Cloud Platform Operations for Edge Cloud Computing

2019. 7.

안 종 석
james@jslab.kr
JS Lab

1

목차

2

목차

I. Virtualization (가상화)
II. Infrastructure as a Service (IaaS)
III. Platform as a Service (PaaS)
IV. Containers (컨테이너)
1. 개요
2. Micro OS
3. Orchestration (오케스트레이션)
4. Unikernels
5. Microservices (마이크로서비스)
6. Software-Defined Networking (SDN) and Containers
7. Software-Defined Storage (SDS) and Containers
V. DevOps and CI/CD
VI. Tools for Cloud Infrastructure (클라우드 인프라 도구)
1. Configuration Management
2. Build & Release
3. Key-Value Pair Store
4. Image Building
5. Debugging, Logging, and Monitoring for Containerized Applications
VII. Service Mesh (서비스 메쉬)
VIII. Internet of Things (IoT)
IX. Serverless Computing, FaaS (Function as a Service)
X. OpenTracing
XI. How to Be Successful in the Cloud

❖ 실습교재 (별도)

3

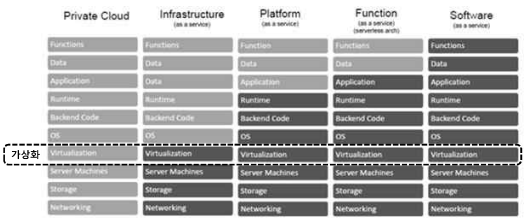
I. Virtualization

- "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".

4

I. Virtualization

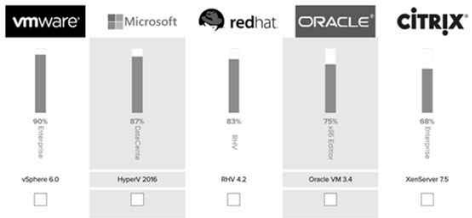
- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)



5

I. 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

6

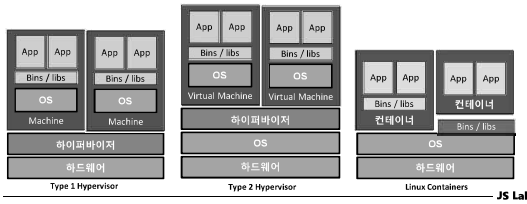
I. Virtualization

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

7

I. Virtualization

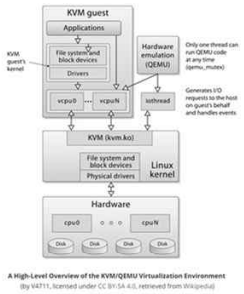
- KVM
- Xen
- VMware
- VirtualBox
- Hyper-V



8

I. Virtualization

- KVM



A High-Level Overview of the KVM/QEMU Virtualization Environment

9

I. 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.

10

I. 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.

11

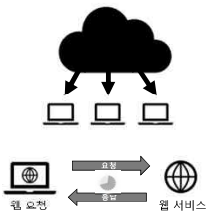
I. Virtualization

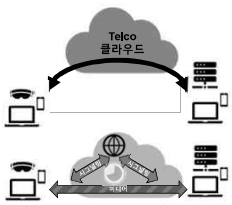
- Vagrant
  - Reproducible environment
  - Management of multiple projects in their restricted environment
  - Sharing the environment with other teammates
  - Keeping the development and deployment environments in sync
  - Running the same VM on different OSes, with a hypervisor like VirtualBox.

12

I. Virtualization

- 5G 코어 인프라의 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



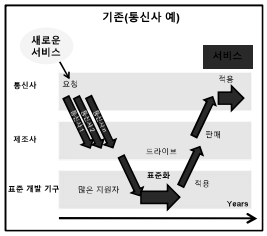


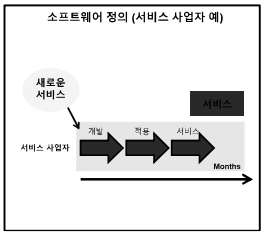
실시간 멀티미디어 애플리케이션
(고화질 스트리밍, 고화질 게임, AR, VR 등을 위한 경로)

13

I. Virtualization

- 통신사와 서비스 사업자의 소프트웨어 정의
- 개발능력 미보유 엔터프라이즈는 서비스 제공 가능 오픈소스나 제조사의 SDx 솔루션 필요





14

I. Virtualization

- Edge vs Central Cloud @ Telco Cloud

	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

15

목차

16

II. Infrastructure as a Service (IaaS)

- Features and Tools
  - t2.nano: 512 MiB of memory, 1 vCPU, 3 CPU Credits/hour, EBS-only, 32-bit or 64-bit platform
  - c4.large: 3.75 GiB of memory, 2 vCPUs, 64-bit platform
  - d2.8xlarge: 244 GiB of memory, 36 vCPUs, 24 x 2000 GB of HDD-based instance storage, 64-bit platform, 10 Gigabit Ethernet.

17

II. Infrastructure as a Service (IaaS)

- 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.

18

II. Infrastructure as a Service (IaaS)

- 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.

19

II. Infrastructure as a Service (IaaS)

- 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.

20

II. Infrastructure as a Service (IaaS)

- 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.

21

II. Infrastructure as a Service (IaaS)

- 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

22

II. Infrastructure as a Service (IaaS)

- 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.

23

II. Infrastructure as a Service (IaaS)

- 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.

24

II. Infrastructure as a Service (IaaS)

- 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.

25

II. Infrastructure as a Service (IaaS)

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

26

II. Infrastructure as a Service (IaaS)

- 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.

27

목차

28

III. Platform as a Service (PaaS)

- 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.

29

III. Platform as a Service (PaaS)

- Cloud Foundry
  - an open source Platform as a Service (PaaS) that provides a choice of clouds, developer frameworks, and application services. It can be deployed on-premise or on IaaS, like AWS, vSphere, or OpenStack. There are many commercial CF cloud providers as well, like IBM Cloud Foundry, SAP Cloud Platform, Pivotal Cloud Foundry, etc.

30

III. Platform as a Service (PaaS)

- Features
  - Application portability
  - Application auto-scaling
  - Centralized platform management
  - Centralized logging
  - Dynamic routing
  - Application health management
  - Role-based application deployment
  - Horizontal and vertical scaling
  - Security
  - Support for different IaaS technologies.

31

III. Platform as a Service (PaaS)

- Cloud Foundry BOSH
  - Cloud Foundry (CF) Application Runtime and Cloud Foundry (CF) Container Runtime. These two platforms are used by Cloud Foundry to run applications and containers.
  - In order to manage the underlying infrastructure for both of them, the Cloud Foundry uses BOSH. BOSH is a cloud-agnostic open source tool for release engineering, deployment, and lifecycle management of complex distributed systems.

32

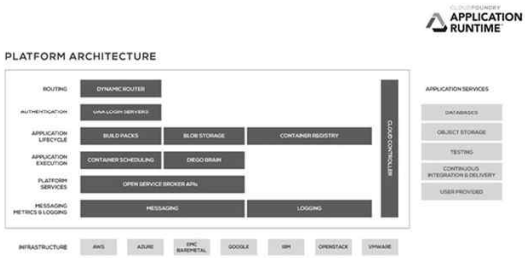
III. Platform as a Service (PaaS)

- Cloud Foundry Platforms: CF Application Runtime
  - CF Application Runtime, previously known as Elastic Runtime, is used by developers to run applications written in any language or framework on the cloud of their choice
  - CF Application Runtime uses buildpacks, which provide the framework and runtime support for the applications.
    - Java
    - Python
    - GO
    - Ruby
    - .Net
    - Node.js
    - PHP.

33

III. Platform as a Service (PaaS)

- The CF Application Runtime Platform Architecture

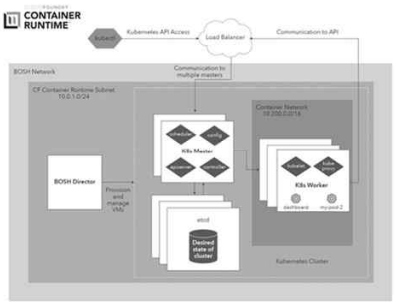


https://www.cloudfoundry.org/application-runtime/

34

III. Platform as a Service (PaaS)

- Cloud Foundry Platforms: CF Container Runtime



https://www.cloudfoundry.org/container-runtime/

35

III. Platform as a Service (PaaS)

- Some of the benefits of using Cloud Foundry are:
  - It is an open source platform, but there are also many commercial Cloud Foundry providers.
  - It offers centralized platform management.
  - It enables horizontal and vertical scaling.
  - It provides infrastructure security.
  - It provides multi-tenant compute efficiency.
  - It offers support for multiple IaaS providers.
  - It supports the full lifecycle: development, testing, and deployment, thus enabling a continuous delivery strategy. It also provides integration with CI/CD tools.
  - It is a simple and flexible solution, supported by an extensive community of developers.
  - It reduces the chance of human errors.
  - It is cost-effective, reducing the overhead for Ops teams.

