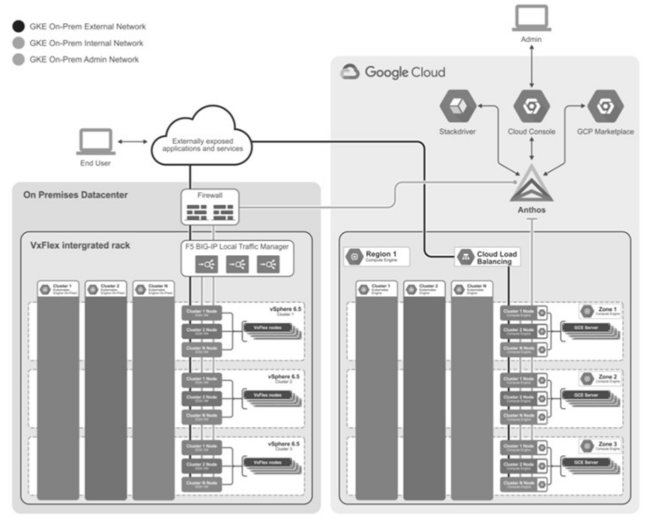
https://www.bespinglobal.com/google-anthos-multi-cloud/ https://blog.aquasec.com/hybrid-cloud-security-google-anthos

JS Lab

62

3장. 클라우드 서비스 (Cloud Services)

□ **VxFlex integrated rack for Google Cloud's Anthos**

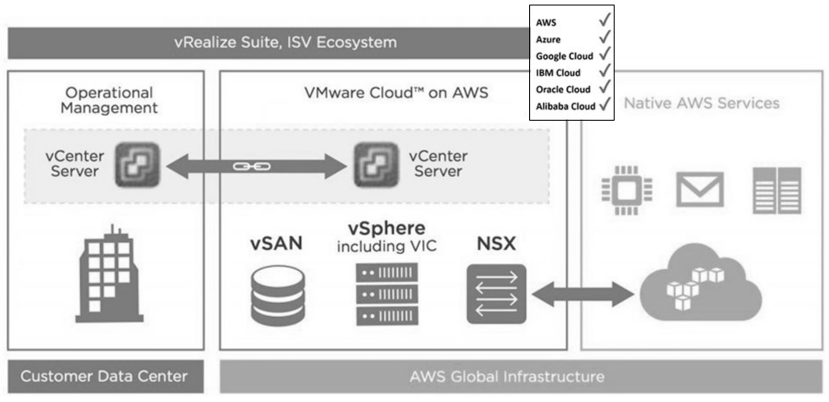


63

3장. 클라우드 서비스 (Cloud Services)

□ **VMware Cloud**

- **VMware Cloud on AWS**
- **지원 확장: Azure, Google, IBM, Oracle, Alibaba and KT Cloud**

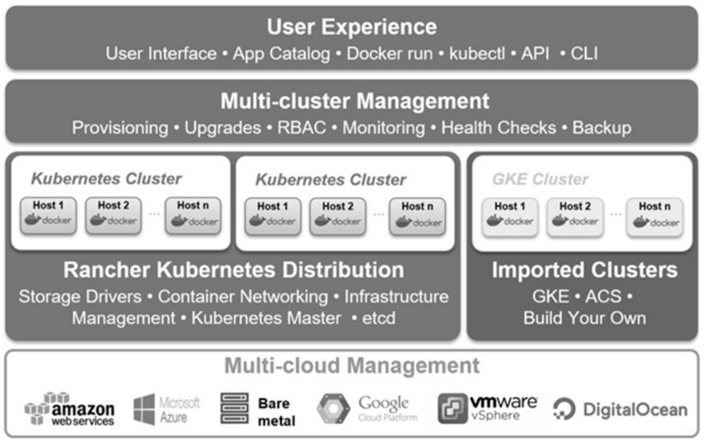


https://www.actualtech.io/getting-hands-dirty-installing-vmware-cloud-aws/

64

3장. 클라우드 서비스 (Cloud Services)

- Rancher Labs
  - Rancher 2.x


User Experience
User Interface • App Catalog • Docker run • kubectl • API • CLI
Multi-cluster Management
Provisioning • Upgrades • RBAC • Monitoring • Health Checks • Backup
Kubernetes Cluster
Host 1
Host 2
Host n
docker
GKE Cluster
Rancher Kubernetes Distribution
Storage Drivers • Container Networking • Infrastructure Management • Kubernetes Master • etcd
Imported Clusters
GKE • ACS • Build Your Own
Multi-cloud Management
amazon webservices
Microsoft Azure
Bare metal
Google Cloud Platform
vmware vSphere
DigitalOcean

https://rancher.com/announcing-rancher-2-0/

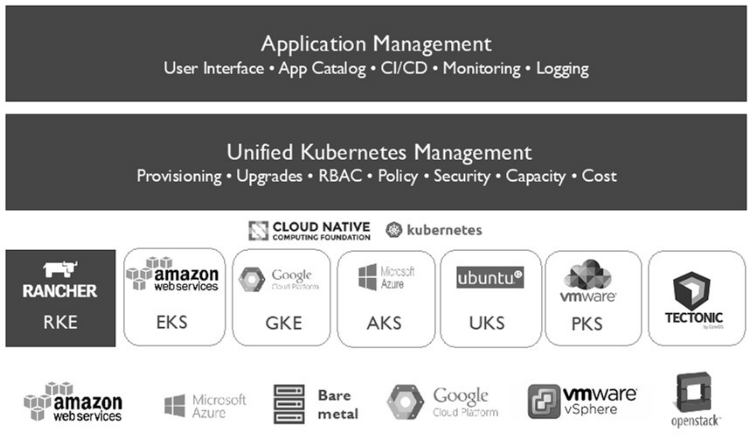
JS Lab

james@jslab.kr

65

3장. 클라우드 서비스 (Cloud Services)

- Rancher Labs
  - Rancher 2.x



https://rancher.com/announcing-rancher-2-0/

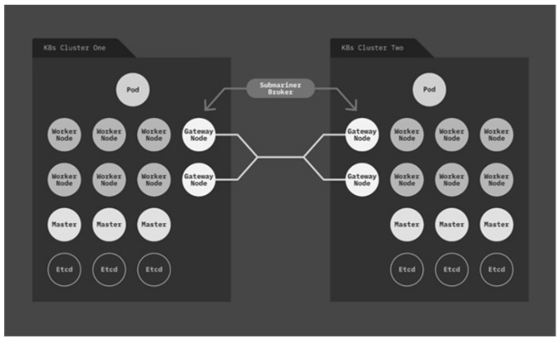
JS Lab

james@jslab.kr

66

3장. 클라우드 서비스 (Cloud Services)

- K8s Cluster간 네트워크 연결 요구
- Open source project 'Submariner' (예)
  - 카산드라(Cassandra)와 같은 HA 데이터베이스의 지역 분산
  - 분산화 트레이스 (Distributed Tracing)
  - Service Mesh의 클러스터들 간에 확대



https://submariner.io/

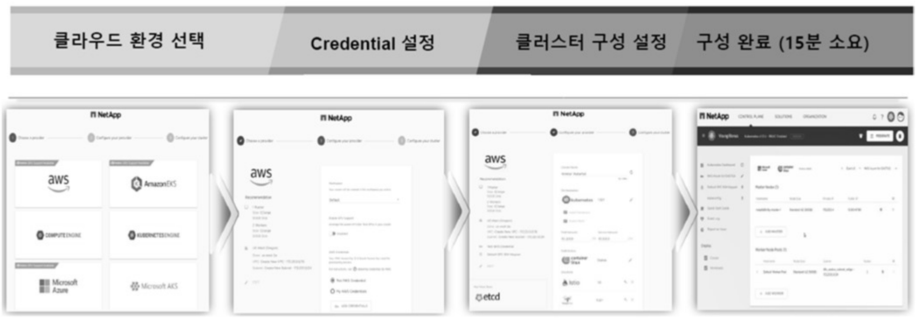
JS Lab

james@jslab.kr

67

3장. 클라우드 서비스 (Cloud Services)

- NKS (NetApp Kubernetes Service)
  - 자동화 기반의 배포 및 관리



JS Lab

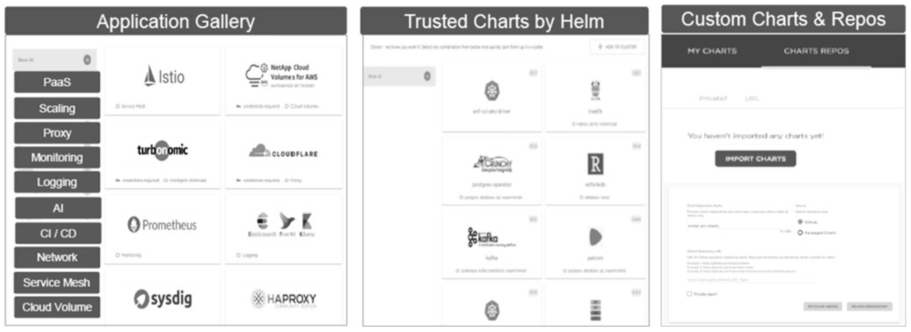
james@jslab.kr

68

3장. 클라우드 서비스 (Cloud Services)

□ NKS (NetApp Kubernetes Service)

- 관리 및 배포를 위한 Cloud Native Solutions Marketplace



james@jslab.kr

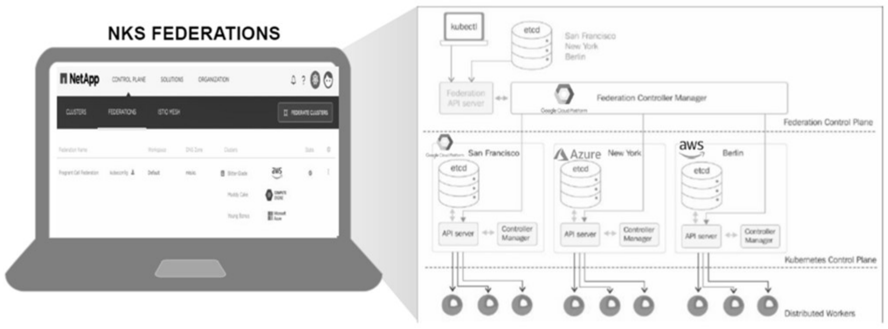
JS Lab

69

3장. 클라우드 서비스 (Cloud Services)

□ NKS (NetApp Kubernetes Service)

- Multi Kubernetes Cluster Multi Federation



james@jslab.kr

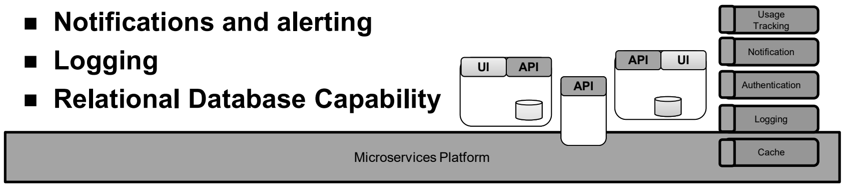
JS Lab

70

james@jslab.kr

3장. 클라우드 서비스 (Cloud Services)

- **Microservices cross platform Capabilities**
  - **Authentication**
  - **Persistent Storage**
  - **Cache**
  - **Load balancing/API Gateway**
  - **Discovery/Lookup**
  - **Monitoring**
    - ✓ Functional/Business KPIs
    - ✓ Non Functional Platform/Container & Infra
  - **Audit, Usage tracking, Billing**
  - **Notifications and alerting**
  - **Logging**
  - **Relational Database Capability**



JS Lab

71

james@jslab.kr

1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ **실습교재 (별도)**

JS Lab

72

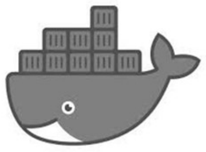
4장. 컨테이너 (Containers)

1. 개요
2. Micro OS
3. Orchestration (오케스트레이션)
4. Unikernels
5. Microservices (마이크로서비스)
6. SDN(소프트웨어 정의 네트워크)과 컨테이너
7. SDS(소프트웨어 정의 스토리지)와 컨테이너

4장. 컨테이너 1. 개요

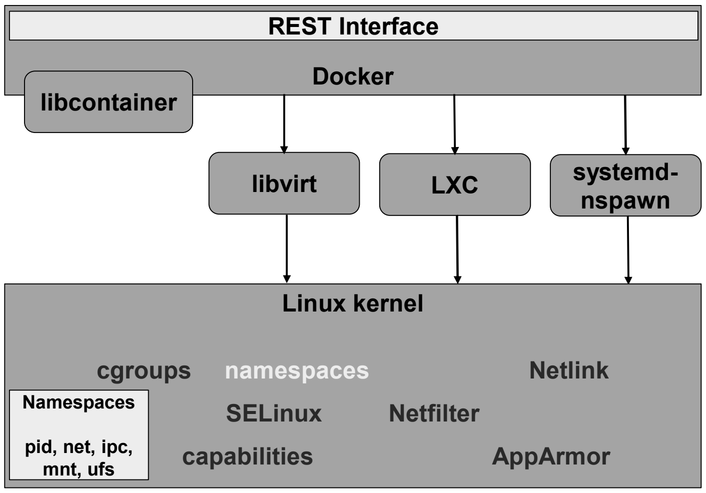
- 용어 정의
- 도커(Docker) 란?

“Company”
“Product”
“Platform”
“CLI Tool”
“Computer Program”



4장. 컨테이너 1. 개요

- How it works?



75

4장. 컨테이너 1. 개요

- **컨테이너는 리눅스 커널의 3가지 요소 기반하며, 리눅스 컨테이너(LXC)는 2008년에 chroot, cgroup, namespace를 조합하여 도입**

  Chroot: jail로 알려져 있으며 1979년 유닉스부터 사용을 했으며 실행중인 프로세스가 사용하는 루트 디렉토리를 외부 디렉토리 트리의 children 에서 접속 할 수 없음

  Namespace: 2002년부터 리눅스 커널에서 사용을 했으며 애플리케이션이 운영 환경에서 네트워킹, 사용자 ID, 파일시스템, 프로세스 트리 등을 완전하게 독립하는 것을 허용

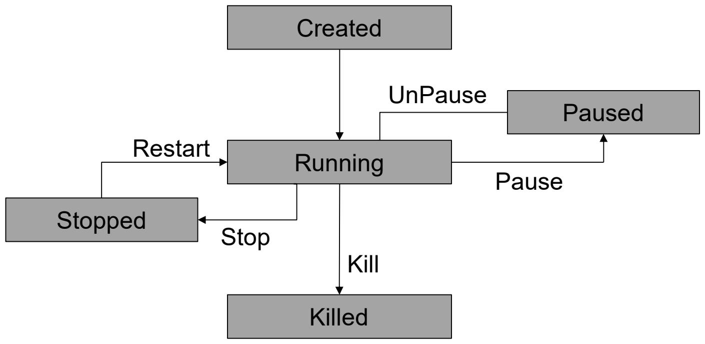
  Cgroup: 2008년부터 리눅스 커널에서 사용 했으며 CPU, 메모리, 디스크 I/O,네트워크 등의 자원을 분리하고 제한함

- **도커(Docker)는 컨테이너 생태계를 대중화 했으며 소프트웨어를 계층화 하는 Union File System(UFS)를 도입하고, 컨테이너 내의 애플리케이션 적용을 자동화**

76

4장. 컨테이너 1. 개요

- **도커 컨테이너 라이프 사이클:** Build한 이미지(Image)는 컨테이너 실행 시 읽기 전용으로 사용하며 컨테이너에서 생성한 계층에서 쓰기와 읽기를 실행



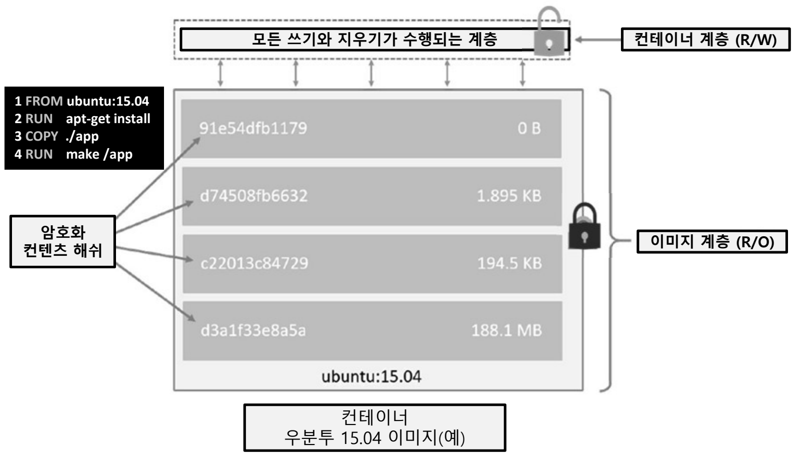
james@jslab.kr

JS Lab

77

4장. 컨테이너 1. 개요

- **이미지 생성: Build한 이미지(Image)는 컨테이너 실행 시 읽기 전용으로 사용하며 컨테이너에서 생성한 계층에서 쓰기와 읽기를 실행**

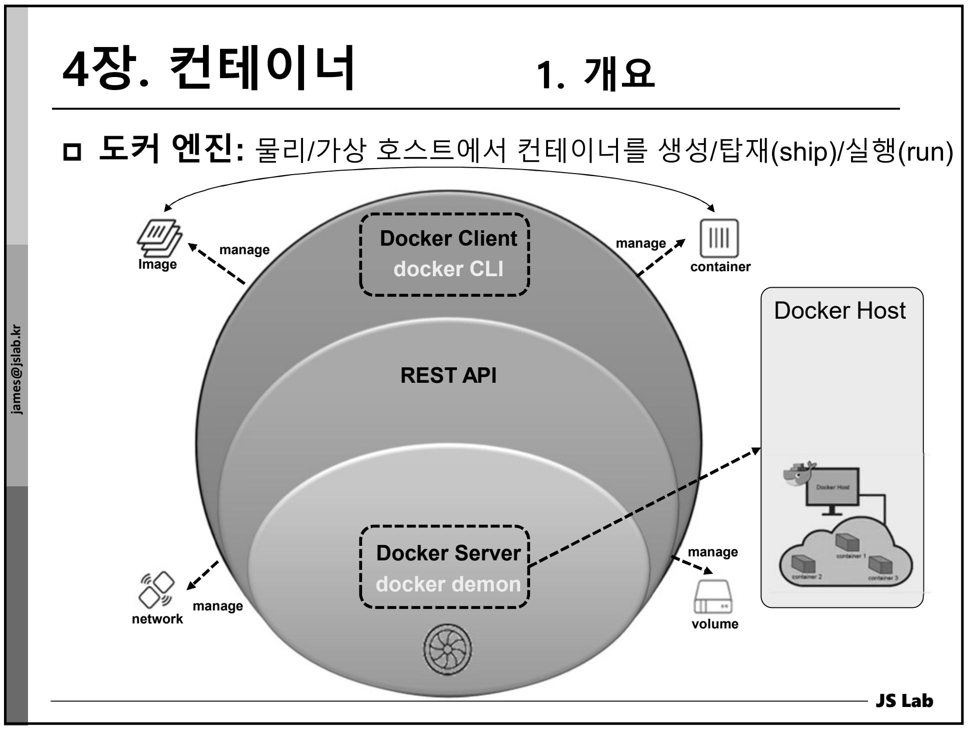


```
1 FROM ubuntu:15.04
2 RUN   apt-get install
3 COPY  ./app
4 RUN   make /app
```

james@jslab.kr

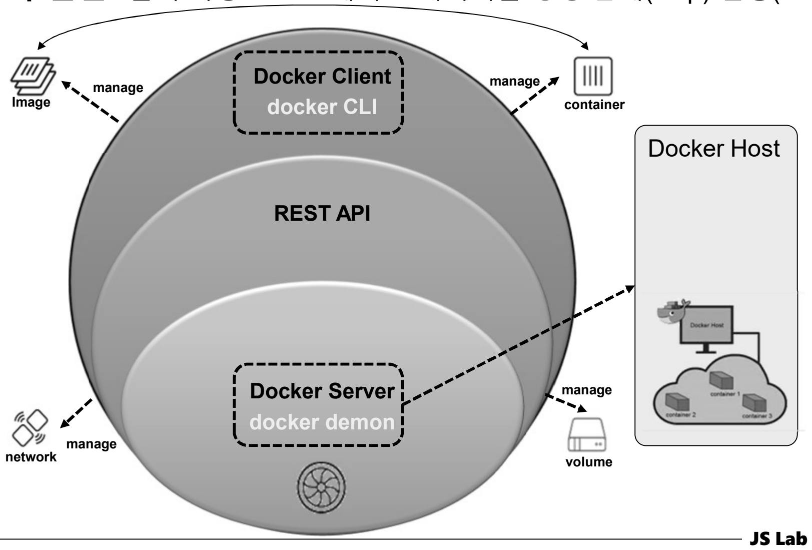
JS Lab

78



4장. 컨테이너 1. 개요

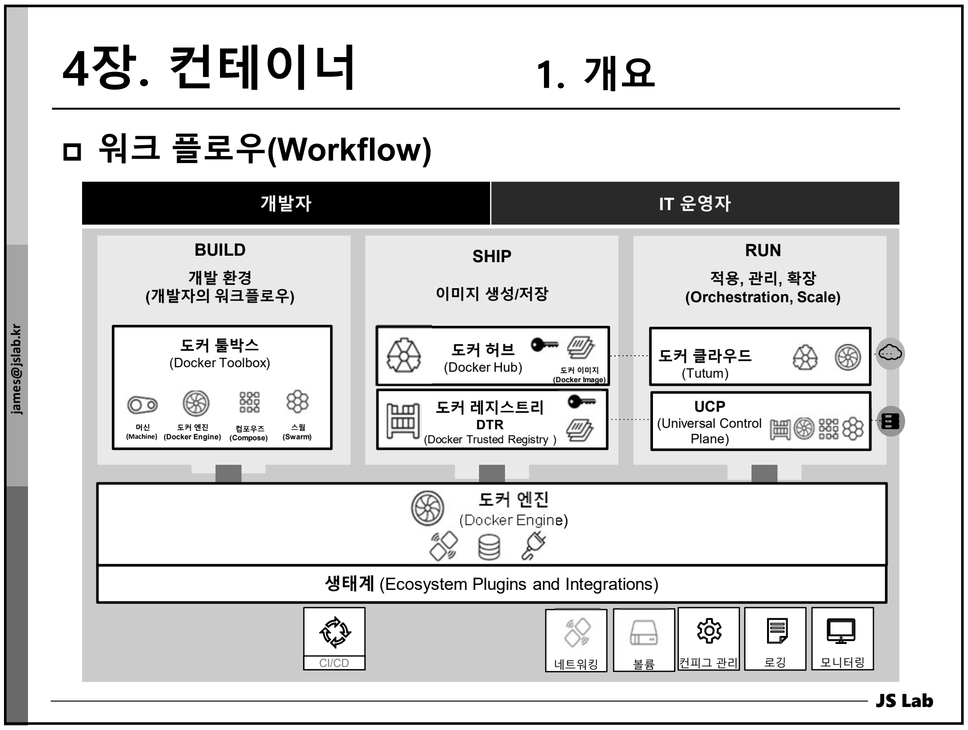
□ **도커 엔진:** 물리/가상 호스트에서 컨테이너를 생성/탑재(ship)/실행(run)



james@jslab.kr

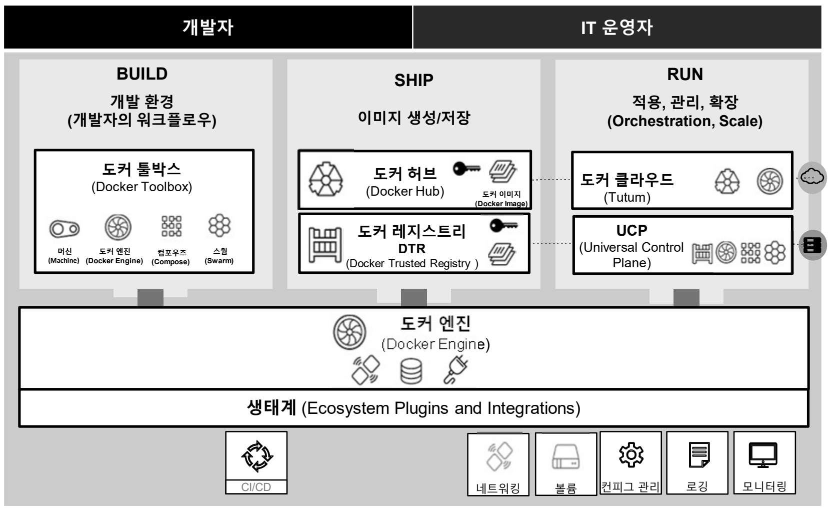
JS Lab

79



4장. 컨테이너 1. 개요

□ **워크 플로우(Workflow)**



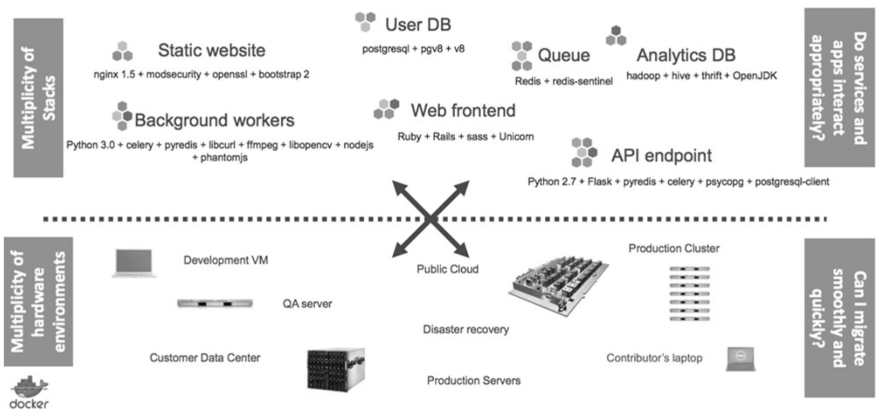
james@jslab.kr

JS Lab

80

4장. 컨테이너 1. 개요

- We need to make sure this works on multiple hardware and platforms, like developer laptops, VMs, data centers, public and private clouds, etc.


Multiplicity of Stacks
Static website
nginx 1.5 + modsecurity + openssl + bootstrap 2
User DB
postgresql + pgv8 + v8
Queue
Redis + redis-sentinel
Analytics DB
hadoop + hive + thrift + OpenJDK
Background workers
Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs
Web frontend
Ruby + Rails + sass + Unicorn
API endpoint
Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client
Do services and apps interact appropriately?
Multiplicity of hardware environments
Development VM
QA server
Public Cloud
Production Cluster
Disaster recovery
Customer Data Center
Production Servers
Contributor's laptop
Can I migrate smoothly and quickly?
docker

Running an Application (by Docker, Inc.; retrieved from docker.com)

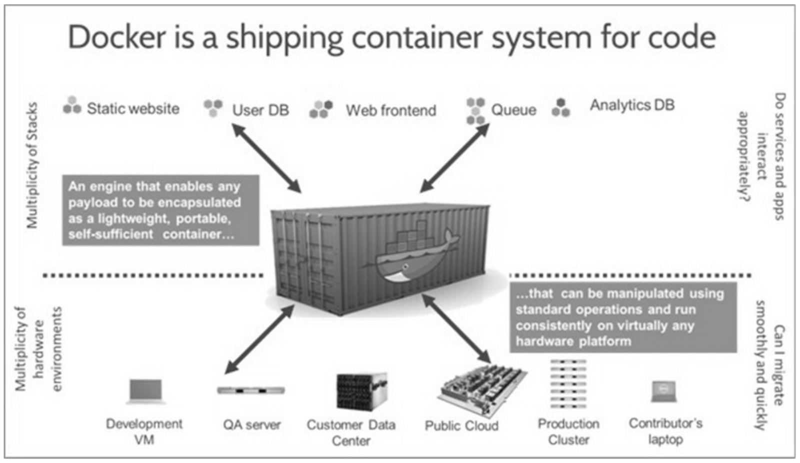
JS Lab

james@jslab.kr

81

4장. 컨테이너 1. 개요

- Our end goal is to run an application.


Docker is a shipping container system for code
Multiplicity of Stacks
Static website
User DB
Web frontend
Queue
Analytics DB
Do services and apps interact appropriately?
An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container...
...that can be manipulated using standard operations and run consistently on virtually any hardware platform
Multiplicity of hardware environments
Development VM
QA server
Customer Data Center
Public Cloud
Production Cluster
Contributor's laptop
Can I migrate smoothly and quickly

A Docker Container (by Docker, Inc.; retrieved from LinkedIn SlideShare)

JS Lab

james@jslab.kr

82

4장. 컨테이너 1. 개요

- Images and Containers
  - In the container world, this box (containing our application and all its dependencies) is referred to as an image. A running instance of this box is referred to as a container. We can spin multiple containers from the same image.
  - An image contains the application, its dependencies and the user-space libraries. User-space libraries like glibc enable switching from the user-space to the kernel-space. An image does not contain any kernel-space components.
  - When a container is created from an image, it runs as a process on the host's kernel. It is the host kernel's job to isolate and provide resources to each container.

4장. 컨테이너 1. 개요

- Namespaces
  - pid - provides each namespace to have the same PIDs. Each container has its own PID 1.
  - net - allows each namespace to have its network stack. Each container has its own IP address.
  - mnt - allows each namespace to have its own view of the filesystem hierarchy.
  - ipc - allows each namespace to have its own interprocess communication.
  - uts - allows each namespace to have its own hostname and domainname.
  - user - allows each namespace to have its own user and group ID number spaces. A root user inside a container is not the root user of the host on which the container is running

4장. 컨테이너 1. 개요

- cgroups
  - blkio
  - cpu
  - cpuacct
  - cpuset
  - devices
  - freezer
  - memory.

4장. 컨테이너 1. 개요

- Union filesystem
  - The Union filesystem allows files and directories of separate filesystems, known as layers, to be transparently overlaid on each other, to create a new virtual filesystem. An image used in Docker is made of multiple layers and, while starting a new container, we merge all those layers to create a read-only filesystem. On top of a read-only filesystem, a container gets a read-write layer, which is an ephemeral layer and it is local to the container.

4장. 컨테이너 1. 개요

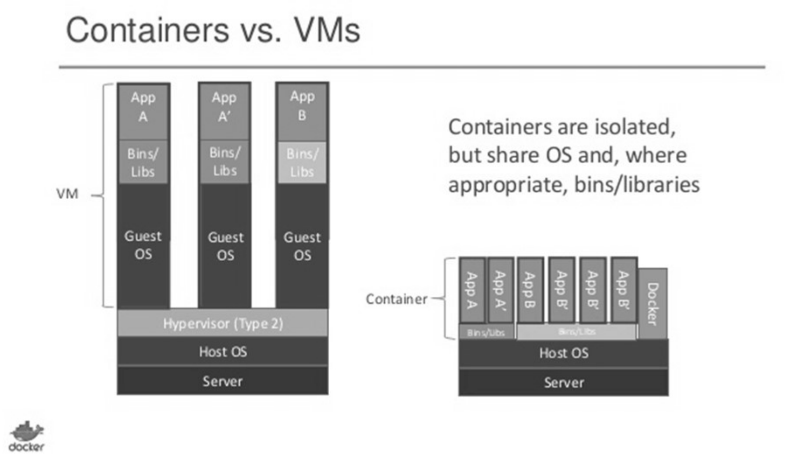
- Container Runtimes (1 of 2)
  - **runC:** Over the past few years, we have seen a rapid growth in the interest and adoption for container technologies. Most of the cloud providers and IT vendors offer support for containers. To make sure there is no vendor locking and no inclination towards a particular company or project, IT companies came together and formed an open governance structure, called The Open Container Initiative, under the auspices of The Linux Foundation. The governance body came up with specifications to create standards on Operating System process and application containers. runC is the CLI tool for spawning and running containers according to these specifications.
  - **Containerd:** containerd is an Open Container Initiative (OCI)-compliant container runtime with an emphasis on simplicity, robustness and portability. It runs as a daemon and manages the entire lifecycle of containers. It is available on Linux and Windows.

4장. 컨테이너 1. 개요

- Container Runtimes (2 of 2)
- **Docker, which is a containerization platform, uses containerd as a container runtime to manage runC containers.**
  - **rkt:** rkt (pronounced "rock-it") is an open source, Apache 2.0-licensed project from CoreOS. It implements the App Container specification.
  - **CRI-O:** CRI-O is an OCI-compatible runtime, which is an implementation of the Kubernetes Container Runtime Interface (CRI). It is a lightweight alternative to using Docker as the runtime for Kubernetes.

4장. 컨테이너 1. 개요

□ 가상머신과 컨테이너



Containers are isolated, but share OS and, where appropriate, bins/libraries

Docker Container vs VMs (by Docker, Inc.)

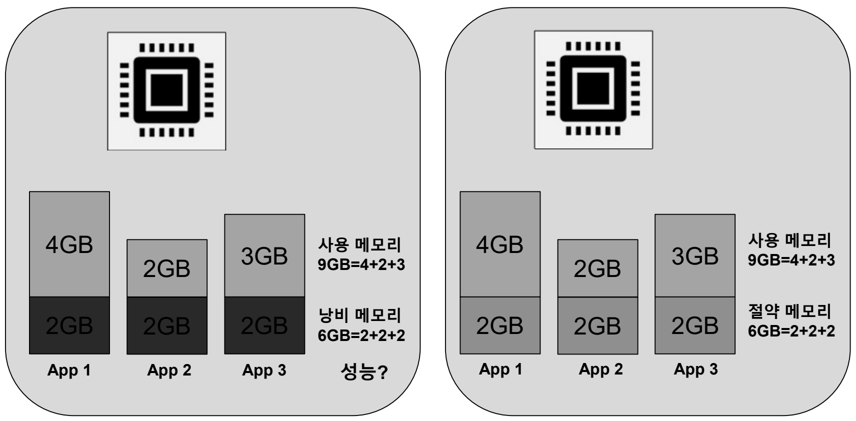
james@jslab.kr

JS Lab

89

4장. 컨테이너 1. 개요

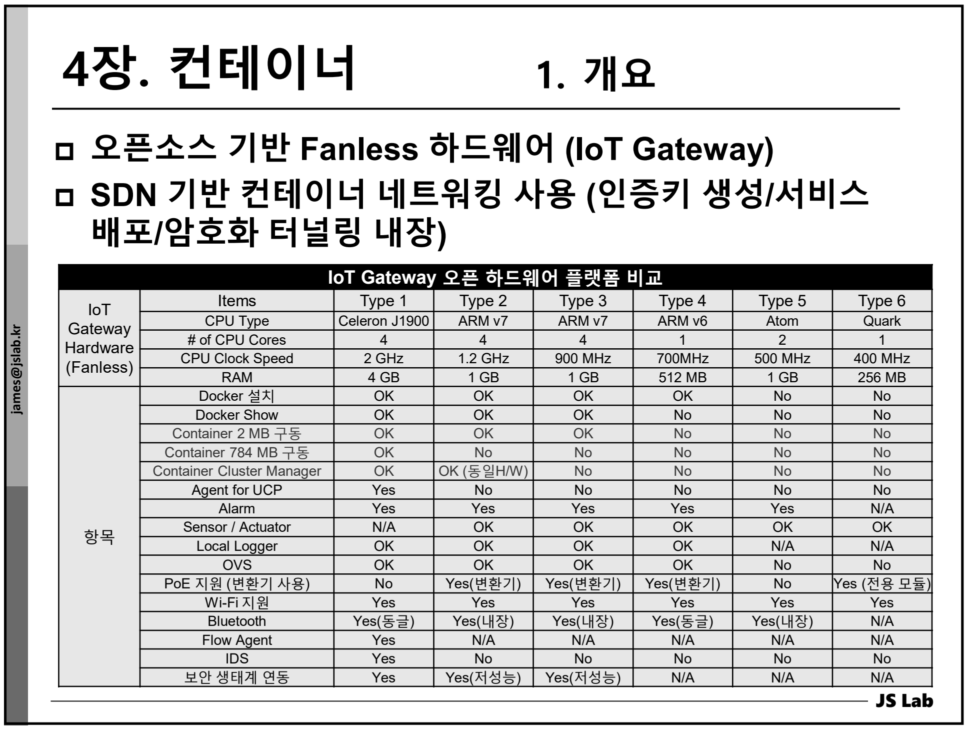
□ 가상머신과 컨테이너 메모리 사용



james@jslab.kr

JS Lab

90



4장. 컨테이너

1. 개요

- 오픈소스 기반 Fanless 하드웨어 (IoT Gateway)
- SDN 기반 컨테이너 네트워킹 사용 (인증키 생성/서비스 배포/암호화 터널링 내장)

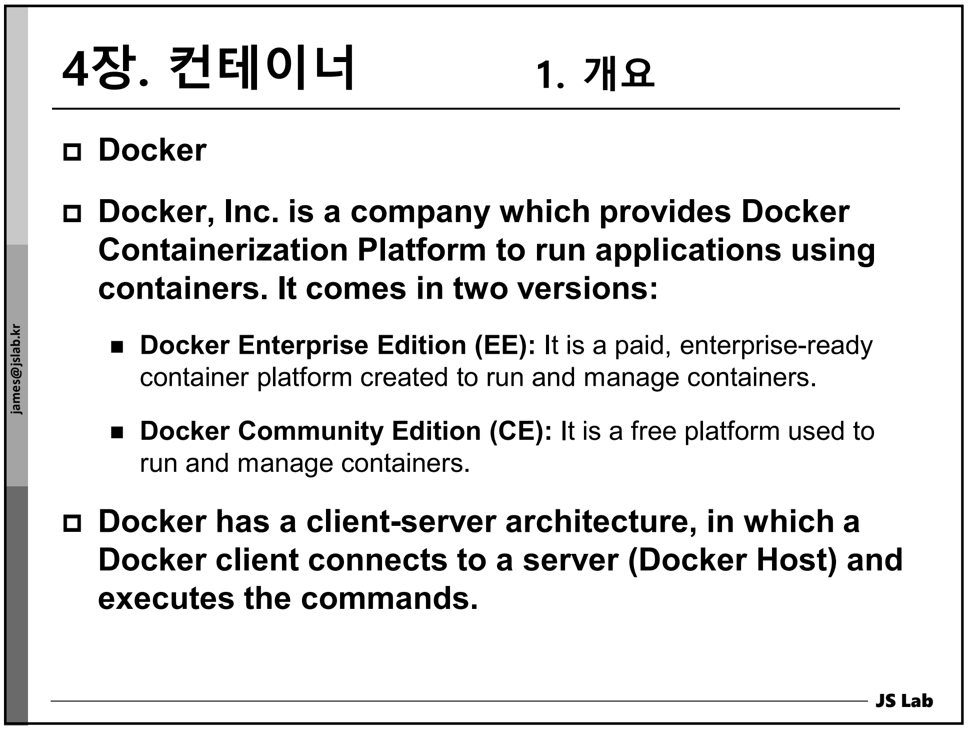
IoT Gateway 오픈 하드웨어 플랫폼 비교

	Items	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6
IoT Gateway Hardware (Fanless)	CPU Type	Celeron J1900	ARM v7	ARM v7	ARM v6	Atom	Quark
	# of CPU Cores	4	4	4	1	2	1
	CPU Clock Speed	2 GHz	1.2 GHz	900 MHz	700MHz	500 MHz	400 MHz
	RAM	4 GB	1 GB	1 GB	512 MB	1 GB	256 MB
항목	Docker 설치	OK	OK	OK	OK	No	No
	Docker Show	OK	OK	OK	No	No	No
	Container 2 MB 구동	OK	OK	OK	No	No	No
	Container 784 MB 구동	OK	No	No	No	No	No
	Container Cluster Manager	OK	OK (동일H/W)	No	No	No	No
	Agent for UCP	Yes	No	No	No	No	No
	Alarm	Yes	Yes	Yes	Yes	Yes	N/A
	Sensor / Actuator	N/A	OK	OK	OK	OK	OK
	Local Logger	OK	OK	OK	OK	N/A	N/A
	OVS	OK	OK	OK	OK	No	No
	PoE 지원 (변환기 사용)	No	Yes(변환기)	Yes(변환기)	Yes(변환기)	No	Yes (전용 모듈)
	Wi-Fi 지원	Yes	Yes	Yes	Yes	Yes	Yes
	Bluetooth	Yes(동글)	Yes(내장)	Yes(내장)	Yes(동글)	Yes(내장)	N/A
	Flow Agent	Yes	N/A	N/A	N/A	N/A	N/A
	IDS	Yes	No	No	No	No	No
	보안 생태계 연동	Yes	Yes(저성능)	Yes(저성능)	N/A	N/A	N/A

james@jslab.kr

JS Lab

91



4장. 컨테이너

1. 개요

- **Docker**
- **Docker, Inc. is a company which provides Docker Containerization Platform to run applications using containers. It comes in two versions:**
  - **Docker Enterprise Edition (EE):** It is a paid, enterprise-ready container platform created to run and manage containers.
  - **Docker Community Edition (CE):** It is a free platform used to run and manage containers.
- **Docker has a client-server architecture, in which a Docker client connects to a server (Docker Host) and executes the commands.**

james@jslab.kr

JS Lab

92

4장. 컨테이너 1. 개요

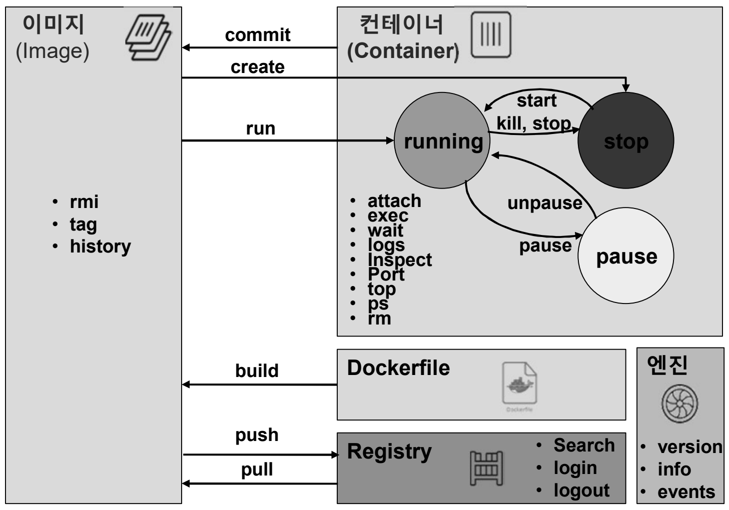
- **Basic Docker Operations**
  - **List images:** $ docker image ls
  - **Pulling an alpine image:** $ docker image pull alpine
  - **Run a container from a locally-available image:** $ docker container run -it alpine sh
  - **Run a container in the background (-d option) from an image:** $ docker container run -d nginx
  - **List only running containers:** $ docker container ls
  - **List all containers:** $ docker container ls -a
  - **Inject a process inside a running container:** $ docker container exec -it <container_id/name> bash
  - **Stop a container:** $ docker container stop <container id/name>
  - **Delete a container:** $ docker container rm <container id/name>

JS Lab

93

4장. 컨테이너 1. 개요

- 도커 명령어 다이어그램



JS Lab

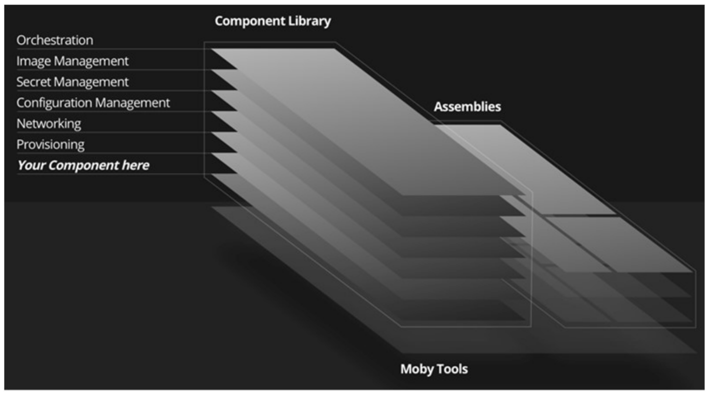
94

4장. 컨테이너 1. 개요

- Benefits of Using Containers
  - They have very little footprint.
  - They can be deployed very fast (within milliseconds).
  - They are a flexible solution, as they can run on any computer, infrastructure, or cloud environment.
  - They can be scaled up or down with ease.
  - There is a very rich ecosystem built around them.
  - Problem containers can be easily and quickly isolated when troubleshooting and solving problems.
  - Containers use less memory and CPU than VMs running similar workloads.
  - Increased productivity with reduced overhead.

4장. 컨테이너 1. 개요

- **Project Moby:** It is an open source project which provides a framework for assembling different container systems to build a container platform like Docker. Individual container systems provide features like image, container, secret management, etc.

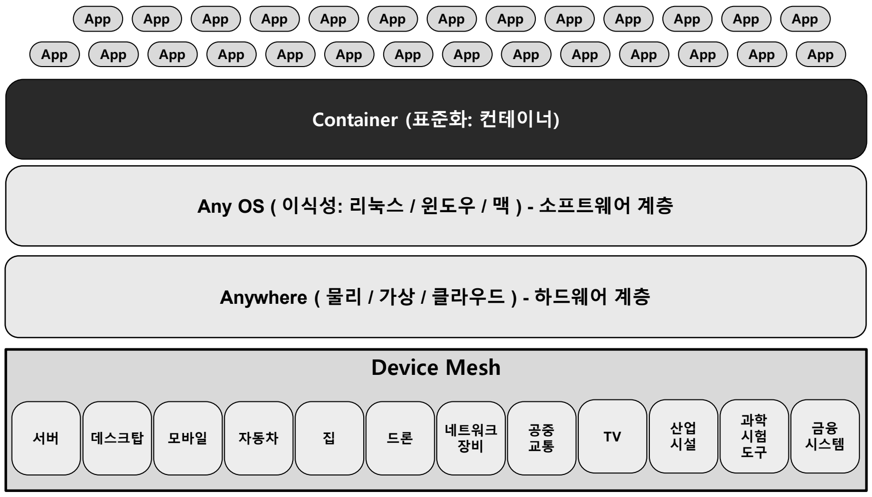


Project Moby (retrieved from https://mobyproject.org/)

4장. 컨테이너

1. 개요

□ 요약


App App App App App App App App App App App App App
App App App App App App App App App App App App App App
Container (표준화: 컨테이너)
Any OS (이식성: 리눅스 / 윈도우 / 맥) - 소프트웨어 계층
Anywhere (물리 / 가상 / 클라우드) - 하드웨어 계층
Device Mesh
서버
데스크탑
모바일
자동차
집
드론
네트워크 장비
공중 교통
TV
산업 시설
과학 시험 도구
금융 시스템

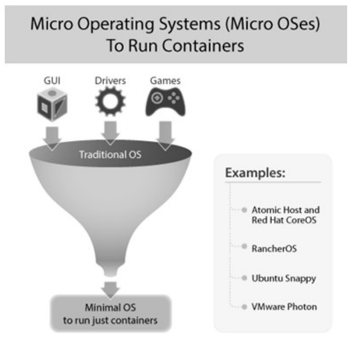
97

4장. 컨테이너

2. Micro OS

□ Micro OSes for containers

- Atomic Host and Red Hat CoreOS
- RancherOS
- Ubuntu Snappy
- VMware Photon


Micro Operating Systems (Micro OSes) To Run Containers
GUI
Drivers
Games
Traditional OS
Minimal OS to run just containers
Examples:
Atomic Host and Red Hat CoreOS
RancherOS
Ubuntu Snappy
VMware Photon

98

4장. 컨테이너 2. Micro OS

- **Atomic Host and Red Hat CoreOS**
  - **Atomic Host is a lightweight operating system, assembled out of a specific RPM content. It allows us to run just containerized applications in a quick and reliable manner. Atomic Host can be based on Fedora, CentOS, or Red Hat Enterprise Linux (RHEL).**
  - **Atomic Host is a sub-project of Project Atomic, which includes other sub-projects, such as Buildah, Cockpit and skopeo.**
  - **Currently, Atomic Host comes out-of-the-box with Kubernetes installed. It also includes several Kubernetes utilities, such as etcd and flannel.**

4장. 컨테이너 2. Micro OS

- **Components of Atomic Host**
  - **rpm-ostree:** One cannot manage individual packages on Atomic Host, as there is no rpm or other related commands. To get any required service, you would have to start a respective container. Atomic Host has two bootable, immutable, and versioned filesystems; one is used to boot the system and the other is used to fetch updates from upstream. rpm-ostree is the tool to manage these two versioned filesystems.
  - **Systemd:** It is used to manage system services for Atomic Host.
  - **Docker:** Atomic Host currently supports Docker as a container runtime.
  - **Kubernetes:** With Kubernetes, we can create a cluster of Atomic Hosts to run applications at scale

4장. 컨테이너 2. Micro OS

- Benefits of Using Atomic Host
  - It is an OS specifically designed to run containerized applications.
  - It enables us to perform quick updates and rollbacks.
  - It provides increased security through SELinux.
  - It can be installed on bare metal, as well as VMs.
  - It can be based on Fedora, CentOS, and Red Hat Enterprise Linux.
  - Nodes can be clustered on Atomic Host using Kubernetes.
  - It has tools like Cockpit, which provide cross-cluster capabilities to deploy and manage applications.

JS Lab

101

4장. 컨테이너 2. Micro OS

- VMware Photon
  - Photon OS™ is a technology preview of a minimal Linux container host provided by VMware. It is designed to have a small footprint and boot extremely quickly on VMware platforms.
  - Photon OS™ comes in two versions - minimal and full. In a minimal version, it offers packages to run containers. In a full version, it includes additional packages to help develop, test, and deploy containerized applications.

JS Lab

102

4장. 컨테이너 2. Micro OS

□ Features and Availability

- Photon OS™ is optimized for VMware products and platforms. It supports Docker, rkt, and the Pivotal Garden container specifications. It also has a new, open source, yum-compatible package manager (tdnf).
- Photon OS™ is a security-hardened Linux. The kernel and other aspects of the Photon OS™ are built with an emphasis on security recommendations given by the Kernel Self-Protection Project (KSPP).
- It can be easily managed, patched, and updated. It also provides support for persistent volumes to store the data of cloud-native applications on VMware vSAN™ .
- If you want to try it out, Photon OS™ is available on Amazon EC2, Google Cloud, and Microsoft Azure.

4장. 컨테이너 2. Micro OS

□ Benefits of Using VMware Photon

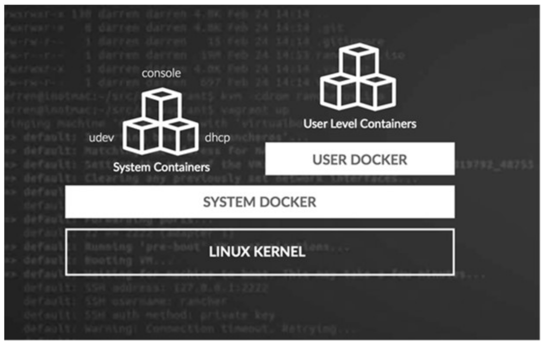
- It is an open source technology with a small footprint.
- It supports Docker, rkt, and Pivotal Garden container runtimes.
- We can use Kubernetes, Swarm and Mesos clusters on top of Photon.
- It boots extremely quickly on VMware platforms.
- It provides an efficient lifecycle management with a yum-compatible package manager.
- Its kernel is tuned for higher performance when its running on VMware platforms.
- It is a security-enhanced Linux as its kernel and other aspects of the operating system are configured according to the security parameter recommendations given by the Kernel Self-Protection Project.

4장. 컨테이너 2. Micro OS

- **RancherOS**
  - **RancherOS is a 20 MB Linux distribution that runs Docker containers. It has the least footprint of all the Micro OSes available nowadays. This is possible because it runs directly on top of the Linux kernel.**
  - **RancherOS is a product provided by Rancher, which is an end-to-end platform used to deploy and run private container services.**

4장. 컨테이너 2. Micro OS

- **Components:** RancherOS runs two instances of the Docker daemon. Just after booting, it starts the first instance of the Docker daemon with PID 1 to run system containers like dhcp, udev, etc. To run user-level containers, the System Docker daemon creates a service to start other Docker daemons.



