Part 3.

아키텍쳐 설계 방법론

비즈니스 요구 사항을 어떻게 분석하여, 시스템 아키텍처로 옮겨 내는지에 대해서 알아본다.
이 과정에서, 아키텍트의 역할과 협업 그리고 어떻게 의사 결정에 관여 하는지를 알아본다.

소프트웨어 아키텍처란?

아키텍처의 정의

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

> “아키텍처는 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”

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

아키텍처 와 디자인 프로세스

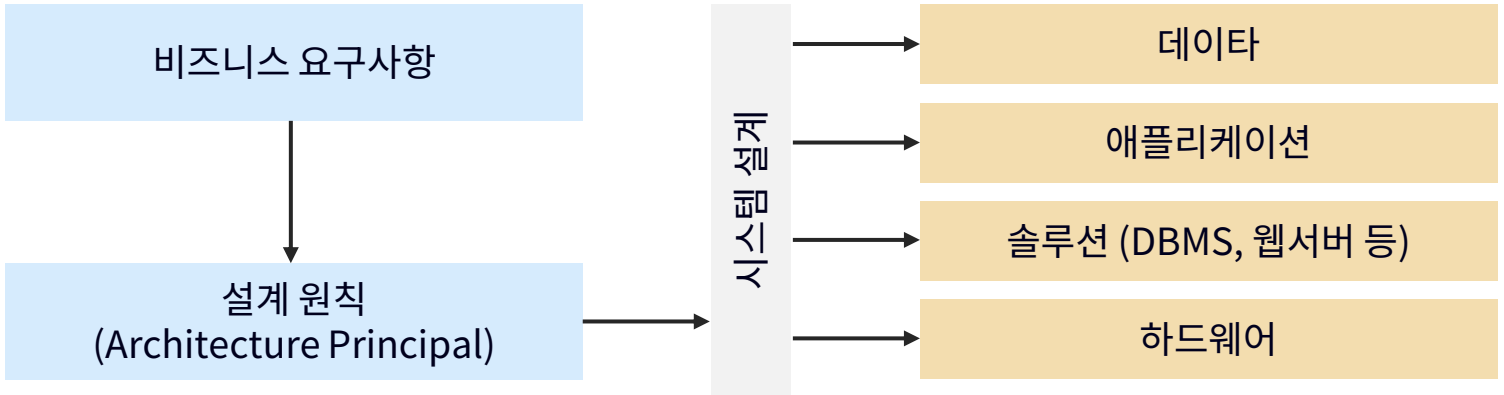
- **소통 효율 증가** : 프로세스 가이드라인에 따라서, 같은 뷰와 같은 언어로 소통을 하기 때문에, 소통 오류가 줄어듬
- **위험도 감소** : 구조화된 프로세스로 설계하기 때문에, 그 과정에서 문제점을 조기에 발견할 수 있음
- **효율성 증가** : 가이드된 프로세스 순서에 따르기 때문에, 시간이 절약됨
- **품질 증가** : 이미 검증된 프로세스와 프레임웍을 사용하기 때문에, 문제 가능성이 적음

“가이드"로 사용하고, 정책으로 사용하지 않는 것을 권장함

아키텍처 설계 프로세스

오늘 부터 우리가 배울 아키텍쳐 설계 방법론은?

- Zachman 프레임웍, Federal enterprise architecture 등 여러 방법론이 있음
- 우리는 토가프(TOGAF) 아키텍처 프레임웍을 축소한 버전을 사용