36

III. Platform as a Service (PaaS)

- OpenShift
  - **OpenShift Online:** With the Online plan, you can deploy your applications on the OpenShift Cluster managed by Red Hat and pay as per your usage. You can deploy up to four services for free.
  - **OpenShift Dedicated:** With the Dedicated plan, you can get your own dedicated OpenShift Cluster, which is managed by Red Hat.
  - **OpenShift Container Platform:** With the Container Platform plan, you can create your own private PaaS on cloud or on-premise.
  - **OpenShift Origin:** All the upstream development on OpenShift happens on GitHub and it is referred to as OpenShift Origin.

37

III. Platform as a Service (PaaS)

- Features
  - **With OpenShift, we can deploy containerized applications. With application images and QuickStart application templates, applications can be deployed with one click.**
  - **As OpenShift uses Kubernetes, we get all the features offered by Kubernetes, like adding or removing nodes at runtime, persistent storage, auto-scaling, etc.**
  - **OpenShift has a framework called Source to Image (S2I), which enables us to create container images from the source code repository to deploy applications easily.**
  - **OpenShift integrates well with Continuous Deployment tools to deploy applications as part of the CI/CD pipeline.**
  - **With CLI, GUI and IDE integration applications can be managed easily.**

38

III. Platform as a Service (PaaS)

- Installing OpenShift
  - **You do not have to worry about the installation if you opt for the Online or Dedicated plans, as the Red Hat engineering or support team will assist you with it. OpenShift can be deployed on clouds like AWS, Azure and Google with just a few clicks.**
  - **Detailed instructions for installing OpenShift are given in the documentation.**

https://docs.openshift.com/enterprise/3.1/install_config/install/prerequisites.html

39

III. Platform as a Service (PaaS)

- Benefits of Using OpenShift (1 of 2)
  - **It is an open source PaaS solution.**
  - **It uses Kubernetes underneath for deployment.**
  - **It can scale applications easily and quickly.**
  - **It provides integration with CI/CD tools.**
  - **It is a simple and flexible solution, supported by an active community of developers.**
  - **It enables developers to be more efficient and productive, allowing them to quickly develop, host, and scale applications in the cloud in a streamlined and standardized manner.**

40

III. Platform as a Service (PaaS)

- Benefits of Using OpenShift (2 of 2)
  - **It enables application portability, meaning that any application created on OpenShift can run on any platform that supports Docker.**
  - **OpenShift users have the choice to deploy their applications on top of different infrastructures (physical or virtual; public, private or hybrid).**
  - **With OpenShift, you can easily build applications with integrated service discovery and persistent storage.**
  - **You can use the web console and the CLI to build and monitor applications.**
  - **OpenShift integrated Docker registry, automatic edge load balancing, cluster logging, and integrated metrics.**

41

III. Platform as a Service (PaaS)

- Heroku is a fully-managed container-based cloud platform, with integrated data services and a strong ecosystem.
  - Node.js
  - Ruby
  - Python
  - Go
  - PHP
  - Clojure
  - Scala
  - Java,

42

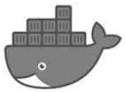
목차

43

IV. Containers 1. 개요

- 용어 정의
- 도커(Docker) 란?

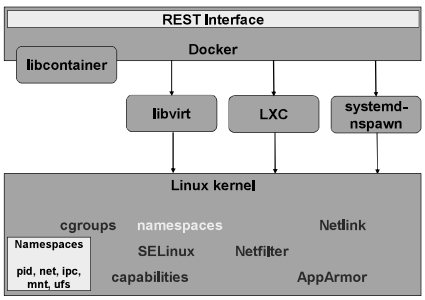
"Company"
"Product"
"Platform"
"CLI Tool"
"Computer Program"



44

IV. Containers 1. 개요

- How it works?



45

IV. Containers 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

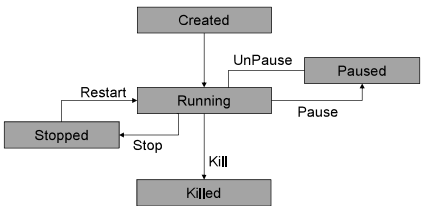
Cgroup: 2008년부터 리눅스 커널에서 사용 했으며 CPU, 메모리, 디스크 I/O,네트워크 등의 자원을 분리하고 제한함

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

46

IV. Containers 1. 개요

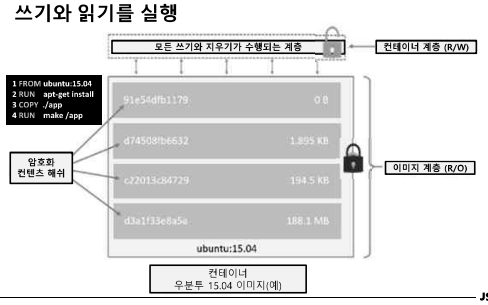
- **도커 컨테이너 라이프 사이클:** Build한 이미지(Image)는 컨테이너 실행 시 읽기 전용으로 사용하며 컨테이너에서 생성한 계층에서 쓰기와 읽기를 실행



47

IV. Containers 1. 개요

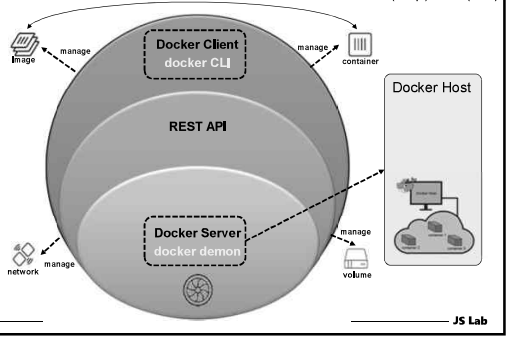
- **이미지 생성: Build한 이미지(Image)는 컨테이너 실행 시 읽기 전용으로 사용하며 컨테이너에서 생성한 계층에서 쓰기와 읽기를 실행**



48

IV. Containers 1. 개요

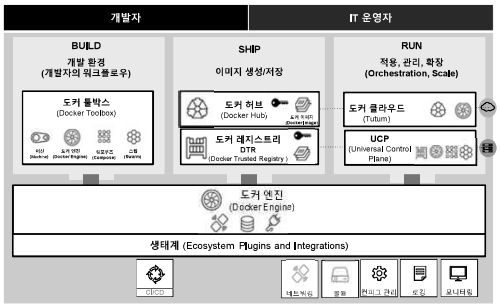
- **도커 엔진:** 물리/가상 호스트에서 컨테이너를 생성/탑재(ship)/실행(run)



49

IV. Containers 1. 개요

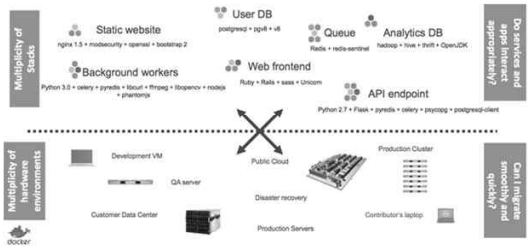
- **워크 플로우(Workflow)**



50

IV. Containers 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.**

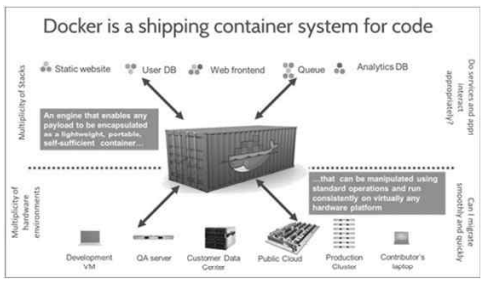


Running an Application (by Docker, Inc.; retrieved from docker.com)

51

IV. Containers 1. 개요

- **Our end goal is to run an application.**



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

52

IV. Containers 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.**

53

IV. Containers 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**

54

IV. Containers 1. 개요

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

55

IV. Containers 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.

56

IV. Containers 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.

57

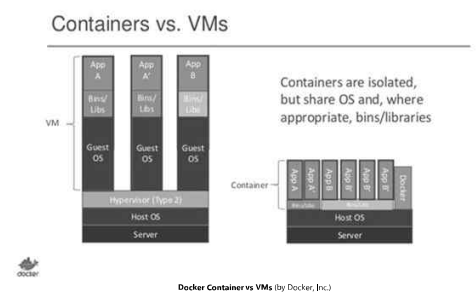
IV. Containers 1. 개요

- 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.

