

# LLM 마스터클래스 #5

LLM 서비스 개발: 아키텍처와 도입



## Part 5-1.

# RAG 복습과 파인 튜닝



# RAG와 Fine tuning 의 차이점

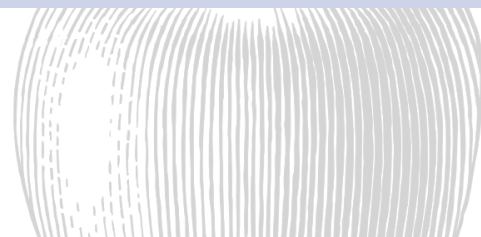
**빡세게 신입 교육 시키기**

**vs**

**오픈북 테스트**

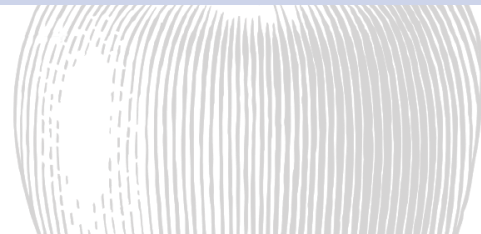
교육 시간이 오래 걸림  
돈이 많이 듦!  
틀릴 수 있음  
교과서가 필요함  
교과서 질에 따라 결과가 다름  
직원이 사표내면 다시 교육해야 함  
"너 누구한테 그런 거 배웠어???"

제대로 된 책 가져 왔습니까?  
책 가져오는 방법을 계획해야 함  
시험장에 책 가지고 가야 함  
책 없으면 망함  
새로운 책이 나오면 그것도 들고 가면 됨  
정보를 믿을 수 있음  
어디서 정보를 가져왔는지 증명 가능  
오래된 책이면? 선전용 빠라면?  
애드립 어려움



# RAG와 Fine tuning 의 차이점

오픈북 테스트	vs	빡세게 신입 교육 시키기
<ul style="list-style-type: none"> <li>-제대로 된 책 가져 왔습니까?</li> <li>-책 가져오는 방법을 계획해야 함</li> <li>-시험장에 책 가지고 가야 함</li> <li>-책 없으면 망함</li> <li>+새로운 책이 나오면 그것도 들고 가면 됨</li> <li>+정보를 믿을 수 있음</li> <li>+어디서 정보를 가져왔는지 증명 가능</li> <li>-오래된 책이면? 선전용 빠라면?</li> <li>-애드립 어려움</li> </ul>		<ul style="list-style-type: none"> <li>-교육 시간이 오래 걸림</li> <li>-돈이 많이 듦!</li> <li>-틀릴 수 있음</li> <li>-교과서가 필요함</li> <li>-교과서 질에 따라 결과가 다름</li> <li>-직원이 사표내면 다시 교육해야 함</li> <li>-“너 누구한테 그런 거 배웠어???”</li> <li>+실시간이 아니라서 빠르고 에러날 확률이 적음</li> <li>+ 한 번 교육하면 파이프라인이 간단함</li> </ul>



## RAG와 Fine tuning 의 차이점

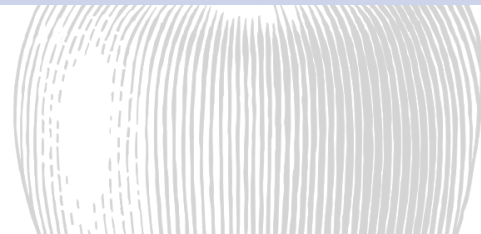
### 텔레프롬프터

VS

### 외워서 하기

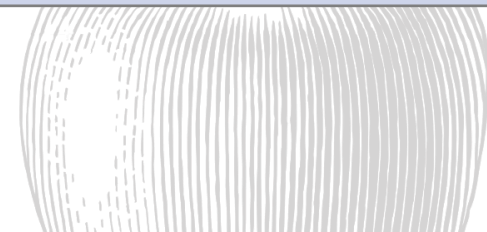
- 원고 생성 시스템 필요
- 생성 시스템이 복잡해짐
- 같은 분야일때 딱히 더 유리하지 않음
- +여러가지 분야 뉴스 가능
- + 여러가지 모델 (아나운서) 바꿔 사용가능
- +여러가지 데이터 소스 사용 가능

- 오래 걸림
- 외운 것만 할 수 있음
- 날짜, 숫자같은 아주 자세한 디테일은 힘들
- 여러 모델 테스트가 힘들
- +전문성 필요에 좋음
- + 일관적인 말투에 좋음
- +비슷비슷한 처리 내용에 좋음
- +훈련된 모델 하나만 있으면 됨