RancherOS Architecture (by Rancher, retrieved from rancher.com)

4장. 컨테이너 2. Micro OS

◘ **Benefits of Using RancherOS**

- **It has the least footprint of all Micro OSes available.**
- **It runs directly on top of the Linux kernel.**
- **It enables us to perform updates and rollbacks in a simple manner.**
- **We can use the Rancher platform to set up Kubernetes.**
- **It boots the containers within seconds.**
- **It automates OS configuration with cloud-init.**
- **It can be customized to add custom system Docker containers using the cloud-init file or Docker Compose.**

107

4장. 컨테이너 3. Orchestration

◘ **클라우드 관리/오케스트레이션/인프라 구성 관리 도구 비교**

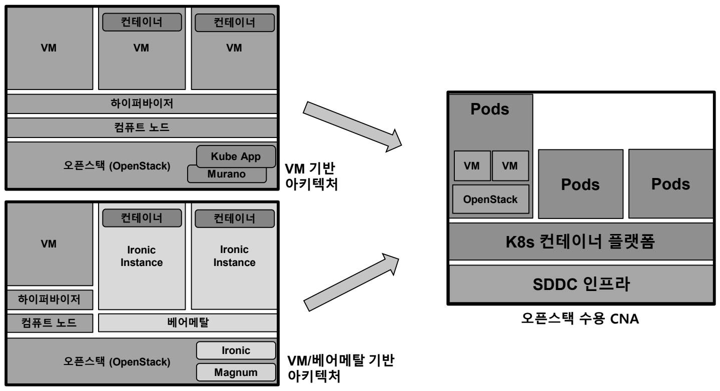
	클라우드 관리 플랫폼 (Infrastructure First)	애플리케이션 플랫폼 (Application First Approach)	자동화/오케스트레이션 플랫폼 (Automation First)
장점	• Single Pane of Glass • 비용 분석과 제어	• 쿠버네티스(Kubernetes or K8s)와 컨테이너는 주요 클라우드에서 지원 • 애플리케이션과 마이크로서비스 중심	• 퍼블릭/프라이빗 클라우드등 다양하고 폭넓은 지원 • 모든 클라우드나 애플리케이션 지원 구조 • 커스터마이징 가능
단점	• 제한적인 애플리케이션 인식 • DevOps프로세스에 부적합 • 제한적인 클라우드 서비스	• 대부분 그린필드에 적합 • 다른 영역에 호환성 부족	• 애플리케이션과 클라우드 단위로 커스터마이징 필요
	RED HAT CLOUDFORMS RIGHTSCALE CLOUD MANAGEMENT IBM vmware MORPHEUS SCALR	OPENSHIFT RANCHER CLOUD FOUNDRY kubernetes	TERRAFORM CLOUDIFY ARIA OASIS TOSCA

108

4장. 컨테이너 3. Orchestration

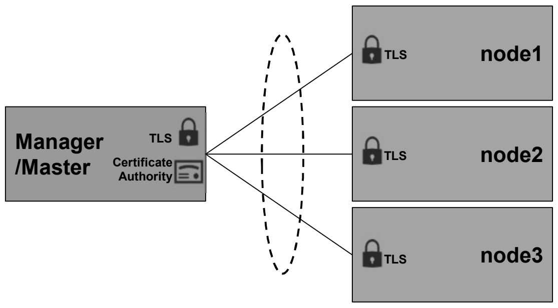
❖ 기존 서비스 보호 CNA 마이그레이션 체계

- VM 기반 아키텍처: Pet 기반 운영 가능
- VM/베어메탈 기반 아키텍처: VM/베어메탈 지원
- 오픈스택 수용 CNA: 호환성을 위한 VM 필요시 오픈스택 구동



4장. 컨테이너 3. Orchestration

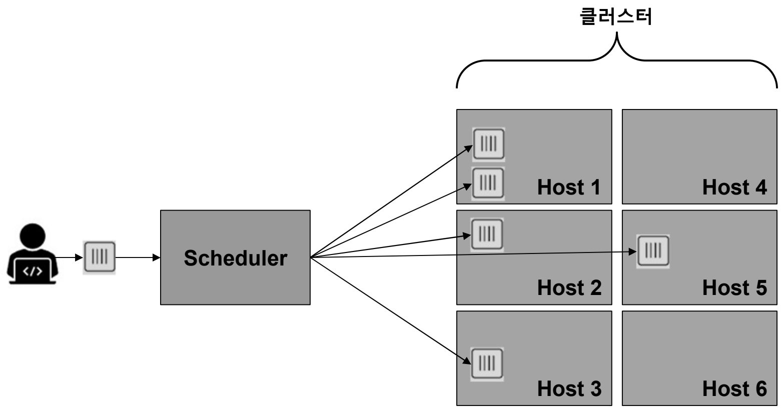
◘ Overlay Network



4장. 컨테이너 3. Orchestration

james@jslab.kr

□ Distributed system scheduler in action



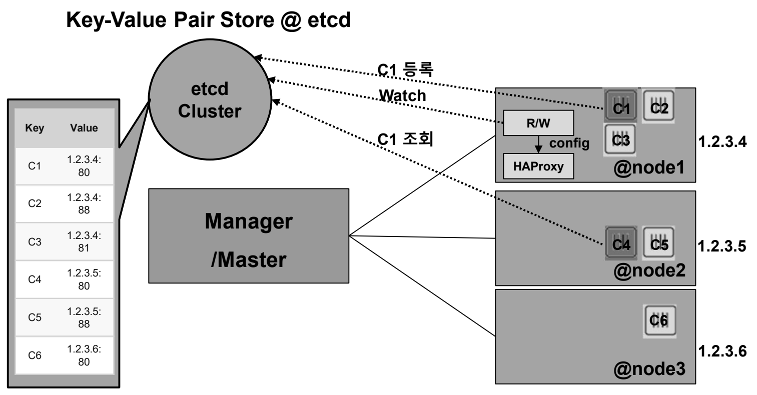
JS Lab

111

4장. 컨테이너 3. Orchestration

james@jslab.kr

□ etcd service discovery setup



Key	Value
C1	1.2.3.4: 80
C2	1.2.3.4: 88
C3	1.2.3.4: 81
C4	1.2.3.5: 80
C5	1.2.3.5: 88
C6	1.2.3.6: 80

JS Lab

112

4장. 컨테이너 3. Orchestration

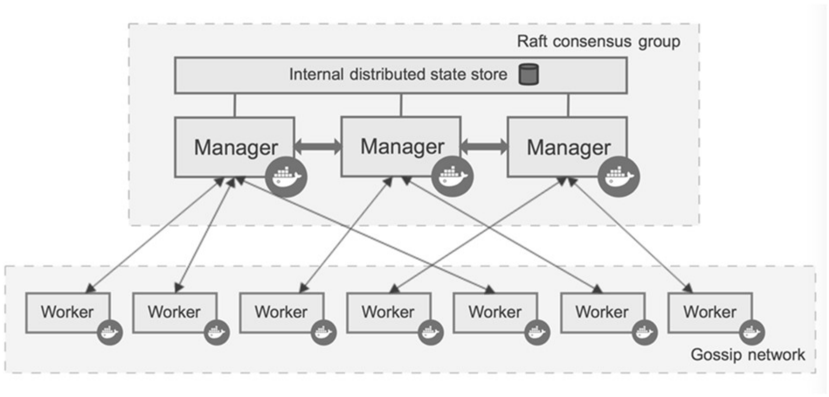
- Container orchestration
  - Who can bring multiple hosts together and make them part of a cluster?
  - Who will schedule the containers to run on specific hosts?
  - How can containers running on one host reach out to containers running on different hosts?
  - Who will make sure that the container has the dependent storage, when it is scheduled on a specific host?
  - Who will make sure that containers are accessible over a service name, so that we do not have to bother about container accessibility over IP addresses?

4장. 컨테이너 3. Orchestration

- Container orchestration tools, along with different plugins (like networking and storage)
  - Docker Swarm
  - Kubernetes
  - Mesos Marathon
  - Nomad
  - Amazon ECS.

4장. 컨테이너 3. Orchestration

- **Docker Swarm:** Docker Swarm is a native container orchestration tool from Docker, Inc. It logically groups multiple Docker engines to create a virtual engine, on which we can deploy and scale applications.



Swarm Cluster Components (by Docker, Inc., retrieved from docker.com)

JS Lab

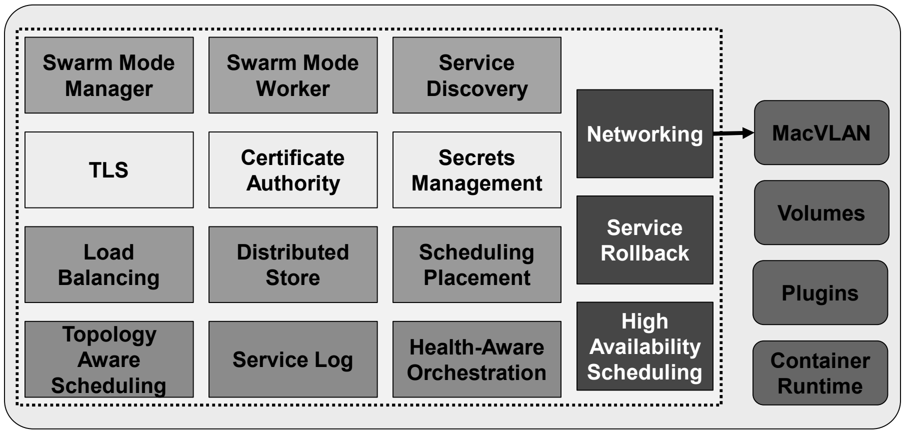
james@jslab.kr

115

4장. 컨테이너 3. Orchestration

- **Docker Swarm Mode 예 (17.06)**
- **Docker는 Swarm과 함께 Kubernetes를 적극 수용**

오케스트레이션 요소



JS Lab

james@jslab.kr

116

4장. 컨테이너 3. Orchestration

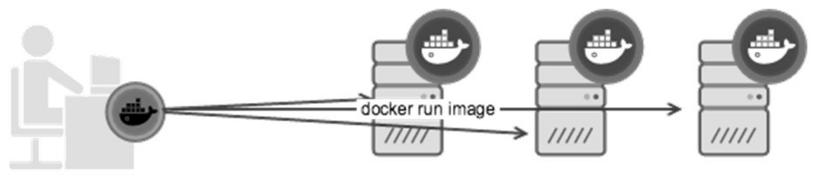
□ Features

- It is compatible with Docker tools and API, so that the existing workflow does not change much.
- It provides native support to Docker networking and volumes.
- It can scale up to large numbers of nodes.
- It supports failover and High Availability for the cluster manager.
- It uses a declarative approach to define the desired state of the various services of the application stack.
- For each service, you can declare the number of tasks you want to run. When you scale up or down, the Swarm manager automatically adapts by adding or removing tasks to maintain the desired state.
- The Docker Swarm manager node constantly monitors the cluster state and reconciles any differences between the actual state and your expressed desired state.
- The communication between the nodes of Docker Swarm is enforced with Transport Layer Security (TLS), which makes it secure by default.
- It supports rolling updates, using which we can control the delay between service deployment to different sets of nodes. If anything goes wrong, you can roll back a task to a previous version of the service.

4장. 컨테이너 3. Orchestration

□ Docker Machine

- Setting up the Docker engine using the VirtualBox driver:
- $ docker-machine create -d virtualbox dev1
- Setting up the Docker engine using DigitalOcean:
- $ docker-machine create --driver digitalocean --digitalocean-access-token=<TOKEN> dev2



Provisioning Docker Hosts on Remote Systems (by Docker, Inc., retrieved from docker.com)

4장. 컨테이너 3. Orchestration

Docker Compose

- Docker Compose allows us to define and run multi-container applications through a configuration file. In a configuration file, we can define services, images or Dockerfiles to use, network, etc. Below we provide a sample of a Compose file

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
volumes:
    db_data:
```

119

4장. 컨테이너 3. Orchestration

Benefits of Using Docker Swarm

- It provides native clustering for Docker.
- It is well integrated with the existing Docker tools and workflow.
- Its setup is easy, straightforward and flexible.
- It manages a cluster of containers as a single entity.
- It provides scalability and supports High Availability.
- Efficiency and productivity are increased by reducing deployment and management time, as well as duplication of efforts.

120

4장. 컨테이너 3. Orchestration

□ **Docker Enterprise Edition**

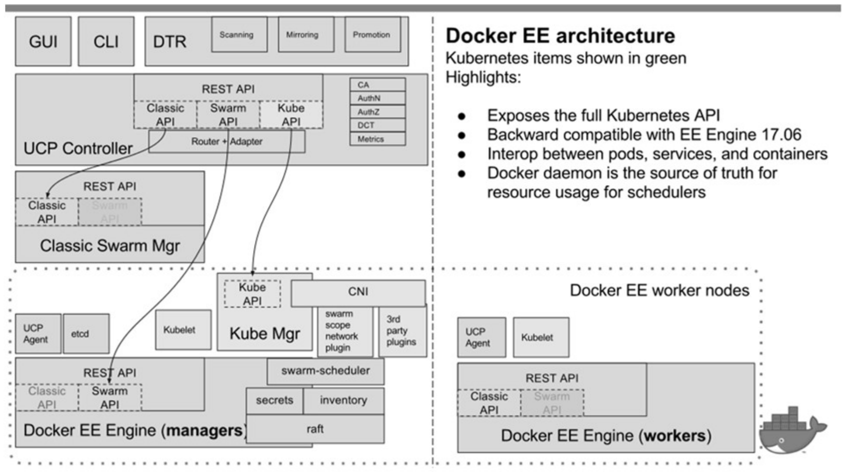
- **Docker EE Engine:** It is a commercially supported Docker Engine for creating images and running Docker containers.
- **Docker Trusted Registry (DTR):** It is a production-grade image registry designed to store images, from Docker, Inc.
- **Universal Control Plane (UCP):** It manages the Kubernetes and Swarm orchestrators, deploys applications using the CLI and GUI, and provides High Availability. UCP also provides role-based access control to ensure that only authorized users can make changes and deploy applications to your cluster.

JS Lab

121

4장. 컨테이너 3. Orchestration

□ **Docker EE Architecture**



Docker EE architecture
Kubernetes items shown in green
Highlights:

- Exposes the full Kubernetes API
- Backward compatible with EE Engine 17.06
- Interop between pods, services, and containers
- Docker daemon is the source of truth for resource usage for schedulers

Docker EE Architecture (by Docker, Inc., retrieved from docker.com)

JS Lab

122

4장. 컨테이너 3. Orchestration

james@jslab.kr

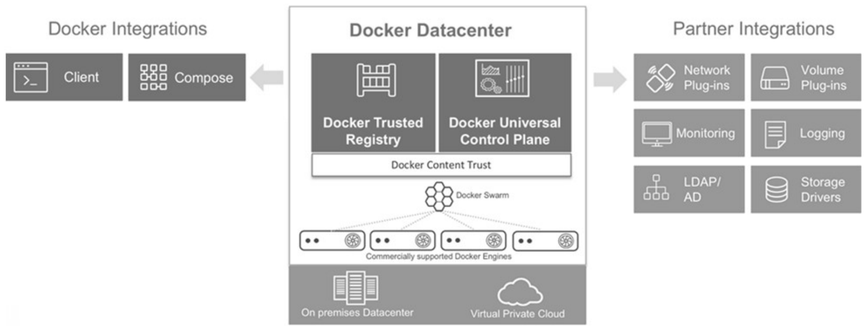
◻ Features and Benefits

- It is a multi-Linux, multi-OS, multi-Cloud solution.
- It supports Docker Swarm and Kubernetes as container orchestrators.
- It provides centralized cluster management.
- It has a built-in authentication mechanism with role-based access control (RBAC).

4장. 컨테이너 3. Orchestration

james@jslab.kr

◻ Docker Datacenter

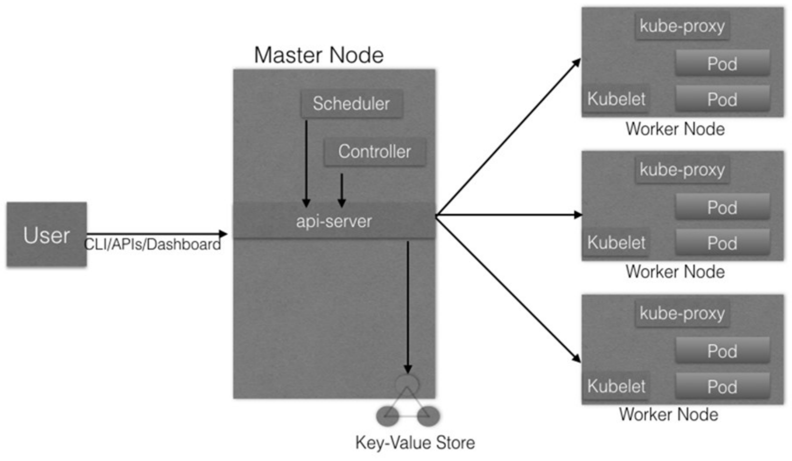


Docker Datacenter (by Docker, Inc., retrieved from docker.com)

4장. 컨테이너

3. Orchestration

◘ **The Kubernetes Architecture**



james@jslab.kr

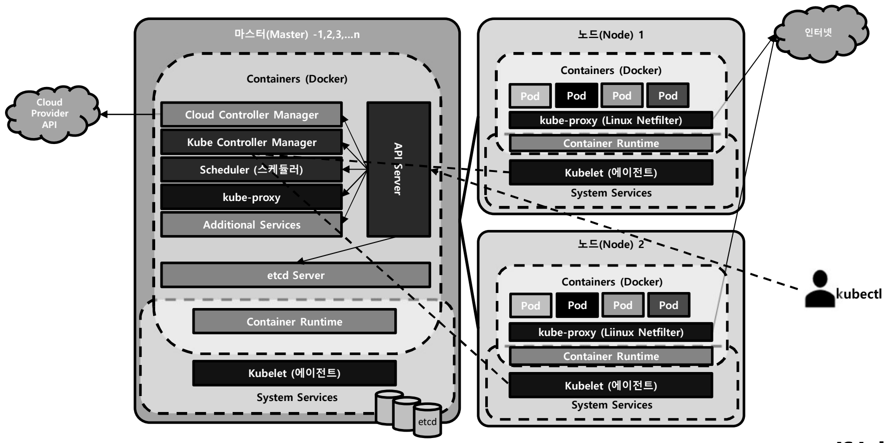
JS Lab

125

4장. 컨테이너

3. Orchestration

◘ **아키텍처 구성: 마스터 노드(Master Node)와 1개 이상의 워커 노드(Worker Node)로 이뤄진 가상/물리 머신의 꾸러미(Set)를 통해 클러스터(Cluster) 구성**



james@jslab.kr

JS Lab

126

4장. 컨테이너

3. Orchestration

◻ The Kubernetes Architecture - Components (1 of 4)

- **Cluster:** The cluster is a group of systems (physical or virtual) and other infrastructure resources used by Kubernetes to run containerized applications.
- **Master Node:** The master is a system that takes pod scheduling decisions and manages the replication and manager nodes. It has three main components: API Server, Scheduler, and Controller. There can be more than one master node.
- **Worker Node:** A system on which pods are scheduled and run. The node runs a daemon called kubelet to communicate with the master node. kube-proxy, which runs on all nodes, allows applications from the external world.

4장. 컨테이너

3. Orchestration

◻ The Kubernetes Architecture - Components (2 of 4)

- **Key-Value Store:** The Kubernetes cluster state is saved in a key-value store, like etcd. It can be either part of the same Kubernetes cluster or it can reside
- **Pod:** The pod is a co-located group of containers with shared volumes. It is the smallest deployment unit in Kubernetes. A pod can be created independently, but it is recommended to use the Replica Set, even if only a single pod is being deployed.
- **Replica Set:** The Replica Set manages the lifecycle of pods. It makes sure that the desired numbers of pods is running at any given point in time

4장. 컨테이너 3. Orchestration

□ The Kubernetes Architecture - Components (3 of 4)

- **Deployments:** Deployments allow us to provide declarative updates for pods and Replica Sets. We can define Deployments to create new resources, or replace existing ones with new ones. Some typical use cases are presented below:
  1. Create a Deployment to bring up a Replica Set and pods.
  2. Check the status of a Deployment to see if it succeeds or not.
  3. Later, update that Deployment to recreate the pods (for example, to use a new image).
  4. Roll back to an earlier Deployment revision if the current Deployment isn't stable.
  5. Pause and resume a Deployment. Below we provide a sample deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
    template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 8
```

4장. 컨테이너 3. Orchestration

□ The Kubernetes Architecture - Components (4 of 4)

- **Service :** The service groups sets of pods together and provides a way to refer to them from a single static IP address and the corresponding DNS name. Below, we provide an example of a service file:
- **Label:** The label is an arbitrary key-value pair which is attached to a resource like pod, Replica Set, etc. In the example above, we defined labels as app and tier.
- **Selector:** Selectors enable us to group resources based on labels. In the above example, the frontend service will select all pods which have the labels app==dockchat and tier==frontend.
- **Volume:** The volume is an external filesystem or storage which is available to pods. They are built on top of Docker volumes.
- **Namespace:** The namespace allows us to partition the cluster into sub-clusters.

```
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    app: dockchat
    tier: frontend
spec:
  type: LoadBalancer
  ports:
  - port: 5000
  selector:
    app: dockchat
    tier: frontend
```

4장. 컨테이너 3. Orchestration

- Features (1 of 2)
  - It automatically places containers based on resource requirements and other constraints.
  - It supports horizontal scaling through the CLI and GUI. It can auto-scale based on the CPU load as well.
  - It supports rolling updates and rollbacks.
  - It supports multiple volume plugins like the GCP/AWS disk, NFS, iSCSI, Ceph, Gluster, Cinder, Flocker, etc. to attach volumes to pods.

4장. 컨테이너 3. Orchestration

- Features (2 of 2)
  - It automatically self-heals by restarting failed pods, rescheduling pods from failed nodes, etc.
  - It deploys and updates secrets for an application without rebuilding the image.
  - It supports batch execution.
  - It support High Availability cluster.
  - It eliminates infrastructure lock-in by providing core capabilities for containers without imposing restrictions.
  - We can deploy and update the application at scale.

4장. 컨테이너 3. Orchestration

□ **Benefits of Using Kubernetes**

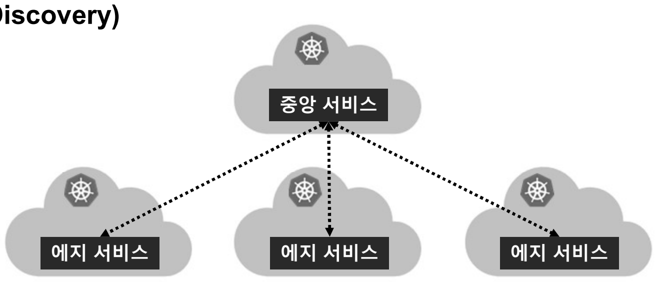
- **It is an open source system that packages all the necessary tools: orchestration, service discovery, load balancing.**
- **It manages multiple containers at scale.**
- **It automates deployment, scaling, and operations of application containers.**
- **It is portable, extensible and self-healing.**
- **It provides consistency across development, testing, and production environments.**
- **It is highly efficient in utilizing resources**

133

4장. 컨테이너 3. Orchestration

□ **Kubernetes 요약:**

- 워크로드 **Agnostic (**컨테이너**, VM, Function)**
- 다양한 요구의 하드웨어 플랫폼 지원
- 역동적 서비스를 위한 앱의 이동과 처리 증가와 감소의 장점
- 상용 적용 확장을 위한 일관된 플랫폼의 검증
- 신개발이 필요하지 않은 수준의 많은 레퍼런스로 개발자가 익숙함
- **Private Cloud**는 기기 관리 필요 **(PXE, DHCP, IPMI, TFTP, Discovery)**



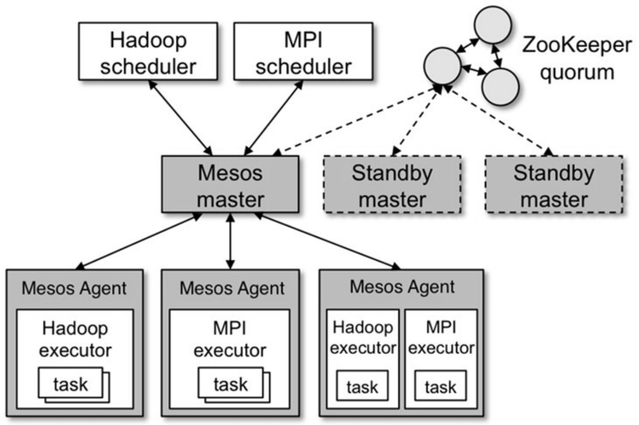
134

4장. 컨테이너 3. Orchestration

- Apache Mesos was created with this idea in mind, so that we can optimally use the resources available, even if we are running disparate applications on a pool of nodes. It helps us treat a cluster of nodes as one big computer, which manages CPU, memory, and other resources across a cluster. Mesos provides functionality that crosses between Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). It is an open source Apache project.

4장. 컨테이너 3. Orchestration

- Apache Mesos



Mesos Components (by The Apache Software Foundation, retrieved from mesos.apache.org)

4장. 컨테이너 3. Orchestration

- Mesos Features
  - It can scale up to 10,000 nodes.
  - It uses ZooKeeper for fault-tolerant replicated master and slaves.
  - It provides support for Docker containers.
  - It enables native isolation between tasks with Linux containers.
  - It allows multi-resource scheduling (memory, CPU, disk, and ports).
  - It uses Java, Python and C++ APIs to develop new parallel applications.
  - It uses WebUI to view cluster statistics.
  - It allows high resource utilization.
  - It helps handling mixed workloads.
  - It provides an easy-to-use container orchestration right out of the box.

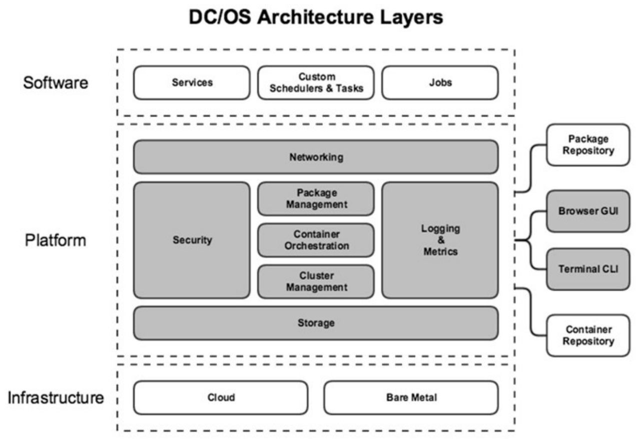
4장. 컨테이너 3. Orchestration

- Mesosphere DC/OS
  - Mesosphere offers a commercial solution on top of Apache Mesos. Mesosphere is one of the primary contributors to the Mesos project and to frameworks like Marathon. Their commercial product, Mesosphere Enterprise DC/OS, offers a one-click install and enterprise features like security, monitoring, user interface, etc. on top of Mesos.
  - Datacenter Operating System (DC/OS) has recently been open sourced by Mesosphere.
  - By default, DC/OS comes with the Marathon framework and others can be added as required.

4장. 컨테이너

3. Orchestration

◻ DC/OS Architecture Layers



DC/OS Architecture Layers (retrieved from mesosphere.com)

139

4장. 컨테이너

3. Orchestration

◻ Marathon framework

- It starts, stops, scales, and updates applications.
- It has a nice web interface, API.
- It is highly available, with no single point of failure.
- It uses native Docker support.
- It supports rolling deploy/restart.
- It allows application health checks.
- It provides artifact staging

140

4장. 컨테이너 3. Orchestration

- DC/OS Master
  - **Mesos Master Process:** It is similar to the master component of Mesos.
  - **Mesos DNS:** It provides service discovery within the cluster, so applications and services running inside the cluster can reach to each other.
  - **Marathon:** It is a framework which comes by default with DC/OS and provides the init system.
  - **ZooKeeper:** It is a high-performance coordination service that manages the DC/OS services.
  - **Admin Router:** It is an open source Nginx configuration created by Mesosphere, providing central authentication and proxy to DC/OS services within the cluster.

141

4장. 컨테이너 3. Orchestration

- DC/OS Agent nodes
  - **Mesos Agent Process:** It runs the mesos-slave process, which is similar to the slave component of Mesos.
  - **Mesos Containerize:** It provides lightweight containerization and resource isolation of executors, using Linux-specific functionality, such as cgroups and namespaces.
  - **Docker Container:** It provides support for launching tasks that contain Docker images.

142

4장. 컨테이너 3. Orchestration

- **Benefits of Using Mesos**
  - **It is an open source solution, but it also has a commercial version.**
  - **It provides support for Docker containers.**
  - **It allows multi-resource scheduling.**
  - **It is highly available and scalable.**
  - **It provides service discovery and load balancing**

143

4장. 컨테이너 3. Orchestration

- **Nomad:** HashiCorp Nomad is a cluster manager and resource scheduler from HashiCorp, which is distributed, highly available, and scales to thousands of nodes. It is especially designed to run microservices and batch jobs, and it supports different workloads, like containers (Docker), VMs, and individual applications. It is also capable of scheduling applications and services on different platforms like Linux, Windows and Mac.

```
# Define the hashicorp/web/frontend job
job "hashicorp/web/frontend" {
   # Run in two datacenters
   datacenters = ["us-west-1", "us-east-1"]

   # Only run our workload on linux
   constraint {
      attribute = "$attr.kernel.name"
      value = "linux"
   }

   # Configure the job to do rolling updates
   update {
      # Stagger updates every 30 seconds
      stagger = "30s"

      # Update a single task at a time
      max_parallel = 1
   }

   # Define the task group
   group "frontend" {
      # Ensure we have enough servers to handle traffic
      count = 10

      task "web" {
         # Use Docker to run our server
         driver = "docker"
         config {
            image = "hashicorp/web-frontend:latest"
         }

         # Ask for some resources
         resources {
            cpu = 500
            memory = 128
            network {
               mbits = 10
               dynamic_ports = ["http"]
            }
         }
      }
   }
}
```

which would use 10 containers from the `hashicorp/web-frontend:latest` Docker image.

144

4장. 컨테이너 3. Orchestration

□ Features

- It handles both cluster management and resource scheduling.
- It supports multiple workloads, like containers (Docker), VMs, unikernels, and individual applications.
- It has multi-datacenter and multi-region support. We can have a Nomad client/server running in different clouds, which form the same logical Nomad cluster.
- It bin-packs applications onto servers to achieve high resource utilization.
- In Nomad, millions of containers can be deployed or upgraded by using the job file.
- It provides a built-in dry run execution facility, which shows the scheduling actions that are going to take place.
- It ensures that applications are running in failure scenarios.
- It supports long-running services, as well as batch jobs and cron jobs.
- It provides a built-in mechanism for rolling upgrades.
- Blue-green and canary deployments can be deployed using a declarative job file syntax.
- If nodes fail, Nomad automatically redeploys the application from unhealthy nodes to healthy nodes.

4장. 컨테이너 3. Orchestration

□ Benefits of Using Nomad

- It is an open source solution that is distributed as a single binary for servers and agents.
- It is highly available and scalable.
- It maximizes resource utilization.
- It provides multi-datacenter and multi-region support.
- When maintenance is done, there is zero downtime to the datacenter and services.
- It simplifies operations, provides flexible workloads, and fast deployment.
- It can integrate with the entire HashiCorp ecosystem of tools like Terraform, Consul, and Vault for provisioning, service discovery, and secrets management.
- It can support a cluster size of more than ten thousand nodes.

4장. 컨테이너 3. Orchestration

□ **Kubernetes Hosted Solutions**

- **Google Kubernetes Engine:** Offers managed Kubernetes clusters on Google Cloud Platform.
- **Amazon Elastic Container Service for Kubernetes (Amazon EKS):** Offers a managed Kubernetes service on AWS.
- **Azure Kubernetes Service (AKS):** Offers a managed Kubernetes clusters Microsoft Azure.
- **Stackpoint.io:** Provides Kubernetes infrastructure automation and management for multiple public clouds.
- **OpenShift Dedicated:** Offers managed Kubernetes clusters powered by Red Hat

4장. 컨테이너 3. Orchestration

□ **Google Kubernetes Engine (GKE)**

- **Google Kubernetes Engine is a fully-managed solution for running Kubernetes on Google Cloud. As we have learned earlier, Kubernetes is used for automating deployment, operations, and scaling of containerized applications.**
- **In GKE, Kubernetes can be integrated with all Google Cloud Platform's services, like Stackdriver monitoring, diagnostics, and logging, identity and access management, etc**

4장. 컨테이너 3. Orchestration

□ **GKE Features and Benefits**

- **It has all of Kubernetes' features.**
- **It runs on a container-optimized OS built and managed by Google.**
- **It is a fully-managed service, so the users do not have to worry about managing and scaling the cluster.**
- **We can store images privately, using the private container registry.**
- **Logging can be enabled easily using Google Cloud Logging.**
- **It supports Hybrid Networking to reserve an IP address range for the container cluster.**
- **It enables a fast setup of managed clusters.**
- **It facilitates increased productivity for Dev and Ops teams.**
- **It is Highly Available in multiple zones and SLA promises 99.5%.**
- **It has Google-grade managed infrastructure.**
- **It can be seamlessly integrated with all GCP services.**
- **It provides a feature called Auto Repair, which initiates a repair process for unhealthy nodes**

4장. 컨테이너 3. Orchestration

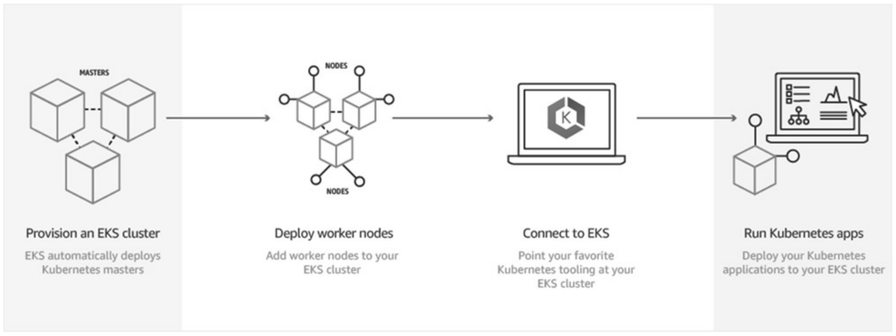
□ **Amazon Elastic Container Service for Kubernetes (EKS)**

- **With Amazon EKS, users don't need to worry about the infrastructure management, deployment and maintenance of the Kubernetes control plane. EKS provides a scalable and highly-available control plane that runs across multiple AWS availability zones. It can automatically detect the unhealthy Kubernetes control plane nodes and replace them.**

- **EKS also supports cluster autoscaling, using which it can dynamically add worker nodes, based on the workload. It also integrates with Kubernetes RBAC (Role-Based Control Access) to support AWS IAM authentication.**

4장. 컨테이너 3. Orchestration

- Amazon Elastic Container Service for Kubernetes (EKS)


MASTERS
NODES
NODES
Provision an EKS cluster
EKS automatically deploys Kubernetes masters
Deploy worker nodes
Add worker nodes to your EKS cluster
Connect to EKS
Point your favorite Kubernetes tooling at your EKS cluster
Run Kubernetes apps
Deploy your Kubernetes applications to your EKS cluster

Amazon Elastic Container Service for Kubernetes (retrieved from docs.aws.amazon.com)

JS Lab

151

4장. 컨테이너 3. Orchestration

- **EKS Features and Benefits**
  - **No need to manage the Kubernetes Control Plane.**
  - **Provides secure communication between the worker nodes and the control plane.**
  - **Supports auto scaling in response to changes in load.**
  - **Well integrated with various AWS services, like IAM and CloudTrail.**
  - **Certified hosted Kubernetes platform.**

JS Lab

152

4장. 컨테이너 3. Orchestration

Azure Kubernetes Service (AKS)

- Azure Kubernetes Service is a hosted Kubernetes service offered by Microsoft Azure.
- AKS offers a fully-managed Kubernetes container orchestration service, which reduces the complexity and operational overhead of managing Kubernetes. AKS handles all of the cluster management tasks, health monitoring, upgrades, scaling, etc.
- AKS also supports cluster autoscaling using which it can dynamically add worker nodes, based on the workload. It supports Kubernetes RBAC (Role-Based Control Access) and can integrate with Azure Active Directory for identity and security management.
- With AKS, users just need to pay for agent/workers nodes that get deployed. Master nodes are managed by AKS for free

4장. 컨테이너 3. Orchestration

AKS Features and Benefits

- No need to manage the Kubernetes Control Plane.
- Supports GUI and CLI-based deployment.
- Integrates well with other Azure services.
- Certified hosted Kubernetes platform.
- Compliant with SOC and ISO/HIPAA/HITRUST

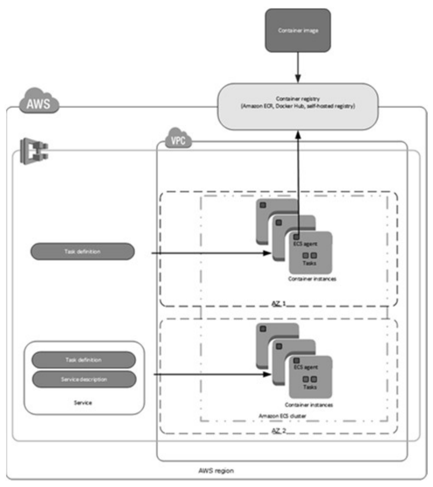
4장. 컨테이너 3. Orchestration

- **Amazon ECS Amazon:** Elastic Container Service (Amazon ECS) is part of the Amazon Web Services (AWS) offerings. It provides a fast and highly scalable container management service that makes it easy to run, stop, and manage Docker containers on a cluster.
  - **Fargate Launch Type:** AWS Fargate allows us to run containers without managing servers and clusters. In this mode, we just have to package our applications in containers along with CPU, memory, networking and IAM policies. We don't have to provision, configure, and scale clusters of virtual machines to run containers, as AWS will take care of it for us.
  - **EC2 Launch Type:** With the EC2 launch type, we can provision, patch, and scale the ECS cluster. This gives more control to our servers and provides a range of customization options.

155

4장. 컨테이너 3. Orchestration

- **EC2 Launch Type**



EC2 Launch Type (retrieved from docs.aws.amazon.com)

156

4장. 컨테이너 3. Orchestration

Amazon ECS Components (1 of 3)

- **Cluster:** It is a logical grouping of tasks or services. With the EC2 launch type, a cluster is also a grouping of container instances.
- **Container Instance**: It is only applicable if we use the EC2 launch type. We define the Amazon EC2 instance to become part of the ECS cluster and to run the container workload.
- **Container Agent:** It is only applicable if we use the Fargate launch type. It allows container instances to connect to your cluster.
- **Scheduler:** It places tasks on the cluster.

JS Lab

157

4장. 컨테이너 3. Orchestration

Amazon ECS Components (2 of 3)

- **Task Definition:** It specifies the blueprint of an application, which consists of one or more containers. Below, you can see an example of a sample task definition file (from docs.aws.amazon.com)

```
{
"containerDefinitions": [
  {
    "name": "wordpress",
    "links": [
      "mysql"
    ],
    "image": "wordpress",
    "essential": true,
    "portMappings": [
      {
        "containerPort": 80,
        "hostPort": 80
      }
    ],
    "memory": 500,
    "cpu": 10
  },
  {
    "environment": [
      {
        "name": "MYSQL_ROOT_PASSWORD",
        "value": "password"
      }
    ],
    "name": "mysql",
    "image": "mysql",
    "cpu": 10,
    "memory": 500,
    "essential": true
  }
],
"family": "hello_world"
}
```

JS Lab

158

4장. 컨테이너 3. Orchestration

□ Amazon ECS Components (3 of 3)

- **Service:** It allows one or more instances of tasks to run, depending on the task definition. Below you can see the template of a service definition (from docs.aws.amazon.com). If there is an unhealthy task, then the service restarts it. One load balancer (ELB) is attached to each service.
- **Task:** It is a running container instance from the task definition.
- **Container:** It is a Docker container created from the task definition.

```
{
   "cluster": "",
   "serviceName": "",
   "taskDefinition": "",
   "loadBalancers": [
      {
         "loadBalancerName": "",
         "containerName": "",
         "containerPort": 0
      }
   ],
   "desiredCount": 0,
   "clientToken": "",
   "role": "",
   "deploymentConfiguration": {
      "maximumPercent": 200,
      "minimumHealthyPercent": 100
   }
}
```

JS Lab

159

4장. 컨테이너 3. Orchestration

□ Amazon ECS Features (1 of 2)

- **It is compatible with Docker.**
- **It provides a managed cluster, so that users do not have to worry about managing and scaling the cluster.**
- **The task definition allows the user to define the applications through a .json file. Shared data volumes, as well as resource constraints for memory and CPU, can also be defined in the same file.**
- **It provides APIs to manage clusters, tasks, etc.**
- **It allows easy updates of containers to new versions.**
- **The monitoring feature is available through AWS CloudWatch.**
- **The logging facility is available through AWS CloudTrail.**
- **It supports third party Docker Registry or Docker Hub.**

JS Lab

160

4장. 컨테이너 3. Orchestration

Amazon ECS Features (2 of 2)

- AWS Fargate allows you to run and manage containers without having to provision or manage servers.
- AWS ECS allows you to build all types of containers. You can build a long-running service or a batch service in a container and run it on ECS.
- You can apply your Amazon Virtual Private Cloud (VPC), security groups and AWS Identity and Access Management (IAM) roles to the containers, which helps maintain a secure environment.
- You can run containers across multiple availability zones within regions to maintain High Availability.
- ECS can be integrated with AWS services like Elastic Load Balancing, Amazon VPC, AWS IAM, Amazon ECR, AWS Batch, Amazon CloudWatch, AWS CloudFormation, AWS CodeStar, AWS CloudTrail, and more.

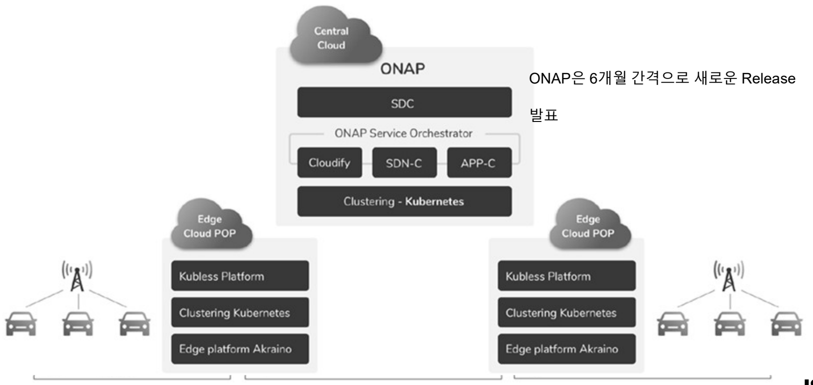
4장. 컨테이너 3. Orchestration

Benefits of Using Amazon ECS

- It provides a managed cluster.
- It is built on top of Amazon EC2.
- It is highly available and scalable.
- It leverages other AWS services, such as CloudWatch Metrics.
- We can manage it using CLI, Dashboard or APIs

4장. 컨테이너 3. Orchestration

- Telco Network Slicing을 위한 오케스트레이션/모니터링
- ONAP 프로젝트의 SDC/SO (예)
  - Kubeless (Kubernetes-native serverless framework)
  - Kata Containers (초경량 VM을 위한 open source project)



ONAP은 6개월 간격으로 새로운 Release 발표

ONAP (Open Network Automation Platform) Service Design and Creation (SDC) Service Orchestrator (SO)

JS Lab

james@jslab.kr

163

4장. 컨테이너 3. Orchestration

- 에지 환경등 데이터센터 밖의 Kubernetes 지원 요구 수용
- Open source project ‘K3s’ (예)
  - K8s 기본 기능을 유지하며 Production을 위해 300만 라인 정도 삭제한 바이너리 40 MB로 메모리 512 MB 사용 수준
  - 알파기능, 클라우드, 스토리지 등의 많은 기능 제거
  - 인텔과 ARM 계열 하드웨어 지원 (x86_64, ARM64, and ARMv7)
  - 6개월 내에 HA 지원 예정
  - Docker는 선택으로 삭제 (containerd 추가)
  - 단일 프로세스에 총합하는 K8s master, Kubelet, containerd
  - SQLite 를 etcd에 추가

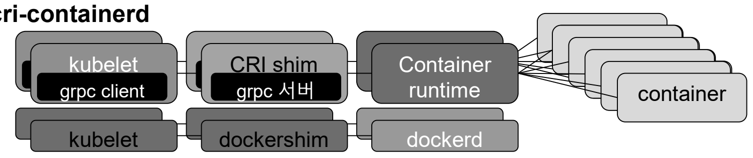
JS Lab

james@jslab.kr

164

4장. 컨테이너 3. Orchestration

- **CRI (Container Runtime Interface):** Kubernetes(K8s)는 특정 런타임과 분리 추상화하며 K8s 1.6(2017년 3월) kublet에서 CRI를 정식 적용
- **CRI 호환 런타임 프로젝트**(배포범위 확대: VM/Hypervisor/베어메탈)
  - **Docker** (CRI shim 라이브러리 사용)
  - **dockershim**
  - **Rkt** (CoreOS에서 파생)
  - **cri-o** (도커와 같이 OCI 호환하는 OCI confirmed 런타임, K8s 프로젝트)
  - **frakti** (하이퍼바이저 용 런타임)
  - **rktlet** (rkt 컨테이너 런타임)
  - **virtlet VM**(QCOW) 런타임
  - **cri-containerd**

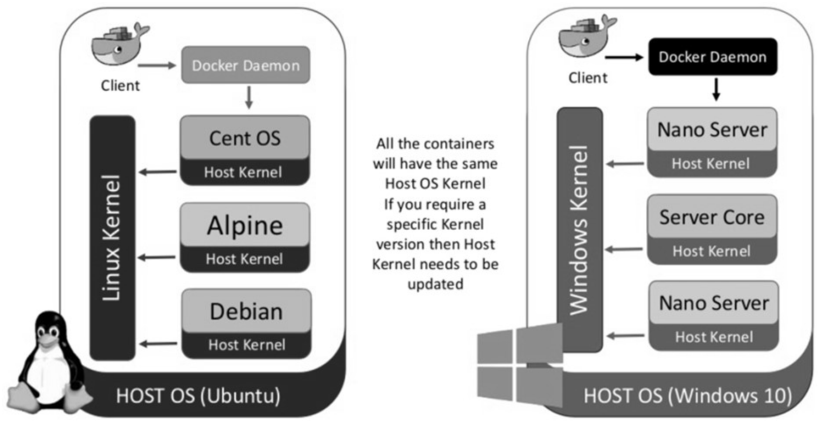


JS Lab

165

4장. 컨테이너 3. Orchestration

- **Linux Kernel**
- **Windows Kernel**

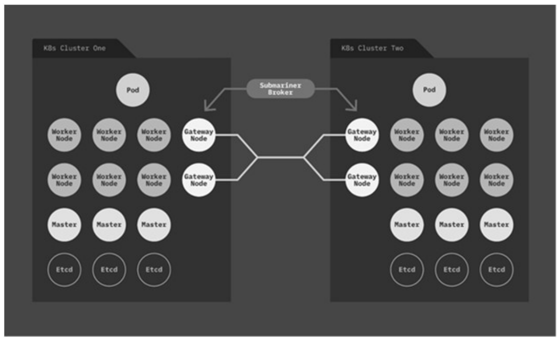


JS Lab

166

4장. 컨테이너 3. Orchestration

- K8s Cluster간 네트워크 연결 요구
- Open source project 'Submariner' (예)
  - 카산드라(Cassandra)와 같은 HA 데이터베이스의 지역 분산
  - 분산화 트레이스 (Distributed Tracing)
  - Service Mesh의 클러스터들 간에 확대



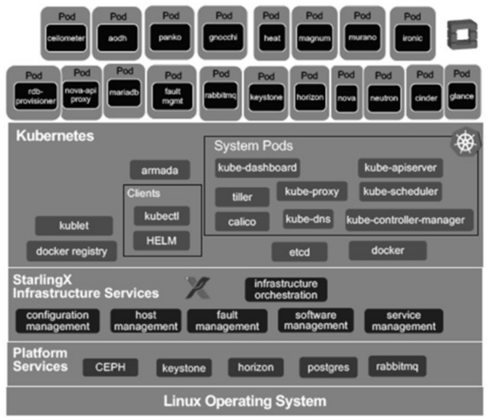
https://submariner.io/

167

4장. 컨테이너 3. Orchestration

- 에지 클라우드 플랫폼
- 필요시 VM 지원 가능한 컨테이너 아키텍처
- StarlingX(예): OpenStack Foundation

STARLINGX



https://www.starlingx.io/

168

4장. 컨테이너 4. Unikernels

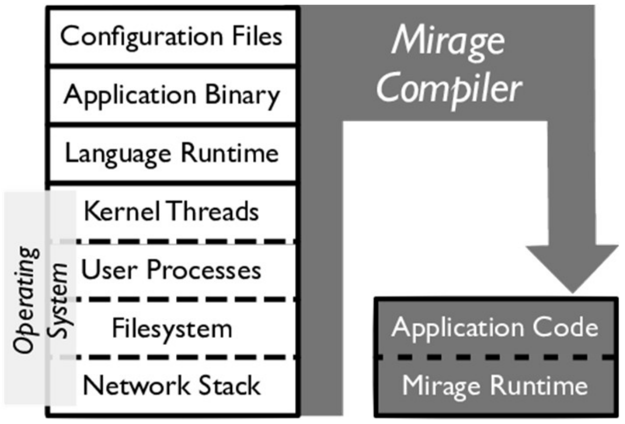
- "Unikernels are specialized, single-address-space machine images constructed by using library operating systems“
- With unikernels, we can also select the part of the kernel needed to run with the specific application. With unikernels, we can create a single address space executable, which has both application and kernel components. The image can be deployed on VMs or bare metal, based on the unikernel's type.

http://unikernel.org/

169

4장. 컨테이너 4. Unikernels

- Creating Specialized VM Images



Comparison of a Traditional OS Stack and a MirageOS Unikernel (by AmirMC/CC BY-SA 3.0, retrieved from Wikipedia)

170

4장. 컨테이너 4. Unikernels

- Creating Specialized VM Images
  - The application code
  - The configuration files of the application
  - The user-space libraries needed by the application
  - The application runtime (like JVM)
  - The system libraries of the unikernel, which allow back and forth communication with the hypervisor.

4장. 컨테이너 4. Unikernels

- Benefits of Unikernels
  - A minimalistic VM image to run an application, which allows us to have more applications per host.
  - A faster boot time.
  - Efficient resource utilization.
  - A simplified development and management model.
  - A more secured application than the traditional VM, as the attack surface is reduced.
  - An easily-reproducible VM environment, which can be managed through a source control system like Git.

4장. 컨테이너 4. Unikernels

◻ Unikernel Implementations

- **Specialized and purpose-built unikernels:** They utilize all the modern features of software and hardware, without worrying about the backward compatibility. They are not POSIX-compliant. Some examples of specialized and purpose-built unikernels are ING, HalVM, MirageOS, and Clive.
- **Generalized 'fat' unikernels:** They run unmodified applications, which make them fat. Some examples of generalized 'fat' unikernels are BSD Rump kernels, OSv, Drawbridge, etc.

173

4장. 컨테이너 4. Unikernels

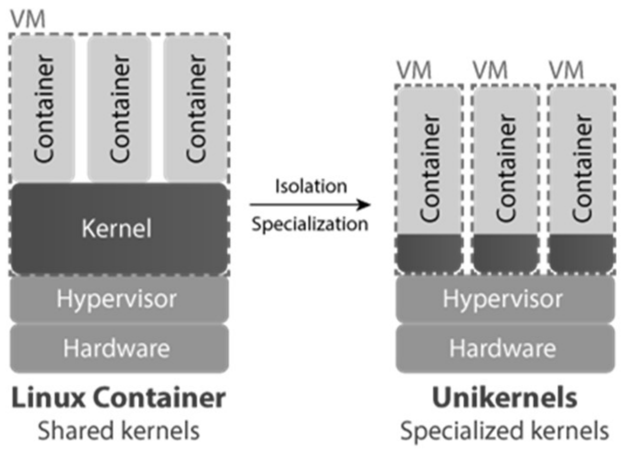
◻ Unikernels and Docker (MirageOS)

- **In January of 2016, Docker bought Unikernels to make them first-class citizen of the Docker ecosystem. Both containers and unikernels can co-exist on the same host. They can be managed by the same Docker binary.**
- **Unikernels helped Docker to run the Docker Engine on top of Alpine Linux on Mac and Windows with their default hypervisors, which are xhyve Virtual Machine and Hyper-V VM respectively.**

174

4장. 컨테이너 4. Unikernels

◻ **Shared Kerner vs. Unikernel**



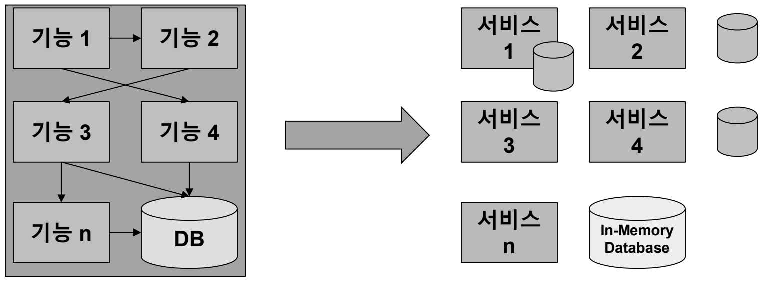
james@jslab.kr

JS Lab

175

4장. 컨테이너 5. Microservices

◻ **마이크로서비스 전환 시 추가 기능 고려**
- 비즈니스 기능을 위한 서비스 연결
- 스탠드얼론 and/or 서비스의 부분 적용
- 비동기 통신
- 성능 개선을 위한 서비스 요소 교체 (프로그램 언어, 데이터베이스 등)
- 일정하지 않은 확장 요구 CI/CD
- 각 엔지니어링 팀은 비즈니스 영역의 이해하고 책임을 소유



james@jslab.kr

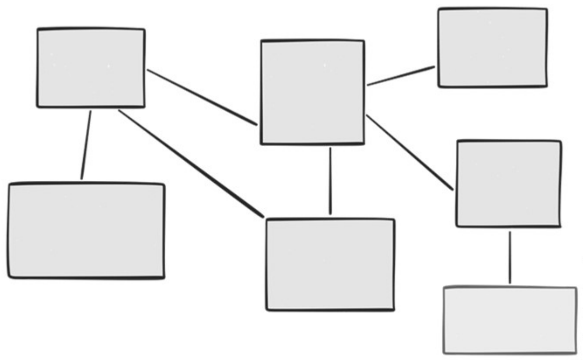
JS Lab

CI/CD (Continuous integration and Continuous delivery)

176

4장. 컨테이너 5. Microservices

- Evolutionary design



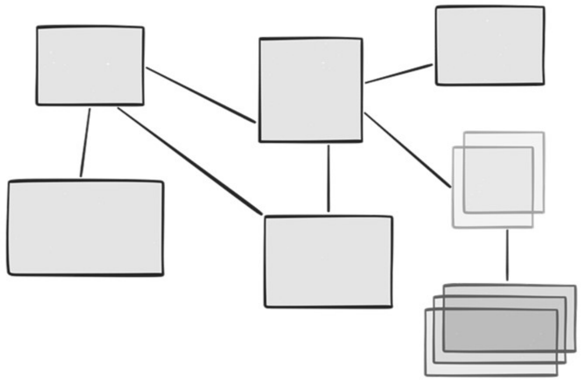
- Remove
- Add
- Replace
- Experimental microservice
- Grow at “no” cost

JS Lab

177

4장. 컨테이너 5. Microservices

- Selective scalability



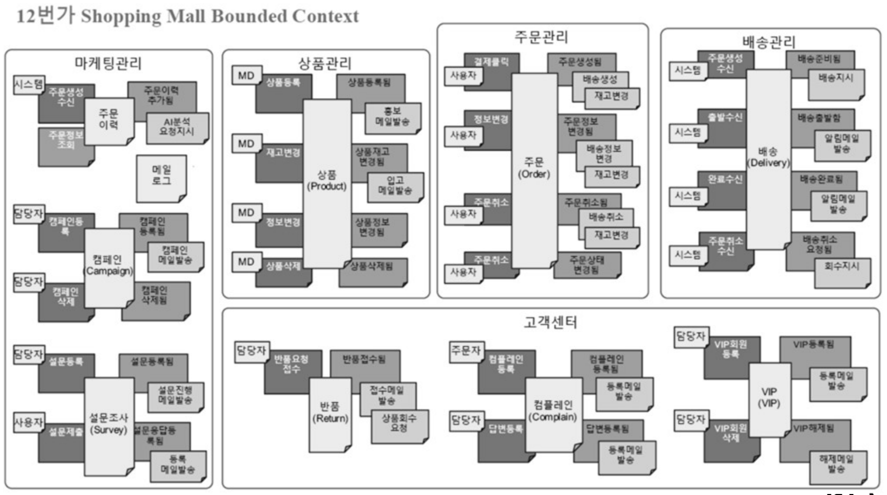
Nb users > 500

JS Lab

178

4장. 컨테이너 5. Microservices

◻ Shopping Mall Bounded Context (예)



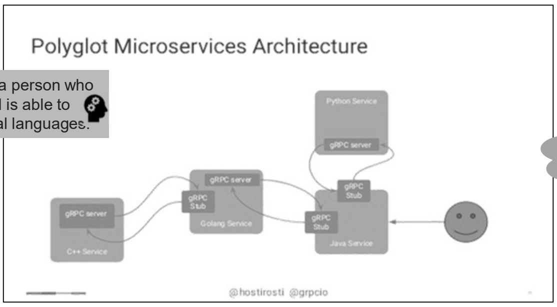
JS Lab

179

4장. 컨테이너 5. Microservices

◻ gRPC 사용 마이크로서비스

- gRPC defines relationship between client and server and enforces strict rules of communication between them. (In REST calls, the request and response are totally de-coupled)
- gRPC supports useful additions like standard error responses and meta data.



polyglot: a person who knows and is able to use several languages.