58

IV. Containers 1. 개요

- 가상머신과 컨테이너

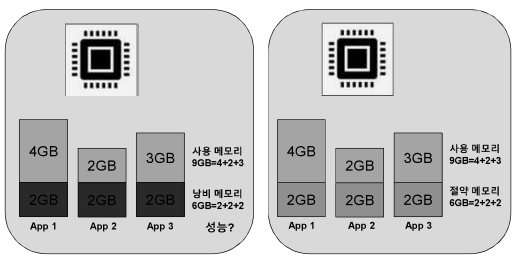


Docker Container vs VMs (by Docker, Inc.)

59

IV. Containers 1. 개요

- 가상머신과 컨테이너 메모리 사용



60

IV. Containers 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
함뚝	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 (동일HW)	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 성태계 연동	Yes	Yes(저성능)	Yes(저성능)	N/A	N/A	N/A

61

IV. Containers 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.

62

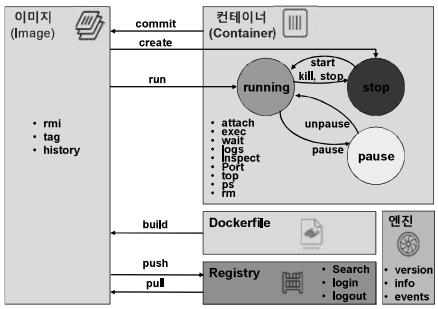
IV. Containers 1. 개요

- 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>

63

IV. Containers 1. 개요

- 도커 명령어 다이어그램



64

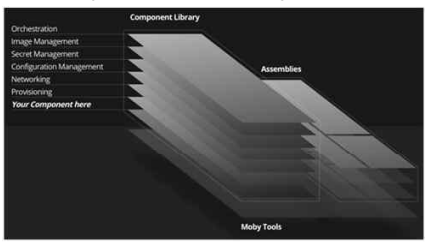
IV. Containers 1. 개요

- 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.

65

IV. Containers 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.

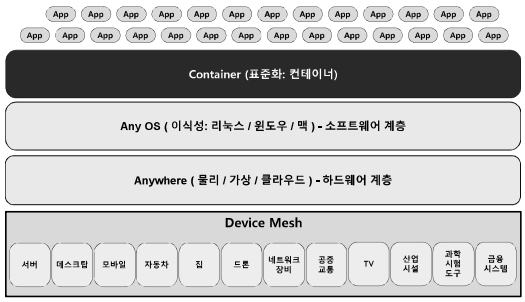


Project Moby (retrieved from https://mobyproject.org/)

66

IV. Containers 1. 개요

□ 요약



67

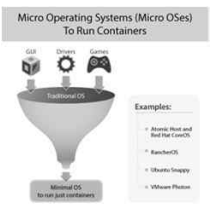
목차

68

IV. Containers 2. Micro OS

□ Micro OSes for containers

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



69

IV. Containers 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.

70

IV. Containers 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

71

IV. Containers 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.

72

IV. Containers 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.

73

IV. Containers 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.

74

IV. Containers 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.

75

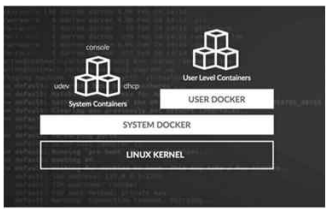
IV. Containers 2. Micro OS

- 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.

76

IV. Containers 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)

77

IV. Containers 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.

78

목차

79

IV. Containers 3. Orchestration

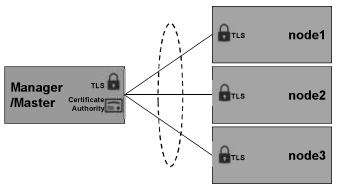
□ 클라우드 관리/오케스트레이션/인프라 구성 관리 도구 비교

	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, IBM, vmware, MORPHEUS, SCALR	OPENSHIFT, RANCHER, CLOUDFOUNDRY, kubernetes	TERRAFORM, CLOUDIFY, ARIA, OASIS TOSCA

80

IV. Containers 3. Orchestration

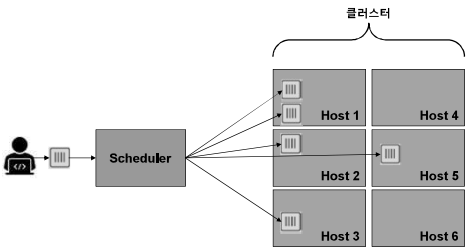
□ Overlay Network



81

IV. Containers 3. Orchestration

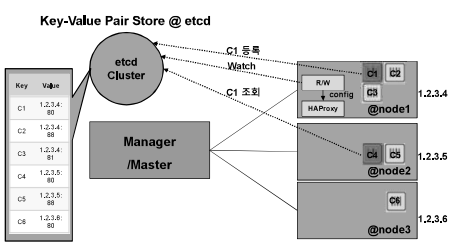
□ Distributed system scheduler in action



82

IV. Containers 3. Orchestration

□ etcd service discovery setup



83

IV. Containers 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?

84

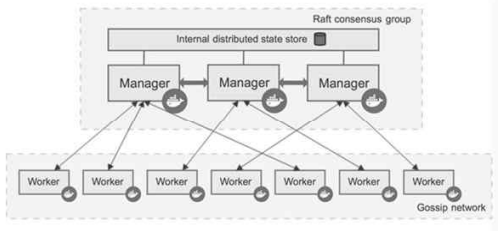
IV. Containers 3. Orchestration

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

85

IV. Containers 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.



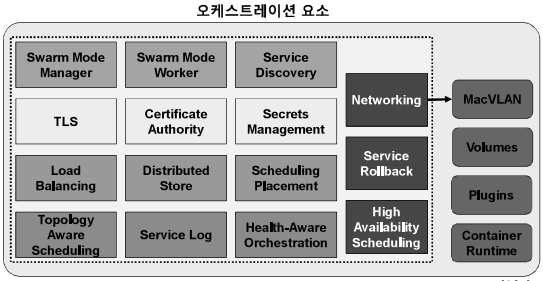
Swarm Cluster Components (by Docker, Inc., retrieved from docker.com)

86

IV. Containers 3. Orchestration

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

오케스트레이션 요소



87

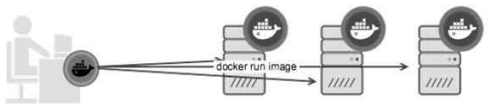
IV. Containers 3. Orchestration

- 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.

