대용량 아키텍쳐 설계

# 아키텍쳐 설계 프로세스

# #목표

"어떤 흐름으로 아키텍쳐를 설계해야 하는지에 대해서 알아본다."

# #1

아키텍쳐 설계 개요

# 소프트웨어 아키텍쳐란?

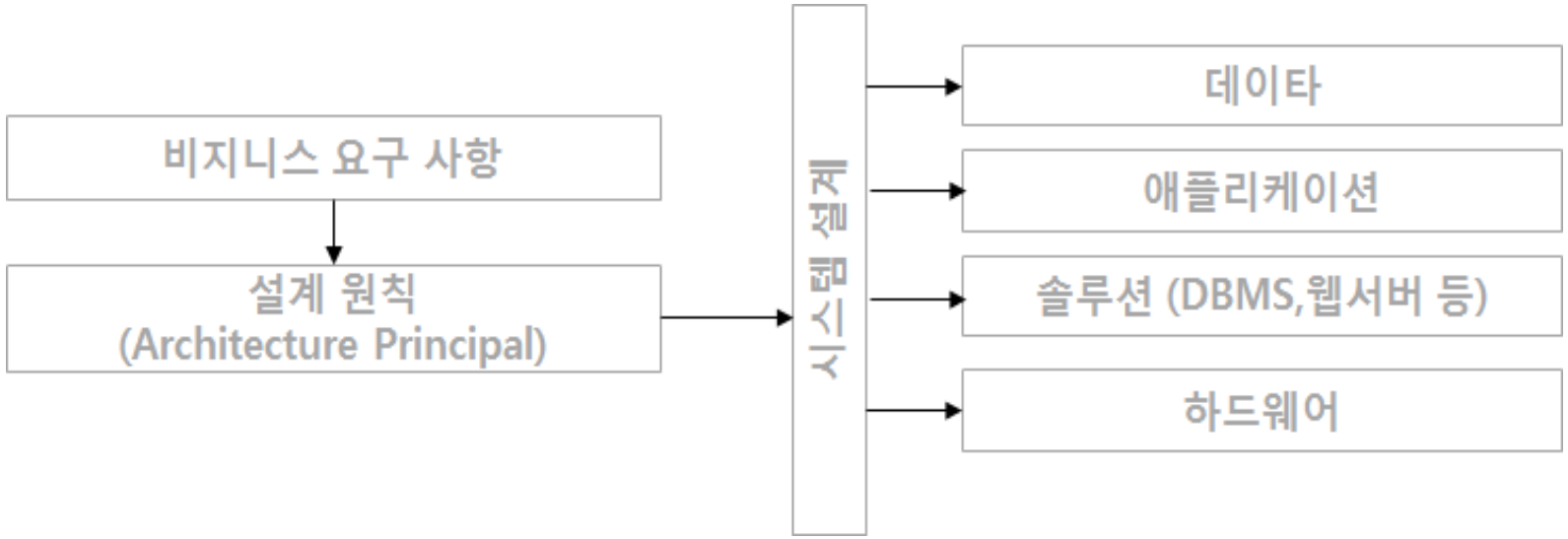
- 아키텍쳐의 정의
  - 수많은 아키텍쳐에 대한 정의가 있음 ( http://www.sei.cmu.edu/architecture/start/glossary/community.cfm )
  - 오늘 설명할 아키텍쳐의 정의는

> "아키텍쳐는 비지니스 요구 사항을 만족하는 시스템을 구축하기 위해서 전체 시스템에 대한 구조를 정의한 문서로, 시스템을 구성하는 컴포넌트와, 그 컴포넌트간의 관계, 그리고, 컴포넌트가 다루는 정보(데이타)를 정의"

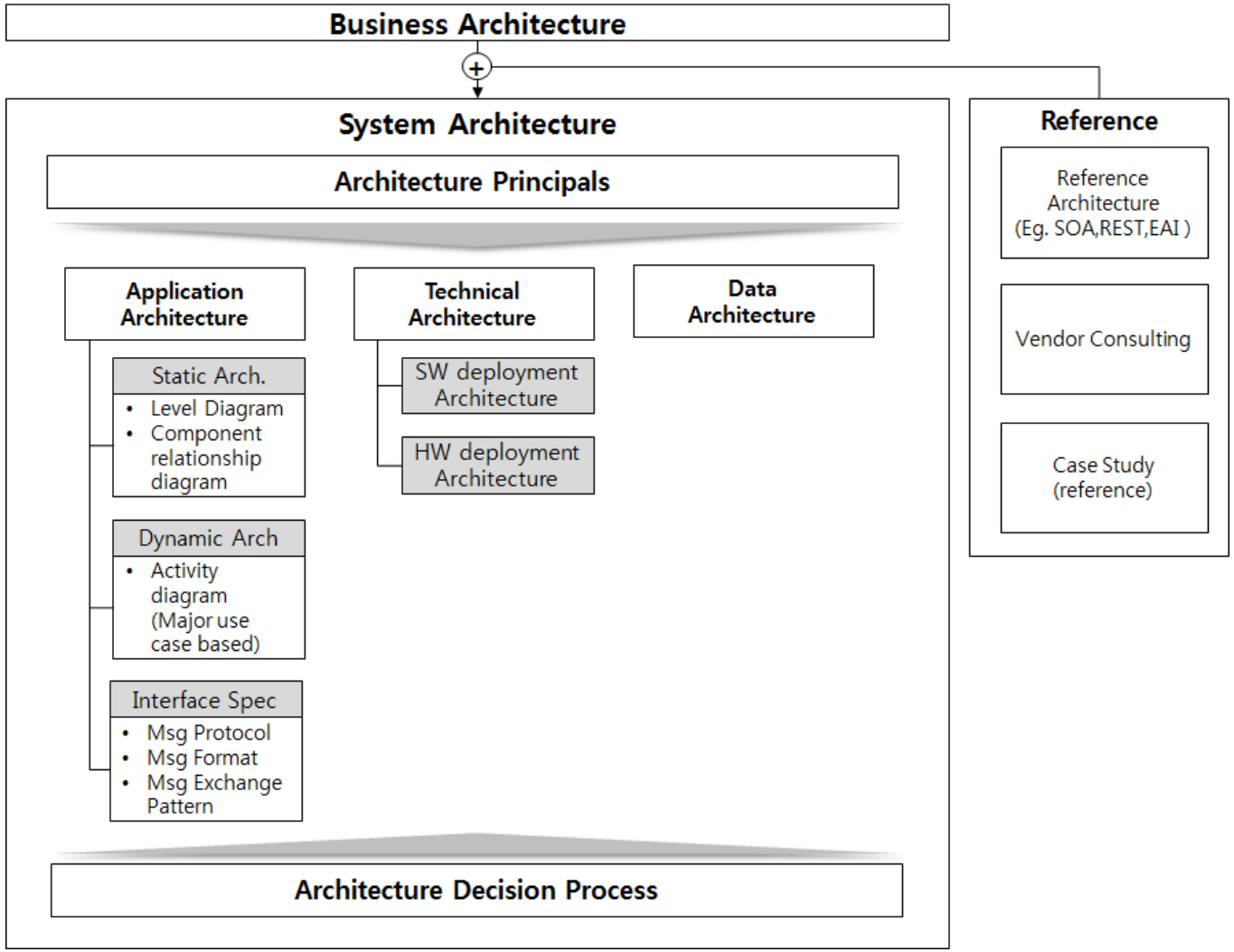
  - 아키텍쳐는 비지니스 요구 사항을 기술로 해석해놓은 것
  - 개발의 방향을 알려주는 지도
  - 의사 소통의 매개체
  - 정답은 없음. 팀의 수준에 맞게, 이해할 수 있는 수준으로, 그러나 모든 내용을 담아야 함

# 아키텍쳐 설계 프로세스

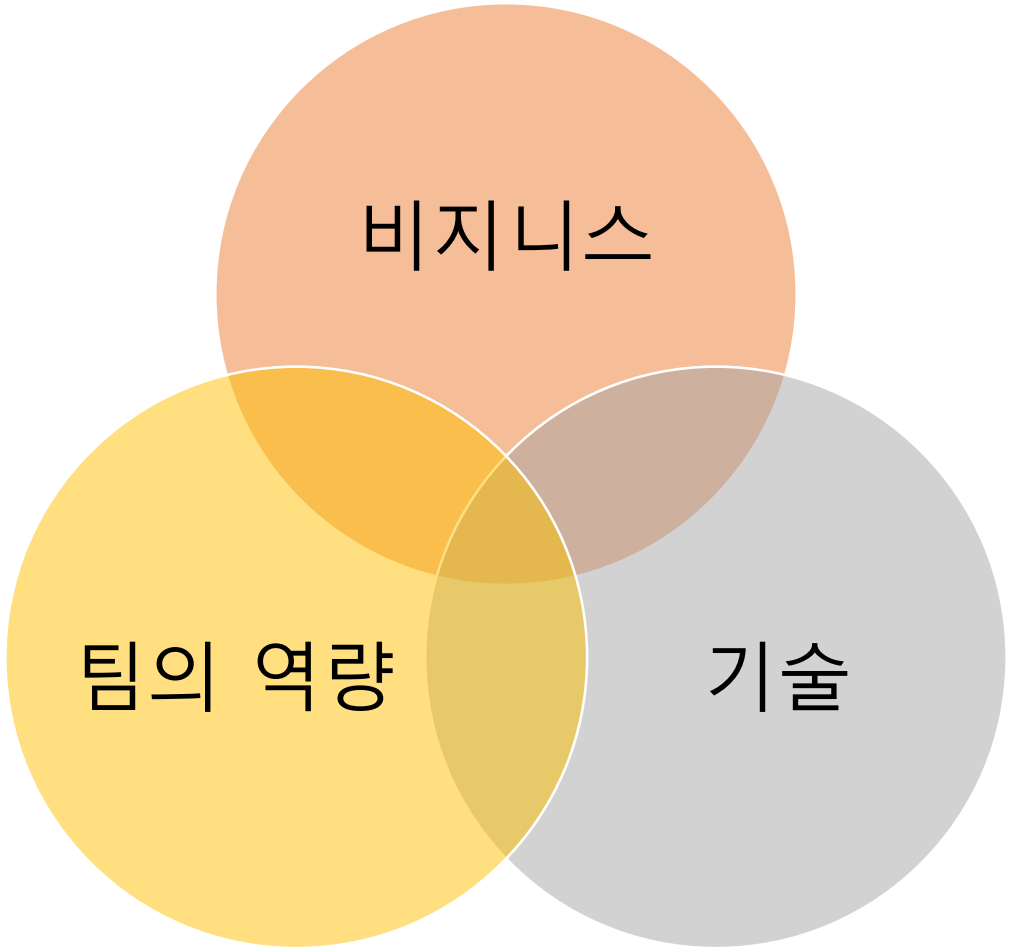
- 오늘 부터 우리가 배울 아키텍쳐 설계 방법론은?
  - Zachman 프레임웍, Federal enterprise architecture 등 여러 방법론이 있음
  - 우리는 토가프(TOGAF) 아키텍쳐 프레임웍을 축소한 버전을 사용