API 엔드포인트 버전닝 회피

RPC(Remote Procedure Calls)는 네트워크 상에서 Local Procedure Call과 같이 보이도록 동작

https://medium.com/@akshitjain_74512/inter-service-communication-with-grpc-d815a561e3a1

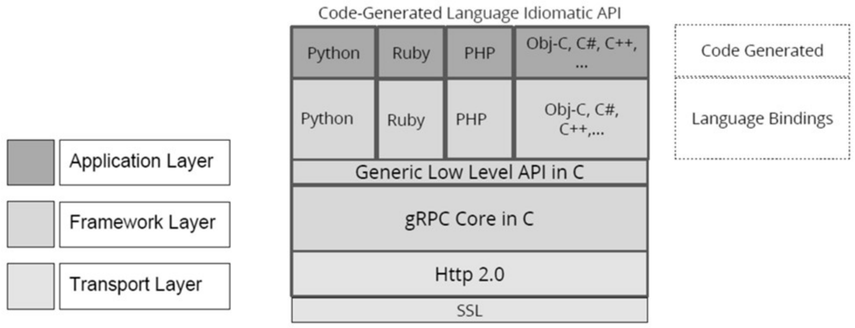
JS Lab

180

4장. 컨테이너 5. Microservices

◻ gRPC 사용 마이크로서비스

- Architecture: Derived Stack for gRPC



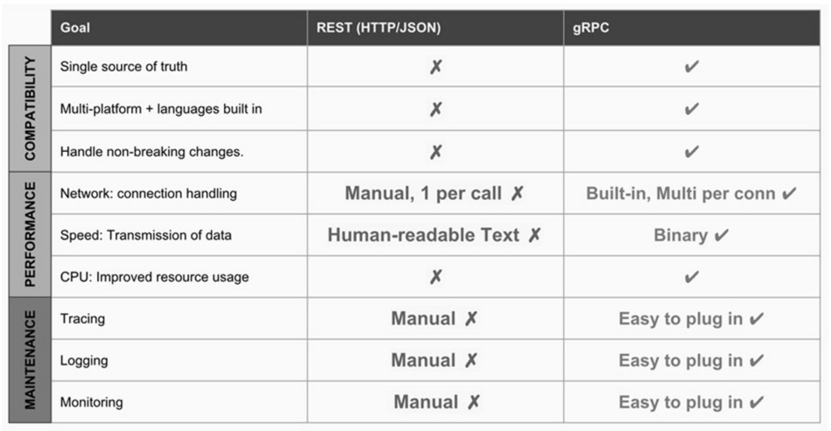
JS Lab

181

4장. 컨테이너 5. Microservices

◻ gRPC 사용 마이크로서비스

- gRPC is faster than REST. gRPC uses HTTP2 by default



	Goal	REST (HTTP/JSON)	gRPC
COMPATIBILITY	Single source of truth	✗	✔
	Multi-platform + languages built in	✗	✔
	Handle non-breaking changes.	✗	✔
PERFORMANCE	Network: connection handling	Manual, 1 per call ✗	Built-in, Multi per conn ✔
	Speed: Transmission of data	Human-readable Text ✗	Binary ✔
	CPU: Improved resource usage	✗	✔
MAINTENANCE	Tracing	Manual ✗	Easy to plug in ✔
	Logging	Manual ✗	Easy to plug in ✔
	Monitoring	Manual ✗	Easy to plug in ✔

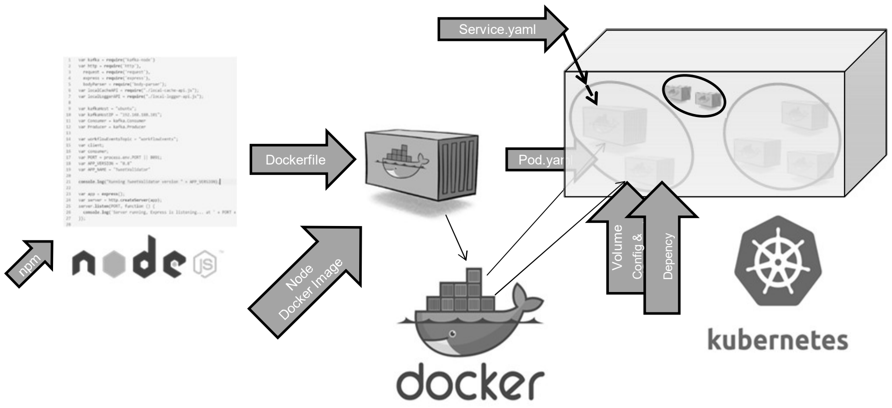
https://devopedia.org/grpc?ref=codebldr

JS Lab

182

4장. 컨테이너 5. Microservices

- **The making of a microservice**
- **인프라 추상화:** 하드웨어, 네트워크, Build, 파이프라인, 서비스 디스커버리, 로드밸런싱



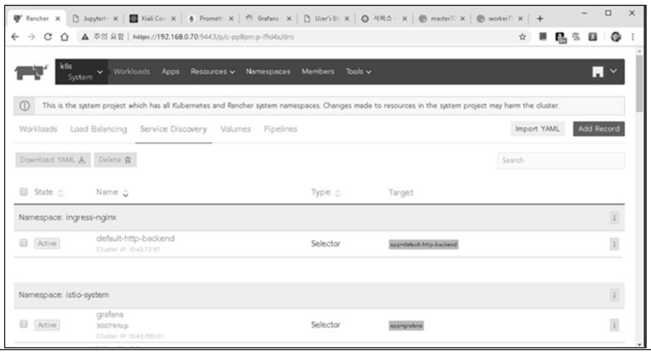
```
var kafka = require('kafka-node')
var http = require('http'),
  request = require('request'),
  express = require('express'),
  bodyParser = require('body-parser');
var localCacheAPI = require("./local-cache-api.js");
var localLoggerAPI = require("./local-logger-api.js");

var kafkaHost = "ubuntu";
var kafkaHostIP = "192.168.188.101";
var Consumer = kafka.Consumer
var Producer = kafka.Producer

var workflowEventsTopic = "workflowEvents";
var client;
var consumer;
var PORT = process.env.PORT || 8091;
var APP_VERSION = "0.8"
var APP_NAME = "TweetValidator"

console.log("Running TweetValidator version " + APP_VERSION);

var app = express();
var server = http.createServer(app);
server.listen(PORT, function () {
  console.log('Server running, Express is listening... at ' + PORT +
});
```

JS Lab

183

4장. 컨테이너 5. Microservices

- **Discovery Service**
  - **Client는 기존에는 1개의 monolithic app을 하였으나 MSA에서는 동시에 여러 개의 서비스를 호출해야 함**
  - **Client는 서비스들의 위치를 알아야 함 (Host/IP/Port 등)**
  - **Client가 마이크로서비스 호출시 제공하는 entry point 실시간 위치를 하드코드(Hard code)보다는 유연한 방법으로 현재의 위치를 제공해야 함**



This is the system project which has all Kubernetes and Rancher system namespaces. Changes made to resources in the system project may harm the cluster.

State	Name	Type	Target
Namespace: ingress-nginx			
Active	default-http-backend Cluster IP: 10.43.72.97	Selector	app=default-http-backend
Namespace: istio-system			
Active	grafana 30079/tcp Cluster IP: 10.43.156.131	Selector	app=grafana

JS Lab

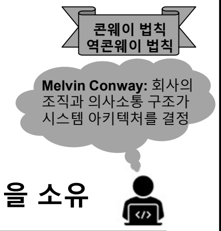
184

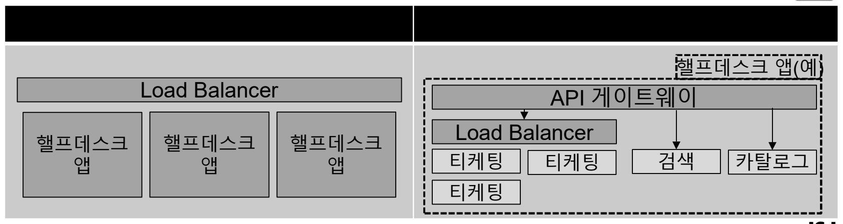
4장. 컨테이너

5. Microservices

□ 조직의 Learning Curve 고려
- ■ 단일 마이크로서비스 (Standalone Microservice) 많이 필요시
- ■ 과도한 의존성으로 시간이 많이 필요하고 코드의 품질이 낮아질 때
- ■ 한가지 요소로 애플리케이션 장애 시

□ 마이크로서비스 전환 시 추가 기능 고려
- ■ 비즈니스 기능을 위한 서비스 연결
- ■ 스탠드얼론 and/or 서비스의 부분 적용
- ■ 각 엔지니어링 팀은 비즈니스 영역의 이해하고 책임을 소유


콘웨이 법칙
역콘웨이 법칙
Melvin Conway: 회사의 조직과 의사소통 구조가 시스템 아키텍처를 결정



james@jslab.kr

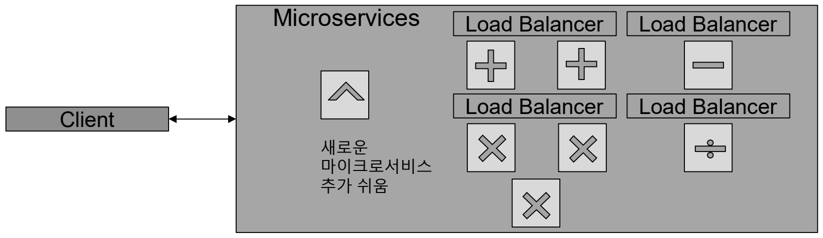
JS Lab

185

4장. 컨테이너

5. Microservices

□ 마이크로서비스의 장점
- ■ 단순성(Simplicity)
- ■ 확장성(Scalability)
- ■ 적용(Continuous delivery)
- ■ 낮은 의존성과 더 많은 자유
- ■ 장애 확인
- ■ 데이터 분리와 분산화
- ■ 다양한 선택



james@jslab.kr

JS Lab

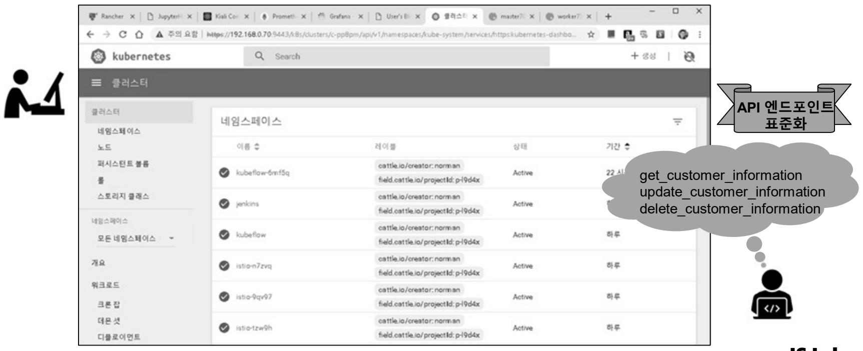
186

4장. 컨테이너

5. Microservices

□ API Gateway

- 모든 호출에 대한 1개의 entry point가 필요
- 내부의 복잡성을 숨김
- 마이크로 서비스 변화/결합/구분/추가/제거 유연함
- Client와 Application사이의 왕복 단순화로 효율성 증대



네임스페이스

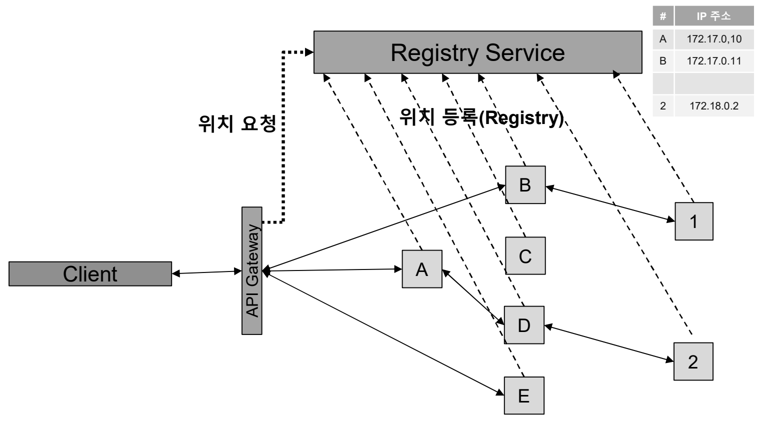
	이름	레이블	상태	기간
✓	kubeflow-6mf5q	cattle.io/creator: norman field.cattle.io/projectId: p-l9d4x	Active	22 시
✓	jenkins	cattle.io/creator: norman field.cattle.io/projectId: p-l9d4x	Active	
✓	kubeflow	cattle.io/creator: norman field.cattle.io/projectId: p-l9d4x	Active	하루
✓	istio-n7zvq	cattle.io/creator: norman field.cattle.io/projectId: p-l9d4x	Active	하루
✓	istio-9qv97	cattle.io/creator: norman field.cattle.io/projectId: p-l9d4x	Active	하루
✓	istio-tzw9h	cattle.io/creator: norman field.cattle.io/projectId: p-l9d4x	Active	하루

4장. 컨테이너

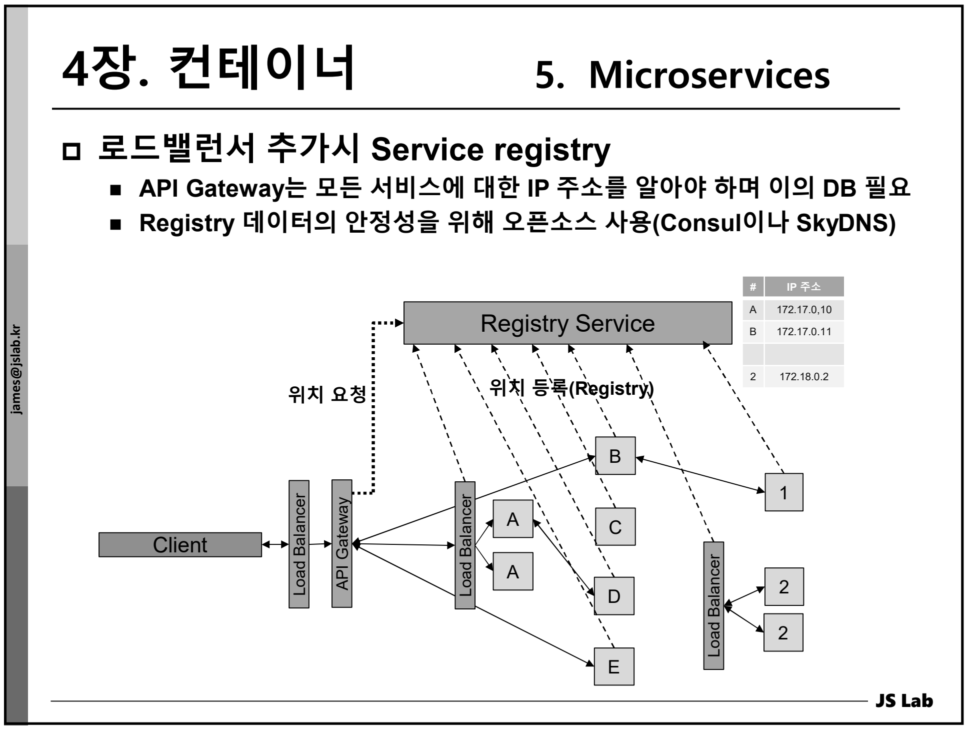
5. Microservices

□ Service registry

- API Gateway는 모든 서비스에 대한 IP 주소를 알고 이의 DB 필요
- Registry 데이터를 위해 오픈소스 사용 (Consul이나 SkyDNS)



#	IP 주소
A	172.17.0,10
B	172.17.0.11
2	172.18.0.2

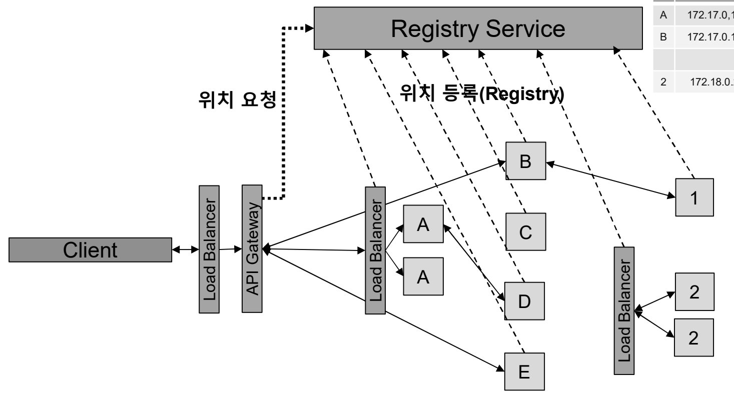


4장. 컨테이너 5. Microservices

□ 로드밸런서 추가시 Service registry

- API Gateway는 모든 서비스에 대한 IP 주소를 알아야 하며 이의 DB 필요
- Registry 데이터의 안정성을 위해 오픈소스 사용(Consul이나 SkyDNS)

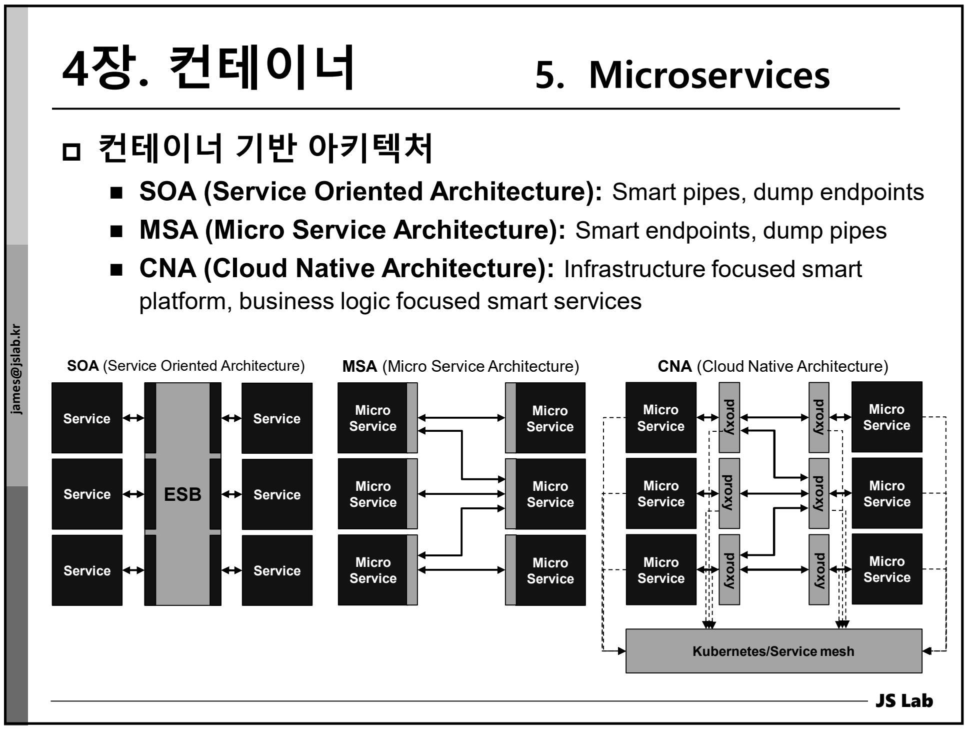
#	IP 주소
A	172.17.0,10
B	172.17.0.11
2	172.18.0.2



james@jslab.kr

JS Lab

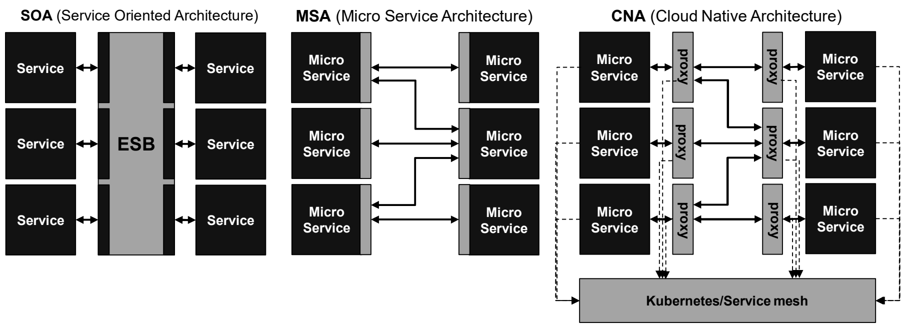
189



4장. 컨테이너 5. Microservices

□ 컨테이너 기반 아키텍처

- **SOA (Service Oriented Architecture):** Smart pipes, dump endpoints
- **MSA (Micro Service Architecture):** Smart endpoints, dump pipes
- **CNA (Cloud Native Architecture):** Infrastructure focused smart platform, business logic focused smart services



james@jslab.kr

JS Lab

190

4장. 컨테이너 5. Microservices

- "Microservices are small, independent processes that communicate with each other to form complex applications which utilize language-agnostic APIs. These services are small building blocks, highly decoupled and focused on doing a small task, facilitating a modular approach to system-building. The microservices architectural style is becoming the standard for building continuously deployed systems".

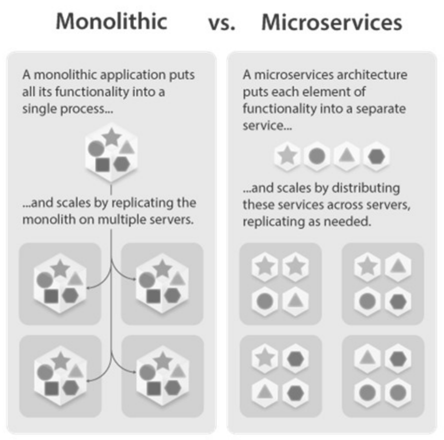
4장. 컨테이너 5. Microservices

- The Technological Advancement towards Microservices
  - With the launch of Amazon Web Services (AWS) in 2006, we can get compute resources on demand from the web or the command-line interface.
  - With the launch of Heroku in 2007, we can deploy a locally-built application in the cloud with just a couple of commands.
  - With the launch of Vagrant in 2010, we can easily create reproducible development environments.

4장. 컨테이너

5. Microservices

◻ Monolithic vs Microservices


Monolithic vs. Microservices
A monolithic application puts all its functionality into a single process...
...and scales by replicating the monolith on multiple servers.
A microservices architecture puts each element of functionality into a separate service...
...and scales by distributing these services across servers, replicating as needed.

JS Lab

193

4장. 컨테이너

5. Microservices

◻ Monolithic vs Microservices

	Monolithic Services	Micro Services
배포/빌드/구동	• 기본적으로 규모가 커질수록 빌드 및 배포, 서버 기동 시간이 오래 걸림	• DevOps의 자동화로, 독립적 스케줄링 가능 • 빠른 배포 가능
개발 독립성	• 한 두사람의 실수가 전체 서비스에 영향	• 개발자/팀간 의존성이 줄어 듬 • 해당 서비스에 대한 책임/권한 상승
소스관리/유지보수	• 전체 코드에 대한 이해가 동반되어야 함 • 기본적인 유지보수에 대한 부담 • 새로운 멤버가 빠르게 이해/적응이 쉽지 않음	• Block 단위의 요구사항 분석 및 처리가 가능 • 쉽고 단순함
확장성/유연성	• 요구사항의 규모에 관계없이 초기 정해진 언어 및 개발방식으로 진행 • 확장성 및 유연성이 떨어짐	• 부하가 많은 특정서비스 한정- 물리적인 확장이 가능 • 요구사항에 최적화된 언어/아키텍쳐/데이터 저장소 사용 가능
개발생산성	• 선택된 언어의 특성에 의존	• 효율적이고 생산성 높은 언어의 선택이 가능 • 전체적인 개발생산성 증대

https://m.blog.naver.com/PostView.nhn?blogId=tmondev&logNo=220750045191&proxyReferer=https%3A%2F%2Fwww.google.com%2F

JS Lab

194

4장. 컨테이너 5. Microservices

How to Break a Monolith into Microservices

- If you have a complex monolith application, then it is not advisable to rewrite the entire application from scratch. Instead, you should start carving out services from the monolith, which implement the desired functionalities for the code we take out from the monolith. Over time, all or most functionalities will be implemented in the microservices architecture.
- We can split the monoliths based on the business logic, front-end (presentation), and data access. In the microservices architecture it is recommended to have a local database for individual services. And, if the services need to access the database from other services, then we can implement an event-driven communication between these services.
- As mentioned earlier, we can split the monolith based on the modules of the monolith application and each time we do it, our monolith shrinks

4장. 컨테이너 5. Microservices

Benefits of Microservices

- There is no language or technology lock-in. As each service works independently, we can choose any language or technology to develop it. We just need to make sure its API endpoints return the expected output.
- Each service in a microservice can be deployed independently.
- We do not have to take an entire application down just to update or scale a component. Each service can be updated or scaled independently. This gives us the ability to respond faster.
- If one service fails, then its failure does not have a cascading effect. This helps in debugging as well.
- Once the code of a service is written, it can be used in other projects, where the same functionality is needed.
- The microservice architecture enables continuous delivery.
- Components can be deployed across multiple servers or even multiple data centers.
- They work very well with container orchestration tools like Kubernetes, DC/OS and Docker Swarm.

4장. 컨테이너 5. Microservices

Challenges and Drawbacks of Microservices (1 of 2)

- **Choosing the right service size:** While breaking the monolith application or creating microservices from scratch, it is very important to choose the right functionality for a service. For example, if we create a microservice for each function of a monolith, then we would end up with lots of small services, which would bring unnecessary complexity.
- **Deployment:** We can easily deploy a monolith application. However, to deploy a microservice, we need to use a distributed environment such as Kubernetes.
- **Testing:** With lots of services and their inter-dependency, sometimes it becomes challenging to do end-to-end testing of a microservice.

4장. 컨테이너 5. Microservices

Challenges and Drawbacks of Microservices (2 of 2)

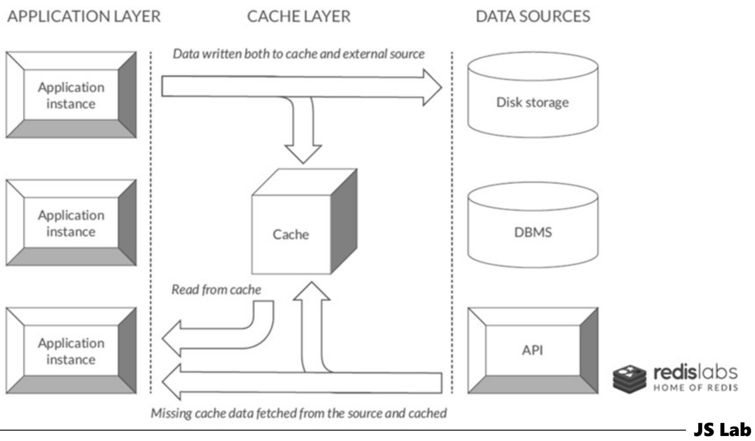
- **Inter-service communication:** Inter-service communication can be very costly if it is not implemented correctly. There are options such as message passing, RPC, etc., and we need to choose the one that fits our requirement and has the least overhead.
- **Managing databases:** When it comes to the microservices' architecture, we may decide to implement a database local to a microservice. But, to close a business loop, we might require changes on other related databases. This can create problems (e.g. partitioned databases).
- **Monitoring:** Monitoring individual services in a microservices environment can be challenging. This challenge is being addressed, and a new set of tools, like Sysdig or Datadog, is being developed to monitor and debug microservices.

4장. 컨테이너

5. Microservices

□ **Application Cache**

- **Faster intermediate data storage is called the application's cache**



JS Lab

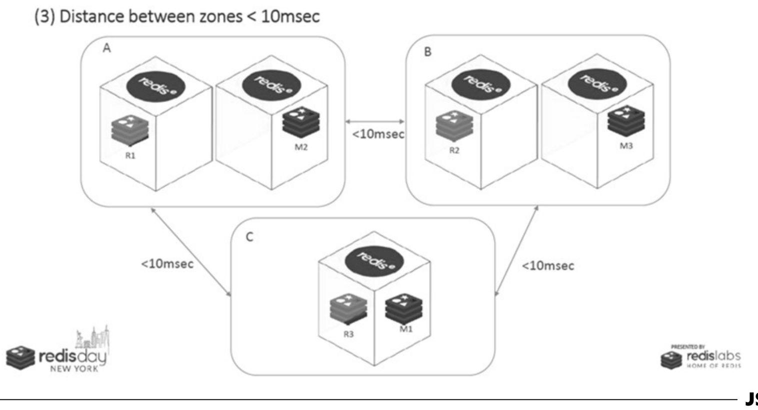
199

4장. 컨테이너

5. Microservices

□ **HA Concept @ Application Cache (예: Radis)**

- **Deploy a multi-az/rack cluster**
- **Master and replica of each shard should be deployed on a different zone**

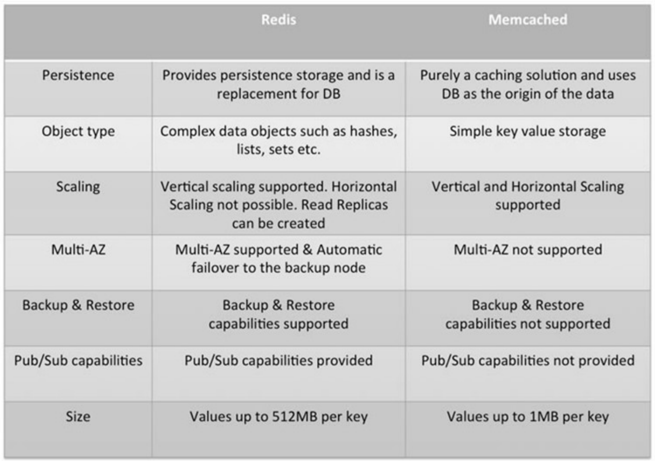


JS Lab

200

4장. 컨테이너 5. Microservices

□ **Radis vs. Memcached @ Application Cache**



	Redis	Memcached
Persistence	Provides persistence storage and is a replacement for DB	Purely a caching solution and uses DB as the origin of the data
Object type	Complex data objects such as hashes, lists, sets etc.	Simple key value storage
Scaling	Vertical scaling supported. Horizontal Scaling not possible. Read Replicas can be created	Vertical and Horizontal Scaling supported
Multi-AZ	Multi-AZ supported & Automatic failover to the backup node	Multi-AZ not supported
Backup & Restore	Backup & Restore capabilities supported	Backup & Restore capabilities not supported
Pub/Sub capabilities	Pub/Sub capabilities provided	Pub/Sub capabilities not provided
Size	Values up to 512MB per key	Values up to 1MB per key

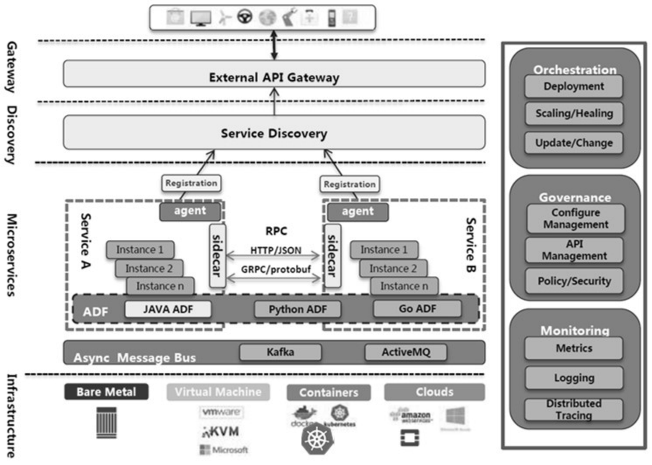
JS Lab

201

4장. 컨테이너 5. Microservices

□ **Telco 5G 서비스는 마이크로 서비스 아키텍처 도입 중**

□ **OMSA** - ONAP Microservice Architecture (예)



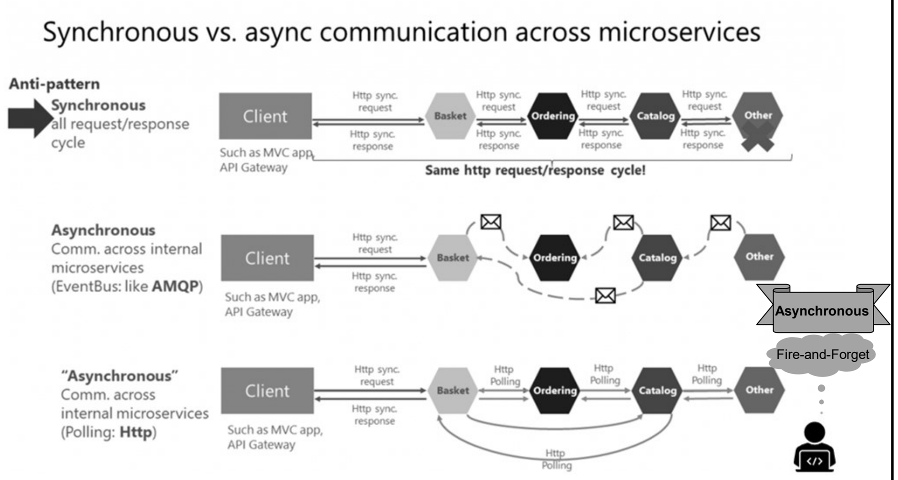
Note: this diagram is a functional view of OMSA, which is not mapped to specific projects

JS Lab

202

4장. 컨테이너 5. Microservices

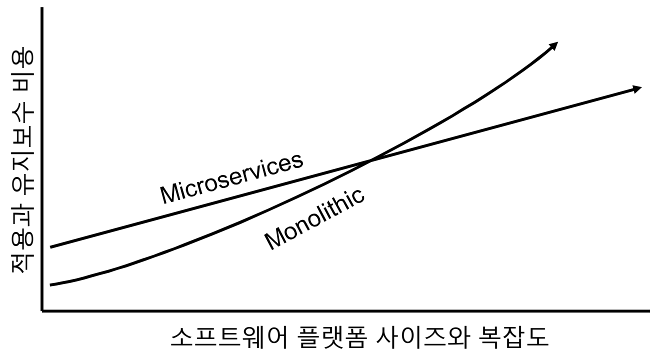
□ **Loosely coupled applications**


Synchronous vs. async communication across microservices
Anti-pattern
Synchronous
all request/response cycle
Client
Such as MVC app, API Gateway
Http sync. request
Http sync. response
Basket
Ordering
Catalog
Other
Same http request/response cycle!
Asynchronous
Comm. across internal microservices (EventBus: like AMQP)
"Asynchronous"
Comm. across internal microservices (Polling: Http)
Http Polling
Asynchronous
Fire-and-Forget

https://www.emagiz.com/blog/loosely-coupled-applications-in-a-microservice-architecture/

4장. 컨테이너 5. Microservices

□ **비용**



4장. 컨테이너 5. Microservices

□ Pros and Cons

Pros	Cons
1. Robust	1. Adds Complexity
2. Scalable	2. Skillset shortage
3. Testable (Local)	3. Confusion on getting the right size
4. Easy to Change and Replace	4. Team need to manage end-to-end of the Service (From UI to Backend to Running in Production).
5. Easy to Deploy	
6. Technology Agnostic	

Managing cloud sprawl

기술 부채

james@jslab.kr

https://azure.microsoft.com/is-is/resources/your-guide-to-managing-cloud-sprawl-ko-kr/

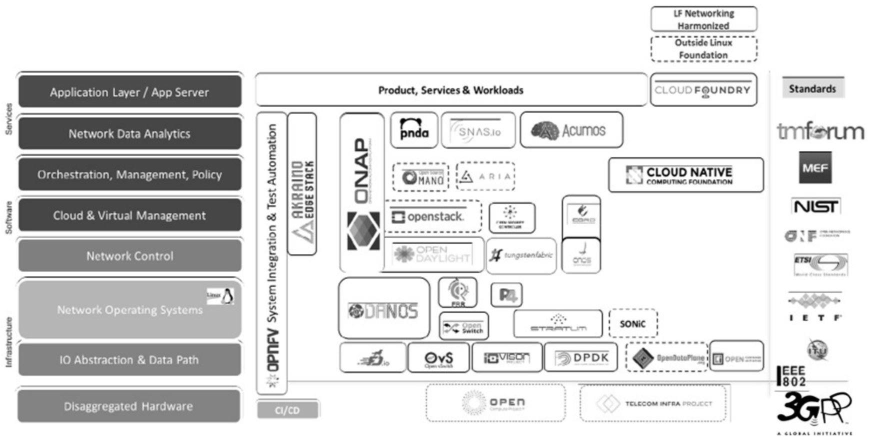
JS Lab

205

4장. 컨테이너 6. SDN

□ 오프소스와 SDN Landscape

□ 리눅스 재단 내/외의 K8s 도입 네트워킹 프로젝트 증가 중



james@jslab.kr

https://www.linuxfoundation.org/projects/networking/

JS Lab

206

4장. 컨테이너 6. SDN

- Software-Defined Networking (SDN) decouples the network control layer from the layer which forwards the traffic. This allows SDN to program the control layer to create custom rules in order to meet the networking requirements.

4장. 컨테이너 6. SDN

- SDN Architecture
  - **Data Plane:** The Data Plane, also called the Forwarding Plane, is responsible for handling data packets and apply actions to them based on rules which we program into lookup-tables.
  - **Control Plane:** The Control Plane is tasked with calculating and programming the actions for the Data Plane. This is where the forwarding decisions are made and where services (e.g. Quality of Service and VLANs) are implemented.
  - **Management Plane:** The Management Plane is the place where we can configure, monitor, and manage the network devices.

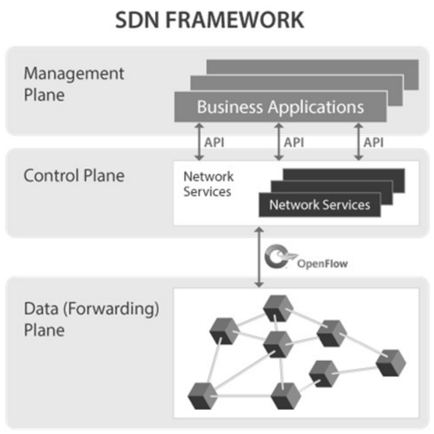
4장. 컨테이너 6. SDN

◻ Activities Performed by a Network Device

- **Ingress and egress packets:** These are done at the lowest layer, which decides what to do with ingress packets and which packets to forward, based on forwarding tables. These activities are mapped as Data Plane activities. All routers, switches, modem, etc. are part of this plane.
- **Collect, process, and manage the network information:** By collecting, processing, and managing the network information, the network device makes the forwarding decisions, which the Data Plane follows. These activities are mapped by the Control Plane activities. Some of the protocols which run on the Control Plane are routing and adjacent device discovery.
- **Monitor and manage the network:** Using the tools available in the Management Plane, we can interact with the network device to configure it and monitor it with tools like SNMP (Simple Network Management Protocol).

4장. 컨테이너 6. SDN

◻ SDN Framework



4장. 컨테이너 6. SDN

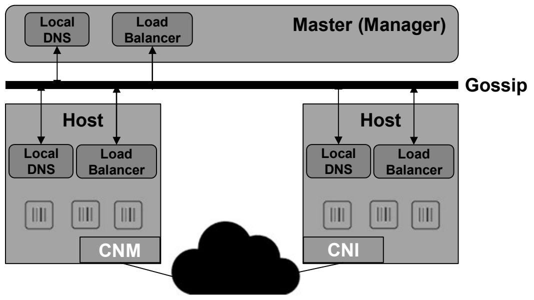
□ **Networking for Containers**

- **Similar to VMs, we need to connect containers on the same host and across hosts. The host kernel uses the network namespace feature of the Linux kernel to isolate the network from one container to another on the system. Network namespaces can be shared as well.**
- **On a single host, when using the virtual Ethernet (vEth) feature with Linux bridging, we can give a virtual network interface to each container and assign it an IP address. With technologies like Macvlan and IPVlan we can configure each container to have a unique and world-wide routable IP address.**

4장. 컨테이너 6. SDN

□ **Container Networking 표준 - CNM / CNI / Gossip**

- **Container Network Model (CNM):** Docker에서 제안, libnetwork에서 사용하며 Cisco Contiv, Kuryr, OVN, Project Calico, VMware, Vwave에서 사용
- **Container Network Interface (CNI):** CoreOS가 제안, K8s, Kurma, rkt, Apache Mesos, Cloud Foundry, Cisco Contiv, Project Calico, Weave, Docker
- **Gossip:** P2P, 네트워크 크기와 무관하게 동작, 일정한 개수의 불특정 노드에 정보를 gossiped, 지정한 오버레이를 통해 다른 노드에 알림



4장. 컨테이너 6. SDN

james@jslab.kr

- Container Networking Standards (1 of 2)
  - **Container Network Model (CNM):** Docker, Inc. is the primary driver for this networking model. It is implemented using the libnetwork project, which has the following utilizations:
    1. **Null:** NOOP implementation of the driver. It is used when no networking is required.
    2. **Bridge:** It provides a Linux-specific bridging implementation based in Linux Bridge.
    3. **Overlay:** It provides a multi-host communication over VXLAN.
    4. **Remote:** It does not provide a driver. Instead, it provides a means of supporting drivers over a remote transport, by which we can write third-party drivers.

JS Lab

213

4장. 컨테이너 6. SDN

james@jslab.kr

- Container Networking Standards (2 of 2)
  - **Container Networking Interface (CNI):** It is a Cloud Native Computing Foundation project which consists of specifications and libraries for writing plugins to configure network interfaces in Linux containers, along with a number of supported plugins. It is limited to provide network connectivity of containers and removing allocated resources when the container is deleted. As such, it has a wide range of support. It is used by projects like Kubernetes, OpenShift, Cloud Foundry, etc.

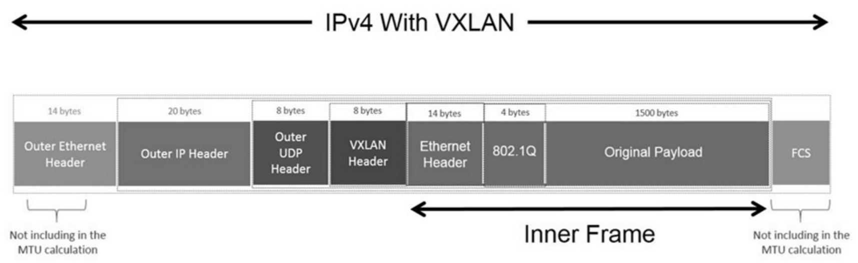
JS Lab

214

4장. 컨테이너 6. SDN

◘ Under-cloud for VxLAN

- **스위치의 VxLAN 지원 필요**
  - ◘ Multicast 지원
  - ◘ Maximum Frame Size 확장
- **Docker** (Swarm Overlay)
- **K8s** (Flannel and Weave)



4장. 컨테이너 6. SDN

◘ VxLAN 기능 향상

- **Instance(VM)의 CPU 처리 향상 (Software)**
- **SmartNIC (Hardware)**
- **Switch (Hardware)**

◘ VxLAN 대체 기능

- **VLAN (MAC based VLAN @ Docker)**
- **BGP**
- **MPLS**
- **OVS w/DPDK**

4장. 컨테이너 6. SDN

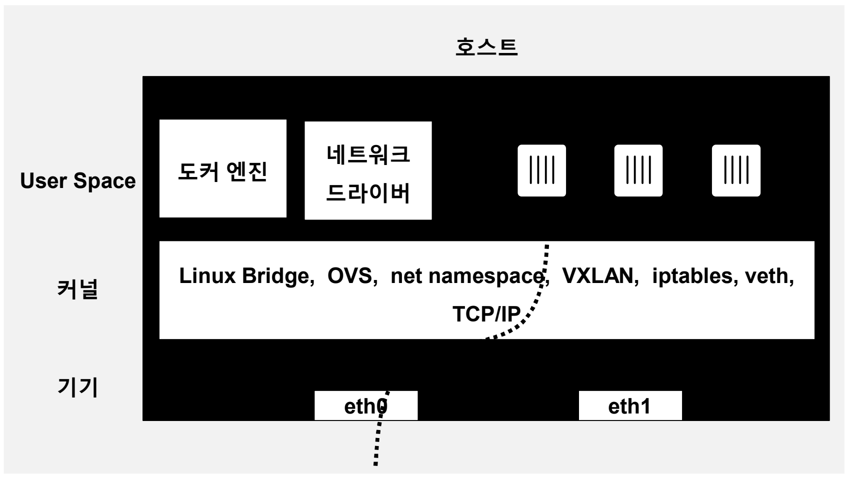
◻ Service Discovery

- **Registration:** When a container starts, the container scheduler registers the mapping in some key-value store like etcd or Consul. And, if the container restarts or stops, then the scheduler updates the mapping accordingly.
- **Lookup:** Services and applications use Lookup to get the address of a container, so that they can connect to it. Generally, this is done using some kind of DNS (Domain Name Server), which is local to the environment. The DNS used resolves the requests by looking at the entries in the key-value store, which is used for Registration. SkyDNS and Mesos-DNS are examples of such DNS services.

217

4장. 컨테이너 6. SDN

◻ 도커 네트워킹은 리눅스 네트워킹


호스트
User Space
커널
기기
도커 엔진
네트워크 드라이버
Linux Bridge, OVS, net namespace, VXLAN, iptables, veth, TCP/IP
eth0
eth1

IPAM: IP Address Management

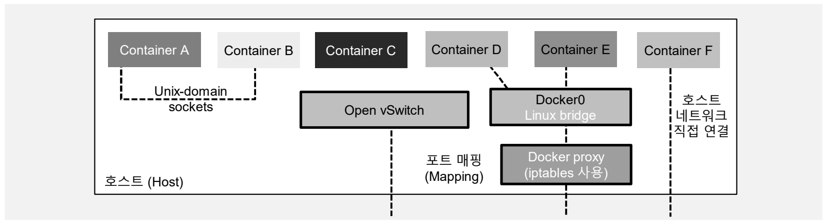
218

4장. 컨테이너

6. SDN

◻ 하이레벨 (High-level) 기능

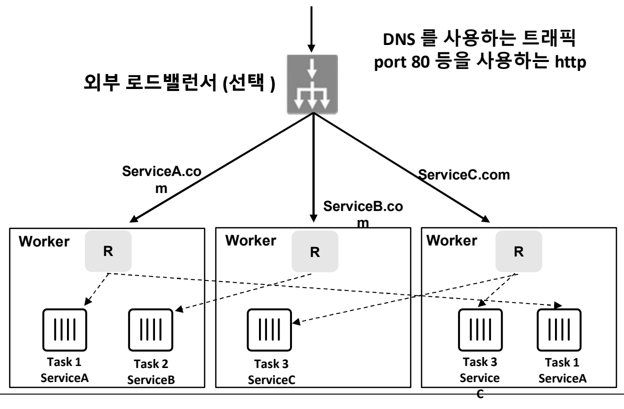
Namespace	/proc에서 프로세스 수준 관리의 컨테이너 네트워킹
Linux Bridge	커널에서 포워딩에 사용하는 L2/MAC을 인식하는 스위치
Open vSwitch	프로그램 가능하고 터널링을 지원하는 개선한 브릿지 (SDN 스위치)
NAT	네트워크 주소 변화 IP address + Ports (Types: SNAT, DNAT)
iptables	커널 내의 정책 엔진으로 패킷전송, 방화벽, NAT를 관리함
Unix domain sockets	단일 호스트 내 통신 기반의 File descriptor, FIFO 파이프로 동작
User-space vs Kernel-space	자원과 성능을 정상화 제어하는 애플리케이션도메인 • 컨테이너(Container) 애플리케이션(applications)은 user-space 에서 실행 • 네트워크 전송은 kernel space에서 실행



4장. 컨테이너

6. SDN

◻ **Docker의 라우팅 메쉬(Routing mesh):** 에지 라우팅을 위한 내장 라우팅 메쉬(routing Mesh)에서 모든 워커 노드(Worker Node)가 인그레스 라우팅 메쉬(Ingress Routing Mesh)에 참여하여 공개된 포트(Published Port)의 접속 요청을 수용하고 포트 변환은 워커노드에서 수행한다. 내부 로드밸런싱 매커니즘은 외부 요청에 동일하게 사용



4장. 컨테이너 6. SDN

- **Listing the Available Networks**
  - **bridge, null, and host are different network drivers available on a single host.**

```
$ docker network ls
NETWORK ID          NAME          DRIVER
6f30debc5baf        bridge        bridge
a1798169d2c0        host          host
4eb0210d40bb        none          null
```

4장. 컨테이너 6. SDN

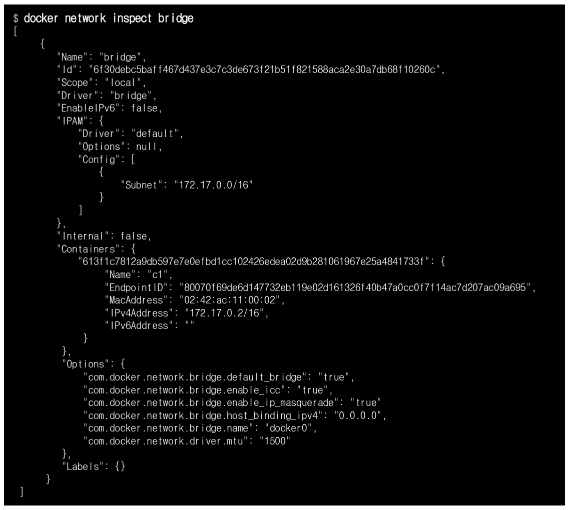
- **Bridge**

```
$ ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:A9:DB:AF:39
          inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
$ docker container run -it --name=c1 busybox /bin/sh
/ # ip a
1: lo: <LOOPBACK,UP,LOWE_UP> mtu 65536 qdisc noqueue qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
7: eth0@if8: <BROADCAST,MULTICAST,UP, LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:acff:fe11:2/64 scope link
       valid_lft forever preferred_lft forever
```

4장. 컨테이너 6. SDN

□ Inspecting a Bridge Network

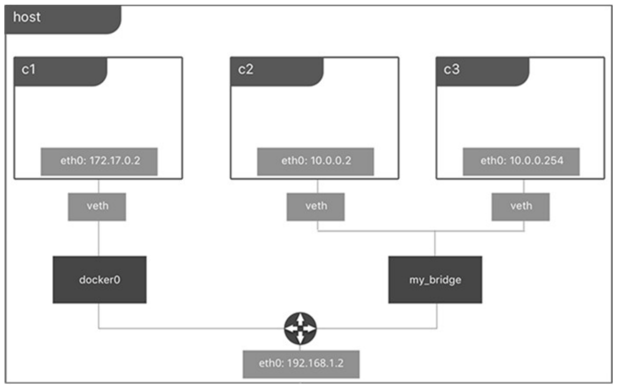


```
$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "6f30debc5baff467d437e3c7c3de673f21b51f821588aca2e30a7db68f10260c",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16"
                }
            ]
        },
        "Internal": false,
        "Containers": {
            "613f1c7812a9db597e7e0efbd1cc102426edea02d9b281061967e25a4841733f": {
                 "Name": "c1",
                 "EndpointID": "80070f69de6d147732eb119e02d161326f40b47a0cc0f7f14ac7d207ac09a695",
                 "MacAddress": "02:42:ac:11:00:02",
                 "IPv4Address": "172.17.0.2/16",
                 "IPv6Address": ""
            }
         },
         "Options": {
             "com.docker.network.bridge.default_bridge": "true",
             "com.docker.network.bridge.enable_icc": "true",
             "com.docker.network.bridge.enable_ip_masquerade": "true"
             "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
             "com.docker.network.bridge.name": "docker0",
             "com.docker.network.driver.mtu": "1500"
         },
         "Labels": {}
      }
  ]
```

4장. 컨테이너 6. SDN

□ Creating a Bridge Network

```
$ docker network create --driver bridge my_bridge
$ docker container run --net=my_bridge -itd --name=c2 busybox
```



Creating a Bridge Network (by Nigel Poulton, retrieved from GitHub)

4장. 컨테이너 6. SDN

◻ Null

```
$ docker container run -it --name=c3 --net=none busybox /bin/sh
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 1disc noqueue qlen 1
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet ::1/128 scope host
       valid_lft forever preferred_lft forever
```

4장. 컨테이너 6. SDN

◻ Host

- **Using the host driver, we can share the host machine's network namespace with a container. By doing so, the container would have full access to the host's network. We can see below that running an ifconfig command inside the container lists all the interfaces of the host system:**

```
$ docker container run -it --name=c4 --net=host  busybox /bin/sh
/ # ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:A9:DB:AF:39

          inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:a9ff:fedb:af39/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:536 (536.0 B)  TX bytes:648 (648.0 B)

eth0      Link encap:Ethernet  HWaddr 08:00:27:CA:BD:10
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:feca:bd10/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3399 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2050 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1021964 (998.0 KiB)  TX bytes:287879 (281.1 KiB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:00:42:F9
          inet addr:192.168.99.100  Bcast:192.168.99.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe00:42f9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:71 errors:0 dropped:0 overruns:0 frame:0
          TX packets:46 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13475 (13.1 KiB)  TX bytes:7754 (7.5 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:16 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:1021964376 (1.3 KiB)  TX bytes:1376 (1.3 KiB)

vethb3bb730 Link encap:Ethernet  HWaddr 4E:7C:8F:B2:2D:AD
          inet6 addr: fe80::4c7c:8fff:feb2:2dad/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:648 (648.0 B)  TX bytes:1296 (1.2 KiB)
```

4장. 컨테이너 6. SDN

- **Sharing Network Namespaces Among Containers (1 of 2)**

```
$ docker container run -it --name=c5 busybox /bin/sh
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever

10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
    link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.3/16 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::42:acff:fe11:3/64 scope link
        valid_lft forever preferred_lft forever
```

4장. 컨테이너 6. SDN

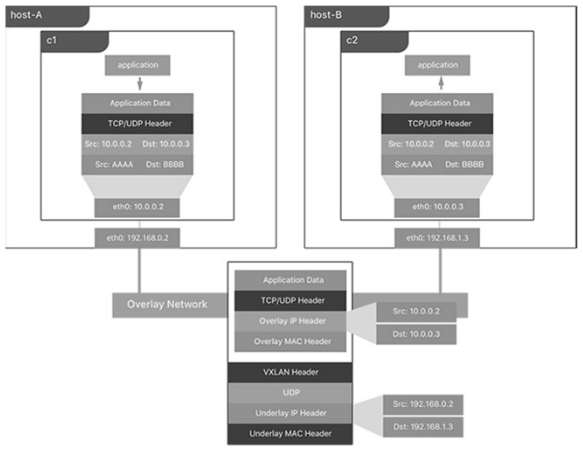
- **Sharing Network Namespaces Among Containers (2 of 2)**

```
$ docker container run -it --name=c6 --net=container:c5 busybox /bin/sh
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever

12: eth0@if13: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
    link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.3/16 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::42:acff:fe11:3/64 scope link
        valid_lft forever preferred_lft forever
```

4장. 컨테이너 6. SDN

james@jslab.kr

◻ Docker Multi-Host Networking



Docker Overlay Driver (by Nigel Poulton, retrieved from GitHub)

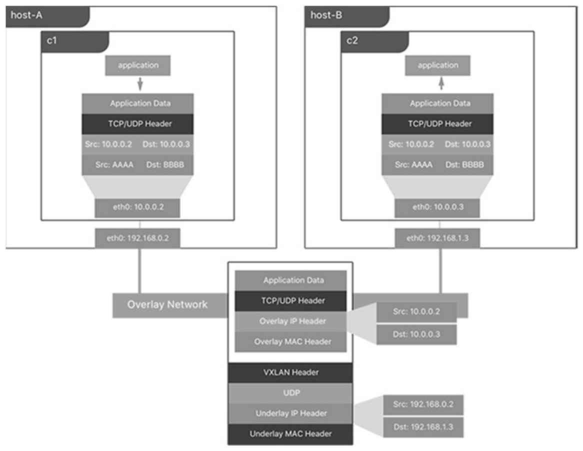
JS Lab

229

4장. 컨테이너 6. SDN

james@jslab.kr

◻ Docker Multi-Host Networking



Docker Overlay Driver (by Nigel Poulton, retrieved from GitHub)

JS Lab

230

4장. 컨테이너 6. SDN

- **Docker Network Driver Plugins:** The functionality of Docker can be extended with plugins. Docker provides plugins for network and volumes.
  - **Weave Network Plugin:** Weave Net provides multi-host container networking for Docker. It also provides service discovery and does not require any external cluster store to save the networking configuration. Weave Net has a Docker Networking Plugin which we can use with Docker deployment.
  - **Kuryr Network Plugin:** It is a part of the OpenStack Kuryr project, which also implements libnetwork's remote driver API by utilizing Neutron, which is OpenStack's networking service

4장. 컨테이너 6. SDN

- **Kubernetes Networking**
  - **As each Pod gets a unique IP, Kubernetes assumes that Pods should be able to communicate with each other, irrespective of the nodes they get scheduled on. There are different ways to achieve this. Kubernetes uses the Container Network Interface (CNI) for container networking and has put down the following rules if someone wants to write drivers for Kubernetes networking:**

    1. All pods can communicate with all other pods without NAT
    2. All nodes running pods can communicate with all pods (and vice versa) without NAT
    3. The IP that a pod sees itself as is the same IP that other pods see it as.