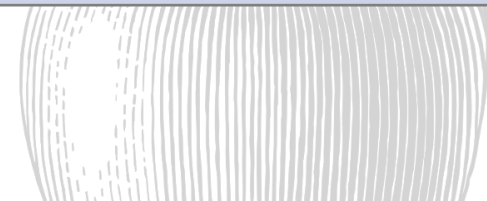
## RAG와 Fine tuning 의 차이점 - 장점

	검색 보강 생성 (RAG)	파인 튜닝
정확성	신뢰할 수 있는 소스에서 관련 정보를 검색하여 결과의 정확성을 높임.	모델을 특정 작업이나 도메인에 맞게 최적화하여 효율성 증가.
의존성	데이터에 기반하여 응답을 생성하므로 신뢰도가 높음.	광범위한 데이터를 사용하여 모델을 재학습시킬 수 있음.
업데이트 가능성	데이터베이스나 데이터셋을 새로운 정보로 업데이트 가능.	새로운 데이터로 지속적으로 모델을 업데이트하여 발전시킬 수 있음.



# RAG와 Fine tuning 의 차이점 - 단점

	검색 보강 생성 (RAG)	파인 튜닝
연산 비용	데이터를 검색하는 추가 단계가 필요하여 연산 비용과 시간이 증가함 (저녁 뉴스)	파인 튜닝 과정이 자원을 많이 소모하며 시간이 많이 걸림 (연극 공연)
데이터 품질 의존성	데이터의 품질과 범위에 매우 의존적임. 부족한 데이터는 효과를 제한할 수 있음.	학습 데이터의 품질과 양에 따라 모델의 성능이 크게 좌우됨 (기본 모델 저하)
확장성	데이터베이스나 데이터셋의 확장이 어렵고 자원 집약적일 수 있음.	다시 트레이닝 해야 함. 모델을 바꿔야 함.
소스 데이터에 대한 과도한 의존	사용 가능한 데이터 외에 새로운 응답을 생성하는 능력이 제한될 수 있음.	
정보의 오류 가능성	검색 오류나 구식 정보로 인해 잘못된 응답이 발생할 수 있음.	트레이닝대로 답을 생성하지 않을 수도 있음



# RAG와 Fine tuning 의 차이점

## 추가 고려 사항

개인정보 보호	RAG는 데이터의 개인정보를 유지하는 반면, 파인튜닝은 벤더에 의해 수행되는 경우 개인정보 문제를 야기할 수 있음.
실시간 성능	파인튜닝은 더 빠른 응답 시간을 제공할 수 있음.

## 결론

- RAG와 파인튜닝은 상호 보완적인 기법
- 최신 응답, 최소한의 허위 생성, 적응력을 위해 RAG를 선택
- 충분한 훈련 데이터가 있는 특정 도메인에서 우수한 정확성을 위해 파인튜닝
- 같이 사용하여 하이브리드 모델을 생성 - 복약지도





# 이럴 때 Fine tuning !

새로운 장르, 혹은 분야	의료, 과학, 보험, 자동차, 리테일 등등 그 분야 내 고유의 언어 혹은 문제 풀이 방식이 있을 때
특정한 일을 수행할 때 퍼포먼스 올리기	번역 혹은 특정한 콘텐츠를 생성할 때
아웃풋 방식이 특수할 때	코드 생성, 텍스트의 분위기 설정, 디테일 레벨 설정 등등
새로운 데이터에 적응할 때	

- 예:
- 고객 지원
  - 번역
  - 콘텐츠 생성
  - 특정 분야 내의 질답 (FAQ)
  - 텍스트 감정 분석



## 이럴 때 No fine tuning !

데이터셋이 아주 작음

원 모델의 훈련 데이터와  
내가 쓰고 싶은 시나리오가 아주 다름

파인 튜닝으로 극복 안 됨

모델을 자주 업데이트 혹은 수정 해야  
함.