# 아키텍쳐 설계 프로세스

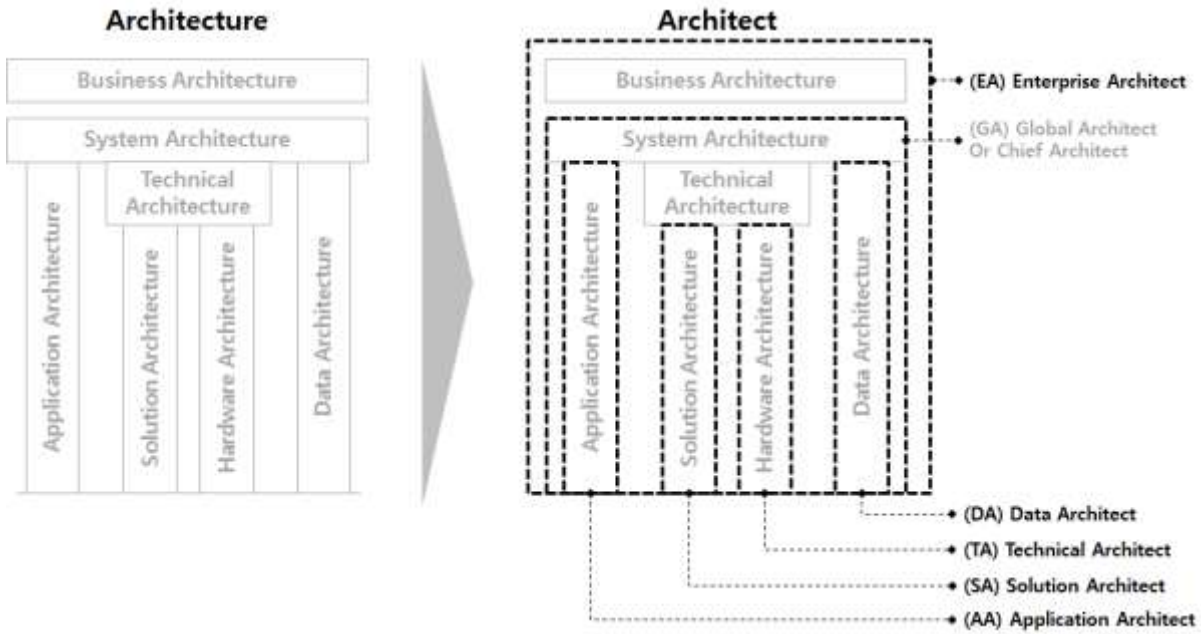

Business Architecture
System Architecture
Architecture Principals
Application Architecture
Technical Architecture
Data Architecture
Static Arch.
• Level Diagram
• Component relationship diagram
Dynamic Arch
• Activity diagram (Major use case based)
Interface Spec
• Msg Protocol
• Msg Format
• Msg Exchange Pattern
SW deployment Architecture
HW deployment Architecture
Architecture Decision Process
Reference
Reference Architecture (Eg. SOA,REST,EAI )
Vendor Consulting
Case Study (reference)

# 아키텍트란?




- 아키텍쳐를 그리는 사람
- 통역자 "비지니스 언어를 기술 언어로"
- 방향 지시자 "개발의 방향, 시스템의 청사진을 제안"

# 역할별 세분화된 아키텍트의 종류




- EA (Enterprise Architect) :
  비지니스와 기술 사이. 전략 수립. 전체 그림
- AA (Application Architect) :
  애플리케이션 구조 설계, 표준 설정
- TA (Technical Architect) :
  하드웨어 인프라 설계
- SA (Solution Architect) :
  특정 소프트웨어 솔루션 구성 설계
- DA (Data Architect) :
  데이타 아키텍쳐 설계

## 좋은 아키텍트에게 필요한 것

- 커뮤니케이션
- 추상화 능력
- 비지니스에 대한 이해
- 기술에 대한 깊은 이해
- 코딩 !!
- 비지니스 요건 팀의 능력, 기술등은 항상 변한다. (아키텍쳐는 항상 변화한다.)

듣는 기술

스토리텔링

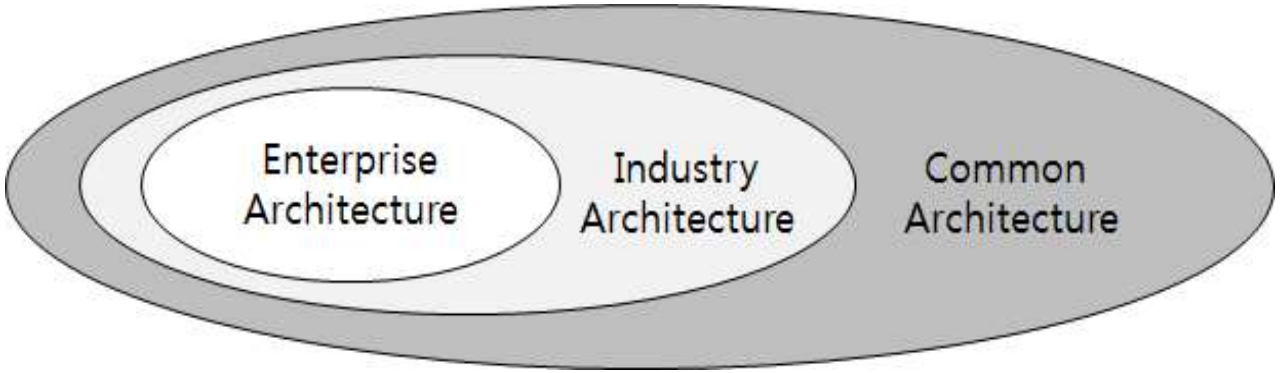
쉽게 설명하는 기술

설득하는 기술

함께 협업하는 기술

# 레퍼런스 아키텍쳐

- 레퍼런스 아키텍쳐
  - 아키텍쳐 설계에 참고할 수 있는 샘플(레퍼런스) 아키텍쳐




- Common Architecture : 업무 도메인 종속성이 없음 – SOA,REST 등
- Industry Architecture : 특정 업무 도메인 종속정 – MES,PLM 등
- Enterprise Architectrue : 특정 회사의 아키텍쳐

# #2

비지니스 아키텍쳐의 설계

# 비지니스 아키텍쳐란?

- 비지니스 아키텍쳐
  - **서비스에 대한 소개**
  - **시스템에 대한 간략한 전체 구조**
  - **사용자에 대한 도메인 모델**
  - **주요 기능에 대한 흐름**
  - 시장 현황과 차별화 전략
  - 비지니스 로드맵과 일정
  - 투자 및 수익 정보 (비용 정보)

"누가 무엇을 하는 시스템이고,
비지니스 가치를 실현하기 위해서 어떤 전략을 취하는지 간략하게 서술되어야 함."

※ MRD (Market Research Description)과 PRD (Product Research Description)의 요약

# 1. 서비스 모델의 정의

- 누가? 어떤 기능을 사용하는지에 대한 정의
- 비지니스적으로 어떤 특징을 갖는지 (시장 차별화 및 수익 모델)이 들어가면 좋음
- 1~2 페이지 정도로 간략하게

To be more clear design, I defined number of assumption like below

Here is use case for the service

"The company provide contents media service to customer. Contents creator in the company uploads contents thru their own CMS (contents management system) and the contents will be provisioned to their service web site. End user will consume the contents thru a their web service web site."

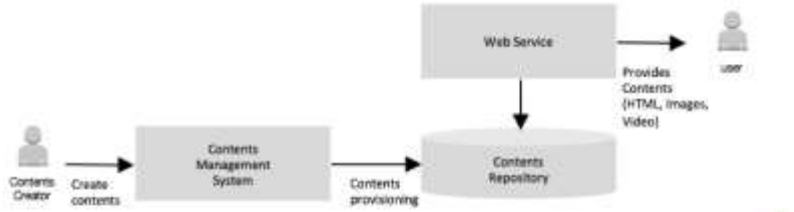



Figure 1 Use case scenario

**Technical assumptions**

- Their system is implemented by using SPA(Single Page Application) web front and REST back end API.
- The contents which are served by this site are image ,video(MP4 only) and HTML.
- Some user data should be stored in RDBMS with encryption to secure user information.
- In a such small start up, it doesn't have enough resource to have big size of operation team.
- The customer wants to their service to global roll out later.

웹을 통해서 컨텐츠 미디어 서비스를 하는 서비스에 대한 모델 정의 예시

# 2. 시장 현황 분석

- 시장의 크기, 경쟁사 분석
- 이를 통해서 전체 시스템의 크기, 성능 용량등 기본적인 기준점 마련이 가능


**소셜 소프트웨어 벤더의 Gartner Magic quadrant chart**


**블로그 서비스에 대한 기능 비교**
http://blog-services-review.toptenreviews.com/

## 3. 비지니스 전략

- 어떻게 가치(돈을 벌것인가?)를 창출할 것인가?
- 차별화 전략은 무엇인가?

모바일 디바이스 관리 서비스 (MDM)

인프라에 대한 관리가 어려운 중소기업을 대상으로,
클라우드 서비스 형태로 손쉽게 직원들의 단말기를 관리할 수 있는 서비스를 저비용으로 제공한다.

## 4. 주요 기능 정의

- 시스템에 대한 핵심 기능을 10~20개 정도로 간략하게 정의
- 누구나 이해할 수 있을 정도로 쉽게
- 블로그 서비스 기능 정의 예
  - 블로그 글쓰기와 읽기 기능을 제공
  - 구글,페이스북,카카오톡 SNS 서비스 계정을 이용한 로그인 기능 제공
  - MS 워드 연동을 통하여 블로그에 글 올리기 기능 제공

## 4. 주요 기능 정의

- 드롭 박스와 같은 개인 스토리지 클라우드 서비스 예

① 사용자가 파일을 업로드 한다.

② 사용자가 파일을 다운로드 받는다.

③ 사용자 기기와 클라우드상의 파일을 동기화 한다.

