

System Architecture Chapter1

(실무적 사례기반 아키텍처 포함)

Dosung(Dave) Kim, Ph.D.

Contents of Table

I. 시스템 아키텍처 개요 및 특징

II. 시스템 아키텍처 관련 구성요소

- A. 서버 및 데이터베이스
- B. 네트워크
- C. 데이터 처리
- D. 스토리지
- E. 부하분산(Load balancing)
- F. 고 가용성(High Availability)

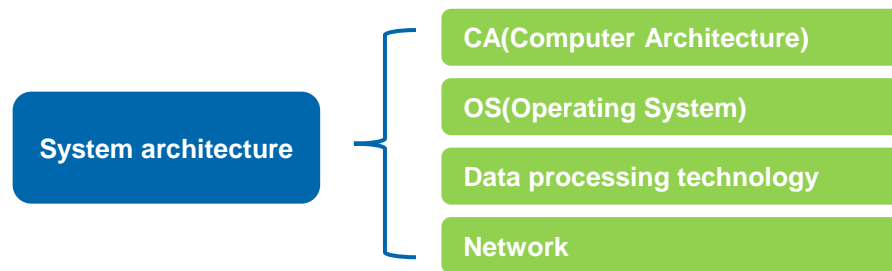
III. 실무적 사례기반 시스템 아키텍처

- A. SaaS기반 프로토타입핑 솔루션
- B. 시스템 반도체 기업 웹사이트
- C. 글로벌제약회사의 임상연구 정보관리시스템
- D. Native아키텍처 기반 설문조사 시스템

시스템 아키텍처 개요 및 특징(1/2)

- 시스템 아키텍처 개요

- 시스템의 구조, 작동을 위한 행위, 관련되는 뷰를 정의하는 개념적 모형
 - 개념적 모형: 실질적인 프로그램들 간의 관계를 나타내는 High level수준의 조직 고유 모델
- 하드웨어와 소프트웨어를 모두 포함한 시스템의 전체적인 구성을 큰 그림으로 표현한 청사진
- 구성요소



- 시스템 아키텍처 특징

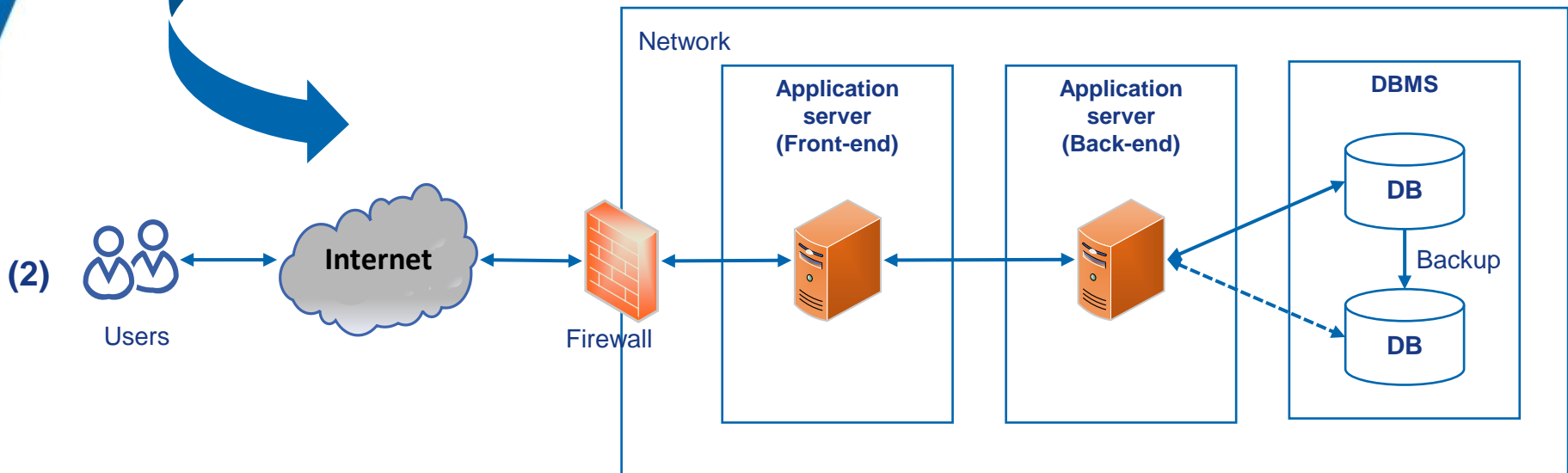
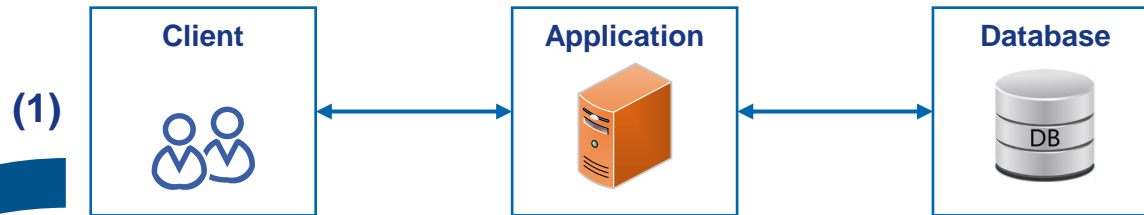
- 설계된 아키텍처(청사진)을 통해 기본적으로 시스템이 어떤 하드웨어 기반으로 소프트웨어를 사용하는지 알 수 있음.
- 하드웨어/소프트웨어의 시스템 간의 상호작용이 정상작동을 위해 어떻게 동작하는지를 쉽게 알 수 있음.
- 기획, 분석, 설계, 구현, 테스트 및 운영을 위해 개발과 관련된 이해관계자들(PM, 개발자, 품질 및 테스트 관리자, 사용자등)의 이해를 도울 수 있음.
- 너무 복잡하면 안되고 심플함을 유지해야 함.

시스템 아키텍처 개요 및 특징(2/2)

- 시스템 아키텍처 특징

- 일반적인 예시

- 1번의 전통적인 어플리케이션 시스템의 단점을 개선하고자 보다 고도화된 2번의 시스템 아키텍처를 설계 및 구현을 함.

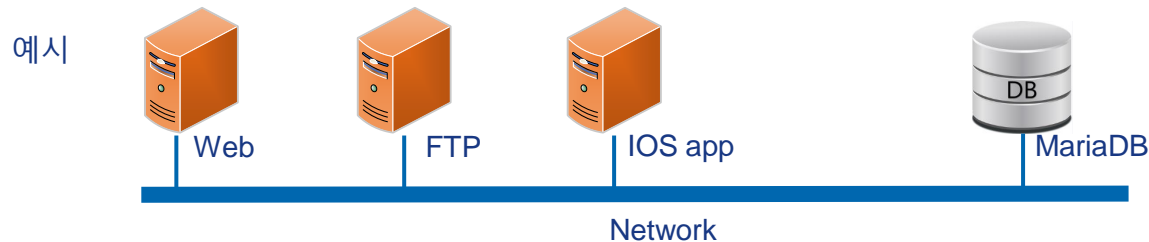


시스템 아키텍처 관련 구성요소(1/36)

• 서버 및 데이터 베이스

– 서버

- 서버(Server)는 클라이언트(사용자)에게 네트워크를 통해 어떤 목적의 서비스를 제공하는 컴퓨터시스템을 의미함.
- 서버에는 서비스를 제공하기 위해 운영체제(Operating system)을 필수적으로 설치 및 설정을 해야 함. 예) 유닉스, 리눅스 및 윈도우등
- 일반적으로 데이터 저장을 위해 스토리지 및 DBMS(데이터베이스 관리 시스템)와 연결되어 작동함.



– 서비스 종류에 따른 서버유형

구분	시스템 종류	설명
파일관리	FTP, samba, sshfs, DAS, NAS 및 NFS등	대용량 파일등을 전송하기 위한 서비스
Application	Web(Nginx,Apache, IIS, lighthttpd) , Andriod, IOS등	실질적인 시스템의 기능로직을 담당하는 서버, 일반적으로 데이터베이스와 연결되어 서비스를 제공함.
데이터베이스	Oracle, MSSQL, MySQL, MariaDB, MongoDB등	데이터베이스가 구동되는 서버. Application 서버와 동적기능처리를 위해 연동됨.

시스템 아키텍처 관련 구성요소(2/36)

- 서버 및 데이터 베이스

- 데이터베이스

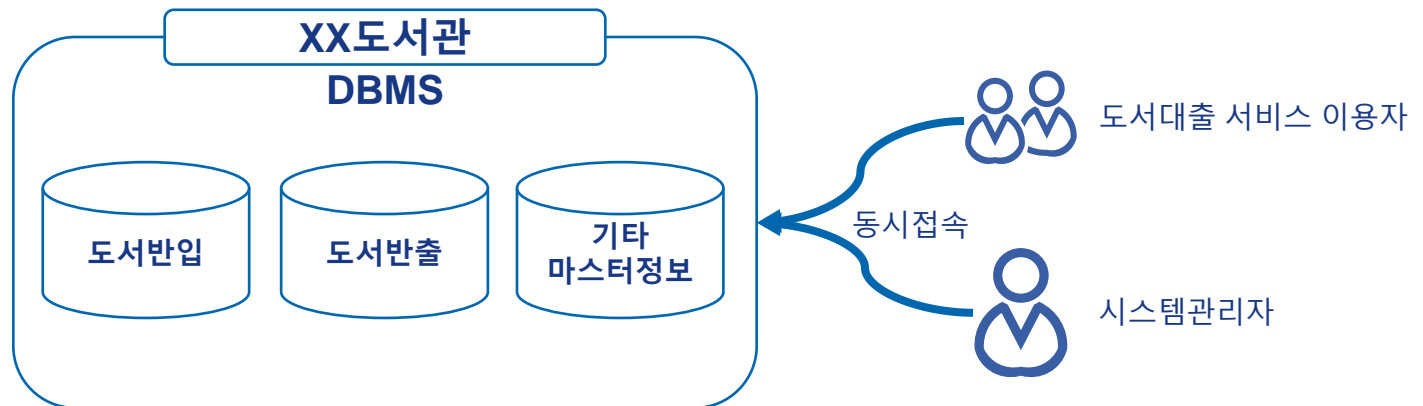
- 시스템적으로 저장된 체계적인 데이터의 집합
 - 이미지, 비디오, 숫자, 단어등 일상생활에서 사용할 수 있는 모든 데이터들이 포함될 수 있음.

- DBMS(Database Management System)

- 데이터베이스를 관리하고 운영하는 소프트웨어
 - 데이터가 저장되어 있는 데이터베이스는 여러 사용자 및 시스템이 동시에 공유 및 접근이 가능해야 함.

- 예시

- 만약에 어느 지역에서 시립도서관 시스템으로 도서를 대출하는 DBMS가 있다면 아래와 같은 시스템 구조로 될 수 있음.



시스템 아키텍처 관련 구성요소(3/36)

• 서버 및 데이터 베이스

– DBMS 종류 및 특징

- Application을 개발하고 데이터를 사용하기 위해서는 DBMS를 설치해야 함. 대표적으로 많이 사용하는 DBMS는 오라클(Oracle), MSSQL, MySQL, MariaDB등이 있음.
- 실무에서는 특히 MySQL과 MariaDB를 많이 사용하고 있음.

DBMS 종류	개발사	호환운영체제	비고
Oracle	Oracle	Unix, Linux and Windows	상용라이선스 존재
MSSQL	Microsoft	Windows 서버	유무료라이선스 존재
Access	Microsoft	Windows	상용라이선스
MySQL	Oracle	Unix, Linux, Windows and Mac	오픈소스(무료)와 상용라이선스 존재
MariaDB	MariaDB	Unix, Linux and Windows	오픈소스(무료)
PostgesSQL	PostgesSQL	Unix, Linux, Windows and Mac	오픈소스(무료)
DB2	IBM	Unix, Linux and Windows	상용라이선스
SQLite	SQLite	Android and IOS	오픈소스(무료), 모바일 전용

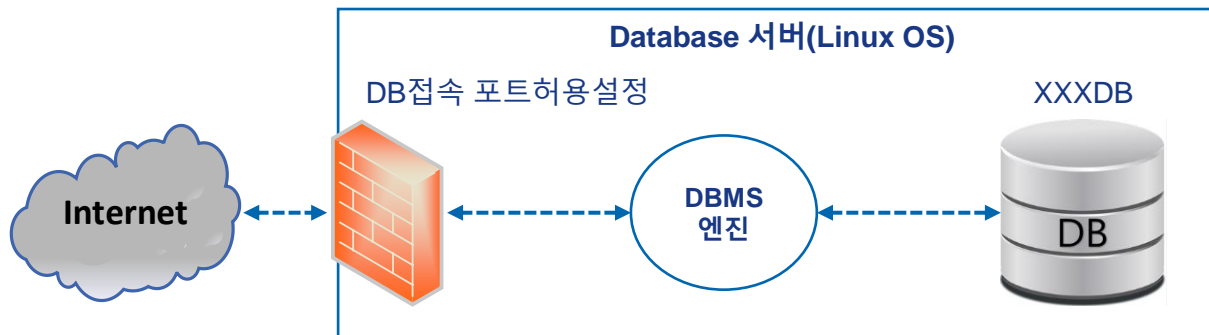


시스템 아키텍처 관련 구성요소(4/36)

- 서버 및 데이터 베이스

- DB서버구축실습

- Application을 개발하여 데이터처리를 하기 위해서는 OS상에 DB서버시스템(DBMS)의 설치 및 설정을 해야함.



- ① 우선 한대의 물리적 또는 논리적(가상머신-Virtual machins)서버에 리눅스 OS를 설치함.
- ② 리눅스 OS가 설치완료 되었다면 개발목적에 맞는 DBMS를 설치함.

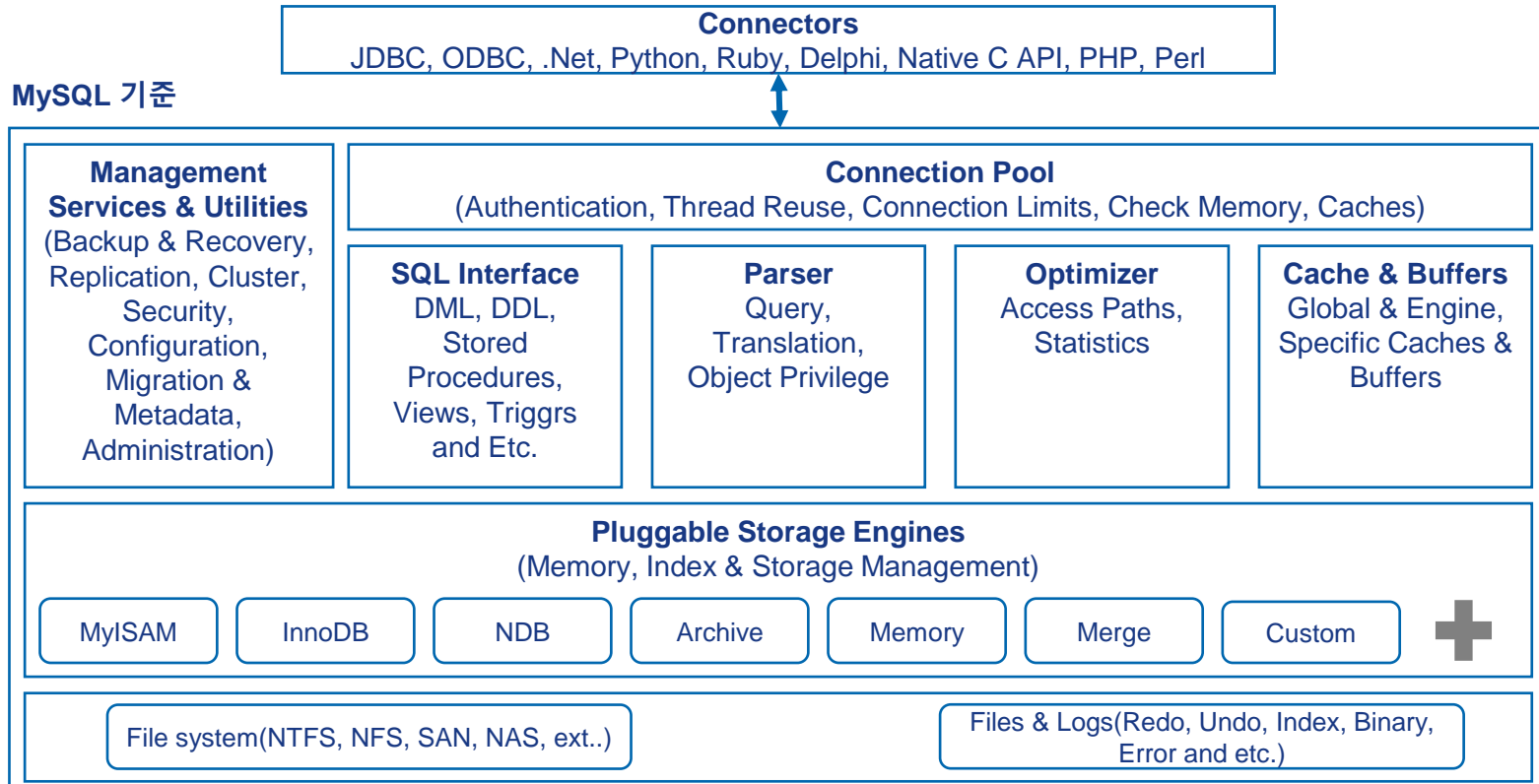
* VM을 하나 생성해서 각 DBMS를 설치하고 개발툴을 사용하여 접속하는 실습을 진행할 것임.