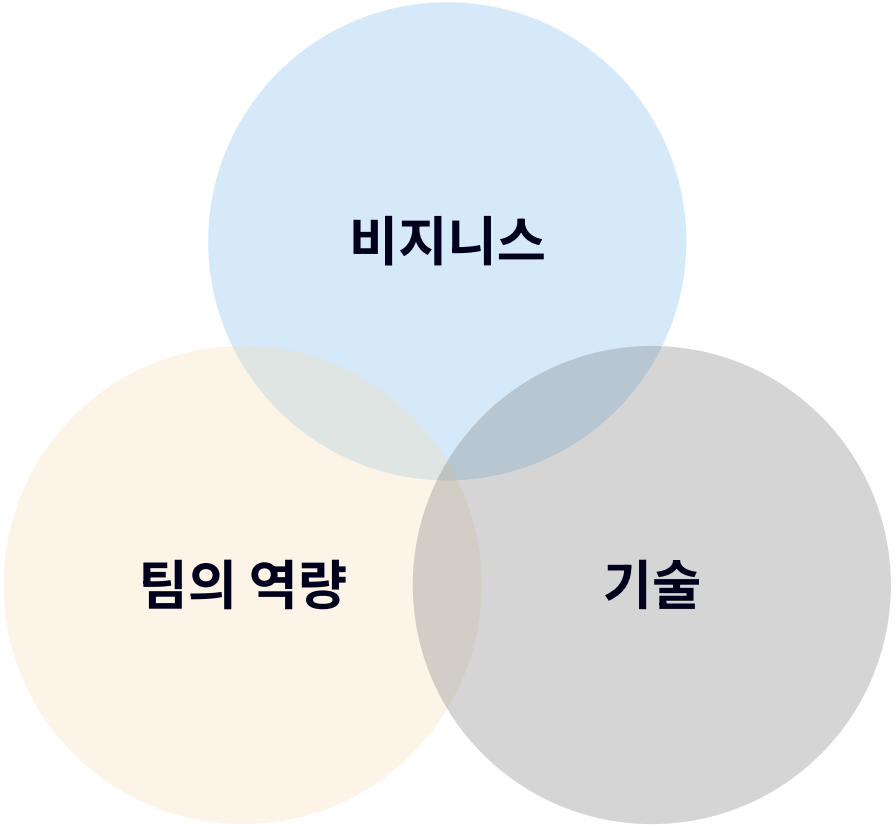


아키텍트

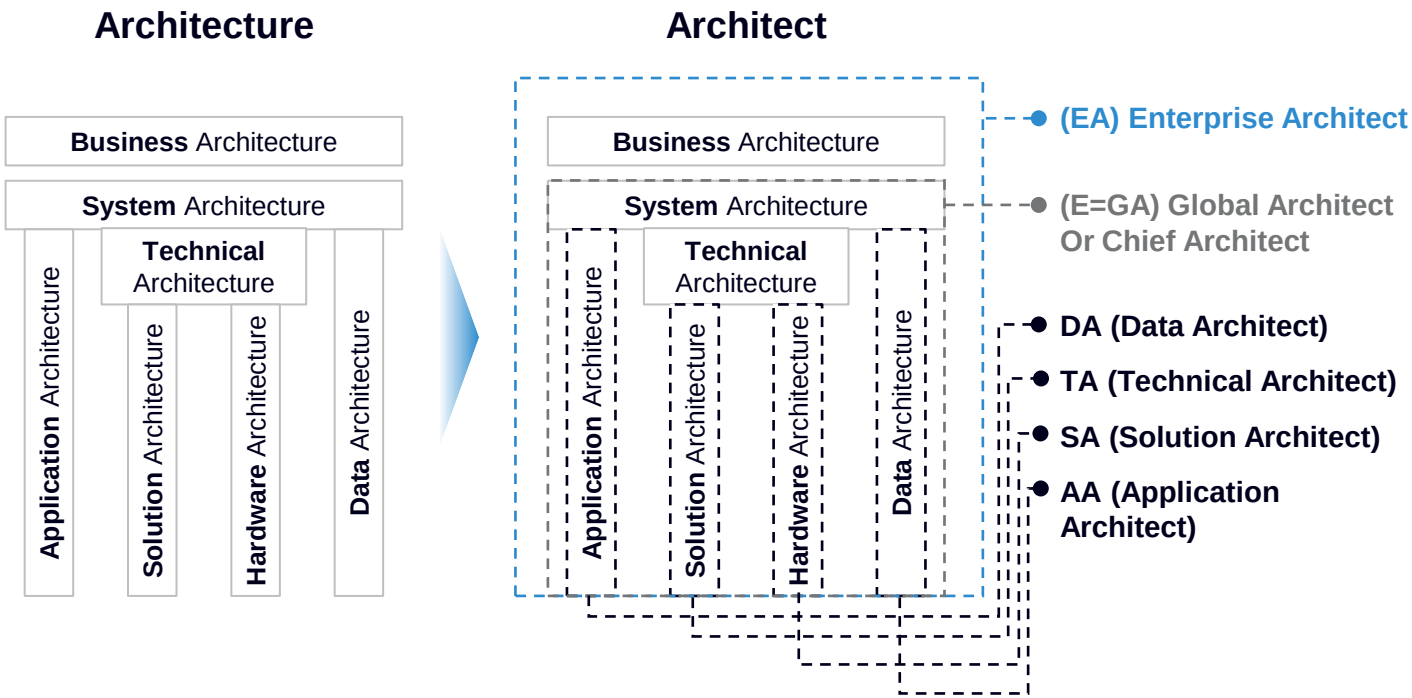
아키텍처를 그리는 사람

통역사
”비지니스 언어를 기술 언어로”

방향 지시자
“개발의 방향, 시스템의 청사진을 제안”



아키텍트의 종류


Architecture
Business Architecture
System Architecture
Technical Architecture
Application Architecture
Solution Architecture
Hardware Architecture
Data Architecture
Architect
Business Architecture
System Architecture
Technical Architecture
Application Architecture
Solution Architecture
Hardware Architecture
Data Architecture
(EA) Enterprise Architect
(E=GA) Global Architect
Or Chief Architect
DA (Data Architect)
TA (Technical Architect)
SA (Solution Architect)
AA (Application Architect)

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

애자일팀화가 되어감에 따라 작은 팀

민첩한 조직의 경우

- **TL** (Tech Leader)
- **PM** (Product Manager)

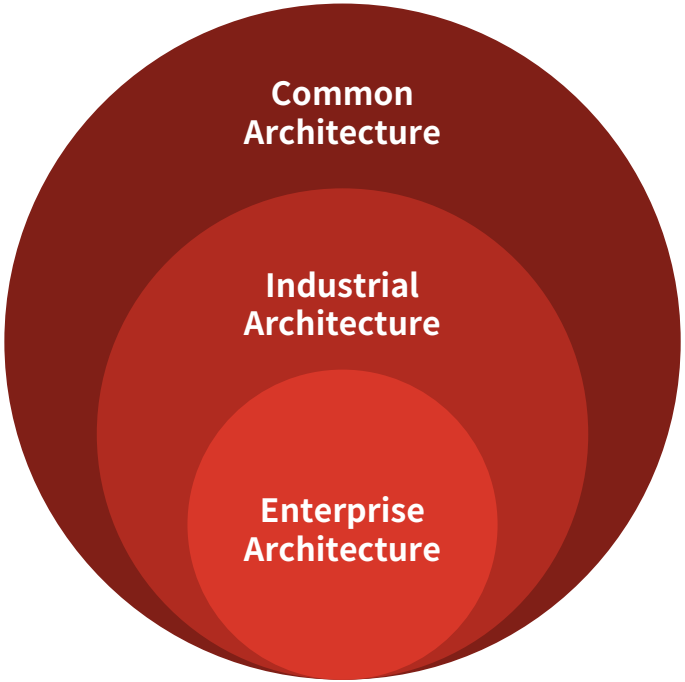
좋은 아키텍트에게 필요한 능력

- 커뮤니케이션
- 추상화 능력
- 비즈니스에 대한 이해
- 기술에 대한 깊은 이해 / 핸즈온 (코딩)
- 문제 해결 능력

비지니스 요건 팀의 능력, 기술 등은 항상 변한다.
(아키텍처는 항상 변화한다.)