4장. 컨테이너 6. SDN

- Kubernetes Networking
  - **Implementations of Kubernetes Networking:**
    1. **Flannel:** Flannel uses the overlay network, as we have seen with Docker, to meet the Kubernetes networking requirements.
    2. **Calico:** Calico, or Project Calico, uses the BGP protocol to meet the Kubernetes networking requirements.

233

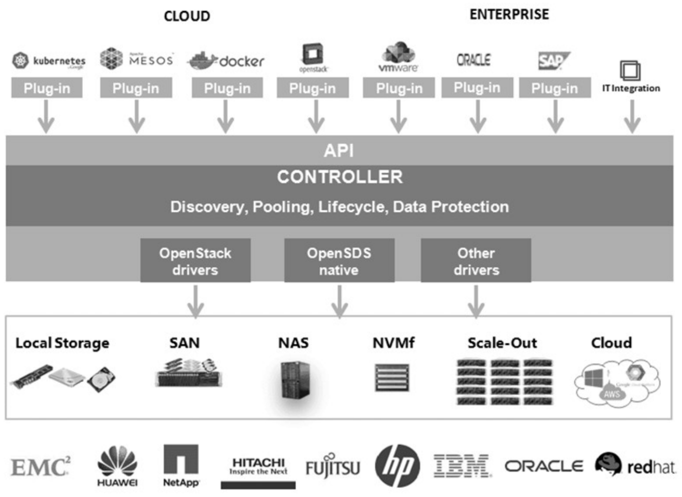
4장. 컨테이너 6. SDN

- Cloud Foundry: Container-to-Container Networking
  - **By default, the Gorouter routes the external and internal traffic to different Cloud Foundry components. Even the application- to-application communication happens via the Gorouter.**
  - **To simplify the app-to-app communication, we can enable container-to-container networking with Cloud Foundry. It implements it using Overlay Networking, which we discussed earlier. With Overlay Networking, each container gets its own unique IP address which is reachable from other application instances.**
  - **Container-to-container also allows network policy creation and enforcement using which sets routing rules based on app, destination app, protocol and ports;, without going through the Gorouter, a load balancer, or a firewall**

234

4장. 컨테이너 7. SDS

◻ OpenSDS Architecture



https://www.opensds.io/blog/opensds-cncf-cluster-integration-testing/

4장. 컨테이너 7. SDS

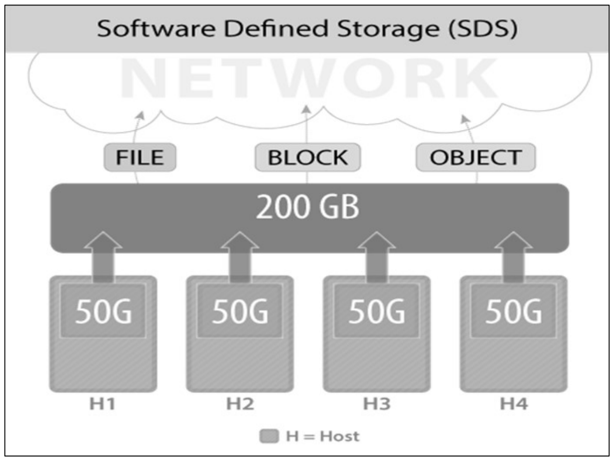
◻ **Software-Defined Storage (SDS)**

- **A form of storage virtualization in which the storage hardware is separated from the software which manages it. By doing this, we can bring hardware from different sources and we can manage them with software. Software can provide different features, like replication, erasure coding, snapshot, etc. on top of the pooled resources. Once the pooled storage is configured in the form of a cluster, SDS allows multiple access methods like File, Block, and Object.**

    1. Ceph
    2. Gluster
    3. FreeNAS
    4. Nexenta
    5. VMware Virtual SAN

4장. 컨테이너 7. SDS

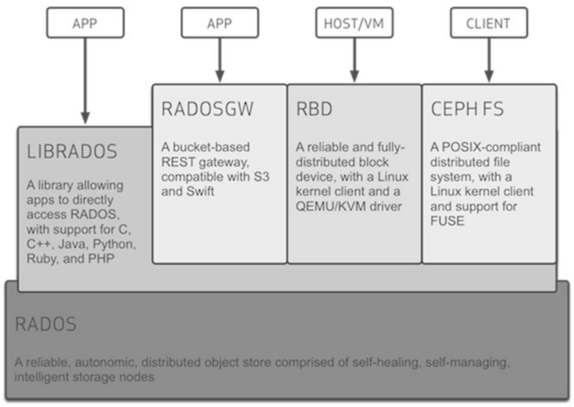
□ **Software-Defined Storage (SDS)**



237

4장. 컨테이너 7. SDS

□ **Ceph Architecture**


APP
APP
HOST/VM
CLIENT
RADOSGW
A bucket-based REST gateway, compatible with S3 and Swift
RBD
A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver
CEPH FS
A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE
LIBRADOS
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP
RADOS
A reliable, autonomic, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

Ceph Architecture (by Red Hat, Inc., retrieved from ceph.com)

238

4장. 컨테이너 7. SDS

- **Reliable Autonomic Distributed Object Store** (RADOS)
  - **It is the object store which stores the objects. This layer makes sure that data is always in a consistent and reliable state. It performs operations like replication, failure detection, recovery, data migration, and rebalancing across the cluster nodes. This layer has the following three major components:**
    1. **Object Storage Device (OSD):** This is where the actual user content is written and retrieved with read operations. One OSD daemon is typically tied to one physical disk in the cluster.
    2. **Ceph Monitors (MON):** Monitors are responsible for monitoring the cluster state. All cluster nodes report to Monitors. Monitors map the cluster state through the OSD, Place Groups (PG), CRUSH and Monitor maps.
    3. **Ceph Metadata Server (MDS):** It is needed only by CephFS, to store the file hierarchy and metadata for files.

4장. 컨테이너 7. SDS

- **Ceph**
  - **Librados:** It is a library that allows direct access to RADOS from languages like C, C++, Python, Java, PHP, etc. Ceph Block Device, RADOSGW, and CephFS are implemented on top of Librados.
  - **Ceph Block Device: This provides the block interface for Ceph. It works as a block device and has enterprise features like thin provisioning and snapshots.**
  - **RADOS Gateway (RADOSGW):** This provides a REST API interface for Ceph, which is compatible with Amazon S3 and OpenStack Swift.
  - **Ceph File System (CephFS):** This provides a POSIX-compliant distributed filesystem on top of Ceph. It relies on Ceph MDS to track the file hierarchy.

4장. 컨테이너 7. SDS

- Benefits of Using Ceph
  - It is an open source storage solution, which supports Object, Block, and Filesystem storage.
  - It runs on any commodity hardware, without any vendor lock-in.
  - It provides data safety for mission-critical applications.
  - It provides automatic balance of filesystems for maximum performance.
  - It is scalable and highly available, with no single point of failure.
  - It is a reliable, flexible, and cost-effective storage solution.
  - It achieves higher throughput by stripping the files/data across the multiple nodes.
  - It achieves adaptive load-balancing by replicating frequently accessed objects over multiple nodes.

241

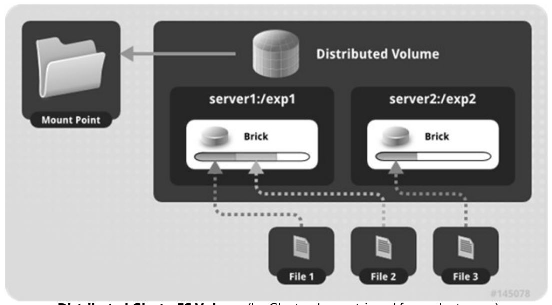
4장. 컨테이너 7. SDS

- GlusterFS
  - According to gluster.org: "Gluster is a scalable, distributed file system that aggregates disk storage resources from multiple servers into a single global namespace

242

4장. 컨테이너 7. SDS

- GlusterFS Volumes
  - distributed GlusterFS volume
  - replicated GlusterFS volume
  - distributed replicated GlusterFS volume
  - dispersed GlusterFS volume
  - distributed dispersed GlusterFS volume.



Distributed GlusterFS Volume (by Gluster, Inc., retrieved from gluster.org)

JS Lab

243

4장. 컨테이너 7. SDS

- The GlusterFS volume can be accessed using one of the following methods:
  - Native FUSE mount
  - NFS (Network File System)
  - CIFS (Common Internet File System)

JS Lab

244

4장. 컨테이너 7. SDS

- Benefits of Using GlusterFS
  - It scales to several petabytes.
  - It can be configured on a commodity hardware.
  - It is an open source storage solution that supports Object, Block, and Filesystem storage.
  - It does not have a metadata server.
  - It is scalable, modular and extensible.
  - It provides features like replication, quotas, geo-replication, snapshots and BitRot detection.
  - It is POSIX-compliant, providing High Availability via local and remote data replication.
  - It allows optimization for different workloads.

4장. 컨테이너 7. SDS

- Introduction to Storage Management for Containers
  - Containers are ephemeral in nature, which means that whatever is stored inside the container would be gone as soon as the container is deleted. It is best practice to store data outside the container, which would be accessible even after the container is gone.
  - In a multi-host environment, containers can be scheduled to run on any host. We need to make sure the data volume required by the container is available on the node on which the container would be running.
  - In this section, we will see how Docker uses the Docker Volume feature to store persistent data and allows vendors to support their storage to its ecosystem, using Docker Volume Plugins. We will start by looking into different storage backends, which Docker supports in order to store images, containers, and other metadata.

4장. 컨테이너 7. SDS

- Docker Storage Backends
  - AUFS (Another Union File System)
  - BtrFS
  - Device Mapper
  - Overlay
  - VFS (Virtual File System)
  - ZFS
  - windowsfilter.

4장. 컨테이너 7. SDS

- Managing Data in Docker
  - **Docker Volumes:** On Linux, Docker Volumes are stored under the /var/lib/docker/volumes folder and they are directly managed by Docker.
  - **Bind Mounts:** Using Bind Mounts, Docker can mount any file or directory from the host system into a container
- In both cases, Docker bypasses the Union Filesystem, which it uses for copy-on-write. The writes happen directly to the host directory.
- On Linux, Docker also supports tmpfs mounts, which are available in the host system's memory. Docker also supports third party volume plugins, which we will look at later.

4장. 컨테이너 7. SDS

- Creating a Container with Volumes
  - To create a container with volume, we can use either the docker container run or the docker container create commands
  - The command would create a Docker volume inside the Docker working directory (default to /var/lib/docker/) on the host system. We can get the exact path via the docker container inspect command
  - We can give a specific name to a Docker volume and then use it for different operations. To create a named volume, we can run the command

```
$ docker container run -d --name web -v /webapp nkhare/myapp
$ docker container inspect web
$ docker volume create --name my-named-volume
```

4장. 컨테이너 7. SDS

- Mounting a Host Directory Inside the Container
  - To do a bind mount, we can run the following command
  - which would mount the host's /mnt/webapp directory to /webapp, while starting the container.

```
$ docker container run -d --name web -v /mnt/webapp:/webapp nkhare/myapp
```

4장. 컨테이너 7. SDS

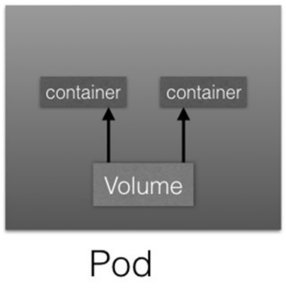
- Volume Plugins for Docker
  - GlusterFS
  - Blockbridge
  - EMC REX-Ray
- Volume plugins are especially helpful when we migrate a stateful container, like a database, on a multi-host environment. In such an environment, we have to make sure that the volume attached to a container is also migrated to the host where the container is migrated or started afresh.

4장. 컨테이너 7. SDS

- GlusterFS Volume Plugin
  - Earlier in this chapter, we saw how we can create a shared storage pool using GlusterFS. Docker provides a plugin for GlusterFS, which we can use to attach/detach storage to one or more containers on-demand.

4장. 컨테이너 7. SDS

- **Volume Management in Kubernetes**
- **Kubernetes uses volumes to attach external storage to the Pods. A volume is essentially a directory, backed by a storage medium. The storage medium and its contents are determined by the volume type.**



253

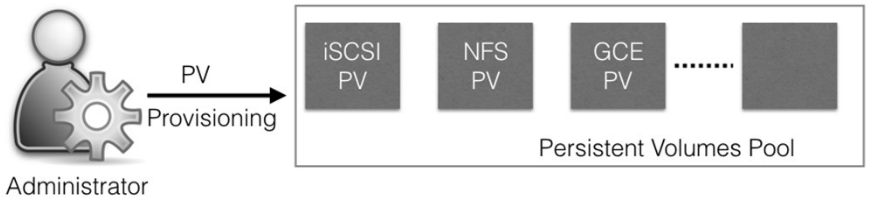
4장. 컨테이너 7. SDS

- **Volume Types**
  - **emptyDir:** An empty volume is created for the Pod as soon as it is scheduled on a worker node. The life of the volume is tightly coupled with the Pod. If the Pod dies, the content of emptyDir is deleted forever.
  - **hostPath:** With the hostPath volume type, we can share a directory from the host to the Pod. If the Pod dies, the content of the volume is still available on the host.
  - **gcePersistentDisk:** With the gcePersistentDisk volume type, we can mount a Google Compute Engine (GCE) persistent disk into a Pod.
  - **awsElasticBlockStore:** With the awsElasticBlockStore volume type, we can mount an AWS EBS volume into a Pod.
  - **Nfs:** With the nfs volume type, we can mount an NFS share into a Pod.
  - **persistentVolumeClaim:** With the persistentVolumeClaim type we can attach a persistent volume to a Pod. We will cover this in the next section.

254

4장. 컨테이너 7. SDS

- **Persistent Volumes**

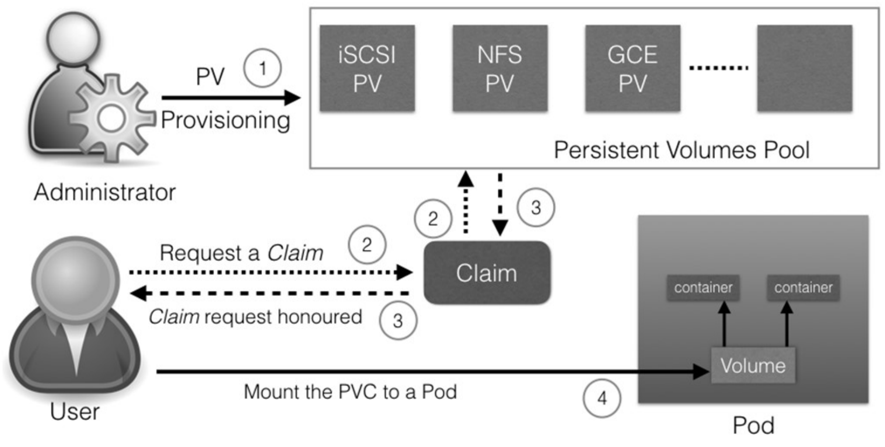


4장. 컨테이너 7. SDS

- **Persistent Volumes**
  - **For dynamic provisioning of PVs, Kubernetes uses the StorageClass resource, which contains pre-defined provisioners and parameters for the PV creation. With PersistentVolumeClaim (PVC), a user sends the requests for dynamic PV creation, which gets wired to the StorageClass resource.**
  - **Some of the volume types that support managing storage using PersistentVolume are:**
    1. GCEPersistentDisk
    2. AWSElasticBlockStore
    3. AzureFile
    4. NFS
    5. iSCSI.

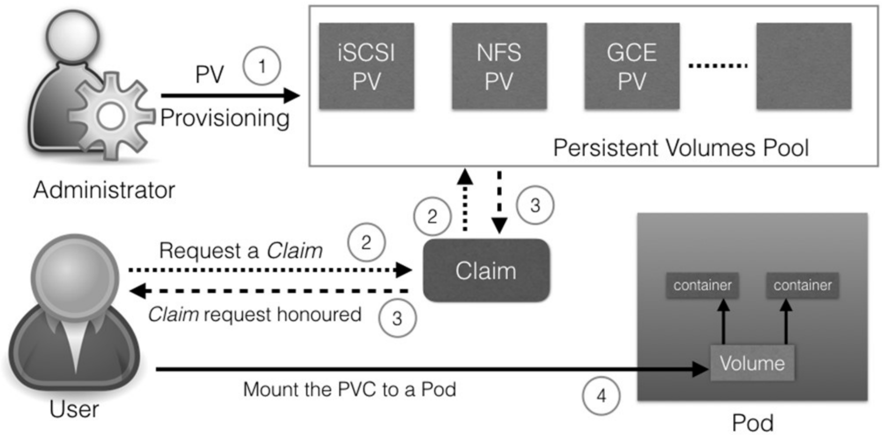
4장. 컨테이너 7. SDS

◻ **Persistent Volumes Claim**



4장. 컨테이너 7. SDS

◻ **Using PVC in a Pod**



4장. 컨테이너 7. SDS

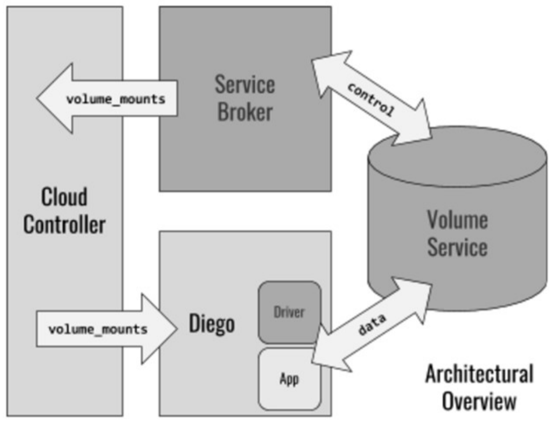
□ Container Storage Interface (CSI)

- At the time this course was released (August 1, 2018), container orchestrators like Kubernetes, Mesos, Docker and Cloud Foundry have their own way of managing external storage using volumes. For a storage vendor, it is difficult to manage different volume plugins for different orchestrators. Storage vendors and community members from different orchestrators are working together to standardize the volume interface to avoid duplicate work. This is being referred to as the Container Storage Interface (CSI). With CSI, the same volume plugin would work for different container Oorchestrators.
- To learn more, take a look at the Container Storage Interface (CSI) Specification posted on GitHub.

259

4장. 컨테이너 7. SDS

□ Cloud Foundry Volume Service



260

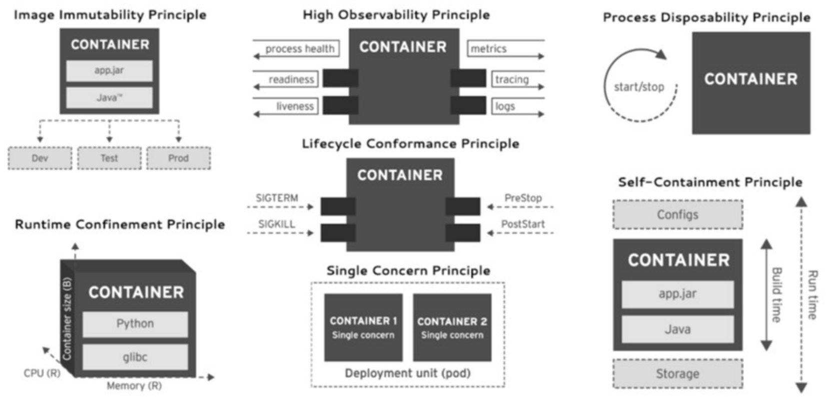
4장. 컨테이너 7. SDS

- **CF Volume Services**
  - **Local-volume-release:** It provides local-volume service for Cloud Foundry applications.
  - **nfs-volume-release:** This allows easy mounting of external NFS shares for Cloud Foundry applications.
  - **Cephfs-bosh-release :** It is an open source shared volumes service, built on top of Ceph.
  - **Efs-volume-release :** It provides driver and service broker for provisioning and mounting Amazon EFS volumes.

261

4장. 컨테이너 7. SDS

- **Container-based Application Design**



https://kubernetes.io/blog/2018/03/principles-of-container-app-design/?fbclid=IwAR2oMrdP0d1Q6LXebtxNPnt--RS5DIlkCIwpaMSL5mmW7VMaQb6hRV8hkd38

262

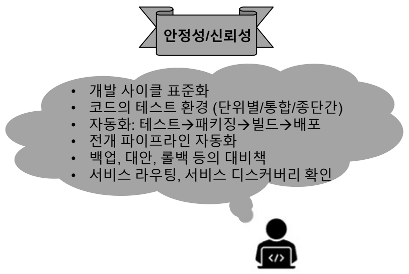
1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ **실습교재 (별도)**

5장. DevOps와 CI/CD

- **DevOps**
- **CI/CD**

5장. DevOps와 CI/CD

- Every industry thrives for better quality and faster innovation. The IT industry is no exception and has to address numerous challenges, like the following:
  - It must quickly go from business idea to market.
  - It must lower the failure rate for new releases.
  - It must have a shorter lead time between fixes.
  - It must have a faster mean time to recovery

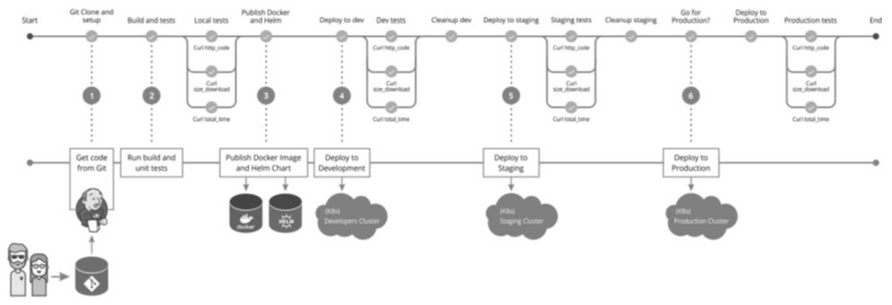


- 개발 사이클 표준화
- 코드의 테스트 환경 (단위별/통합/종단간)
- 자동화: 테스트→패키징→빌드→배포
- 전개 파이프라인 자동화
- 백업, 대안, 롤백 등의 대비책
- 서비스 라우팅, 서비스 디스커버리 확인

265

5장. DevOps와 CI/CD

- CI/CD Tools – Container-based
  - Immutable Image
  - API Testing
  - Blue/Green, Canary
  - 도구: Jenkins + Docker + Spinnaker + Helm + Kubernetes



266

5장. DevOps와 CI/CD

- The collaborative work between Developers and Operations is referred to as DevOps. DevOps is more of a mindset, a way of thinking, versus a set of processes implemented in a specific way.
- Besides Continuous Integration (CI), DevOps also enables Continuous Deployment (CD), which can be seen as the next step of CI. In CD, we deploy the entire application/software automatically, provided that all the tests' results and conditions have met the expectations.

5장. DevOps와 CI/CD

- Jenkins
  - Jenkins is one of the most popular and used tools for doing any kind of automation. It is an open source automation system which can provide Continuous Integration and Continuous Deployment. It is written in Java.
  - CloudBees is one of the primary sponsors for the Jenkins open source project. CloudBees provides different products based on top of Jenkins.

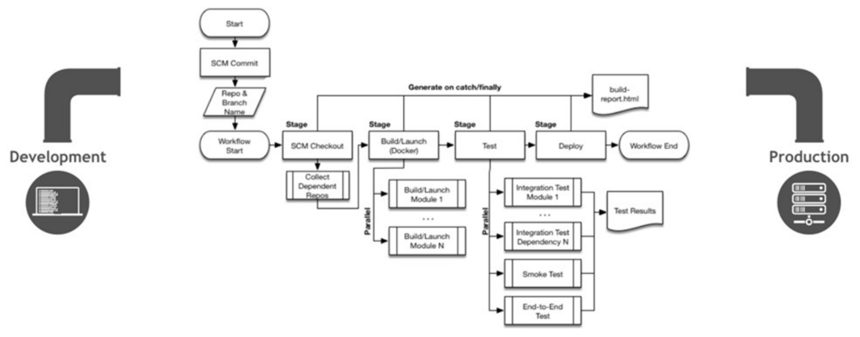
5장. DevOps와 CI/CD

- **Jenkins Functionality**
  - **Durable: Pipelines can survive both planned and unplanned restarts of your Jenkins master.**
  - **Pausable: Pipelines can optionally stop and wait for human input or approval before completing the jobs for which they were built.**
  - **Versatile: Pipelines support complex real-world CD requirements, including the ability to fork or join, loop, and work in parallel with each other.**
  - **Extensible: The Pipeline plugin supports custom extensions to its DSL (domain scripting language) and multiple options for integration with other plugins".**

269

5장. DevOps와 CI/CD

- **The flowchart below illustrates a sample deployment using the Pipeline Plugin:**



Jenkins Pipeline (by Jenkins/CC BY-SA 4.0, retrieved from jenkins.io)

270

5장. DevOps와 CI/CD

- Benefits of Using Jenkins
  - It is an open source automation system.
  - It supports Continuous Integration and Deployment.
  - It is extensible through plugins.
  - It can be easily installed, configured, and distributed.

james@jslab.kr

JS Lab

271

5장. DevOps와 CI/CD

- Travis CI
- Travis CI is a hosted, distributed CI solution for projects hosted only on GitHub.
- To run the test with CI, first we have to link our GitHub account with Travis and select the project (repository) for which we want to run the test. In the project's repository, we have to create a .travis.yml file, which defines how our build should be executed step-by-step.

james@jslab.kr

JS Lab

272

5장. DevOps와 CI/CD

- Executing Build with Travis (1 of 2)
- A typical build with Travis consists of two steps:
  - install: to install any dependency or pre-requisite.
  - script: to run the build script.

Travis CI란?

- Travis CI는 Github에서 진행되는 오픈소스 프로젝트를 위한 지속적인 통합 (Continuous Integration) 서비스이다.
- Private repository는 유료로 일정 금액을 지불하고 사용할 수 있다.

JS Lab

273

5장. DevOps와 CI/CD

- Executing Build with Travis (2 of 2)
- We can also add other optional steps, including the deployment steps. Following are all the build options one can put in a .travis.yml file.
  - before_install
  - install
  - before_script
  - script
  - after_success or after_failure
  - OPTIONAL before_deploy
  - OPTIONAL deploy
  - OPTIONAL after_deploy
  - after_script

JS Lab

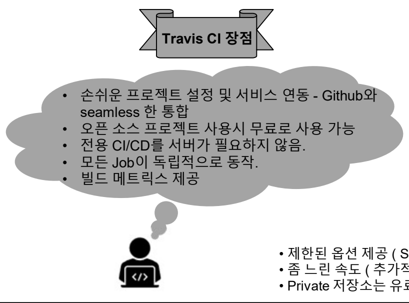
274

5장. DevOps와 CI/CD

- Travis CI Characteristics
  - **Travis CI supports different databases, like MYSQL, RIAK, memcached, radis, etc. We can also use Docker during the build.**
  - **Travis CI supports most languages. To see a detailed list of the languages supported, please take a look at the "Language-specific Guides" page.**
  - **After running the test, we can deploy the application in many cloud providers, such as Heroku, AWS Codedeploy, Cloud Foundry, OpenShift, etc. A detailed list of providers is available in the "Supported Providers" page by Travis CI.**

5장. DevOps와 CI/CD

- Benefits of Using Travis CI
- Some of the benefits of using Travis CI are:
  - **It is a hosted, distributed solution integrated with GitHub.**
  - **It can be easily set up and configured.**
  - **It is free for open source projects.**
  - **It supports testing for different versions of the same runtime.**



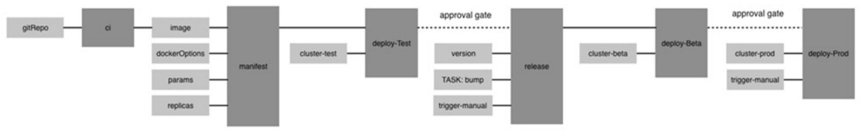
- 손쉬운 프로젝트 설정 및 서비스 연동 - Github와 seamless 한 통합
- 오픈 소스 프로젝트 사용시 무료로 사용 가능
- 전용 CI/CD를 서버가 필요하지 않음.
- 모든 Job이 독립적으로 동작.
- 빌드 메트릭스 제공

Travis CI 단점

- 제한된 옵션 제공 (Site GUI에 할 수 있는게 제한되었음)
- 좀 느린 속도 (추가적인 비용을 들여도 제한된 성능)
- Private 저장소는 유료 Plan (Enterprise)

5장. DevOps와 CI/CD

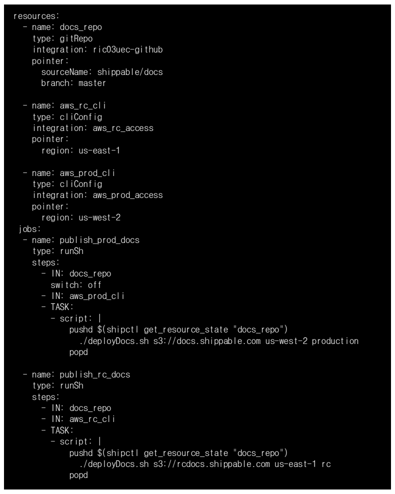
- **Shippable**
- **As per Shippable website:** "Shippable is a DevOps Assembly Lines Platform that helps developers and DevOps teams achieve CI/CD and make software releases frequent, predictable, and error-free. We do this by connecting all your DevOps tools and activities into a event-driven, stateful workflow".
-



The `shippable.yml` Structure (by Shippable, Inc., retrieved from shippable.com)

5장. DevOps와 CI/CD

- **Testing with Shippable**
  - **To run CI tests with Shippable, we have to create a configuration file inside the project's source code repository which we want to test, called shippable.yml.**



```
resources:
  - name: docs_repo
    type: gitRepo
    integration: ric03uec-github
    pointer:
      sourceName: shippable/docs
      branch: master

  - name: aws_rc_cli
    type: cliConfig
    integration: aws_rc_access
    pointer:
      region: us-east-1

  - name: aws_prod_cli
    type: cliConfig
    integration: aws_prod_access
    pointer:
      region: us-west-2
 jobs:
  - name: publish_prod_docs
    type: runSh
    steps:
      - IN: docs_repo
        switch: off
      - IN: aws_prod_cli
      - TASK:
        - script: |
            pushd $(shipctl get_resource_state "docs_repo")
              ./deployDocs.sh s3://docs.shippable.com us-west-2 production
            popd

  - name: publish_rc_docs
    type: runSh
    steps:
      - IN: docs_repo
      - IN: aws_rc_cli
      - TASK:
        - script: |
            pushd $(shipctl get_resource_state "docs_repo")
              ./deployDocs.sh s3://rcdocs.shippable.com us-east-1 rc
            popd
```

5장. DevOps와 CI/CD

- Programming Languages Supported by Shippable
- Currently, Shippable supports the following programming languages for CI:
  - C/C++
  - Clojure
  - Go
  - Java
  - Node.js
  - PHP
  - Python
  - Ruby
  - Scala.

5장. DevOps와 CI/CD

- Integrations
  - Shippable integrates well with all popular CI/CD and DevOps tools like GitHub, Bitbucket, Docker Hub, JUnit, Kubernetes, Slack, Terraform, etc.
  - For more details about these and other tools please refer to Shippable's website.

5장. DevOps와 CI/CD

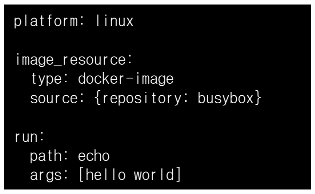
- Benefits of Using Shippable
- Some of the benefits of using Shippable are:
  - Builds are faster.
  - It supports builds against multiple runtimes, environment variables, and platforms.
  - It supports on-premise systems for builds.
  - It achieves Continuous Delivery by automating CI and DevOps activities.
  - It optimizes DevOps operations.
  - It provides security with native secrets management and RBAC.

5장. DevOps와 CI/CD

- Concourse
- Concourse is an open source CI/CD system, which was started by Alex Suraci and Christopher Brown in 2014. Later, Pivotal sponsored the project. It is written in the Go language.

5장. DevOps와 CI/CD

- With Concourse, we run series of tasks to perform desired operations. Each task runs inside a container. Using series of tasks along with resources, we can build a job or pipeline. Following is a sample task file:



```
platform: linux

image_resource:
  type: docker-image
  source: {repository: busybox}

run:
  path: echo
  args: [hello world]
```

JS Lab

283

5장. DevOps와 CI/CD

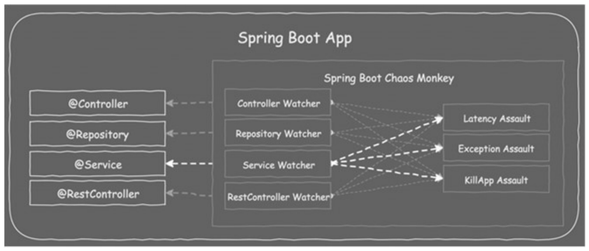
- Benefits of Using Concourse
  - It is an open source tool.
  - It can be set up on-premise or on cloud.
  - It is a very simple way to configure and manage CI pipelines.
  - It has good visualization for our pipeline and tasks.
  - It can be scaled across many servers.
  - In Concourse, the necessary data to run the pipeline can be provided by resources. These resources never affect the performance of a worker.

JS Lab

284

5장. DevOps와 CI/CD

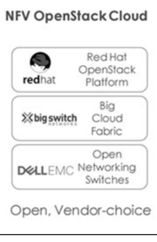
- **Chaos Monkey**
  - **Continuous Delivery at Netflix:** Netflix designed Chaos Monkey to test system stability by enforcing failures via the pseudo-random termination of instances and services within Netflix's architecture.
  - **Kube Monkey:** Kubernetes Pod Chaos Monkey, Chaos Toolkit, and Gremlin tools, which can be deployed on Kubernetes clusters to execute Chaos Experiments and create more resilient applications.

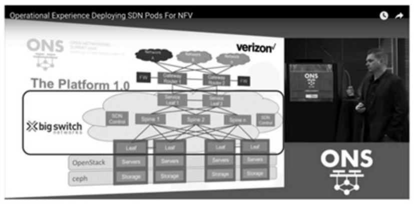


https://www.gremlin.com/chaos-monkey/chaos-monkey-alternatives/kubernetes/

5장. DevOps와 CI/CD

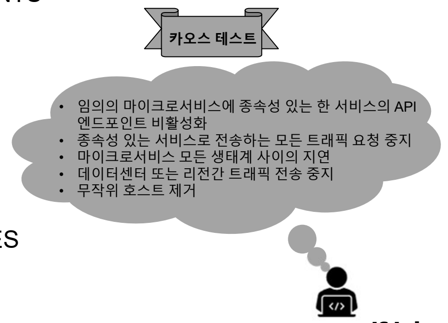
- **Chaos Monkey Test @ Telecom**
  - **30분간 640개 임의 요소 장애 발생시 성능 저하 없음**
    - 32 leaf / 6 spine / 16 rack pod
    - 1,000 OpenStack neutron network
    - 48,000 VM (혼합 구성: Hadoop worker / traffic generator)
    - 매 30 초 간격 Controller 장애
    - 매 8초 간격 스위치 장애
    - 매 4초 간격 링크 장애





5장. DevOps와 CI/CD

- Chaos Monkey
  - 장점:
    - PREPARES YOU FOR RANDOM FAILURES
    - ENCOURAGES DISTRIBUTION
    - ENCOURAGES REDUNDANCY
    - DISCOURAGES WASTEFUL DEPENDENCIES
    - DISCOVERING IMPROVEMENTS
    - BUILT INTO SPINNAKER
  - 단점
    - REQUIRES SPINNAKER
    - REQUIRES MYSQL
    - LIMITED FAILURE MODE
    - LACK OF COORDINATION
    - NO RECOVERY CAPABILITIES
    - NO USER INTERFACE



카오스 테스트

- 임의의 마이크로서비스에 종속성 있는 한 서비스의 API 엔드포인트 비활성화
- 종속성 있는 서비스로 전송하는 모든 트래픽 요청 중지
- 마이크로서비스 모든 생태계 사이의 지연
- 데이터센터 또는 리전간 트래픽 전송 중지
- 무작위 호스트 제거

287

5장. DevOps와 CI/CD

- Cloud Native CI/CD
- In the Cloud Native approach, we design a package and run applications on top of our infrastructure (on-premise or cloud) using operations tools like containers, container orchestration and services like continuous integration, logging, monitoring, etc. Kubernetes along with the tooling around it meets our requirements to run Cloud Native applications.

288

1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ **실습교재 (별도)**

6장. 도구(Tools)

1. **Configuration Management**
2. **Build & Release**
3. **Key-Value Pair Store**
4. **Image Building**
5. **Debugging, Logging, and Monitoring for Containerized Applications**
6. **Immutable Infrastructure**

6장. 도구(Tools) 1. Config Mgmt

Tools (1 of 2)

- **Helm:** It is the package manager for Kubernetes. Packages are referred as Charts. Using Helm, we can package, share, install or upgrade an application. It was recently incubated as a CNCF project.
- **Draft:** It is being promoted as a developer tool cloud native application on Kubernetes. With Draft, we can containerize and deploy an application on Kubernetes.
- **Skaffold:** It is a tool from Google that helps us build and deploy the code to the Kubernetes development environment each time the code changes locally. It also supports Helm.

6장. 도구(Tools) 1. Config Mgmt

Tools (2 of 2)

- **Argo:** It is a container-native workflow engine for Kubernetes. Its use cases include running Machine Learning, Data Processing and CI/CD tasks on Kubernetes.
- **Jenkins X:** Jenkins has been a very popular tool for doing CI/CD and it can be used on Kubernetes as well. But the Jenkins team built a new cloud native CI/CD, Jenkins X from the ground up, to run directly on Kubernetes.
- **Spinnaker:** It is an open source multi-cloud continuous delivery platform from Netflix for releasing software changes with high velocity. It supports all the major cloud providers like Amazon Web Services, Azure, Google Cloud Platform and OpenStack. It supports Kubernetes natively.

6장. 도구(Tools) 1. Config Mgmt

GitOps and SecOps

- **While we are talking about CI/CD, let's talk about GitOps and SecOps practices as well, which have emerged recently. So far, we have been using Git as a single point of truth for code commits. With GitOps, it is extended to the entire system. Git becomes the single point of truth for code, deployment configuration, monitoring rules, etc.. With just a Git pull request, we can do/update the complete application deployment. Some of the examples of GitOps are Weave Flux and GitKube.**
- **With SecOps, security would not be an afterthought when it comes tp application deployment. It would be included along with the application deployment, to avoid any security risks.**

6장. 도구(Tools) 1. Config Mgmt

Ansible

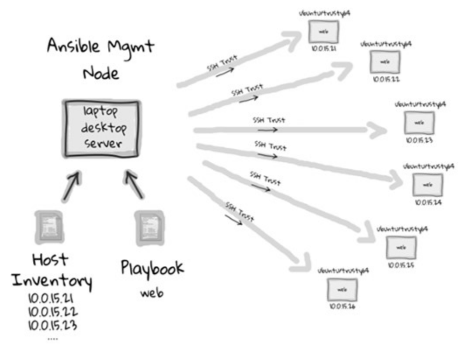
- **Red Hat's Ansible is an easy-to-use, open source configuration management tool. It is an agentless tool which works on top of SSH. Ansible also automates cloud provisioning, application deployment, orchestration, etc.**

6장. 도구(Tools) 1. Config Mgmt

Ansible

- **Nodes: To list the the nodes which we want to manage in an inventory file, we must do the following:**

```
[webservers]
www1.example.com
www2.example.com

[dbservers]
db0.example.com
db1.example.com
```



nsible Multi-Node Deployment Workflow (retrieved from sysadmincasts.com)

6장. 도구(Tools) 1. Config Mgmt

Ansible

- **The nodes can be grouped together as shown in the picture above. Ansible also supports dynamic inventory files for cloud providers like AWS and OpenStack. The management node connects to these nodes with a password or it can do passwordless login, using SSH keys.**

- **Ansible ships with a default set of modules, like packaging, network, etc., which can be executed directly or via Playbooks. We can also create custom modules.**

6장. 도구(Tools) 1. Config Mgmt

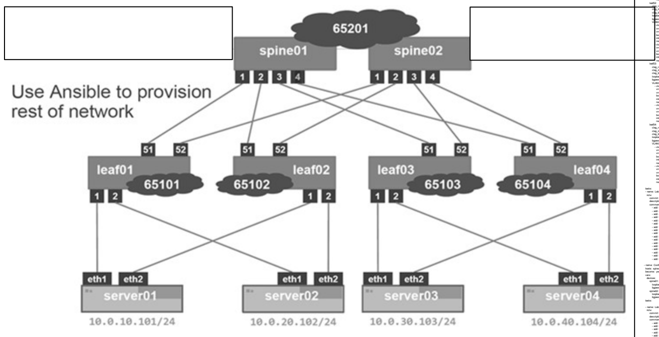
- **Ansible**
  - **Playbooks**
    1. The Ansible management node connects to the nodes mentioned in the inventory file and runs the tasks mentioned in the playbook. A management node can be installed on any Unix-based system like Linux, Mac OS X, etc. It can manage any node which supports SSH and Python 2.4 or later.
    2. Ansible Galaxy is a free site for finding, downloading, and sharing community-developed Ansible roles.

```
# This playbook deploys the whole application stack in this site.
- name: apply common configuration to all nodes
  hosts: all
  remote_user: root

  roles:
    - common

- name: configure and deploy the webservers and application code
  hosts: webservers
  remote_user: root

  roles:
    - web

- name: deploy MySQL and configure the databases
  hosts: dbservers
  remote_user: root

  roles:
    - db

The sample tasks mentioned in the playbook are:

---
# These tasks install http and the php modules.

- name: Install http and php etc
  yum: name={{ item }} state=present
  with_items:
   - httpd
   - php
   - php-mysql
   - git
   - libsemanage-python
   - libselinux-python

- name: insert iptables rule for httpd
  lineinfile: dest=/etc/sysconfig/iptables create=yes state=present
regexp="{{ httpd_port }}" insertafter="^:OUTPUT "
              line="-A INPUT -p tcp  --dport {{ httpd_port }} -j  ACCEPT"
  notify: restart iptables

- name: http service state
  service: name=httpd state=started enabled=yes
```

nsible Multi-Node Deployment Workflow (retrieved from sysadmincasts.com)

6장. 도구(Tools) 1. Config Mgmt

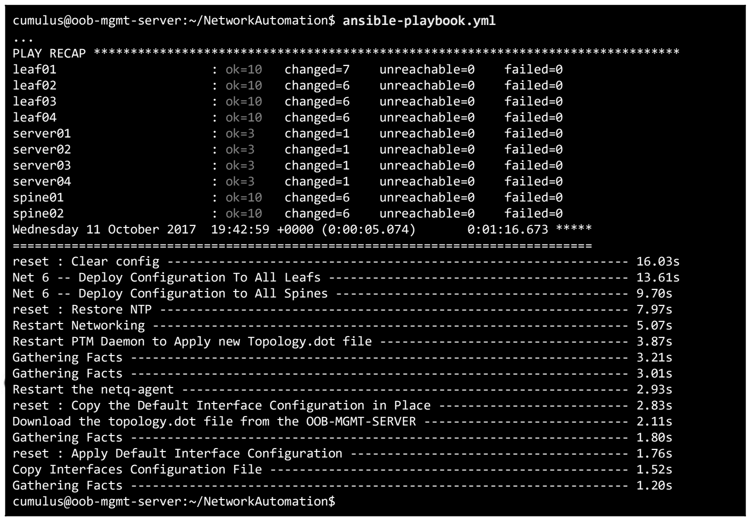
- **Ansible Template (Playbook)**
- **ansible-playbook.yml**



```
- name: Wipe Configuration
  hosts: spine,leaf
  become: yes
  gather_facts: no
  roles:
  - reset

- name: Lab 1 -- Setup PTM
  hosts: spine,leaf
  become: yes
  gather_facts: no
  tasks:
  - name: Download the topology.dot file from the OOB-MGMT-SERVER
    get_url:
      url: http://192.168.0.254/topology.dot
      dest: /etc/ptm.d/topology.dot
      mode: 0755

  - name: Restart PTM Daemon to Apply new Topology.dot file
    service: name=ptmd state=restarted


- name: Configure CLAG on Leaf Switches
  hosts: leaf01:leaf02:leaf03:leaf04
  become: yes
  vars:
   devices:
     leaf01:
       clag_role: "primary"
       clag_mac: "44:38:39:FF:00:01"
       clag_backupip: "192.168.0.12"
       loopback: "10.255.255.1/32"
       bgpasn: "65101"
       VLANS:
         - vlan: "10"
           vrrip: "10.0.10.1/24"
           vrrmac: "00:00:00:00:1a:10"
           bond: "SERVER01"
           interface: "swp1"
           realip: "10.0.10.2/24"
         - vlan: "20"
           vrrip: "10.0.20.1/24"
           vrrmac: "00:00:00:00:1a:20"
           bond: "SERVER02"
           interface: "swp2"
           realip: "10.0.20.2/24"
     leaf02:
       clag_role: "secondary"
       clag_mac: "44:38:39:FF:00:01"
       clag_backupip: "192.168.0.11"
       loopback: "10.255.255.2/32"
       bgpasn: "65102"
       VLANS:
         - vlan: "10"
           vrrip: "10.0.10.1/24"
           vrrmac: "00:00:00:00:1a:10"
           bond: "SERVER01"
           interface: "swp1"
           realip: "10.0.10.3/24"
         - vlan: "20"
           vrrip: "10.0.20.1/24"
           vrrmac: "00:00:00:00:1a:20"
           bond: "SERVER02"
           interface: "swp2"
           realip: "10.0.20.3/24"
     leaf03:
       clag_role: "primary"
       clag_mac: "44:38:39:FF:00:02"
       clag_backupip: "192.168.0.14"
       loopback: "10.255.255.3/32"
       bgpasn: "65103"
       VLANS:
         - vlan: "30"
           vrrip: "10.0.30.1/24"
           vrrmac: "00:00:00:00:1a:30"
           bond: "SERVER03"
           interface: "swp1"
           realip: "10.0.30.2/24"
         - vlan: "40"
           vrrip: "10.0.40.1/24"
           vrrmac: "00:00:00:00:1a:40"
           bond: "SERVER04"
           interface: "swp2"
           realip: "10.0.40.2/24"
     leaf04:
       clag_role: "secondary"
       clag_mac: "44:38:39:FF:00:02"
       clag_backupip: "192.168.0.13"
       loopback: "10.255.255.4/32"
       bgpasn: "65104"
       VLANS:
         - vlan: "30"
           vrrip: "10.0.30.1/24"
           vrrmac: "00:00:00:00:1a:30"
           bond: "SERVER03"
           interface: "swp1"
           realip: "10.0.30.3/24"
         - vlan: "40"
           vrrip: "10.0.40.1/24"
           vrrmac: "00:00:00:00:1a:40"
           bond: "SERVER04"
           interface: "swp2"
           realip: "10.0.40.3/24"

  tasks:
  - name: Lab 6 -- Deploy Configuration To All Leafs
    nclu:
      commit: true
      description: "Deploy interface configuration"
      commands:
        - add clag peer sys-mac {{ devices[ansible_hostname].clag_mac }} interface swp49-50 {{ devices[ansible_hostname].clag_role }} backup-ip {{ devices[ansible_hostname].clag_backupip }}
        - add clag port bond {{ devices[ansible_hostname].VLANS[0].bond}} interface {{ devices[ansible_hostname].VLANS[0].interface }} clag-id 1
        - add clag port bond {{ devices[ansible_hostname].VLANS[1].bond}} interface {{ devices[ansible_hostname].VLANS[1].interface }} clag-id 2
        - add bond {{ devices[ansible_hostname].VLANS[0].bond }} bridge access {{ devices[ansible_hostname].VLANS[0].vlan }}
        - add bond {{ devices[ansible_hostname].VLANS[1].bond }} bridge access {{ devices[ansible_hostname].VLANS[1].vlan }}
        - add vlan {{ devices[ansible_hostname].VLANS[0].vlan }} ip address {{ devices[ansible_hostname].VLANS[0].realip }}
        - add vlan {{ devices[ansible_hostname].VLANS[1].vlan }} ip address {{ devices[ansible_hostname].VLANS[1].realip }}
        - add vlan {{ devices[ansible_hostname].VLANS[0].vlan }} ip address-virtual {{ devices[ansible_hostname].VLANS[0].vrrmac }} {{ devices[ansible_hostname].VLANS[0].vrrip }}
        - add vlan {{ devices[ansible_hostname].VLANS[1].vlan }} ip address-virtual {{ devices[ansible_hostname].VLANS[1].vrrmac }} {{ devices[ansible_hostname].VLANS[1].vrrip }}
        - add loopback lo ip address {{ devices[ansible_hostname].loopback }}
        - add bgp autonomous-system {{ devices[ansible_hostname].bgpasn }}
        - add bgp neighbor swp51 interface remote-as external
        - add bgp neighbor swp52 interface remote-as external
        - add bgp network {{ devices[ansible_hostname].loopback }}
        - add bgp network {{ devices[ansible_hostname].VLANS[0].vrrip }}
        - add bgp network {{ devices[ansible_hostname].VLANS[1].vrrip }}
        - add bgp bestpath as-path multipath-relax

- name: Configure Spines
  hosts: spine
  become: yes
  vars:
   devices:
     spine01:
       loopback: "10.255.255.101/32"
       bgpasn: "65201"
     spine02:
       loopback: "10.255.255.102/32"
       bgpasn: "65201"
  tasks:

  - name: Lab 6 -- Deploy Configuration to All Spines
    nclu:
      commit: true
      description: "Deploy interface configuration"
      commands:
        - add loopback lo ip address {{ devices[ansible_hostname].loopback }}
        - add bgp autonomous-system {{ devices[ansible_hostname].bgpasn }}
        - add bgp neighbor swp1 interface remote-as external
        - add bgp neighbor swp2 interface remote-as external
        - add bgp neighbor swp3 interface remote-as external
        - add bgp neighbor swp4 interface remote-as external
        - add bgp network {{ devices[ansible_hostname].loopback }}
        - add bgp bestpath as-path multipath-relax

- name: Restart NetQ
  hosts: spine,leaf
  become: yes
  tasks:
  - name: Restart the netq-agent
    service: name=netq-agent state=restarted

- name: Server Configuration
  hosts: server
  become: yes
  tasks:
  - name: Copy Interfaces Configuration File
    copy: src=./{{ ansible_hostname }}.cfg dest=/etc/network/interfaces

  - name: Restart Networking
    service: name=networking state=restarted
```

6장. 도구(Tools) 1. Config Mgmt

□ **Operation for Ansible "ansible-playbook.yml"**



```
cumulus@oob-mgmt-server:~/NetworkAutomation$ ansible-playbook.yml
...
PLAY RECAP ****************************************************************************
leaf01                      : ok=10   changed=7    unreachable=0    failed=0
leaf02                      : ok=10   changed=6    unreachable=0    failed=0
leaf03                      : ok=10   changed=6    unreachable=0    failed=0
leaf04                      : ok=10   changed=6    unreachable=0    failed=0
server01                    : ok=3    changed=1    unreachable=0    failed=0
server02                    : ok=3    changed=1    unreachable=0    failed=0
server03                    : ok=3    changed=1    unreachable=0    failed=0
server04                    : ok=3    changed=1    unreachable=0    failed=0
spine01                     : ok=10   changed=6    unreachable=0    failed=0
spine02                     : ok=10   changed=6    unreachable=0    failed=0
Wednesday 11 October 2017  19:42:59 +0000 (0:00:05.074)       0:01:16.673 *****
===============================================================================
reset : Clear config -------------------------------------------------------------- 16.03s
Net 6 -- Deploy Configuration To All Leafs ---------------------------------------- 13.61s
Net 6 -- Deploy Configuration to All Spines --------------------------------------- 9.70s
reset : Restore NTP --------------------------------------------------------------- 7.97s
Restart Networking ---------------------------------------------------------------- 5.07s
Restart PTM Daemon to Apply new Topology.dot file --------------------------------- 3.87s
Gathering Facts ------------------------------------------------------------------- 3.21s
Gathering Facts ------------------------------------------------------------------- 3.01s
Restart the netq-agent ------------------------------------------------------------ 2.93s
reset : Copy the Default Interface Configuration in Place ------------------------- 2.83s
Download the topology.dot file from the OOB-MGMT-SERVER --------------------------- 2.11s
Gathering Facts ------------------------------------------------------------------- 1.80s
reset : Apply Default Interface Configuration ------------------------------------- 1.76s
Copy Interfaces Configuration File ------------------------------------------------ 1.52s
Gathering Facts ------------------------------------------------------------------- 1.20s
cumulus@oob-mgmt-server:~/NetworkAutomation$
```

JS Lab

299

6장. 도구(Tools) 1. Config Mgmt

□ **Benefits of Using Ansible**

- **It is an easy-to-use open source configuration management tool.**
- **It is an agentless tool.**
- **It automates cloud provisioning, application deployment, orchestration, etc.**
- **It provides consistent, reliable, and secure management.**
- **It is supported by a large and active community of developers.**
- **It has a low learning curve.**
- **It provides role-based access control.**
- **It is available for all major operating systems.**

JS Lab

300

6장. 도구(Tools) 1. Config Mgmt

- Puppet
  - **Puppet is an open source configuration management tool. It mostly uses the agent/master (client/server) model to configure the systems. The agent is referred to as the Puppet Agent and the master is referred to as the Puppet Master. The Puppet Agent can also work locally and is then referred to as Puppet Apply.**
  - **Besides Puppet, the company also has an enterprise product called Puppet Enterprise and provides services and training around that as well.**

6장. 도구(Tools) 1. Config Mgmt

- Puppet Agent
  - **We need to install Puppet Agent on each system we want to manage/configure with Puppet. Each agent:**
    1. Connects securely to the Puppet Master to get the series of instructions in a file referred to as the Catalog File.
    2. Performs operations from the Catalog File to get to the desired state.
    3. Sends back the status to the Puppet Master.
  - **Puppet Agent can be installed on the following platforms:**
    1. Linux
    2. Windows
    3. Mac OSX.