시스템 아키텍처 관련 구성요소(5/36)

• 서버 및 데이터 베이스

– MySQL / MariaDB 엔진 아키텍처

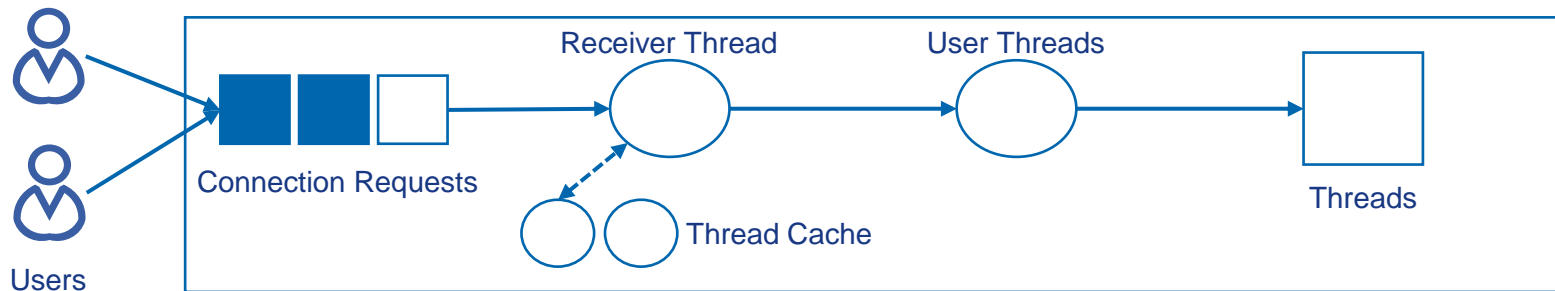
- 일반적인 RDBMS와 비슷하게 MySQL/MariaDB의 도 엔진영역과 데이터영역으로 나눌 수 있음.
- MySQL은 다른 RDBMS와 다르게 데이터영역에서 사용할 수 있는 스토리지 엔진이 여러 개가 존재함. 따라서 그 중에 필요한 엔진을 선택해서 사용할 수 있음.
- 많이 사용하는 스토리지 엔진은 MyISAM 및 InnoDB가 있음.



시스템 아키텍처 관련 구성요소(6/36)

- 서버 및 데이터 베이스

- MySQL / MariaDB 엔진 아키텍처
- Connection Pool
 - 사용자들은 Connection Pool을 통해 세션 및 스레드 처리를 한다.



- 왼쪽의 사용자들은 TCP/IP통신 및 해당 DB의 포트를 통해 DB에 접속 요청을 보냄. MySQL은 3306포트
- 연결요청들은 계속 적재가 되며 Receiver Thread가 각 연결요청에 대해 하나의 User Thread로 할당해줌. 그 이후에 실질적인 DB Processing은 User Thread가 처리 함.
- SQL Interface
 - DB에서 사용하는 SQL명령어, 프로시저 및 함수등의 기능을 담당하는 부분
- Parser
 - SQL Interface로부터 실행되는 SQL명령어를 DBMS에서 처리될 수 있도록 DB언어로 변환하는 역할
- Optimizer
 - Parser를 통해 변환된 구문에 대해서 최적화 하는 부분임. 이를 위해 Executer가 실행담당 함.
- Cache & Buffers
 - 속도등 퍼포먼스 개선을 고려하여 실행에 관련된 데이터 및 참조사항, 객체들의 구조 및 스토리지 엔진이 사용 할 내용을 임시적으로 저장하는 메모리 공간임.

시스템 아키텍처 관련 구성요소(7/36)

• 서버 및 데이터 베이스

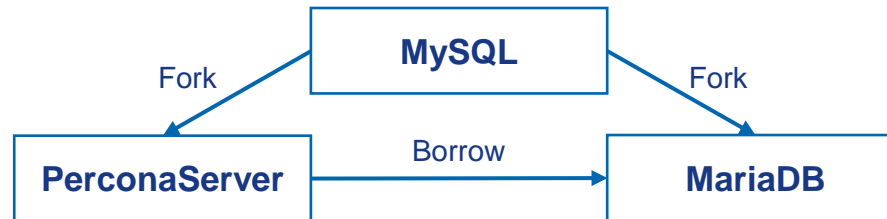


MariaDB

– DB서버구축실습

– MariaDB일 경우

- MySQL과는 높은 수준의 호환성을 제공함. 따라서 실행되는 프로그램들은 거의 mysqlxxx로 MySQL과 동일함. 결국 MySQL과 MariaDB는 마이그레이션도 쉽게 할 수 있음.
- MariaDB는 MySQL개발자가 만든 것이며 MySQL을 fork하여 오픈소스기반으로 서비스화 한 것임.



- 상위 그림을 보면 알 수 있듯이 MariaDB의 스토리지 엔진은 MySQL & PerconaServer의 엔진을 모두 사용하고 있다는 것을 알 수 있음.
- 다음과 같은 MariaDB와 MySQL의 비교정보를 정리할 수 있음.

공통점

같은 실행 프로그램 및 유틸리티

클라이언트와의 API와 통신프로토콜이 호환됨.

원자성, 일관성, 독립성 및 내구성등 ACID규정을 준수

SQL기반으로 데이터를 관리하고 처리함

기본적으로 오픈소스 소프트웨어임.

SSL/TLS연결지원, 사용자 인증 및 권한부여, 암호화등 유사한 보안기능을 제공함.

시스템 아키텍처 관련 구성요소(8/36)

• 서버 및 데이터 베이스



- MariaDB일 경우
 - 스토리지 엔진 비교

용도	MariaDB	MySQL
메모리 엔진	MySQL 기본 코드가 포함된 메모리 스토리지 엔진	MariaDB 메모리 스토리지 엔진과 매우 흡사함.
디스크기반 임시 테이블 엔진	<ul style="list-style-type: none"> - 기본적으로 Aria엔진을 사용함. - InnoDB의 인덱싱기술과 비슷하여 레코드 데이터까지 모두 메모리 Cache를 사용할 수 있음. 따라서 MySAM 스토리지 엔진보다 빠름. 	기본적으로 MySAM 스토리지 엔진을 사용함.
트랜잭션 지원 엔진	<ul style="list-style-type: none"> - 과거에는 PerconaServer의 XtraDB가 사용됨. - 10.0.7버전부터는 InnoDB가 기본이 됨. 	MariaDB와 동일하게 InnoDB 스토리지 엔진을 사용함.
NoSQL지원엔진	Cassandra의 데이터를 MariaDB서버를 통해 접근함.	Memcached같은 In-Memory DB의 플러그인을 제공함.

- 기본적으로 여러 벤치마킹 및 실무적 사례를 보면 DBMS의 성능은 현재 MariaDB가 조금 더 앞서는 것으로 분석되고 있음.
- MariaDB도 지속적으로 개선되고 있으며 오픈라이선스라 비용측면에서 MySQL보다는 이점이 있음. 하지만 MySQL은 상대적으로 구축사례를 통한 검증부분 및 많은 Reference들을 많이 가지고 있고 상용버전의 라이선스체제로 기술지원이 가능하여 실무에서도 많이 사용함.
- 자신의 프로젝트의 시스템에는 어떤 DB가 적합한지 분석, 선택, 설계 및 구현을 하여 테스트(검증)하는 Action이 필요함!

시스템 아키텍처 관련 구성요소(9/36)

- 서버 및 데이터 베이스

- DB서버구축실습
- 리눅스 OS를 설치 후 개발목적에 맞는 DBMS를 설치함.
- MySQL일 경우
 - 우선 리눅스의 패키지 및 커널의 업데이트를 해줘야 함.
 - yum update -y
 - 해당 OS버전의 MySQL Repository설치를 함.



```
root@DB-TEST:~  
[root@DB-TEST ~]# yum install -y https://dev.mysql.com/get/mysql80-community-release-el7-9.noarch.rpm
```

- Mysql설치

```
[root@DB-TEST ~]# yum install -y mysql-server
```

- 설치 완료 후 해당 버전이 제대로 설치되었는지 확인함.

```
root@DB-TEST:~  
[root@DB-TEST ~]# mysqld -V  
/usr/sbin/mysqld Ver 8.0.34 for Linux on x86_64 (MySQL Community Server - GPL)
```

- MySQL 자동시작 등록 및 시작

```
root@DB-TEST:~  
[root@DB-TEST ~]# systemctl enable mysqld && systemctl start mysqld  
[root@DB-TEST ~]# |
```

- MySQL데몬이 제대로 작동되는지 확인함.

```
[root@DB-TEST ~]# systemctl status mysqld  
● mysqld.service - MySQL Server  
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)  
   Active: active (running) since 2023-08-07 01:13:49 KST; 2min 51s ago  
     Docs: man:mysqld(8)  
           http://dev.mysql.com/doc/refman/en/using-systemd.html  
  Process: 57007 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited, status=0/SUCCESS)  
 Main PID: 57073 (mysqld)  
    Status: "Server is operational"  
   CGroup: /system.slice/mysqld.service  
           └─57073 /usr/sbin/mysqld  
  
8월 07 01:13:42 DB-TEST systemd[1]: Starting MySQL Server...  
8월 07 01:13:49 DB-TEST systemd[1]: Started MySQL Server.
```

시스템 아키텍처 관련 구성요소(10/36)

- 서버 및 데이터 베이스

- MySQL일 경우

- MySQL데몬 정상작동 확인 후 임시 암호를 발급하여 접속함.

```
root@DB-TEST:~  
[root@DB-TEST ~]# grep 'temporary password' /var/log/mysql.log  
2023-08-06T16:13:44.962208Z 6 [Note] [MY-010454] [server] A temporary password is generated for root@localhost: FNO*<19Bms*g
```

- 임시로 발급된 암호를 이용하여 아래와 같이 MySQL관리콘솔에 접속함.

```
[root@DB-TEST ~]# mysql -u root -p  
Enter password:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 8  
Server version: 8.0.34  
  
Copyright (c) 2000, 2023, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> |
```

- Root 관리자의 암호를 다시 초기화 후 권한을 얻은 후 show databases; 등의 기본 명령어 실행

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'Qwer1234!';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| sys |  
+-----+  
4 rows in set (0.01 sec)
```

- MySQL은 기본적으로 TCP 3306 포트를 사용하기 때문에 OS방화벽에서 열어줘야 함.

```
root@DB-TEST:~  
[root@DB-TEST ~]# firewall-cmd --permanent --add-port=3306/tcp  
success  
[root@DB-TEST ~]# |
```

시스템 아키텍처 관련 구성요소(11/36)

- 서버 및 데이터 베이스

- DB서버구축실습

- MariaDB일 경우

- MySQL의 과정이 비슷하며 최근에는 Docker container기반으로 구축하여 많이 사용함.
 - MySQL설치과정과 같이 yum(dnf) update -y 로 커널 및 시스템 패키지 업데이트 함.
 - MariaDB yum repository 추가

```
root@DB-TEST:~  
[root@DB-TEST ~]# vi /etc/yum.repos.d/MariaDB.repo
```

```
root@DB-TEST:~  
[mariadb]  
name = MariaDB  
baseurl = http://yum.mariadb.org/10.4/centos7-amd64  
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB  
gpgcheck=1
```

- MariaDB 설치

```
[root@DB-TEST ~]# yum install -y MariaDB  
Loaded plugins: fastestmirror  
Loading mirror speeds from cached hostfile  
* base: mirror.kakao.com  
* extras: mirror.kakao.com  
* updates: mirror.kakao.com
```

- MariaDB가 제대로 설치되어 있는지 버전 확인

```
root@DB-TEST:~  
[root@DB-TEST ~]# mariadb --version  
mariadb Ver 15.1 Distrib 10.4.30-MariaDB, for Linux (x86_64) using readline 5.1
```

시스템 아키텍처 관련 구성요소(12/36)

- 서버 및 데이터 베이스
 - DB서버구축실습
 - MariaDB일 경우
 - MariaDB 자동시작 등록, 데몬시작 및 root 암호 변경

```
root@DB-TEST:/etc/yum.repos.d
[root@DB-TEST yum.repos.d]# systemctl enable mariadb
Created symlink from /etc/systemd/system/mysql.service to /usr/lib/systemd/system/mariadb.service.
Created symlink from /etc/systemd/system/mysqld.service to /usr/lib/systemd/system/mariadb.service.
Created symlink from /etc/systemd/system/multi-user.target.wants/mariadb.service to /usr/lib/systemd/system/mariadb.service.
[root@DB-TEST yum.repos.d]# systemctl start mariadb
[root@DB-TEST yum.repos.d]# /usr/bin/mysqladmin -u root password
New password:
Confirm new password:
[root@DB-TEST yum.repos.d]#
```

- MariaDB 접속 및 쿼리문 실행

```
[root@DB-TEST yum.repos.d]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.4.30-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| test |
+-----+
4 rows in set (0.000 sec)