듣는 기술/공감하는
+
문제 해결 능력
+
쉽게 설명하는 기술
+
프로젝트 관리 능력
+
함께 협업하는 기술

레퍼런스 아키텍처



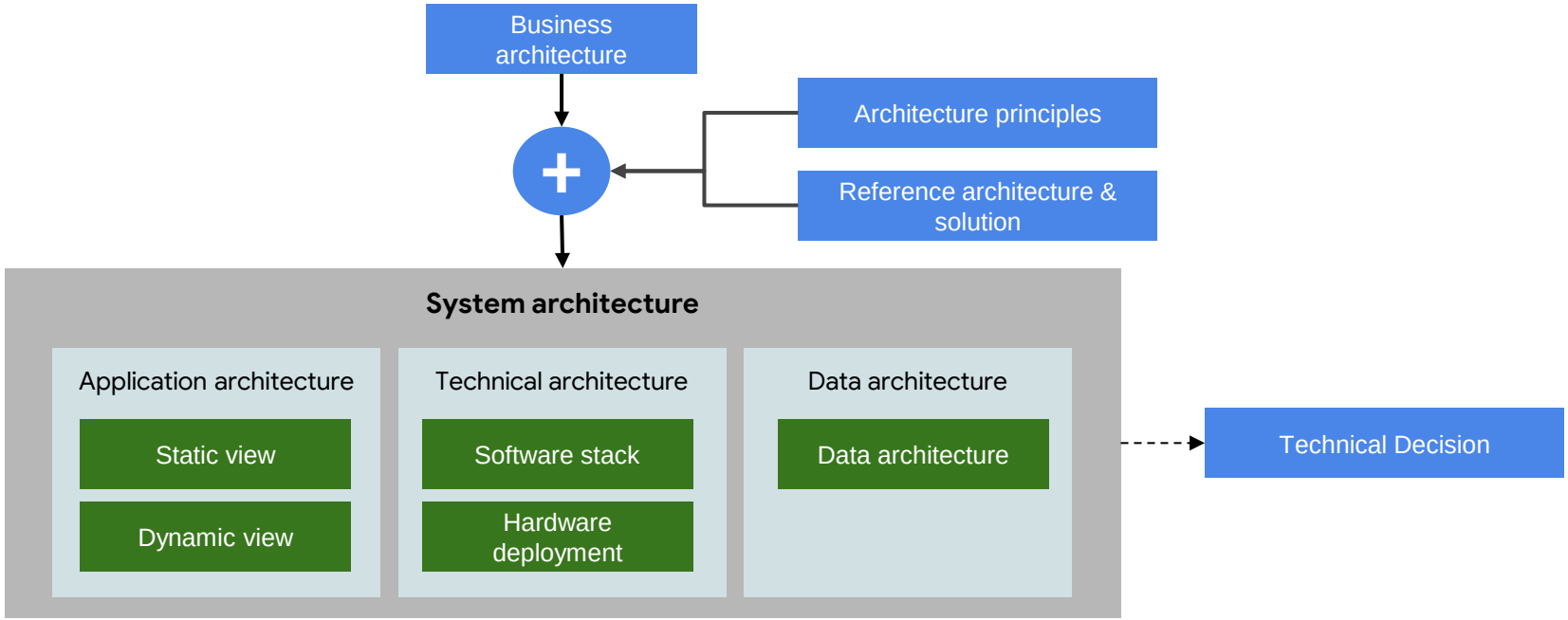
아키텍처 설계에 참고할 수 있는 아키텍처

- **Common** Architecture: 업무 도메인 종속성이 없음 – SOA,REST 등
- **Industry** Architecture : 특정 업무 도메인 종속성 – MES,PLM 등
- **Enterprise** Architecture : 특정 회사의 아키텍처

비즈니스 아키텍처 설계

시스템 아키텍처의 구성

토카프 (TOAF) 방법론 기반의 아키텍처 디자인 프레임워크



비즈니스 이해

소프트웨어를 개발하기 전에, 구현하고자 하는 시스템의 요구 사항과 비즈니스 컨텍스트를 이해한다.

- 풀고자 하는 문제가 무엇인가?
- 서비스가 추구하고자 하는 가치
- 서비스가 돈을 버는 방법
- 리소스 (개발 기간, 인력 자원)

비즈니스 구성원에 대한 이해

모바일 콘텐츠 서비스의 사용자 정의 예제

사용자

서비스를 통해서
컨텐츠를 보거나
댓글을 다는 사람

에디터

서비스에
컨텐츠를 제작해서
업로드 하는 사람

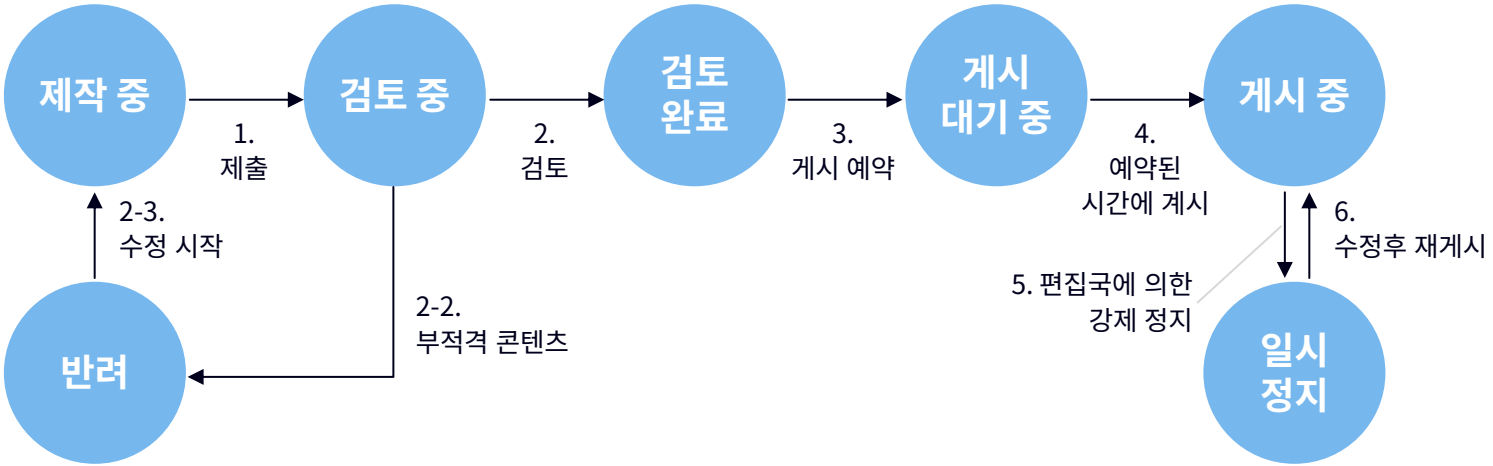
편집국

에디터에 의해 업로드 된
컨텐츠를 검수하여,
퍼블리싱 하는 사람

댓글 운영자

댓글을 모니터링 하여
불량 댓글을 삭제하고
불량 사용자를 관리

업무 프로세스에 대한 이해



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.

Fastcampus

업무 프로세스에 대한 이해

주요 업무 프로세스 정의 예시 : 피키캐스트 콘텐츠 심사 절차

기능 목록

개발하고자 하는 제품의 기능과 그 우선순위 (UI 목업을 첨부하기도함)