88

IV. Containers 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)

89

IV. Containers 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:
```


90

IV. Containers 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.

91

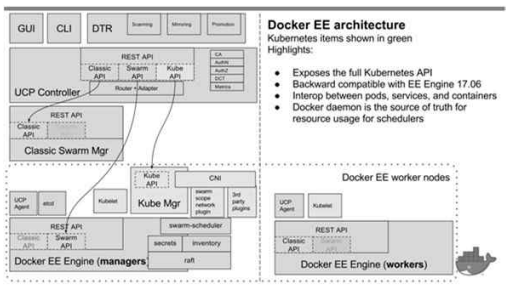
IV. Containers 3. Orchestration

- 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.

92

IV. Containers 3. Orchestration

- Docker EE Architecture



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

93

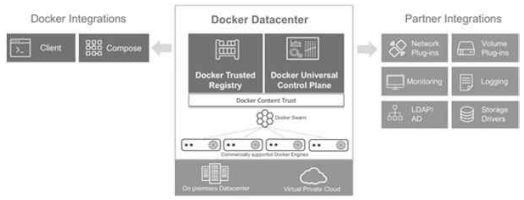
IV. Containers 3. Orchestration

- 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).

94

IV. Containers 3. Orchestration

- Docker Datacenter

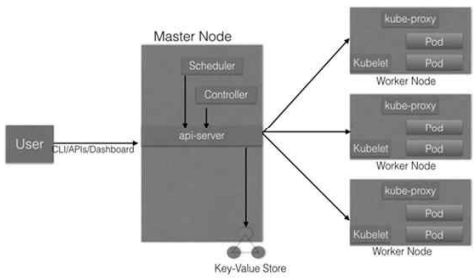


Docker Datacenter (By Docker, Inc., retrieved from docker.com)

95

IV. Containers 3. Orchestration

- The Kubernetes Architecture



96

IV. Containers — 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.

97

IV. Containers — 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

98

IV. Containers — 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
```


99

IV. Containers — 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
```


100

IV. Containers — 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.**

101

IV. Containers — 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.**

102

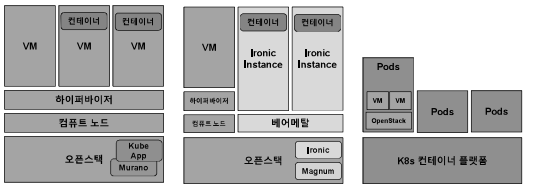
IV. Containers 3. Orchestration

- 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

103

IV. Containers 3. Orchestration

- K8s 적용 클라우드 인프라 비교
  - VM 기반 컨테이너 아키텍처: Pet 기반 운영 가능
  - VM 기반 베어메탈 컨테이너: 베어메탈 지원
  - 오픈스택/베어메탈 컨테이너 혼용: 레거시 VM 필요시 오픈스택 구동

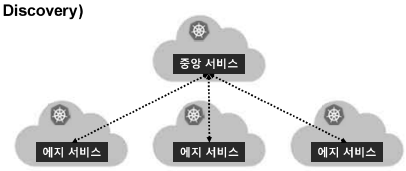


VM 기반 컨테이너 아키텍처 VM 기반 베어메탈 컨테이너 오픈스택과 베어메탈 컨테이너 혼용

104

IV. Containers 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 Cloud는 기기 관리 필요 (PXE, DHCP, IPMI, TFTP, Discovery)



105

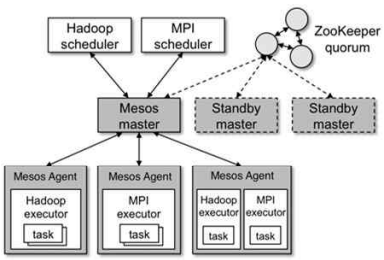
IV. Containers 3. Orchestration

- 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.

106

IV. Containers 3. Orchestration

- Apache Mesos



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

107

IV. Containers 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.

108

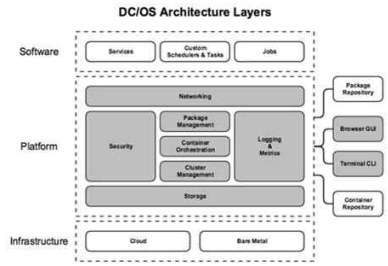
IV. Containers 3. Orchestration

- 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.

109

IV. Containers 3. Orchestration

- DC/OS Architecture Layers



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

110

IV. Containers 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

111

IV. Containers 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.

112

IV. Containers 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.

113

IV. Containers 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

114

IV. Containers — 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-east-1", "us-east-2"]

  # 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.

115

IV. Containers — 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.

116

IV. Containers — 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.

117

IV. Containers — 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

118

IV. Containers — 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

119

IV. Containers — 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

120

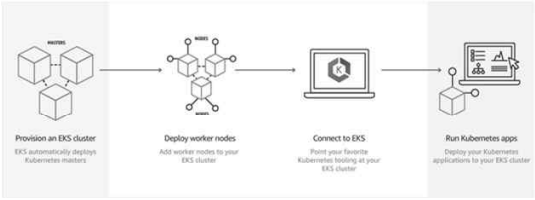
IV. Containers 3. Orchestration

- 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.

121

IV. Containers 3. Orchestration

- Amazon Elastic Container Service for Kubernetes (EKS)



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

122

IV. Containers 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.

123

IV. Containers 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

124

IV. Containers 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

125

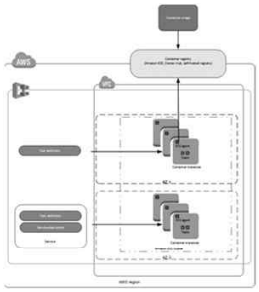
IV. Containers 3. Orchestration

- **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.

126

IV. Containers — 3. Orchestration

- EC2 Launch Type



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

127

IV. Containers — 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.

128

IV. Containers — 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"
}
```