MariaDB [(none)]> |
```


시스템 아키텍처 관련 구성요소(13/36)

• 서버 및 데이터 베이스

– NoSQL

- 테이블-컬럼 스키마 없이 분산환경에서 Key-Value기반 단순검색 및 추가작업이 용이한 DBMS

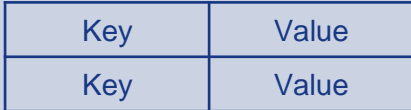
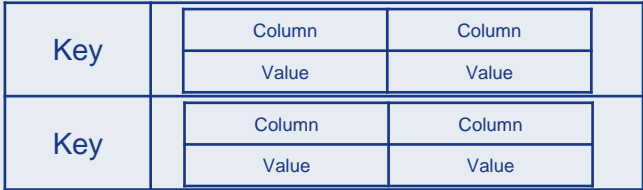
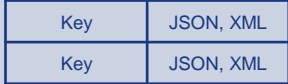
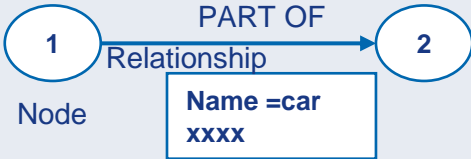
구분	특징	설명
데이터베이스 측면	스키마 유연성	- 컬럼 등 스키마 변경 자유로움 - 재가동 없이 스키마 구조 변경
	복잡쿼리 불필요	- 조인 기능없음 - 조인에 따른 쿼리 복잡성 해소
	트랜잭션 보장	- Column Family, Super Column - 트랜잭션 수행이 보장됨
	확장성	- 데이터량 증가에 따른 데이터베이스 확장성 우선 지원
데이터모델 측면	비관계형 모델	- 데이터 간 관계 정의불가 - 단순, 명료한 데이터 모델
	조회 성능 향상	- 단일테이블 접근 필요 정보확보 - 조인이 불필요하여 성능향상

– NoSQL과 관계형 DB비교

항목	NoSQL	관계형DB
데이터모델	- 스키마리스 - Key-Value관계	- 행/열로 구성 - 스키마, 인덱스등
ACID속성	- 유연한 데이터구조 - ACID속성 절충	- ACID속성이 중요 (원자, 일관, 독립 및 영속성)
확장	- Scale-Out방식 - 하드웨어 분산	- Scale-Up방식 - 하드웨어 집적도

시스템 아키텍처 관련 구성요소(14/36)

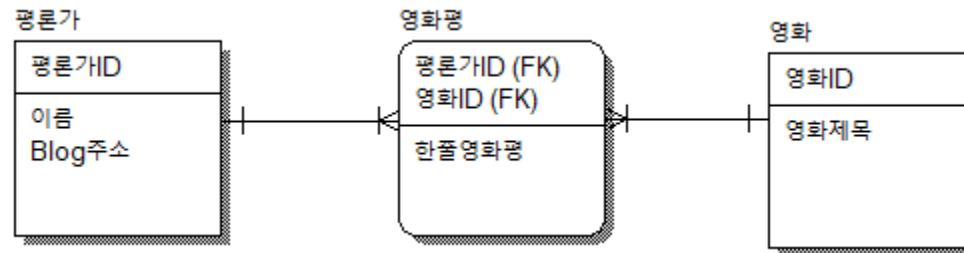
- 서버 및 데이터 베이스
 - NoSQL의 구조

모델구조	개념도	설명	DB종류
Key/Value Store	 <p style="text-align: center;">⋮</p>	<ul style="list-style-type: none"> - Unique한 Key에 하나의 Value형태모델 	AWS DynamoDB
Column Family		<ul style="list-style-type: none"> - Column Family Key내 (Column Value) 조합으로 된 여러 필드를 갖는 모델 	카산드라, HBase
Document Store	<p>User: { name: "Dave.Kim" .sex: "male" .address: {state: "Seoul" nationality: "Korea" } }</p> 	<ul style="list-style-type: none"> - 저장되는 Value데이터가 Document타입 - XML, JSON, YAML등 구조화된 타입으로 복잡한 계층 구조 표현 	MongoDB, CouchDB
Graph		<ul style="list-style-type: none"> - 정점(Node)과 에지(Edge) 데이터 표현 	Neo4J, AllegroGraph

시스템 아키텍처 관련 구성요소(15/36)

- 서버 및 데이터 베이스
 - NoSQL의 데이터 모델링 사례

1. 관계형 모델



- 영화평 ER모델로 평론가 여러 영화 한 줄 평 작성, 한 영화는 여러 평론가로부터 한 줄 평 작성

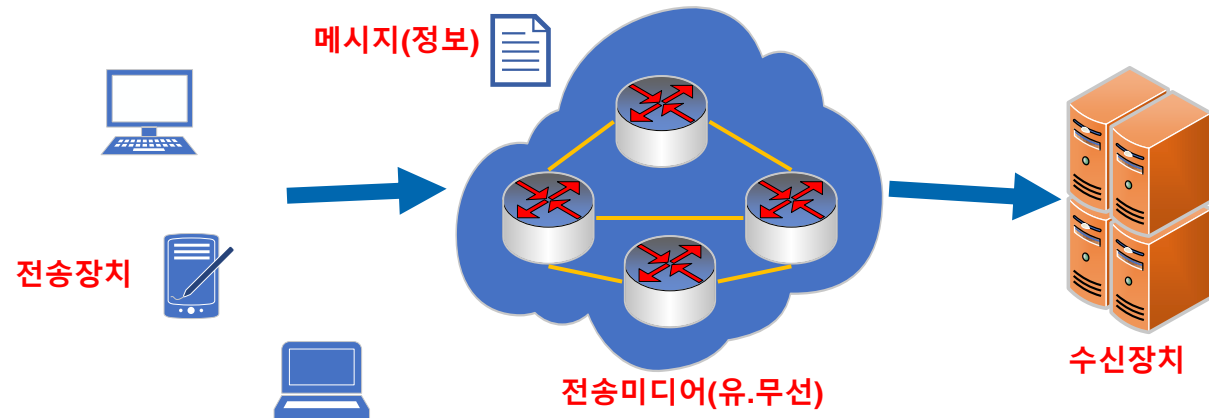
2. NoSQL모델(기본적으로 nested 엔티티로 구성, 테이블 변경 설계 가능)

Table	영화평론가	
Row key	평론가 ID	
Columns	이름 string Blog 주소 string	영화평
		영화ID byte[10] 영화제목 string
		한줄영화평 string

시스템 아키텍처 관련 구성요소(16/36)

• 네트워크

- 전송매체(Transmission Media)로 서로 연결해 데이터를 교환하는 시스템들의 모음
- 모뎀, LAN, 케이블 및 무선매체등 통신설비를 갖춘 컴퓨터를 서로 연결하는 조직이나 체계, 통신망
- 구성요소



구성요소	설명
메시지	통신을 하고자 하는 정보로 텍스트, 숫자, 그림, 또는 비디오 정보등으로 구성
전송장치	컴퓨터, 워크스테이션, 전화단말기, 비디오, 카메라 등 메시지(혹은 데이터) 전송장치
수신장치	메시지를 수신하는 장치, Ex) 서버
전송미디어	메시지가 전달되는 실제 전송로
프로토콜	데이터통신과 관련된 규칙들로 구성 Ex) TCP, UDP, FTP, HTTP등

시스템 아키텍처 관련 구성요소(17/36)

• 네트워크

- 네트워크 규모에 따른 분류

구분	설명
LAN(Local Area Network)	- 근거리 통신 네트워크로 건물과 같은 일정지역 내의 네트워크를 구성하는 형태
MAN(Metropolitan Area Network)	- 대도시 정도의 넓은 지역을 연결하기 위한 네트워크 구성형태 - IEEE802.16 working group에서 무선 MAN프로토콜 승인
WAN(Wide Area Network)	- 광범위한 지역을 수용하며, 하나의 국가내에서 도시와 도시, 혹은 국가와 국가간을 연결하는 목적으로 수백~수천km까지의 범위를 포함할 수 있도록 구성된 광역네트워크 시스템 - 흔히 인터넷이라고도 함.
PAN(Personal Area Network)	- 10m이내의 단거리 네트워크로, IEEE802.15 위원회에서 표준화됨 - 무선 PAN기술로 블루투스, 지그비등이 있음.
BAN(Body Area Network)	- 사람이 착용하는 옷이나 인체에 부착된 여러장치로부터 구성된 네트워크를 통해 데이터를 주고받음.

- OSI 7계층 및 TCP/IP 3계층

- Onpremise / Cloud의 네트워크 시스템들이 OSI7계층을 기준으로 작동하기 때문에 반드시 이해를 해야 함.

OSI	TCP/IP모델(인터넷 모델)	해당되는 시스템 및 역할
응용계층	응용계층	L7 로드밸런싱, 웹방화벽, HTTP, FTP, SMTP, Telnet
표현계층		데이터의 인코딩, 디코딩, 암호화, 복호화, ASCII, MPEG, JPEG
세션계층		통신장비의 연결관리, NetBIOS, SAP, SDP
전송계층	전송계층	L4로드밸런싱, TCP, UDP, ARP, RARP
네트워크계층	인터넷계층	L3백본 스위치, IP, IPX, X25, ICMP, IGMP
데이터 링크계층	네트워크계층	L2스위치, 이더넷, 토큰링, HDLC, LLC, PPP, MAC
물리계층		RS232C, RS449/442/423

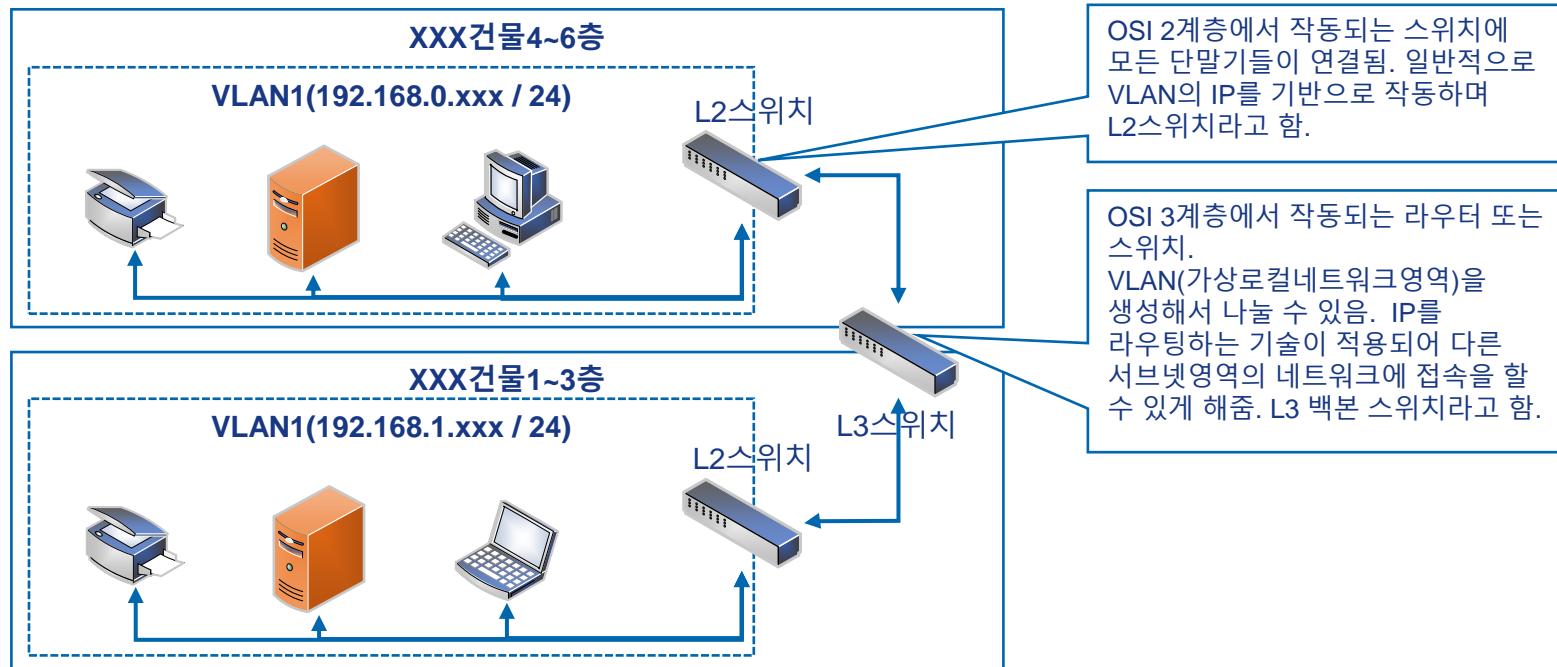
시스템 아키텍처 관련 구성요소(18/36)

• 네트워크

– LAN to LAN 망구조의 시스템

