



Software Architecture Design

Part 2-System Architecture Design

Jin Hyun Kim
Dept. of Information and Communication
Gyeongsang Univ.
jin-kim@gnu.ac.kr
<http://jin-kim.net>

내용

- 시스템 아키텍처 설계
 - 아키텍처 설계 원칙
 - 애플리케이션 아키텍처
 - 테크니컬(인프라) 아키텍처
 - 솔루션 아키텍처
 - 데이터 아키텍처

References



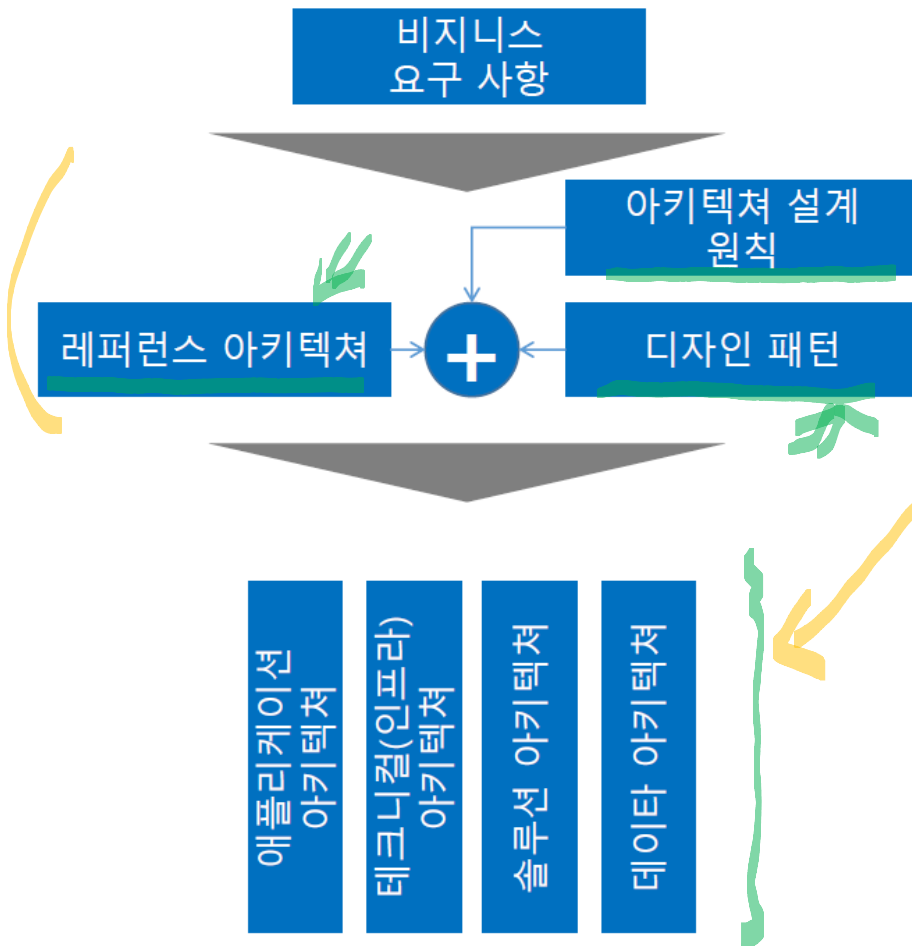
대용량 아키텍처 설계

아키텍처 설계 프로세스

Terry Cho

gnu

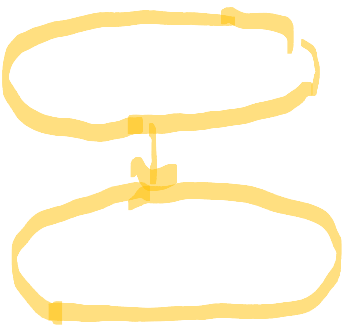
시스템 아키텍처의 구성



- 아키텍처 설계 원칙
- 애플리케이션 아키텍처
- 테크니컬(인프라) 아키텍처
- 솔루션 아키텍처
- 데이터 아키텍처

아키텍처 설계 원칙 (Architecture principals)

- 아키텍처 설계의 원칙
- 비용, 비기능적인 설계 원칙
- 디자인 의사 결정이 필요할 때 의사 결정의 기준이 됨
- 7~15개 정도가 적절



개인 스토리지 서비스 Architecture Principals

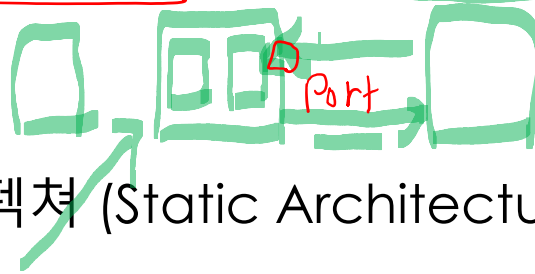
- 퍼블릭 클라우드 (아마존, MS)에 종속성이 없으며, 기업내(On-Prem) 배포가 가능해야 한다.
- 미국 정부 수준의 보안 수준을 충족해야 한다.
- 글로벌 서비스를 충족해야 한다
- 모바일, PC 등 멀티 디바이스를 지원해야 한다.
- 사용자 인터페이스가 사용하기 쉬워야 한다.

:

아키텍처 설계시 주의 사항

- 아키텍처 문서를 만들기 위해서 설계를 하는게 아니라 소통하기 위해서 하는 것이 아키텍처 설계
- 최종 아키텍처라는 것은 없다. 계속해서 진화 한다.
(Evolutionary architecture : 진화적 아키텍처)
- 오버 디자인 주의 (나중에 비즈니스가 잘되면 그때 바꾸자)
- 팀 전체가 아키텍처를 이해하고 있도록 만드는게 아키텍트의 역할.
- “전체 그림중에서 개발자 A씨의 역할은?”

- 소프트웨어 (애플리케이션)에 대한 아키텍처 정의
- 구성 요소 : 컴포넌트, 컴포넌트간 관계, 호출 순서, 통신 인터페이스



1. 정적 아키텍처 (Static Architecture)

- 계층 모델
- 컴포넌트간 관계



2. 동적 아키텍처 (Dynamic Architecture)

Statechart - Dig

3. 인터페이스 정의서 (Interface Definition Spec)

4. 상세 아키텍처 (Detail Architecture)

gnu

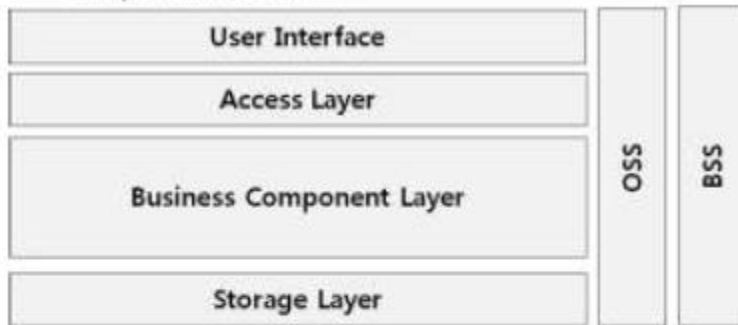
- 정적 아키텍처 (Static architecture) / 계층 모델 정의
 - 애플리케이션을 구성하는 컴포넌트들을 정의
 - 계층(Layer)별로 정의 하여 상세화 (보통 3~4단계가 적절)
- ※ 간단해 보이지만 매우 중요함. (향후 시스템의 구조, 팀 구조에 영향)