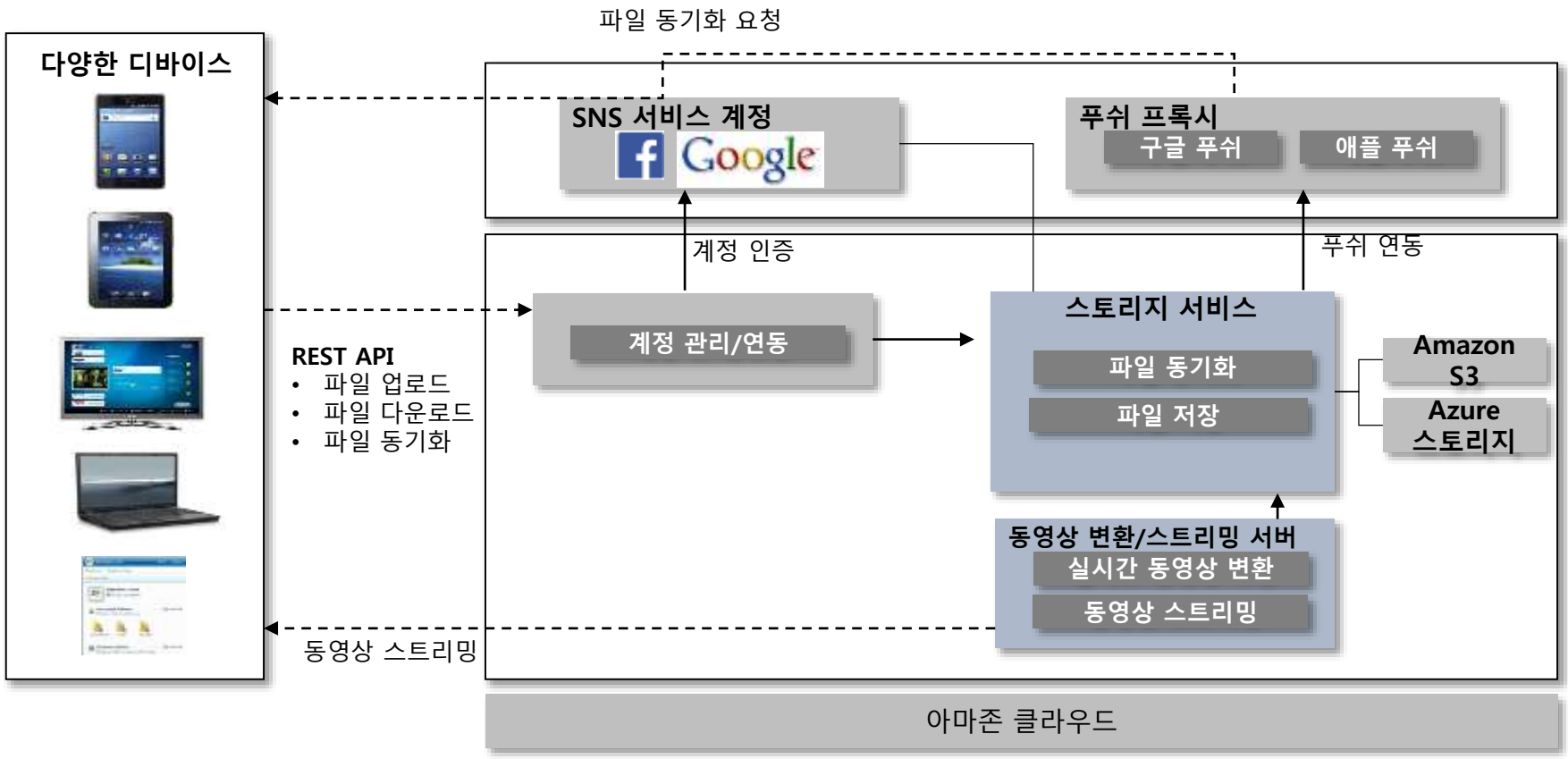
④ 사용자가 클라우드에 저장된 동영상을 사용자 기기의 포맷에 맞춰 동기화한다

⑤ 사용자가 이미지 파일에 대해서 썸 네일 목록을 본다.

⑥ 사용자가 저장한 파일은 변경되더라도 5회 이전까지 되돌림이 가능하다

# 5. 전체 아키텍쳐 정의

- 서비스를 제공하기 위한 전체 시스템 아키텍쳐 정의
  - 주요 시스템 컴포넌트, 시스템간의 연계, 시스템 사용자가 표기되어야 함




개인 스토리지 클라우드 서비스의 전체 아키텍쳐 예시

## 6. 비지니스 도메인 모델

- 서술되는 내용
  - 시스템 사용자와 사용자간의 관계 정의, (**사용자 권한**)
  - 시스템에 정의된 에셋 (글, 댓글) 간의 관계 정의 (ERD와 유사)
  - 주요 업무 프로세스 정의
  - **상태 전이** (Finite State Machine : FSM)
- 대부분은 PRD (Product requirement definition : 요구사항 명세서)에 정의되기 때문에 간략하게 요약하는 것이 좋다. ~~(개발자는 PRD를 상세하게 읽지 않는다.)~~

# 6. 비지니스 도메인 모델

- 모바일 컨텐츠 서비스의 사용자 정의 예제

**사용자**

서비스를 통해서 컨텐츠를 보거나 댓글을 다는 사람

**에디터**

서비스에 컨텐츠를 제작해서 업로드 하는 사람

**편집국**

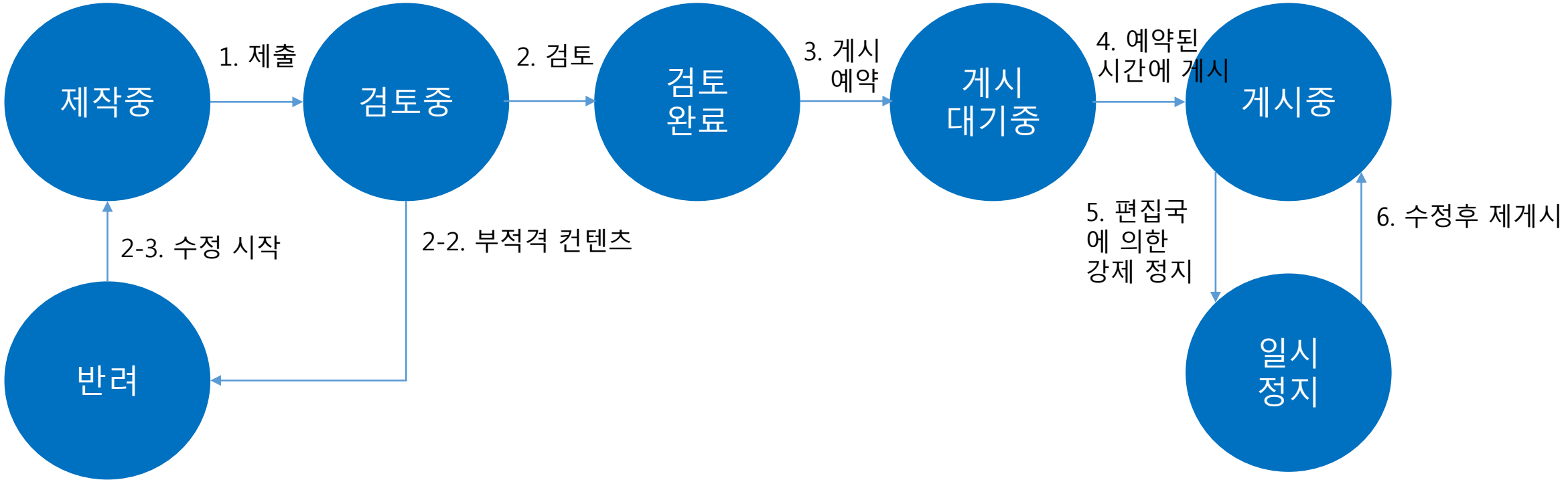
에디터에 의해 업로드 된 컨텐츠를 검수하여, 퍼블리싱 하는 사람

**댓글 운영자**

댓글을 모니터링 하여 불량 댓글을 삭제하고 불량 사용자를 관리

# 6. 비지니스 도메인 모델

- 모바일 컨텐츠 미디어 서비스에서 컨텐츠의 상태 전이도 예제




1. 에디터가 컨텐츠를 작성한 후 "제출"을 하면 컨텐츠는 검토중 상태로 변경된다.

2. 편집국이 컨텐츠를 검토한 후 문제가 없으면 검토 완료 처리를 한다.

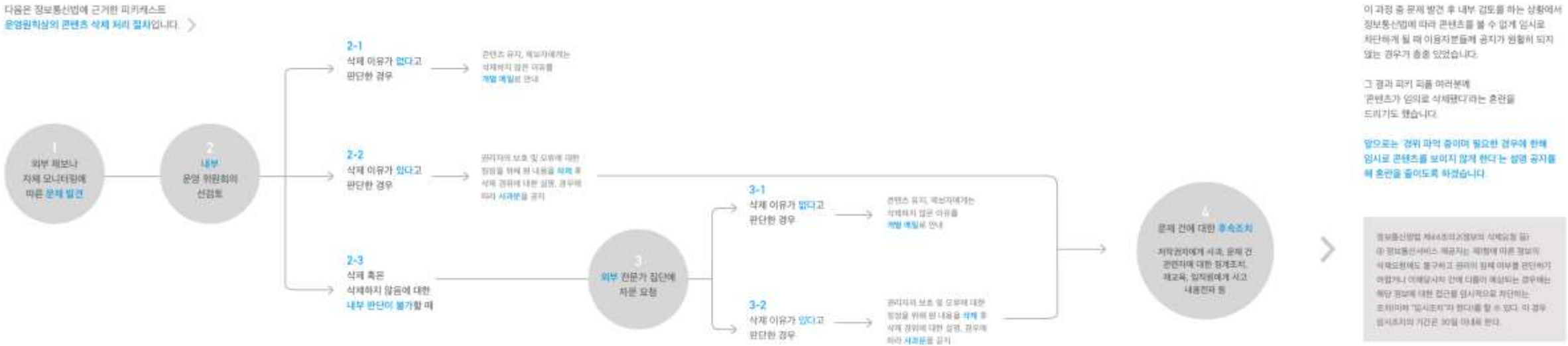
2-2. 컨텐츠에 부적격 내용이 있을 경우 (19세금, 폭력성), 반려 의견을 첨부하여 반려 처리한다.

2-3. 에디터는 반료된 컨텐츠를 통보 받고 수정을 시작 한다.