129

IV. Containers — 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
    }
}
```


130

IV. Containers — 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.**

131

IV. Containers — 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.**

132

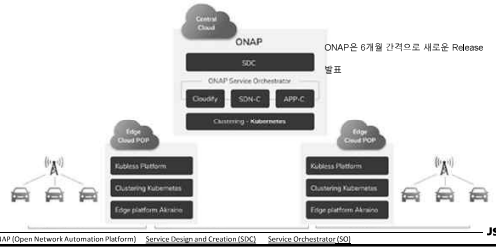
IV. Containers 3. Orchestration

- 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

133

IV. Containers 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)



134

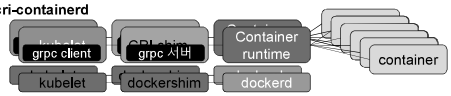
IV. Containers 3. Orchestration

- 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

135

IV. Containers 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



136

IV. Containers 3. Orchestration

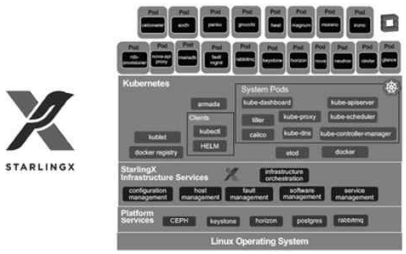
- K8s Cluster간 네트워크 연결 요구
- Open source project 'Submariner' (예)
  - 카산드라(Cassandra)와 같은 HA 데이터베이스의 지역 분산
  - 분산화 트레이스 (Distributed Tracing)
  - Service Mesh의 클러스터들 간에 확대

https://submariner.io/

137

IV. Containers 3. Orchestration

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



https://www.starlingx.io/

138

목차

139

IV. Containers 4. Unikernels

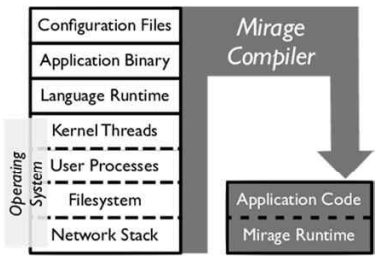
- "Unikernels are specialized, single-address-space machine images constructed by using library operating systems"
- With unikernels, we can also select the part of the kernel needed to run with the specific application. With unikernels, we can create a single address space executable, which has both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

http://unikernel.org/

140

IV. Containers 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)

141

IV. Containers 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.

142

IV. Containers 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.

143

IV. Containers 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, HaVM, 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.

144

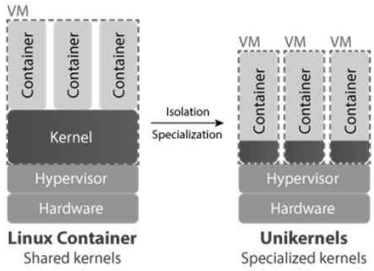
IV. Containers 4. Unikernels

- 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.

145

IV. Containers 4. Unikernels

- Shared Kerner vs. Unikernel



146

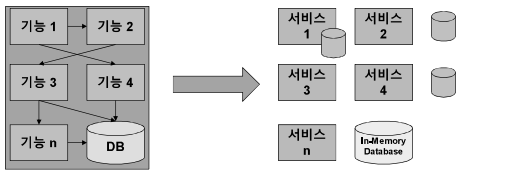
목차

❖ 실습교재 (별도)

147

IV. Containers 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

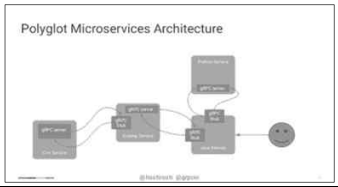


CI/CD (Continuous integration and Continuous delivery)

148

IV. Containers 5. Microservices

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

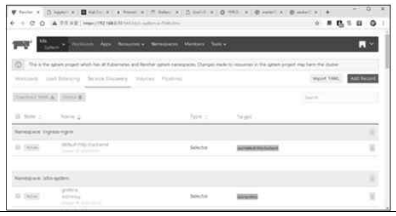


https://medium.com/@akshitjain_74512/inter-service-communication-with-grpc-c815a561e3a1

149

IV. Containers 5. Microservices

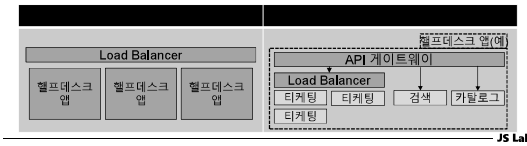
- Discovery Service
  - Client는 기존에는 1개의 monolithic app을 하였으나 MSA에서는 동시에 여러 개의 서비스를 호출해야 함
  - Client는 서비스들의 위치를 알아야 함 (Host/IP/Port 등)
  - Client가 마이크로서비스 호출시 제공하는 entry point 실시간 위치를 하드코드(Hard code)보다는 유연한 방법으로 현재의 위치를 제공해야 함



150

IV. Containers 5. Microservices

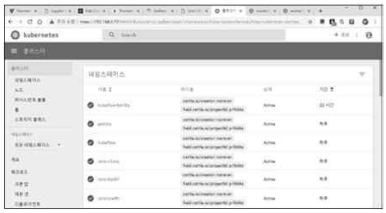
- 조직의 Learning Curve 고려
  - 단일 마이크로서비스 (Standalone Microservice) 많이 필요시
  - 과도한 의존성으로 시간이 많이 필요하고 코드의 품질이 낮아질 때
  - 한가지 요소로 애플리케이션 장애 시
- 마이크로서비스 전환 시 추가 기능 고려
  - 비즈니스 기능을 위한 서비스 연결
  - 스탠드얼론 and/or 서비스의 부분 적용
  - 각 엔지니어링 팀은 비즈니스 영역의 이해하고 책임을 소유



151

IV. Containers 5. Microservices

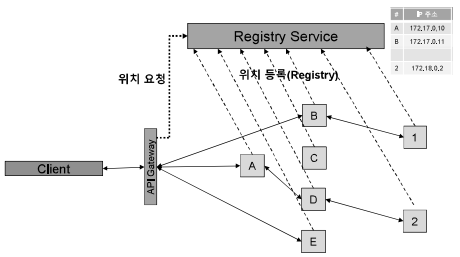
- API Gateway
  - 모든 호출에 대한 1개의 entry point가 필요
  - 내부의 복잡성을 숨김
  - 마이크로 서비스 변화/결합/구분/추가/제거 유연함
  - Client와 Application사이의 왕복 단순화로 효율성 증대



152

IV. Containers 5. Microservices

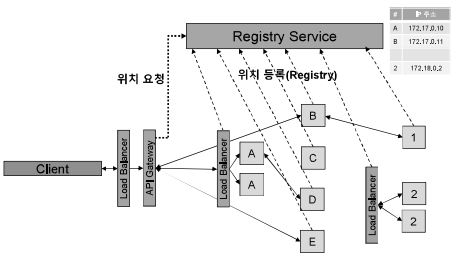
- Service registry
  - API Gateway는 모든 서비스에 대한 IP 주소를 알고 이의 DB 필요
  - Registry 데이터를 위해 오픈소스 사용 (Consul이나 SkyDNS)



153

IV. Containers 5. Microservices

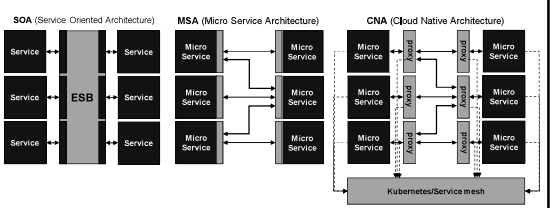
- 로드밸런서 추가시 Service registry
  - API Gateway는 모든 서비스에 대한 IP 주소를 알아야 하며 이의 DB 필요
  - Registry 데이터의 안정성을 위해 오픈소스 사용(Consul이나 SkyDNS)



154

IV. Containers 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



155

IV. Containers 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".**

156

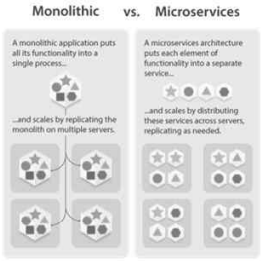
IV. Containers — 5. Microservices

- 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.

157

IV. Containers — 5. Microservices

- Monolithic vs Microservices



158

IV. Containers — 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

159

IV. Containers — 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.

160

IV. Containers — 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.

161

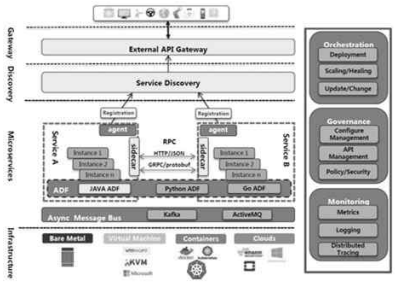
IV. Containers — 5. Microservices

- 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.

162

IV. Containers 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

163

목차

164

IV. Containers 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.**

165

IV. Containers 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.

166

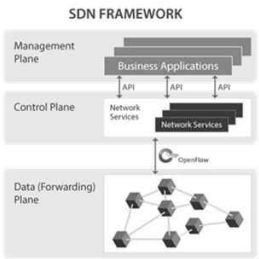
IV. Containers 6. SDN

- **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).

167

IV. Containers 6. SDN

- **SDN Framework**



168

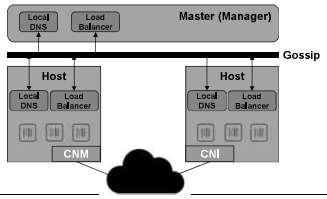
IV. Containers 6. SDN

- 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.**

169

IV. Containers 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



170

IV. Containers 6. SDN

- 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.

171

IV. Containers 6. SDN

- 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.

172

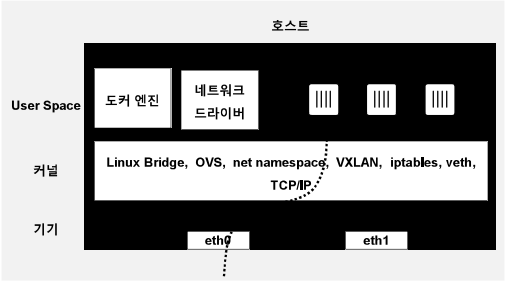
IV. Containers 6. SDN

- 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.

173

IV. Containers 6. SDN

- 도커 네트워킹은 리눅스 네트워킹



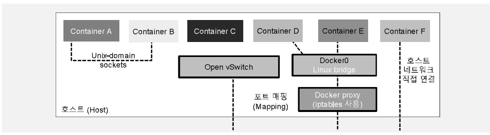
IPAM: IP Address Management

174

IV. Containers 6. SDN

▫ 하이레벨 (High-level) 기능

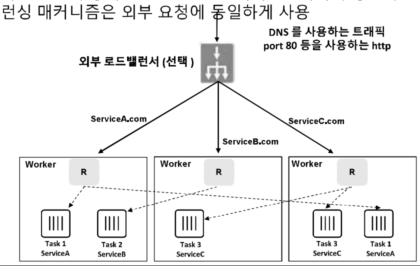
Namespace	/proc에서 프로세스 수준 관리의 컨테이너 네트워킹
Linux Bridge	커널에서 포워딩에 사용하는 L2/MAC을 인식하는 스위치
Open vSwitch	프로그램 가능하고 터널링을 지원하는 개선한 브릿지 (SDN 스위치)
NAT	네트워크 주소 변화 IP address + Ports (Types: SNAT, DNAT)
iptables	커널 내의 정책 엔진으로 패킷전송, 방화벽, NAT를 관리함
Unix domain sockets	단일 호스트 내 통신 기반의 File descriptor, FIFO 파이프로 동작
User-space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



175

IV. Containers 6. SDN

▫ **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 로드 밸런싱 매커니즘은 외부 요청에 동일하게 사용



176

IV. Containers 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
```