좀 더 단순한 방법이 있음

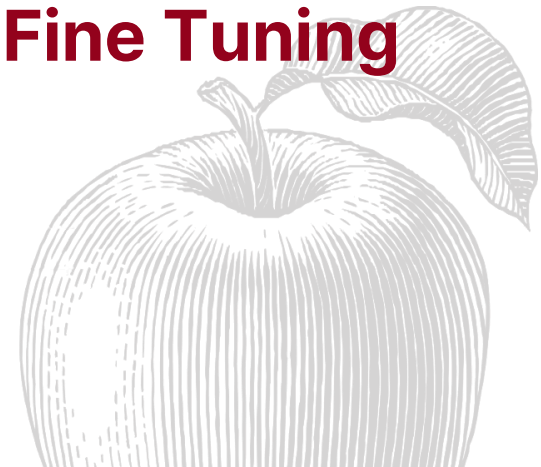


## 마지막 복습 - 더 간단하게 구분하는 방법

- 빠릿빠릿한 신입이 데이터 잘 뽑아오면 됨.
- 지시사항이 조금씩 다름
- 정보 종류가 한군데에 잘 정리되어 있지 않고 중구난방임
- 정보가 자주 바뀜
- 일 시킬 때 똑같은 말을 엄청 반복해야 함
- 일 하려면 기본적으로 알아야 하는 전문용어가 아주 많음
- 생성할 때 공식이나 룰이 많음
- 아웃풋 포맷도 따라야 하는 규칙이 아주 많음
- 어느 정도 노하우가 필요한 일임
- 그런 노하우나 전문용어가 많긴 하지만 자주 바뀌진 않음  
(예: 약품 성분, 병 이름)

▶ **RAG**

▶ **Fine Tuning**



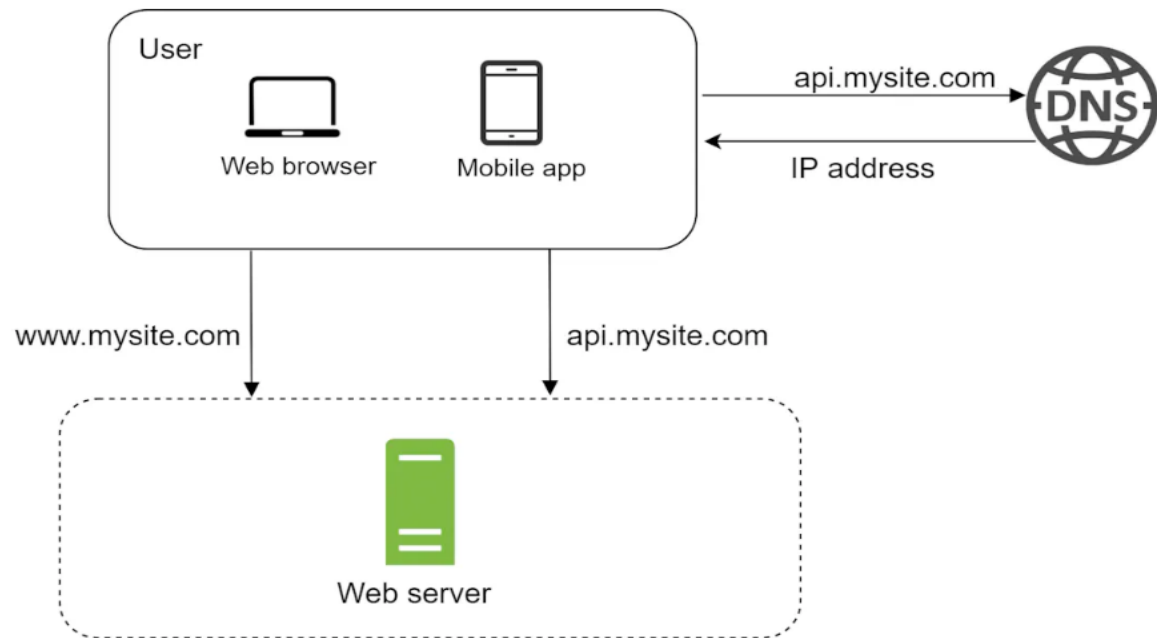
## Part 5-2.