6장. 도구(Tools) 1. Config Mgmt

◻ Puppet Master

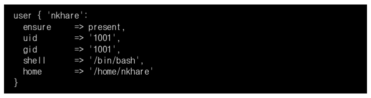
- **Compiles the Catalog File for hosts based on the system, configuration, manifest file, etc.**
- **Sends the Catalog File to agents when they query the master.**
- **Has information about the entire environment, such as host information, metadata (like authentication keys), etc.**
- **Gathers reports from each agent and then prepares the overall report.**

303

6장. 도구(Tools) 1. Config Mgmt

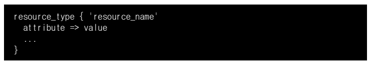
◻ The Catalog File

- **Puppet prepares a Catalog File based on the manifest file. A manifest file is created using the Puppet Code:**



```
user { 'nkhare':
  ensure     => present,
  uid        => '1001',
  gid        => '1001',
  shell      => '/bin/bash',
  home       => '/home/nkhare'
}
```

- **which defines and creates a user nkhare with:**
  1. UID/GID as 1001
  2. The login shell is set to /bin/bash
  3. The home directory is set to /home/nkhare.
- **A manifest file can have one or more sections of code, like we exemplified above, and each of these sections of code can have a signature like the following:**



```
resource_type { 'resource_name'
  attribute => value
  ...
}
```

304

6장. 도구(Tools) 1. Config Mgmt

- Puppet Tools
  - **Puppet also has nice tooling around it, like:**
    1. Centralized reporting (PuppetDB), which helps us generate reports, search a system, etc.
    2. Live system management
    3. Puppet Forge, which has ready-to-use modules for manifest files from the community.

305

6장. 도구(Tools) 1. Config Mgmt

- Benefits of Using Puppet
  - **It is an open source configuration management tool.**
  - **It provides scalability, automation, and centralized reporting.**
  - **It is available on all major operating systems.**
  - **It provides role-based access control.**

306

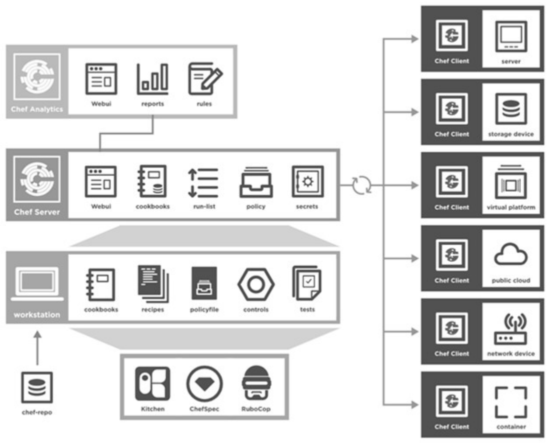
6장. 도구(Tools) 1. Config Mgmt

□ **Chef**

- **Develop cookbooks and recipes.**
- **Synchronize chef-repo with the version control system.**
- **Run command line tools.**
- **Configure policy, roles, etc.**
- **Interact with nodes to do a one-off configuration.**

6장. 도구(Tools) 1. Config Mgmt

□ **Chef Overview**

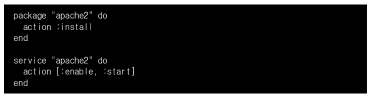


Chef Overview (by Chef Software, Inc./CC BY-SA 3.0, retrieved from chef.io)

6장. 도구(Tools) 1. Config Mgmt

Chef Cookbook

- **Recipes:** A recipe is the most fundamental unit for configuration, which mostly contains resources, resource names, attribute-value pairs, and actions.



```
package "apache2" do
  action :install
end

service "apache2" do
  action [:enable, :start]
end
```

- **Attributes:** An attribute helps us define the state of the node. After each chef-client run, the node's state is updated on the Chef Server
- **Knife:** provides an interface between a local chef-repo and the Chef Server

6장. 도구(Tools) 1. Config Mgmt

Supported Platforms

- **Chef supports the following platforms for Chef Client:**
  1. Unix-based systems
  2. Mac OS X
  3. Windows
  4. Cisco IOS XR and NX-OS.
- **Chef Server is supported on the following platforms:**
  1. Red Hat Enterprise Linux
  2. Ubuntu Linux.
- **Chef also has a GUI built on top of Chef Server, which can help ups running the cookbook from the browser, prepare reports, etc.**

6장. 도구(Tools) 1. Config Mgmt

Benefits of Using Chef

- Chef is an open source systems integration framework.
- It provides automation, scalability, High Availability and consistency in deployment.
- It is available for all major operating systems.
- It provides role-based access control.
- It provides real-time visibility with Chef Analytics.

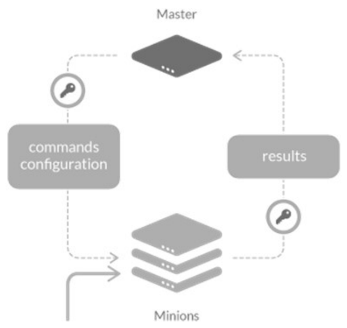
6장. 도구(Tools) 1. Config Mgmt

Salt Open

- Salt Open is an open source configuration management system built on top of a remote execution framework. It uses the client/server model, where the server sends commands and configurations to all the clients in a parallel manner, which the clients run, returning back the status.
- SaltStack, Inc., which stands behind Salt Open, offers also an enterprise product called SaltStack Enterprise.

6장. 도구(Tools) 1. Config Mgmt

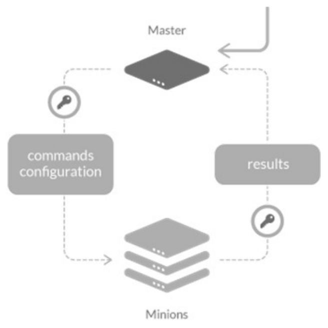
- Salt Minions
  - **Minions can be installed on:**
    1. Unix-based systems
    2. Windows
    3. Mac OS X.



Salt Minions (by SaltStack, Inc., retrieved from saltstack.com)

6장. 도구(Tools) 1. Config Mgmt

- Salt Masters
  - **A server is referred to as a Salt master. Multi-master is also supported.**



Salt Master (by SaltStack, Inc., retrieved from saltstack.com)

6장. 도구(Tools) 1. Config Mgmt

- Other Components: Modules, Returners, Grains, Pillar Data
  - Remote execution is based on Modules and Returners.
  - Modules provide basic functionality, like installing packages, managing files, managing containers, etc. All support modules are listed in the Salt documentation. We can also write custom modules.
  - With Returners, we can save a minion's response on the master or other locations. We can use default Returners or write custom ones.
  - All the information collected from minions is saved on the master. The collected information is referred to as Grains. Private information like cryptographic keys and other specific information about each minion which the master has is referred to as Pillar Data. Pillar Data is shared between the master and the individual minion.
  - By combining Grains and Pillar Data, the master can target specific minions to run commands on them. For example, we can query the hostname of all the nodes where the installed OS is Fedora 23.
  - With the above tools and information in hand, the master can easily set up a minion with a specific state. This can also be referred to as Configuration Management. Salt has different State Modules to manage a state.

6장. 도구(Tools) 1. Config Mgmt

- Benefits of Using Salt Open
  - It is an open source configuration management system.
  - It provides automation, High Availability and an event-driven infrastructure.
  - It provides role-based access control.
  - It supports agent and agentless deployments.
  - It is available for all major operating systems.

6장. 도구(Tools) 2. Build & Release

- Infrastructure as a Code helps us create a near production-like environment for development, staging, etc. With some tooling around them, we can also create the same environments on different cloud providers.
- By combining Infrastructure as a Code with versioned software, we are guaranteed to have a reproducible build and release environment every time. In this chapter, we will take a look into two such tools: Terraform and BOSH.

317

6장. 도구(Tools) 2. Build & Release

- **Terraform**
  - Terraform is a tool that allows us to define the infrastructure as code. This helps us deploy the same infrastructure on VMs, bare metal or cloud. It helps us treat the infrastructure as software. The configuration files can be written in HCL (HashiCorp Configuration Language).

Terraform

```
provider "nks" {
  # Set environment variable NKS_API_TOKEN with your API token from NKS
  # Set environment variable NKS_API_URL with API endpoint,
  # defaults to NKS production enviroment.
}
  # Organization
data "nks_organization" "default" {
   name = "${var.organization_name}"
```

318

6장. 도구(Tools) 2. Build & Release

□ **Terraform Providers**

- **IaaS: AWS, DigitalOcean, GCE, OpenStack, etc.**
- **PaaS: Heroku, Cloud Foundry, etc.**
- **SaaS: Atlas, DNSimple, etc.**

6장. 도구(Tools) 2. Build & Release

□ **Features**

- **Infrastructure as Code:** Infrastructure is described using a high-level configuration syntax. This allows a blueprint of your datacenter to be versioned and treated as you would any other code. Additionally, infrastructure can be shared and re-used.
- **Execution Plans:** Terraform has a "planning" step where it generates an execution plan. The execution plan shows what Terraform will do when you call apply. This lets you avoid any surprises when Terraform manipulates infrastructure.
- **Resource Graph:** Terraform builds a graph of all your resources, and parallelizes the creation and modification of any non-dependent resources. Because of this, Terraform builds infrastructure as efficiently as possible, and operators get insight into dependencies in their infrastructure.
- **Change Automation:** Complex changesets can be applied to your infrastructure with minimal human interaction. With the previously mentioned execution plan and resource graph, you know exactly what Terraform will change and in what order, avoiding many possible human errors"

6장. 도구(Tools) 2. Build & Release

- Benefits of Using Terraform
  - It allows to build, change, and version infrastructure in a safe and efficient manner.
  - It can manage existing, as well as customized service providers. It is agnostic to underlying providers.
  - It can manage a single application or an entire datacenter.
  - It is a flexible abstraction of resources.

321

6장. 도구(Tools) 2. Build & Release

- BOSH
- According to bosh.io, "BOSH is an open source tool for release engineering, deployment, lifecycle management, and monitoring of distributed systems".
  - Amazon Web Services EC2
  - OpenStack
  - VMware vSphere
  - vCloud Director.
- With the Cloud Provider Interface (CPI), BOSH supports additional IaaS providers such as Google Compute Engine and Apache CloudStack.

322

6장. 도구(Tools)　　2. Build & Release

Key Concepts

- **Stemcell:** A Stemcell is a versioned, IaaS-specific, operating system image with some pre-installed utilities (e.g. BOSH Agent). Stemcells do not contain any application-specific code. The BOSH team is in charge of releasing stemcells, which are listed on the "Stemcells" web page.
- **Release:** A Release is placed on top of a Stemcell, and consists of a versioned collection of configuration properties, templates, scripts, source code, etc., to build and deploy software.
- **Deployment:** A Deployment is a collection of VMs which are built from Stemcells, populated with specific Releases on top of them and having disks to keep persistent data.
- **BOSH Director:** The BOSH Director is the orchestrator component of BOSH, which controls the VM creation and deployment. It also controls software and service lifecycle events. We need to upload Stemcells, Releases and Deployment manifest files to the Director. The Director processes the manifest file and does the deployment.

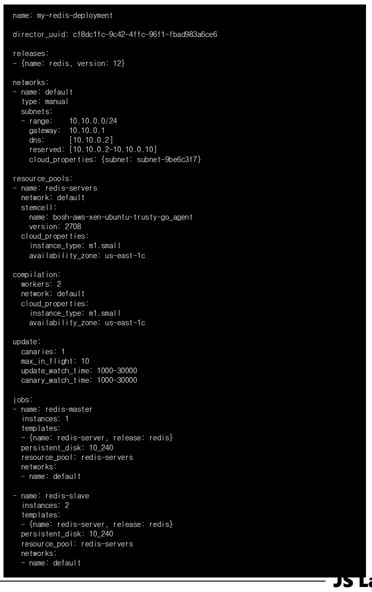
JS Lab

323

6장. 도구(Tools)　　2. Build & Release

Sample Deployment

- **Next, we will reproduce an example of a sample deployment manifest from the BOSH website. This sample deployment manifest must be then uploaded to the BOSH Director:**



```
name: my-redis-deployment

director_uuid: cf8dc1fc-9c42-4ffc-96f1-fbad983a6ce6

releases:
- {name: redis, version: 12}

networks:
- name: default
  type: manual
  subnets:
  - range:    10.10.0.0/24
    gateway:  10.10.0.1
    dns:      [10.10.0.2]
    reserved: [10.10.0.2-10.10.0.10]
    cloud_properties: {subnet: subnet-9be6c3f7}

resource_pools:
- name: redis-servers
  network: default
  stemcell:
    name: bosh-aws-xen-ubuntu-trusty-go_agent
    version: 2708
  cloud_properties:
    instance_type: m1.small
    availability_zone: us-east-1c

compilation:
  workers: 2
  network: default
  cloud_properties:
    instance_type: m1.small
    availability_zone: us-east-1c

update:
  canaries: 1
  max_in_flight: 10
  update_watch_time: 1000-30000
  canary_watch_time: 1000-30000

jobs:
- name: redis-master
  instances: 1
  templates:
  - {name: redis-server, release: redis}
  persistent_disk: 10_240
  resource_pool: redis-servers
  networks:
  - name: default

- name: redis-slave
  instances: 2
  templates:
  - {name: redis-server, release: redis}
  persistent_disk: 10_240
  resource_pool: redis-servers
  networks:
  - name: default
```

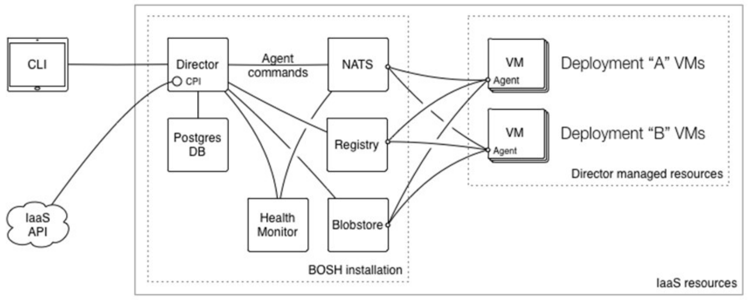
JS Lab

324

6장. 도구(Tools) 2. Build & Release

□ **Benefits of Using BOSH**

- **It is an open source tool "for release engineering, deployment, lifecycle management, and monitoring of distributed systems" (bosh.io).**
- **It supports IaaS providers like AWS, OpenStack, VMware vSphere, Google Compute Engine, etc.**



JS Lab

325

6장. 도구(Tools) 2. Build & Release

□ **관리 표준과 오픈소스**

표준	오픈소스
OASIS Advancing open standards for the information society	OPEN-O
OASIS TOSCA — Topology and Orchestration Specification for Cloud Applications (TOSCA)	Open Source MANO
tmforum	CLOUDIFY
MEF METRO ETHERNET FORUM	ARIA
ETSI	ONAP OPEN NETWORK AUTOMATION PLATFORM
IETF	

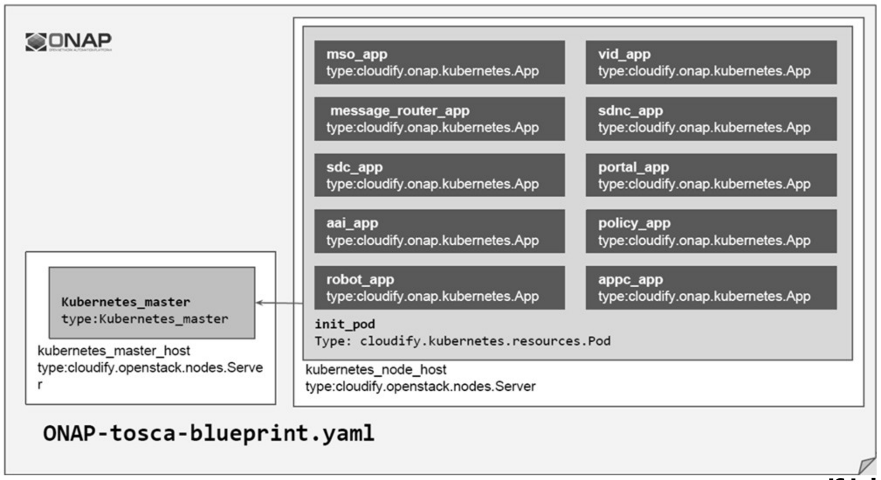
JS Lab

326

6장. 도구(Tools)

2. Build & Release

- Telco용 멀티벤더 환경 관리 표준 TOSCA
- ONAP TOSCA 탬플릿 (예: Public Cloud, Private Cloud 적용가능)


ONAP
mso_app
type:cloudify.onap.kubernetes.App
vid_app
type:cloudify.onap.kubernetes.App
message_router_app
type:cloudify.onap.kubernetes.App
sdnc_app
type:cloudify.onap.kubernetes.App
sdc_app
type:cloudify.onap.kubernetes.App
portal_app
type:cloudify.onap.kubernetes.App
aai_app
type:cloudify.onap.kubernetes.App
policy_app
type:cloudify.onap.kubernetes.App
robot_app
type:cloudify.onap.kubernetes.App
appc_app
type:cloudify.onap.kubernetes.App
init_pod
Type: cloudify.kubernetes.resources.Pod
Kubernetes_master
type:Kubernetes_master
kubernetes_master_host
type:cloudify.openstack.nodes.Server
kubernetes_node_host
type:cloudify.openstack.nodes.Server
ONAP-tosca-blueprint.yaml

https://gerrit.onap.org/r/gitweb?p=oom.git;a=blob;f=onap-blueprint.yaml

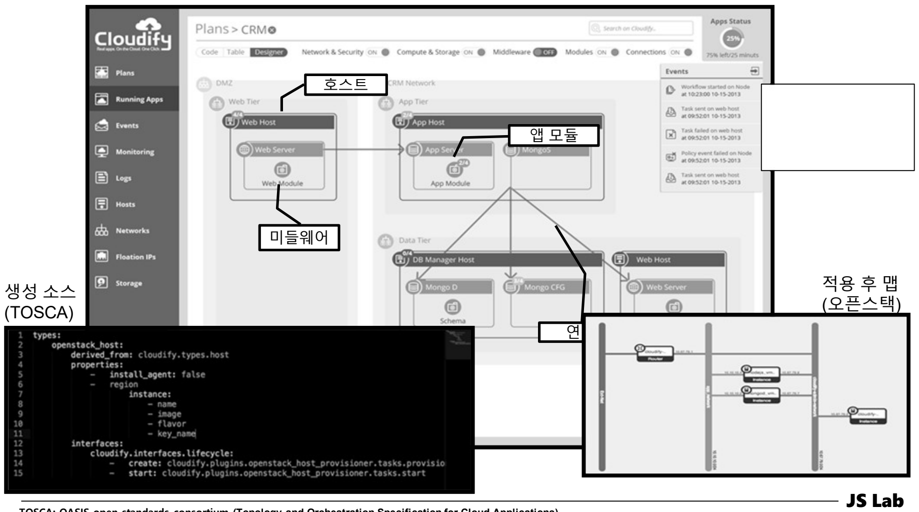
JS Lab

327

6장. 도구(Tools)

2. Build & Release

- **멀티 클라우드 오케스트레이션 :** 표준 TOSCA 기반 GUI 서비스 (TOSCA 표준 적용 오픈소스 Cloudify 예)



```
types:
    openstack_host:
        derived_from: cloudify.types.host
        properties:
            -   install_agent: false
            -   region
                instance:
                    - name
                    - image
                    - flavor
                    - key_name
        interfaces:
            cloudify.interfaces.lifecycle:
                -   create: cloudify.plugins.openstack_host_provisioner.tasks.provisio
                -   start: cloudify.plugins.openstack_host_provisioner.tasks.start
```

TOSCA: OASIS open standards consortium (Topology and Orchestration Specification for Cloud Applications)

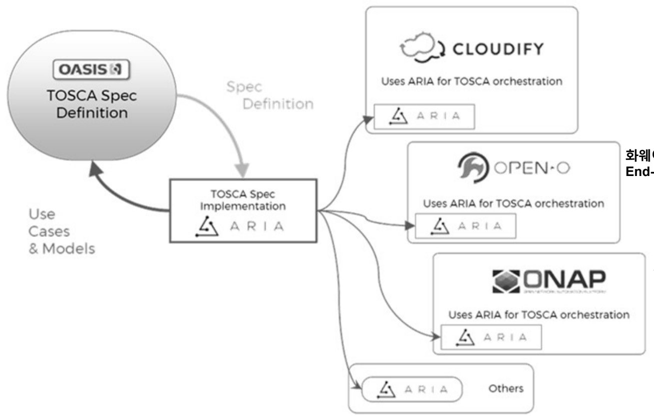
JS Lab

328

6장. 도구(Tools) 2. Build & Release

▫ 표준 TOSCA 스펙 적용 오픈소스 'ARIA'

오케스트레이션이 TOSCA 프로파일 지원을 위한 Python 라이브러리
TOSCA 애플리케이션 생성을 위한 SDK
CLI Tools: 오케스트레이션을 위한 TOSCA 템플릿


OASIS
TOSCA Spec Definition
Spec Definition
Use Cases & Models
TOSCA Spec Implementation
ARIA
CLOUDIFY
Uses ARIA for TOSCA orchestration
OPEN-O
Uses ARIA for TOSCA orchestration
ONAP
Uses ARIA for TOSCA orchestration
ARIA
Others

화웨이 제안 - Telco 클라우드의 End-to-End 오케스트레이션

AT&T의 ECOMP (Enhanced Control, Orchestration, Management & Policy)

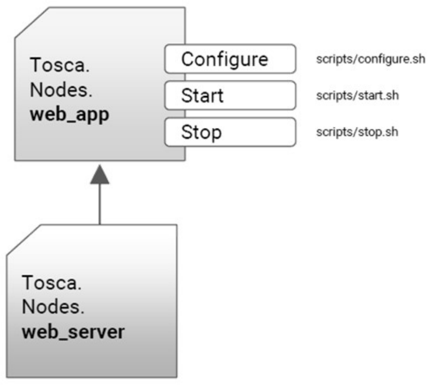
JS Lab

329

6장. 도구(Tools) 2. Build & Release

▫ 표준 TOSCA 적용 'ARIA' Hello World 예 (TOSCA/ARIA)

- 소스 공개
- helloworld.yaml



```
tosca_definitions_version: tosca_simple_yaml_1_0
node_types:
  HelloWorld:
    derived_from: tosca:WebApplication
    requirements:
      - host:
          # Override to allow for 0 occurrences
          capability: tosca:Container
          occurrences: [ 0, UNBOUNDED ]
topology_template:
  inputs:
  node_templates:
    hello_world:
      type: HelloWorld
      capabilities:
        app_endpoint:
          properties:
            protocol: http
            port: 9090
      interfaces:
        Standard:
          configure: scripts/configure.sh
          start: scripts/start.sh
          stop: scripts/stop.sh
  outputs:
    port:
      type: integer
value: { get_property: [ hello_world, app_endpoint, port ] }
```

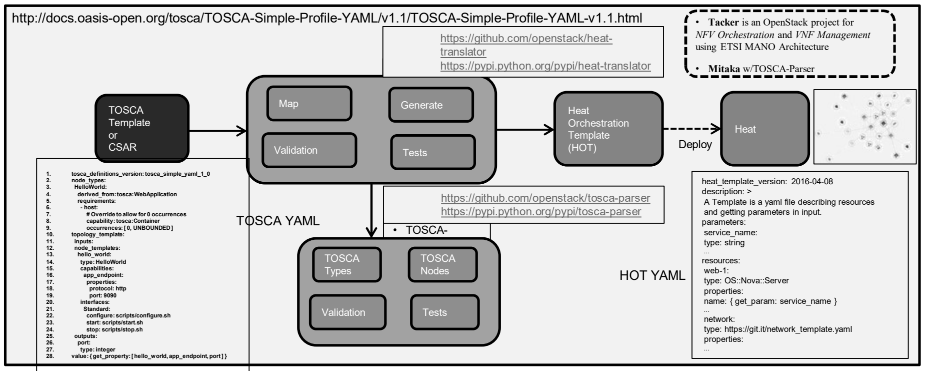
https://github.com/apache/incubator-ariatosca/blob/master/examples/hello-world/hello-world.yaml

JS Lab

330

6장. 도구(Tools) 2. Build & Release

- **TOSCA 처리 과정 (오픈스택 예)**
  - **TOSCA Simple Profile for Network Functions Virtualization (NFV)**
  - **OSCA Simple Profile in YAML Version 1.1** (30 January 2018)
  - **토스카 파서(TOSCA-Parser):** Parser for TOSCA Simple Profile in YAML
  - **히트번역기(Heat-Translator):** An OpenStack project to map and translate non-Heat (e.g. TOSCA) templates to Heat Orchestration Template (HOT)


http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.1/TOSCA-Simple-Profile-YAML-v1.1.html
https://github.com/openstack/heat-translator
https://pypi.python.org/pypi/heat-translator
TOSCA Template or CSAR
Map
Generate
Validation
Tests
Heat Orchestration Template (HOT)
Deploy
Heat
https://github.com/openstack/tosca-parser
https://pypi.python.org/pypi/tosca-parser
TOSCA YAML
• TOSCA-
TOSCA Types
TOSCA Nodes
Validation
Tests
HOT YAML

- **Tacker** is an OpenStack project for *NFV Orchestration* and *VNF Management* using ETSI MANO Architecture
- **Mitaka** w/TOSCA-Parser

```
tosca_definitions_version: tosca_simple_yaml_1_0
node_types:
  HelloWorld:
    derived_from: tosca:WebApplication
    requirements:
      - host:
          # Override to allow for 0 occurrences
          capability: tosca:Container
          occurrences: [ 0, UNBOUNDED ]
topology_template:
  inputs:
  node_templates:
    hello_world:
      type: HelloWorld
      capabilities:
        app_endpoint:
          properties:
            protocol: http
            port: 9090
      interfaces:
        Standard:
          configure: scripts/configure.sh
          start: scripts/start.sh
          stop: scripts/stop.sh
  outputs:
    port:
      type: integer
value: { get_property: [ hello_world, app_endpoint, port ] }
```

```
heat_template_version:  2016-04-08
description: >
 A Template is a yaml file describing resources
 and getting parameters in input.
parameters:
 service_name:
 type: string
 ...
resources:
 web-1:
 type: OS::Nova::Server
 properties:
 name: { get_param: service_name }
 ...
 network:
 type: https://git.it/network_template.yaml
 properties:
 ...
```

CSAR (Cloud Service Archive)   TOSCA (Topology and Orchestration Specification for Cloud Applications)

JS Lab

331

6장. 도구(Tools) 3. Key-Value Pair

- **The Key-Value Pair Storage provides the functionality to store or retrieve the value of a key. Most of the Key Value stores provide REST APIs to do operations like GET, PUT, DELETE etc., which helps when doing all the operations over HTTP. Some examples of Key-Value stores are:**
  - **etcd**
  - **Consul.**

JS Lab

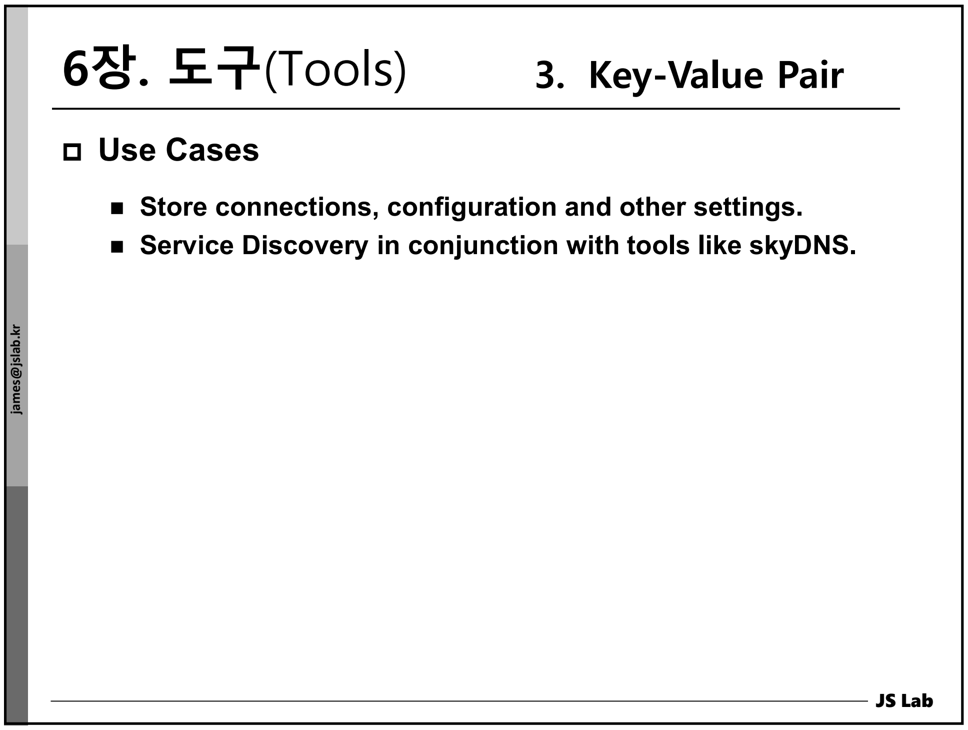
332

6장. 도구(Tools) 3. Key-Value Pair

- etcd
- etcd is an open source distributed key-value pair storage, which is based on the Raft consensus algorithm. It was started by CoreOS and it is written in Go.

6장. 도구(Tools) 3. Key-Value Pair

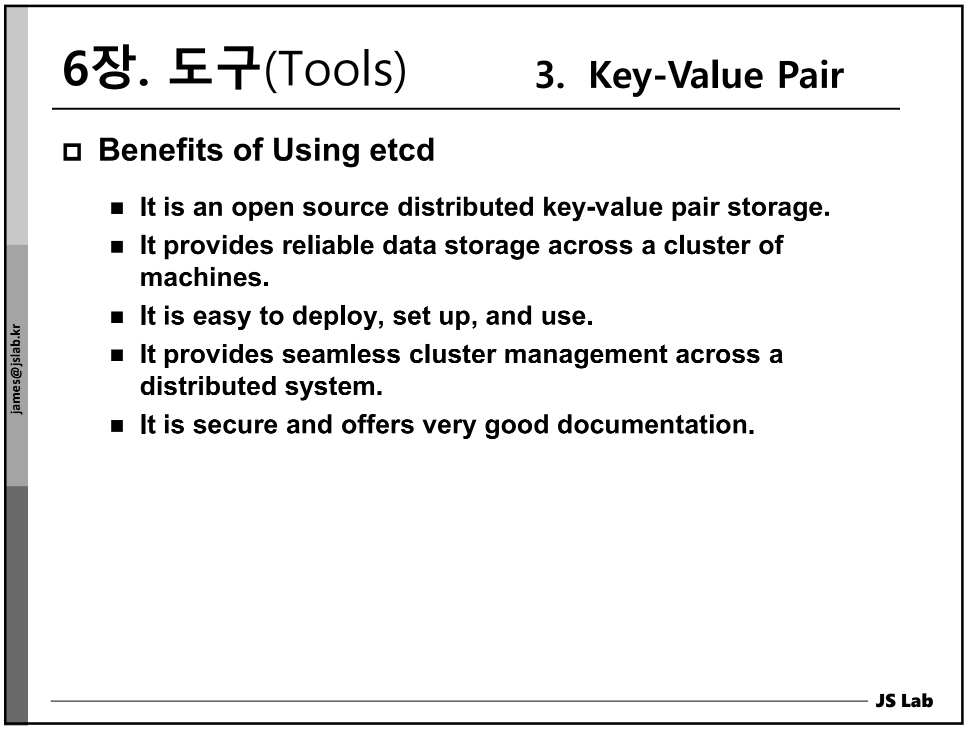
- Features
  - etcd can be configured to run standalone or in a cluster. In a cluster mode, it can gracefully handle the master election during network partitions and can tolerate machine failures, including the master.
  - We can also watch on a value of a key, which allows us to do certain operations based on the value change.
  - It is currently being used in many projects like Kubernetes, Fleet, Locksmith, vulcand.



6장. 도구(Tools) 3. Key-Value Pair

- Use Cases
  - Store connections, configuration and other settings.
  - Service Discovery in conjunction with tools like skyDNS.

335



6장. 도구(Tools) 3. Key-Value Pair

- Benefits of Using etcd
  - It is an open source distributed key-value pair storage.
  - It provides reliable data storage across a cluster of machines.
  - It is easy to deploy, set up, and use.
  - It provides seamless cluster management across a distributed system.
  - It is secure and offers very good documentation.

336

6장. 도구(Tools) 3. Key-Value Pair

◻ Consul

- **Consul is a distributed, highly-available system which can be used for service discovery and configuration.**
- **Other than providing a distributed key-value store, it also provides features like:**
  1. Service discovery in conjunction with DNS or HTTP
  2. Health checks for services and nodes
  3. Multi-datacenter support.

337

6장. 도구(Tools) 3. Key-Value Pair

◻ Use Cases

- **Store connections, configuration and other settings.**
- **Service discovery in conjunction with DNS or HTTP.**

338

6장. 도구(Tools) 3. Key-Value Pair

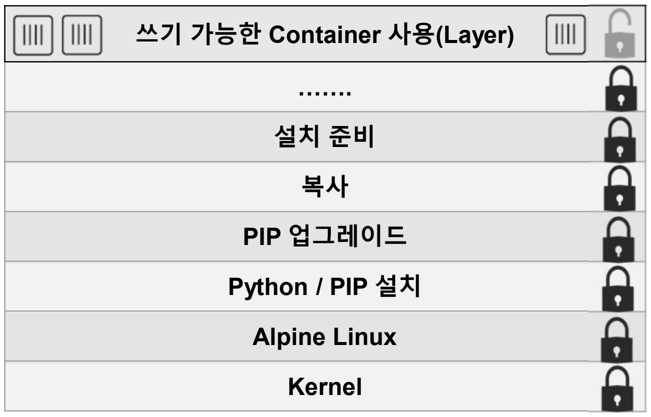
- Benefits of Using Consul
  - It is a distributed, highly-available system which can be used for service discovery and configuration.
  - It provides health checks for services and nodes.
  - It provides out-of-the-box native multi-datacenter support.
  - It implements embedded service discovery.

6장. 도구(Tools) 4. Image Building

- In an immutable infrastructure environment we prefer to replace an existing service with a new one, to fix any problems/bugs or to perform an update.
- To start a new service or replace an older service with a new one, we need the image from which the service can be started. These images should be created in an automated fashion. In this section, we will take a look into the creation of Docker images and VM images for different cloud platforms using Packer.

6장. 도구(Tools) 4. Image Building

- 이미지: 도커 컨테이너의 기반이며, 앱을 표현함
- 이미지 레이어

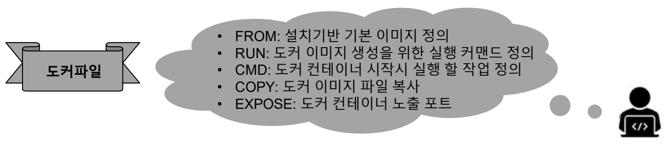


341

6장. 도구(Tools) 4. Image Building

- Dockerfiles
- FROM, MAINTAINER, RUN, EXPOSE, and CMD are different kinds of instructions which are followed by arguments. The instructions are well documented in the Docker Documentation.

```
FROM fedora
MAINTAINER Neependra Khare <neependra.khare@gmail.com>
RUN dnf -y update && dnf clean all
RUN dnf -y install nginx && dnf clean all
RUN echo "daemon off;" >> /etc/nginx/nginx.conf
RUN echo "nginx on Fedora" > /usr/share/nginx/html/index.html

EXPOSE 80

CMD [ "/usr/sbin/nginx" ]
```



- FROM: 설치기반 기본 이미지 정의
- RUN: 도커 이미지 생성을 위한 실행 커맨드 정의
- CMD: 도커 컨테이너 시작시 실행 할 작업 정의
- COPY: 도커 이미지 파일 복사
- EXPOSE: 도커 컨테이너 노출 포트

342

6장. 도구(Tools) 4. Image Building

- **Multi-Stage Dockerfile**
- **The COPY --from=buildstep line copies only the built artifact from the previous stage into this new stage. The C SDK and any intermediate artifacts are not copied in the final image.**

```
# stage - 1
FROM ubuntu AS buildstep
RUN apt-get update && apt-get install -y build-essential gcc
COPY hello.c /app/hello.c
WORKDIR /app
RUN gcc -o hello hello.c && chmod +x hello

# stage - 2
FROM ubuntu
RUN mkdir -p /usr/src/app/
WORKDIR /usr/src/app
COPY --from=buildstep /app/hello ./hello
COPY ./start.sh ./start.sh
ENV INITSYSTEM=on
CMD ["bash", "/usr/src/app/start.sh"]
```

6장. 도구(Tools) 4. Image Building

- **Packer**
- **Packer from HashiCorp is an open source tool for creating virtual images from a configuration file for different platforms.**

```
{
  "variables": {
    "aws_access_key": "abc",
    "aws_secret_key": "xyz",
    "atlas_token": "123"
  },
  "builders": [{
    "type": "amazon-ebs",
    "access_key": "{{user `aws_access_key`}}",
    "secret_key": "{{user `aws_secret_key`}}",
    "region": "us-west-2",
    "source_ami": "ami-9abea4fb",
    "instance_type": "t2.micro",
    "ssh_username": "ubuntu",
    "ami_name": "packer-example {{timestamp}}"
  }, {
  "type": "googlecompute",
  "account_file": "user.json",
  "project_id": "useapp",
  "source_image": "ubuntu-1404-trusty-v20160114e",
  "zone": "us-central1-a",
  "image_name": "myimage"
  }],
  "provisioners": [{
    "type": "shell",
    "inline": [
      "sleep 30",
      "#!/bin/bash",
      "sudo apt-get -y update",
      "sudo apt-get -y install apache2",
      .....
      .....
    ]
  }],
  "post-processors": [{
        "type": "atlas",
        "only": ["amazon-ebs"],
        "token": "{{user `atlas_token`}}",
        "artifact": "user/stack",
        "artifact_type": "amazon.image",
        "metadata": {
          "created_at": "{{timestamp}}"
        }
      }, {
        "type": "atlas",
        "only": ["googlecompute"],
        "token": "{{user `atlas_token`}}",
        "artifact": "user/stack",
        "artifact_type": "googlecompute.image",
        "metadata": {
          "created_at": "{{timestamp}}"
        }
  }]
}
```

6장. 도구(Tools) 4. Image Building

◻ Steps to Create Virtual Images

- **Building the base image:** This is defined under the builders section of the configuration file. Packer supports the following platforms: Amazon EC2 (AMI), DigitalOcean, Docker, Google Compute Engine, OpenStack, Parallels (PVM), QEM, VirtualBox (OVF), VMware (VMX). Other platforms can be added via plugins. In the example we provided, we have created images for Amazon EC2 and Google Compute Engine.
- **Provision the base image to do configuration:** Once we built a base image, we can then provision it to do further configuration changes, install software, etc. Packer supports different provisioners like Shell, Ansible, Puppet, Chef, etc. In the example provided, we are doing provisioning with Shell.
- **Perform post-build operations:** In the post-operations, we can copy/move the resulted image to a central repository, create a Vagrant box, etc.

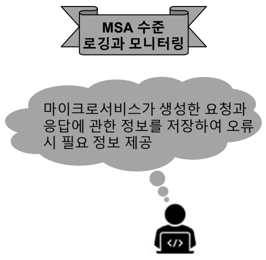
6장. 도구(Tools) 4. Image Building

◻ Benefits of Using Packer

- **It is an open source tool for creating virtual images from a configuration file for different platforms.**
- **It is easy to use.**
- **It automates the creation of images.**
- **It provides an extremely fast infrastructure deployment, from which both development and production are benefiting.**
- **It offers multi-provider portability.**

6장. 도구(Tools) 5. Debug Log Monitor

- **When we experience problems in development or production, we resort to debugging, logging and monitoring tools to find the root cause of the problem. Some of the tools which we should be familiar with are:**
  - **strace**
  - **SAR (System Activity Reporter)**
  - **tcpdump**
  - **GDB (GNU Project Debugger)**
  - **syslog**
  - **Nagios**
  - **Zabbix.**


MSA 수준
로깅과 모니터링
마이크로서비스가 생성한 요청과 응답에 관한 정보를 저장하여 오류 시 필요 정보 제공

347

6장. 도구(Tools) 5. Debug Log Monitor

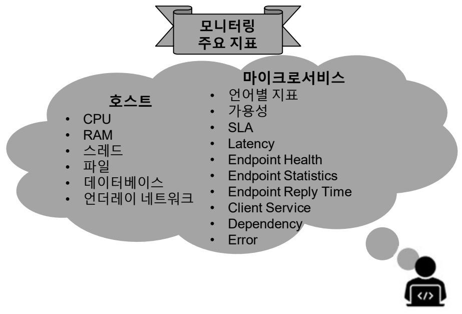
- **We can use the same tools on bare metal and VMs, but containers bring interesting challenges:**
  - **Containers are ephemeral, so, when they die, all the metadata (e.g. logs) gets deleted as well, unless we store it in some other location.**
  - **Containers do not have kernel space components.**
  - **We want to have a container's footprint as low as possible. Installing debugging and monitoring tools increases the footprint size.**
  - **Collecting per container statistics, debugging information individually and then analyzing data from multiple containers is a tedious process.**

348

6장. 도구(Tools)

5. Debug Log Monitor

- Below are some of the tools which we can use for containerized applications:
  - **Debugging:** Docker CLI, Sysdig
  - **Logging:** Docker CLI, Docker Logging Driver
  - **Monitoring:** Docker CLI, Sysdig, cAdvisor/Heapster, Prometheus, Datadog, New Reli



349

6장. 도구(Tools)

5. Debug Log Monitor

- Native Docker Features for Debugging
  - **Debugging:**
    1. docker inspect
    2. docker logs
  - **Logging:**
    1. docker logs
    2. Docker Logging Drivers: With the logging driver we can choose a Docker daemon wide or per container logging policy. Depending on the policy, Docker forwards the logs to the corresponding drivers. Docker supports the following drivers: jsonfile, syslog, journald, gelf (Graylog Extended Log Format), fluentd, awslogs, splunk. Once the logs are saved in a central location, we can use the respective tools to get the insights.
  - **Monitoring:**
    1. docker stats
    2. docker top

350

6장. 도구(Tools) 5. Debug Log Monitor

- Sysdig provides an on-cloud and on-premise platform for container security, monitoring and forensics. According to sysdig.com, sysdig is
  - "strace + tcpdump + htop + iftop + lsof + awesome sauce".

351

6장. 도구(Tools) 5. Debug Log Monitor

- **It has two open source tools along with their paid enterprise class offerings.**
  - **Sysdig:** It saves low-level system information from the running Linux instance, on which we can apply filters and do further analysis.
  - **Sysdig Monitor:** It is a paid offering that provides additional features on top of the open source version.
  - **Sysdig Falco:** It is a container-native tool which can help us gain visibility of containers and applications down to the finest details. It can collect information at system, network and file level. With rule-sets, we can provide our container security information and then take action based on them. For example, if a container does not satisfy the security requirements, Falco can kill the container, notify someone, etc.
  - **Sysdig Secure:** It is also a paid offering that provides additional features on top of the open source version.

352

6장. 도구(Tools) 5. Debug Log Monitor

- Features of Sysdig Tools
  - Sysdig tools have native support to many applications, infrastructure and container technologies, including Docker, Kubernetes, Mesos, AWS, and Google Cloud Platform.
  - Paid offerings provide alerting, dashboard, team management, etc.
  - They offer a programmatic interface with every part of Sysdig Monitor.

6장. 도구(Tools) 5. Debug Log Monitor

- Benefits of Sysdig Tools
  - The tools capture low-level system information from the running Linux instance and containers.
  - They offer native support for all Linux container technologies like Docker, LXC, etc.
  - They are easy to install.
  - They are built to run in production, minimizing performance overhead and the risk of crashes.
  - They are Kubernetes-aware.

6장. 도구(Tools) 5. Debug Log Monitor

- **cAdvisor**
  - **cAdvisor (Container Advisor) is an open source tool to collect resource usage and performance characteristics for the host system and running containers. It collects, aggregates, processes, and exports information about running containers. As of now, it has native support for Docker and should also support other container runtimes out of the box.**

355

6장. 도구(Tools) 5. Debug Log Monitor

- **Using cAdvisor**
  - **We can enable the cAdvisor tool to start collecting statistics with the following command:**

```
sudo docker run ₩
  --volume=/:/rootfs:ro ₩
  --volume=/var/run:/var/run:rw ₩
  --volume=/sys:/sys:ro ₩
  --volume=/var/lib/docker/:/var/lib/docker:ro ₩
  --publish=8080:8080 ₩
  --detach=true ₩
  --name=cadvisor ₩
  google/cadvisor:latest
```

  - **and point the browser to http://host_IP:8080 to get the live statistics. cAdvisor exposes its raw and processed statistics via a versioned remote REST API. It also supports exporting statistics for InfluxDB. cAdvisor exposes container statistics as Prometheus metrics. Prometheus is an open source community-driven system and service monitoring toolkit.**

356

6장. 도구(Tools) 5. Debug Log Monitor

- Heapster
  - Heapster enables container cluster monitoring and performance analysis. It currently supports Kubernetes natively. Heapster collects and interprets various signals, like compute resource usage, lifecycle events, etc., and exports cluster metrics via REST endpoints. Kubedash, a performance analytics UI for Kubernetes, uses those endpoints.

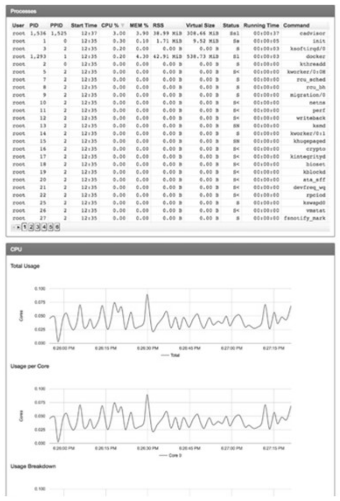
james@jslab.kr

JS Lab

357

6장. 도구(Tools) 5. Debug Log Monitor

- Host System Resource Usage with cAdvisor



User	PID	PPID	Start Time	CPU %	MEM %	RSS	Virtual Size	Status	Running Time	Command
root	1,536	1,525	12:37	3.00	3.90	38.99 MiB	308.66 MiB	Ssl	00:00:37	cadvisor
root	1	0	12:35	0.30	0.10	1.71 MiB	9.52 MiB	Ss	00:00:05	init
root	3	2	12:35	0.20	0.00	0.00 B	0.00 B	S	00:00:03	ksoftirqd/0
root	1,293	1	12:35	0.20	4.30	42.91 MiB	538.73 MiB	Sl	00:00:03	docker
root	2	0	12:35	0.00	0.00	0.00 B	0.00 B	S	00:00:00	kthreadd
root	5	2	12:35	0.00	0.00	0.00 B	0.00 B	S<	00:00:00	kworker/0:0H
root	7	2	12:35	0.00	0.00	0.00 B	0.00 B	S	00:00:00	rcu_sched
root	8	2	12:35	0.00	0.00	0.00 B	0.00 B	S	00:00:00	rcu_bh
root	9	2	12:35	0.00	0.00	0.00 B	0.00 B	S	00:00:00	migration/0
root	10	2	12:35	0.00	0.00	0.00 B	0.00 B	S<	00:00:00	netns
root	11	2	12:35	0.00	0.00	0.00 B	0.00 B	S<	00:00:00	perf
root	12	2	12:35	0.00	0.00	0.00 B	0.00 B	S<	00:00:00	writeback
root	13	2	12:35	0.00	0.00	0.00 B	0.00 B	SN	00:00:00	ksmd
root	14	2	12:35	0.00	0.00	0.00 B	0.00 B	S	00:00:00	kworker/0:1
root	15	2	12:35	0.00	0.00	0.00 B	0.00 B	SN	00:00:00	khugepaged
root	16	2	12:35	0.00	0.00	0.00 B	0.00 B	S<	00:00:00	crypto
root	17	2	12:35	0.00	0.00	0.00 B	0.00 B	S<	00:00:00	kintegrityd
root	18	2	12:35	0.00	0.00	0.00 B	0.00 B	S<	00:00:00	bioset
root	19	2	12:35	0.00	0.00	0.00 B	0.00 B	S<	00:00:00	kblockd
root	20	2	12:35	0.00	0.00	0.00 B	0.00 B	S<	00:00:00	ata_sff
root	21	2	12:35	0.00	0.00	0.00 B	0.00 B	S<	00:00:00	devfreq_wq
root	22	2	12:35	0.00	0.00	0.00 B	0.00 B	S<	00:00:00	rpciod
root	25	2	12:35	0.00	0.00	0.00 B	0.00 B	S	00:00:00	kswapd0
root	26	2	12:35	0.00	0.00	0.00 B	0.00 B	S<	00:00:00	vmstat
root	27	2	12:35	0.00	0.00	0.00 B	0.00 B	S	00:00:00	fsnotify_mark

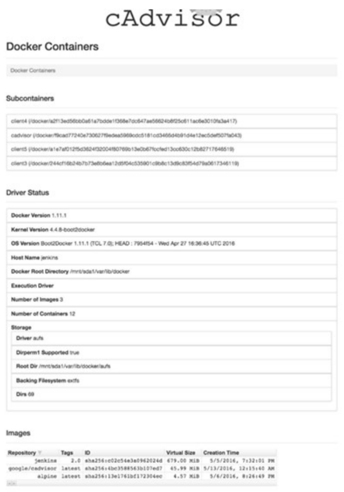
james@jslab.kr

JS Lab

358

6장. 도구(Tools) 5. Debug Log Monitor

- **Docker Host Specific Details with cAdvisor**



cAdvisor

Docker Containers

Docker Containers

Subcontainers

client4 (/docker/a2f13ed56bb0a61a7bdde1f368e7dc647ae56624b8f25c611ac6e3010fa3a417)
cadvisor (/docker/f9cad77240e730627f9edea5969cdc5181cd3466d4b91d4e12ec5def507fa043)
client5 (/docker/a1e7af012f5d3624f32004f80769b13e0b67fccfed13cc630c12b82717646519)
client3 (/docker/244cf16b24b7b73e8b6ea12d5f04c535901c9b8c13d9c83f54d79a0617346119)

Driver Status

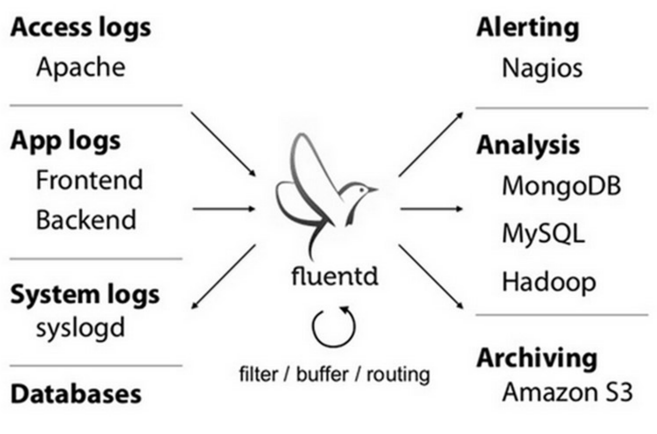
Docker Version 1.11.1
Kernel Version 4.4.8-boot2docker
OS Version Boot2Docker 1.11.1 (TCL 7.0); HEAD : 7954f54 - Wed Apr 27 16:36:45 UTC 2016
Host Name jenkins
Docker Root Directory /mnt/sda1/var/lib/docker
Execution Driver
Number of Images 3
Number of Containers 12
Storage
Driver aufs
Dirperm1 Supported true
Root Dir /mnt/sda1/var/lib/docker/aufs
Backing Filesystem extfs
Dirs 69

Images

Repository	Tags	ID	Virtual Size	Creation Time
jenkins	2.0	sha256:c02c54e3a0962024d	679.00 MiB	5/5/2016, 7:32:01 PM
google/cadvisor	latest	sha256:4bc3588563b107ed7	45.99 MiB	5/13/2016, 12:15:40 AM
alpine	latest	sha256:13e1761bf172304ec	4.57 MiB	5/6/2016, 8:26:49 PM

6장. 도구(Tools) 5. Debug Log Monitor

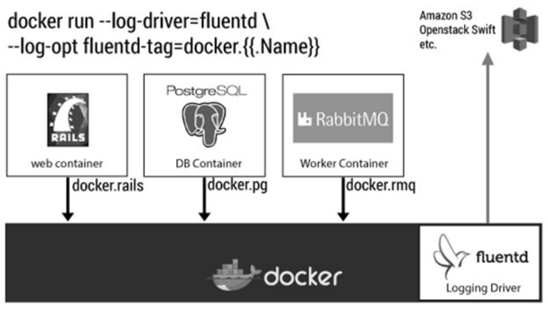
- **Fluentd**



luentd Architecture (by Treasure Data, Inc., retrieved from fluentd.org)

6장. 도구(Tools) 5. Debug Log Monitor

- Docker Support for Fluentd
- From version 1.8, Docker supports logging drivers and Fluentd is one of them.



```
docker run --log-driver=fluentd \
--log-opt fluentd-tag=docker.{{.Name}}
```

Fluentd and Docker Integrated (by Treasure Data, Inc., retrieved from fluentd.org)

6장. 도구(Tools) 5. Debug Log Monitor

- Benefits of Using Fluentd
  - It is an open source data collector.
  - It is simple, fast, and flexible.
  - It is performant and developer-friendly.

6장. 도구(Tools) 5. Debug Log Monitor

- Datadog
- Datadog provides monitoring and analytics as a service for Development and OPs teams. Some of the systems, applications and services it connects to are:
  - Amazon EC2
  - Apache
  - Java
  - MySQL
  - CentOS.

363

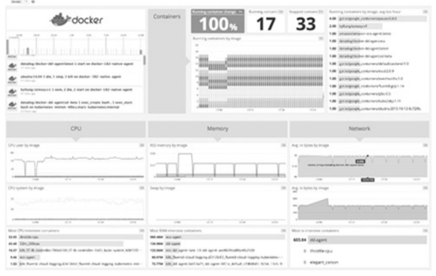
6장. 도구(Tools) 5. Debug Log Monitor

- A detailed list of integration can be found in the documentation it provides. We need to install an agent in the host system, which sends the data to the Datadog's server. Once the data is sent, we can:
  - Build an interactive dashboard.
  - Search and co-relate matrices and events.
  - Share the matrices and events.
  - Get alerts.

364

6장. 도구(Tools) 5. Debug Log Monitor

- Docker Containers: Kubernetes Monitoring with Datadog
  - The number of nodes in the cluster
  - The running and stopped containers
  - The most resource-consuming pods
  - Docker logs, etc.



Docker Containers - Kubernetes Monitoring with Datadog (by Datadog, Inc., retrieved from Datadog documentation)

6장. 도구(Tools) 5. Debug Log Monitor

- Benefits of Using Datadog
  - It comes pre-integrated with well-known third-party applications.
  - It provides a seamless workflow, regardless of platform, location or language.
  - It configures information filtration to get only needed metrics.
  - It allows us to enable the system to send alerts or notifications when serious issues arise.
  - It offers tools for team collaboration.
  - It is scalable.

https://www.datadoghq.com/

6장. 도구(Tools) 5. Debug Log Monitor

- Prometheus
- Prometheus is an open source tool used for system monitoring and alerting. It was originally developed by SoundCloud and is now one of the incubated projects at CNCF Foundation.
- Prometheus is suitable for recording any purely numeric time series data. It works well for both machine-centric monitoring like CPU, memory usage, and monitoring of highly dynamic service-oriented architectures. It is primarily written in Go.