177

IV. Containers 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
```


178

IV. Containers 6. SDN

▫ Inspecting a Bridge Network

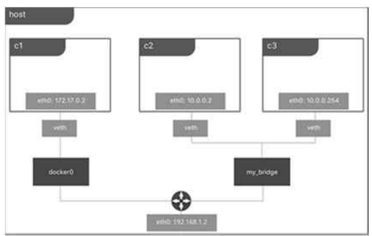
```
$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "6f30debc5baff467d437e3c7c3de673f21b51f821588aca2e30a7db68f10260c",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16"
                }
            ]
        },
        "Internal": false,
        "Containers": {
            "6131 1c7612a9db597e7e0efbd1cc102426edea02d9b281061967e25a4841733f": {
                "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": {}
    }
]
```


179

IV. Containers 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)

180

IV. Containers 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
```


181

IV. Containers 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:

182

IV. Containers 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
```


183

IV. Containers 6. SDN

- Sharing Network Namespaces Among Containers (2 of 2)

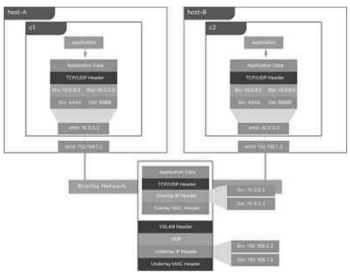
```
$ docker container run -it --name=c6 --net=container:c5 busybox /bin/sh
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128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


184

IV. Containers 6. SDN

- Docker Multi-Host Networking

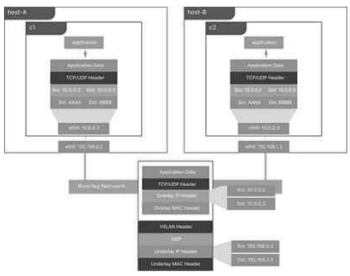


Docker Overlay Driver (by Nigel Poulton, retrieved from GitHub)

185

IV. Containers 6. SDN

- Docker Multi-Host Networking



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

186

IV. Containers 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

187

IV. Containers 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.

188

IV. Containers 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.

189

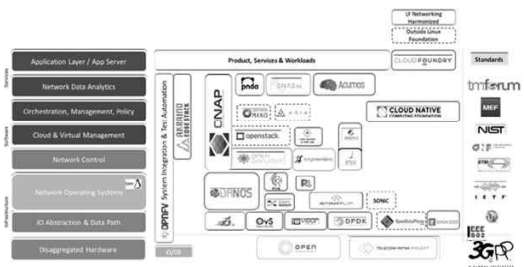
IV. Containers 6. SDN

- **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**

190

IV. Containers 6. SDN

- **오프소스와 SDN Landscape**
- **리눅스 재단 내/외의 K8s 도입 네트워킹 프로젝트 증가 중**



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

191

목차

192

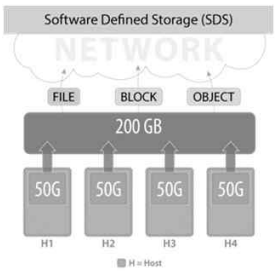
IV. Containers 7. SDS

- **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

193

IV. Containers 7. SDS

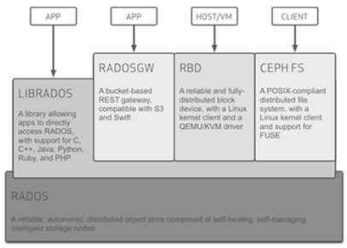
- **Software-Defined Storage (SDS)**



194

IV. Containers 7. SDS

- **Ceph Architecture**



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

195

IV. Containers 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.

196

IV. Containers 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.

197

IV. Containers 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.**

198

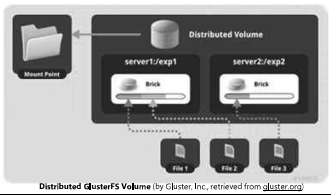
IV. Containers 7. SDS

- 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

199

IV. Containers 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)

200

IV. Containers 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)

201

IV. Containers 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.

202

IV. Containers 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.

203

IV. Containers 7. SDS

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

204

IV. Containers 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.

205

IV. Containers 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
```


206

IV. Containers 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
```


207

IV. Containers 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.

208

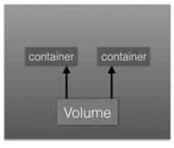
IV. Containers 7. SDS

- 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.

209

IV. Containers 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.



Pod

210

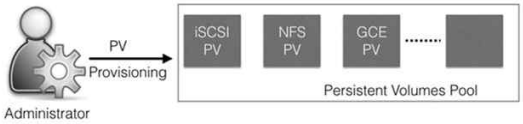
IV. Containers 7. SDS

- 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.

211

IV. Containers 7. SDS

- Persistent Volumes



212

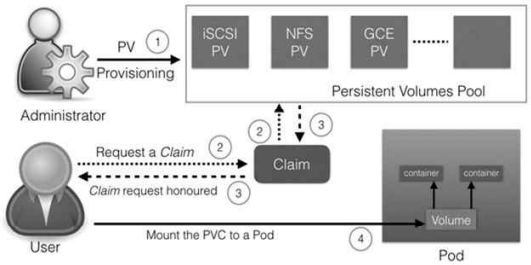
IV. Containers 7. SDS

- 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.

213

IV. Containers 7. SDS

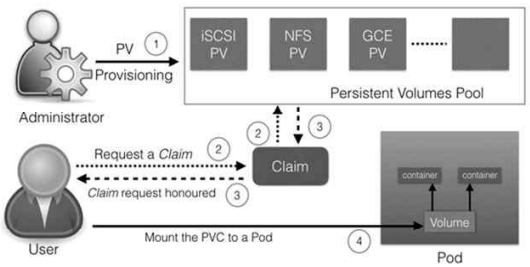
- Persistent Volumes Claim



214

IV. Containers 7. SDS

- Using PVC in a Pod



215

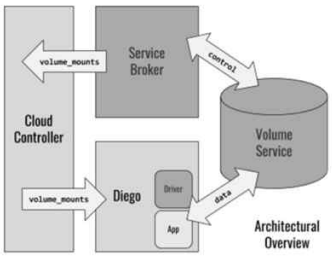
IV. Containers 7. SDS

- 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.**

216

IV. Containers 7. SDS

- Cloud Foundry Volume Service



217

IV. Containers 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.

218

목차

219

V. DevOps and 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**

220

V. DevOps and 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.**

221

V. DevOps and 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.**

222

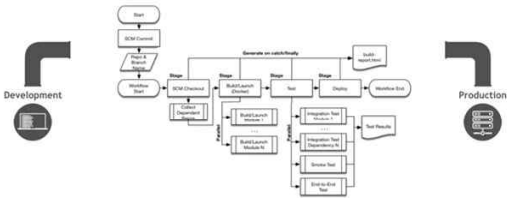
V. DevOps and CI/CD

- 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".

223

V. DevOps and 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)

224

V. DevOps and 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.

225

V. DevOps and 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.

226

V. DevOps and 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.

227

V. DevOps and 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

228

V. DevOps and CI/CD

- Travis CI Characteristics
  - Travis CI supports different databases, like MYSQL, RIAK, memcached, 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.

229

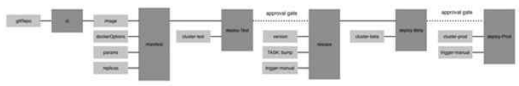
V. DevOps and CI/CD

- 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.

230

V. DevOps and 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".
-

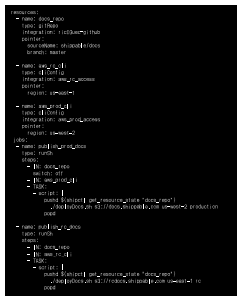


The `shippable.yml` Structure (by Shippable, Inc., retrieved from shippable.com)

231

V. DevOps and 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.



232

V. DevOps and 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.

233

V. DevOps and 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.

234

V. DevOps and 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.

235

V. DevOps and 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.

236

V. DevOps and CI/CD

- 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.