# 일반적인 distributed IT 시스템의 패턴 이해

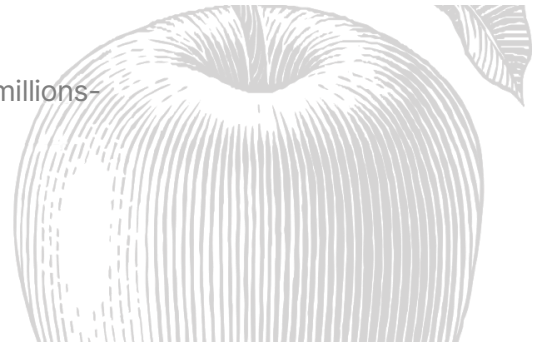


# Scaling system

- 가장 간단한 모델
- 홈페이지 주세요!
- 서버 1개
- 카탈로그 포함
- 데이터 베이스 통합

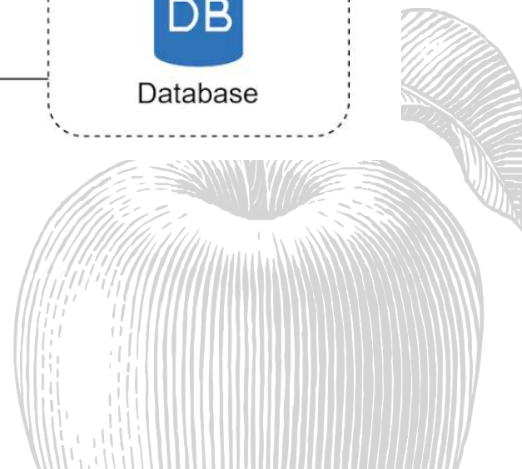
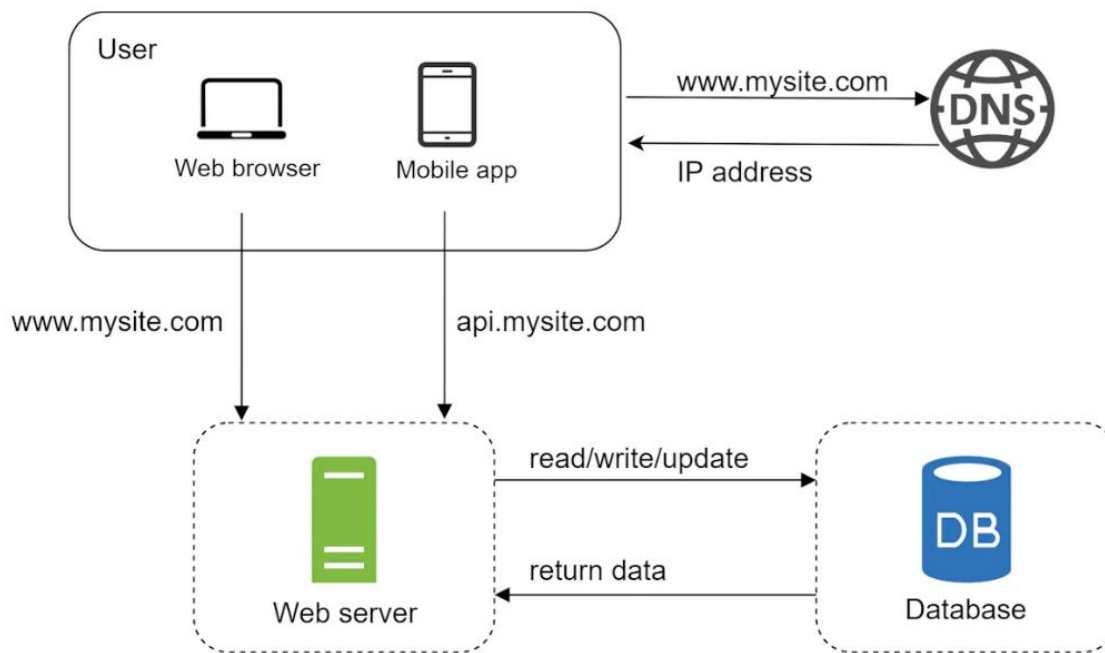


<https://bytebytego.com/courses/system-design-interview/scale-from-zero-to-millions-of-users>