프로메테우스

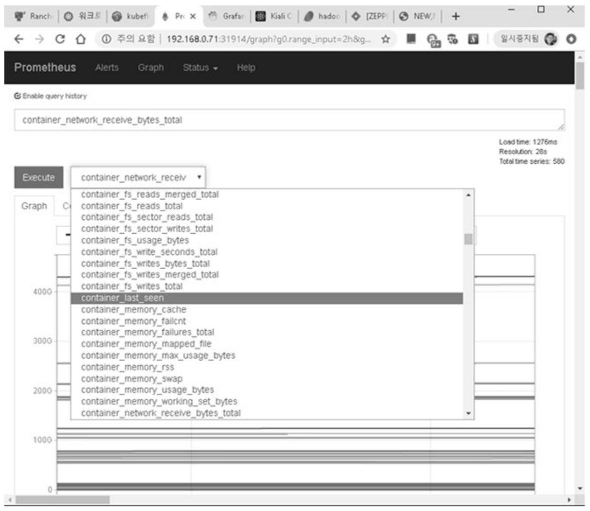
다차원 시계열 데이터베이스로서
경고 문자와 그래프 계산을 지원

JS Lab

367

6장. 도구(Tools) 5. Debug Log Monitor

- Prometheus



192.168.0.71:31914/graph?g0.range_input=2h&g...

container_network_receive_bytes_total

Load time: 1276ms
Resolution: 28s
Total time series: 580

container_network_receiv

container_fs_reads_merged_total
container_fs_reads_total
container_fs_sector_reads_total
container_fs_sector_writes_total
container_fs_usage_bytes
container_fs_write_seconds_total
container_fs_writes_bytes_total
container_fs_writes_merged_total
container_fs_writes_total
container_last_seen
container_memory_cache
container_memory_failcnt
container_memory_failures_total
container_memory_mapped_file
container_memory_max_usage_bytes
container_memory_rss
container_memory_swap
container_memory_usage_bytes
container_memory_working_set_bytes
container_network_receive_bytes_total

JS Lab

368

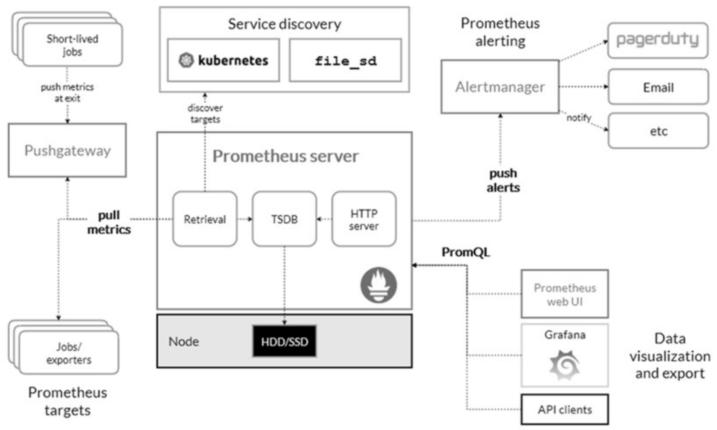
6장. 도구(Tools) 5. Debug Log Monitor

▫ Prometheus Features

- It is very reliable.
- It supports multi-dimensional data model with time series data identified by metric name and key/value pairs.
- It supports a query language to effectively query the collected time series data.
- It support metrics collection through pull- and push-based mechanism.
- It can discover target endpoints via service discovery or static configuration.
- It can connect with external tools like Grafana and Pagerduty for dashboarding and alerting.
- It supports client libraries for programming language like Go, Java, Python, etc. to add instrumentation to their code.

6장. 도구(Tools) 5. Debug Log Monitor

▫ Prometheus Architecture



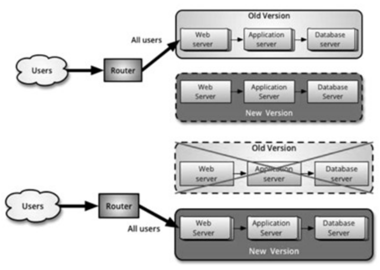
Prometheus Architecture (retrieved from prometheus.io)

6장. 도구(Tools) — 6. Immutable Infra.

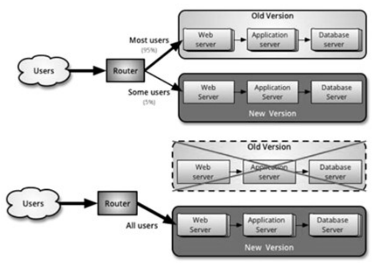
□ **Deployments**

- **Blue/Green Deployments**
- **Canary Deployments**

Blue/Green Deployments

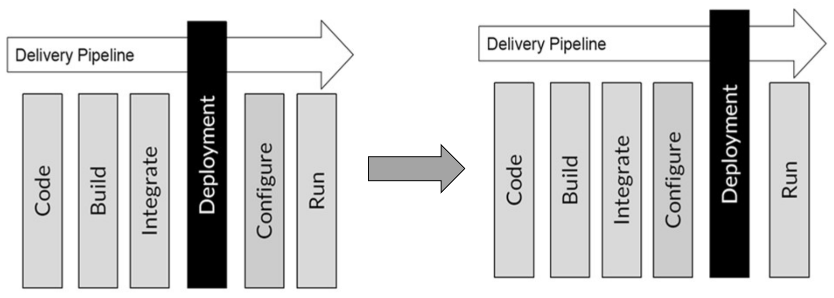


Canary Deployments



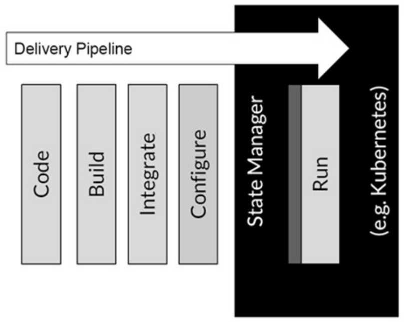
6장. 도구(Tools) — 6. Immutable Infra.

□ **Shifting Configuration BEFORE Deployment**



6장. 도구(Tools) — 6. Immutable Infra.

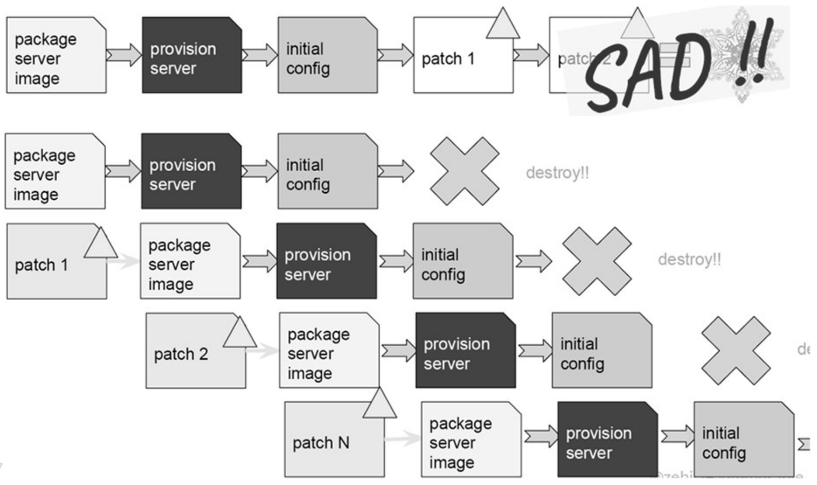
- Which Enables… Delegating Operations
  - If you can make your artifacts immutable then you can delegate management of them to a platform like Kubernetes.
  - Kubernetes does not configure infrastructure. It maintains state based on a manifest.



373

6장. 도구(Tools) — 6. Immutable Infra.

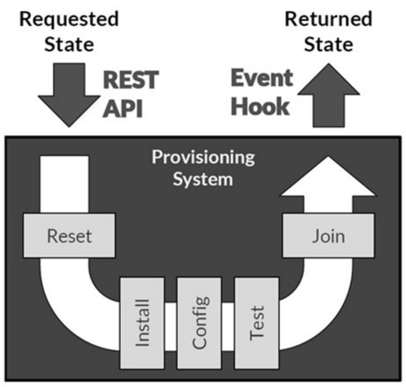
- Immutable is a DevOps Pattern



374

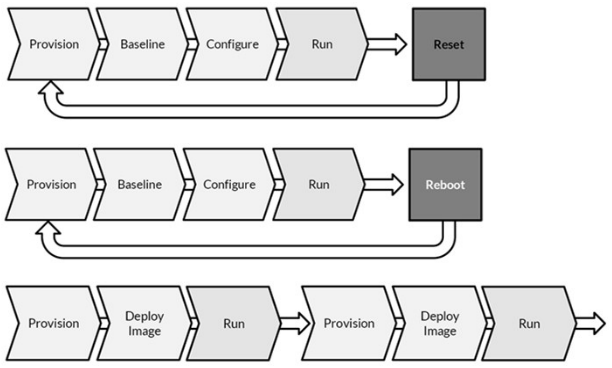
6장. 도구(Tools) 6. Immutable Infra.

- **Cloud-like Integration and Staged Workflow**
- **Immutable Provisioning systems treat infrastructure as a black box**



6장. 도구(Tools) 6. Immutable Infra.

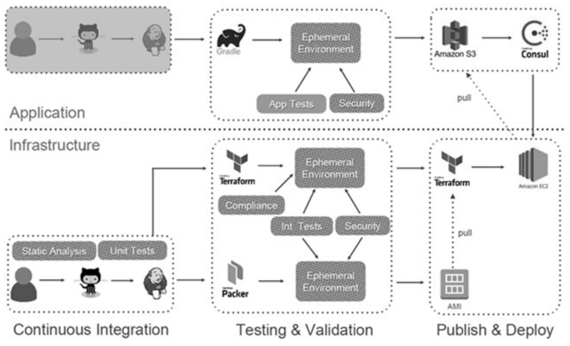
- **Immutable Patterns**
  - **Baseline + Configuration**
  - **Live Boot + Configuration**
  - **Image Deploy**



6장. 도구(Tools) 6. Immutable Infra.

□ **Containerized Services**

- **Infra Module - Container Management System**
- **Fully Decoupled from Apps**
- **Apps are deployed with Container Management System specific tools**



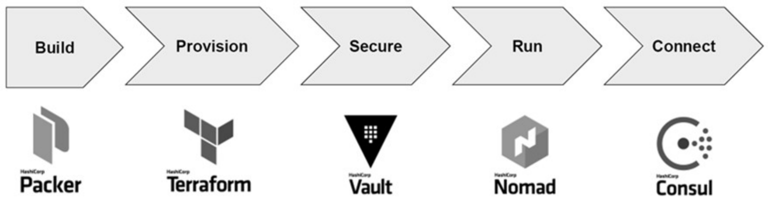
377

6장. 도구(Tools) 6. Immutable Infra.

□ **How do you do this with immutability?**

- **머신 프로비져닝 (How do we provision a machine?)**
- **요소간 연결 (How do we connect various parts?)**
- **릴리즈 (How do we release?)**
- **비용 제어 (How can we keep costs in control?)**
- **서비스 방향 (Where have all the services gone?)**
- **인프라 접속 (Who has access?)**

□ **Tooling @ Cloud (예)**



378

6장. 도구(Tools) 6. Immutable Infra.

- K8s
  - Orchestration
  - Deployment
  - Scaling
  - Data Plane
- Istio
  - Policy Enforcement
  - Traffic Management
  - Telemetry
  - Control Plane

379

1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ **실습교재 (별도)**

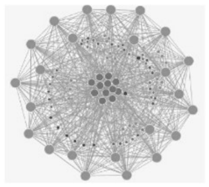
380

7장. 서비스 메시 (Service Mesh)

- Control Plane
- Data Plane
- Istio

7장. 서비스 메시 (Service Mesh)

- Service mesh
- Service mesh is a network communication infrastructure layer for a microservices-based application. When multiple microservices are communicating to each other, a service mesh allows us to decouple resilient communication patterns such as circuit breakers and timeouts from the application code.



- 서비스 장애가 다른 서비스에 영향을 주는 문제 해결
- 특정 서비스가 느려지거나 응답이 없으면 연결된 서비스들의 장애 전파 해결

7장. 서비스 메시 (Service Mesh)

- **Data Plane and Control Plane**
- **Similar to Software-Defined Networking, which we explored earlier, service mesh also has Data and Control Planes.**
  - **Service Mesh Data Plane:** It provides features that we mentioned on the previous page. It touches every packet/request in the system.
  - **Service Mesh Control Plane:** It provides policy and configuration for the Data Plane. For example, by using the control plane, we can specify settings for load balancing, circuit breaker, etc.

7장. 서비스 메시 (Service Mesh)

- **Features and Implementation of Service Mesh**
  - **Communication: It provides flexible, reliable, and fast communication between various service instances.**
  - **Circuit Breakers:** It restricts traffic to the unhealthy service instances.
  - **Routing**: It gives a REST request for /foo from the local service instance, to which the service is connected.
  - **Retries and Timeouts:** It can automatically retry requests on certain failures and can timeout requests after a specified period.
  - **Service Discovery:** It discovers healthy, available instances of services.
  - **Authentication and Authorization:** It can authenticate and authorize incoming requests.
  - **Transport Layer Security (TLS) Encryption:** It can secure service-to-service communication using TLS

7장. 서비스 메시 (Service Mesh)

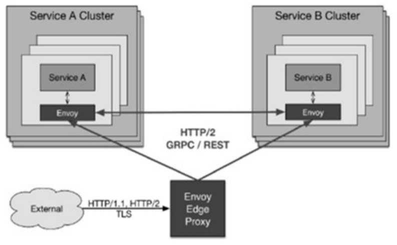
Service Mesh Project Landscape

- **Data Plane**
  1. Linkerd
  2. NGINX
  3. HAProxy
  4. Envoy
  5. Traefik.
- **Control Plane**
  1. Istio
  2. Nelson
  3. SmartStack.

7장. 서비스 메시 (Service Mesh)

Envoy (1 of 2)

- **Envoy is a Cloud Native Computing Foundation(CNCF) project, which was originally built by Lyft. It is an open source project that provides an L7 proxy and communication bus for large, modern, service-oriented architectures.**
- **Envoy has an out-of-process architecture, which means it is not dependent on the application code. It runs alongside the application and communicates with the application on a localhost. We referred to this earlier as a sidecar pattern.**



envoy →

7장. 서비스 메시 (Service Mesh)

Envoy (2 of 2)

- With the Envoy sidecar implementation, applications need not be aware of the network topology. Envoy can work with any language and can be managed independently.
- Envoy can be configured as service and edge proxy. In the service type of configuration, it is used as a communication bus for all traffic between microservices. With the edge type of configuration, it provides a single point of ingress to the external world.

  - L4/L7 Proxy로 동작
  - 해당 서비스의 pod내에 sidecar 형태로 배포
  - 동적 서비스 디스커버리, 로드밸런싱, TLS client certification
  - HTTP1/2, TCP, gRPC 프로토콜 지원
  - Circuit breaker, Health Check, Timeout, Auto retry, Fault injection, 부하 제한 등의 기능
  - HTTP L7 라우팅 지원을 통한 URL 기반 라우팅, 버퍼링, 서버간 부하분산량 조절
  - 다양한 통계추적과 서비스에 대한 다양한 가시성(visibility) 제공

7장. 서비스 메시 (Service Mesh)

Features and Benefits of Envoy

- It is an open source project.
- It makes the network transparent to the applications.
- Due to its out-of-process architecture, it can be run alongside any language or runtime.
- It has support for HTTP/2 and gRPC for both incoming and outgoing connections.
- It provides all the features of service mesh that we mentioned earlier, like load balancing, service discovery, circuit breakers, etc.
- It provides good monitoring using statistics, logging, and distributed tracing.
- It can provide SSL communication.

7장. 서비스 메시 (Service Mesh)

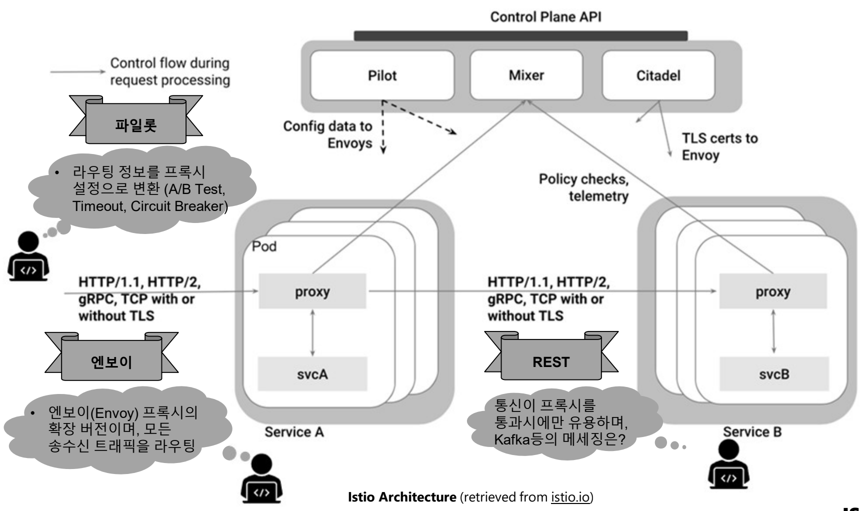
- Istio
- Istio is one of the most popular service mesh solutions. It is an open source platform, backed by companies like Google, IBM and Lyft.
- Istio is divided into the following two planes:
  - **Data Plane:** It is composed of a set of Envoy proxies, which are deployed as sidecars to provide a medium for communication and control all network communication between microservices.
  - **Control Plane**: It manages and configures proxies to route traffic. It enforces policies at runtime and collects telemetry.

이스티오

- 모니터링, 로그분석 추적 기술의 통합
- 마이크로서비스간 통신 암호화(TLS)
- 고급 라우팅(예: A/B 테스트)
- 복원력(서킷 브레이크등)

7장. 서비스 메시 (Service Mesh)

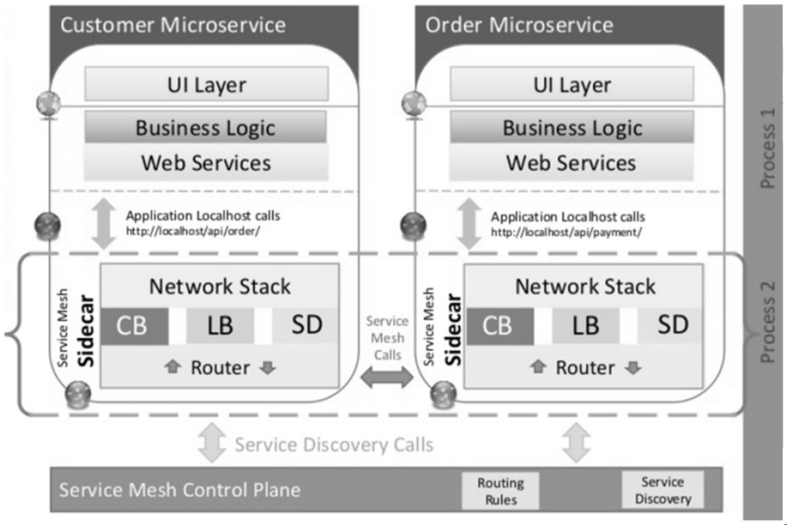
- Istio Architecture


Control Plane API
Pilot
Mixer
Citadel
Control flow during request processing
파일롯
• 라우팅 정보를 프록시 설정으로 변환 (A/B Test, Timeout, Circuit Breaker)
Config data to Envoys
TLS certs to Envoy
Policy checks, telemetry
Pod
HTTP/1.1, HTTP/2, gRPC, TCP with or without TLS
proxy
svcA
HTTP/1.1, HTTP/2, gRPC, TCP with or without TLS
proxy
svcB
엔보이
• 엔보이(Envoy) 프록시의 확장 버전이며, 모든 송수신 트래픽을 라우팅
REST
통신이 프록시를 통과시에만 유용하며, Kafka등의 메세징은?
Service A
Service B

Istio Architecture (retrieved from istio.io)

7장. 서비스 메시 (Service Mesh)

□ Service Mesh 관리 (예: Sidecar Design Pattern 기반 관리)



Circuit Breaker(CB), Load Balancer(LB), Service Discovery(SD)

7장. 서비스 메시 (Service Mesh)

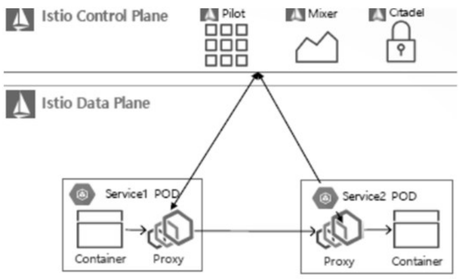
□ Istio Components

- **Envoy:** Istio uses an extended version of the Envoy proxy, using which it implements features like dynamic service discovery, load balancing, TLS termination, circuit breakers, health checks, etc. Envoy is deployed as sidecars.
- **Mixer:** Mixer enforces access control and policies for Istio. It also collects telemetry data from the Envoy proxy and other services.
- **Pilot:** Pilot provides service discovery for the Envoy sidecars, traffic management capabilities for intelligent routing and resiliency. It creates the Envoy-specific configurations based on the high level rules and propagates them to the sidecars at runtime.
- **Citadel:** Citadel provides end user and service-to-service authentication. With the 0.5 release, Istio also supports Role-Based Access Control (RBAC).

7장. 서비스 메시 (Service Mesh)

Istio Components

- **Envoy Proxy: 제어 및 관리, 서비스 디스커버리, 로드 밸런싱, 서비스 호출/ 응답 관련 로깅, 모니터링, 트래픽 분할, 접근제어, 통신 암호화**
- **Pilot : envoy에 대한 설정 관리를 수행, 서비스 및 트래픽 관리, Policy 적용**
- **Mixer: 액세스 컨트롤, 정책 통제 그리고 각종 모니터링 지표의 수집**
- **Citadel: 보안에 관련된 기능을 담당하는 모듈로 서비스를 사용하기 위한 사용자 인증 (Authentication)과 인가(Authorization)을 담당**



7장. 서비스 메시 (Service Mesh)

Features and Benefits of Istio

- **Policy Enforcement:** Istio can configure policies for inter-service communication. They can be applied at runtime, without reconfiguring services.
- **Telemetry:** Istio provides rich telemetry data using which we can gain understanding of service dependencies and traffic flow to quickly identify issues.

- 서로 다른 버전의 서비스를 배포해 놓고 버전별로 트래픽의 양을 조절할 수 있는 기능
- 네트워크 패킷의 내용을 기반으로 라우팅이 가능 (L7 기능)
- 헬스체크 및 서비스 디스커버리 기능을 통해 장애 감지 및 조치 하는 기능
- 서비스 간의 호출 안정성을 위해서 Retry, Timeout, Circuit breaker
- 기본적으로 envoy 를 통해서 통신하는 모든 트래픽을 자동으로 TLS 를 이용해서 암호화
- 네트워크 트래픽 ,호출 관계 , 서비스의 응답 시간 등의 다양한 지표를 수집하여 모니터링

7장. 서비스 메시 (Service Mesh)

- Match condition
- Destination

```
- match:
  - headers:
      end-user:
        exact: jason
```

```
route:
- destination:
    host: reviews
    subset: v2
```

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: bookinfo
spec:
  hosts:
    - bookinfo.com
  http:
  - match:
    - uri:
        prefix: /reviews
    route:
    - destination:
        host: reviews
  - match:
    - uri:
        prefix: /ratings
    route:
    - destination:
        host: ratings
...

  http:
  - match:
      sourceLabels:
        app: reviews
    route:
...
```

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: my-destination-rule
spec:
  host: my-svc
  trafficPolicy:
    loadBalancer:
      simple: RANDOM
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
    trafficPolicy:
      loadBalancer:
        simple: ROUND_ROBIN
  - name: v3
    labels:
      version: v3
```

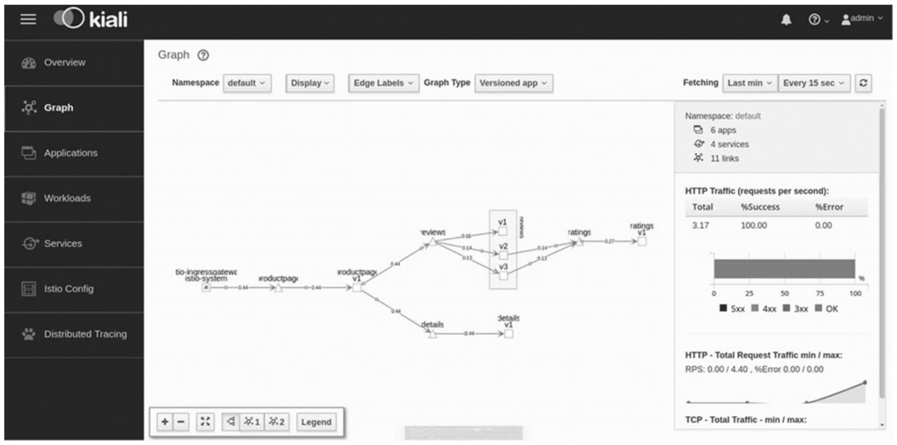
```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: ext-host-gwy
spec:
  selector:
    app: my-gateway-controller
  servers:
  - port:
      number: 443
      name: https
      protocol: HTTPS
    hosts:
    - ext-host.example.com
    tls:
      mode: SIMPLE
      serverCertificate: /tmp/tls.crt
      privateKey: /tmp/tls.key
```

https://istio.io/docs/concepts/traffic-management/

JS Lab

7장. 서비스 메시 (Service Mesh)

- **Service Mesh 관리** (예: Sidecar Design Pattern 기반 관리)

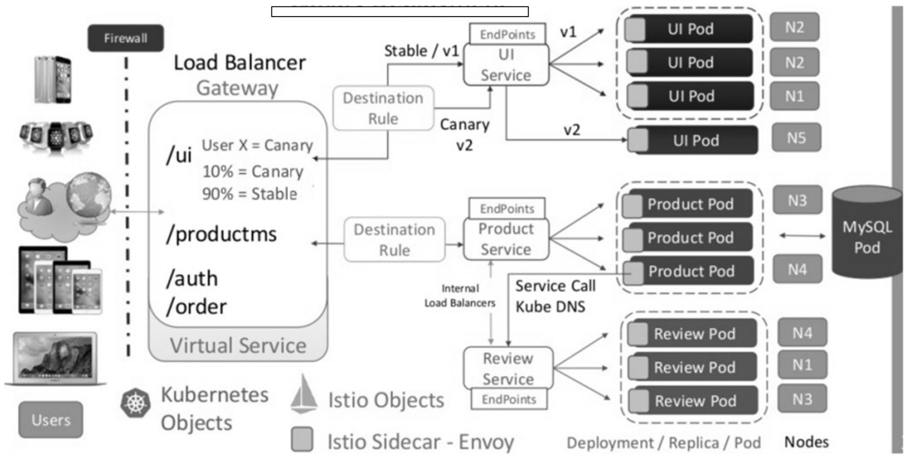


https://github.com/ruzickap/k8s-istio-demo

JS Lab

7장. 서비스 메시 (Service Mesh)

◘ **서비스 업그레이드 운영** (예: A/B Testing, Canary Deployment)



Circuit Breaker(CB), Load Balancer(LB), Service Discovery(SD)

JS Lab

james@jslab.kr

397

7장. 서비스 메시 (Service Mesh)

◘ **이스티오의 복원력**

- **전체 마이크로서비스 시스템 다운 시킬 수 있는 연쇄 실패를 피하는 것이 중요**
- **복원력 측정과 구현**

```
$ istioctl get routerule ratings-test-delay -o yaml
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: ratings-test-delay
  namespace: default
  ...
spec:
  destination:
    name: ratings
  httpFault:
    delay:
      fixedDelay: 7.000s
      percent: 100
  match:
    request:
      headers:
        cookie:
          regex: ^(.*?;)?(user=jason)(;.*)?$
  precedence: 2
  route:
  - labels:
      version: v1
```

```
$ istioctl get routerules ratings-test-abort -o yaml
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: ratings-test-abort
  namespace: default
  ...
spec:
  destination:
    name: ratings
  httpFault:
    abort:
      httpStatus: 500
      percent: 100
  match:
    request:
      headers:
        cookie:
          regex: ^(.*?;)?(user=jason)(;.*)?$
  precedence: 2
  route:
  - labels:
      version: v1
```

JS Lab

james@jslab.kr

398

7장. 서비스 메시 (Service Mesh)

Linkerd

- Linkerd is an open source network proxy and one of the Cloud Native Computing Foundation (CNCF) projects.
- It supports all features of the service mesh listed earlier. Linkerd can also be installed per host/instance, in addition to the sidecar.

399

7장. 서비스 메시 (Service Mesh)

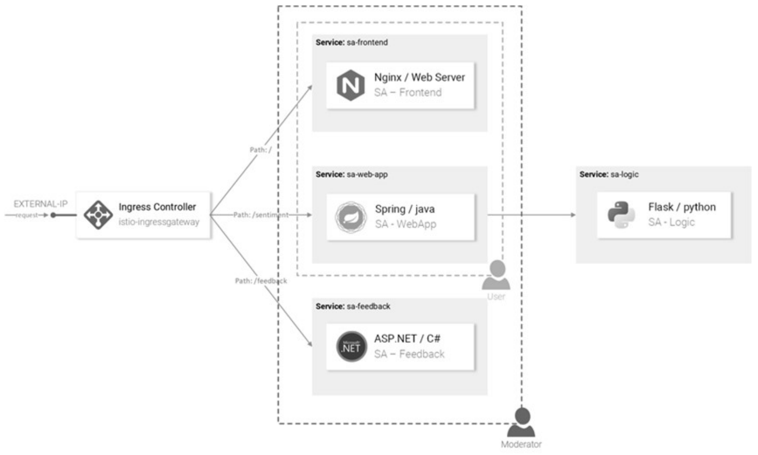
Features and Benefits of Using Linkerd

- It is open source.
- It can be run on different platforms like VM, Docker, Kubernetes, DC/OS, Amazon ECS.
- It's fast, scalable and performant.
- It runs as a transparent proxy alongside existing applications and integrates with the existing infrastructure.
- Integrates with most service discovery systems.

400

7장. 서비스 메시 (Service Mesh)

□ Authorization Concept



https://medium.com/google-cloud/back-to-microservices-with-istio-part-2-authentication-authorization-b079f77358ac

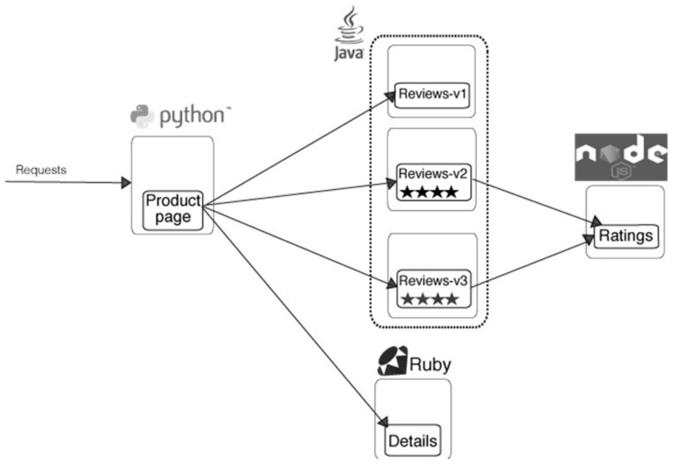
JS Lab

james@jslab.kr

401

7장. 서비스 메시 (Service Mesh)

□ Authorization Concept

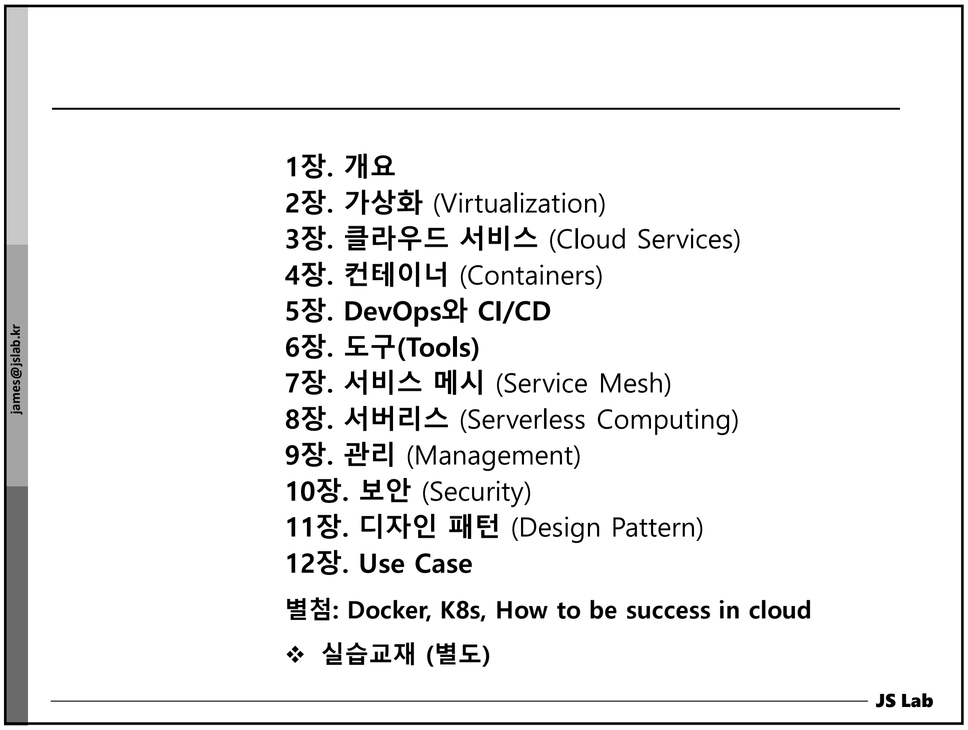


https://thenewstack.io/tutorial-explore-istios-traffic-rules-and-telemetry-capabilities/

JS Lab

james@jslab.kr

402

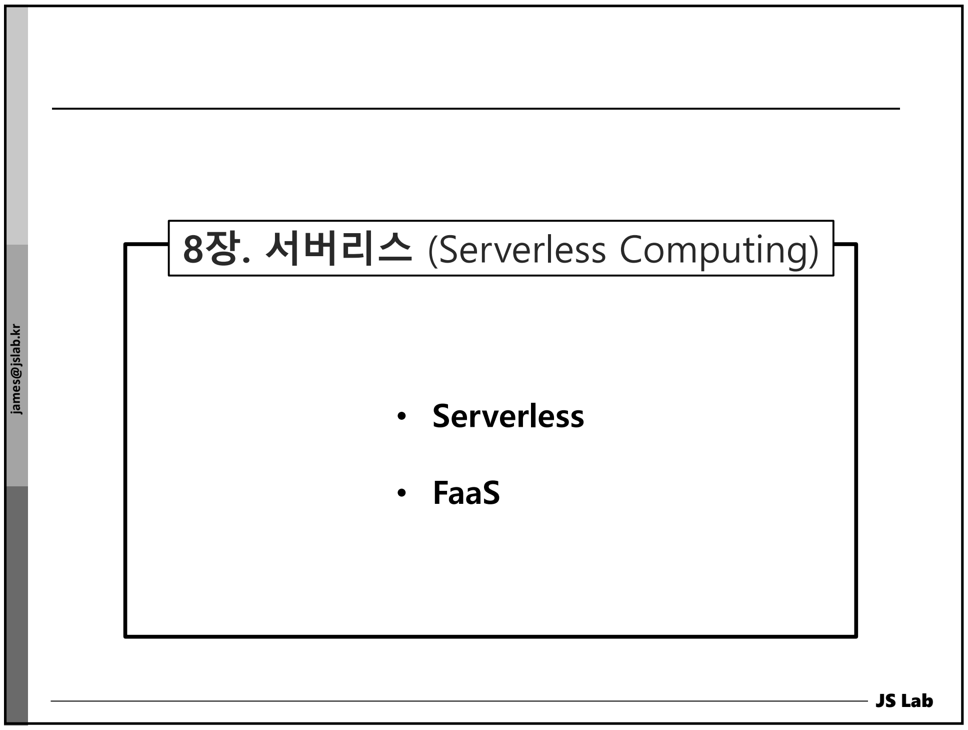


1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud

❖ **실습교재 (별도)**

JS Lab

403



8장. 서버리스 (Serverless Computing)

- **Serverless**
- **FaaS**

JS Lab

404

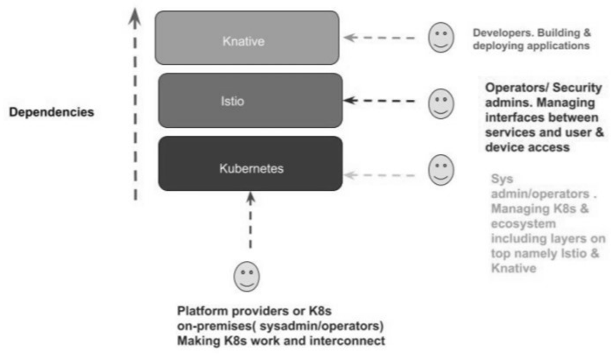
8장. 서버리스 (Serverless Computing)

- Serverless computing or serverless is a way to run applications without concerns about provisioning of computer servers or resources. However, it definitely doesn't mean that there are no servers involved. It is similar to the wireless Internet - the wires exist, but they are not visible for end users.
- Serverless computing requires compute resources to run applications, but the server management and the capacity planning decisions are completely abstracted from developers and users.

405

8장. 서버리스 (Serverless Computing)

- CI/CD Tools - Serverless
  - Infrastructure As A Code
  - Application Configuration 과Infra
  - Configuration을 하나의 설정에 통합 - 단일도구


Knative
Istio
Kubernetes
Dependencies
Developers. Building & deploying applications
Operators/ Security admins. Managing interfaces between services and user & device access
Sys admin/operators . Managing K8s & ecosystem including layers on top namely Istio & Knative
Platform providers or K8s on-premises(sysadmin/operators) Making K8s work and interconnect

406

8장. 서버리스 (Serverless Computing)

Serverless Computing (1 of 2)

- In serverless computing, we generally write applications/functions that focus and master one thing in particular. We then upload that application on the cloud provider, which gets invoked via different events, such as HTTP requests, webhooks, etc.
- The most common use case of serverless computing is to run any stateless applications like data processing or real time stream processing, etc.; though, it can augment a stateful application. Internet of Things and ChatBots are very good use cases for serverless computing.

8장. 서버리스 (Serverless Computing)

Serverless Computing (1 of 2)

- All major cloud providers like AWS, Google Cloud Platform or Microsoft Azure have serverless offerings. We will explore them in this chapter. We can also build our own serverless solutions on container orchestrators like Docker Swarm, Kubernetes, etc.
- Most often, serverless computing is referred via services offered by cloud providers. We will also follow the same practice. In addition, we will explicitly talk about custom or self-managed serverless computing whenever needed.

8장. 서버리스 (Serverless Computing)

Features and Benefits of Serverless Computing

- **No Server Management:** When we use serverless offerings by cloud providers, no server management and capacity planning has to be done on our side. It is all taken care of by the cloud providers.
- **Cost-Effective:** We only need to pay for a CPU time when our applications/functions are executed. There is no charge when code is not running. Also, there is no need to rent or purchase fixed quantities of servers.
- **Flexible Scaling:** We don't need to set up or tune autoscaling. Applications are automatically scaled up/down based on demand.
- **Automated High Availability and Fault Tolerance:** High availability and fault tolerance automatically come from underling providers. Developers don't need to specifically program for that.

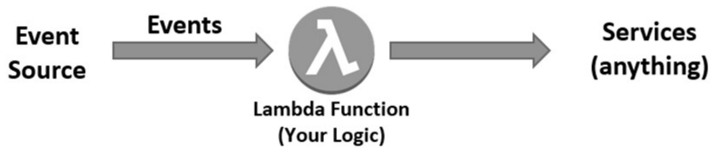
8장. 서버리스 (Serverless Computing)

Drawbacks of Serverless Computing

- **Vendor Lock-In:** Serverless features and implementation vary vendor to vendor. So, the same application/function may not behave in the same way if you change the provider, and changing your provider can incur additional expenses.
- **Multitenancy and Security:** You cannot be sure what other applications/functions run beside you, which brings multitenancy and security concerns.
- **Performance:** If the application is not in use, the service provider can take it down, which will affect the performance.
- **Resource Limits:** Cloud providers put resource limits for our serverless applications/functions. Therefore, it is safer not to run high-performance, resource-intensive workloads using serverless solutions.
- **Monitoring and Debugging:** It is more difficult to monitor serverless applications than those running on traditional servers.

8장. 서버리스 (Serverless Computing)

- Introduction to AWS Lambda
  - **AWS Lambda is the serverless service offered by Amazon Web Services (AWS). AWS Lambda can be triggered in different ways, like an HTTP request, a new document upload to S3, a scheduled job, an AWS Kinesis data stream, or a notification from AWS Simple Notification Service, etc.**



AWS Lambda (retrieved from aws.amazon.com)

JS Lab

411

8장. 서버리스 (Serverless Computing)

- Features and Benefits of AWS Lambda
  - **Integrating with other AWS services:** It integrates well with other AWS services, such as CloudWatch.
  - **Building custom backend services:** It can create new backend services for applications that can be triggered using the Lambda API.
  - **Bringing users own code:** It supports different programming languages like Node.js, Java, C#, Go, and Python. This allows users to run and deploy their custom applications with AWS Lambda.

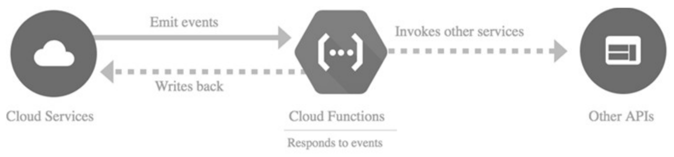
JS Lab

412

8장. 서버리스 (Serverless Computing)

◻ Google Cloud Functions

- Google Cloud Functions is Google's serverless service, which offers all the serverless benefits/features mentioned earlier. An application that runs with Cloud Functions can connect to cloud services.
- Google Cloud Functions is written in Javascript and executed in the Node.js v6.14.0 environment on the Google Cloud Platform. We can run Google Cloud Functions in any standard Node.js environment, which makes portability and local testing simple and easy.



Google Cloud Functions (retrieved from cloud.google.com)

8장. 서버리스 (Serverless Computing)

◻ Features and Benefits of Google Cloud Functions

- **Integrating with other Google services:** It connects well with other Google services like GCP, Firebase, Google Assistant, etc.
- **Supports JavaScript (Node.js) and Python:** It supports JavaScript (Node.js) and Python programming languages to write serverless functions.

8장. 서버리스 (Serverless Computing)

- Azure Functions
  - C#
  - JavaScript
  - F#
  - Python (experimental)
  - PHP (experimental)
  - TypeScript (experimental)
  - Batch (.cmd; .bat) (experimental)
  - Bash (experimental)
  - PowerShell (experimental).

415

8장. 서버리스 (Serverless Computing)

- **Features and Benefits of Azure Functions**
  - **Integration with other Azure services:** It integrates well with other Azure services, like Azure Event Hubs and Azure Storage.
  - **Bringing your own dependencies:** It supports NuGet and NPM, so you can use your favorite libraries.
  - **Integrated security:** It provides OAuth security for HTTP-triggered functions with OAuth providers such as Azure Active Directory, Facebook, Google, Twitter, and Microsoft Account.
  - **Open source:** Azure Functions has an open source runtime.

416

8장. 서버리스 (Serverless Computing)

- **Serverless Computing and Containers**
  - **Earlier in this chapter, we learned that with serverless computing we can run applications/functions which do one thing and do it well. Also, earlier in the course, we saw that with container images, we can package applications and run them as containers. We run containers using different container runtimes and orchestrate them using container orchestrators like Kubernetes, Docker Swarm, Amazon ECS, etc.**

8장. 서버리스 (Serverless Computing)

- **Projects That Use Containers to Execute Serverless Applications (1 of 2)**
  - **Azure Container Instances:** Azure Container Instances (ACI) is the service offered by Microsoft Azure, using which we run containers without managing servers. It provides hypervisor isolation for each container group to ensure containers run in isolation without sharing a kernel.
  - **AWS Fargate:** AWS Fargate is the service offered by Amazon Web Services, using which we can run containers without managing servers. It runs container top of Amazon ECS and EKS services.
  - **OpenFaaS:** OpenFaaS is an open source project, using which we can run containers on top of Docker Swam or Kubernetes.

8장. 서버리스 (Serverless Computing)

Projects That Use Containers to Execute Serverless Applications (1 of 2)

- **Kubeless:** Kubeless is an open source project from Bitnami, which provides a serverless framework on Kubernetes.
- **Fission:** Fission is an open source project from Platform9, which provides a serverless framework on Kubernetes.
- **virtual-kubelet:** virtual-kubelet connects Kubernetes to other APIs and masquerades them as Kubernetes nodes. These other APIs include ACI, Fargate, IoT Edge, etc.

1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ **실습교재 (별도)**

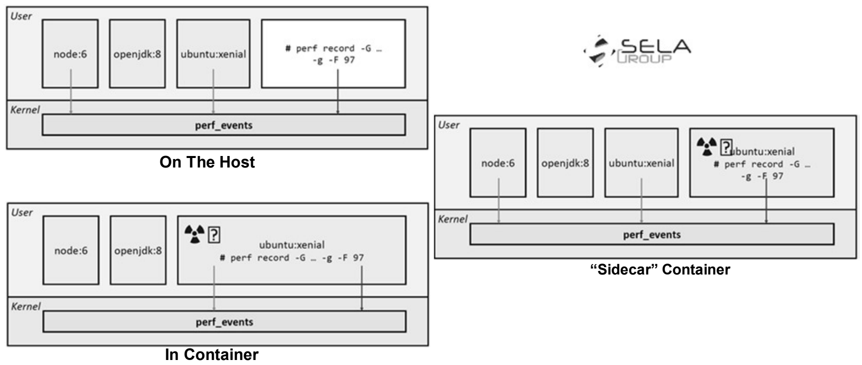
9장. 관리 (Management)

- **Trace**
- **Log**
- **Monitor**

421

9장. 관리 (Management)

- **Tool Deployment**
  - **호스트 내 설치 (On The Host)**
  - **컨테이너 내 설치 (In Container)**
  - **도구 전용 컨테이너 (“Sidecar” Container)**



422

9장. 관리 (Management)

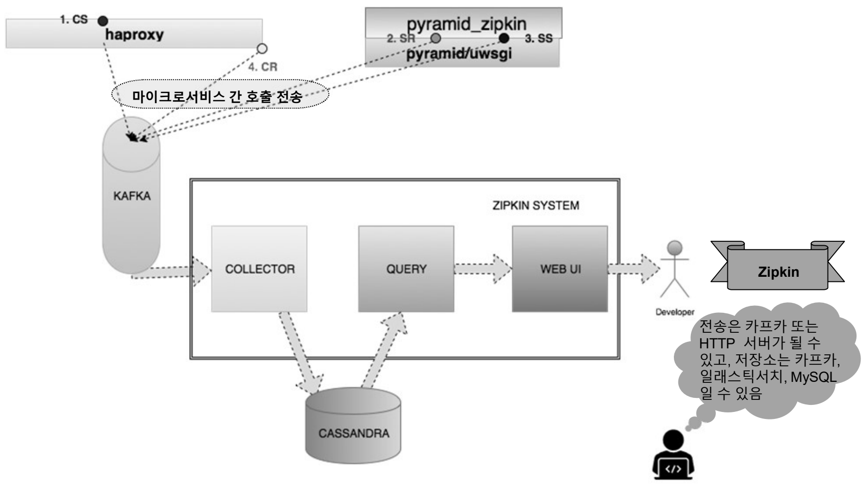
- Organizations are adopting microservices for agility, easy deployment, scaling, etc. However, with a large number of services working together, sometimes it becomes difficult to pinpoint the root cause when we face some kind of latency issues or unexpected behavior. To overcome such situations, we would need to instrument the behavior of each participating service in our microservices-based application. After collecting instrumented data from the participating service, we should be able to combine them to get visibility of the entire system. This is generally referred to as distributed tracing.

9장. 관리 (Management)

- There are various distributed tracing tools like Zipkin, Dapper, HTrace, and X-Trace, but they instrument applications using their own specific APIs, which are not compatible with each other. Due to this tight coupling, developers do not feel very comfortable with them. Enter OpenTracing, which offers consistent, expressive, vendor-neutral APIs with just O(1) configuration changes. OpenTracing is an open source project under CNCF.

9장. 관리 (Management)

- Tracing with Zipkin

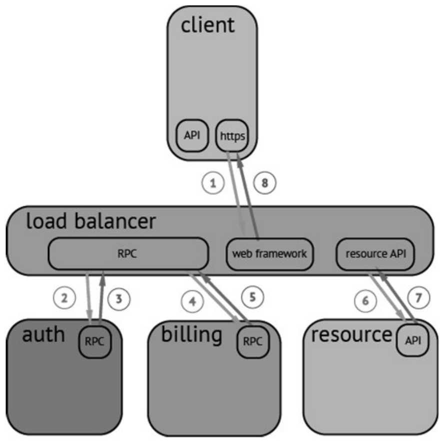

1. CS
haproxy
4. CR
pyramid_zipkin
2. SR
3. SS
pyramid/uwsgi
마이크로서비스 간 호출 전송
KAFKA
ZIPKIN SYSTEM
COLLECTOR
QUERY
WEB UI
Developer
Zipkin
CASSANDRA
전송은 카프카 또는 HTTP 서버가 될 수 있고, 저장소는 카프카, 일래스틱서치, MySQL 일 수 있음

https://engineeringblog.yelp.com/2016/04/distributed-tracing-at-yelp.html

425

9장. 관리 (Management)

- Tracing with OpenTracing

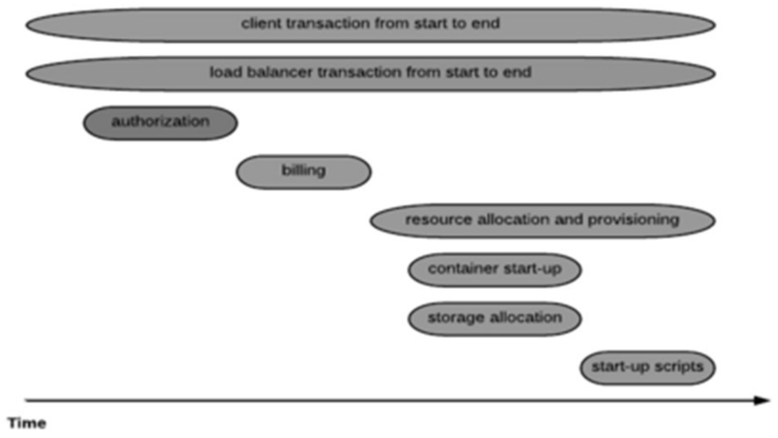


A General Tracing for Microservices-based Applications (retrieved from opentracing.io)

426

9장. 관리 (Management)

Visualization of a Trace Collected via OpenTracing


client transaction from start to end
load balancer transaction from start to end
authorization
billing
resource allocation and provisioning
container start-up
storage allocation
start-up scripts
Time

Visualisation of a Trace Collected via OpenTracing (retrieved from opentracing.io)

9장. 관리 (Management)

Language Support for OpenTracing

- **OpenTracing APIs are available for the following programming languages:**
  1. Go
  2. Python
  3. JavaScript
  4. Java
  5. C Sharp (C#)
  6. Objective-C
  7. C++
  8. Ruby
  9. PHP.

9장. 관리 (Management)

OpenTracing Supported Tracers

- **Using OpenTracing, we collect tracing spans for our services and then forward them to different tracers. Tracers are then used to monitor and troubleshoot microservices-based applications. Following are some of the supported tracers for OpenTracing:**

1. Jaeger
2. LightStep
3. Hawkular APM.

9장. 관리 (Management)

Jaeger

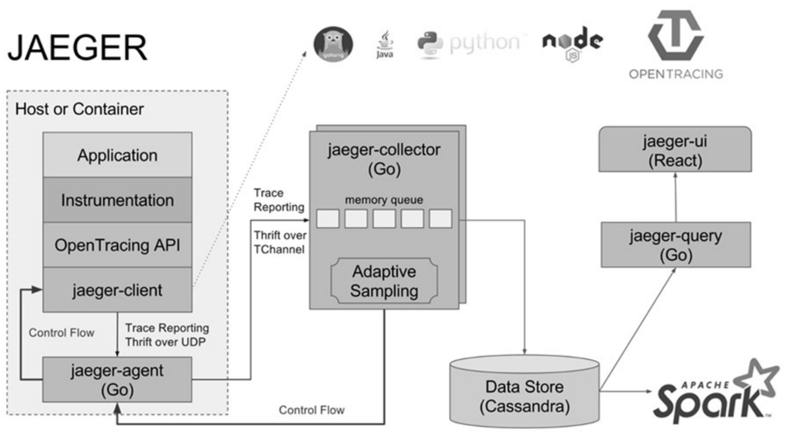
- **Jaeger is an open source tracer which is compatible with the OpenTracing data model to support spans. Jaeger was open sourced by Uber Technologies and is now part of CNCF. It can be used for the following:**

1. Distributed content propagation
2. Distributed transaction monitoring
3. Root cause analysis
4. Service dependency analysis
5. Performance/latency optimization.