Feature (Lv 1)	Feature (Lv 2)	Details
사용자 가입 및 로그인	사용자는 구글 계정으로 로그인 한다.	사용자는 구글 계정으로 로그인한다.
	사용자는 페이스북 계정으로 로그인한다.	사용자는 페이스북 계정으로 로그인한다
비디오 피드 리스트 보기	사용자는 최근 업데이트된 순서대로 비디오 피드를 본다	사용자는 상하로 스크롤되는 비디오 피드를 볼 수 있다. 비디오 피드는 1초간 스크롤하지 않으면 현재 화면에 있는 비디오가 미리보기로 플레이된다.
비디오 촬영 및 업로드	사용자는 암벽등반하는 장면을 비디오로 촬용한다	사용자는 암벽 등반하는 비디오를 앱에서 촬영할 수 있다.
	사용자는 촬영한 비디오를 업로드할 수 있다.	사용자는 촬영한 비디오를 촬영이 끝나자 마자 업로드하거나 또는 비디오 갤러리에서 이미 촬영된 비디오를 선택하여 업로드한다. 사용자는 비디오 업로드시 제목과,설명을 추가하여 업로드한다. 업로드시에 위치 정보를 같이 업로드 한다. GPS 정보를 이용하여 위치를 업로드 하거나 또는 근처에 이미 등록된 암벽등반장이 있을 경우, 암벽등반장 이름을 목록으로 보여주고 선택할 수 있게 한다.
	사용자는 비디오를 업로드할때 해쉬태그를 추가할 수 있다.	사용자는 비디오 등록시, 설명 부분에 해쉬태그를 추가할 수 있다.
	사용자는 비디오 등록시 암벽등반장 등록할 수 있다.	사용자는 비디오 등록시 암벽등반장을 GPS 정보를 이용하여 등록할 수 있다. 암벽등반장은 이름과 위치로 등록하며, 이미 다른 사람이 등록했을 경우 유사한 이름의 등반장은 하나로 합친다.
비디오 보기	사용자는 피드에서 비디오를 선택해서 플레이할 수 있다.	사용자가 비디오 피드에서 비디오른 클릭하면 비디오가 재생된다.
	사용자는 비디오를 플레이중에 비디오에 대한 댓글을 달 수 있다.	사용자는 비디오를 재생하는 화면에서 댓글을 달아서 자신의 의견을 등록할 수 있다.
	사용자는 비디오의 댓글중에 부적절한 댓글을 신고할 수 있다.	사용자는 비디오에 적절하지 않은 댓글이 있을 경우 신고하기 버튼을 이용하여 관리자에게 신고하고 삭제를 요청할 수 있다. 신고된 댓글은 관리자의 판단이 있을때까지 내용을 표시하지 않고 대신, "신고된 댓글입니다."라고 표시된다.
	사용자는 부적절한 비디오를 신고할 수 있다.	사용자는 비디오 재생하기나 또는 목록보기에서 적절하지 않은 비디오를 신고할 수 있으며, 신고에 대한 내용을 간단한 메모로 작성할 수 있다.
	사용자는 비디오를 추천할 수 있다.	사용자는 비디오를 재생하면서 마음에 드는 비디오에 "추천하기" 버튼을 누를 수 있다.
비디오 검색	사용자는 비디오 검색을 위해서 사진을 촬영할 수 있다.	사용자는 유사한 암벽등반 코스를 검색하기 위해서, 실내 암벽등반 코스를 사진을 촬영할 수 있다.
	사용자는 촬영한 사진을 이용하여 유사한 암벽의 비디오를 검색할 수 있다.	사용자는 이미 갤러리에 저장된 실내 암벽등반 코스나 또는 방금 촬영한 암벽등산 코스 사진을 이용하여, 유사한 코스의 비디오를 검색할 수 있다. 일반 사용자는 일주일에 10개의 비디오를 사진 검색을 통하여 검색할 수 있으며, 유료 사용자는 무제한으로 검색을 할 수 있다.
	사용자는 비디오를 검색할때, 현재 위치와 유사한 위치에 있는 비디오를 검색할 수 있다.	사용자가 사진 기반으로 검색할때, 유사한 비디오를 사진뿐만 아니라, 사용자가 GPS를 활성화한 경우, GPS 정보를 이용하여, 반경 2KM내에서 촬영한 영상을 우선으로 검색한다.
	사용자는 비디오를 검색할때 검색어를 추가하여 검색할 수 있다.	사용자는 검색시, 필요하면 검색 키워드를 추가하여 검색할 수 있다. 이 검색 키워드는 비디오 설명에 있는 텍스트와 해쉬 태그를 검색한다.
저장하기	유료 사용자는 자신이 마음에 드는 비디오를 북마크식으로 저장해놓을 수 있다.	유료 사용자는 자신에게 유용한 비디오를 북마크를 통해서 저장할 수 있다.
	유료 사용자는 북마크에 저장된 비디오의 목록을 볼 수 있다.	유료 사용자는 자신이 북마크한 비디오 목록을 볼 수 있다.

PRD (Product Requirement Description)

개발하고자 하는 제품의 목적, 기능, 특징을 명확하게 정의한 문서

1. **제품 개요**: 제품이 해결하려는 문제나 목표를 설명한다.
2. **대상 사용자**: 제품을 사용할 주된 사용자 그룹을 정의한다.
3. **핵심 기능**: 제품이 가져야 할 주요 기능과 그 우선순위를 명시한다.
4. **성능 요구사항**: 속도, 확장성, 신뢰성 등 제품이 충족해야 할 성능 기준을 설정한다.
5. **디자인 요구사항**: UI/UX 관련 요구사항이 들어가며, 사용자 인터페이스나 사용자 경험에 대한 가이드라인을 포함할 수 있다.
6. **기술적 요구사항**: 제품이 어떤 기술 스택이나 플랫폼을 사용해야 하는지에 대한 정보를 제공한다.
7. **제약 조건**: 예산, 일정, 기술적 한계 등 제품 개발 과정에서 고려해야 할 제약을 명시한다.

시스템 아키텍처 설계

아키텍처 설계 원칙 (Architecture Principles)

- 아키텍처 설계의 원칙
- 비기능 (안정성, 비용, 구현 기간 등)
- 설계 과정에서 의사 결정이 필요할 때, 의사 결정의 기준이 됨
- 3~5개 정도

실내 암벽 등반앱

- **운영 비용을 최소화 한다.**
- **쾌적하고 빠른 앱 경험을 제공한다.**
- **고객의 피드백을 빠르게 반영할 수 있도록 한다. (개발 속도)**

아키텍처 설계시 주의사항

- 아키텍처 문서를 만들기 위해서 설계를 하는게 아니라 소통 하기 위해서 하는 것이 아키텍쳐 설계
- 최종 아키텍처 라는 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?”

애플리케이션 아키텍처

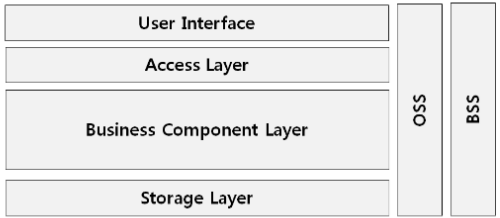
- 소프트웨어 (애플리케이션)에 대한 아키텍처 정의
- 구성 요소 : 컴포넌트, 컴포넌트간 관계, 호출 순서, 통신 인터페이스
  - 정적 아키텍처 (Static Architecture)
    - 계층 모델
    - 컴포넌트간 관계
  - 동적 아키텍처 (Dynamic Architecture)
  - 인터페이스 정의서 (Interface Definition Spec)
  - 상세 아키텍처 (Detail Architecture)