애플리케이션 아키텍처

애플리케이션 아키텍처

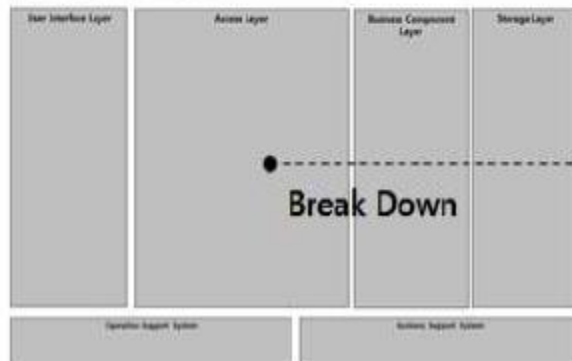
• Component Level 0



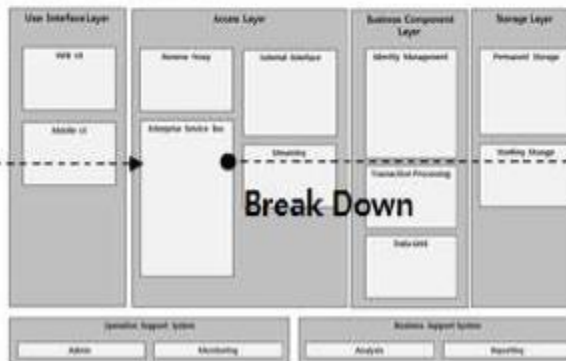
• Component Level 0 Description

Level 0	Description
User Interface	웹, 모바일 등 사용자 인터페이스 계층
Access Layer	비즈니스 로직을 OPEN API를 이용하여 외부로 서비스 하는 계층 외부 시스템과의 인터페이스를 제공
Business Component Layer	비즈니스 로직을 구현하는 계층
Storage Layer	데이터를 저장하는 계층
OSS	운영, 모니터링 관리를 위한 계층
BSS	비즈니스를 위한 지표 분석 리포팅 서비스