:

# 6. 비지니스 도메인 모델

- 주요 업무 프로세스 정의 예시 : 피키캐스트 컨텐츠 삭제 처리 절차

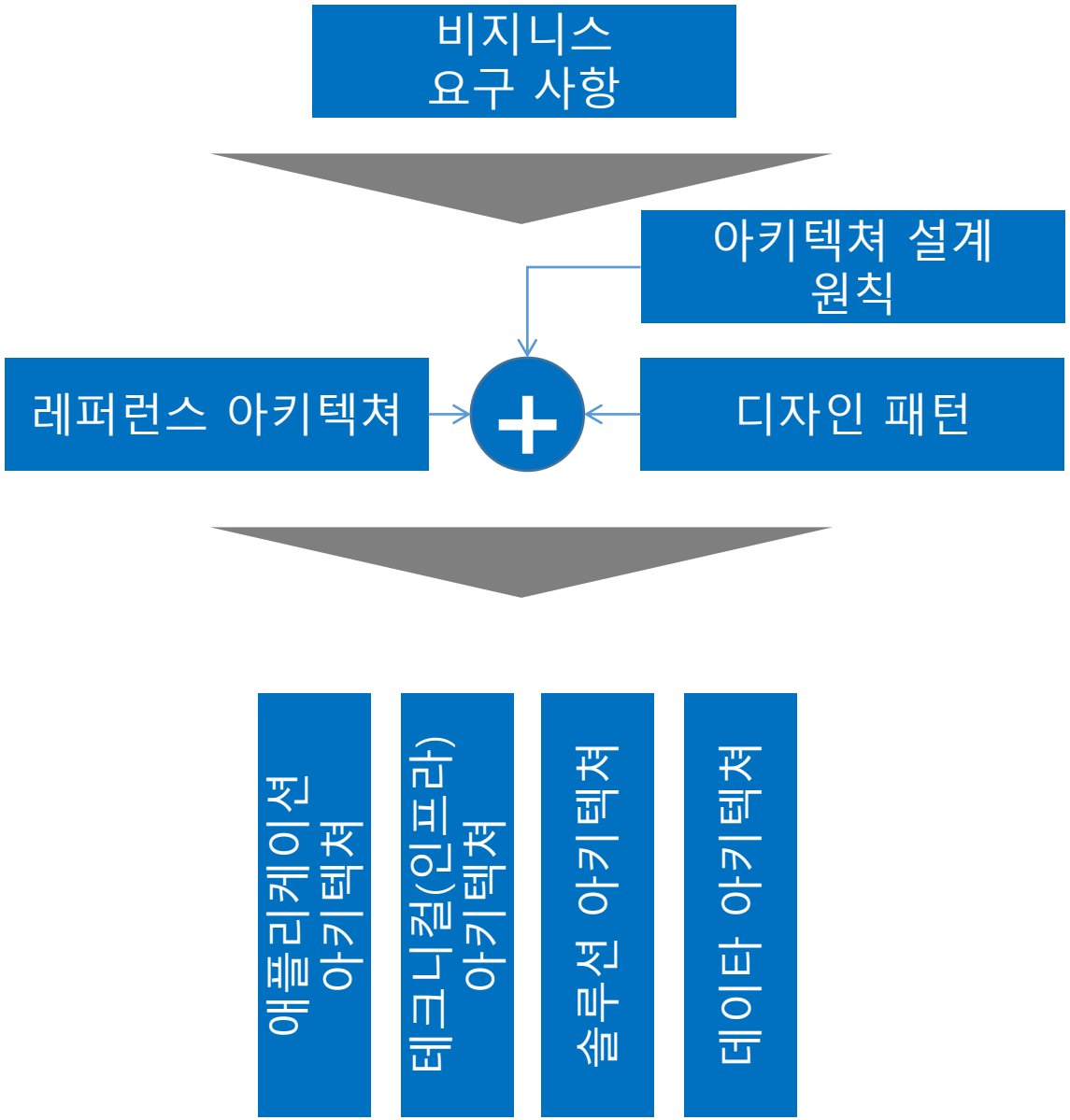



원본 출처 : http://www.pikicast.com/#!/menu=userpage&section=0&uid=16

# #3

시스템 아키텍쳐의 설계

# 시스템 아키텍쳐의 구성




- 아키텍쳐 설계 원칙
- 애플리케이션 아키텍쳐
- 테크니컬(인프라) 아키텍쳐
- 솔루션 아키텍쳐
- 데이타 아키텍쳐

# 아키텍쳐 설계 원칙 (Architecture principals)

- 아키텍쳐 설계의 원칙
- 비용, 비기능적인 설계 원칙
- 디자인 의사 결정이 필요할때 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,PC등 멀티 디바이스를 지원해야 한다.
- 사용자 인터페이스가 사용하기 쉬워야 한다.

:

# 아키텍쳐 설계시 주의 사항

- 아키텍쳐 문서를 만들기 위해서 설계를 하는게 아니라 소통 하기 위해서 하는 것이 아키텍쳐 설계
- 최종 아키텍쳐라는 것은 없다. 계속해서 진화 한다. (Evolutionary architecture : 진화적 아키텍쳐)
- 오버 디자인 주의 (나중에 비지니스가 잘되면 그때 바꾸자)
- 팀 전체가 아키텍쳐를 이해하고 있도록 만드는게 아키텍트의 역할.

  “전체 그림중에서 개발자 A씨의 역할은?”

# 애플리케이션 아키텍쳐

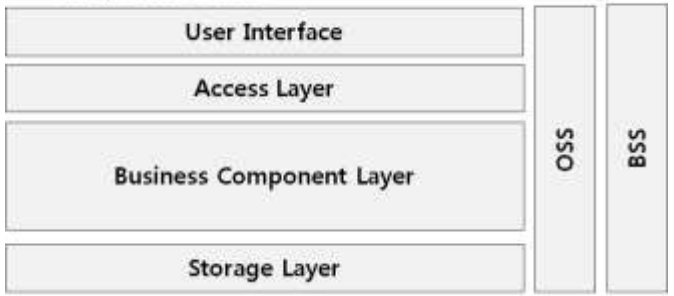
- 소프트웨어 (애플리케이션)에 대한 아키텍쳐 정의
- 구성 요소 : 컴포넌트, 컴포넌트간 관계, 호출 순서, 통신 인터페이스
  - 정적 아키텍쳐 (Static Architecture)
    - 계층 모델
    - 컴포넌트간 관계
  - 동적 아키텍쳐 (Dynamic Architecture)
  - 인터페이스 정의서 (Interface Definition Spec)
  - 상세 아키텍쳐 (Detail Architecture)

# 애플리케이션 아키텍쳐

- 정적 아키텍쳐 (Static architecture) / 계층 모델 정의
  - 애플리케이션을 구성하는 컴포넌트들을 정의
  - 계층(Layer)별로 정의 하여 상세화 (보통 3~4단계가 적절)

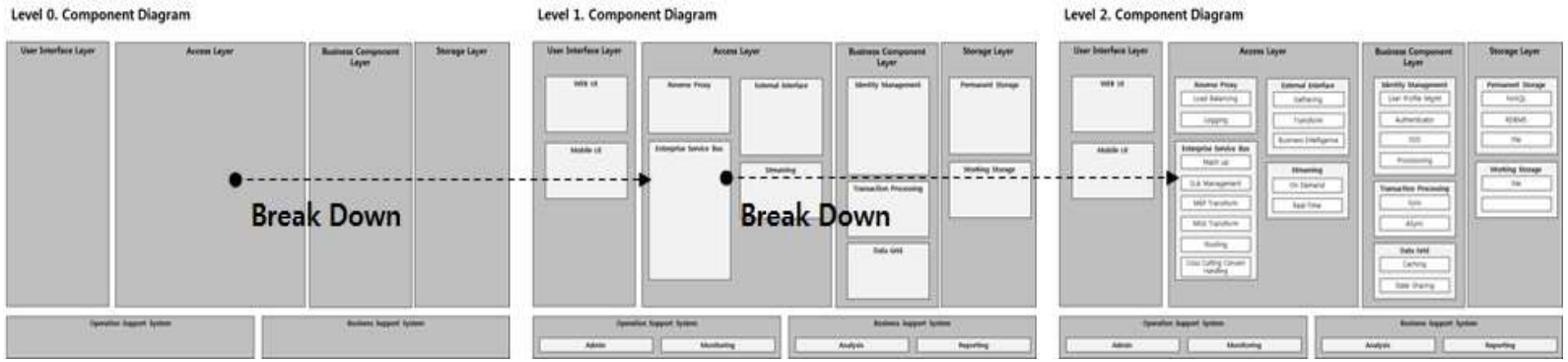
※ 간단해 보이지만 매우 중요함. (향후 시스템의 구조, 팀 구조에 영향)

- Component Level 0



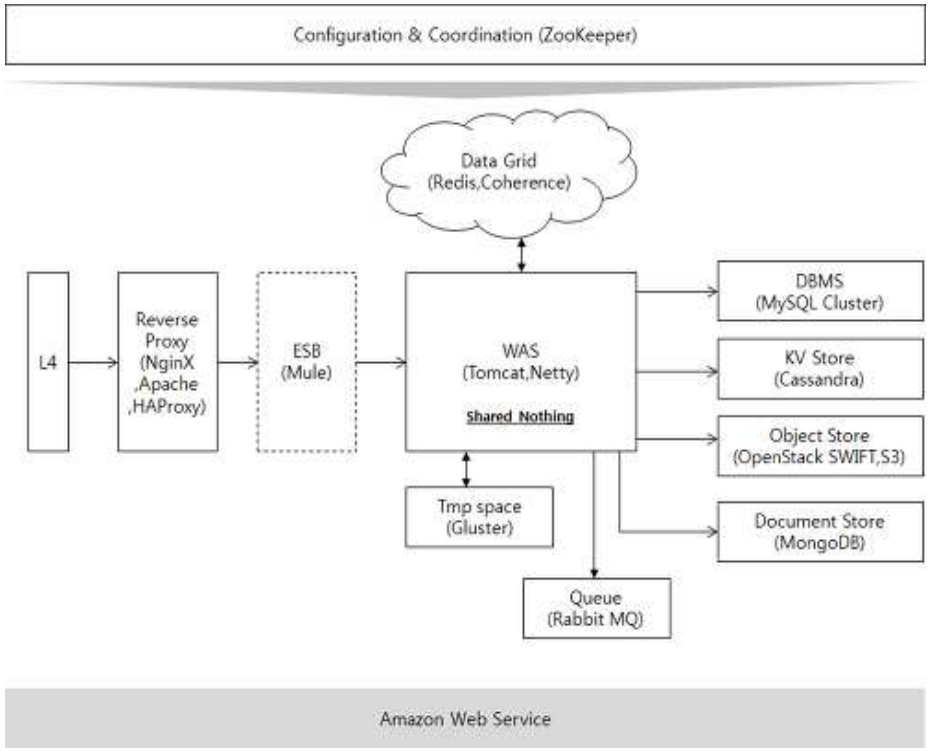

- Component Level 0 Description

| Level 0 | Description |
|---|---|
| User Interface | 웹,모바일등 사용자 인터페이스 계층 |
| Access Layer | 비지니스 로직을 OPEN API를 이용하여 외부로 서비스 하는 계층<br>외부 시스템과의 인터페이스를 제공 |
| Business Component Layer | 비지니스 로직을 구현하는 계층 |
| Storage Layer | 데이타를 저장하는 계층 |
| OSS | 운영,모니터링 관리를 위한 계층 |
| BSS | 비지니스를 위한 지표 분석 리포팅 서비스 |