애플리케이션 아키텍처

정적 아키텍처 (Static architecture) / 계층 모델 정의

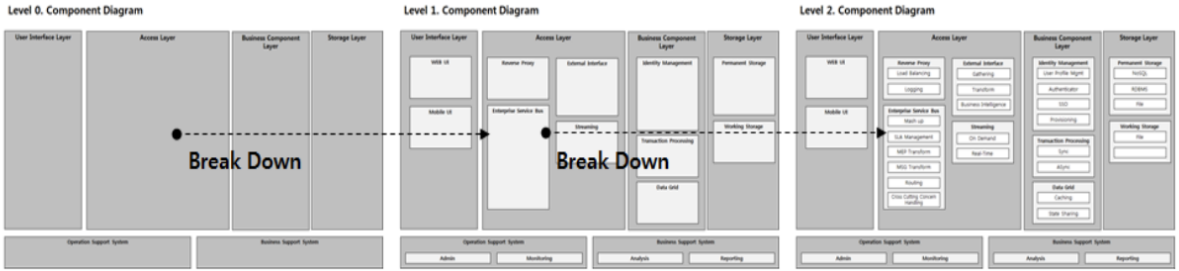
- 애플리케이션을 구성하는 컴포넌트들을 정의
- 계층(Layer)별로 정의 하여 상세화 (보통 3~4단계가 적절)
- 간단해 보이지만 매우 중요함. (향후 시스템의 구조, 팀 구조에 영향)

• Component Level 0



• Component Level 0 Description

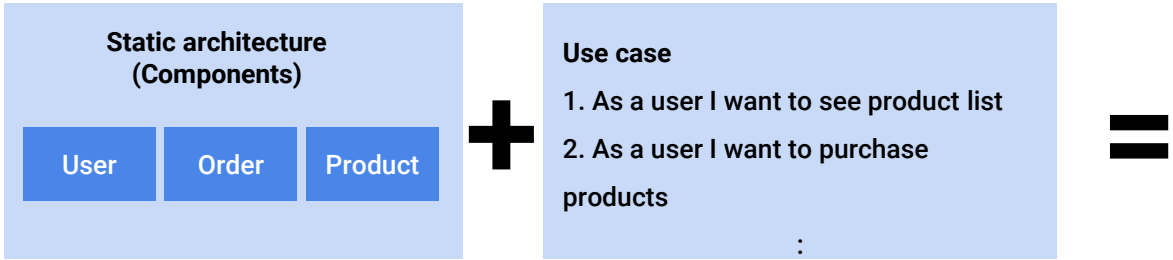
Level 0	Description
User Interface	웹,모바일등 사용자 인터페이스 계층
Access Layer	비지니스 로직을 OPEN API를 이용하여 외부로 서비스 하는 계층 외부 시스템과의 인터페이스를 제공
Business Component Layer	비지니스 로직을 구현하는 계층
Storage Layer	데이타를 저장하는 계층
OSS	운영,모니터링 관리를 위한 계층
BSS	비지니스를 위한 지표 분석 리포팅 서비스

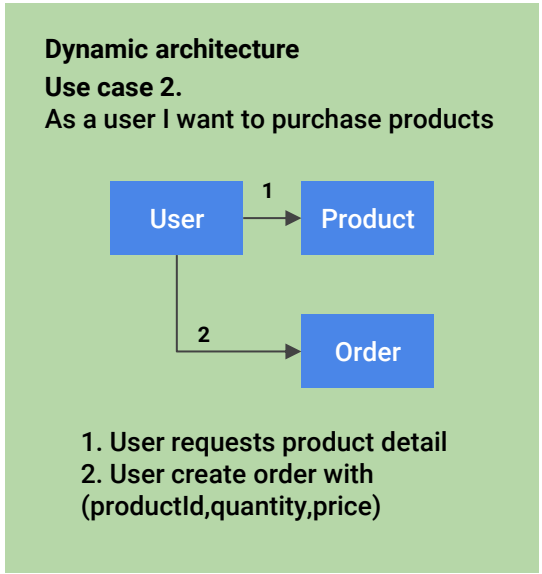


애플리케이션 아키텍처

동적 아키텍처 (Dynamic Architecture)

- 컴포넌트 정의후, 메이저 시나리오별로 컴포넌트간의 상호 작용을 정의
- 컴포넌트간 상호작용은 숫자를 넣어서 순서를 표시하고, 구체적인 동작과 필요한 데이터를 표시한다. (향후 각 화살표가 API로 맵핑됨)





애플리케이션 아키텍처

상세 아키텍처

- 구현에 대해서 상세한 설명(작업지시)가 필요한 경우 별도로 작성

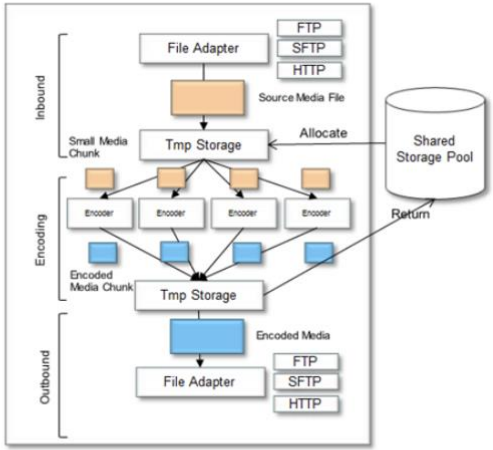
Transcoding Component Architecture

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.



< 분산 트랜스코딩 아키텍처 예시>

테크니컬 아키텍처

하드웨어 아키텍처

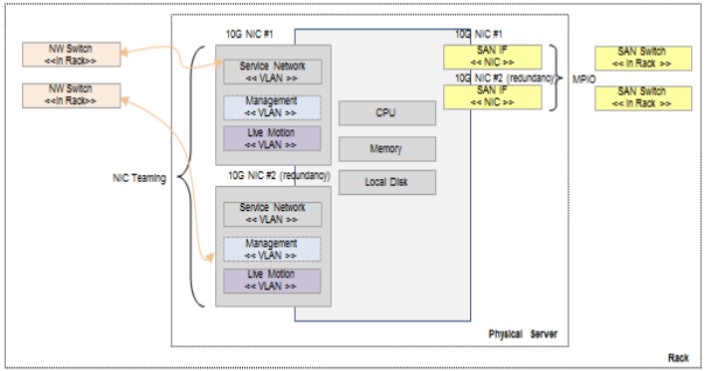
- 서버 아키텍처
- 네트워크 구성
- 스토리지 구성
- 랙 디자인
- 글로벌 디플로이 (데이터센터) 구조