- 같은 건물 또는 위치에서 물리적/논리적 네트워크 영역을 나누어 구현함.
- 일반적으로 VLAN(Virtual LAN)을 기반으로 구현을 함.
- 각 VLAN의 영역(IP대역 및 게이트웨이, 서브넷)은 다름.
- 각 다른 영역의 VLAN을 연결하기 위해서는 OSI 3계층의 라우터 또는 스위치를 이용함.
- End-point 즉, 서버, PC등의 단말기등의 연결을 위해서는 OSI 2계층의 스위치를 이용함.

– 예시(네트워크 구성도)



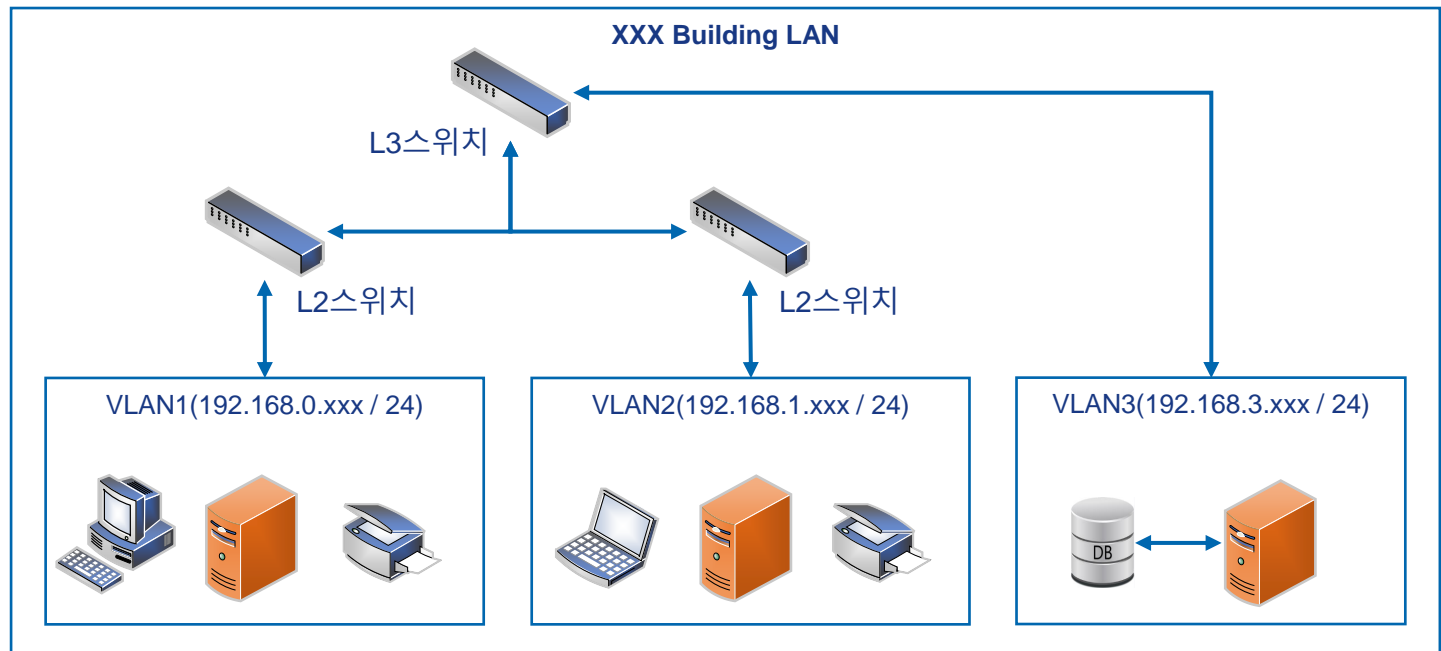
시스템 아키텍처 관련 구성요소(19/36)

• 네트워크

– LAN to LAN 망구조의 시스템

- 15페이지의 네트워크구성도를 시스템아키텍처 측면에서 아래와 같이 표현할 수 있음.
- L3스위치는 VLAN1~3의 세그먼트를 나누어 생성하고 각기 다른 네트워크영역을 할당함.
- L2스위치는 L3스위치로부터 할당받은 VLAN정보를 기반으로 각 시스템에 해당 영역으로 IP를 할당함.
- 만약에 VLAN1의 서버가 VLAN3의 DBMS에 접속하려면 L3스위치에서 VLAN접속 허용을 해줘야 함.

– 예시(Onpremise)



시스템 아키텍처 관련 구성요소(20/36)

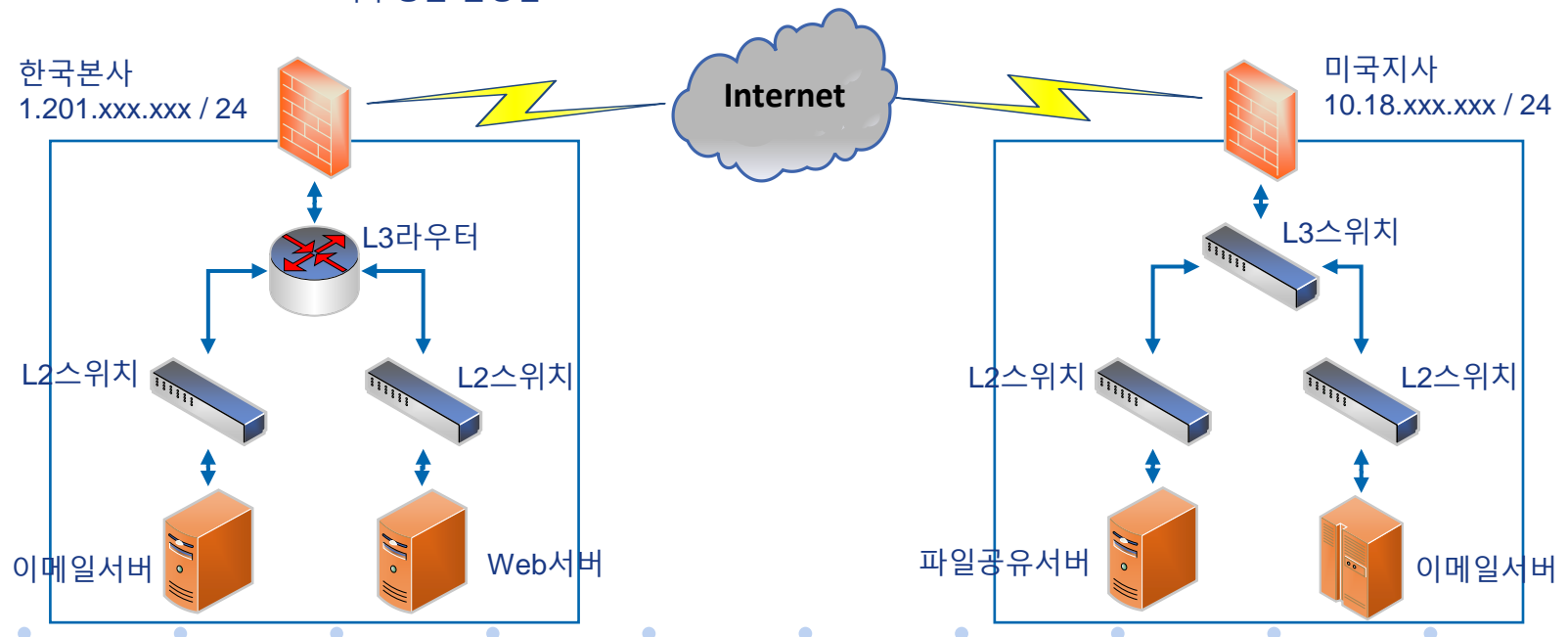
• 네트워크

– WAN to LAN 망구조의 시스템

- WAN은 서로 다른 지역 및 국가를 연결할 수 있는 광대역 네트워크 시스템임. 따라서 서로 다른 네트워크에 접속을 위해 OSI 3계층의 라우팅 기술이 필요함.
- 라우팅기술을 사용할 수 있는 네트워크 장비는 L3백본스위치 또는 라우터가 있음.

– 예시(Onpremise)

- WAN의 인터넷을 통해 한국과 미국간의 네트워크 연결이 됨. 이를 위해 외부와의 보안관리를 위해 반드시 방화벽이 설치되어야 함.
- 한국지사와 미국지사간의 네트워크 연결을 위해서 OSI3계층의 라우터와 L3스위치를 이용하여 IP라우팅을 설정함.

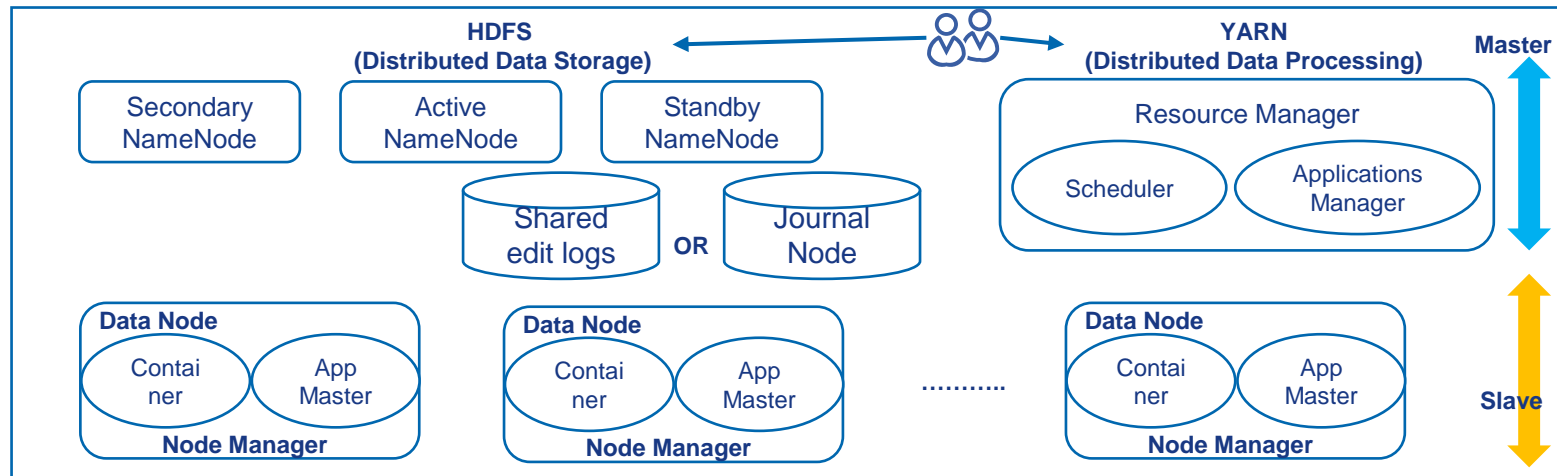


시스템 아키텍처 관련 구성요소(21/36)

- 데이터처리
 - 하둡2.0 (Hadoop 2.0)

개념	하둡 1.0 대비 개선점
기존 Hadoop 1.0시스템의 네임 노드 SPOF(Single Point of Failure) 취약점을 보완한 YARN기반 빅데이터 분산처리 시스템	<ul style="list-style-type: none"> Job Tracker분리 YARN기반 분산처리 확대 Name Node 고가용성 지원

- 하둡2.0의 구성도 및 구성요소

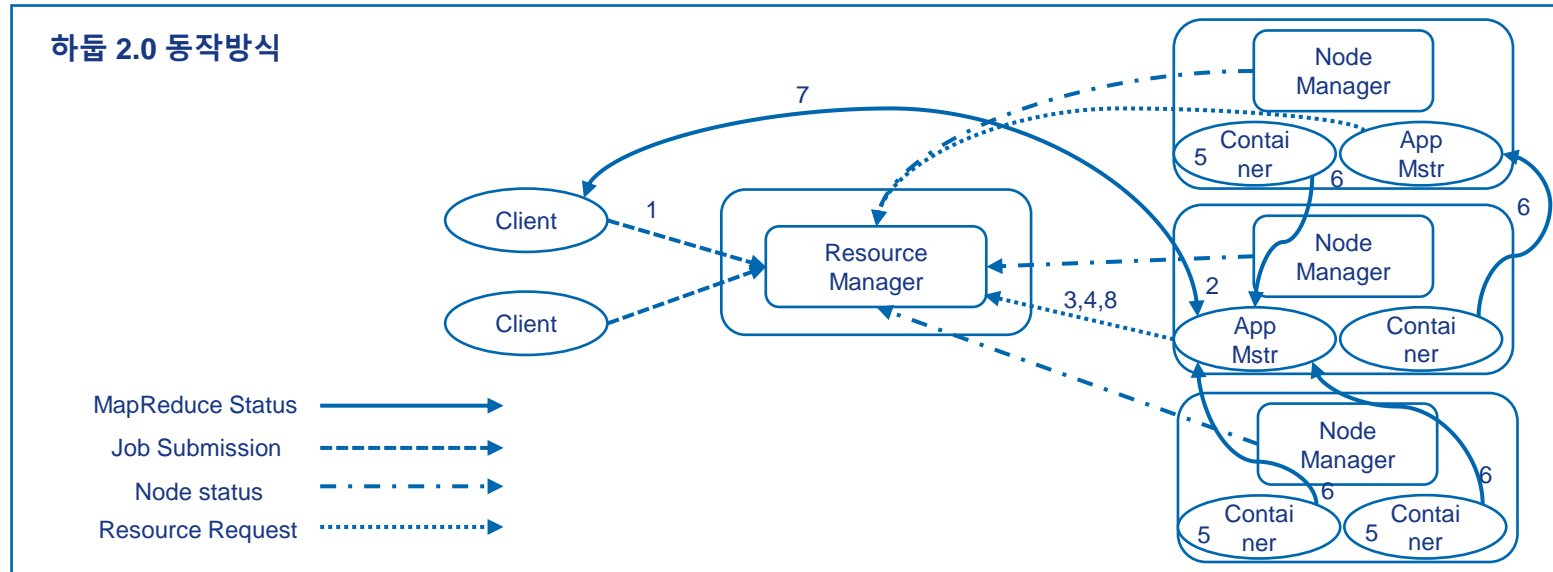


구분	구성요소	설명
Resource manager	Scheduler	- Node Manager 자원관리
	App Manager	- App Maser 실행, 상태관리
	Resource Tracker	- 관리 설정 정보저장
Node Manager	App Master	- 프로그램 마스터 역할
	Container	- CPU, Disk Memory 지원

시스템 아키텍처 관련 구성요소(22/36)

데이터처리

하둡 2.0 동작방식



프로세스	동작
1	- 클라이언트는 필요 데이터 포함 응용프로그램 제출
2	- Resource Manger는 App Master실행
3	- App Master가 Resource Manager에 등록, 클라이언트가 Resource Manager와 통신
4	- App Master는 Container에 적절한 자원요청
5	- Container할당 시 App Master는 Node Master에 실행 스펙제공 및 Container실행
6	- 응용프로그램 코드는 Container에서 실행, 진행률, 상태 등 정보는 App Master에 제공
7	- 클라이언트는 App Master와 상태정보 통신
8	- 프로그램 완료 시 Resource Manager의 등록해제, Container를 다른 용도로 가용하도록 종료

시스템 아키텍처 관련 구성요소(23/36)

- 데이터처리

- 스파크(Apache Spark)

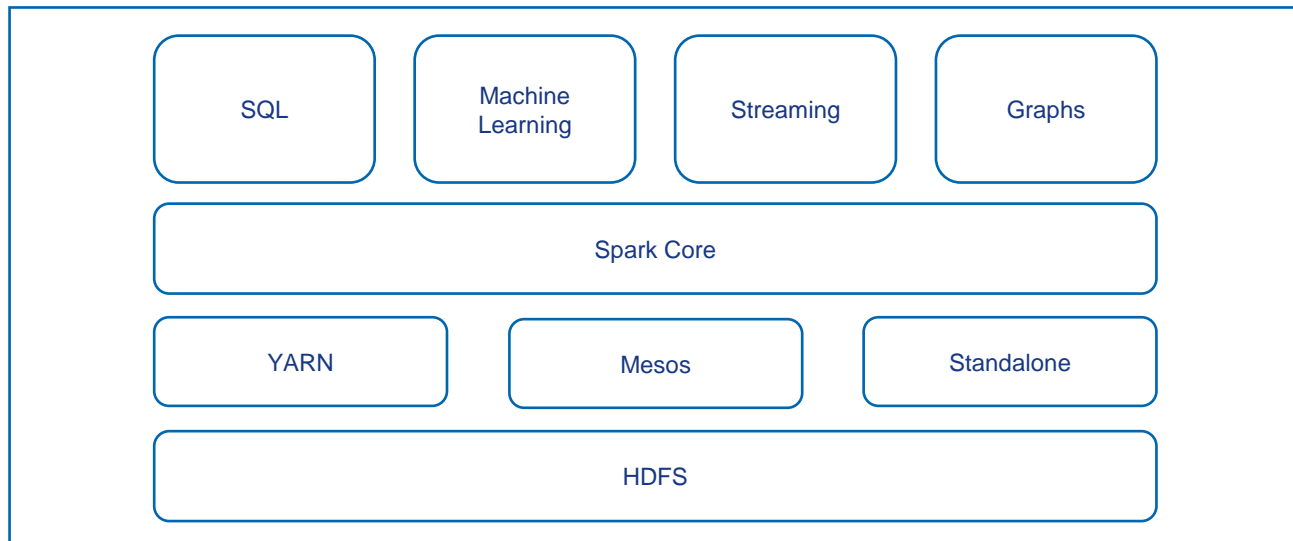
- 디스크 I/O를 효율화하고 데이터 분석작업에 용이한 인메모리 컴퓨팅 기반 데이터 분산처리 시스템

- 특징

구분	설명
HDFS사용	- 하둡의 파일시스템 기반 동작
직관적 이해	- 스칼라기반 최소화코드로 작성
RDD(Resilient Distribute Dataset)	- RDD 단위로 데이터 연산을 수행함.

- 스파크 구성도 및 구성요소

- 스파크 구성도



시스템 아키텍처 관련 구성요소(24/36)

- 데이터처리

- 스파크 구성도 및 구성요소

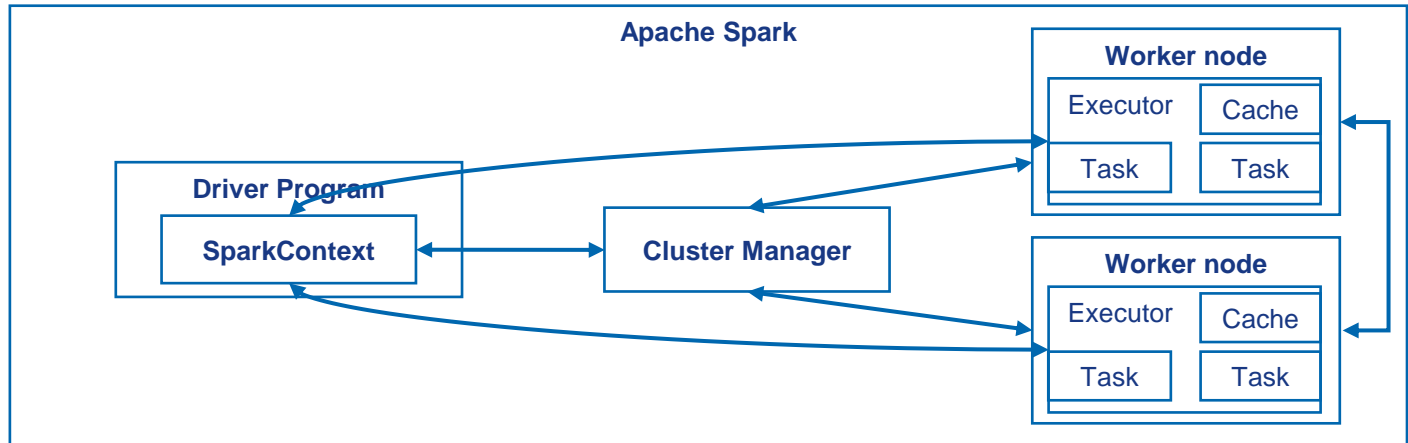
- 23Page의 스파크 구성요소에 대한 설명은 아래 표와 같음.
 - 구성요소

구분	구성요소	설명
구성요소	Spark Core	- 하둡과 같은 일괄처리를 담당하며 다른 프레임워크의 기반으로 되어 있음.
	SQL	- 데이터웨어하우스처럼 SQL에서 인터랙티브 분석가능
	Streaming	- IoT센서 데이터나 SNS데이터등 실시간으로 스트리밍 처리함.
	자원스케줄링	- 자원 스케줄링 기능(YARN)사용 - 스케줄링을 위해 메소스 계층배치
요소기술	RDD (Resilient Distribute Dataset)	- 데이터 내 장애성 보유구조 - 데이터 집합의 추상적객체 개념
	RDD연산자	- RDD에 대란 병렬 데이터 처리 연산지원 및 연산자 제공
	인터랙션	- 함수형 프로그래밍이 가능하도록 Scala를 사용하여 쉘 사용가능함.
	작업스케줄링	- RDD가 변화되는 과정을 그래프로 표현하고 스케줄링함.

시스템 아키텍처 관련 구성요소(25/36)

- 데이터처리

- 스파크 동작방식



- Spark Application : Driver와 Executor의 프로세스로 실행되는 프로그램
- Driver Program : Spark application을 실행하는 프로세스
- SparkContext : Cluster Manager와 연결되는 객체
- Cluster Manager : Spark Application의 리소스들을 효율적으로 배분함.
- Executor : Work node의 Task실행을 담당하는 프로세스

- 스파크 동작순서

- ① Client는 어플리케이션을 제출함.
- ② Driver는 main함수를 실행하고 SparkContext를 생성함. 그리고 SparkContext는 Cluster Manager와 연결됨. 이때 클러스터내부의 Work node에서 Executor 사용준비할 수 있게 됨.