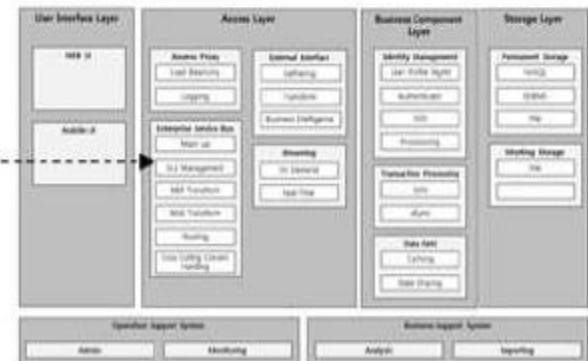
Level 0. Component Diagram



Level 1. Component Diagram



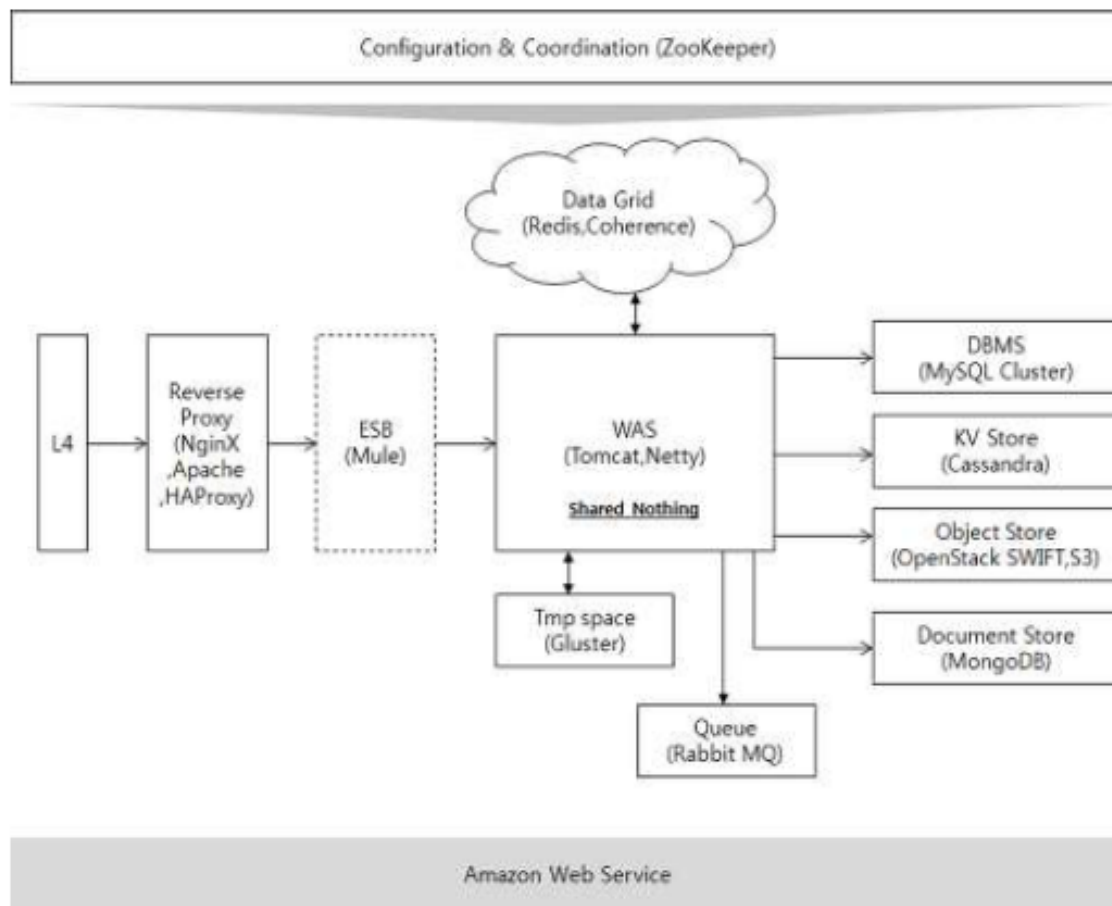
Level 2. Component Diagram



- 정적 아키텍처

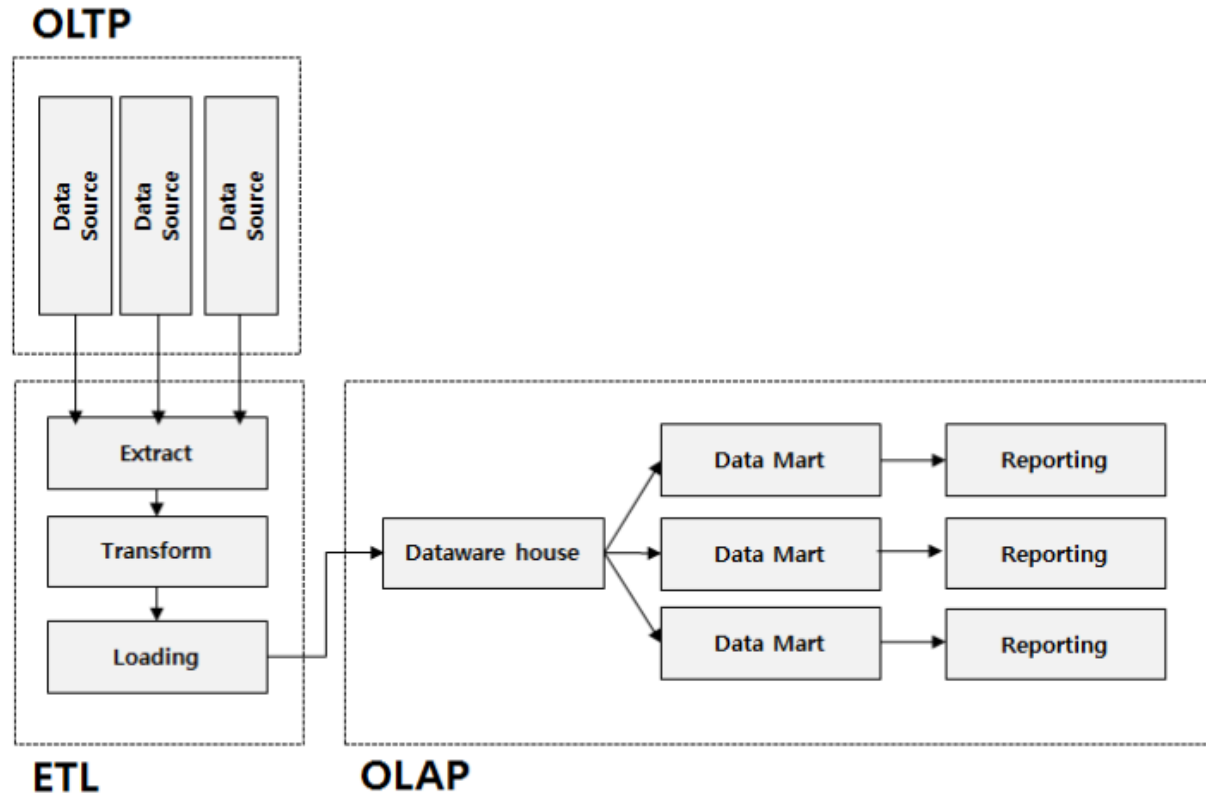
(Static architecture) /
컴포넌트간의 관계 정의

- 계층모델에서 정의된 컴포넌트간의 상호 연계성을 정의



파일 스토리지 서비스 아키텍처 예제

- 정적 아키텍처
(Static architecture) /
컴포넌트간의 관계 정의
 - 예) OLAP
(OnLine Analytics Processing) 기
반의 분석 시스템 아키텍처



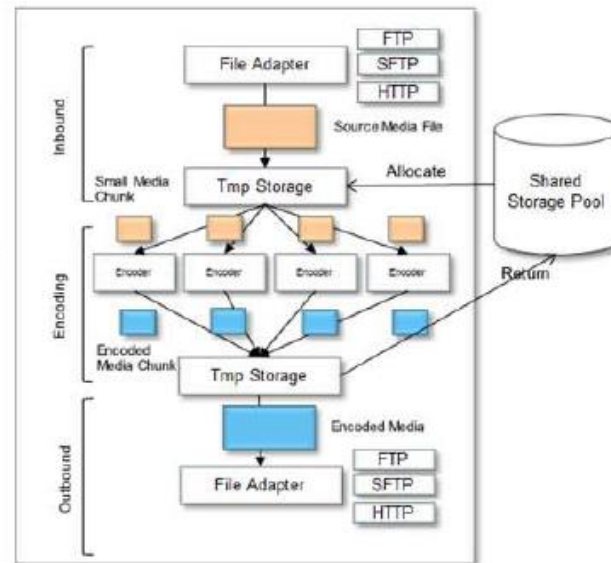
- 상세 아키텍처 (Detail Architecture)
 - 상세한 아키텍처에 대한 흐름에 대해서는 별도로 정의
- ### Transcoding Component Architecture

Encoding workflow (Parallel Encoding) detail

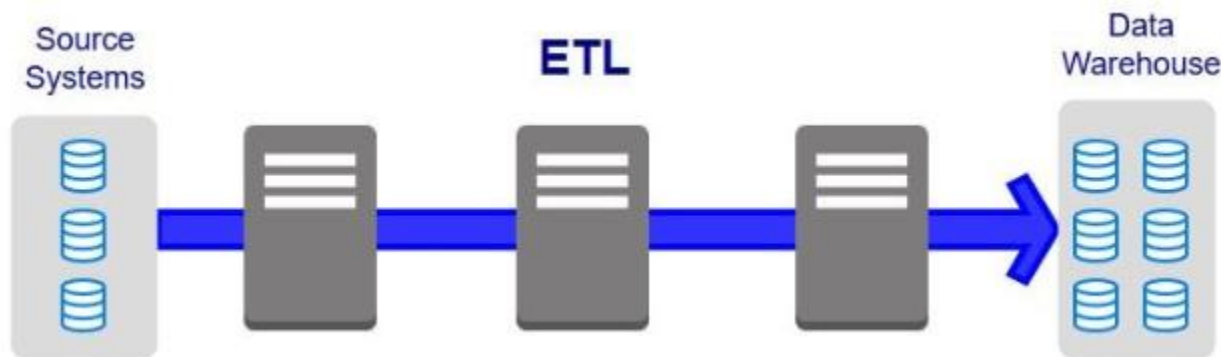
- 1) File Adapter gets source file.
- 2) File Adapter allocate temp storage area to store source file.
- 3) The source file is stored into the temp storage area.
- 4) The source file is spited to multiple small chunk.
- 5) Multiple Encoders are invoked and each encoder encode the small chunks. After all the chunks has been encoded, it is merged one file. (Similar to Map & Reduce). By using parallel processing it enables us to maximize hardware resource utilization.
- 6) After finishing the encoding, encoded result file is transferred into destination by using File Adapter.

Tmp Storage Consideration

Tmp Storage is just used to store source file and encoded result file – (Temporary working space)
Requirement is provide high performance IO but reliability is not required. (If IO fail is occurred just retry it.)
It is recommended using Local Disk in physical server.



- 인터페이스 정의
 - 프로토콜 정의 (REST, FTP, Google protocol buffer)
 - 메시지 포맷 정의 (REST(REpresentational State Transfer) 정의서)
 - 메시지 전달 방식 정의 (aka. Message Exchange Pattern. MEP)
 - 동기/비동기
 - ETL (Extract, Transform, Load) 방식



애플리케이션 아키텍처

애플리케이션
아키텍처

1 "Hello World!"