테크니컬 아키텍처

하드웨어 아키텍처 / 서버 아키텍처

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

❑ **Physical Server Architecture**



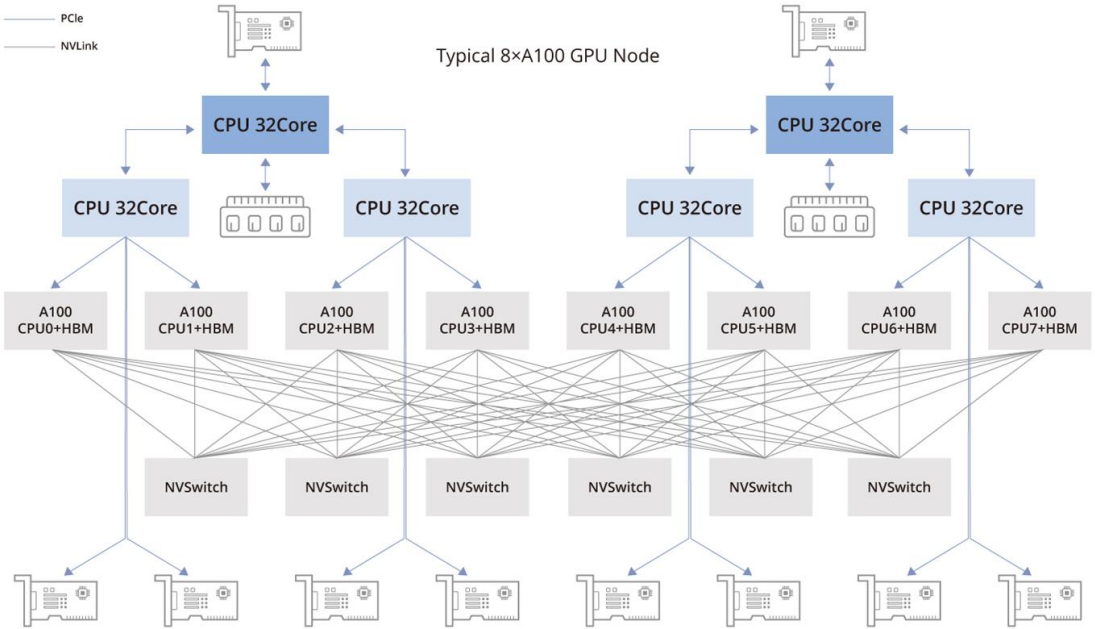
Physical Server component

- 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 아키텍처

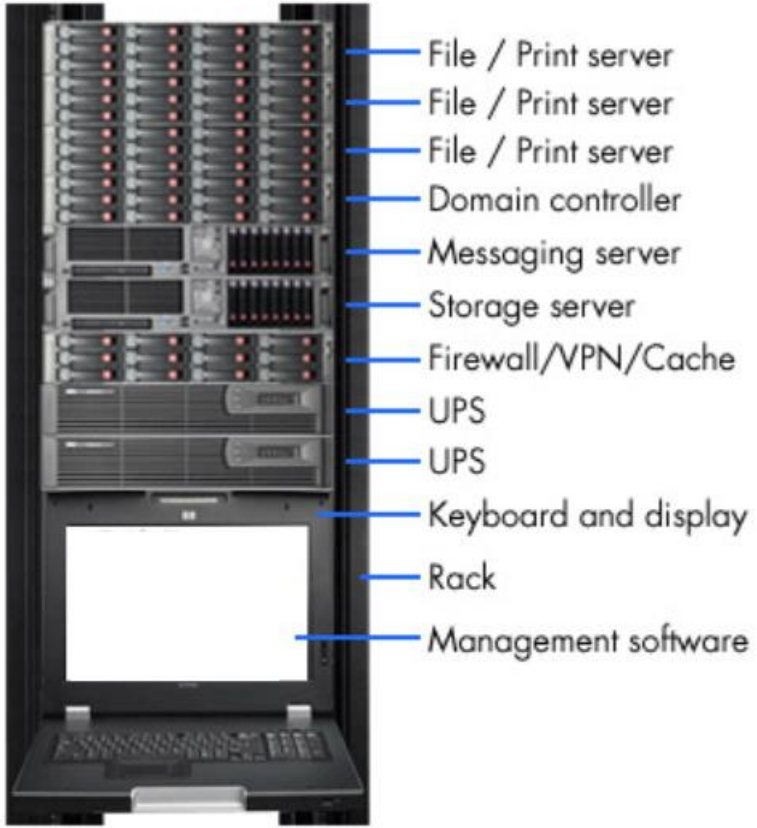
머신러닝 학습 서버 아키텍처



테크니컬 아키텍처

하드웨어 아키텍처 / 랙 디자인

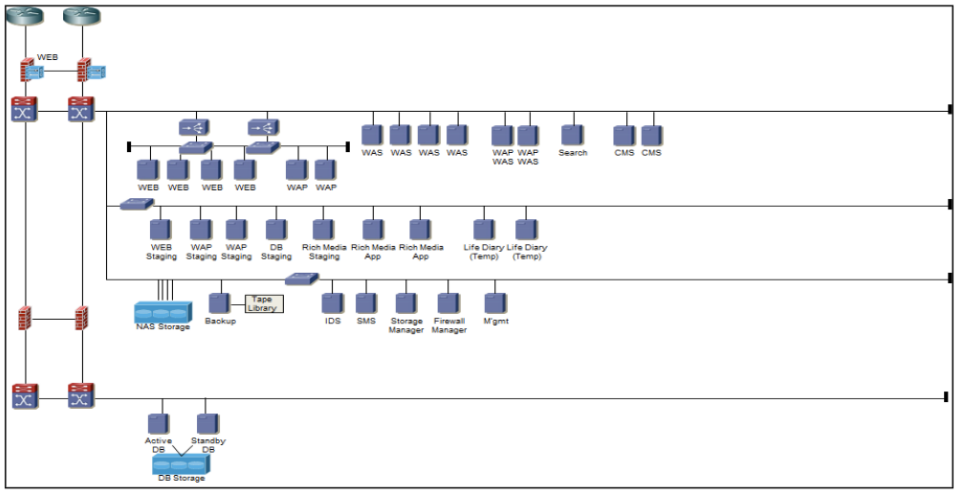
- 랙에 서버 배치 구조
- 케이블링 (스위치, 랙간 케이블링)
- KVM,UPS 부가 장비 배치
- 사용 전략 및 발열량 고려