- ③ Driver는 Cluster Manager로부터 Executor실행을 위한 리소스 요청을 함.
- ④ SparkContext는 작업할 내용을 Task단위로 분할하여 Executor로 전달함.
- ⑤ Work node의 각 Executor는 작업을 진행하고 결과를 저장하고 Driver에게 알려줌.

시스템 아키텍처 관련 구성요소(26/36)

• 스토리지

- Application, 문서, 미디어, 사용자관련 기본 설정정보, 네트워크정보등 상세한 디지털정보를 수집하고 보관하는 장소
- 데이터 관리 및 빅데이터의 핵심적인 구성요소라고 할 수 있음.
- 스토리지는 크게 DAS(Direct Attached Storage), NAS(Network Attached Storage) 및 SAN(Storage Area Network) 로 나눌 수 있음.
- 스토리지 유형별 특징

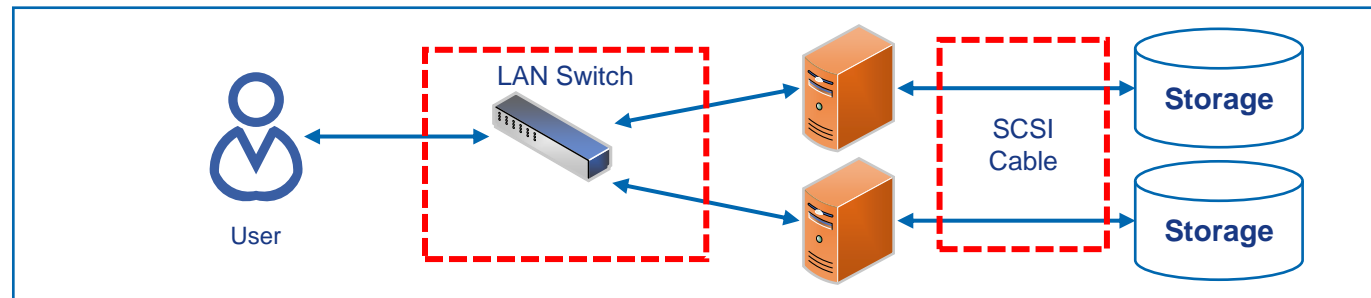
구분	DAS	NAS	SAN
구성요소	Application서버 및 스토리지	Application서버, 전용파일서버 및 스토리지	Application서버 및 스토리지
접속장치	없음	Ethernet Switch	광채널스위치
파일시스템 공유	불가능	가능	불가능
스토리지 공유	가능	가능	가능
파일시스템 관리	Application 서버	파일서버	Application 서버
접속속도영향요인	채널속도	LAN 및 채널속도	채널속도
특징	소규모의 독립된 구성에 적합함.	파일공유를 위한 가장 안정적이고 신뢰성 있는 솔루션	유연성/확장성/편의성등이 가장 좋음.
스토리지 저장단위	블록	파일	블록
장점	서버와 직접연결로 쉽게 용량확장이 가능함.	네트워크를 통해 데이터를 공유하므로 여러장치들을 쉽게 연결할 수 있음.	광케이블을 통해 속도가 빠름. 또한 용량을 위한 확장이 용이함.
단점	시스템과 파일공유가 불가능하고 데이터크기가 커지면 효율성이 떨어짐.	DAS에 비해 용량확장제한이 있음. 또한 네트워크상황에 따라 병목현상이 발생할 수 있음.	가격이 상대적으로 비쌈. 광채널 네트워크를 사용하기 때문에 운영이 힘들.

시스템 아키텍처 관련 구성요소(27/36)

- 스토리지

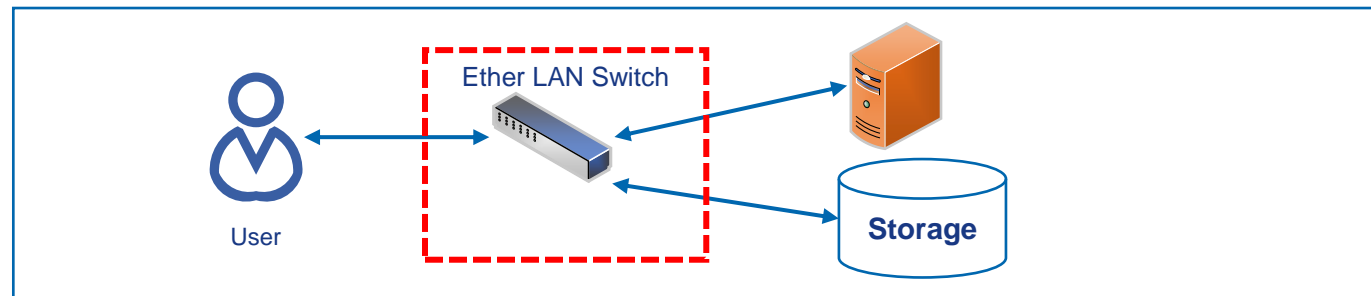
- DAS 구성도

- 서버에 SCSI컨트롤러등의 카드로 직접 연결이 됨. 흔히 우리가 사용하는 PC 및 서버에 설치되는 하드디스크를 의미함.



- NAS 구성도

- 네트워크시스템 기반으로 연결된 큰 공유 스토리지
 - 이기종 플랫폼의 스토리지 풀들을 중장 집중화 하여 공유할 수 있음.
 - 네트워크 프로토콜에 따른 프로세스 과부하가능, 스토리지 I/O에 따른 LAN대역폭 소모증가가 생길 수 있음. 일반적으로 DAS에 비해 속도가 느림.

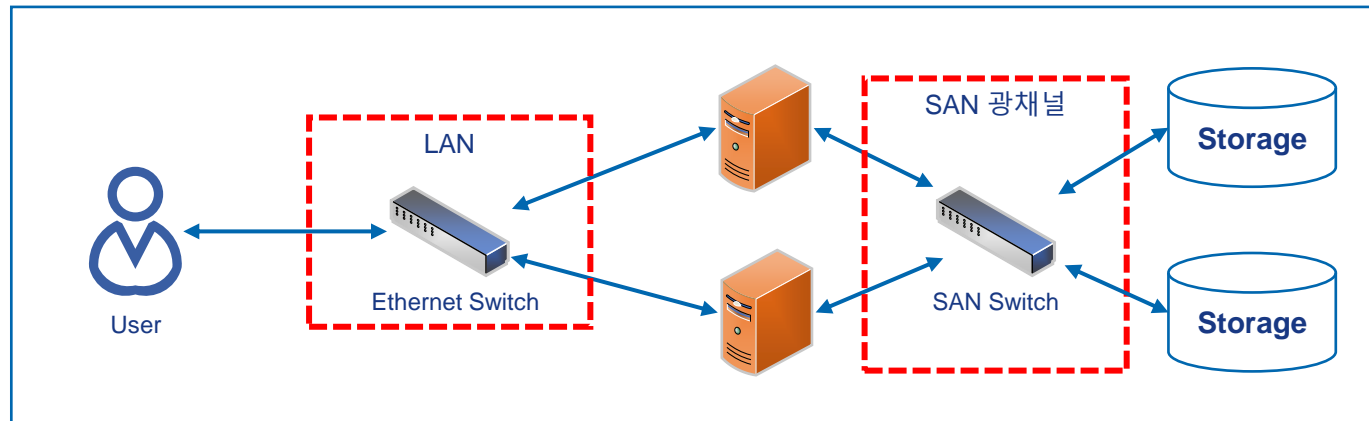


시스템 아키텍처 관련 구성요소(28/36)

- 스토리지

- SAN 구성도

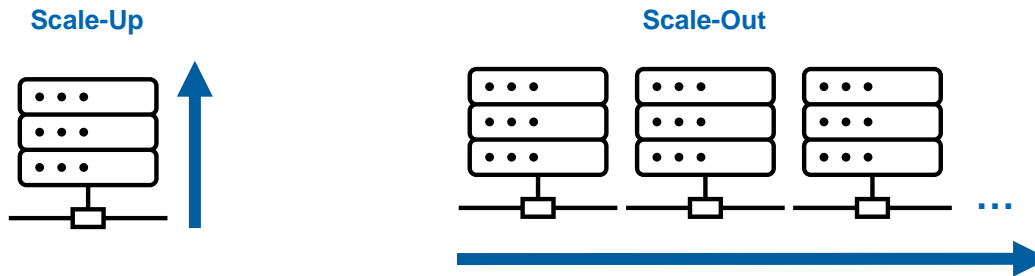
- DAS/NAS의 단점들을 극복하기 위해 발전된 스토리지 유형임.
 - 서버와 스토리지간의 연결을 물리적 네트워크 및 스위치로 직접 연결함.
 - 스트리지와 서버는 SAN전용 광채널 네트워크로 연결, 서버와 사용자는 LAN으로 연결됨. 따라서 속도가 매우 빠름.



시스템 아키텍처 관련 구성요소(29/36)

- 부하분산(Load Balancing)

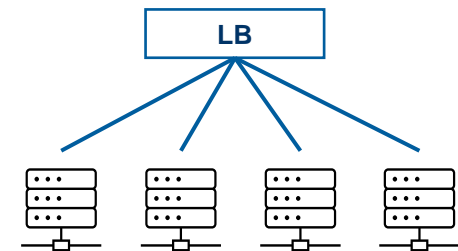
- 네트워크 상에서 한곳에 가중되고 있는 부하를 전체적으로 분산시키는 기술
- 저장장치와 같은 컴퓨터자원들에게 작업을 나누게 하는 것
- 크게 Scale-UP, Scale-Out방법이 있음.



- 로드밸런싱은 OSI계층 중 4계층과 7계층에 속함. 즉 TCP/UDP 프로토콜 또는 HTTP, HTTPS등의 Application단의 프로토콜을 기반으로 부하부산을 시킴.
- 로드분산을 위해서 네트워크장비로는 L4 및 L7스위치가 있음. 클라우드에서는 대표적으로 AWS의 ELB(Elastic Load Balancer)와 HAProxy를 사용함.



L4/L7 Switch (Hardware), AWS ELB/HAProxy(Software)



시스템 아키텍처 관련 구성요소(30/36)

• 부하분산(Load Balancing)

- 부하분산을 위한 알고리즘 방식

- 크게 5가지의 알고리즘이 존재함. 일반적으로 가장 많이 사용하는 라운드로빈방식이 있음.
- 시스템의 목적에 맞게 부하분산을 적합하게 해줘야 함.

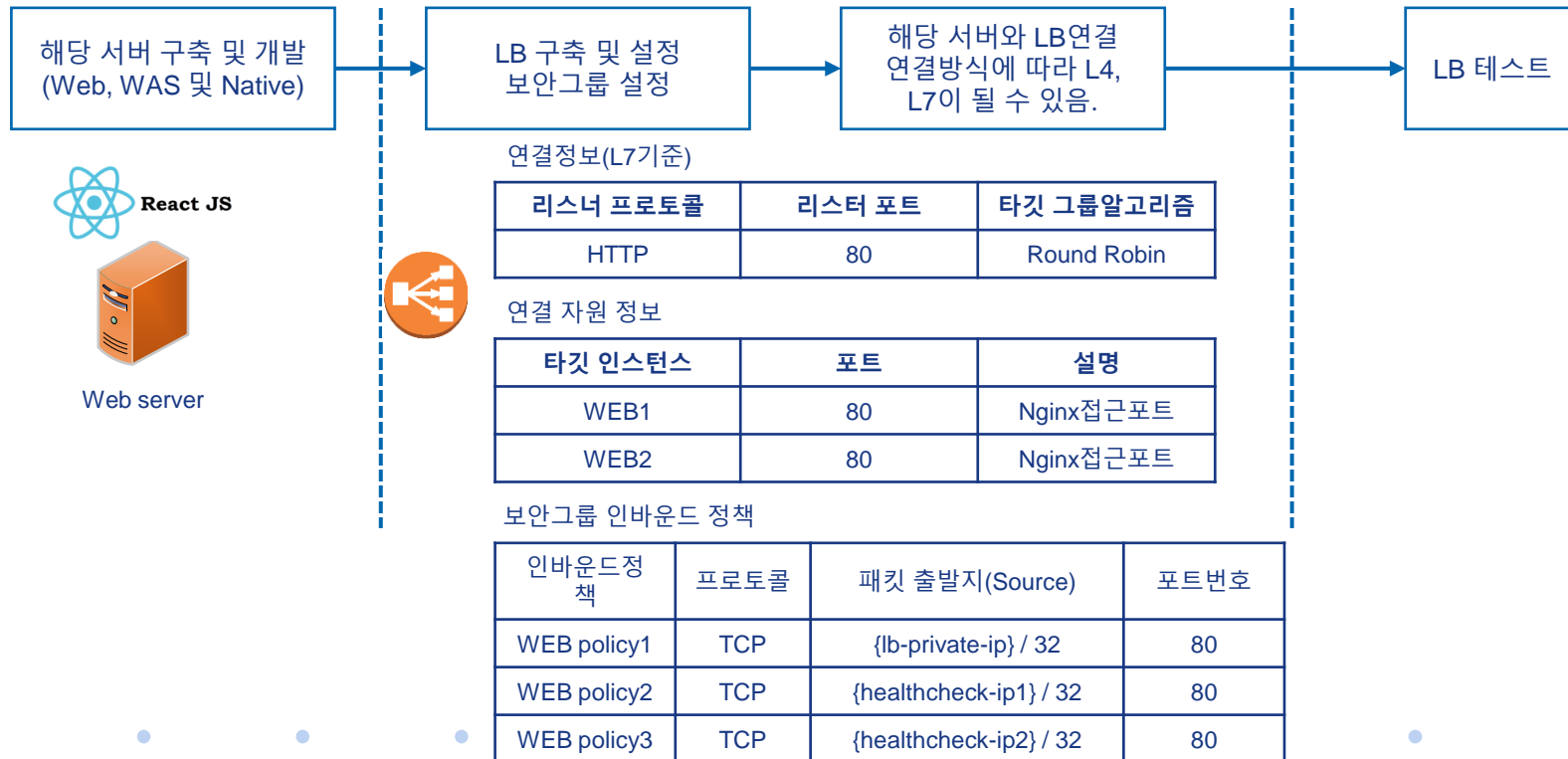
구분	설명
라운드로빈	- 서버에 들어온 요청을 순서대로 배정하는 방식 - Client의 요청을 순서대로 분배하기 때문에 여러대의 서버가 동일한 스펙을 가지고 있음. - 서버와의 연결이 오래 지속되지 않는 경우에 적합
가중 라운드로빈	- 각각 서버마다 가중치를 주고 가중치가 높은 것부터 우선적으로 배분하는 방식 - 서버들의 트래픽 처리 능력이 상이한 경우 사용되는 부하 방식
IP해시 방식	- Client의 IP주소를 특정서버로 매핑하여 요청을 처리하는 방식 - IP를 해싱하고 로드를 분산하기 때문에 사용자가 항상 동일한 서버로 연결되는 것이 보장됨.
최소연결방식	- Client로부터 요청이 들어온 시점에 가장 적은 연결상태를 보이는 서버에 우선적으로 트래픽을 분산하는 방식 - 연결 세션이 길어지거나 분배된 트래픽이 일정하지 않은 경우에 적합함.
최소 응답시간	- 가장 적은 연결상태와 짧은 응답시간을 보이는 서버에 우선적으로 로드를 배분하는 방식 - 서버의 현재 연결 상태와 응답시간을 모두 고려하여 트래픽을 분산함.

시스템 아키텍처 관련 구성요소(31/36)

부하분산(Load Balancing)

예시

- 퍼블릭 클라우드상에서는 기본적으로 로드밸런싱 서비스를 제공함.
- Onpremise상에서는 별도의 L4 및 L7의 하드웨어 스위치를 설치함. 아니면 소프트웨어적으로 HAProxy 및 Nginx를 이용할 수 있음.
- 아래와 같은 절차로 시스템 구축을 하고 로드밸런서를 설치하여 해당서버의 트래픽을 분산시켜줘야 함.



시스템 아키텍처 관련 구성요소(32/36)

• 부하분산(Load Balancing)

– 예시

- 로드밸런싱으로 다른 솔루션은 Nginx의 로드밸런싱 기능을 사용할 수 있음.
- Nginx를 프론트엔드 서버에 설치 후 nginx.conf파일에서 아래와 같이 설정을 함. Default LB알고리즘은 라운드로빈으로 되어 있음.

vi /etc/nginx/nginx.conf 실행