The simplest thing that does something



Python | Java

2 Work queues

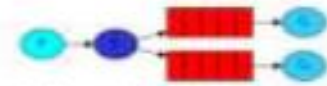
Distributing tasks among workers



Python | Java

3 Publish/Subscribe

Sending messages to many consumers at once



Python | Java

4 Routing

Receiving messages selectively



Python | Java

5 Topics

Receiving messages based on a pattern



Python | Java

6 RPC

Remote procedure call implementation

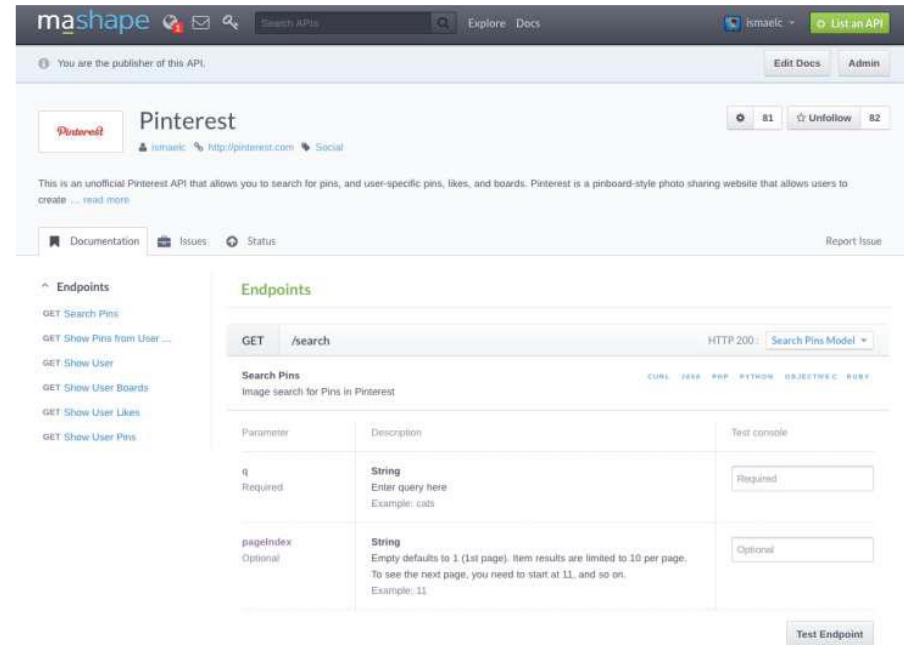


Python | Java

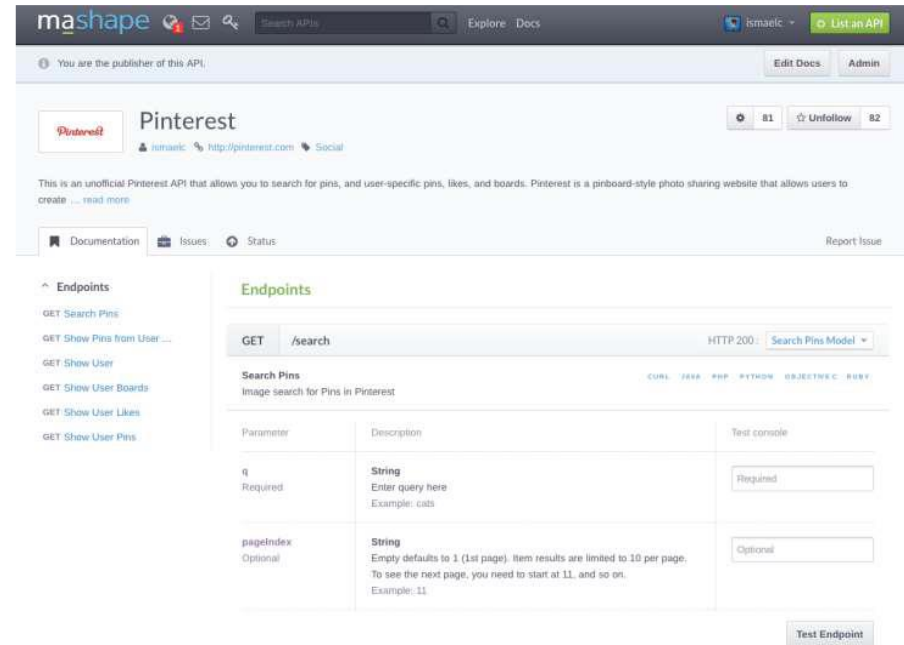
Rabbit MQ에서 정의된 MEP 양식



- 인터페이스 정의 / REST API
 - SWAGGER와 같은 툴을 사용하기도 함
 - 검토 / 퍼블리싱 2단계로 문서를 유지하는게 좋음
 - 검토 : 위키 또는 문서
 - 퍼블리싱 : SWAGGER



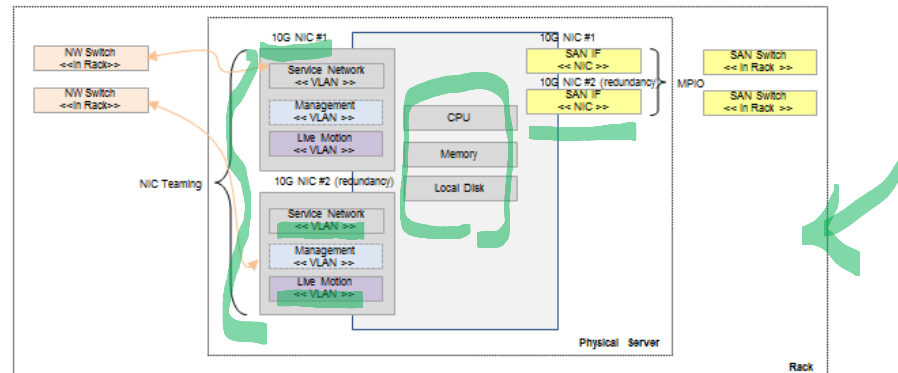
- 인터페이스 정의 / REST API
 - SWAGGER와 같은 툴을 사용하기도 함
 - 검토 / 퍼블리싱 2단계로 문서를 유지하는게 좋음
 - 검토 : 위키 또는 문서
 - 퍼블리싱 : SWAGGER



- 하드웨어 아키텍처
 - 서버 아키텍처
 - 네트워크 구성
 - 스토리지 구성
 - 랙 디자인 구성
 - 글로벌 디플로이 구조
- 솔루션 아키텍처
 - 데이터베이스, 미들웨어, 웹서버등의 구성 아키텍처
 - ※ 클러스터링 등

- 하드웨어 아키텍처 / 서버 아키텍처
 - CPU, 내장 디스크, 메모리 구성, 네트워크 인터페이스 구성등
 - 서버 타입 정의 (웹서버, 데이터 베이스 서버등)

□ Physical Server Architecture



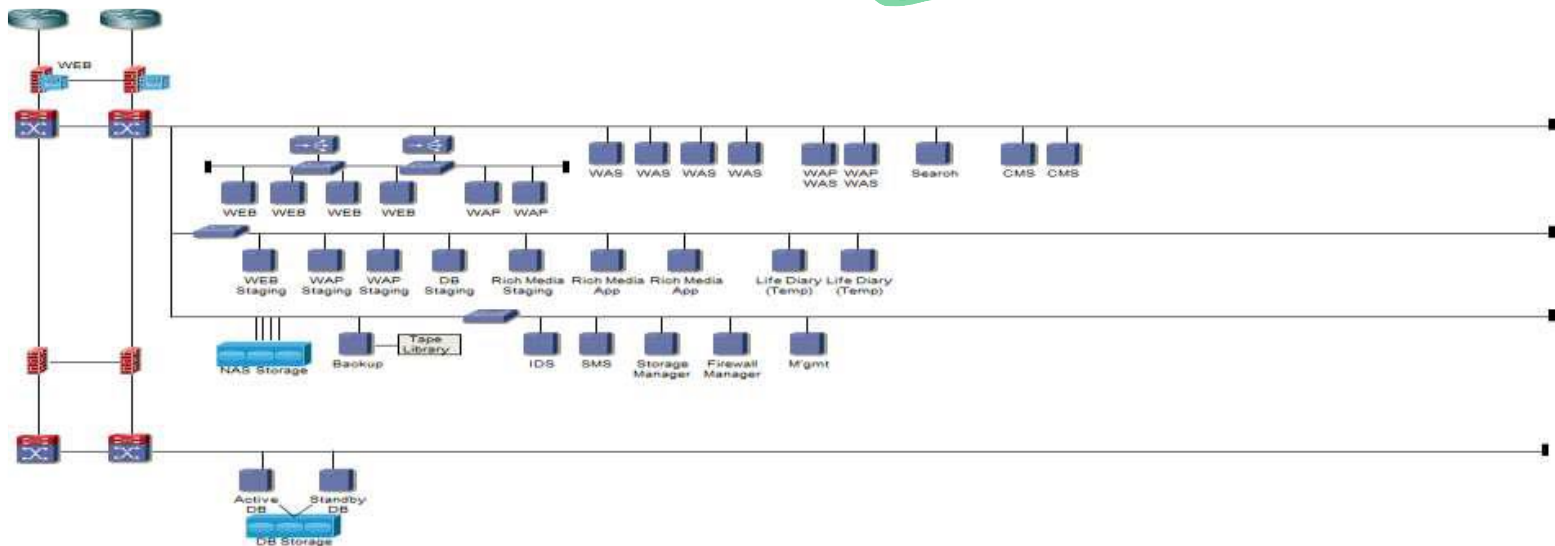
Physical Server component

- CPU Architecture
 - SLAT (Second Level Address Translation) is recommended for virtualization environment – (Intel EPT, AMD NPT)
 - Current hypervisor limitation of physical core : virtual core is 1:8
 - Recommended ratio is 1:4
- Disk
 - Local Disk is used for booting Hypervisor
 - Disk mirroring is recommended for disk failure
 - Only small size of disk is required. (SATA type disk is enough)
- Memory
 - Commonly 2GB memory per VM is recommended
 - 2GB base memory is required for Hypervisor (it is different depends on Hypervisor)