테크니컬 아키텍처

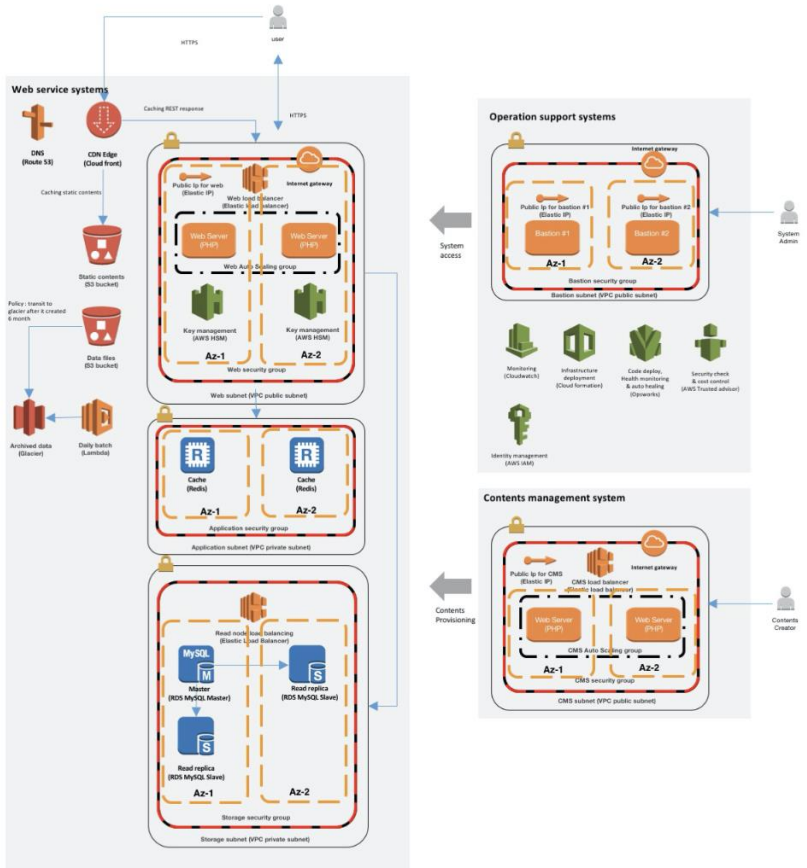
하드웨어 아키텍처 / 네트워크 아키텍처

- 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

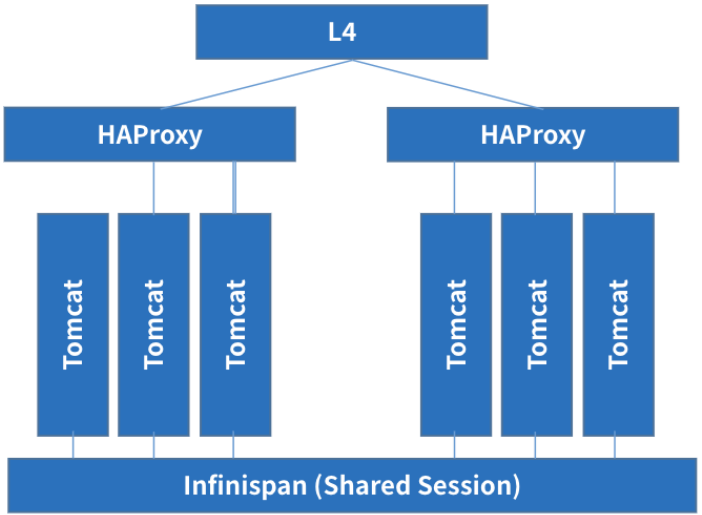
아마존 클라우드 기반 배포 모델 설계 예



솔루션 아키텍처

솔루션 아키텍처

- 데이터 베이스, 미들웨어, 웹서버등의 구성 (클러스터링)
- 솔루션들간의 설정이나 연동 방식을 서술



데이터 아키텍처

시스템에 저장되는 데이터 아키텍처

- 데이터 구조 (ERD)
- 저장 장소 및 솔루션
- 보안 처리 아키텍처 (키 관리, 전송 암호화, 저장 암호화)
- 데이터 생명 주기 관리 (생성, 백업, 폐기)
- 개인 정보 관리 정책
- 데이터 리니지 정의 (시스템간 데이터 흐름)
- 데이터 시스템간 연동 (ETL,CDC)