# 애플리케이션 아키텍쳐

- 정적 아키텍쳐 (Static architecture) / 컴포넌트간의 관계 정의
  - 계층모델에서 정의된 컴포넌트간의 상호 연계성을 정의

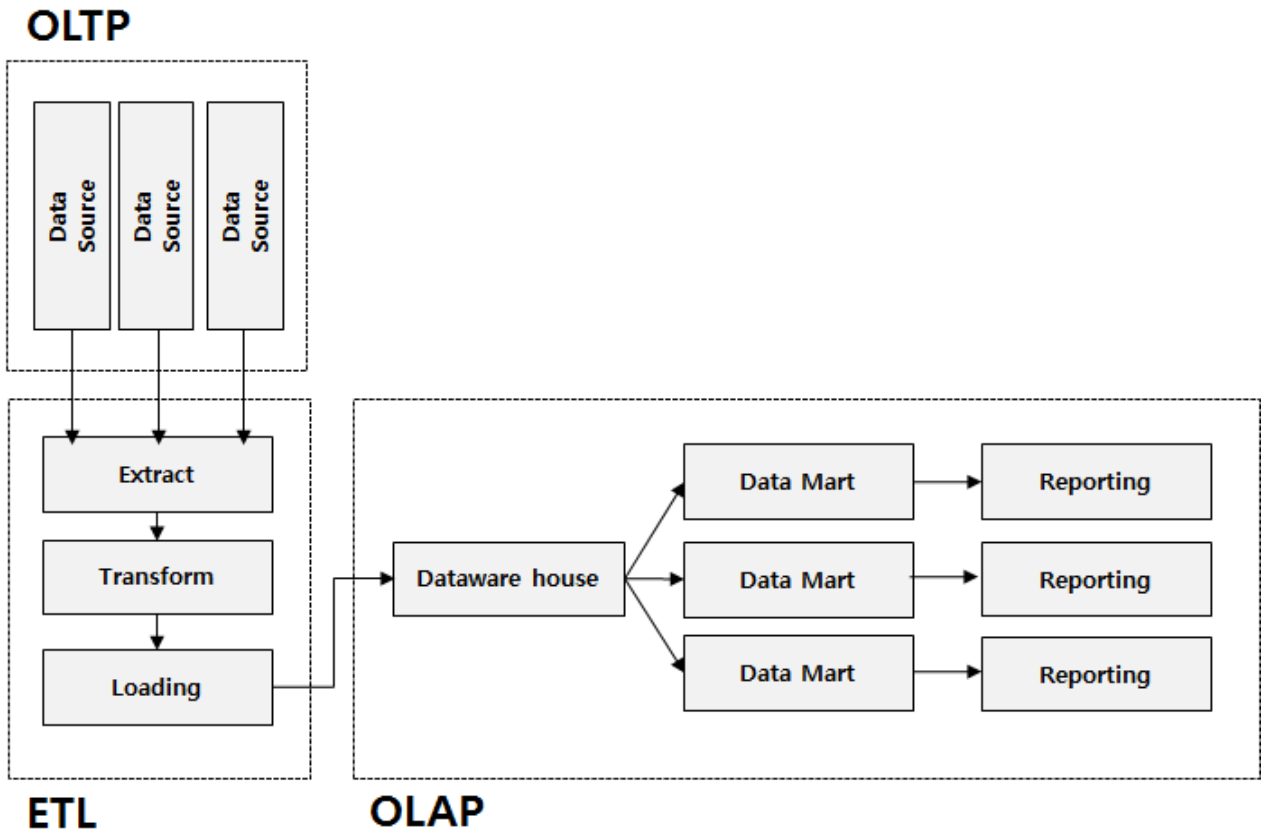



파일 스토리지 서비스 아키텍쳐 예제

# 애플리케이션 아키텍쳐

- 정적 아키텍쳐 (Static architecture) / 컴포넌트간의 관계 정의

  예) OLAP 기반의 분석 시스템 아키텍쳐



# 애플리케이션 아키텍쳐

- 상세 아키텍쳐 (Detail Architecture)
  - 상세한 아키텍쳐에 대한 흐름에 대해서는 별도로 정의

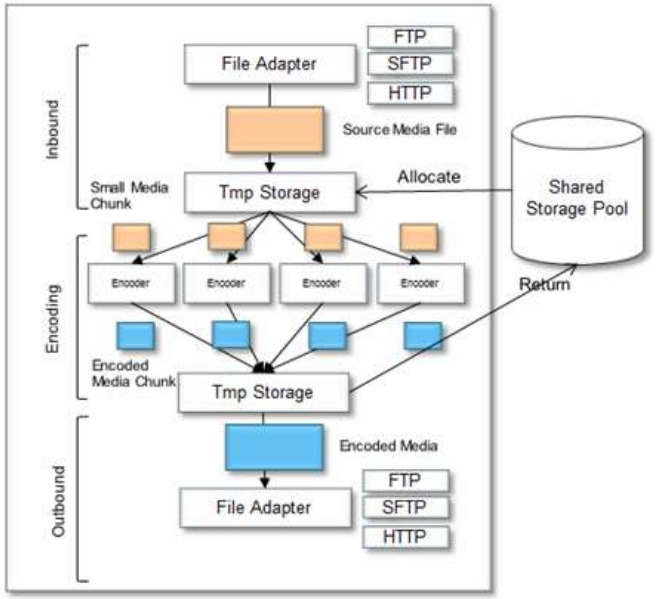
## Transcoding Component Architecture

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 spirited 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.

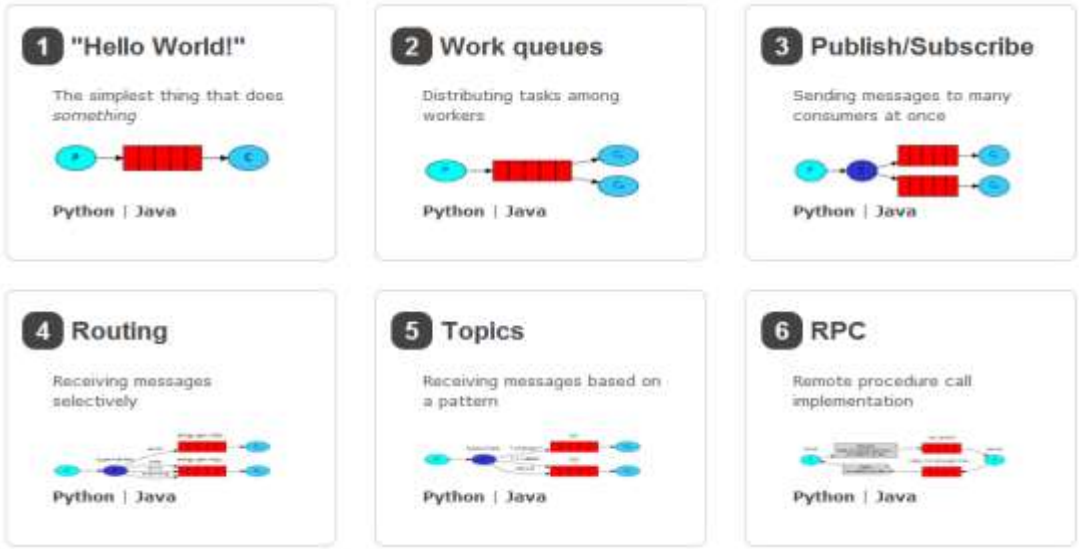
Tmp Storage Consideration

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.



# 애플리케이션 아키텍쳐

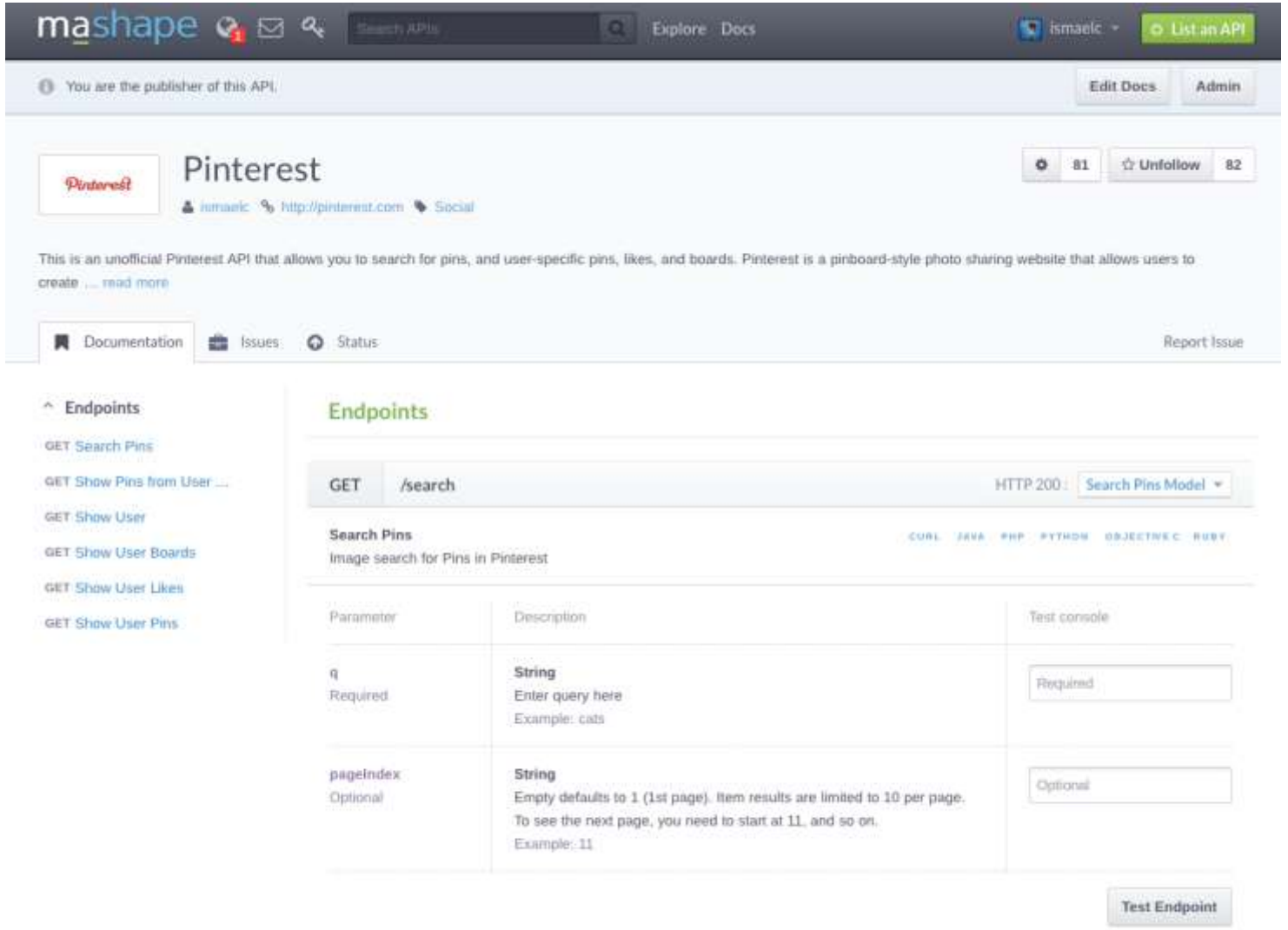
- 인터페이스 정의
  - 프로토콜 정의 (REST, FTP, Google protocol buffer)
  - 메세지 포맷 정의 (REST API 정의서)
  - 메세지 전달 방식 정의 (aka. Message Exchange Pattern. MEP)
    - 동기/비동기
    - ETL 방식
      :