237

V. DevOps and 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 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]
```


238

V. DevOps and 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.

239

V. DevOps and 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.

240

목차

241

VI. 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.

242

VI. 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.

243

VI. 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.**

244

VI. 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.**

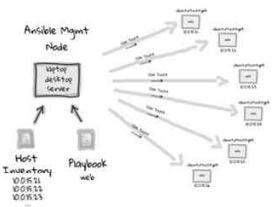
245

VI. 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)

246

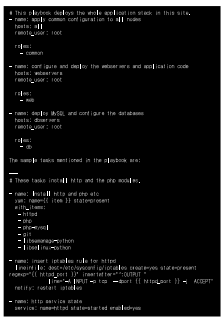
VI. Tools 1. Config Mgmt

- 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.

247

VI. 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.

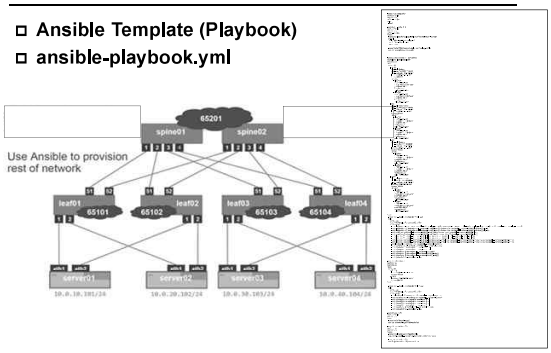


ansible Multi-Node Deployment Workflow (retrieved from sysadmincasts.com)

248

VI. Tools 1. Config Mgmt

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



249

VI. 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 ---------------------------------------------------------------- 1.88s
reset : Apply Default Interface Configuration ---------------------------------- 1.76s
Copy Interfaces Configuration File --------------------------------------------- 1.52s
Gathering Facts ---------------------------------------------------------------- 1.20s
cumulus@oob-mgmt-server:~/NetworkAutomation$
```