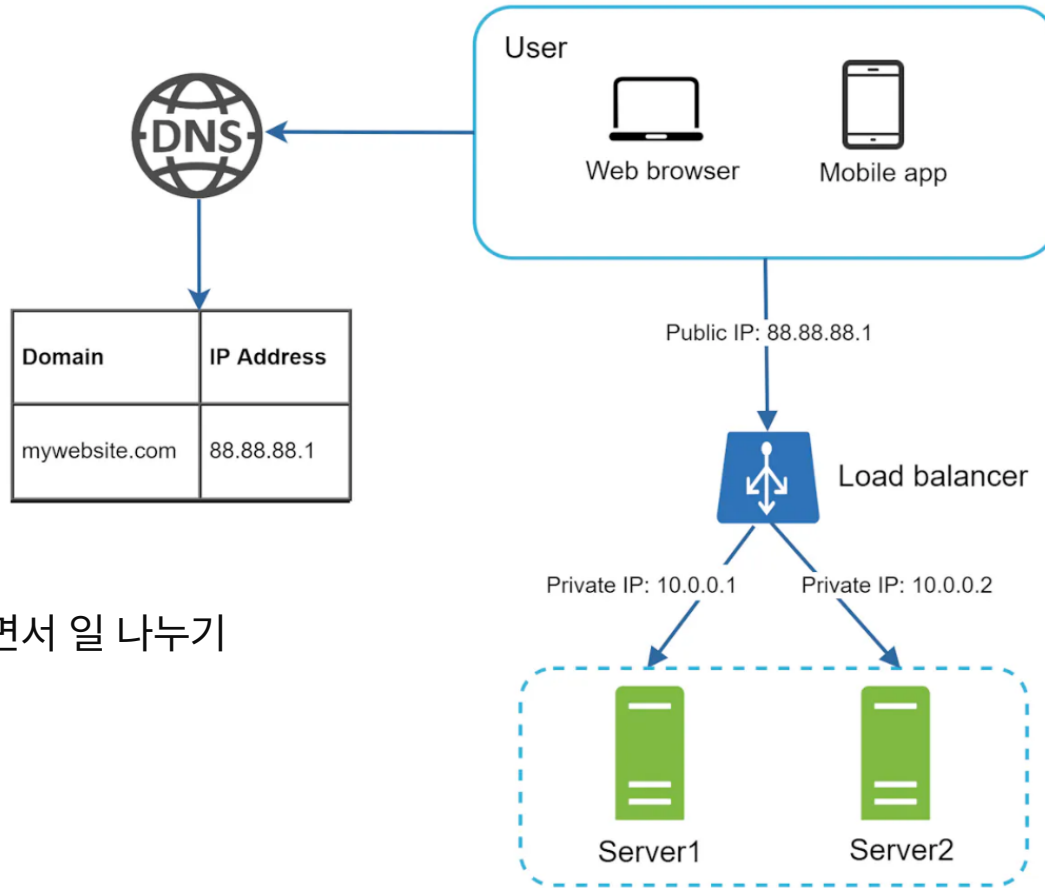


# Scaling system

- 데이터가 필요
- 서버 2+



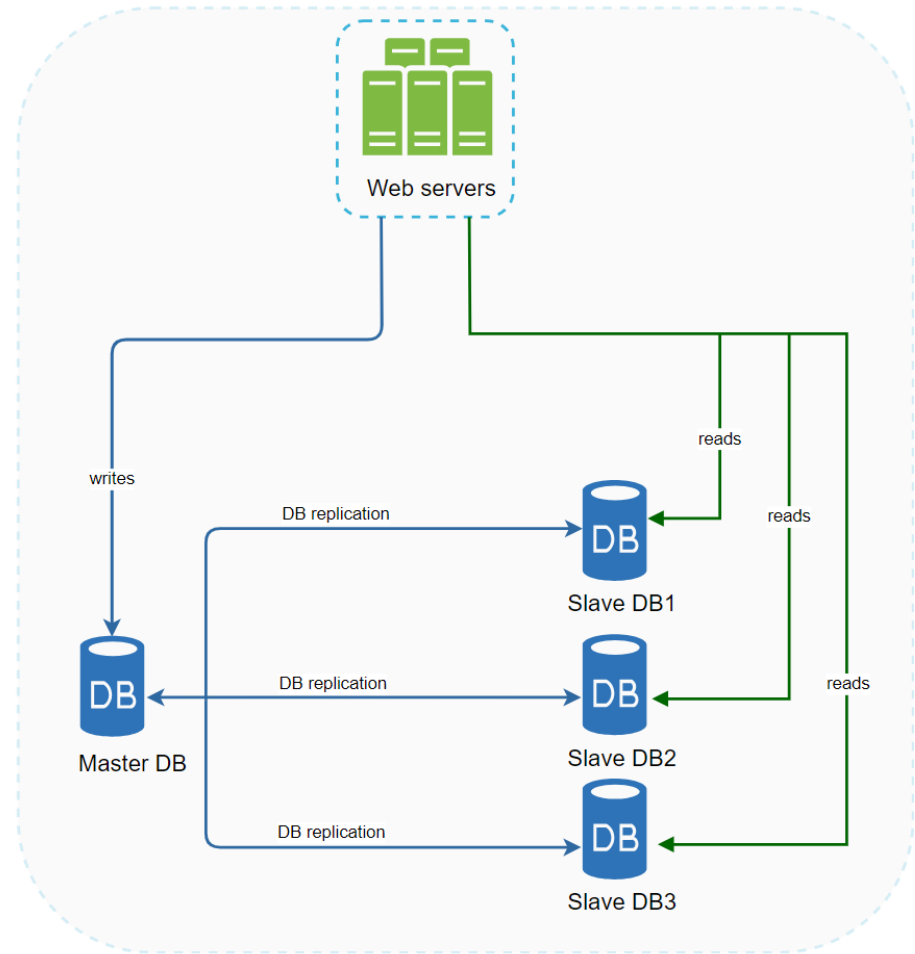
# Scaling system



- 상황 보가면서 일 나누기