Rabbit MQ에서 정의된 MEP 양식

# 애플리케이션 아키텍쳐

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



# 테크니컬 아키텍쳐

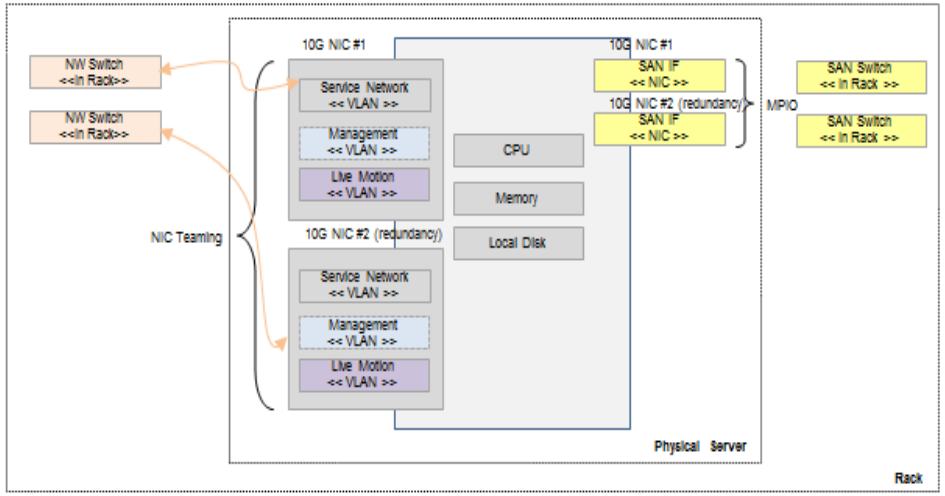
- 하드웨어 아키텍쳐
  - 서버 아키텍쳐
  - 네트워크 구성
  - 스토리지 구성
  - 랙 디자인 구성
  - 글로벌 디플로이 구조
- 솔루션 아키텍쳐
  - 데이타베이스, 미들웨어, 웹서버등의 구성 아키텍쳐

    ※ 클러스터링 등

# 테크니컬 아키텍쳐

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

## ❑ Physical Server Architecture



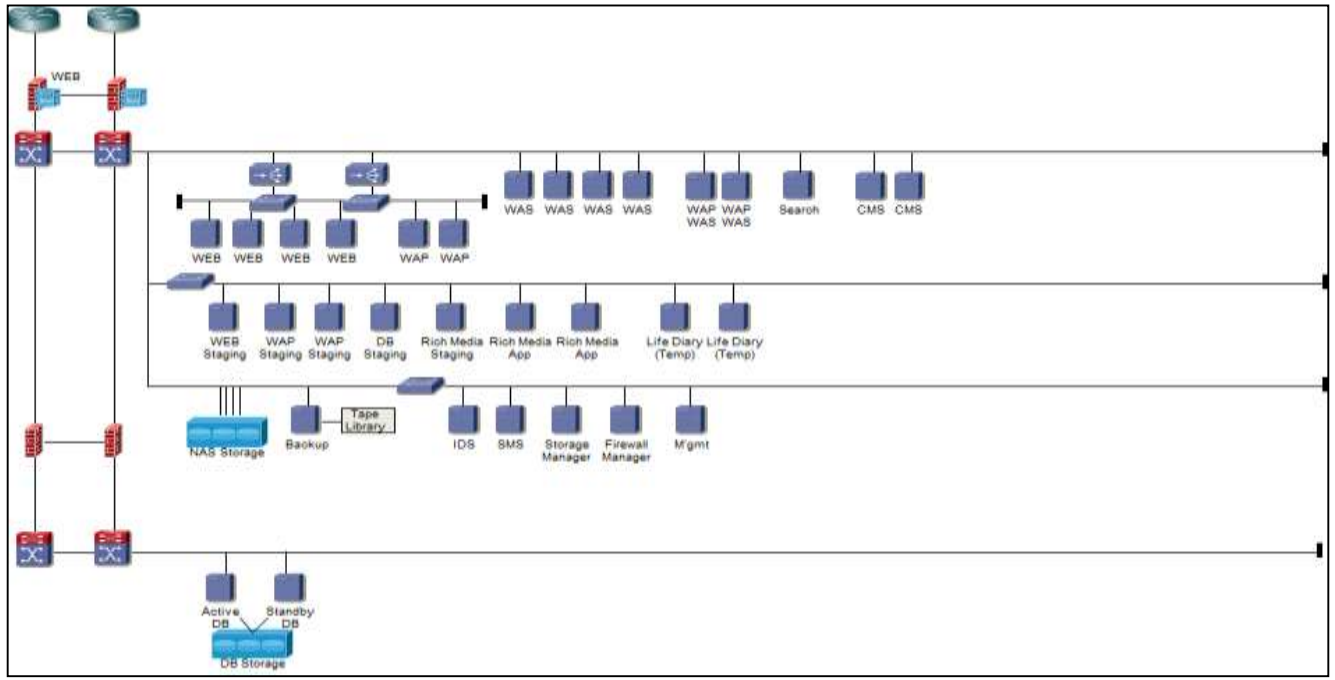

**Physical Server component**

- 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)

클라우드 서버 설계

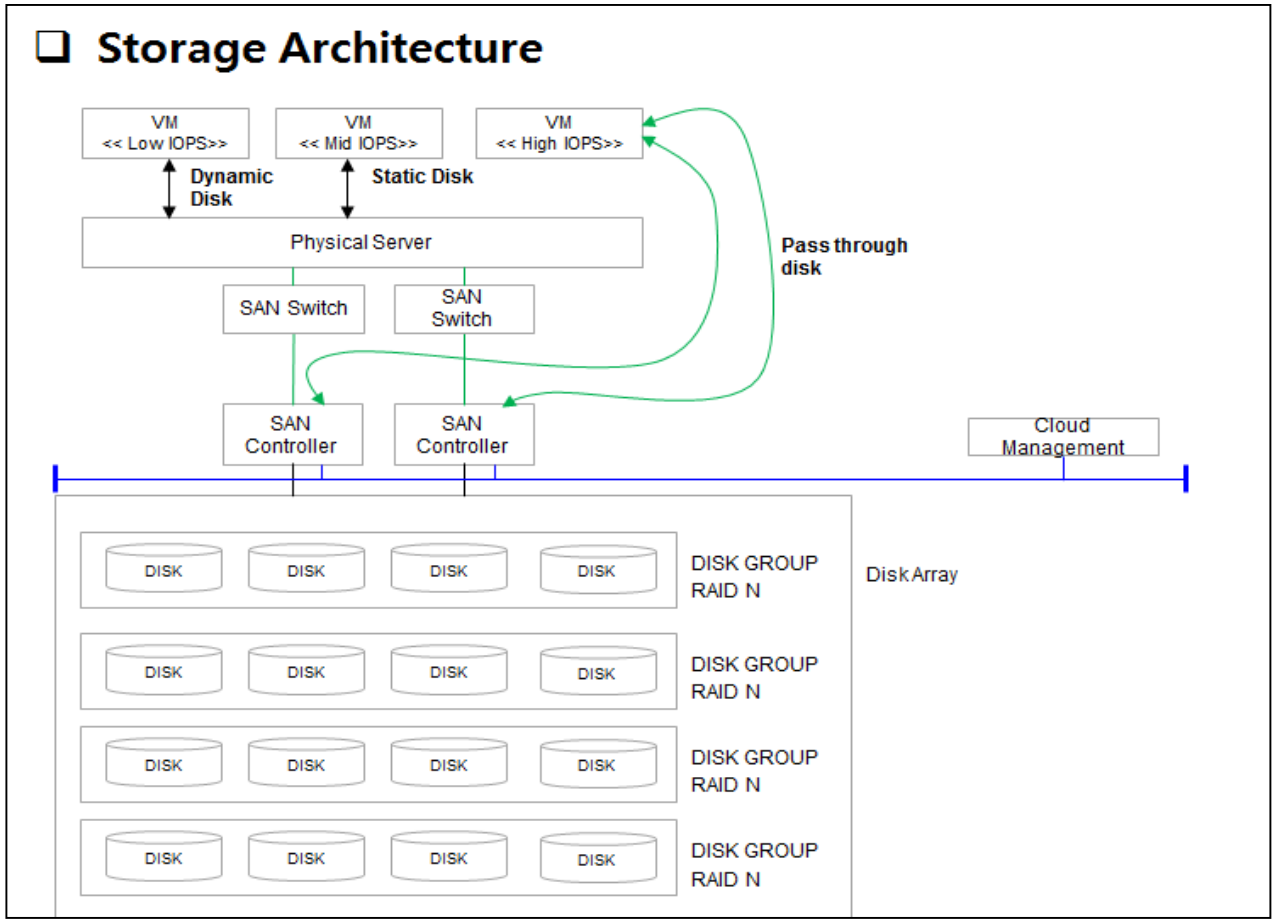
# 테크니컬 아키텍쳐

- 하드웨어 아키텍쳐 / 네트워크 아키텍쳐
  - 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