```
root@localhost:~
```

```
root@012873f3a301:/# vi /etc/nginx/nginx.conf
```

```
http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format main '$remote_addr - $remote_user [$time_local] "$request"
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log    /var/log/nginx/access.log  main;

    sendfile      on;
    #tcp_nopush   on;

    keepalive_timeout 65;

    #gzip on;

    #include /etc/nginx/conf.d/*.conf;
    upstream tomcat {
        server 192.168.0.60:10006 max_fails=3 fail_timeout=3s;
        server 192.168.0.60:10008 max_fails=3 fail_timeout=3s;
    }

    server {
        listen 80;
        listen [::]:80;
        server_name localhost;
        location / {
            root /usr/share/nginx/html;
            index index.html index.htm;
            proxy_pass http://tomcat;
            proxy_set_header Host $http_host;
        }
    }
}
```

주석처리

Nginx loadbalancing 설정

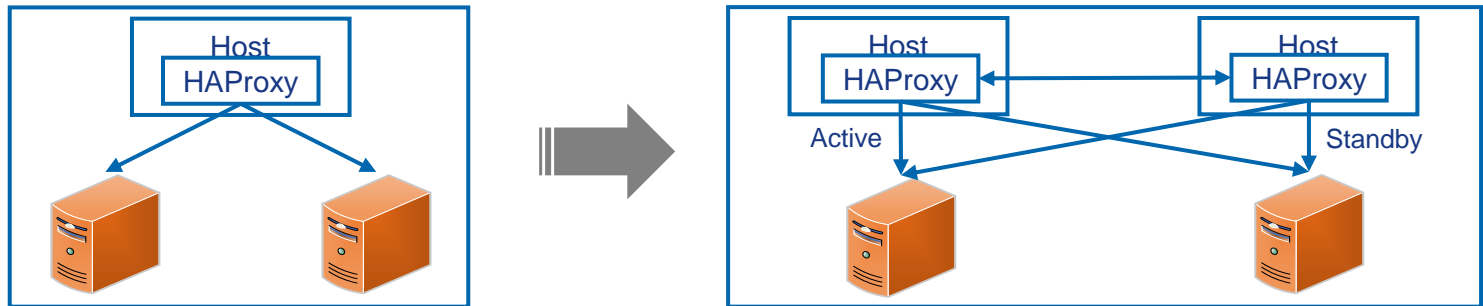
내용추가

2대의 웹서버정보가 있는 것을 확인할 수 있음. 최대한 3번의 접속실패와 3초의 연결실패가 발생할 시 정상 작동하는 서버로 트래픽을 보내는 헬스체크 기능을 제공하는 것을 볼 수 있음.

시스템 아키텍처 관련 구성요소(33/36)

고 가용성(High Availability)

- 정보시스템이 지속적인 비즈니스를 위해 허용되는 긴 시간동안 운영이 가능하게 하는 시스템이나 컴포넌트를 의미함.
- 일반적인 100%의 가용성을 기준으로 “절대 다운 및 고장나지 않는다.”와 같이 표현할 수 있지만 결코 만족하기 힘든 기준임. 따라서 “파이브 나인(Five 9)”이라고 99.999%라고 표현을 많이 함.
- 정보시스템은 수 많은 컴퓨팅기술 및 네트워크 시스템들로 구성되고 운영되기 때문에 고 가용성을 위해서는 계획, 백업, 장애극복 및 데이터 백업등에 집중되어 있음.
- 목적
 - 비즈니스의 서비스의 다운타임을 최소화하여 정보시스템의 가용성을 극대화 하는 것임.
- 방법
 - 클러스터링의 컨셉으로 이중화(Redundancy)가 기본임. 따라서 최소 2대의 시스템이 필요함.
 - 크게 Active-Standby 와 Active-Active 구간의 구성방법이 있음.
 - 아래 그림처럼 하나의 접점 SPOF(Single Point of Failure)구조에서는 시스템 다운에 리스크를 가지고 있음. 따라서 Active서버시스템이 다운된다면 자동으로 Standby의 백업 서버 시스템이 작동되도록 하는 것이 현재 사용하고 있는 기본적인 고 가용성의 방법임.



시스템 아키텍처 관련 구성요소(34/36)

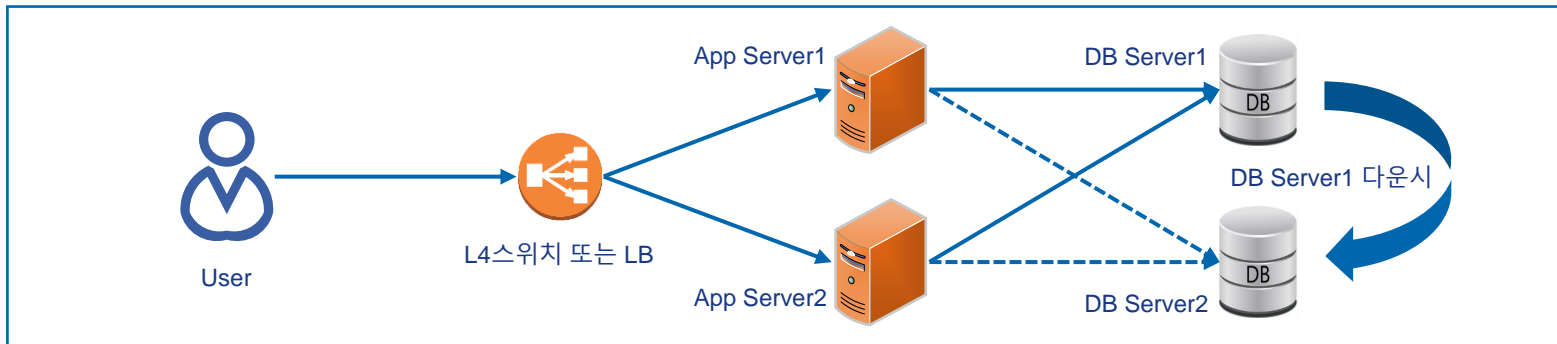
고 가용성(High Availability)

Active-Active / Active-Standby의 비교

구분	Active-Active	Active-Standby
동작방식	동시에 서로 교대로 실행	마스터-슬레이브 구성으로 동작
다운타임	없음	Standby로 넘어가는 다운타임 발생
부하관련	부하분산	부하집중
서비스 응답시간	응답시간빠름	응답시간지연
동기화	아르싱크(Rsync) 방식, 동기화 지연없음	Tar방식, 동기화 지연발생
서비스 연속성	서비스 연속성 좋음	서비스 연속성 보통

Active-Active 구성

- 로드밸런싱 기능을 가진 L4스위치등을 통해 1번 2번서버로 나누어 처리되도록 구성함.
- 앱서버 뒤에는 무중단시스템 구조를 위해 DB서버도 2개로 이중화를 해야 함.
- 일반적으로 DB서버는 Replica설정을 하여 DB서버2도 실시간 복제되게끔 구동 시킴.

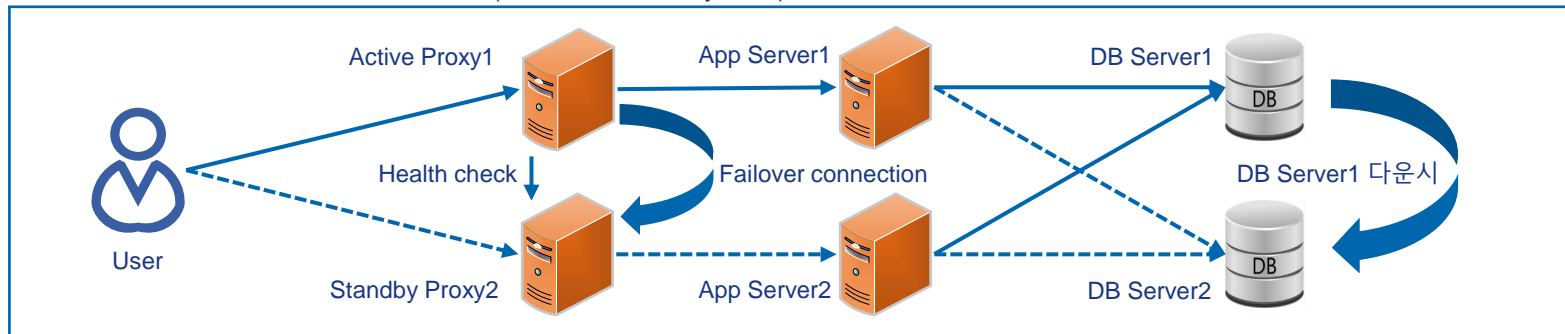


시스템 아키텍처 관련 구성요소(35/36)

- 고 가용성(High Availability)

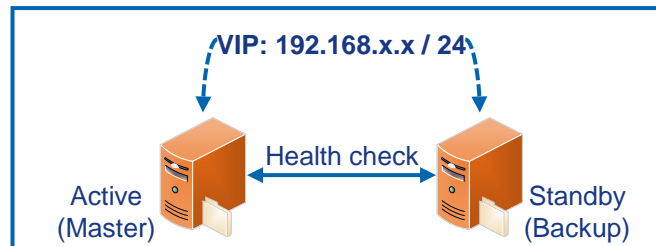