Jaeger
Zipkin과 유사하며 집킨 포맷 뿐만 아니라 OpenTracing 표준의 추적 정보 수집 가능

9장. 관리 (Management)

□ **Jaeger Architecture**



Jaeger Architecture (retrieved from jeagertracing.io)

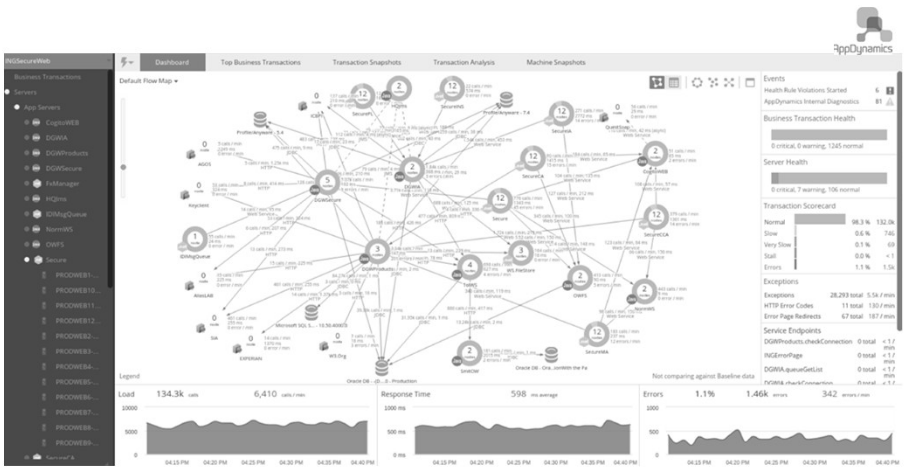
9장. 관리 (Management)

□ **Features and Benefits of Jaeger**

- **Open source tracer, which natively supports OpenTracing.**
- **Support for languages like Java, Go, Python, Node.js and C#.**
- **Highly scalable.**
- **Supports multiple storage backends.**
- **Modern user interface (UI).**
- **Observability via Prometheus.**

9장. 관리 (Management)

□ **monitoring**



https://www.zdnet.com/article/appdynamics-launches-analytics-collaboration-tools/

JS Lab

james@jslab.kr

433

9장. 관리 (Management)

□ **제조사 네트워킹 솔루션의 클라우드 오케스트레이션 연동**

- **클라우드 네이티브(Cloud Native)화 데이터센터 네트워킹 분야 집중**
- **컨테이너, 서비스 메쉬, 마이크로서비스, 변경 불가능 인프라 (Immutable Infrastructure) 및 선언적 API를 사용하는 접근 방식**

제조사	솔루션 이름	오케스트레이션 연동
Big Switch Networks	Big Cloud Fabric	쿠버네티스, 오픈스택, VMware, OpenShift
Huawei	CloudFabric	오픈스택, FusionSphere, ManageOne, Red Hat, Mirantis
Lenovo	RackSwitch	오픈스택, VMware vCloud Suite, Microsoft Azure Stack, Tungsten Fabric
Netronome	Agilio SmartNIC	오픈스택
Plexxi	Plexxi HCN	쿠버네티스, 오픈스택, vCloud, Nutanix
ZTE	ZENIC	쿠버네티스, 오픈스택
Dell EMC	Z9100-ON Switch	쿠버네티스, 오픈스택, VMware vCloud Suite
Altoline	99xx/69xx	쿠버네티스, 오픈스택, VMware vCloud Suite
Mellanox	Open Composable Networks	오픈스택, VMware vCloud Suite, NEO
Cisco	Application Policy Infrastructure Controller (APIC)	VMware vCloud Suite
Ericsson	Cloud SDN	쿠버네티스, 오픈스택

제조사	솔루션 이름	오케스트레이션 연동
Juniper Networks	Contrail	오픈스택
Nuage Networks	Virtualized Services Platform(VSP)	쿠버네티스, 오픈스택, VMware vCloud Suite, 클라우드스택
Pluribus	Netvisor OS, Adaptive Cloud Fabric	VMware vCloud Suite, Ansible, Puppet, Chef
FlowEngine	FlowEngine TDE-2000	오픈스택, VMware vCloud Suite
Red Hat	NFV solution	쿠버네티스, 오픈스택
VMware	NSX	쿠버네티스, 오픈스택, VMware vCloud Suite
Wind River	Titanium Cloud	오픈스택
A10	Thunder ADC	오픈스택, VMware vCloud Suite
Cumulus	Cumulus Linux	오픈스택
ipinfusion	OcNOS	오픈스택
Pulse Secure	Pulse Access Suite	쿠버네티스, 오픈스택, VMware vCloud Suite

JS Lab

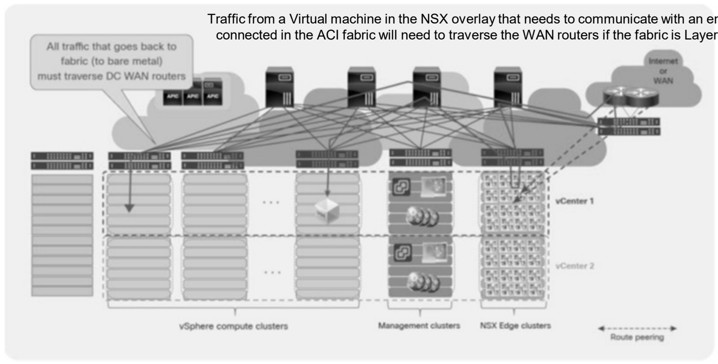
james@jslab.kr

434

9장. 관리 (Management)

- 엔터프라이즈와 텔레콤 환경은 Leaf-Spine 물리 구성 인프라 기반 클라우드 서비스 관리 필요
- VxLAN 터널링 사용 호스트 클러스터링 고려
- 테넌트간 소프트웨어 계층의 routing/switching 관리

Traffic from a Virtual machine in the NSX overlay that needs to communicate with an endpoint connected in the ACI fabric will need to traverse the WAN routers if the fabric is Layer-2 only

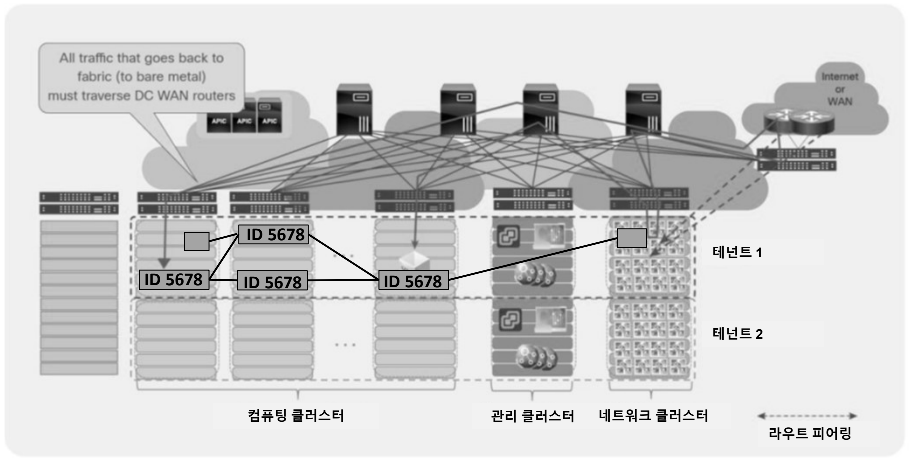


JS Lab

435

9장. 관리 (Management)

- Correlation IDs over Physical Infrastructure

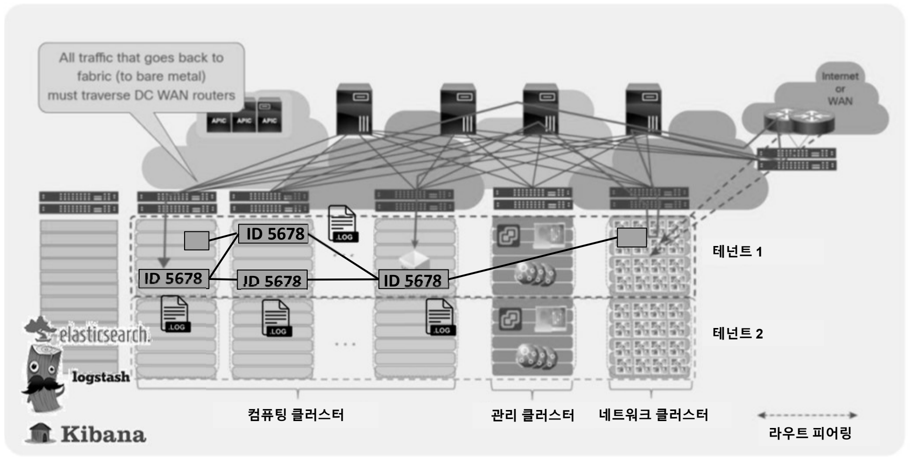


JS Lab

436

9장. 관리 (Management)

- Centralized logging



437

10장. 보안 (Security)

1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ 실습교재 (별도)

438

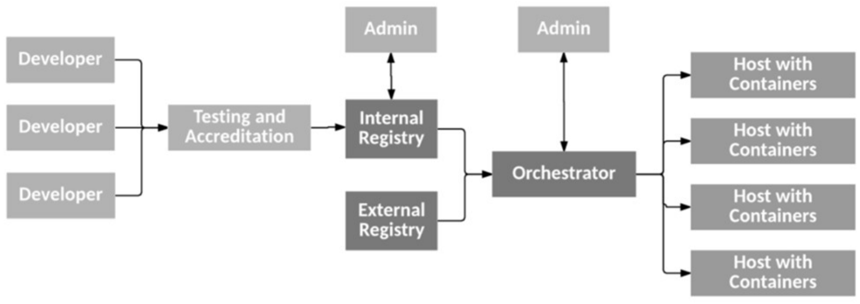
10장. 보안 (Security)

- **가상환경 보안**
- **컨테이너 환경 보안**
- **DevSecOps**

439

10장. 보안 (Security)

□ **Application Container Security Guide (NIST)**

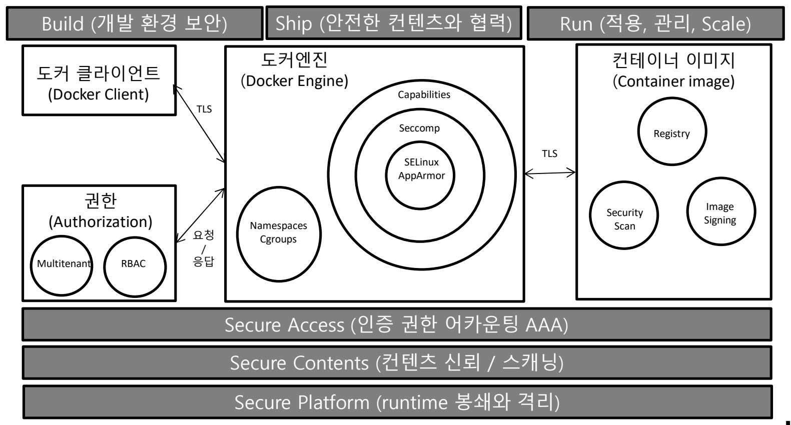


Traditional security solutions, such as intrusion prevention systems (IPSs) and web application firewalls (WAFs), often do not provide suitable protection for containers. They may not be able to operate at the scale of containers, manage the rate of change in a container environment, and have visibility into container activity. Utilize a container-native security solution that can monitor the container environment and provide precise detection of anomalous and malicious activity within it.

440

10장. 보안 (Security)

- **도커 보안 서비스 구성: 구성 요소간 TLS 기반 연결, 인증 서버 사용, 리눅스 제공 보안 기능 사용하며, AAA기반 안전한 접속(Access)/신뢰하는 이미지 등의 컨텐츠(Contents) 제공/방화벽 IDS 등의 보안 강화 플랫폼(Platform) 제공**



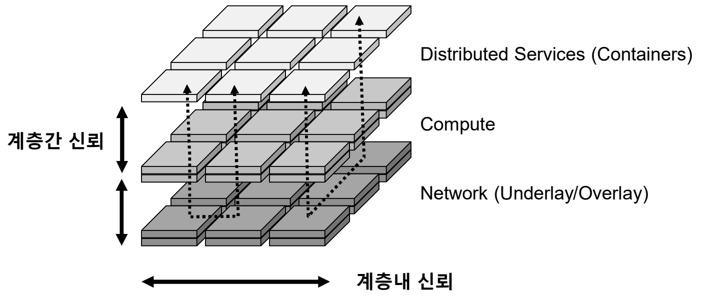
JS Lab

james@jslab.kr

441

10장. 보안 (Security)

- **계층간 보안 고려**
  - **계층간 신뢰: 계층내 인증/터널링 연결하여 계층간 분리, 성능 이슈**
  - **계층내 신뢰: 논리적/물리적 분리로 계층 내의 폐쇄그룹 구성**



JS Lab

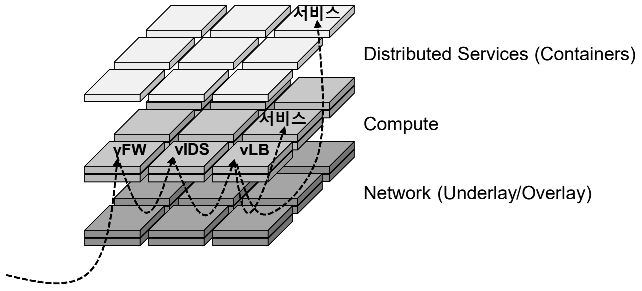
james@jslab.kr

442

10장. 보안 (Security)

Security as a Service (SaaS)

- SFC (Service Function Chaining)
- SR-aware SF (Segment Routing, Linux-based NFV Infrastructure)

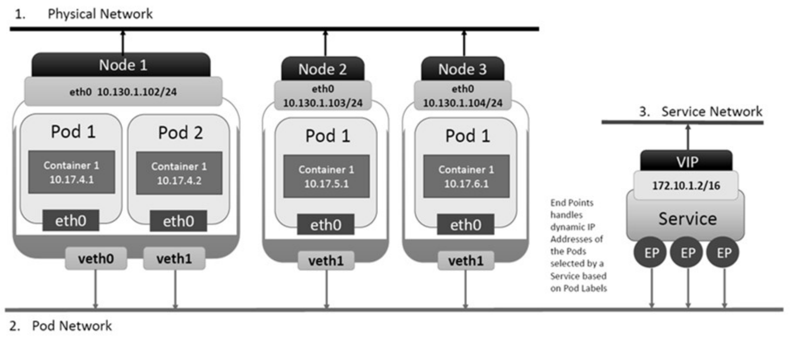


https://datatracker.ietf.org/wg/sfc/documents/ https://datatracker.ietf.org/doc/rfc7498/

10장. 보안 (Security)

쿠버네티스 네트워크

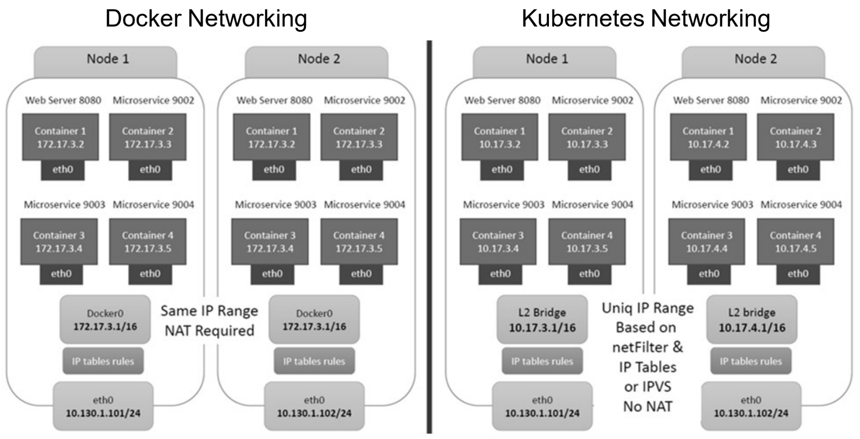
- Physical Network: 10.0.0.0/8
- Pod Network: 172.0.0.0/11
- Service Network: 192.168.0.0/16



https://github.com/meta-magic/kubernetes_workshop

10장. 보안 (Security)

◻ 도커와 쿠버네티스 네트워크 비교



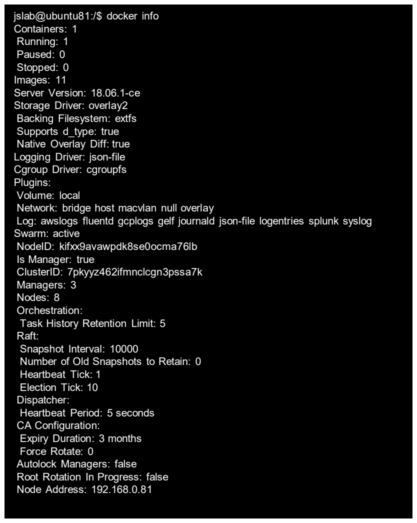
https://github.com/meta-magic/kubernetes_workshop

JS Lab

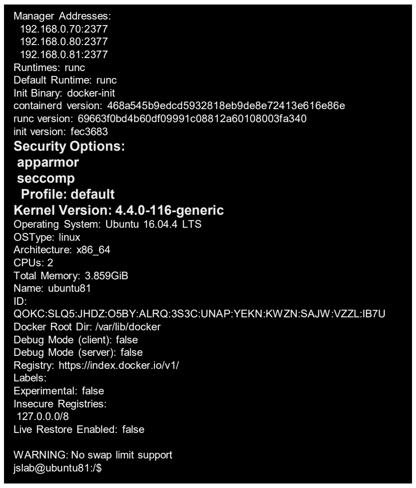
445

10장. 보안 (Security)

◻ Docker Information



```
jslab@ubuntu81:/$ docker info
Containers: 1
 Running: 1
 Paused: 0
 Stopped: 0
Images: 11
Server Version: 18.06.1-ce
Storage Driver: overlay2
 Backing Filesystem: extfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file logentries splunk syslog
Swarm: active
 NodeID: kifxx9avawpdk8se0ocma76lb
 Is Manager: true
 ClusterID: 7pkyyz462ifmnclcgn3pssa7k
 Managers: 3
 Nodes: 8
 Orchestration:
  Task History Retention Limit: 5
 Raft:
  Snapshot Interval: 10000
  Number of Old Snapshots to Retain: 0
  Heartbeat Tick: 1
  Election Tick: 10
 Dispatcher:
  Heartbeat Period: 5 seconds
 CA Configuration:
  Expiry Duration: 3 months
  Force Rotate: 0
 Autolock Managers: false
 Root Rotation In Progress: false
 Node Address: 192.168.0.81
```



```
Manager Addresses:
  192.168.0.70:2377
  192.168.0.80:2377
  192.168.0.81:2377
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 468a545b9edcd5932818eb9de8e72413e616e86e
runc version: 69663f0bd4b60df09991c08812a60108003fa340
init version: fec3683
Security Options:
 apparmor
 seccomp
  Profile: default
Kernel Version: 4.4.0-116-generic
Operating System: Ubuntu 16.04.4 LTS
OSType: linux
Architecture: x86_64
CPUs: 2
Total Memory: 3.859GiB
Name: ubuntu81
ID:
QOKC:SLQ5:JHDZ:O5BY:ALRQ:3S3C:UNAP:YEKN:KWZN:SAJW:VZZL:IB7U
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false

WARNING: No swap limit support
jslab@ubuntu81:/$
```

https://github.com/docker/docker/blob/master/profiles/seccomp/default.json

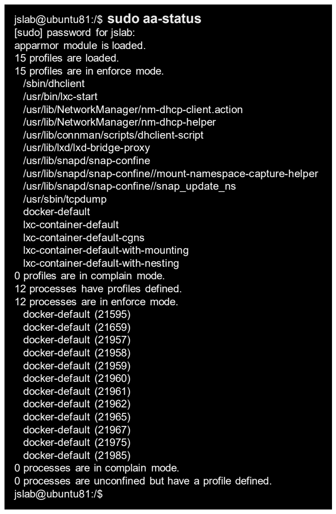
JS Lab

446

10장. 보안 (Security)

- **AppArmor**
  - **리눅스 커널 보안 모듈**
  - **sudo aa-status**
  - **Sane defaults:**
    쓰기 차단: /proc/{num}, /proc/sys, /sys
    Mount 차단
  - **https://github.com/docker/docker/blob/master/profiles/apparmor/template.go**

```
$ docker container run --rm -it /
  --security-opt apparmor=custom-profile hello-world
```



```
jslab@ubuntu81:/$ sudo aa-status
[sudo] password for jslab:
apparmor module is loaded.
15 profiles are loaded.
15 profiles are in enforce mode.
   /sbin/dhclient
   /usr/bin/lxc-start
   /usr/lib/NetworkManager/nm-dhcp-client.action
   /usr/lib/NetworkManager/nm-dhcp-helper
   /usr/lib/connman/scripts/dhclient-script
   /usr/lib/lxd/lxd-bridge-proxy
   /usr/lib/snapd/snap-confine
   /usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /usr/lib/snapd/snap-confine//snap_update_ns
   /usr/sbin/tcpdump
   docker-default
   lxc-container-default
   lxc-container-default-cgns
   lxc-container-default-with-mounting
   lxc-container-default-with-nesting
0 profiles are in complain mode.
12 processes have profiles defined.
12 processes are in enforce mode.
   docker-default (21595)
   docker-default (21659)
   docker-default (21957)
   docker-default (21958)
   docker-default (21959)
   docker-default (21960)
   docker-default (21961)
   docker-default (21962)
   docker-default (21965)
   docker-default (21967)
   docker-default (21975)
   docker-default (21985)
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
jslab@ubuntu81:/$
```

james@jslab.kr

JS Lab

https://github.com/docker/docker/blob/master/profiles/apparmor/template.go

447

10장. 보안 (Security)

- **SecComp**
  - **기본 정책은 40여개의 syscalls 차단 (Linux는 300 syscalls 이상)**
  - **기본 정책의 목표는 민감한 out-of-the-box 정책**
  - **You should consider the default policy as moderately protective while providing wide application compatibility**
  - **https://github.com/docker/docker/blob/master/profiles/seccomp/default.json**

```
$ docker run --rm -it \
  --security-opt seccomp=/path/to/seccomp/profile.json \
  hello-world
```

```
$ docker run --rm -it \
  --security-opt seccomp=unconfined \
  hello-world
```

james@jslab.kr

JS Lab

https://github.com/docker/docker/blob/master/profiles/seccomp/default.json

448

10장. 보안 (Security)

◻ **Application Container Security Guide (NIST)**

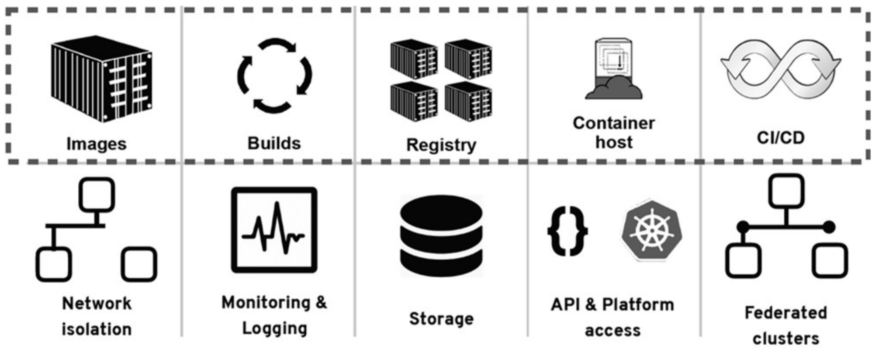
- **Tailor the organization's operational culture and technical processes to support the new way of developing, running, and supporting applications made possible by containers**
- **Use container-specific host OSs instead of general-purpose ones to reduce attack surfaces**
- **Only group containers with the same purpose, sensitivity, and threat posture on a single host OS kernel to allow for additional defense in depth**
- **Adopt container-specific vulnerability management tools and processes for images to prevent compromises**
- **Consider using hardware-based countermeasures to provide a basis for trusted computing.**
- **Use container-aware runtime defense tools.**

NIST(National Institute of Standards and Technology) Special Publication 800-190

10장. 보안 (Security)

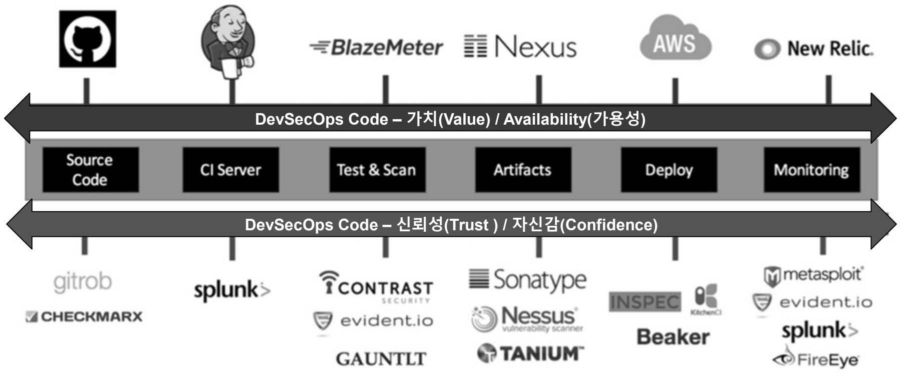
◻ **컨테이너 환경 보안**

- **추가 고려: Images, Builds, Registry, Container Host, CI/CD**



10장. 보안 (Security)

◻ DevSecOps 구축사례

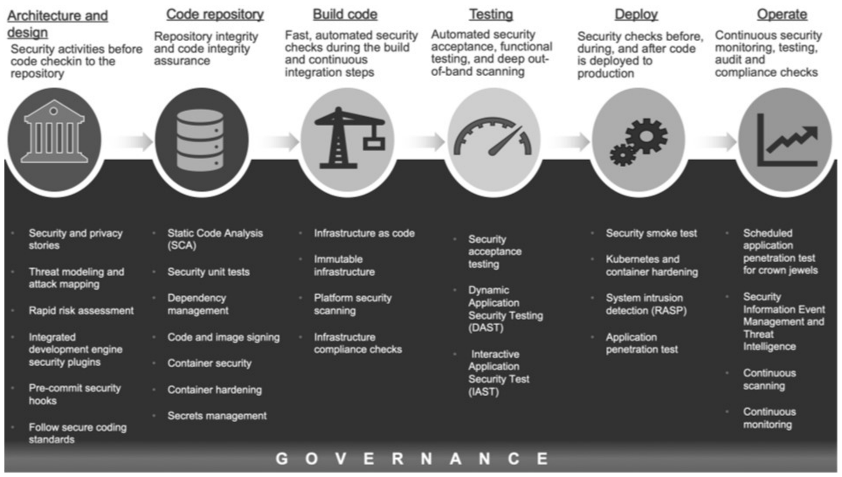


JS Lab

451

10장. 보안 (Security)

◻ Reference DevSecOps lifecycle (IBM)



Architecture and design
Security activities before code checkin to the repository

- Security and privacy stories
- Threat modeling and attack mapping
- Rapid risk assessment
- Integrated development engine security plugins
- Pre-commit security hooks
- Follow secure coding standards

Code repository
Repository integrity and code integrity assurance

- Static Code Analysis (SCA)
- Security unit tests
- Dependency management
- Code and image signing
- Container security
- Container hardening
- Secrets management

Build code
Fast, automated security checks during the build and continuous integration steps

- Infrastructure as code
- Immutable infrastructure
- Platform security scanning
- Infrastructure compliance checks

Testing
Automated security acceptance, functional testing, and deep out-of-band scanning

- Security acceptance testing
- Dynamic Application Security Testing (DAST)
- Interactive Application Security Test (IAST)

Deploy
Security checks before, during, and after code is deployed to production

- Security smoke test
- Kubernetes and container hardening
- System intrusion detection (RASP)
- Application penetration test

Operate
Continuous security monitoring, testing, audit and compliance checks

- Scheduled application penetration test for crown jewels
- Security Information Event Management and Threat Intelligence
- Continuous scanning
- Continuous monitoring

G O V E R N A N C E

JS Lab

452

1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ **실습교재 (별도)**

11장. 디자인 패턴 (Design Pattern)

- **구성**
- **Design Pattern References**

11장. 디자인 패턴 (Design Pattern)

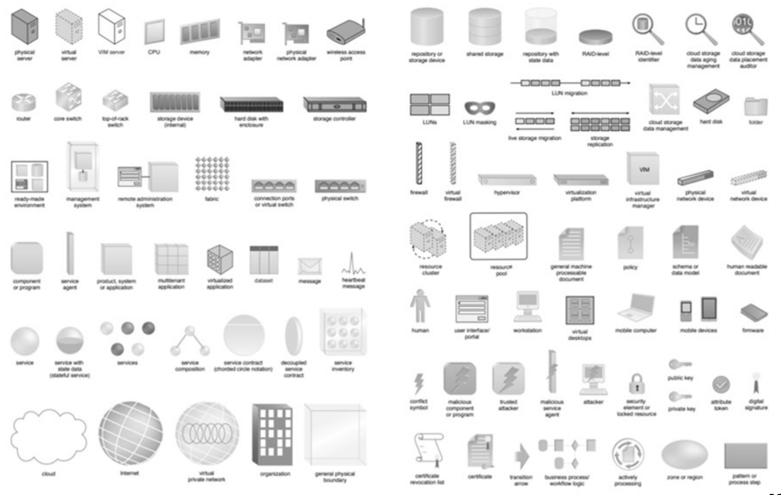
□ 클라우드 디자인 패턴

- 디자인 패턴은 클라우드에서 안정적이고 확장성 있는 안전한 애플리케이션을 빌드하는 데 유용
- 각 패턴은 패턴이 해결하는 문제, 패턴을 적용하기 위한 고려
- 클라우드 플랫폼에 호스팅, 분산 시스템과 관련
  - □ 가용성
  - □ 데이터 관리
  - □ 디자인 및 구현
  - □ 메시징
  - □ 관리 및 모니터링
  - □ 성능 및 확장성
  - □ 복원력
  - □ 보안

455

11장. 디자인 패턴 (Design Pattern)

□ 클라우드 디자인 패턴 구성 (2015)

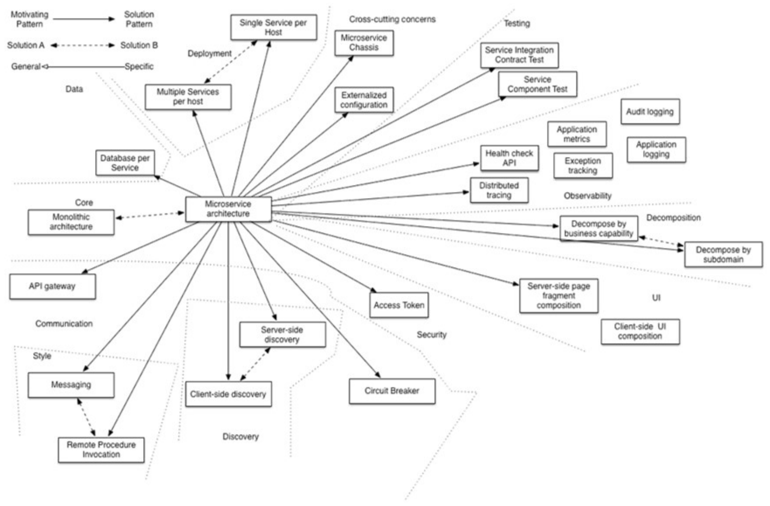


Erl, Thomas. Cloud Computing Design Patterns (The Prentice Hall Service Technology Series from Thomas Erl) . Pearson Education.

456

11장. 디자인 패턴 (Design Pattern)

◻ Pattern: Microservice Architecture



https://microservices.io/patterns/microservices.html

JS Lab

457

11장. 디자인 패턴 (Design Pattern)

◻ 패턴 카탈로그 (MS Azure 예)

특사	소비자 서비스 또는 애플리케이션을 대신하여 네트워크 요청을 전송하는 도우미 서비스를 만듭니다.
손상 방지 레이어	현대식 애플리케이션과 레거시 시스템 사이에 외관 또는 어댑터 레이어를 구현합니다.
프런트 엔드에 대한 백 엔드	특정 프런트 엔드 애플리케이션 또는 인터페이스에서 사용할 별도의 백 엔드 서비스를 만듭니다.
격벽	하나가 고장 나더라도 나머지는 정상적으로 작동하도록 애플리케이션의 요소를 여러 풀에 격리합니다.
Cache-Aside	필요할 때 데이터를 데이터 저장소에서 캐시로 로드
연출	중앙 오케스트레이터에 의존하는 대신 각 서비스에서 비즈니스 작업이 처리되는 시기와 방법을 결정하도록 합니다.
회로 차단기	원격 서비스 또는 리소스에 연결할 때 해결하는 데 걸리는 시간이 유동적인 오류를 처리합니다.
클레임 검사	큰 메시지를 클레임 검사 및 페이로드로 분할하면 메시지 버스의 과부하를 피할 수 있습니다.
보정 트랜잭션	여러 단계로 나뉘어 있지만 결국에는 일관적인 작업을 정의하는 일련의 단계에서 수행한 작업을 실행 취소합니다.
경쟁 소비자	여러 동시 소비자가 동일한 메시징 채널에 수신된 메시지를 처리할 수 있게 해 줍니다.
컴퓨팅 리소스 통합	여러 작업을 단일 계산 단위로 통합합니다.
CQRS	별도의 인터페이스를 사용하여 데이터를 업데이트하는 작업과 데이터를 읽는 작업을 분리합니다.
이벤트 소싱	추가 전용 저장소를 사용하여 도메인의 데이터에 대해 수행된 작업을 설명하는 일련의 이벤트 전체를 기록합니다.
외부 구성 저장소	구성 정보를 애플리케이션 배포 패키지에서 중앙 위치로 이동합니다.
페더레이션 ID	외부 ID 공급자에게 인증을 위임합니다.
게이트 키퍼	클라이언트와 애플리케이션 또는 서비스 간 브로커 역할을 하며, 요청을 검사 및 정리하고, 요청 및 데이터를 전달하는 전용 호스트 인스턴스를 사용하여 애플리케이션 및 서비스를 보호합니다.
게이트웨이 집계	게이트웨이를 사용하여 여러 개별 요청을 단일 요청으로 집계합니다.
게이트웨이 오프로딩	공유 또는 특수 서비스 기능을 게이트웨이 프록시에 오프로드합니다.
게이트웨이 라우팅	단일 엔드포인트를 사용하여 요청을 여러 서비스에 라우팅합니다.
상태 엔드포인트 모니터링	외부 도구가 노출된 엔드포인트를 통해 주기적으로 액세스할 수 있는 기능 검사를 애플리케이션 내부에 구현합니다.
인덱스 테이블	쿼리에서 자주 참조하는 데이터 저장소의 필드에 대한 인덱스를 만듭니다.
리더 선택	인스턴스 중 하나를 다른 인스턴스를 관리하는 리더로 선택하여 분산된 애플리케이션의 공동 작업 인스턴스 컬렉션이 수행하는 작업을 조정합니다.
구체화된 뷰	데이터가 필요한 쿼리 작업에 대해 이상적으로 포맷되지 않은 경우 하나 이상의 데이터 저장소에 있는 데이터에 대한 미리 채워진 뷰를 생성합니다.
파이프 및 필터	복잡한 처리를 수행하는 작업을 재사용 가능한 일련의 별도 요소로 분류합니다.
우선 순위 큐	우선 순위가 높은 요청을 우선 순위가 낮은 요청보다 먼저 받아서 처리하도록 서비스로 전송된 요청의 우선 순위를 지정합니다.
게시자/구독자	애플리케이션이 발신자와 수신자를 연결하지 않고 여러 관심 있는 소비자에게 이벤트를 비동기적으로 알릴 수 있습니다.
큐 기반 부하 평준화	작업 그리고 그 작업이 일시적인 높은 부하를 부드럽게 처리하기 위해 호출하는 서비스 사이에서 버퍼 역할을 하는 큐를 사용합니다.
다시 시도	이전에 실패한 작업을 투명하게 다시 시도하여 서비스 또는 네트워크 리소스에 연결하려 할 때 애플리케이션을 사용하여 예상된 일시적 오류를 처리합니다.
Scheduler 에이전트 감독자	서비스 및 기타 원격 리소스의 분산된 집합에서 일련의 작업을 조정합니다.
분할	데이터 저장소를 수평 파티션 또는 분할 집합으로 나눕니다.
사이드카	격리 및 캡슐화를 제공하는 별도의 프로세스 또는 컨테이너에 애플리케이션 구성 요소를 배포합니다.
정적 콘텐츠 호스팅	정적 콘텐츠를 클라이언트에 직접 제공할 수 있는 클라우드 기반 스토리지 서비스에 배포합니다.
스트랭글러	특정 기능을 새로운 애플리케이션 및 서비스로 점진적으로 교체하여 레거시 시스템을 단계적으로 마이그레이션합니다.
제한	애플리케이션 인스턴스, 개별 테넌트 또는 서비스 전체의 리소스 사용량을 제어합니다.
발레 키	클라이언트에 특정 리소스 또는 서비스에 대한 제한된 직접 액세스를 제공하는 토큰 또는 키를 사용합니다.

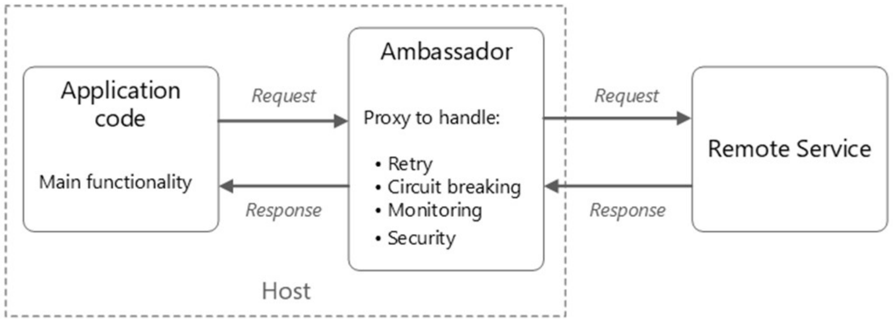
https://docs.microsoft.com/ko-kr/azure/architecture/patterns/

JS Lab

458

11장. 디자인 패턴 (Design Pattern)

□ 특사(Ambassador) 패턴

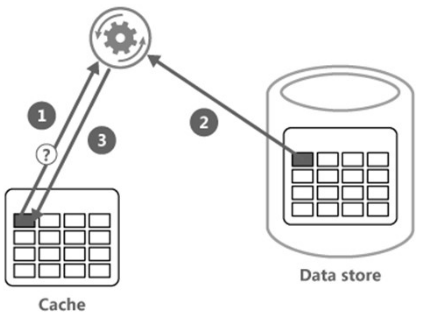


https://docs.microsoft.com/ko-kr/azure/architecture/patterns/

459

11장. 디자인 패턴 (Design Pattern)

□ Cache-Aside pattern



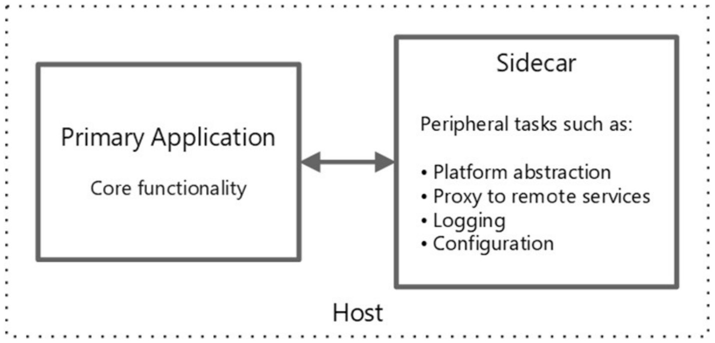
1: Determine whether the item is currently held in the cache.
2: If the item is not currently in the cache, read the item from the data store.
3: Store a copy of the item in the cache.

https://docs.microsoft.com/ko-kr/azure/architecture/patterns/

460

11장. 디자인 패턴 (Design Pattern)

□ 사이드카(Sidecar) 패턴



https://docs.microsoft.com/ko-kr/azure/architecture/patterns/

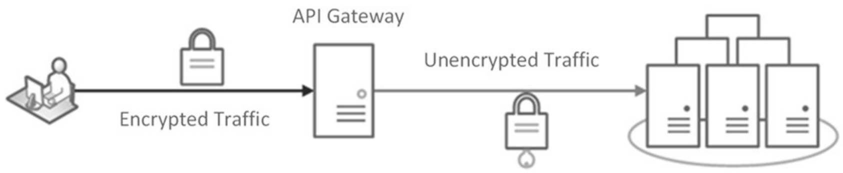
JS Lab

james@jslab.kr

461

11장. 디자인 패턴 (Design Pattern)

□ 게이트웨이 오프로딩 패턴



https://docs.microsoft.com/ko-kr/azure/architecture/patterns/

JS Lab

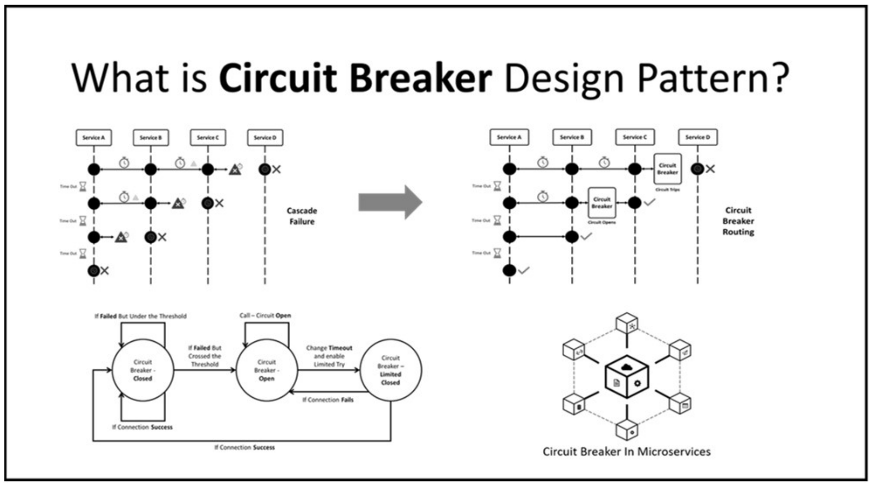
james@jslab.kr

462

james@jslab.kr

11장. 디자인 패턴 (Design Pattern)

- What is Circuit Breaker Design Pattern?



https://digitalvarys.com/what-is-circuit-breaker-design-pattern/?fbclid=IwAR1uukaeNh-vJZXeLhQP2MZvpeq7NVeNVKHmjNEPubzrH8vISlwr_Dedph8

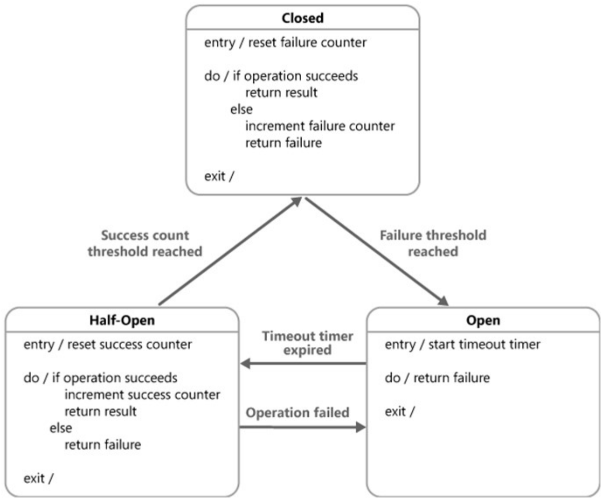
JS Lab

463

james@jslab.kr

11장. 디자인 패턴 (Design Pattern)

- 회로 차단기 패턴


Closed
entry / reset failure counter
do / if operation succeeds
return result
else
increment failure counter
return failure
exit /
Success count threshold reached
Failure threshold reached
Half-Open
entry / reset success counter
do / if operation succeeds
increment success counter
return result
else
return failure
exit /
Timeout timer expired
Operation failed
Open
entry / start timeout timer
do / return failure
exit /

https://docs.microsoft.com/ko-kr/azure/architecture/patterns/

JS Lab

464

1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ **실습교재 (별도)**

465

12장. Use Case

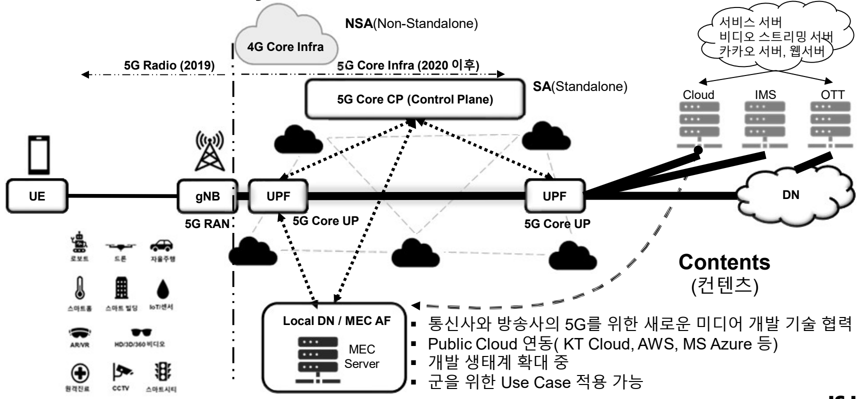
- **5G Core Infrastructure**
- **AI Infrastructure**
- **IoT**

466

12장. Use Case

□ **5G와 MEC (Mobile | Multi-access Edge Computing)**

- **에지의 데이터센터: 국사의 데이터센터화 기지국 확대 고려**
- **5G는 4G EPC 코어 공유 서비스 시작: 5G 코어 적용 확대 중**
- **MEC는 Eco-system 확대 영역: API 제공 및 B2B 등의 모델 확대**



- 통신사와 방송사의 5G를 위한 새로운 미디어 개발 기술 협력
- Public Cloud 연동(KT Cloud, AWS, MS Azure 등)
- 개발 생태계 확대 중
- 군을 위한 Use Case 적용 가능

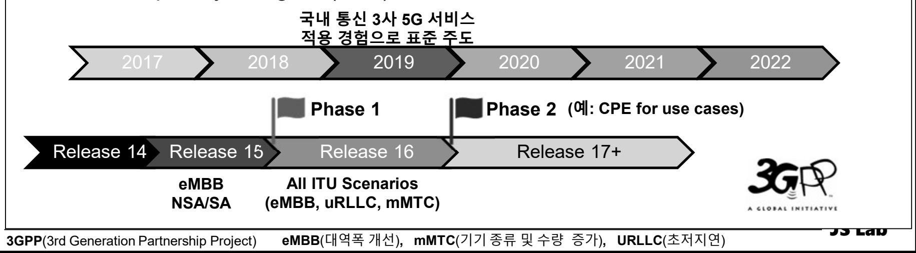
RAN(Radio Access Network) **MEC**(Mobile | Multi-access Edge Computing) **AF**(Application Function) **UPF**(User Plane Function)

467

12장. Use Case

□ **5G 표준 Roadmap 고려**

- **Phase 1 (3GPP Rel. 15, 2018년 6월)**
- **Phase 2 (3GPP Rel. 16, 2019년 12월 이후 freeze 예상)**
- **3GPP Rel. 17은 5G 개선 (2020년 시작)**
- **국내 통신 3사 5G 서비스 시작 (2019년)**
- **표준 적용은 대개 18개월정도 예상**
- **3GPP는 5G Radio 주파수를 2 부분으로 진행중**
  Frequency Range 1 (FR1): 450 MHz – 7.125 GHz
  Frequency Range 2 (FR2): 24.25 GHz – 52.6 GHz



3GPP(3rd Generation Partnership Project) **eMBB**(대역폭 개선), **mMTC**(기기 종류 및 수량 증가), **URLLC**(초저지연)

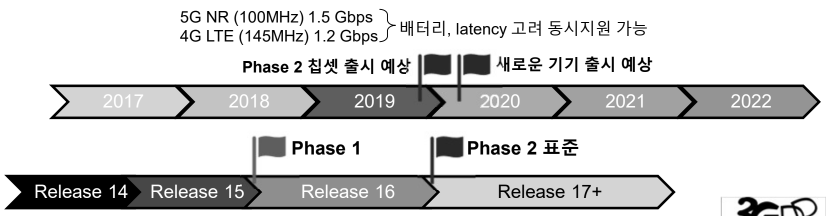
468

12장. Use Case

5G 표준과 Market의 Radio 환경 변화/발전

- **Phase 1:** Chipset, Device, Operator
- **Phase 2:** Next Chipset, Next Device, Full Scale Commercial Service

주파수	Sub 6GHz		Above 6GHz		
Operator	<3GHz	3~5 GHz	6~24 GHz	24~30 GHz	30~40 GHz
SKT		3.6~3.7 GHz (100MHz)		28.1~29.0 GHz (900MHz)	
KT		3.5~3.6 GHz (100MHz)		26.5~27.3 GHz (800MHz)	
LGU+		3.42~3.5 GHz (80MHz)		27.3~28.1 GHz (800MHz)	



- **Large Bandwidth:** Cband(~100MHz) / mmWave(~400MHz)
- **New Air Interface:** f-OFDM, Polar Code, LDPC, UL & DL decoupling
- **Massive MIMO:** 4T4R → 64T64R

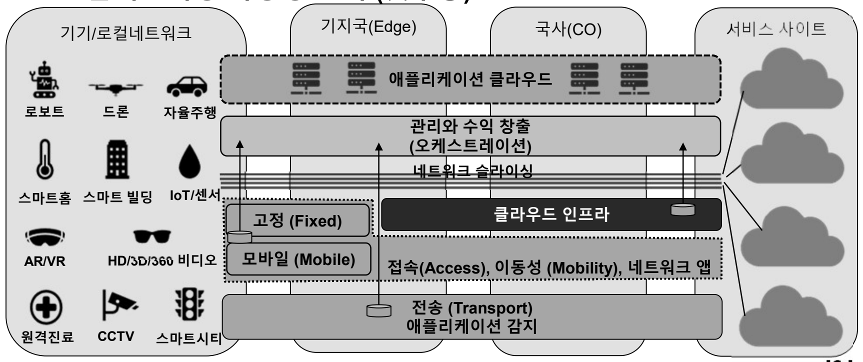
JS Lab

469

12장. Use Case

5G 코어 인프라(Core Infra)

- **Edge(기지국)와 Central Office(국사)의 데이터센터 화 진행**
- **클라우드 네이티브화:** (오픈소스 기반 App 서비스, 관리, 인프라)
- **네트워크 슬라이싱 (종단간 Network Slicing)**
- **클러스터링 확장성 고려 (갯수등)**



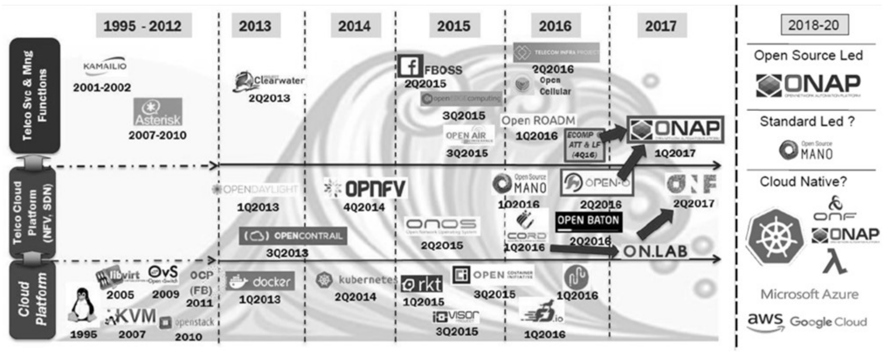
JS Lab

470

12장. Use Case

□ Telecom 운영지원 시스템(OSS)의 미래

- **오픈소스 (예):** Open sources like Linux, OpenStack, KVM and others coming from the IT cloud platform, are becoming the foundation for a telco cloud platform based on network functions virtualization (NFV) and software-defined networking (SDN).



https://inform.tmforum.org/features-and-analysis/2017/03/open-source-future-oss/ **ONAP**(Open Networking Automation Platform)

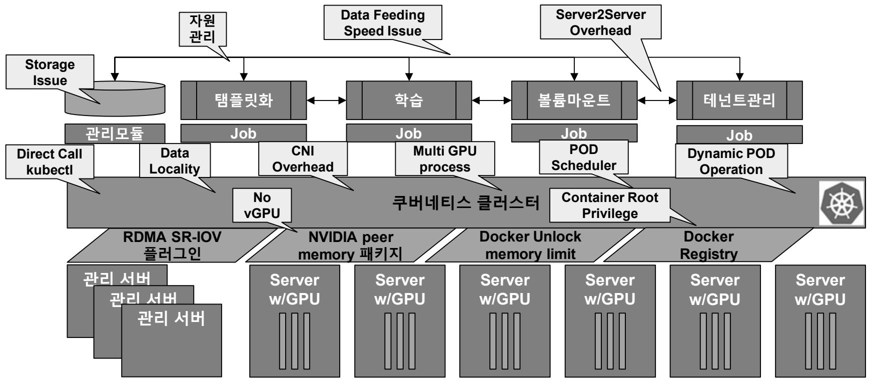
JS Lab

471

12장. Use Case

□ MSA 수용 클라우드 네이티브 아키텍처 ‘CNA’ 체계

- **K8s Based ML Platform**

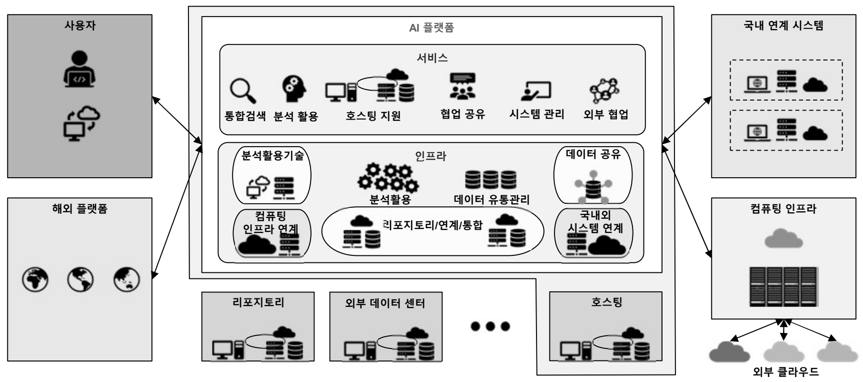


JS Lab

472

12장. Use Case

- 클라우드 네이디브 기반 AI 플랫폼 서비스
  - 플랫폼 서비스 제공
  - 클라우드 서비스 인프라 기반 서비스
  - 클라우드 서비스를 위한 SDDC 인프라 기반 추상화 계층 제공



12장. Use Case

- IoT
- **According to Wikipedia:** ”The Internet of Things (IoT) is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these things to connect and exchange data, creating opportunities for more direct integration of the physical world into computer-based systems, resulting in efficiency improvements, economic benefits and reduced human exertions".

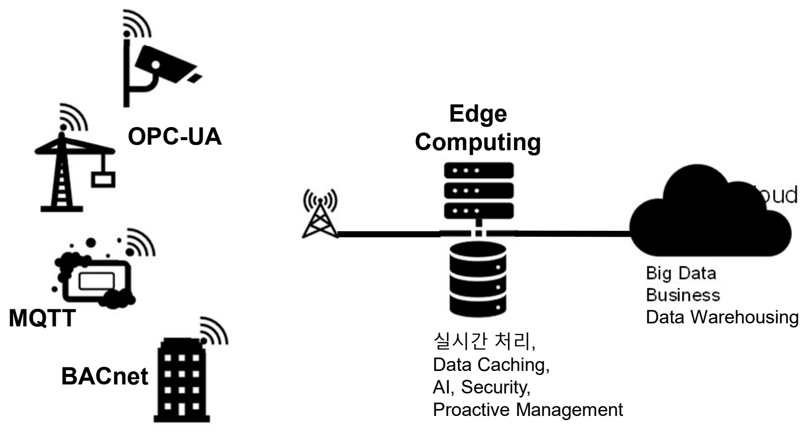
12장. Use Case

- **IoT Use Cases**
  - **Consumer Applications:** Wearable devices like watches, connected vehicles, smart home.
  - **Infrastructure:** Monitoring and controlling railway tracks and wind turbines. IoT also offers preventive maintenance.
  - **Manufacturing:** Asset management, optimizing supply-chain.
  - **Agriculture:** Collecting data on temperature, rainfall, humidity, soil content, etc.
  - **Energy Management:** Optimizing energy consumption.
  - **Environmental Monitoring:** Monitoring air and water quality, soil and atmospheric conditions, etc.
  - **Medical and Healthcare:** Remote health monitoring and emergency notification, etc.