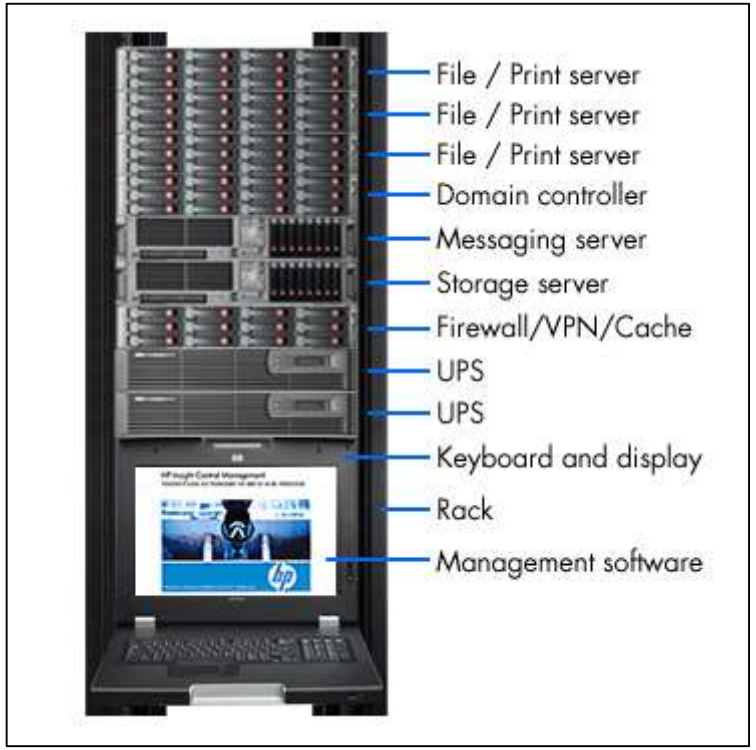
# 테크니컬 아키텍쳐

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



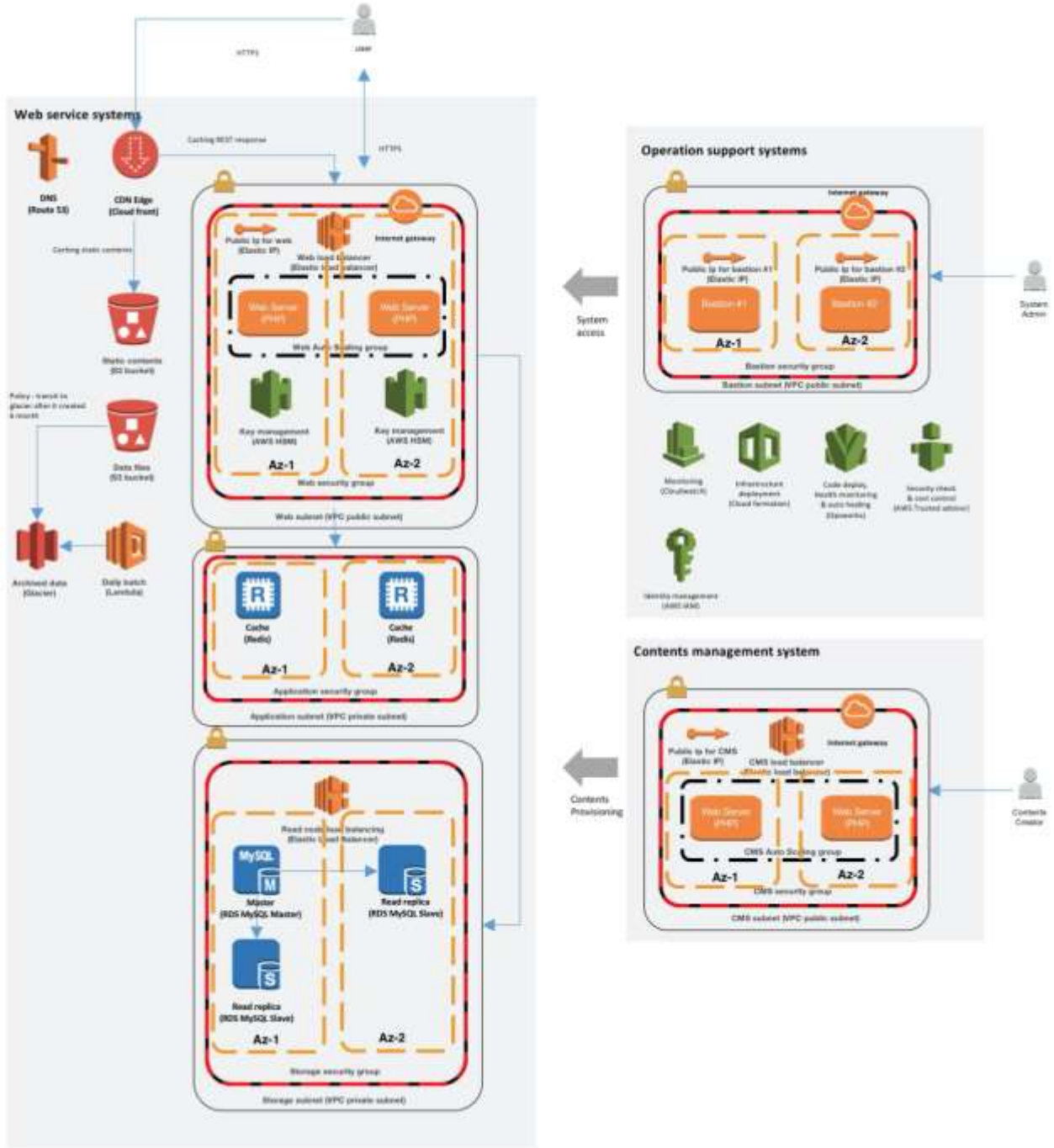
# 테크니컬 아키텍쳐

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



# 테크니컬 아키텍쳐

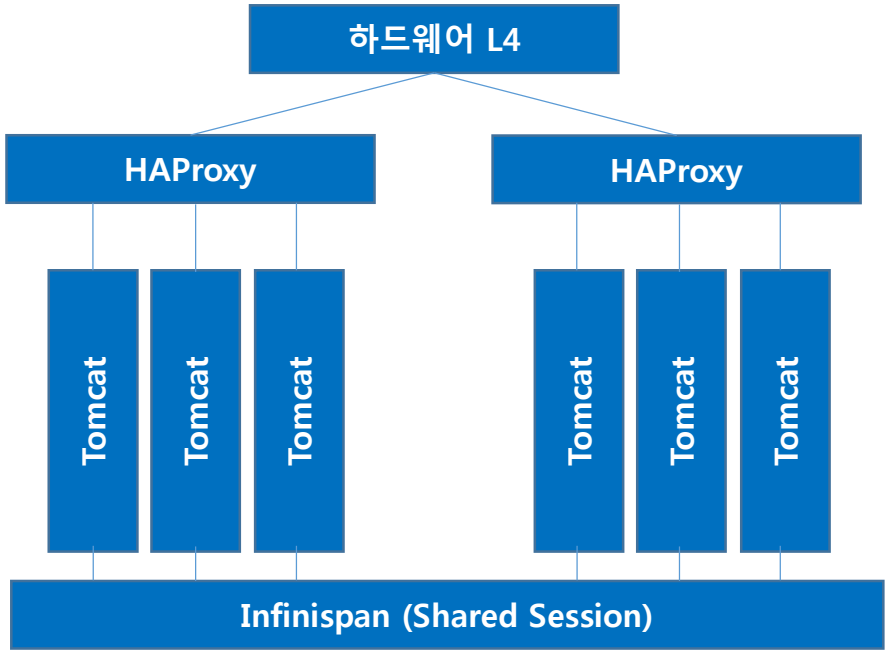
- 아마존 클라우드 기반 배포 모델 설계 예시



# 솔루션 아키텍쳐

- 솔루션 아키텍쳐
  - 데이타 베이스, 미들웨어등의 구성과 배포 구조 정의

※ 특히 클러스터링,로드 밸런싱, HA 구조 등에 대한 정의




Tomcat 클러스터 배포 구조

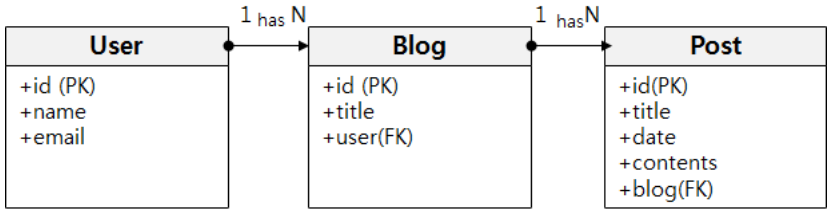
# 데이타 아키텍쳐

- 시스템에 저장되는 데이타에 대한 아키텍쳐 정의
  - 데이타 구조
  - 저장 장소 및 솔루션
  - 보안 처리 아키텍쳐 (암호화등)
  - 데이타 생명 주기 관리 (생성, 백업, 폐기까지의 정책)