테크니컬 아키텍처

테크니컬(인프라) 아키텍처

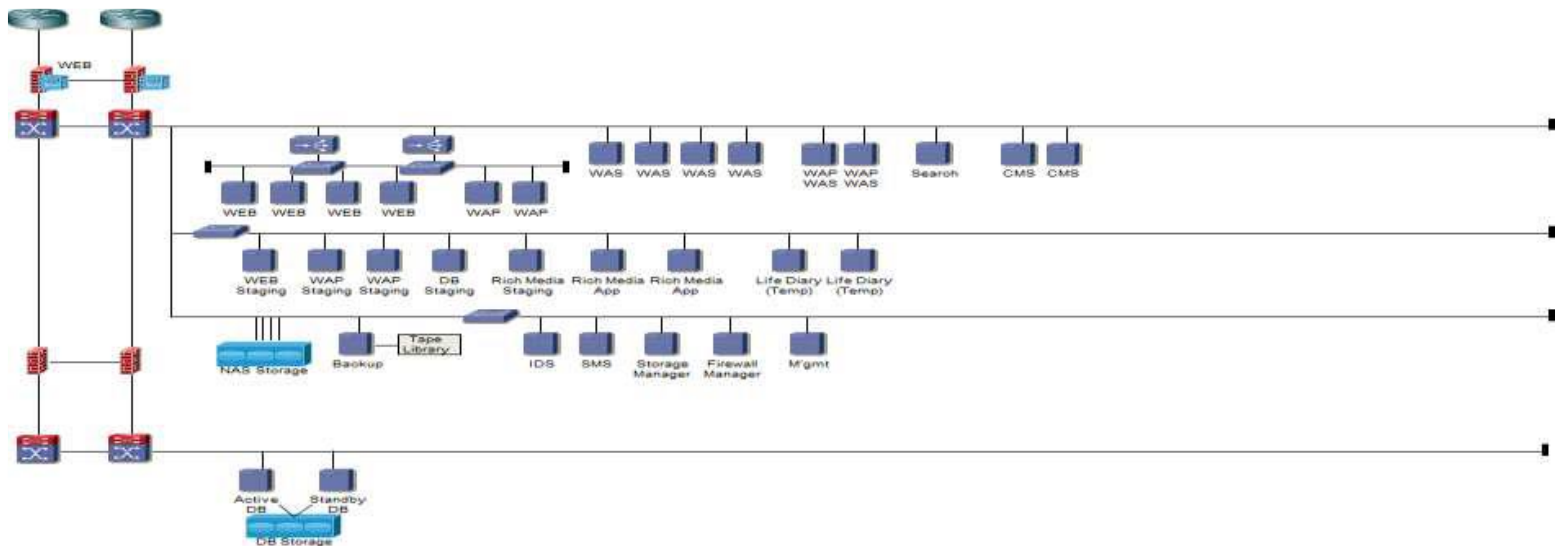
- 하드웨어 아키텍처 / 네트워크 아키텍처
 - 망 종류 - 스토리지 네트워크, 관리 네트워크, 서비스 네트워크 등
 - 외부 네트워크와 연결하기 위한 라우터
 - 백본이 되는 L2, 로드밸런싱을 위한 L4,L7
 - 보안을 위한 IPS, 내부 IP를 이용하기 위한 NAT
 - LAN 구성, 방화벽 구성
 - Subnet 구성 등



테크니컬 아키텍처

테크니컬(인프라) 아키텍처

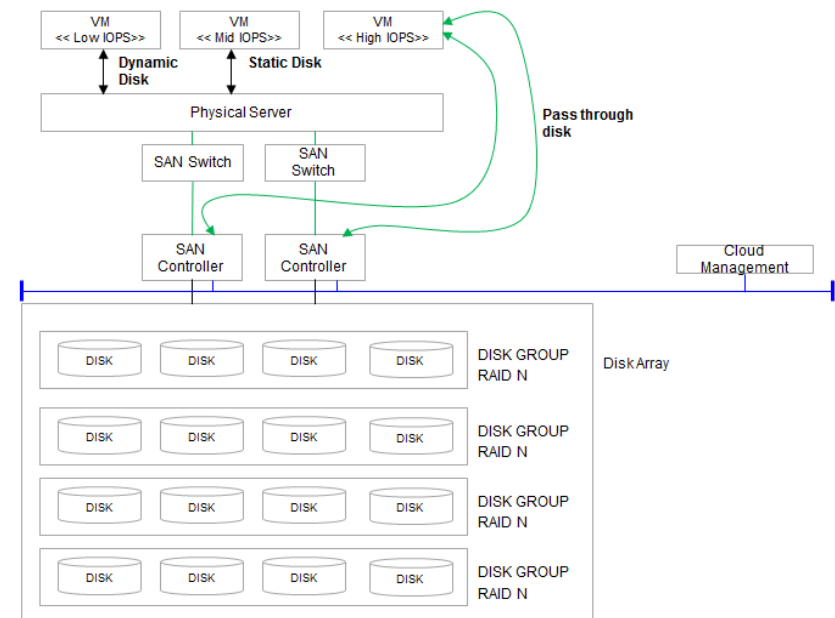
- 하드웨어 아키텍처 / 네트워크 아키텍처
 - 망 종류 - 스토리지 네트워크, 관리 네트워크, 서비스 네트워크 등
 - 외부 네트워크와 연결하기 위한 라우터
 - 백본이 되는 L2, 로드밸런싱을 위한 L4,L7
 - 보안을 위한 IPS, 내부 IP를 이용하기 위한 NAT
 - LAN 구성, 방화벽 구성
 - Subnet 구성 등



- 하드웨어 아키텍처 / 스토리지 아키텍처

- 스토리지 타입 정의 (DAS, NFS, SAN 등)
- 스토리지 컨트롤러, 물리 디스크
- (SAS, SATA, RPM 등)
- 스토리지 연결 네트워크
- RAID 구성등을 정의