475

12장. Use Case

- 에지 컴퓨팅 (Edge Computing) :데이터를 발생하는 사물 옆이나 내장하는 형태의 컴퓨팅



BACnet Building Automation and Control (BAC) networks **OPC** Unified Architecture (**OPC UA**) - Open Platform Communications

476

12장. Use Case

Network for IoT

- **Short-range Wireless:** Bluetooth mesh networking, Light fidelity (Li-Fi), Radio-frequency identification (RFID), Near-field communication (NFC), Wi-Fi, Zigbee
- **Medium-range Wireless:** Wi-Fi HaLow, LTE Advanced
- **Long-range Wireless:** Low-Power Wide-Area Network (LPWAN), Very small aperture terminal (VSAT)
- **Wired:** Ethernet, Multimedia over Coax Alliance (MoCA), Power-line communication (PLC)

12장. Use Case

Computing for IoT

- **Similar to networking, computing has evolved for IoT. We would need to handle situations like following:**
  1. Latency between the actual device and the datacenter (cloud) where we store the actual data.
  2. Not sending unnecessary data, like frequent keep-alive messages from the devices/sensors to the backend datacenter.
- **To handle situations like those above, Distributed Computing Architecture has evolved to include edge and fog computing. By using edge and fog computing we bring computing applications, data, and services away from some central nodes (datacenter) to the other logical extreme (the edge) of the Internet.**

12장. Use Case

- Data Management and Analytics for IoT
  - **IoT devices are just one part of the ecosystem. We need to regularly collect data from them, transmit it, store it and analyze it to make smart decisions. In some cases, we need to make decisions in real time, such as in the case of self-driving cars.**
  - **With IoT devices, we generate a variety of data in large volumes at very high speeds, which makes very good use for Big Data technologies like Apache Hadoop. We can make smart decisions, then augment this via machine learning and artificial intelligence.**

12장. Use Case

- IoT Solutions Provided by Different Cloud Providers
  - **Amazon Web Services (AWS):** AWS IoT services offers product and services like Amazon FreeRTOS, AWS IoT Core, AWS IoT Device Management, AWS IoT Analytics, and many others.
  - **Google Cloud Platform (GCP):** Google Cloud IoT is the IoT offering from Google Cloud Platform.
  - **Microsoft Azure:** Azure IoT offers services like IoT Hub, IoT Central, etc. It also has the Azure IoT Edge service to provide artificial intelligence (AI), Azure services, and custom logic directly on cross-platform IoT devices.

12장. Use Case

IoT Challenges

- **Scale:** With millions of devices added every year, we need to have infrastructure to support their network with specific storage requirements.
- **Security:** How can we make sure that there is always a secure communication between connection endpoints? Also, if there is a breach, how much information can one get from the system?
- **Privacy:** There are great concerns about privacy. For example, should one share his/her entire health information to some third party companies/services?
- **Interoperability:** Newer devices have the interfaces for IoT communication, but what about other legacy devices that are not IoT-enabled?

1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud
❖ **실습교재 (별도)**

별첨

- Docker
- K8s
- How to Be Successful in the Cloud

별첨

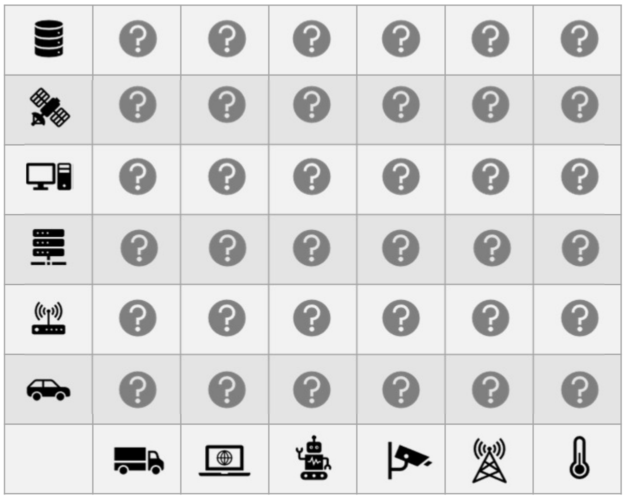
□ **Trends – App Dev / Infrastructure**

- **Cloud-Native (public and private clouds)**
- **Containers**
- **Micro-service Architecture**

Device-Native @ Telco Cloud

별첨

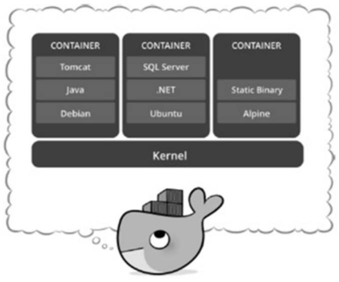
- Device Mesh 환경을 위한 Device Native 기술 필요



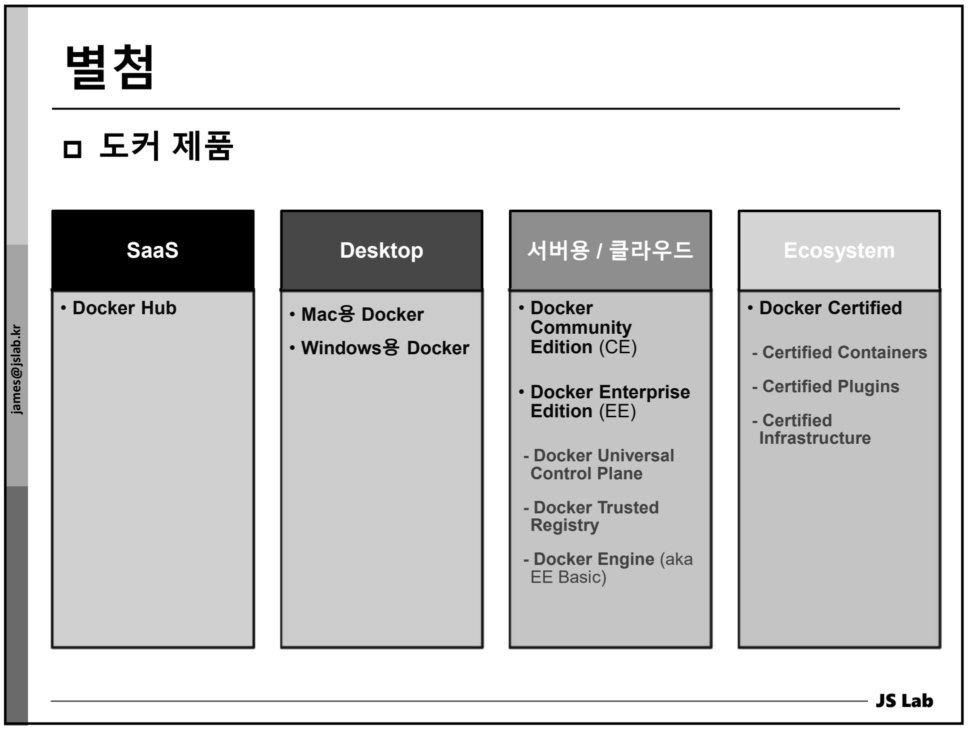
485

별첨

- 도커(Docker) 란?
  - 패키징 표준화
  - 앱(App)의존성 분리
  - 동일 OS 커널 공유
  - 모든 주요 Linux 계열에서 동작
  - 윈도우 서버 2016 / 1809 이상 에서도 지원



486



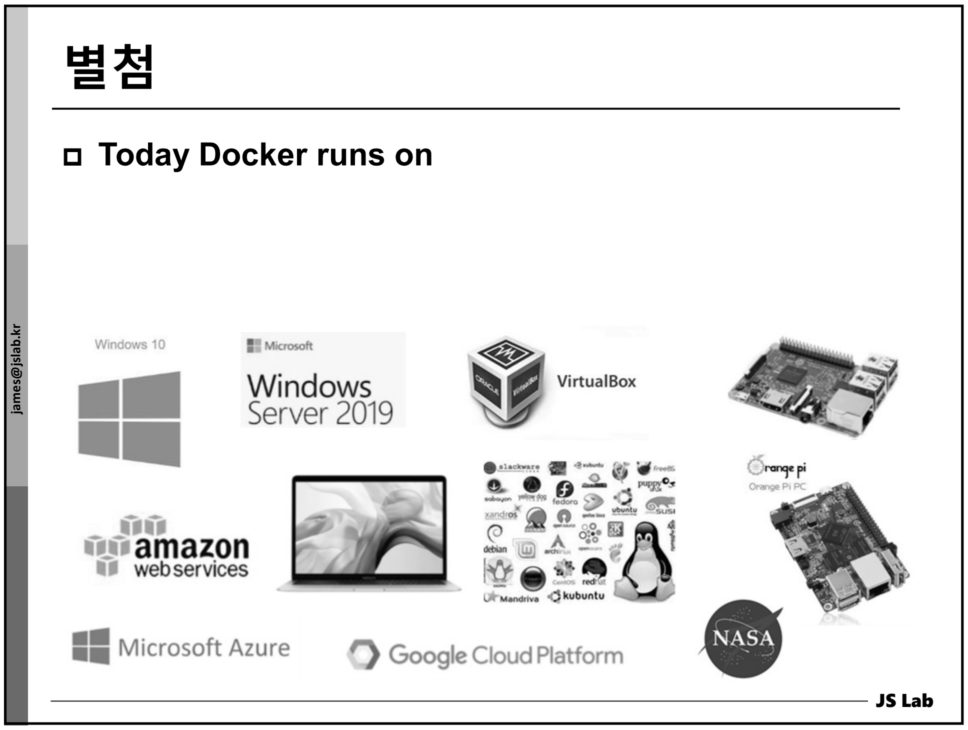
별첨

◻ 도커 제품

SaaS	Desktop	서버용 / 클라우드	Ecosystem
• Docker Hub	• Mac용 Docker • Windows용 Docker	• Docker Community Edition (CE) • Docker Enterprise Edition (EE) - Docker Universal Control Plane - Docker Trusted Registry - Docker Engine (aka EE Basic)	• Docker Certified - Certified Containers - Certified Plugins - Certified Infrastructure

JS Lab

487



별첨

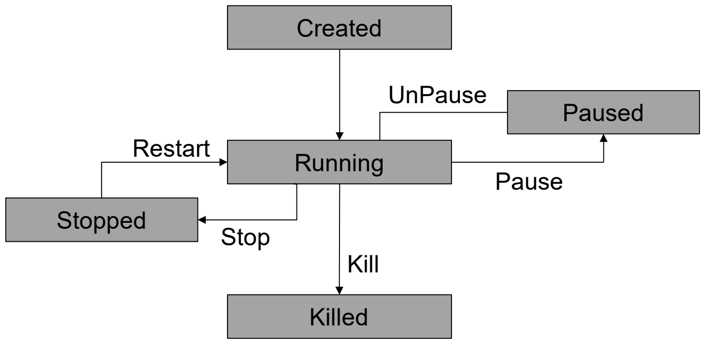
◻ Today Docker runs on

JS Lab

488

별첨

- 도커 컨테이너 라이프 사이클: Build한 이미지(Image)는 컨테이너 실행 시 읽기 전용으로 사용하며 컨테이너에서 생성한 계층에서 쓰기와 읽기를 실행



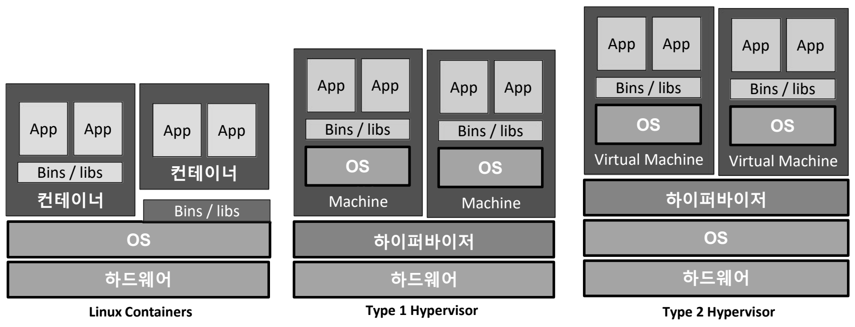
james@jslab.kr

JS Lab

489

별첨

- 도커 컨테이너: 동시에 복수개의 루트파일 시스템을 실행하는 호스트 리눅스커널을 사용하며, 각각의 루트파일 시스템을 컨테이너(Container)라고 함
  - 가상머신보다 적은 자원 사용
  - 스케일링을 가능하게 해주는 관리 도구 포함



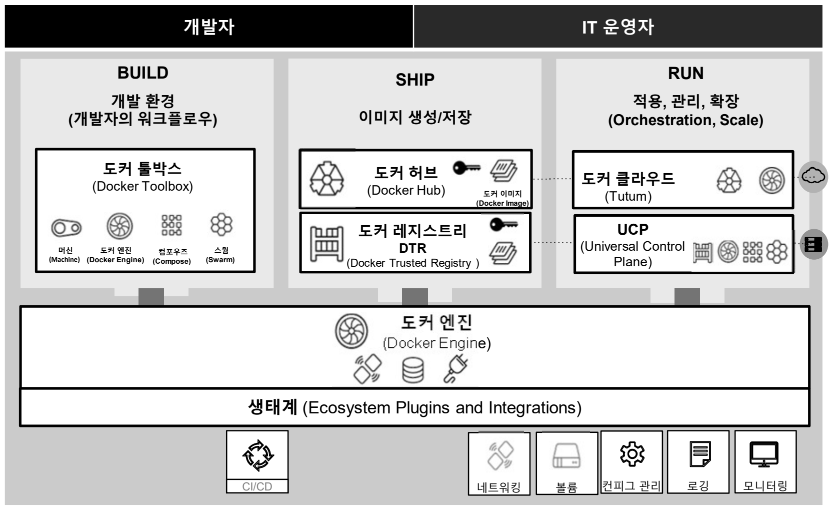
james@jslab.kr

JS Lab

490

별첨

□ 워크 플로우(Workflow) @ 도커



491

별첨

□ Docker CE요약

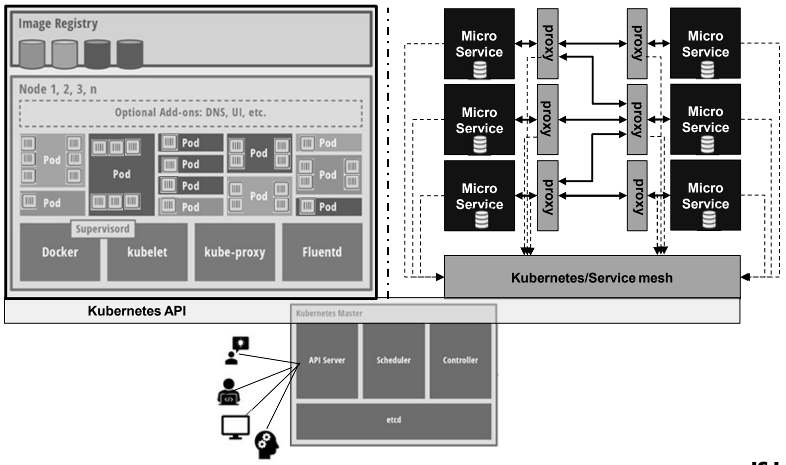
- Container : 호스트 운영체제와 독립적인 환경을 생성하는 개념
- 도커(Docker): 소프트웨어 컨테이너 플랫폼
- 도커 기본 기능: 이미지, 컨테이너, 엔진, 레지스트리
- 오케스트레이션: 도커 머신, 도커 스웜, 도커 컴포우즈

기본 기능	오케스트레이션
도커 이미지 (Docker Image)	머신 (Machine) : 도커 설치한 인프라 제공
도커 컨테이너 (Docker Container)	스웜 (Swarm) : 호스트 클러스터링 제공 (Docker 1.12 이후 내장)
도커 엔진 (Docker Engine)	컴포우즈 (Compose) : 복수 컨테이너 적용 (Docker 1.13 이후 내장)
레지스트리 서비스 (Docker Hub / Docker Trusted Registry)	

492

별첨

- K8s master: API, Scheduler, Controller
- Application을 위한 Compute, Networking, Storage



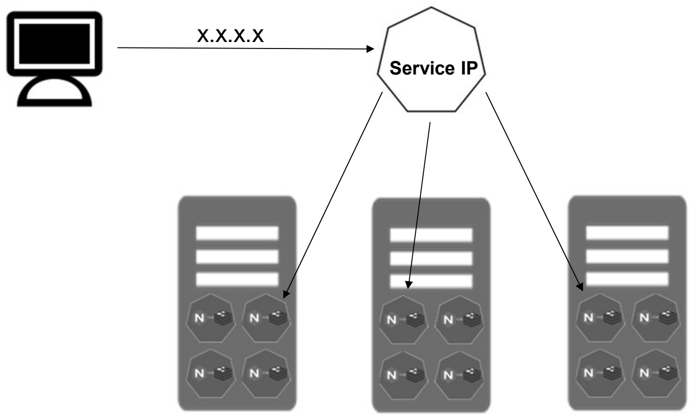
james@jslab.kr

JS Lab

493

별첨

- Services @ K8s



james@jslab.kr

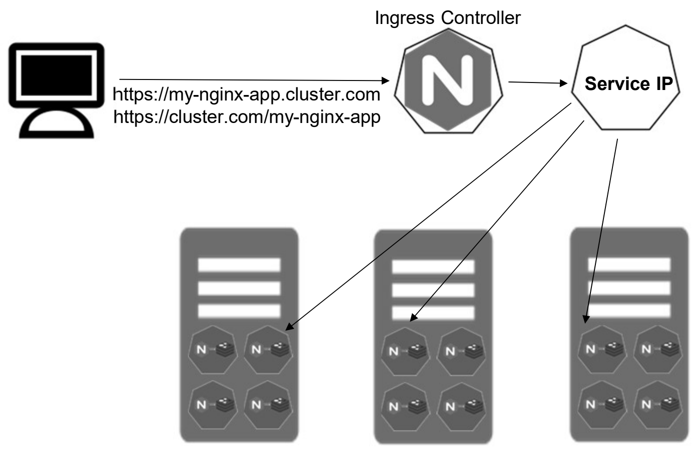
https://rancher.com/announcing-rancher-2-0/

JS Lab

494

별첨

Ingresses @ K8s


Ingress Controller
https://my-nginx-app.cluster.com
https://cluster.com/my-nginx-app
Service IP

https://rancher.com/announcing-rancher-2-0/

JS Lab

495

별첨

Config maps @ K8s

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: dev-config
  namespace:
default
data:
  Item1: dev-stuff
  Item2: more dev
stuff
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: dev-config
  namespace:
default
data:
  Item1: qa-stuff
  Item2: more qa
stuff
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: dev-config
  namespace: default
data:
  Item1: prod-stuff
  Item2: more prod
stuff
```

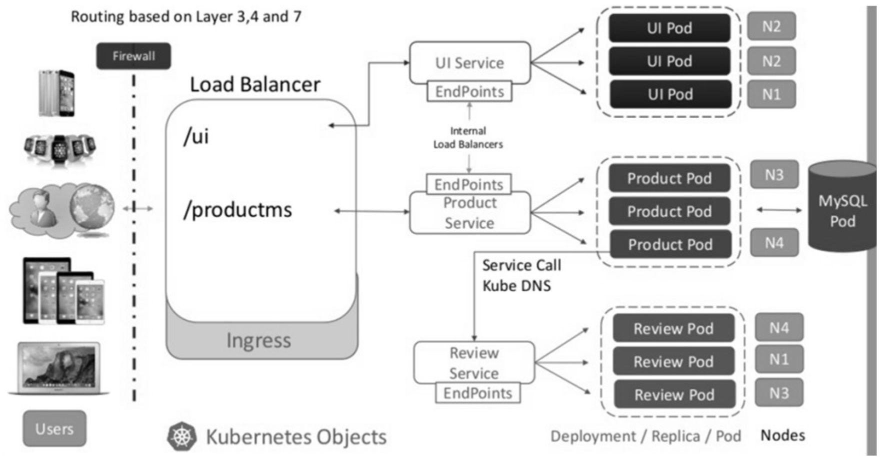
https://rancher.com/announcing-rancher-2-0/

JS Lab

496

별첨

- **Docker / K8s – Network Stack** (Shopping Portal 예)
- **Routing based on L3/L4/L7**

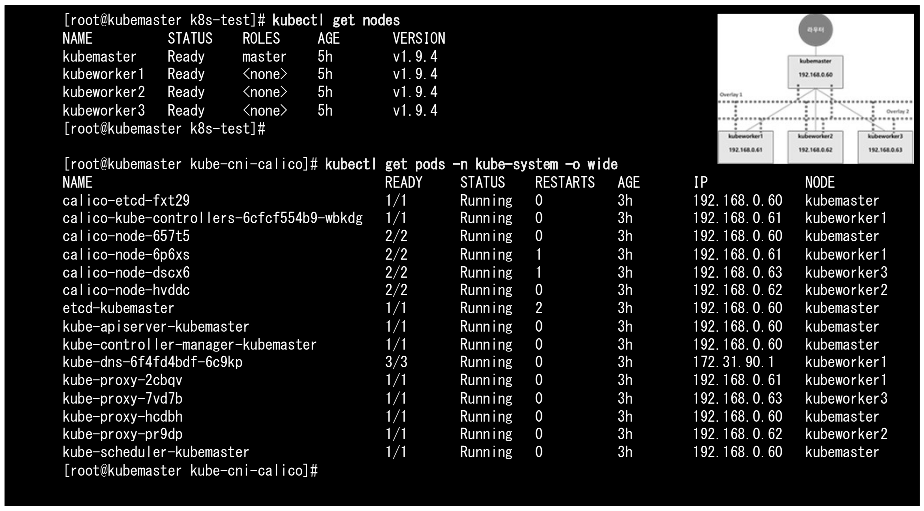


JS Lab

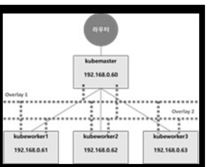
497

별첨

- **설치 구성(예): K8s 설치 호스트 노드와 Pod 구성 확인**



```
[root@kubemaster k8s-test]# kubectl get nodes
NAME          STATUS    ROLES     AGE       VERSION
kubemaster    Ready     master    5h        v1.9.4
kubeworker1   Ready     <none>    5h        v1.9.4
kubeworker2   Ready     <none>    5h        v1.9.4
kubeworker3   Ready     <none>    5h        v1.9.4
[root@kubemaster k8s-test]#

[root@kubemaster kube-cni-calico]# kubectl get pods -n kube-system -o wide
NAME                                       READY     STATUS    RESTARTS   AGE       IP              NODE
calico-etcd-fxt29                          1/1       Running   0          3h        192.168.0.60    kubemaster
calico-kube-controllers-6cfcf554b9-wbkdg   1/1       Running   0          3h        192.168.0.61    kubeworker1
calico-node-657t5                          2/2       Running   0          3h        192.168.0.60    kubemaster
calico-node-6p6xs                          2/2       Running   1          3h        192.168.0.61    kubeworker1
calico-node-dscx6                          2/2       Running   1          3h        192.168.0.63    kubeworker3
calico-node-hvddc                          2/2       Running   0          3h        192.168.0.62    kubeworker2
etcd-kubemaster                            1/1       Running   2          3h        192.168.0.60    kubemaster
kube-apiserver-kubemaster                  1/1       Running   0          3h        192.168.0.60    kubemaster
kube-controller-manager-kubemaster         1/1       Running   0          3h        192.168.0.60    kubemaster
kube-dns-6f4fd4bdf-6c9kp                   3/3       Running   0          3h        172.31.90.1     kubeworker1
kube-proxy-2cbqv                           1/1       Running   0          3h        192.168.0.61    kubeworker1
kube-proxy-7vd7b                           1/1       Running   0          3h        192.168.0.63    kubeworker3
kube-proxy-hcdbh                           1/1       Running   0          3h        192.168.0.60    kubemaster
kube-proxy-pr9dp                           1/1       Running   0          3h        192.168.0.62    kubeworker2
kube-scheduler-kubemaster                  1/1       Running   0          3h        192.168.0.60    kubemaster
[root@kubemaster kube-cni-calico]#
```

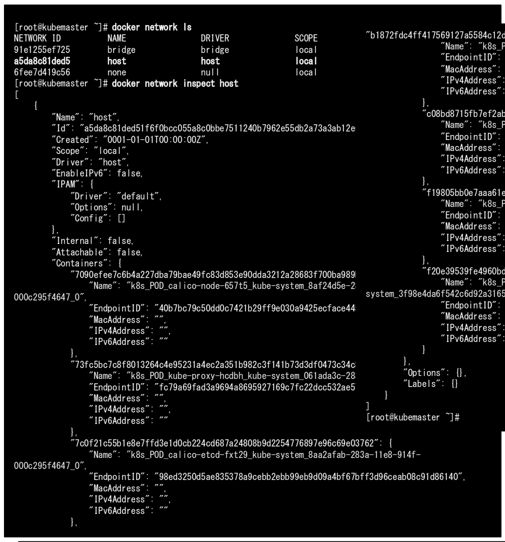


JS Lab

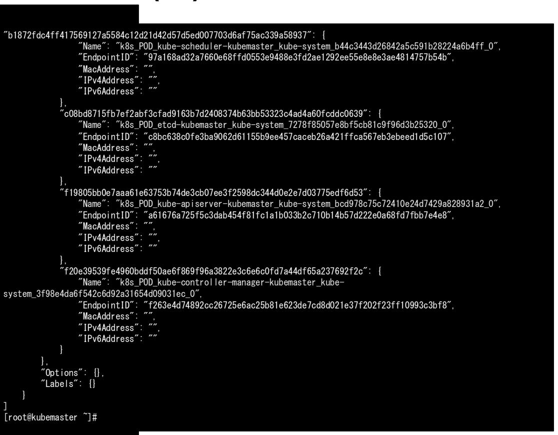
498

별첨

ㅁ K8s를 위한 도커 네트워크 구성(예)



```
[root@kubemaster ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
91e1255ef725        bridge              bridge              local
a5da8c81ded5        host                host                local
6fee7d419c56        none                null                local
[root@kubemaster ~]# docker network inspect host
[
    {
        "Name": "host",
        "Id": "a5da8c81ded51f6f0bcc055a8c0bbe7511240b7962e55db2a73a3ab12e
        "Created": "0001-01-01T00:00:00Z",
        "Scope": "local",
        "Driver": "host",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": []
        },
        "Internal": false,
        "Attachable": false,
        "Containers": {
            "7090efee7c6b4a227dba79bae49fc83d853e90dda3212a28683f700ba989
                "Name": "k8s_POD_calico-node-657t5_kube-system_8af24d5e-2
000c295f4647_0",
                "EndpointID": "40b7bc79c50dd0c7421b29ff9e030a9425ecface44
                "MacAddress": "",
                "IPv4Address": "",
                "IPv6Address": ""
            },
            "73fc5bc7c8f8013264c4e95231a4ec2a351b982c3f141b73d3df0473c34c
                "Name": "k8s_POD_kube-proxy-hcdbh_kube-system_061ada3c-28
                "EndpointID": "fc79a69fad3a9694a8695927169c7fc22dcc532ae5
                "MacAddress": "",
                "IPv4Address": "",
                "IPv6Address": ""
            },
            "7c0f21c55b1e8e7ffd3e1d0cb224cd687a24808b9d2254776897e96c69e03762": {
                "Name": "k8s_POD_calico-etcd-fxt29_kube-system_8aa2afab-283a-11e8-914f-
000c295f4647_0",
                "EndpointID": "98ed3250d5ae835378a9cebb2ebb99eb9d09a4bf67bff3d96ceab08c91d86140",
                "MacAddress": "",
                "IPv4Address": "",
                "IPv6Address": ""
            },
```



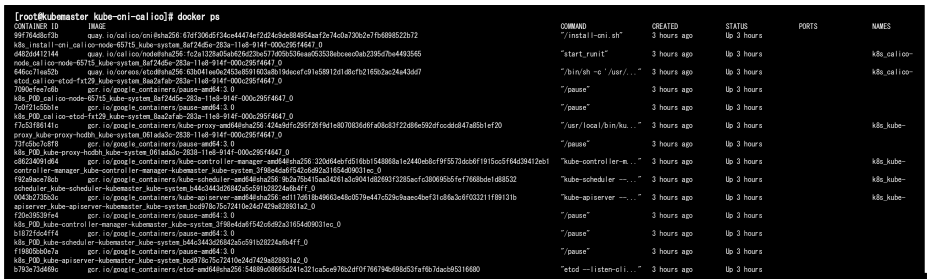
```
        "b1872fdc4ff417569127a5584c12d21d42d57d5ed007703d6af75ac339a58937": {
                "Name": "k8s_POD_kube-scheduler-kubemaster_kube-system_b44c3443d26842a5c591b28224a6b4ff_0",
                "EndpointID": "97a168ad32a7660e68ffd0553e9488e3fd2ae1292ee55e8e8e3ae4814757b54b",
                "MacAddress": "",
                "IPv4Address": "",
                "IPv6Address": ""
            },
            "c08bd8715fb7ef2abf3cfad9163b7d2408374b63bb53323c4ad4a60fcddc0639": {
                "Name": "k8s_POD_etcd-kubemaster_kube-system_7278f85057e8bf5cb81c9f96d3b25320_0",
                "EndpointID": "c8bc638c0fe3ba9062d61155b9ee457caceb26a421ffca567eb3ebeed1d5c107",
                "MacAddress": "",
                "IPv4Address": "",
                "IPv6Address": ""
            },
            "f19805bb0e7aaa61e63753b74de3cb07ee3f2598dc344d0e2e7d03775edf6d53": {
                "Name": "k8s_POD_kube-apiserver-kubemaster_kube-system_bcd978c75c72410e24d7429a828931a2_0",
                "EndpointID": "a61676a725f5c3dab454f81fc1a1b033b2c710b14b57d222e0a68fd7fbb7e4e8",
                "MacAddress": "",
                "IPv4Address": "",
                "IPv6Address": ""
            },
            "f20e39539fe4960bddf50ae6f869f96a3822e3c6e6c0fd7a44df65a237692f2c": {
                "Name": "k8s_POD_kube-controller-manager-kubemaster_kube-
system_3f98e4da6f542c6d92a31654d09031ec_0",
                "EndpointID": "f263e4d74892cc26725e6ac25b81e623de7cd8d021e37f202f23ff10993c3bf8",
                "MacAddress": "",
                "IPv4Address": "",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
[root@kubemaster ~]#
```

JS Lab

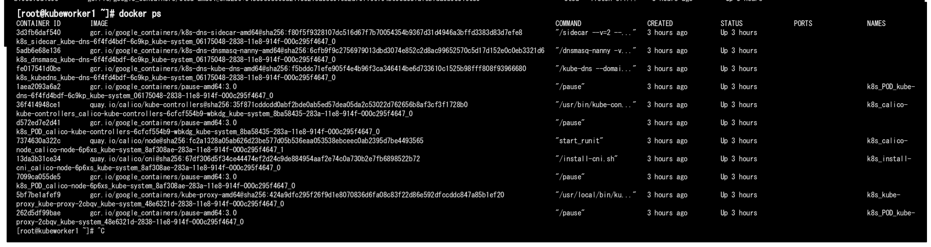
499

별첨

ㅁ K8s @ 도커: 호스트에서 K8s Master/Worker 구성 확인



```
[root@kubemaster kube-cni-calico]# docker ps
CONTAINER ID        IMAGE                                                                                                       COMMAND                  CREATED             STATUS              PORTS               NAMES
99f764d8cf3b        quay.io/calico/cni@sha256:67df306d5f34ce44474ef2d24c9de884954aaf2e74c0a730b2e7fb6898522b72                 "/install-cni.sh"        3 hours ago         Up 3 hours
k8s_install-cni_calico-node-657t5_kube-system_8af24d5e-283a-11e8-914f-000c295f4647_0
d482dd412144        quay.io/calico/node@sha256:fc2a1328a05ab626d23be577d05b536eaa053538ebceec0ab2395d7be4493565                "start_runit"            3 hours ago         Up 3 hours                              k8s_calico-
node_calico-node-657t5_kube-system_8af24d5e-283a-11e8-914f-000c295f4647_0
646cc71ea52b        quay.io/coreos/etcd@sha256:63b041ee0e2453e8591603a8b19decefc91e58912d1d8cfb2165b2ac24a43dd7                "/bin/sh -c '/usr/..."   3 hours ago         Up 3 hours                              k8s_calico-
etcd_calico-etcd-fxt29_kube-system_8aa2afab-283a-11e8-914f-000c295f4647_0
7090efee7c6b        gcr.io/google_containers/pause-amd64:3.0                                                                    "/pause"                 3 hours ago         Up 3 hours
k8s_POD_calico-node-657t5_kube-system_8af24d5e-283a-11e8-914f-000c295f4647_0
7c0f21c55b1e        gcr.io/google_containers/pause-amd64:3.0                                                                    "/pause"                 3 hours ago         Up 3 hours
k8s_POD_calico-etcd-fxt29_kube-system_8aa2afab-283a-11e8-914f-000c295f4647_0
f7c53f86141c        gcr.io/google_containers/kube-proxy-amd64@sha256:424a9dfc295f26f9d1e8070836d6fa08c83f22d86e592dfccddc847a85b1ef20   "/usr/local/bin/ku..."   3 hours ago         Up 3 hours                              k8s_kube-
proxy_kube-proxy-hcdbh_kube-system_061ada3c-2838-11e8-914f-000c295f4647_0
73fc5bc7c8f8        gcr.io/google_containers/pause-amd64:3.0                                                                    "/pause"                 3 hours ago         Up 3 hours
k8s_POD_kube-proxy-hcdbh_kube-system_061ada3c-2838-11e8-914f-000c295f4647_0
c86234091d64        gcr.io/google_containers/kube-controller-manager-amd64@sha256:320d64ebfd516bb1548868a1e2440eb8cf9f5573dcb6f1915cc5f64d39412eb1   "kube-controller-m..."   3 hours ago         Up 3 hours                              k8s_kube-
controller-manager_kube-controller-manager-kubemaster_kube-system_3f98e4da6f542c6d92a31654d09031ec_0
f92a9ace78cb        gcr.io/google_containers/kube-scheduler-amd64@sha256:9b2a75b415aa34261a3c9041d82693f3285acfc380695b5fef7668bde1d88532   "kube-scheduler --..."   3 hours ago         Up 3 hours                              k8s_kube-
scheduler_kube-scheduler-kubemaster_kube-system_b44c3443d26842a5c591b28224a6b4ff_0
0043b2735b3c        gcr.io/google_containers/kube-apiserver-amd64@sha256:ed117d618b49663e48c0579e447c529c9aaec4bef31c86a3c6f033211f89131b   "kube-apiserver --..."   3 hours ago         Up 3 hours                              k8s_kube-
apiserver_kube-apiserver-kubemaster_kube-system_bcd978c75c72410e24d7429a828931a2_0
f20e39539fe4        gcr.io/google_containers/pause-amd64:3.0                                                                    "/pause"                 3 hours ago         Up 3 hours
k8s_POD_kube-controller-manager-kubemaster_kube-system_3f98e4da6f542c6d92a31654d09031ec_0
b1872fdc4ff4        gcr.io/google_containers/pause-amd64:3.0                                                                    "/pause"                 3 hours ago         Up 3 hours
k8s_POD_kube-scheduler-kubemaster_kube-system_b44c3443d26842a5c591b28224a6b4ff_0
f19805bb0e7a        gcr.io/google_containers/pause-amd64:3.0                                                                    "/pause"                 3 hours ago         Up 3 hours
k8s_POD_kube-apiserver-kubemaster_kube-system_bcd978c75c72410e24d7429a828931a2_0
b793e73d469c        gcr.io/google_containers/etcd-amd64@sha256:54889c08665d241e321ca5ce976b2df0f766794b698d53faf6b7dacb95316680   "etcd --listen-cli..."   3 hours ago         Up 3 hours
```



```
[root@kubeworker1 ~]# docker ps
CONTAINER ID        IMAGE                                                                                                       COMMAND                  CREATED             STATUS              PORTS               NAMES
3d3fb6daf540        gcr.io/google_containers/k8s-dns-sidecar-amd64@sha256:f80f5f9328107dc516d67f7b70054354b9367d31d4946a3bffd3383d83d7efe8   "/sidecar --v=2 --..."   3 hours ago         Up 3 hours
k8s_sidecar_kube-dns-6f4fd4bdf-6c9kp_kube-system_06175048-2838-11e8-914f-000c295f4647_0
5adb6e68e136        gcr.io/google_containers/k8s-dns-dnsmasq-nanny-amd64@sha256:6cfb9f9c2756979013dbd3074e852c2d8ac99652570c5d17d152e0c0eb3321d6   "/dnsmasq-nanny -v..."   3 hours ago         Up 3 hours
k8s_dnsmasq_kube-dns-6f4fd4bdf-6c9kp_kube-system_06175048-2838-11e8-914f-000c295f4647_0
fe017541d0be        gcr.io/google_containers/k8s-dns-kube-dns-amd64@sha256:f5bddc71efe905f4e4b96f3ca346414be6d733610c1525b98fff808f93966680   "/kube-dns --domai..."   3 hours ago         Up 3 hours
k8s_kubedns_kube-dns-6f4fd4bdf-6c9kp_kube-system_06175048-2838-11e8-914f-000c295f4647_0
1aea2093a6a2        gcr.io/google_containers/pause-amd64:3.0                                                                    "/pause"                 3 hours ago         Up 3 hours                              k8s_POD_kube-
dns-6f4fd4bdf-6c9kp_kube-system_06175048-2838-11e8-914f-000c295f4647_0
36f414948ce1        quay.io/calico/kube-controllers@sha256:35f871cddcdd0abf2bde0ab5ed57dea05da2c53022d762656b8af3cf3f1728b0   "/usr/bin/kube-con..."   3 hours ago         Up 3 hours                              k8s_calico-
kube-controllers_calico-kube-controllers-6cfcf554b9-wbkdg_kube-system_8ba58435-283a-11e8-914f-000c295f4647_0
d572ed7e2d41        gcr.io/google_containers/pause-amd64:3.0                                                                    "/pause"                 3 hours ago         Up 3 hours
k8s_POD_calico-kube-controllers-6cfcf554b9-wbkdg_kube-system_8ba58435-283a-11e8-914f-000c295f4647_0
7374630a322c        quay.io/calico/node@sha256:fc2a1328a05ab626d23be577d05b536eaa053538ebceec0ab2395d7be4493565                "start_runit"            3 hours ago         Up 3 hours                              k8s_calico-
node_calico-node-6p6xs_kube-system_8af308ae-283a-11e8-914f-000c295f4647_1
13da3b31ce34        quay.io/calico/cni@sha256:67df306d5f34ce44474ef2d24c9de884954aaf2e74c0a730b2e7fb6898522b72                 "/install-cni.sh"        3 hours ago         Up 3 hours                              k8s_install-
cni_calico-node-6p6xs_kube-system_8af308ae-283a-11e8-914f-000c295f4647_0
7099ca055de5        gcr.io/google_containers/pause-amd64:3.0                                                                    "/pause"                 3 hours ago         Up 3 hours
k8s_POD_calico-node-6p6xs_kube-system_8af308ae-283a-11e8-914f-000c295f4647_0
5bf7be1afef9        gcr.io/google_containers/kube-proxy-amd64@sha256:424a9dfc295f26f9d1e8070836d6fa08c83f22d86e592dfccddc847a85b1ef20   "/usr/local/bin/ku..."   3 hours ago         Up 3 hours                              k8s_kube-
proxy_kube-proxy-2cbqv_kube-system_48e6321d-2838-11e8-914f-000c295f4647_0
262d5df99bae        gcr.io/google_containers/pause-amd64:3.0                                                                    "/pause"                 3 hours ago         Up 3 hours                              k8s_POD_kube-
proxy-2cbqv_kube-system_48e6321d-2838-11e8-914f-000c295f4647_0
[root@kubeworker1 ~]# ^C
```

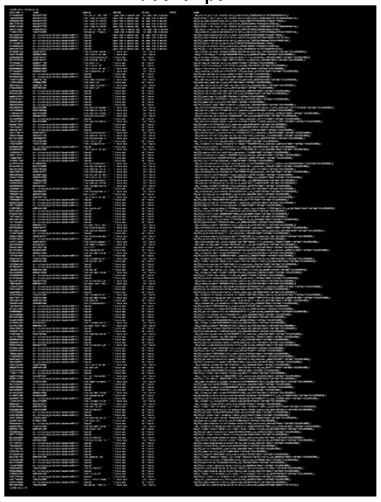
JS Lab

500

별첨

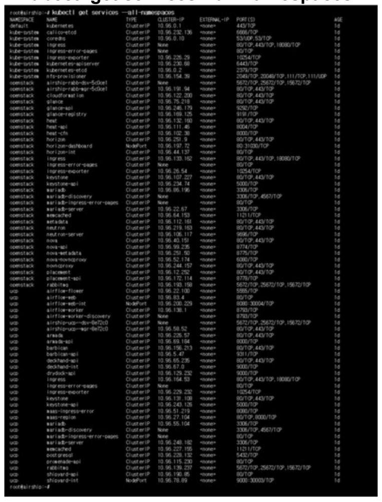
- Docker and Kubernetes for Airship

docker ps



[illegible]

kubectl get services --all-namespaces



```
root@airship:~# kubectl get services --all-namespaces
NAMESPACE     NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                                AGE
default       kubernetes                    ClusterIP   10.96.0.1       <none>        443/TCP                                1d
kube-system   calico-etcd                   ClusterIP   10.96.232.136   <none>        6666/TCP                               1d
kube-system   coredns                       ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP                          1d
kube-system   ingress                       ClusterIP   None            <none>        80/TCP,443/TCP,18080/TCP               1d
kube-system   ingress-error-pages           ClusterIP   None            <none>        80/TCP                                 1d
kube-system   ingress-exporter              ClusterIP   10.96.226.29    <none>        10254/TCP                              1d
kube-system   kubernetes-apiserver          ClusterIP   10.96.230.68    <none>        6443/TCP                               1d
kube-system   kubernetes-etcd               ClusterIP   10.96.0.2       <none>        2379/TCP                               1d
kube-system   nfs-provisioner               ClusterIP   10.96.154.39    <none>        2049/TCP,20048/TCP,111/TCP,111/UDP     1d
openstack     airship-rabb-dsv-5c0ce1       ClusterIP   None            <none>        5672/TCP,25672/TCP,15672/TCP           1d
openstack     airship-rabb-mgr-5c0ce1       ClusterIP   10.96.191.94    <none>        80/TCP,443/TCP                         1d
openstack     cloudformation                ClusterIP   10.96.122.200   <none>        80/TCP,443/TCP                         1d
openstack     glance                        ClusterIP   10.96.75.218    <none>        80/TCP,443/TCP                         1d
openstack     glance-api                    ClusterIP   10.96.246.179   <none>        9292/TCP                               1d
openstack     glance-registry               ClusterIP   10.96.169.125   <none>        9191/TCP                               1d
openstack     heat                          ClusterIP   10.96.132.160   <none>        80/TCP,443/TCP                         1d
openstack     heat-api                      ClusterIP   10.96.111.46    <none>        8004/TCP                               1d
openstack     heat-cfn                      ClusterIP   10.96.102.38    <none>        8000/TCP                               1d
openstack     horizon                       ClusterIP   10.96.251.9     <none>        80/TCP,443/TCP                         1d
openstack     horizon-dashboard             NodePort    10.96.197.72    <none>        80:31000/TCP                           1d
openstack     horizon-int                   ClusterIP   10.96.44.137    <none>        80/TCP                                 1d
openstack     ingress                       ClusterIP   10.96.133.162   <none>        80/TCP,443/TCP,18080/TCP               1d
openstack     ingress-error-pages           ClusterIP   None            <none>        80/TCP                                 1d
openstack     ingress-exporter              ClusterIP   10.96.26.54     <none>        10254/TCP                              1d
openstack     keystone                      ClusterIP   10.96.107.227   <none>        80/TCP,443/TCP                         1d
openstack     keystone-api                  ClusterIP   10.96.234.74    <none>        5000/TCP                               1d
openstack     mariadb                       ClusterIP   10.96.86.196    <none>        3306/TCP                               1d
openstack     mariadb-discovery             ClusterIP   None            <none>        3306/TCP,4567/TCP                      1d
openstack     mariadb-ingress-error-pages   ClusterIP   None            <none>        80/TCP                                 1d
openstack     mariadb-server                ClusterIP   10.96.22.67     <none>        3306/TCP                               1d
openstack     memcached                     ClusterIP   10.96.64.153    <none>        11211/TCP                              1d
openstack     metadata                      ClusterIP   10.96.112.161   <none>        80/TCP,443/TCP                         1d
openstack     neutron                       ClusterIP   10.96.219.163   <none>        80/TCP,443/TCP                         1d
openstack     neutron-server                ClusterIP   10.96.106.117   <none>        9696/TCP                               1d
openstack     nova                          ClusterIP   10.96.40.151    <none>        80/TCP,443/TCP                         1d
openstack     nova-api                      ClusterIP   10.96.99.235    <none>        8774/TCP                               1d
openstack     nova-metadata                 ClusterIP   10.96.251.50    <none>        8775/TCP                               1d
openstack     nova-novncproxy               ClusterIP   10.96.52.174    <none>        6080/TCP                               1d
openstack     novncproxy                    ClusterIP   10.96.244.157   <none>        80/TCP,443/TCP                         1d
openstack     placement                     ClusterIP   10.96.12.252    <none>        80/TCP,443/TCP                         1d
openstack     placement-api                 ClusterIP   10.96.172.114   <none>        8778/TCP                               1d
openstack     rabbitmq                      ClusterIP   10.96.193.158   <none>        5672/TCP,25672/TCP,15672/TCP           1d
ucp           airflow-flower                ClusterIP   10.96.22.100    <none>        5555/TCP                               1d
ucp           airflow-web                   ClusterIP   10.96.83.4      <none>        80/TCP                                 1d
ucp           airflow-web-int               NodePort    10.96.200.229   <none>        8080:30004/TCP                         1d
ucp           airflow-worker                ClusterIP   10.96.138.1     <none>        8793/TCP                               1d
ucp           airflow-worker-discovery      ClusterIP   None            <none>        8793/TCP                               1d
ucp           airship-ucp--dsv-8e72c0       ClusterIP   None            <none>        5672/TCP,25672/TCP,15672/TCP           1d
ucp           airship-ucp--mgr-8e72c0       ClusterIP   10.96.58.52     <none>        80/TCP,443/TCP                         1d
ucp           armada                        ClusterIP   10.96.226.57    <none>        80/TCP,443/TCP                         1d
ucp           armada-api                    ClusterIP   10.96.69.184    <none>        8000/TCP                               1d
ucp           barbican                      ClusterIP   10.96.156.213   <none>        80/TCP,443/TCP                         1d
ucp           barbican-api                  ClusterIP   10.96.5.47      <none>        9311/TCP                               1d
ucp           deckhand-api                  ClusterIP   10.96.65.235    <none>        80/TCP,443/TCP                         1d
ucp           deckhand-int                  ClusterIP   10.96.67.0      <none>        9000/TCP                               1d
ucp           drydock-api                   ClusterIP   10.96.129.232   <none>        9000/TCP                               1d
ucp           ingress                       ClusterIP   10.96.164.53    <none>        80/TCP,443/TCP,18080/TCP               1d
ucp           ingress-error-pages           ClusterIP   None            <none>        80/TCP                                 1d
ucp           ingress-exporter              ClusterIP   10.96.229.232   <none>        10254/TCP                              1d
ucp           keystone                      ClusterIP   10.96.131.108   <none>        80/TCP,443/TCP                         1d
ucp           keystone-api                  ClusterIP   10.96.243.126   <none>        5000/TCP                               1d
ucp           maas-ingress-error            ClusterIP   10.96.51.219    <none>        8080/TCP                               1d
ucp           maas-region                   ClusterIP   10.96.27.104    <none>        80/TCP,8000/TCP                        1d
ucp           mariadb                       ClusterIP   10.96.55.104    <none>        3306/TCP                               1d
ucp           mariadb-discovery             ClusterIP   None            <none>        3306/TCP,4567/TCP                      1d
ucp           mariadb-ingress-error-pages   ClusterIP   None            <none>        80/TCP                                 1d
ucp           mariadb-server                ClusterIP   10.96.248.182   <none>        3306/TCP                               1d
ucp           memcached                     ClusterIP   10.96.227.155   <none>        11211/TCP                              1d
ucp           postgresql                    ClusterIP   10.96.228.132   <none>        5432/TCP                               1d
ucp           promenade-api                 ClusterIP   10.96.115.230   <none>        80/TCP                                 1d
ucp           rabbitmq                      ClusterIP   10.96.139.237   <none>        5672/TCP,25672/TCP,15672/TCP           1d
ucp           shipyard-api                  ClusterIP   10.96.190.85    <none>        80/TCP                                 1d
ucp           shipyard-int                  NodePort    10.96.78.89     <none>        9000:30003/TCP                         1d
root@airship:~#
```

james@jslab.kr

JS Lab

501

별첨

- How to Be Successful in the Cloud
  - With cloud computing's pay-as-you-go and software-defined everything models, startups now have a very low barrier to take an enterprise assignment. And, with open source tools and ecosystems built around them, startups can innovate and adapt very fast.

james@jslab.kr

JS Lab

502

별첨

- How to Be Successful in the Cloud
  - Any company, be it small or big, has to innovate fast, listen to customer feedback, and then iterate over it. To do that, companies need to bring in DevOps practices and allow small teams to manage the entire lifecycle of their products, from Development to Support. Technologies like Container as a Service and Continuous Integration and Deployment allow small teams to have access of Development, QA and Deployment environments. This allows individual teams that work in an enterprise to work like startups, which is good for business.

별첨

- How to Be Successful in the Cloud
  - Nowadays, IT is becoming part of the business process. Due to the emergence of cloud technologies, DevOps, microservices architecture, etc., each business is expected to respond to customer feedback sooner rather than later.

별첨

- How to Be Successful in the Cloud
  - Container technologies like Docker and their ecosystems are helping us standardize and streamline the way we package and deploy code. If we want to adopt the technologies we discussed in this course, we realize that Developers, QA and OPs cannot work in silos. To be efficient and successful, they have to work together and need to have basic knowledge about each others' workflows. This is very different from the traditional IT approach and has an initial steep learning curve. The bigger the company, the bigger the challenges.

505

별첨

- How to Be Successful in the Cloud
- Developing The Necessary Skills Set
  - Different cloud offerings (IaaS, PaaS, SaaS) and cloud models (public, private, and hybrid)
  - Container technologies like Docker, Kubernetes, and their ecosystem
  - DevOps
  - Continuous Integration and Continuous Deployment
  - Software Defined Networking and Storage
  - Debugging, Logging, and Monitoring cloud applications.

506

별첨

- How to Be Successful in the Cloud
- Challenges
  - Once you decide to move to the cloud, you will inevitably face some challenges. Most of them you will encounter for the first time. In this section, we will talk about some of the challenges that you may encounter in your journey to the cloud.

별첨

- How to Be Successful in the Cloud
- Choosing the Right Cloud Provider
  - There are different cloud providers like Amazon AWS, Google Cloud Platform, Microsoft Azure, OpenStack, etc. Each of them provides different services.
  - Similarly, there are different cloud models like public, private, and hybrid. Each of them has their advantages and shortcomings. For example, the hybrid model is useful when you want to keep your data on-premise and serve the request from public clouds.
  - Companies have to spend a significant amount of time to evaluate different cloud providers and models in the beginning, as it affects the overall operations and costs.

별첨

- How to Be Successful in the Cloud
- Choosing the Right Technology Stack
  - To avail fully of the benefits provided by the Cloud, we have to choose the right technology stack as well. For example, should we go for IaaS or PaaS solutions? Should we choose VMs or containers to deploy the applications? Most of these questions have multiple answers. As a company, you need to hire cloud architects who can make the right decision for you.

별첨

- How to Be Successful in the Cloud
- Cloud Cost Management
  - Studies say that, by moving to the cloud, an organization can save a good amount of money as compared to setting up on-premise solutions. One thing we have to be very careful with is to ensure that we know when this is true, and when it is not, for any given use case. Most cloud providers can now predict and manage the cost based on usage, which can help companies keep track of their spending.

별첨

- How to Be Successful in the Cloud
- Security Concerns
  - Security is one of the biggest concerns when moving towards the cloud. Companies worry about privacy, data access, accountability, account control, multi-tenancy, etc. For example, with 100% public cloud deployment, the entire data is managed on the cloud, which is not the preferred way for many companies and may not comply with regulations like FISMA (Federal Information Security Modernization Act) or HIPAA (Health Insurance Portability and Accountability Act). In such cases, companies adopt the Hybrid Cloud approach.

별첨

- How to Be Successful in the Cloud
- Security Concerns
  - Organizations like the Cloud Security Alliance (CSA) “promote the use of best practices for providing security assurance within Cloud Computing, and to provide education on the uses of Cloud Computing to help secure all other forms of computing”.
  - Nowadays, there are many third-party tools available which can do security audits for applications deployed on the cloud.

별첨

- How to Be Successful in the Cloud
- Vendor Lock-In
  - Companies also worry about vendor lock-in when it comes to cloud computing. Cloud providers allow migration from other providers, but that is not an ideal solution. With containers becoming mainstream and using innovative solutions like Kubernetes, Docker Enterprise Edition, etc. on top of them, we can deploy the applications across datacenters on top of different cloud providers. The use of containers definitely addresses part of the vendor lock-in problem.

별첨

- How to Be Successful in the Cloud
- Resistance from Existing Employees
  - This is one of the biggest challenges companies are facing. With cloud technologies, the existing workflow is changing dramatically, and that is the need of the hour: to keep up with the business demands. Oftentimes, many IT employees resist learning new things and changing their ways, which is not good for the business. To avoid or minimize this problem, the top management of a company has to guide its employees through the change process of learning new technologies. On the other hand, as an employee, you should also be committed to continuous education and lifelong learning, as learning new things will help you stay relevant in a competitive and ever-changing industry.

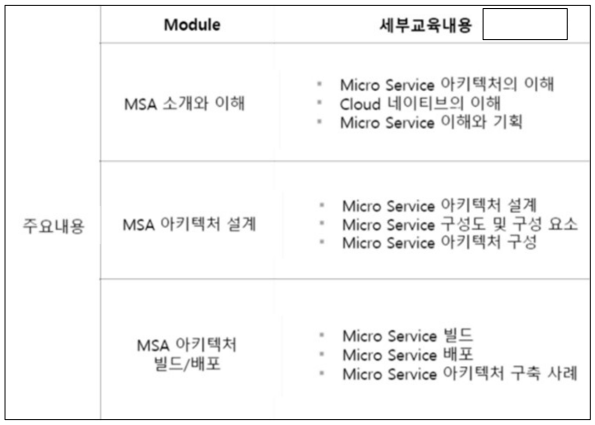
1장. 개요
2장. 가상화 (Virtualization)
3장. 클라우드 서비스 (Cloud Services)
4장. 컨테이너 (Containers)
5장. DevOps와 CI/CD
6장. 도구(Tools)
7장. 서비스 메시 (Service Mesh)
8장. 서버리스 (Serverless Computing)
9장. 관리 (Management)
10장. 보안 (Security)
11장. 디자인 패턴 (Design Pattern)
12장. Use Case
별첨: Docker, K8s, How to be success in cloud

❖ **실습교재 (별도)**

515

516

❖본 과정에서는 **MSA**를 이용해 대용량, 대규모 개발에 적합한 경량화 및 변형된 아키텍처로 설계/구축법을 학습합니다.
❖대상: 소프트웨어 아키텍트, 소프트웨어 관리자



	Module	세부교육내용
주요내용	MSA 소개와 이해	• Micro Service 아키텍처의 이해 • Cloud 네이티브의 이해 • Micro Service 이해와 기획
	MSA 아키텍처 설계	• Micro Service 아키텍처 설계 • Micro Service 구성도 및 구성 요소 • Micro Service 아키텍처 구성
	MSA 아키텍처 빌드/배포	• Micro Service 빌드 • Micro Service 배포 • Micro Service 아키텍처 구축 사례

JS Lab

517