250

VI. 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.

251

VI. 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.

252

VI. 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.

253

VI. 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.**

254

VI. 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
}
```


255

VI. 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.

256

VI. 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.**

257

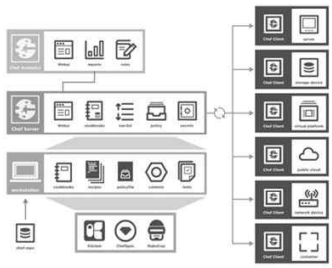
VI. Tools — 1. Config Mgmt

- **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.**

258

VI. Tools — 1. Config Mgmt

- Chef Overview



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

259

VI. 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

260

VI. 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.**

261

VI. 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.**

262

VI. 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.**

263

VI. Tools — 1. Config Mgmt

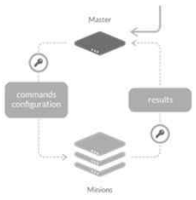
- Salt Minions
  - **Minions can be installed on:**
    1. Unix-based systems
    2. Windows
    3. Mac OS X.

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

264

VI. 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)

265

VI. 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.

266

VI. 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.

267

목차

268

VI. 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.

269

VI. 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).

270

VI. Tools — 2. Build & Release

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

271

VI. 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"

272

VI. 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.

273

VI. 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.

274

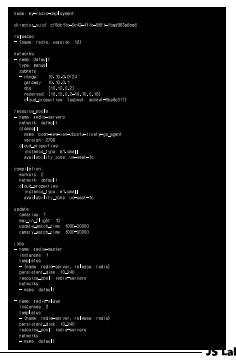
VI. Tools — 2. Build & Release

- 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.

275

VI. 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:



276

VI. 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.

277

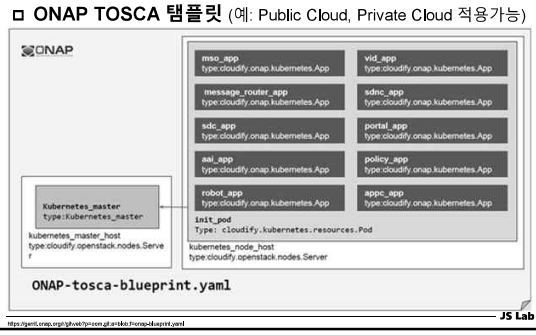
VI. Tools 2. Build & Release

- 관리 표준과 오픈소스

278

VI. Tools 2. Build & Release

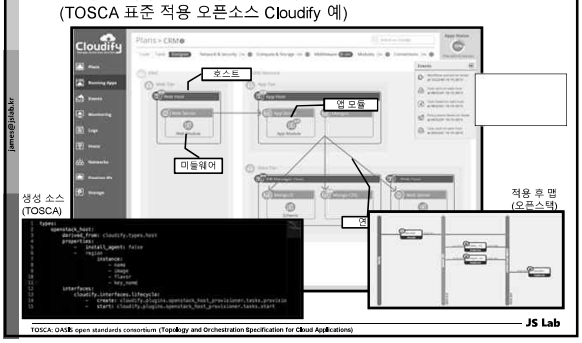
- Telco용 멀티벤더 환경 관리 표준 TOSCA
- ONAP TOSCA 탬플릿 (예: Public Cloud, Private Cloud 적용가능)



279

VI. Tools 2. Build & Release

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

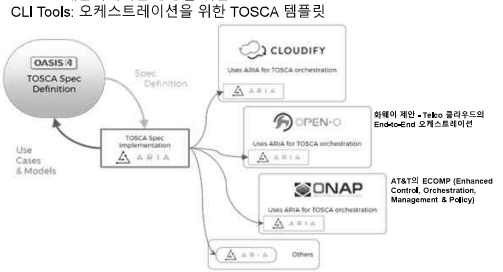


280

VI. Tools 2. Build & Release

- 표준 TOSCA 스펙 적용 오픈소스 'ARIA'

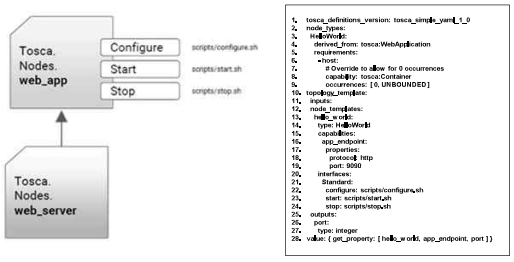
오케스트레이션이 TOSCA 프로파일 지원을 위한 Python 라이브러리
TOSCA 애플리케이션 생성을 위한 SDK
CLI Tools: 오케스트레이션을 위한 TOSCA 템플릿



281

VI. Tools 2. Build & Release

- 표준 TOSCA 적용 'ARIA' Hello World 예 (TOSCA/ARIA)
  - 소스 공개
  - helloworld.yaml

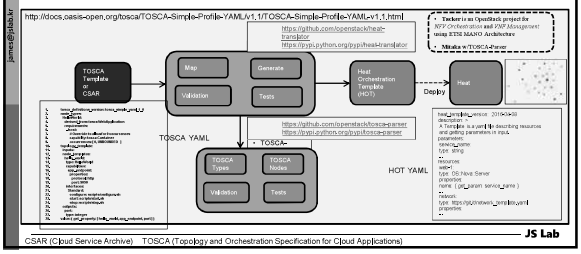


https://github.com/apache/incubator-ariatosca/blob/master/examples/hello-world/hello-world.yaml

282

VI. 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)



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

283

목차

284

VI. 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.

285

VI. 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.

286

VI. 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.

287

VI. Tools — 3. Key-Value Pair

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

288

VI. 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.

289

VI. 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.

290

VI. Tools — 3. Key-Value Pair

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

291

VI. 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.

292

목차

293

VI. Tools — 4. Image Building

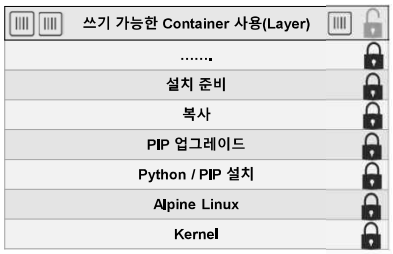
- In an immutable infrastructure environment we prefer to replace an existing service with a new one, to fix any problems/bugs or to perform an update.
- To start a new service or replace an older service with a new one, we need the image from which the service can be started. These images should be 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.

294

VI. Tools — 4. Image Building

❖ **이미지:** 도커 컨테이너의 기반이며, 앱을 표현함

❖ **이미지 레이어**



295

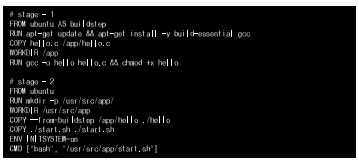
VI. Tools — 4. Image Building

- **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.**

296

VI. 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.**



297

VI. Tools — 4. Image Building

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

298

VI. 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.

299

VI. 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.**

300

목차

301

VI. 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.

302

VI. 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.

303

VI. 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

304

VI. 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

305

VI. 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".

306

VI. 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.

307

VI. 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.

308

VI. 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.

309

VI. 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.

310

VI. 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.

311

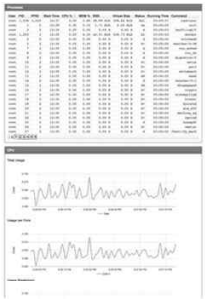
VI. Tools — 5. Debug Log Monitor

- 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.

312

VI. Tools — 5. Debug Log Monitor

- Host System Resource Usage with cAdvisor



313

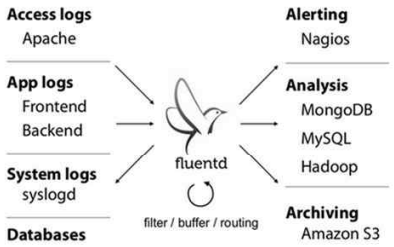
VI. Tools — 5. Debug Log Monitor

- Docker Host Specific Details with cAdvisor

314

VI. Tools — 5. Debug Log Monitor

- Fluentd

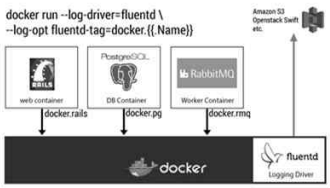


fluentd Architecture (by Treasure Data, Inc., retrieved from fluentd.org)

315

VI. Tools — 5. Debug Log Monitor

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



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

316

VI. 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.

317

VI. 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.

318

VI. 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.

319

VI. 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)

320

VI. 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.

321

VI. 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.

322

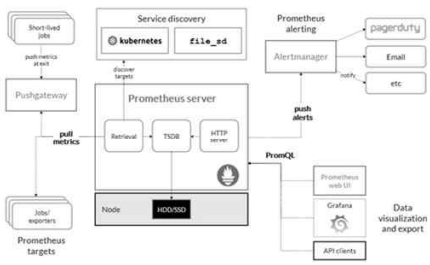
VI. Tools — 5. Debug Log Monitor

- 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.

323

VI. Tools — 5. Debug Log Monitor

- Prometheus Architecture



Prometheus Architecture (retrieved from prometheus.io)

324

목차

325

VII. 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.

326

VII. 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.

327

VII. 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

328

VII. 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.

329

VII. 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.**

330

VII. 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.

331

VII. 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.

332

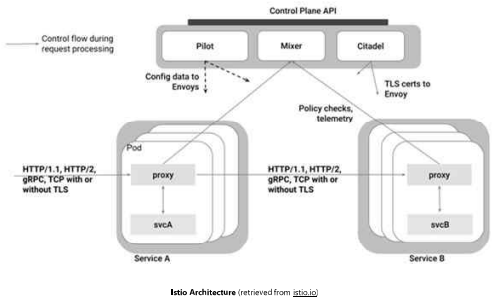
VII. Service Mesh

- 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.

333

VII. Service Mesh

- Istio Architecture



Istio Architecture (retrieved from istio.io)

334

VII. 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).

335

VII. 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.

336

VII. 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.

337

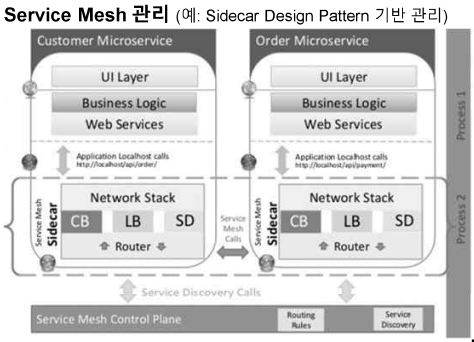
VII. Service Mesh

- 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.

338

VII. Service Mesh

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

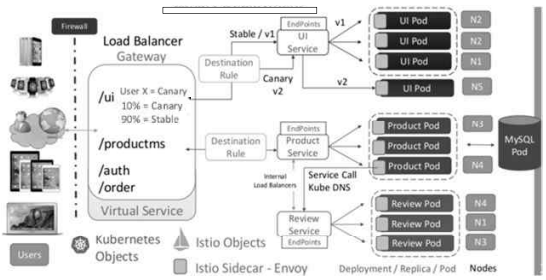


Circuit Breaker(CB), Load Balancer(LB), Service Discovery(SD)

339

VII. Service Mesh

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



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

340

목차

341

VIII.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".

342

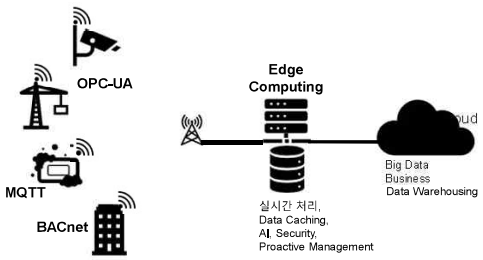
VIII.IoT

- 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.

343

VIII.IoT

- 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

344

VIII.IoT

- 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)

345

VIII.IoT

- 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.**

346

VIII.IoT

- 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.**

347

VIII.IoT

- 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.

348

VIII.IoT

- 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?

349

목차

350

IX. Serverless, FaaS

- 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.

351

IX. Serverless, FaaS

- 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.

352

IX. Serverless, FaaS

- 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.

353

IX. Serverless, FaaS

- 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.

354

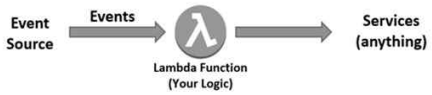
IX. Serverless, FaaS

- 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.

355

IX. Serverless, FaaS

- 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)

356

IX. Serverless, FaaS

- 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.

357

IX. Serverless, FaaS

- 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)

358

IX. Serverless, FaaS

- 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.

359

IX. Serverless, FaaS

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

360

IX. Serverless, FaaS

- **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.

361

IX. Serverless, FaaS

- **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.**

362

IX. Serverless, FaaS

- **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.

363

IX. Serverless, FaaS

- **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.

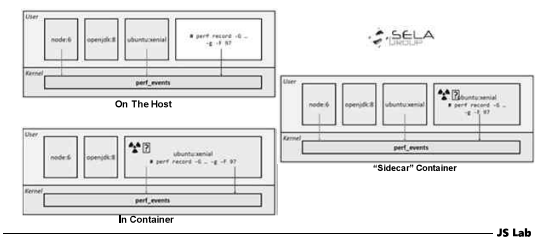
364

목차

365

X. OpenTracing

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



366

X. OpenTracing

- 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.

367

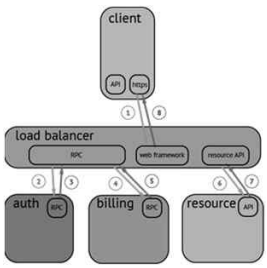
X. OpenTracing

- 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.

368

X. OpenTracing

- Tracing with OpenTracing

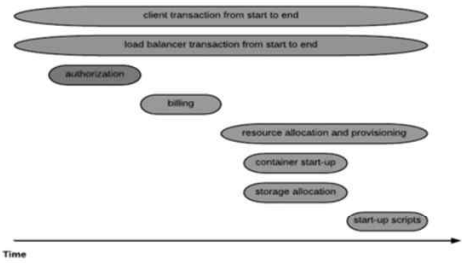


A General Tracing for Microservices-based Applications (retrieved from opentracing.io)

369

X. OpenTracing

- Visualization of a Trace Collected via OpenTracing



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

370

X. OpenTracing

- 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.

371

X. OpenTracing

- 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.

372

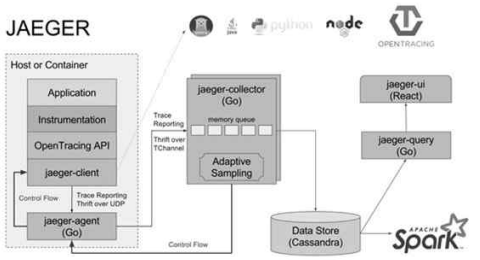
X. OpenTracing

- 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.

373

X. OpenTracing

- Jaeger Architecture



Jaeger Architecture (retrieved from jeagertracing.io)

374

X. OpenTracing

- 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.

375

목차

376

XI. 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.

377

XI. 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.

378

XI. 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.

379

XI. 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.

380

XI. 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.

381

XI. 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.

382

XI. 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.

383

XI. 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.

384

XI. 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.

385

XI. 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.

386

XI. 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.

387

XI. 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.

388

XI. 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.

389

목차

390

391