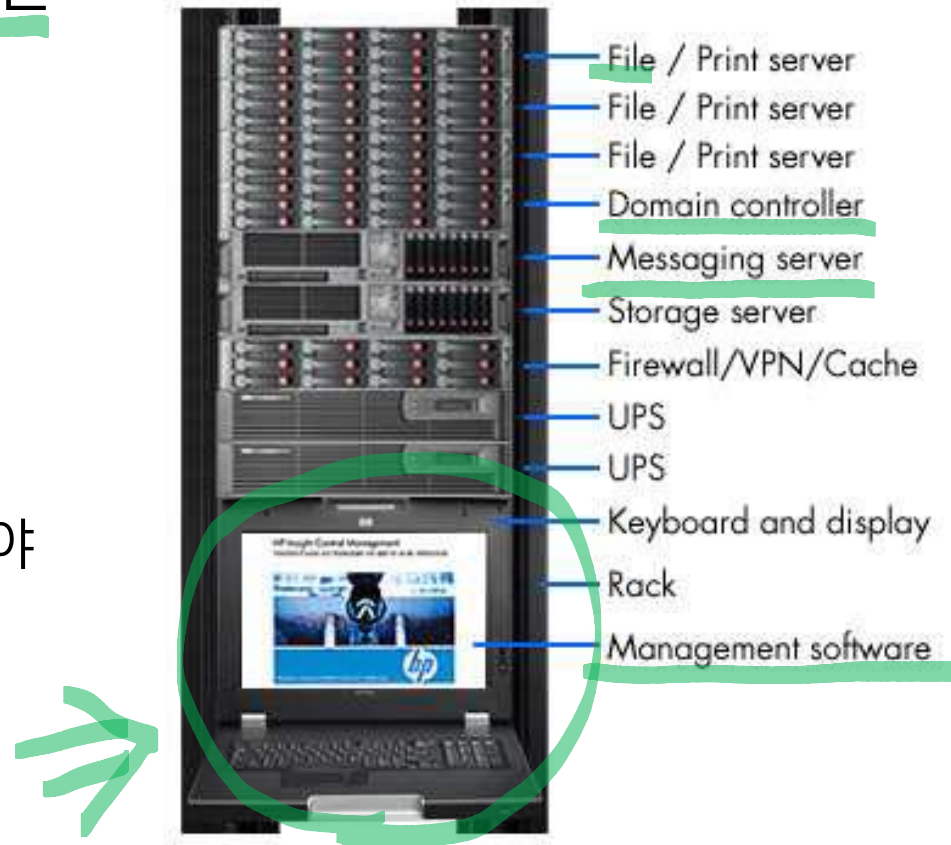
Storage Architecture



테크니컬 아키텍처

테크니컬(인프라)
아키텍처

- 하드웨어 아키텍처 / 랙 디자인
아키텍처 (물리 서버 배치)
 - 랙에 서버 배치 구조
 - 케이블링 정의
 - KVM, UPS 등 부가 장비 배치
 - 사용 전력, 발열량이 고려되어야 함