- Active-Standby 구성

- Active-Active구성과 다르게 부하분산이 아닌 장애발생 시 Standby시스템으로 서비스를 넘기는 Failover개념임.
- Failover작동이 되려면 주기적으로 Active 서버시스템의 작동 및 연결에 대한 Health check가 필요함. 이러한 작업을 Heart beat라고 함.
- 실시간으로 99.999%의 무중단은 아니지만 비즈니스의 지속성을 위한 다운타임 허용기준에 반드시 충족해야 함. 이 부분은 DRP(Disaster Recovery Plan)과 연결됨.



- 예시(HAProxy & Keepalived)

- 일반적으로 시스템 이중화를 하기 위해 HAProxy & Keepalived솔루션을 많이 사용함.
- 가상IP(VIP)를 기반으로 여러노드간의 모니터링을 하고 노드의 장애발생 시 백업노드로 Failover되도록 하는 이중화 지원 소프트웨어임.

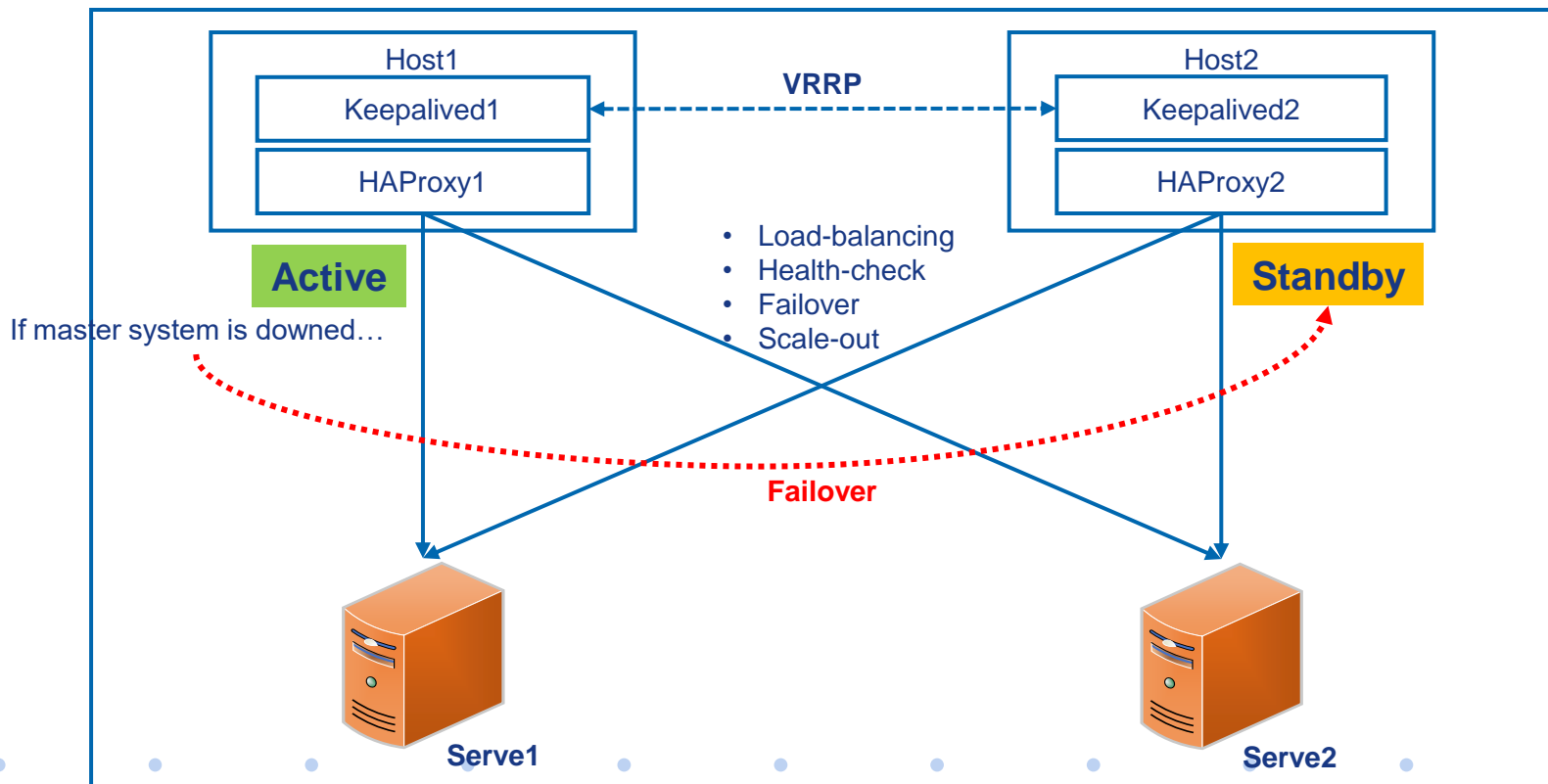


시스템 아키텍처 관련 구성요소(36/36)

고 가용성(High Availability)

– 예시: (HAProxy & Keepalived)를 이용한 이중화 아키텍처

- HAProxy & Keepalived 시스템은 프론트엔드 또는 백엔드의 서버앞에 위치할 수 있으며 VRRP(Virtual Router Redundancy Protocol)을 이용하여 이중화를 구성할 수 있음.
- 아래 그림처럼 Server1에 문제 발생 시 자동으로 Server2로 넘어가게끔 구성되어 있음. 이를 위해 VRRP는 VIP라는 가상의 게이트웨이를 설정하여 수시로 서버와의 상태를 체크함.



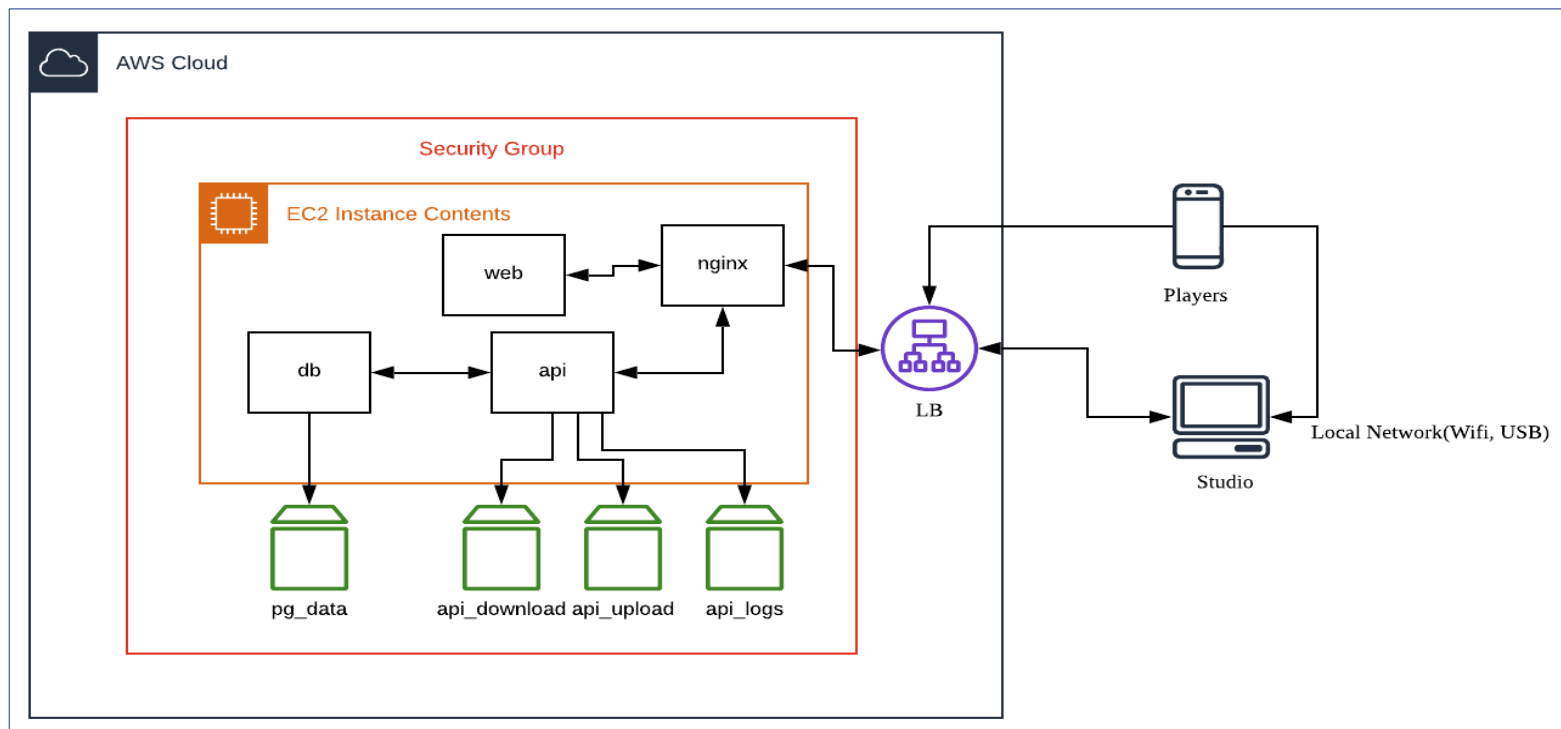
실무적사례기반 시스템 아키텍처(1/9)

- SaaS기반 프로토타이핑 솔루션

- XXX사의 프로토타이핑 소프트웨어 개발을 위한 시스템 아키텍처

- 북미중심으로 비즈니스를 전개 시키고 있는 Startup업체로서 Figma와 경쟁을 하고 UI/UX디자인 툴을 개발하는 솔루션업체임.
 - 그림을 보면 AWS기반의 private cloud로 native솔루션이 구축되어 있음. 외부에서 보안접속이 되도록 Security Group으로 허용된 트래픽만 통과 되게끔 했음.
 - 각 서버들은 AWS의 EC2에 생성되었음. 프론트엔드에서는 Nginx / web서버를 구축했고 백엔드의 API서버가 Nginx서버와 연동되게끔 구성되어 있음.
 - 백엔드의 DB와 각 로그파일들의 관리를 위해 스토리지들이 설치되어 있는 것을 볼 수 있음.

- 기능중심 아키텍처

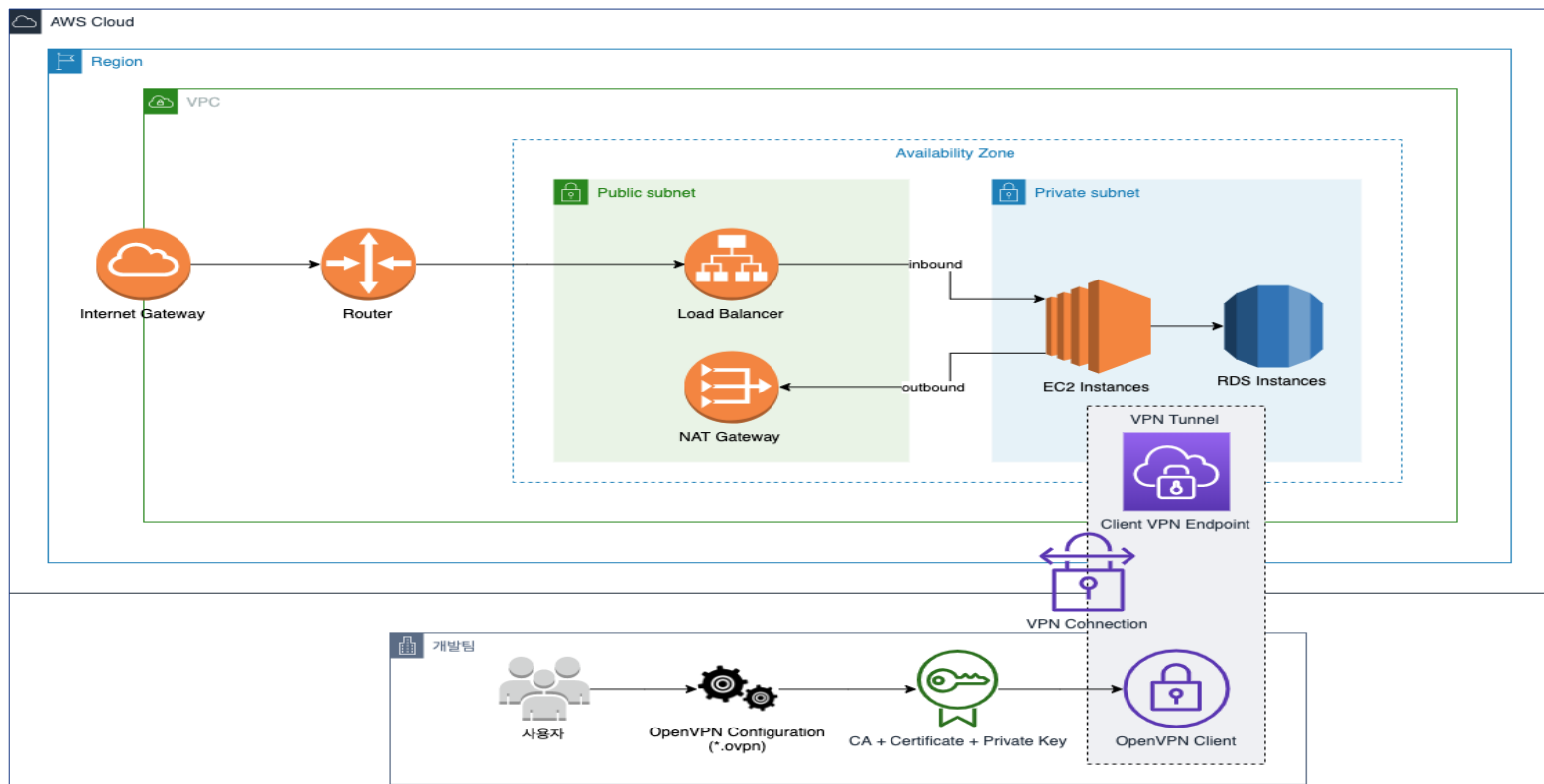


실무적사례기반 시스템 아키텍처(2/9)

• SaaS기반 프로토타이핑 솔루션

– AWS Component기반 아키텍처

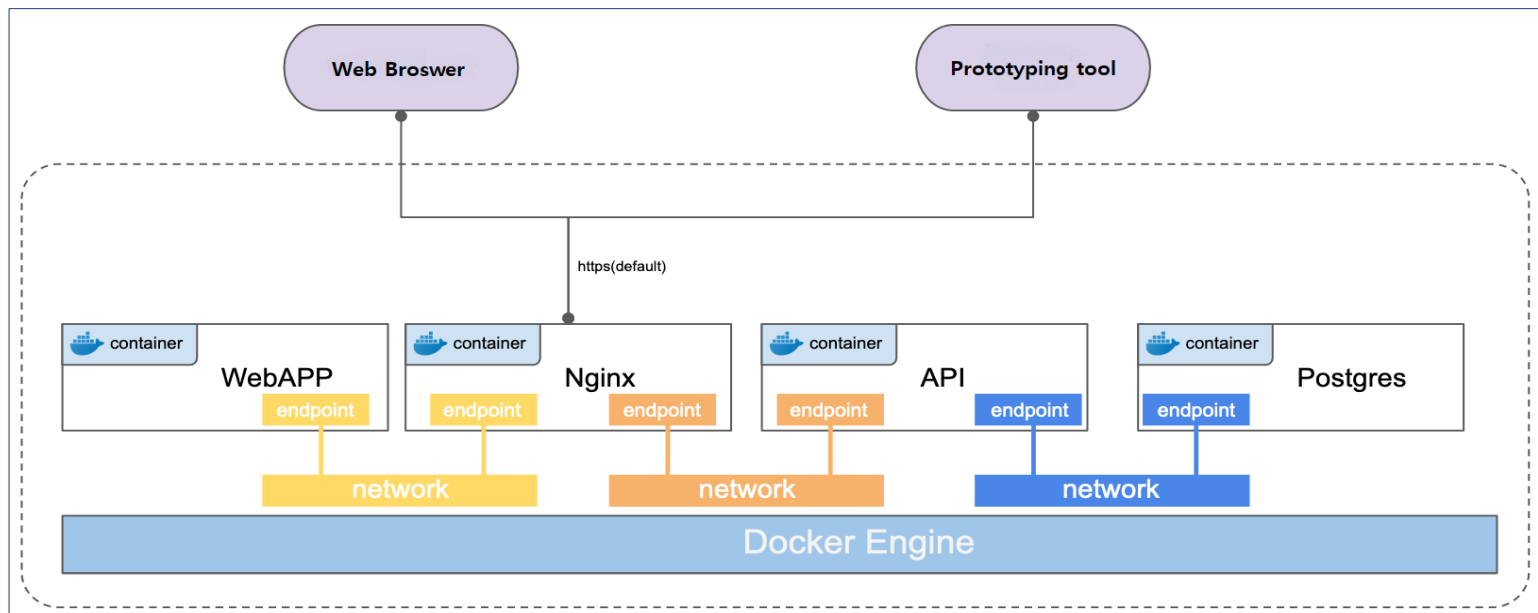
- 기본적으로 AWS클라우드기반의 Private native 아키텍처임.
- 외부에서 사용자들은 인터넷 게이트웨이를 통해 접속을 할 수 있으며 OSI 3계층의 라우터를 통해 해당 목적지인 Public subnet을 찾아 접속을 할 수 있음.
- 사용자들은 토큰을 통해 Private subnet에서 구동되는 시스템들을 LB & NAT Gateway를 통해 보안을 유지하고 접속할 수 있음.
- 개발팀들도 정보보안을 위해 허용된 VPN세션으로만 터널링을 생성하여 접속될 수 있게끔 함.



실무적사례기반 시스템 아키텍처(3/9)

- SaaS기반 프로토타이핑 솔루션

- 기본적으로 AWS EC2에 Docker 컨테이너 기술을 이용하여 구축 및 개발을 함.
- 아래 그림을 보면 크게 프론트엔드의 Nginx 웹서버, 백엔드의 WebAPP서버, API 및 Postgres DB서버가 존재하는 것을 볼 수 있음.
- 웹기반의 시스템은 일반적으로 3티어의 구조를 가지고 있음. 또한 API시스템을 구축하기 위해 Native구조로 연결이 되어 있는 것을 볼 수 있음.
- 모든 기능별 시스템들은 endpoint들이 단계별로 접속될 수 있게끔 분리된 네트워크로 연결됨. 사용자들은 외부에서 웹서버인 Nginx에 처음으로 접속되는 것을 볼 수 있음.
- 실무에서 도커의 활용이 절대적으로 많다는 것을 나타내고 있음.



실무적사례기반 시스템 아키텍처(4/9)

- 시스템반도체 기업 웹사이트

- XXX사의 시스템 반도체 개발업체의 웹시스템 아키텍처
 - 프로젝트 개발 언어 및 아키텍처

구분	개발환경 및 언어
프론트엔드	HTML, Javascript(React.JS)
백엔드	RESTAPI, JAVA/SpringBoot, MariaDB
인프라환경	AWS 클라우드 서비스