# 데이타 아키텍쳐

- 데이타 구조
  - Conceptual Modeling


  - Logical Modeling



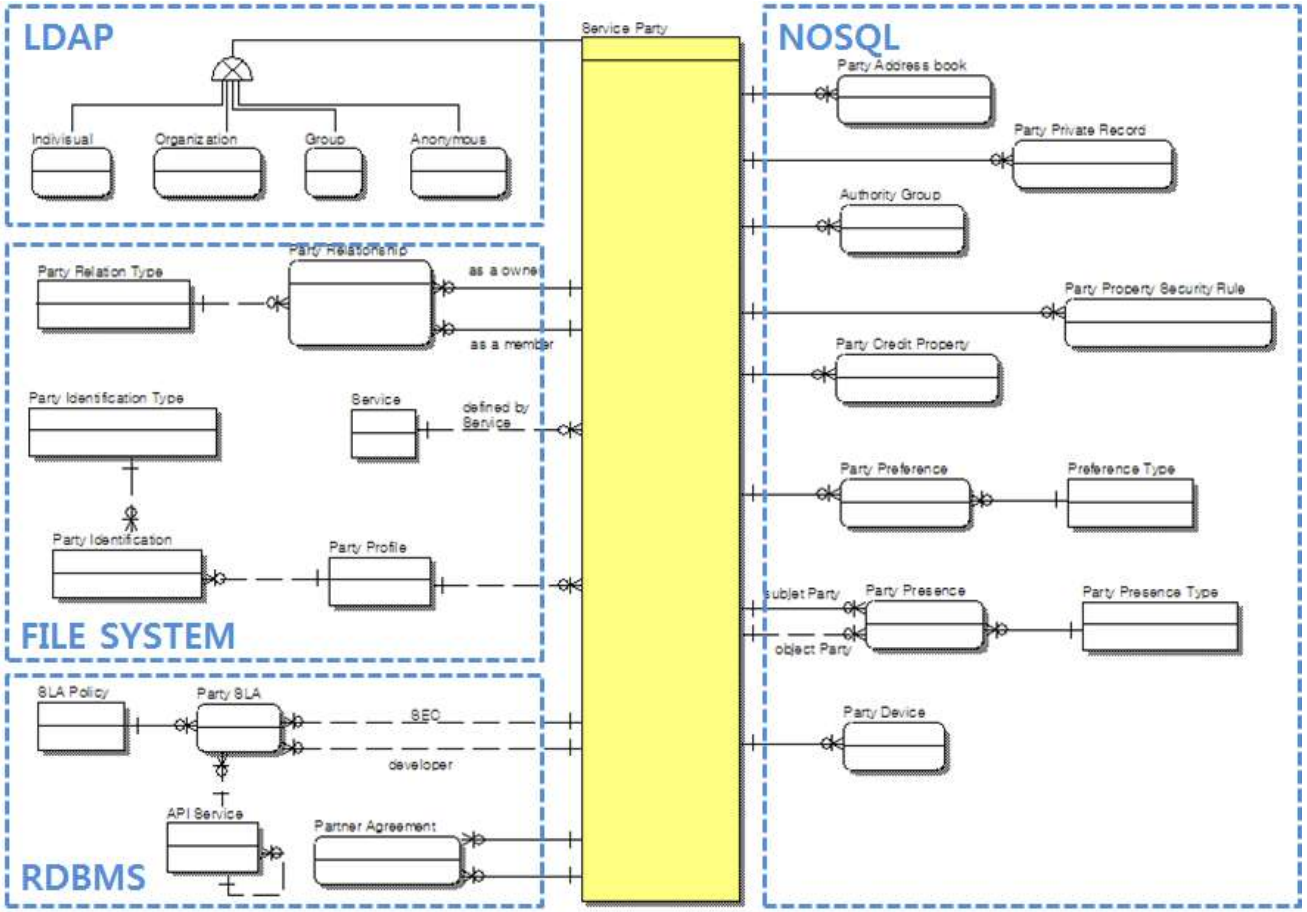

  - Implementation Modeling


**TABLE : USER**

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

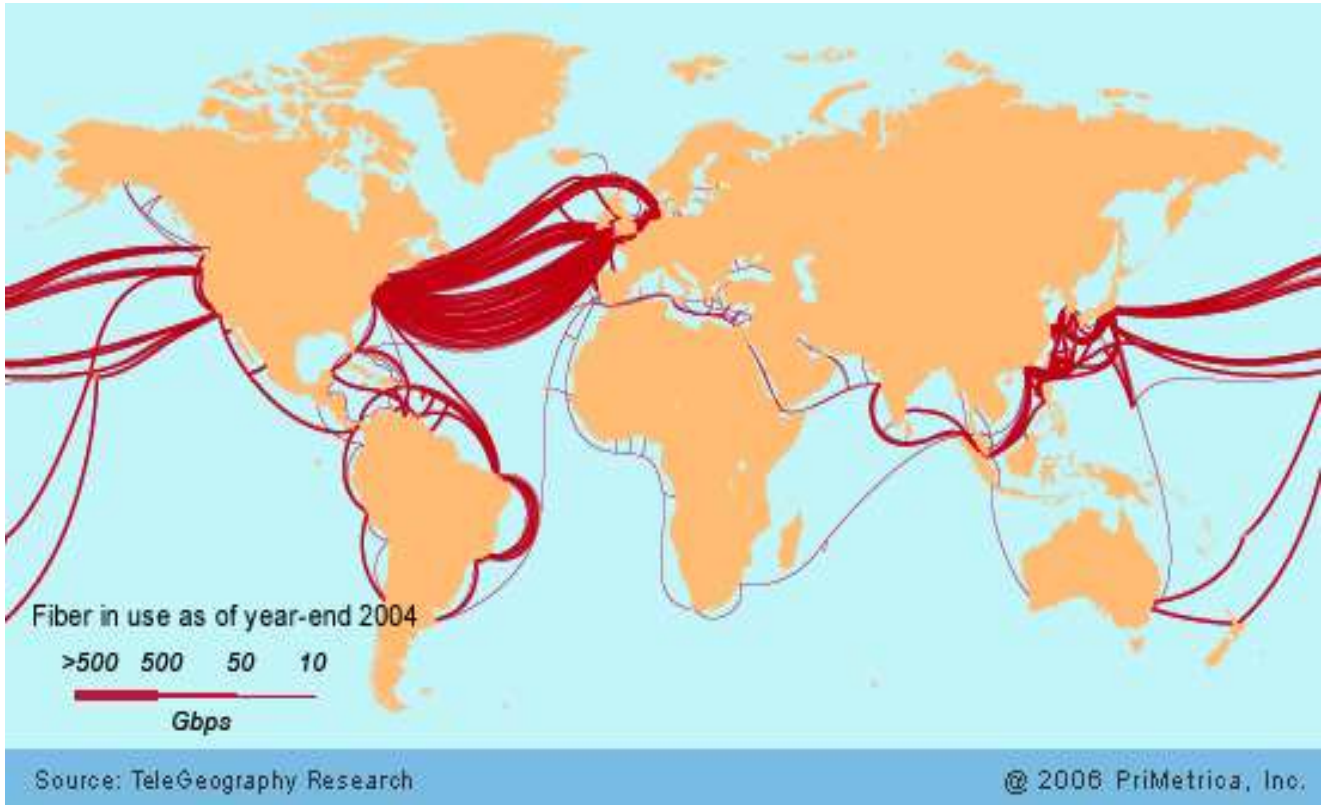
# 테크니컬 아키텍쳐

- 데이타 아키텍쳐 / 배포 구조
  - 데이타 저장소
    - 정의된 데이타 모델에 대한 솔루션별 저장소 아키텍쳐 정의



# 테크니컬 아키텍쳐

- 글로벌 ROLL OUT 아키텍쳐
  - 데이타 센타간 시스템 배포 구조 및 연동 구조, 데이타 복제 방식등을 정의

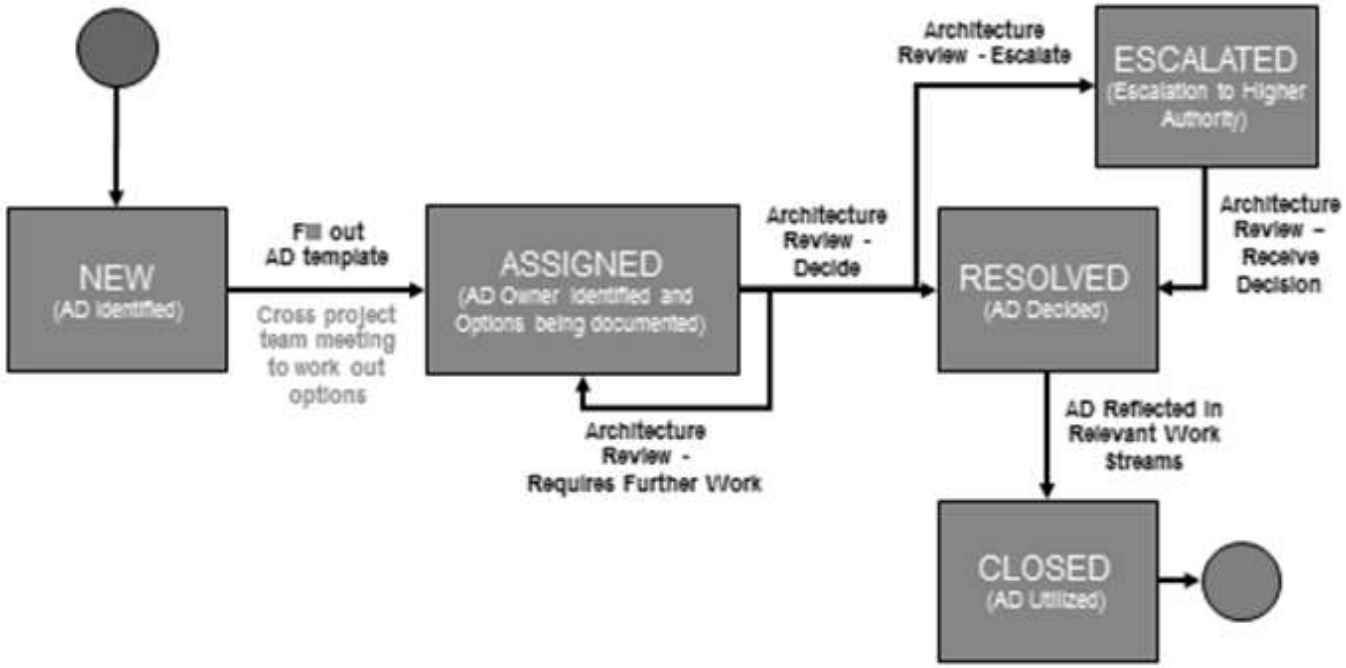


# #4

아키텍쳐 의사 결정 프로세스

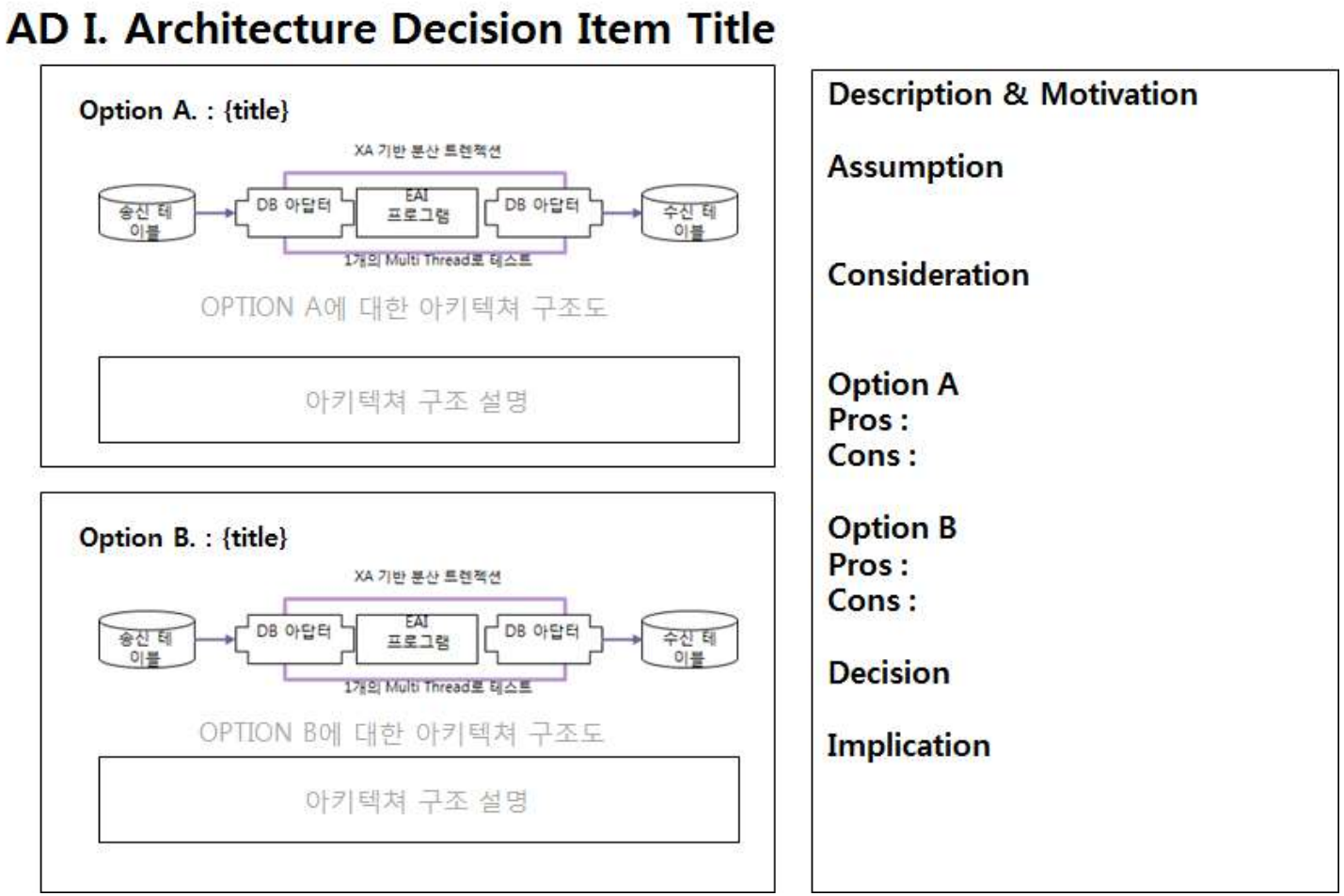
# 아키텍쳐 의사 결정 프로세스

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



# 아키텍쳐 의사 결정 템플릿 (AD 템플릿)

- 각 옵션별 아키텍쳐에 대한 설명
- 옵션별 장단점, 의사 결정 결과, 임팩트등을 한장에 설명
- 잘 모아 놓는게 중요 (나중에 딴소리 또는 왜 그렇게 했는지 설명)



End of document