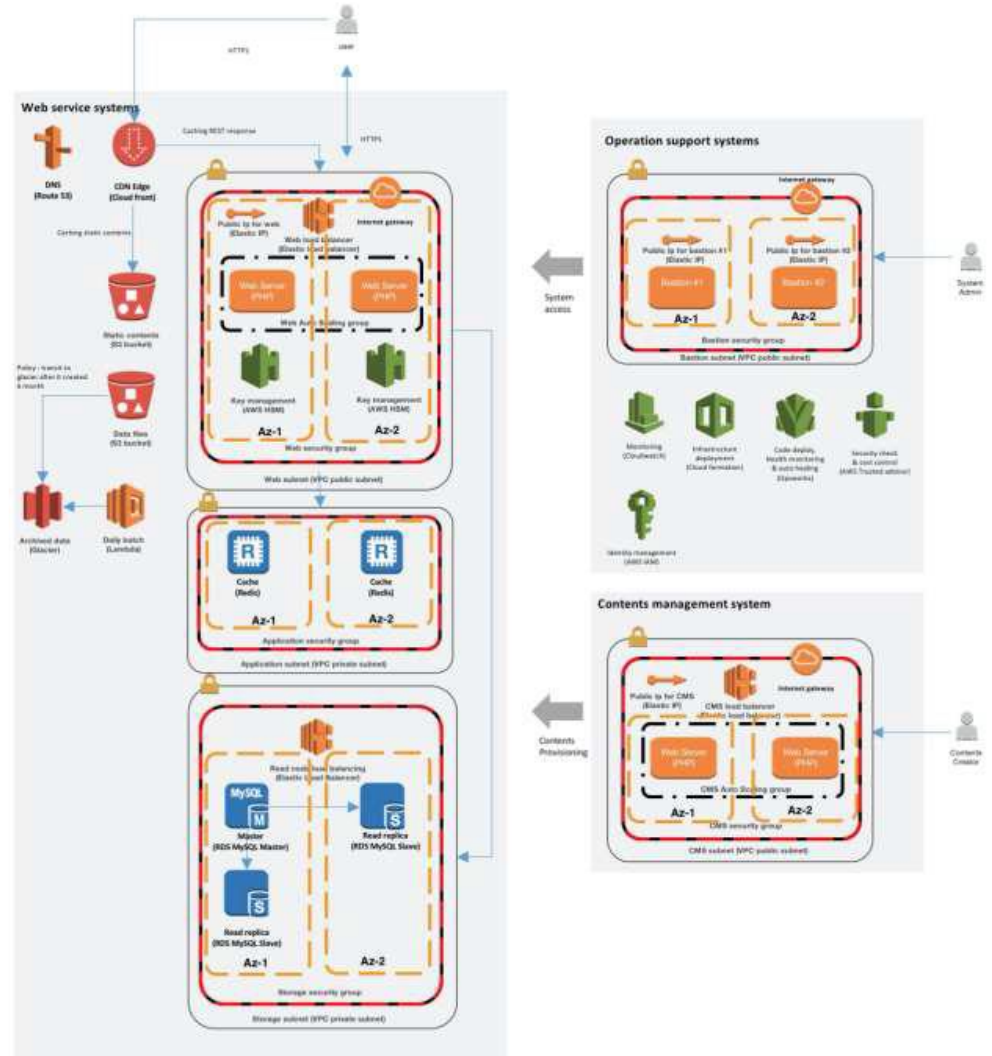


grm

테크니컬 아키텍처

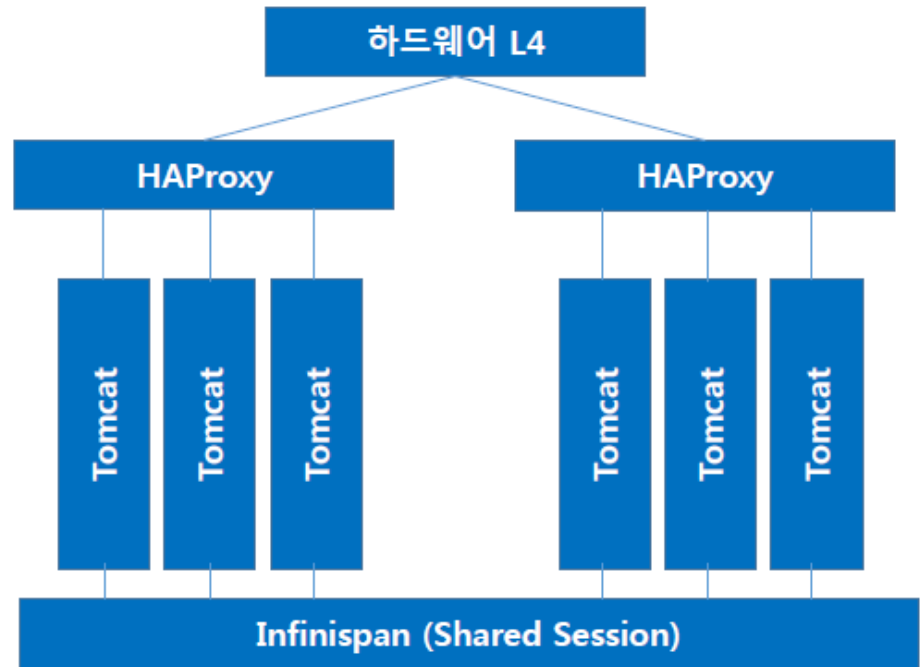
테크니컬(인프라) 아키텍처

- 아마존 클라우드 기반 배포 모델설계 예시



- 데이터 베이스, 미들웨어 등의 구성과 배포 구조 정의

※ 특히 클러스터링, 로드 밸런싱, HA 구조 등에 대한 정의

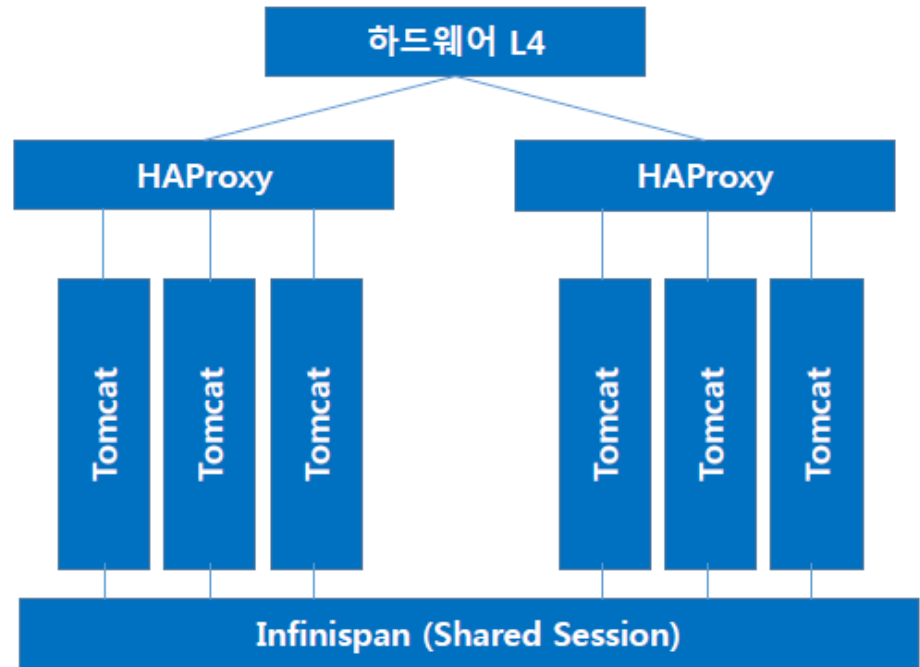


Tomcat 클러스터 배포 구조

- 솔루션 아키텍처

- 데이터 베이스, 미들웨어 등의 구성과 배포 구조 정의

※ 특히 클러스터링, 로드 밸런싱, HA 구조등에 대한 정의



Tomcat 클러스터 배포 구조

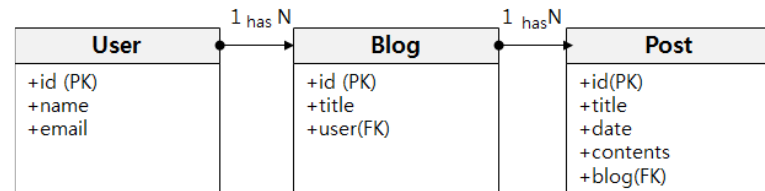
- 시스템에 저장되는 데이터에 대한 아키텍처 정의
 - 데이터 구조
 - 저장 장소 및 솔루션
 - 보안 처리 아키텍처 (암호화등)
 - 데이터 생명 주기 관리 (생성, 백업, 폐기까지의 정책)

- 데이터 구조

- Conceptual Modeling



- Logical Modeling



- Implementation Modeling



TABLE : USER

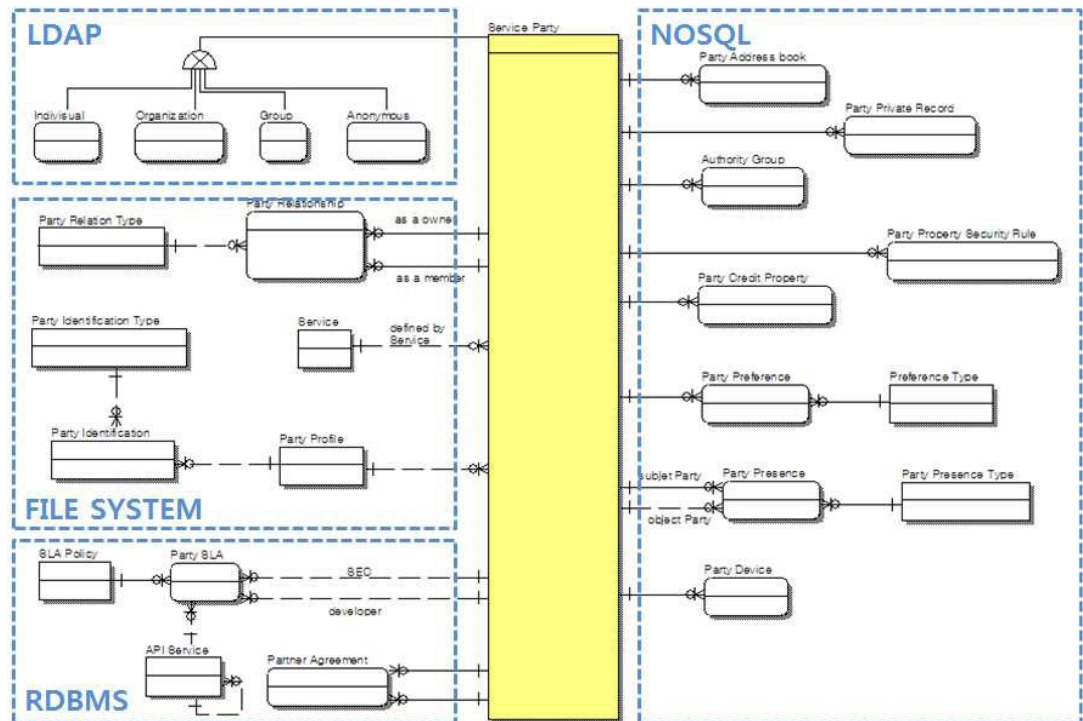
column	data type	Index
id	VARCHAR(255)	PK
name	VARCHAR(255)	
email	VARCHAR(255)	



데이터 관련 테크니컬 아키텍처

데이터 아키텍처

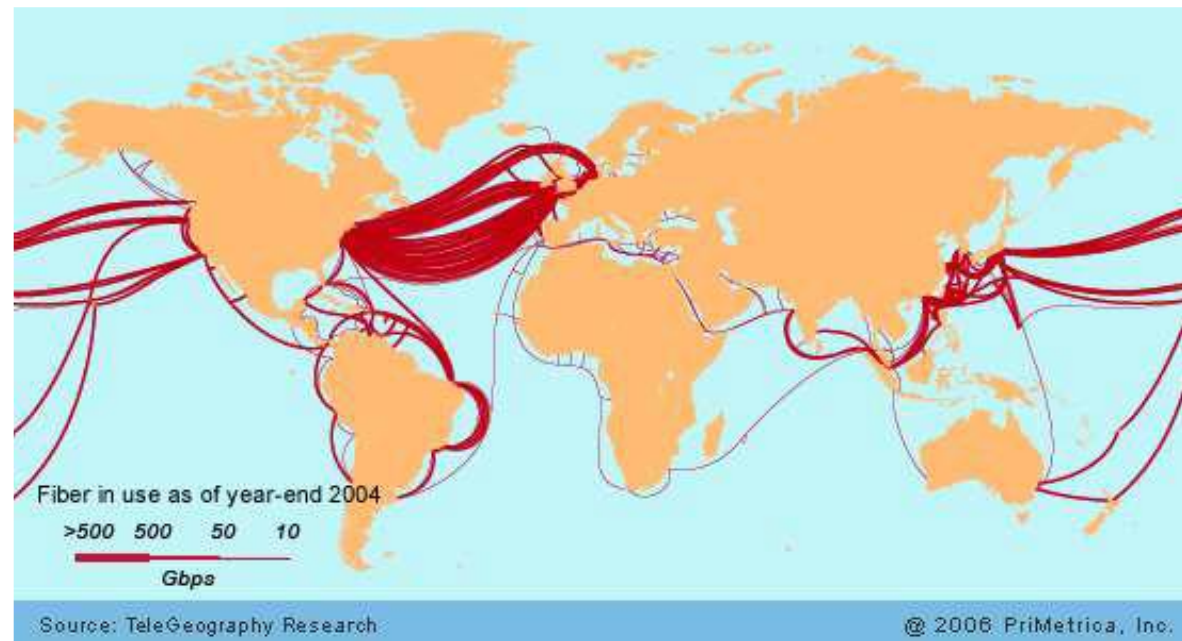
- 데이터 아키텍처 /
배포 구조
 - 데이터 저장소
 - 정의된 데이터 모델
에 대한 솔루션별
 - 저장소 아키텍처 정
의