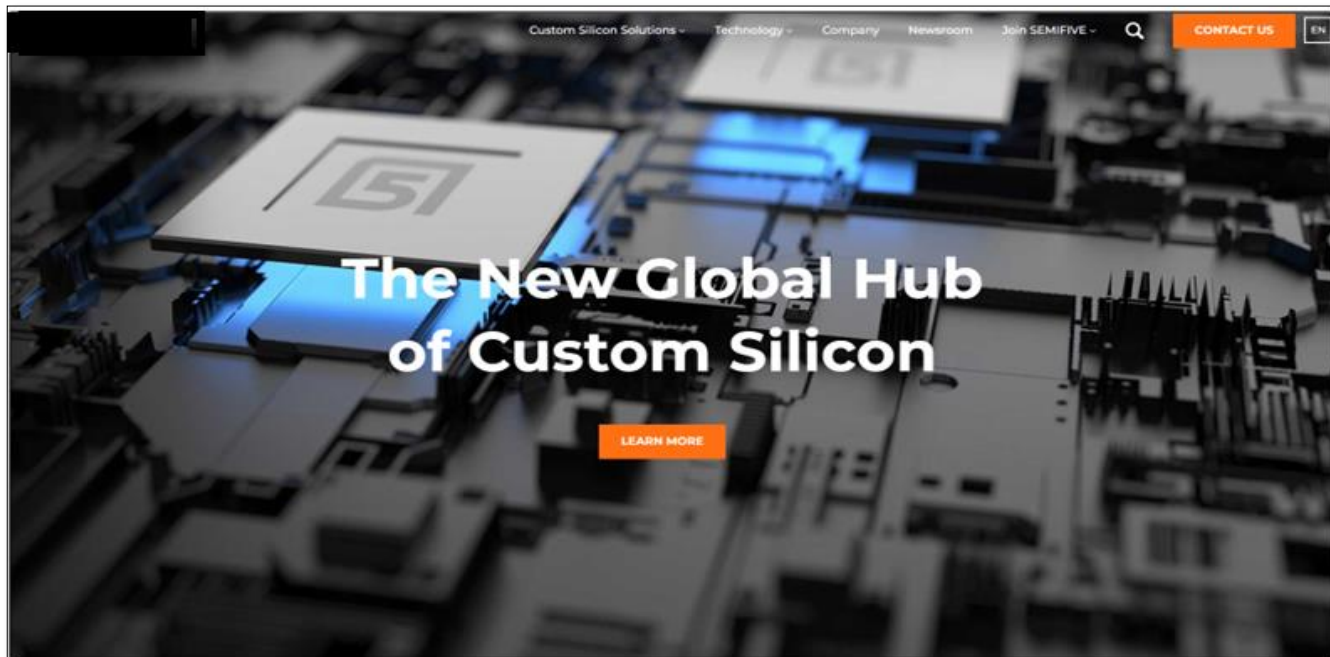
React JS



Spring Boot



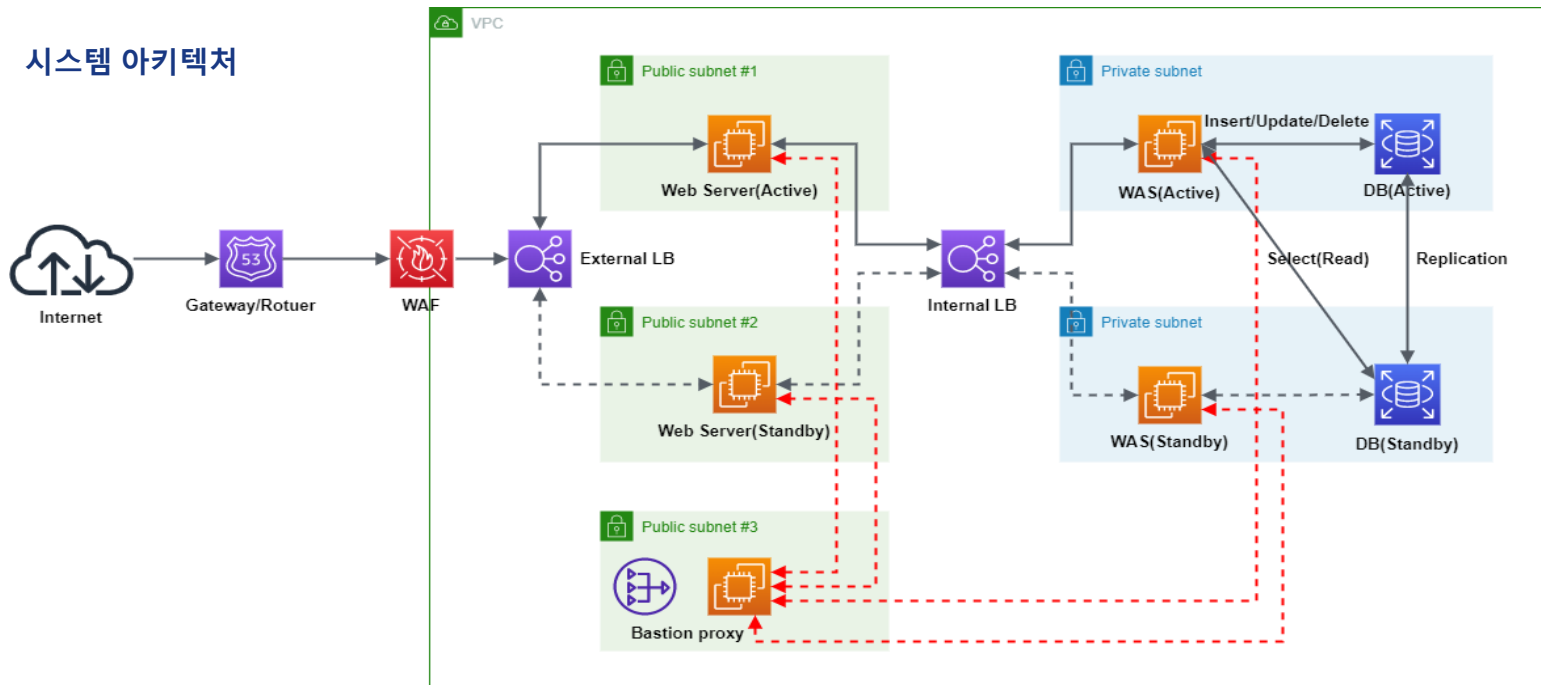
- 웹사이트 화면(Index page)



실무적사례기반 시스템 아키텍처(5/9)

• 시스템반도체 기업 웹사이트

- 아키텍처패턴 중 티어패턴의 3티어 아키텍처기반의 솔루션임.
- 아래그림에서 보면 외부에서 사용자들은 인터넷을 통해 해당 사이트의 DNS접속을 위해 Router에 접속하게 됨. 웹솔루션의 보안을 위해 외부/내부의 통제를 담당하는 웹방화벽(WAF)을 거치게 되고 누구나 접속을 할 수 있는 Public subnet에 접근함.
- Public zone에서는 프론트엔드의 웹서버가 작동됨. 이시스템들의 부하를 분산시키기 위해 외부 로드밸런서가 작동이 되는 것을 볼 수 있음. 또한 개발 및 운영팀들이 보안접속을 위해 Bation proxy가 구축되었음.
- Private zone은 백엔드 시스템(WAS & DB)가 개발 및 구축되어 있음. 이들의 부하부산을 위해 내부 로드밸런서가 설치 및 구동되고 있는 것을 볼 수 있음.
- 모든 시스템의 고 가용성을 위해 Active-Standby로 구성되어 있음. 백엔드의 DB서버는 Replica 구성 및 설정을 통해 백업 및 Failover의 효과를 볼 수 있음.



실무적사례기반 시스템 아키텍처(6/9)

• 글로벌제약회사의 임상연구 정보관리시스템

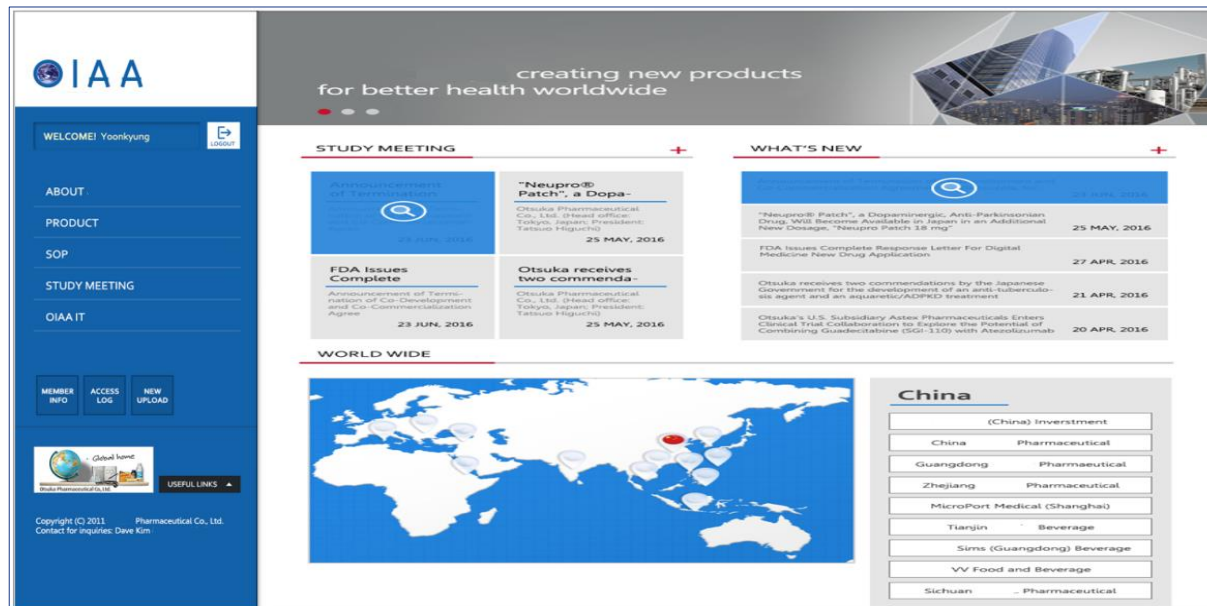
– XXX제약회사의 시스템 아키텍처

- 웹기반의 시스템으로 동남아시아의 임상연구 데이터를 통합관리 및 공유하는 시스템임.
- 프로젝트 개발 언어 및 아키텍처

구분	개발환경 및 언어
프론트엔드	HTML, CSS, Javascript
백엔드	JAVA/Spring, MySQL DB
인프라환경	On-premise 클라우드 시스템



– 웹사이트 화면(Index page)

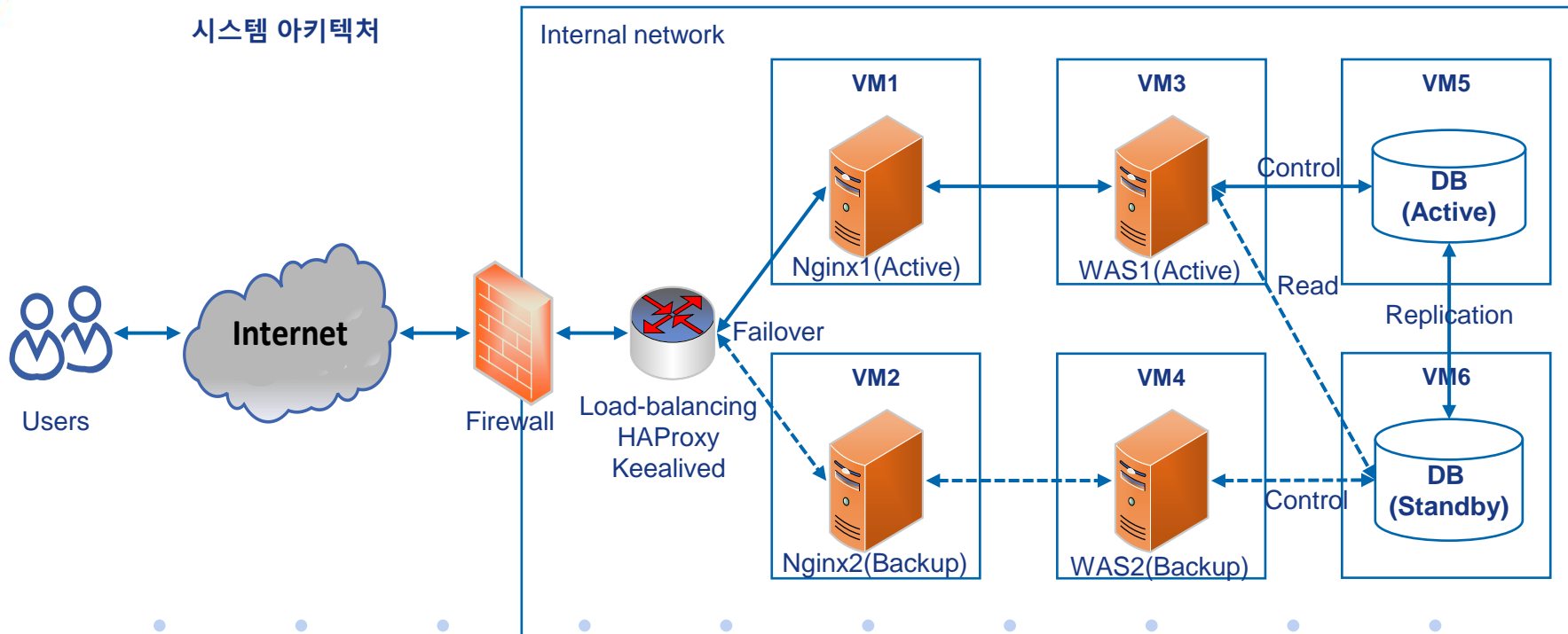


실무적사례기반 시스템 아키텍처(7/9)

• 시스템반도체 기업 웹사이트

- 아키텍처패턴 중 티어패턴의 3티어 아키텍처기반의 솔루션임. 대부분의 웹기반의 시스템은 2~3티어로 개발 됨.
- 그림의 아키텍처에서는 총 6개의 VM을 생성하여 구축되어 있는 것을 볼 수 있음.
- 외부 사용자들은 인터넷을 통해 접속이 가능하며 인프라 보안을 위해 웹방화벽이 외부/내부 네트워크망의 필터링을 담당함. 다시 말해, 허용된 포트와 트래픽만 통과하게끔 하여 외부의 공격을 막을 수 있음.
- 내부에서는 부하분산을 위해 HAProxy 로드밸런서가 설치되었음. 또한 고 가용성을 위해 Keepalived가 설치되어 Active의 시스템들에서 문제가 생기면 바로 Backup시스템들이 구동 될 수 있도록 구성되어 있음.
- 백엔드의 DB도 부하분산을 위해 Insert/Update/Delete는 Active DB에서, Read는 Standby DB에서 구동되도록 설계됨.
- DB또한 Replica설정을 통해 실시간으로 Standby로 백업 되게끔 하여 DR(Disaster Recovery) 요구사항에 맞게끔 함.

시스템 아키텍처

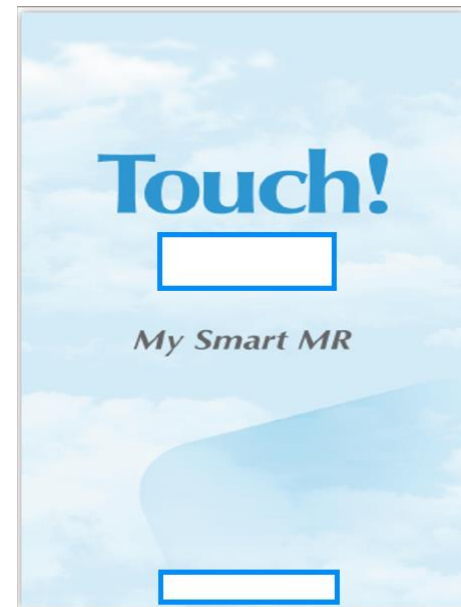
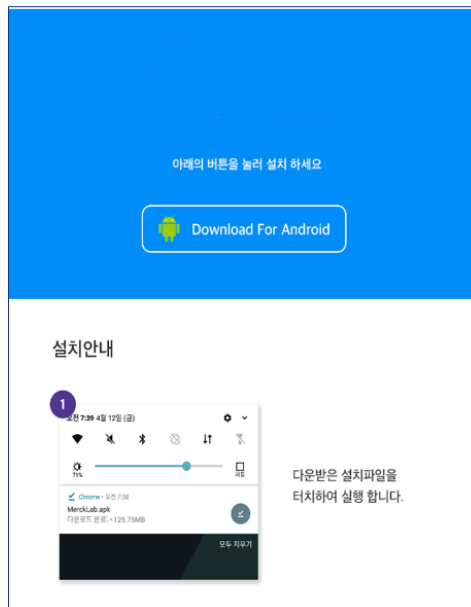


실무적사례기반 시스템 아키텍처(8/9)

- **Native아키텍처 기반 설문조사 시스템**
 - XXX사의 Android/IOS 플랫폼 기반 설문조사 시스템
 - 프로젝트 개발 언어 및 아키텍처

구분	개발환경 및 언어
프론트엔드	HTML, CSS, Javascript
백엔드	JAVA, MariaDB
인프라환경	AWS 클라우드 서비스

- 시작 화면(Index page)



실무적사례기반 시스템 아키텍처(9/9)

• Native아키텍처 기반 설문조사 시스템

- 아키텍처패턴 중 티어패턴에 속하며, 2티어로 구성되어 있음.
- 외부의 사용자들은 Gateway / DNS서버서비스를 통해 도메인주소로 목적지로 찾아갈 수 있음.
- 외부에서의 보안공격 및 이슈를 관리하기 위해 방화벽이 설치되어 있음.
- Public zone은 외부에서 방화벽을 통과한 사용자들이 접속을 할 수 있으며 프론트 엔드에 개발된 인증서버가 설치되어 있음. 인증서버는 백엔드의 Native application서버에 접속될 수 있게끔 토큰기반으로 인증처리를 담당함.
- 보안 및 사설IP접속을 위해 Public zone에서는 NAT라우팅을 하여 Private zone으로 접속할 수 있게끔 함.
- 백엔드의 DB서버는 이중화가 되어 있으며 DR(Disaster Recovery)기반으로 백업구성이 되어 있음.

시스템 아키텍처