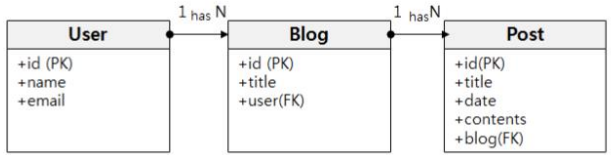
데이터 아키텍처

데이터 구조

- ERD 등을 이용하여 정보 구조 정의

- Conceptual Modeling

- Logical Modeling



- Implementation Modeling

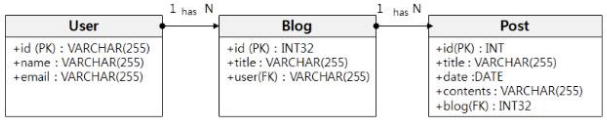


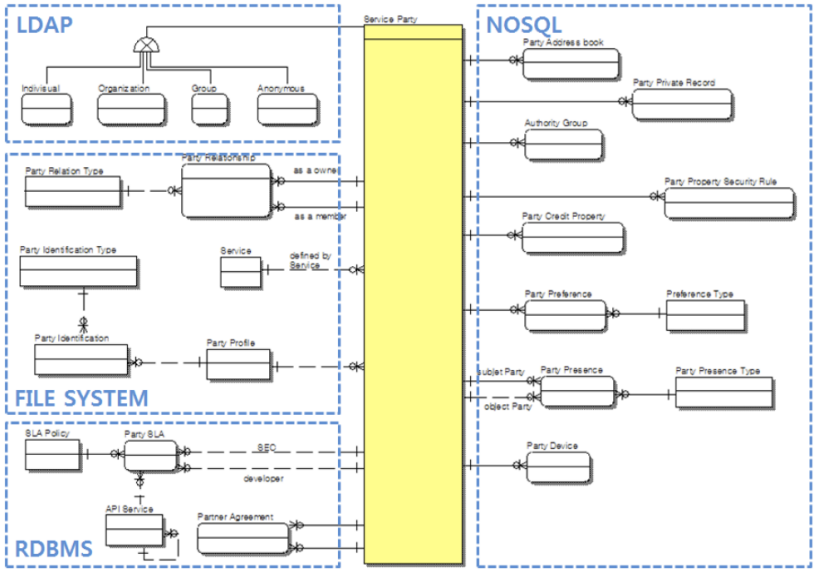
TABLE : USER

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

데이터 아키텍처

데이터 아키텍처 / 배포 구조

- 데이터 저장소
  정의된 데이터 모델에 대한 솔루션 별 데이터 저장소 아키텍처 정의

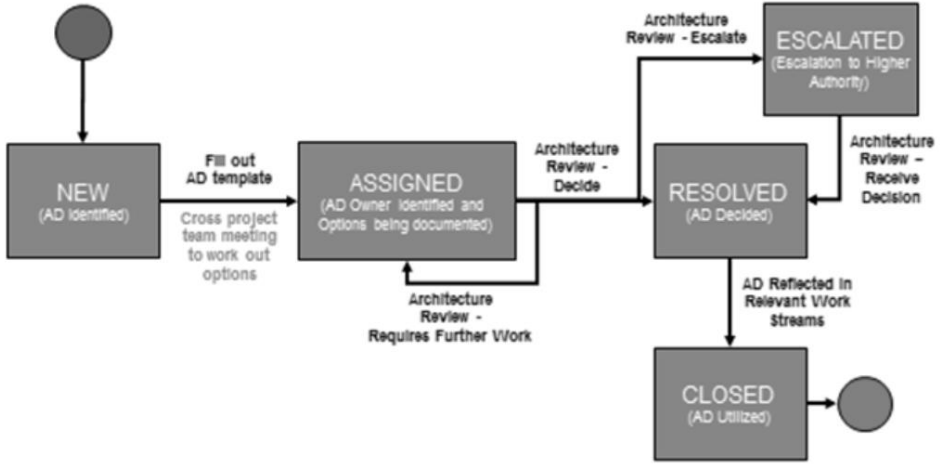


아키텍처 의사 결정

아키텍처 의사 결정 프로세스

Architecture Decision (aka AD)

- 아키텍처적인 의사 결정이 필요한 경우
- 비용, 기술 선택, 조직의 역량, 회사 전략
- 최고 의사 결정 조직이 있어야함 (CTO, 치프 아키텍트)



아키텍처 의사 결정 템플릿

Option 1.

Describe summary of architecture option with simple diagram

Option 2.

Description and motivation

Describe use case of the architecture and summary of background

Assumption

Describe assumption. (It is mainly restriction to design architecture, Such as capacity, maturity of Development team, schedule etc)

Consideration

Consideration point to design architecture like architecture principles

Option 1.

- **Pros :** *Describe pros of option 1.*
- **Cons :** *Describe cons of option 1.*

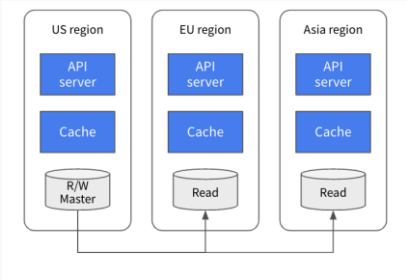
Option 2.

- **Pros :** *Describe pros of option 2.*
- **Cons :** *Describe cons of option 2.*

아키텍처 의사 결정 템플릿 (예시)

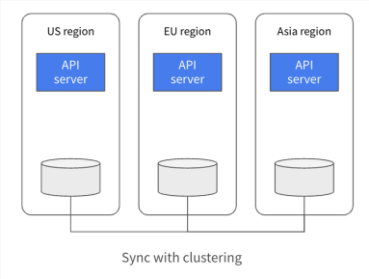
글로벌 배포 서비스에서 리전간 데이터 동기화 아키텍처

Option 1.



- US region DB를 Master DB로 사용하고 모든 write를 US에 수행
- EU,Asia는 US DB를 Replication 하고, Read로만 사용.
- 성능은 캐쉬를 이용하여 보상

Option 2.



- Region 간 Replication이 가능한 NoSQL DB를 사용
- 각 Region별 Multi master

Description and motivation

3개의 리전을 커버하는 글로벌 SNS 서비스에서, 리전간에 데이터 베이스 동기화를 어떻게 할것인가?

Assumption

1억 사용자를 기준으로 하고, 리전별 데이터베이스의 크기는 1PB로 가정한다.

Consideration

솔루션 도입 비용, 유지 보수, 데이터의 일관성 유지

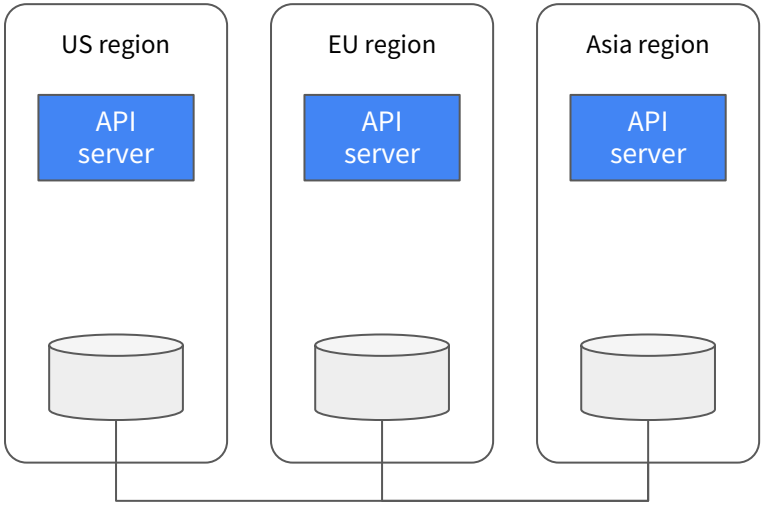
Option 1. RDBMS의 Read replica

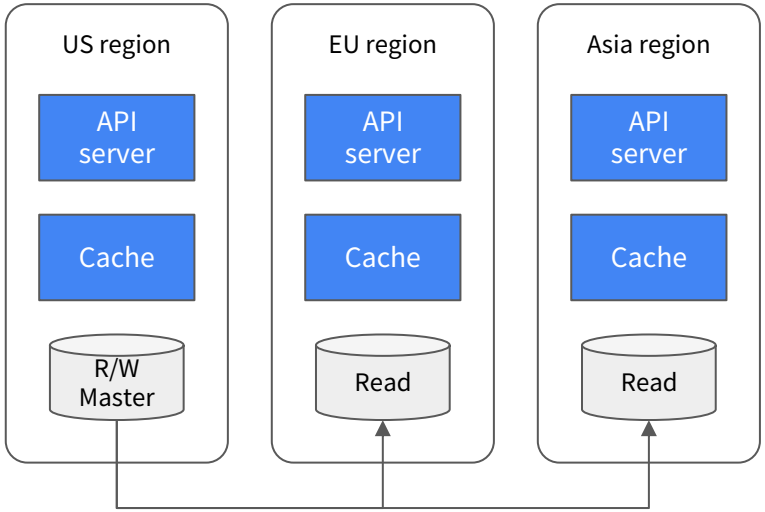
- Pros : *낮은 비용*
- Cons : *추가적인 캐쉬 레이어 사용이 필요하고, 데이터 동기화시 2~3 sec의 딜레이가 필요함.*

Option 2. Global replication이 가능한 NoSQL 사용

- Pros : *구현이 쉽고, 데이터 일관성에 문제가 없음*
- Cons : *Option1 대비, 년 5M의 추가 비용이 소요됨(라이센스 비용)*

감사합니다.





Sync with clustering