데이터 관련 테크니컬 아키텍처

테크니컬(인프라)
아키텍처

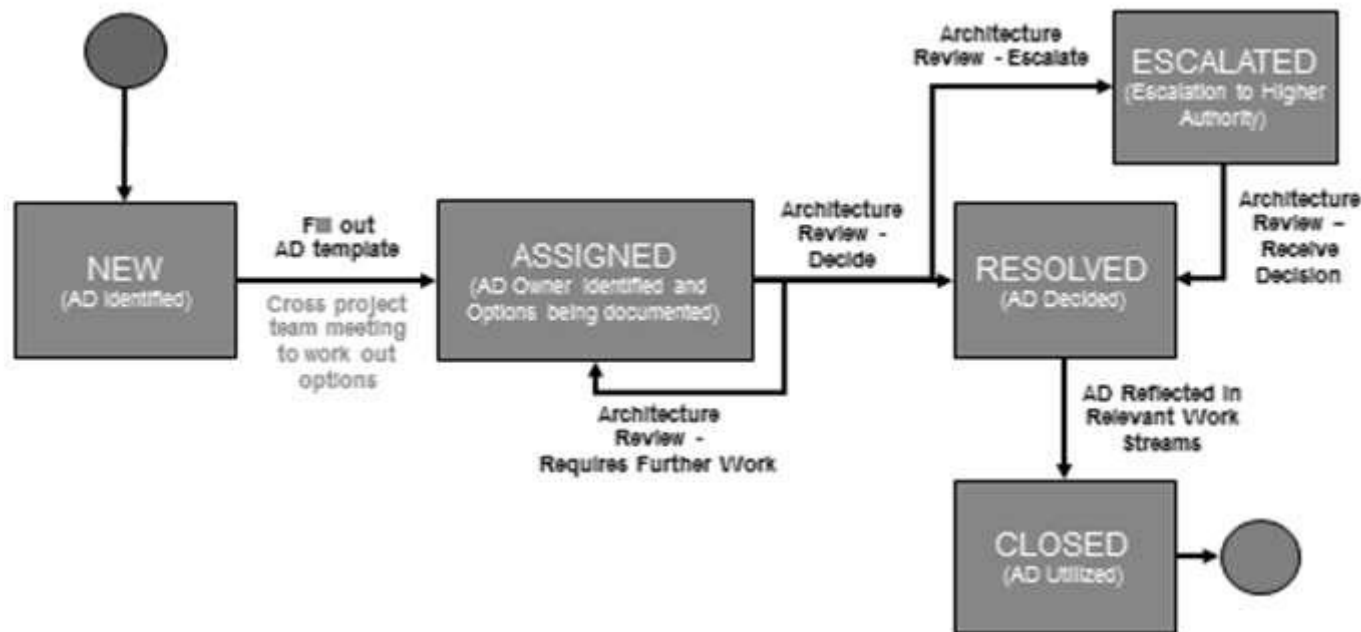
- 글로벌 ROLL OUT
아키텍처
 - 데이터 센타간 시스템 배포 구조 및 연동 구조, 데이터 복제 방식등을 정의



gnu

아키텍처 의사 결정 프로세스



- Architecture Decision (aka. AD)
 - 아키텍처적인 의사 결정이 필요한 경우
 - 요인 : 비용, 기술 선택, 조직의 보유 능력, 회사 전략 등
 - 최고 의사 결정 조직이 있어야 함. (CTO, Chief 아키텍트 등)



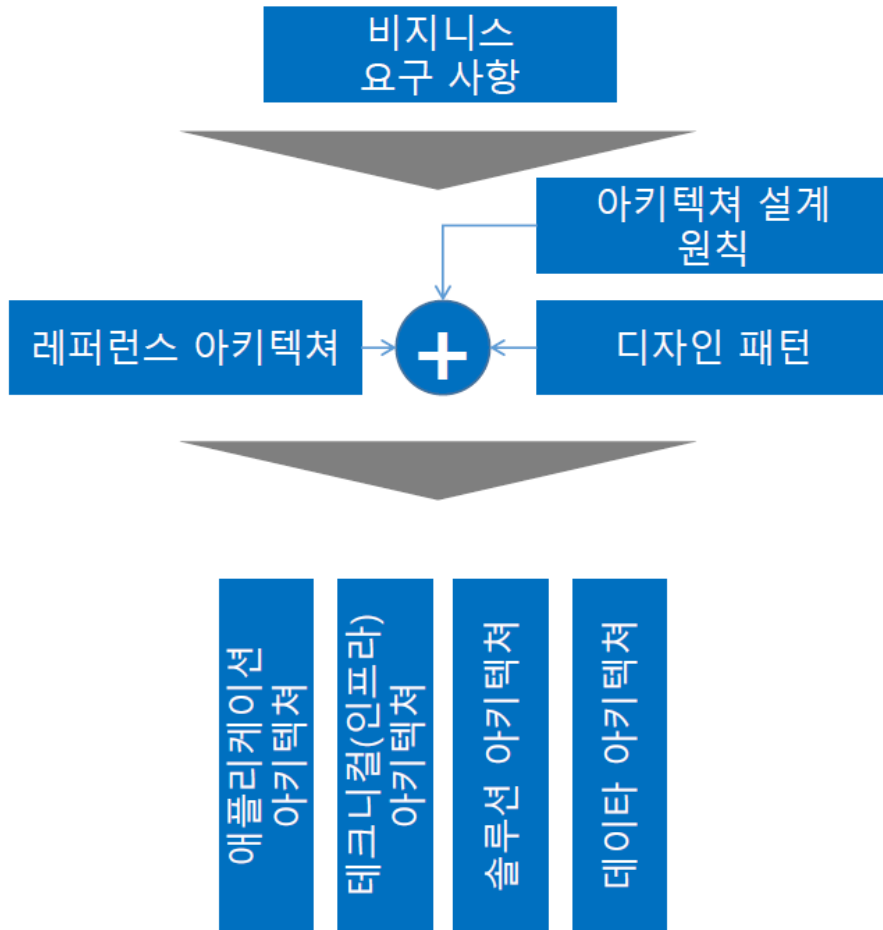
아키텍처 의사 결정 템플릿 (AD 템플릿)

- 각 옵션별 아키텍처에 대한 설명
- 옵션별 장단점, 의사 결정 결과, 임팩트 등을 한장에 설명
 - 잘 모아 놓는게 중요 (나중에 따소리 또는 왜 그렇게 했는지 설명)

AD I. Architecture Decision Item Title

<p>Option A. : {title}</p> <p>XA 기반 분산 트랜잭션</p>  <p>OPTION A에 대한 아키텍처 구조도</p> <p>아키텍처 구조 설명</p>	<p>Description & Motivation</p> <p>Assumption</p> <p>Consideration</p> <p>Option A Pros : Cons :</p> <p>Option B Pros : Cons :</p> <p>Decision</p> <p>Implication</p>
<p>Option B. : {title}</p> <p>XA 기반 분산 트랜잭션</p>  <p>OPTION B에 대한 아키텍처 구조도</p> <p>아키텍처 구조 설명</p>	

결론



- 아키텍처 설계 원칙
- 애플리케이션 아키텍처
- 테크니컬(인프라) 아키텍처
- 솔루션 아키텍처
- 데이터 아키텍처

다음...

- Formal model of systems for safety critical systems