# Scaling system

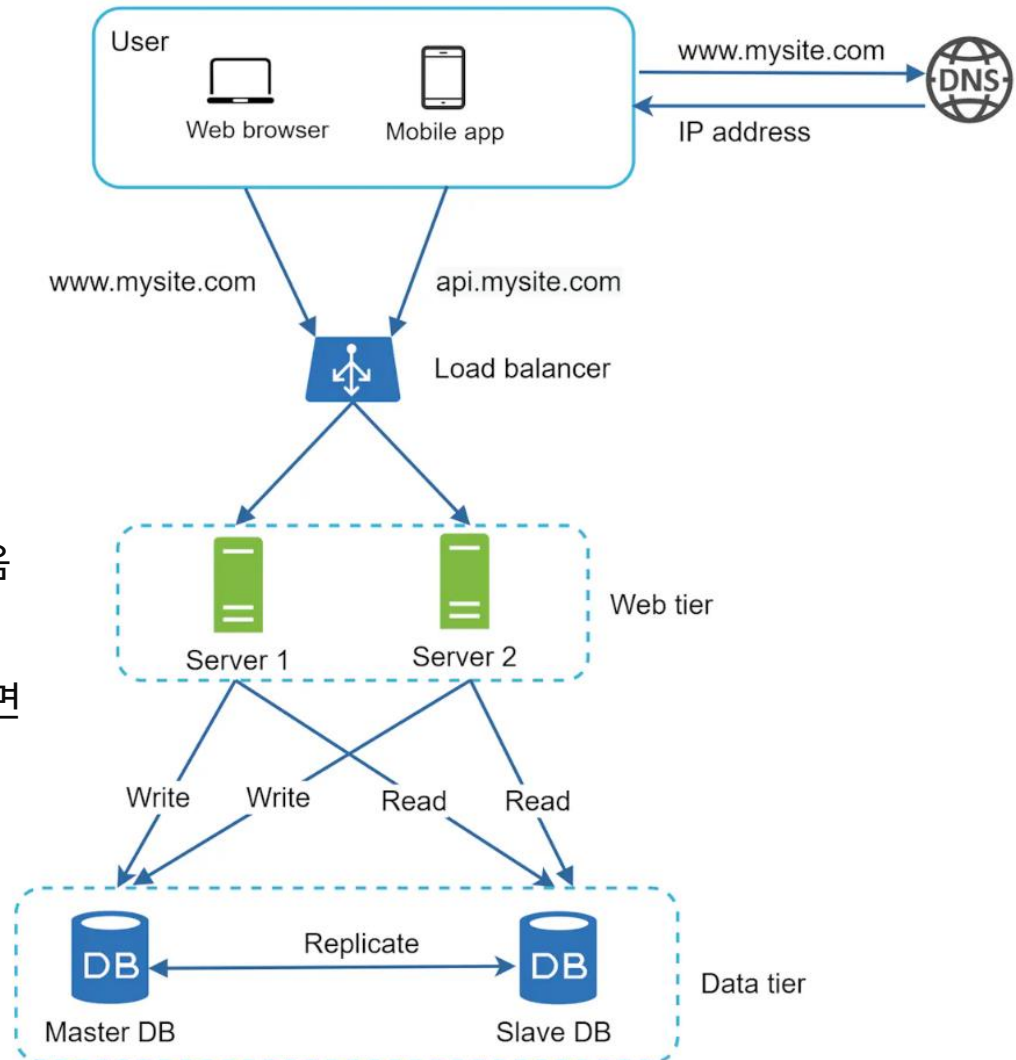
- 데이터 베이스 확장
- 카탈로그 읽기만 하는 것은 상관 없음
- 읽기 vs 쓰기
- 그런데 완판 된 걸 재고 있다고 올리면 ..?





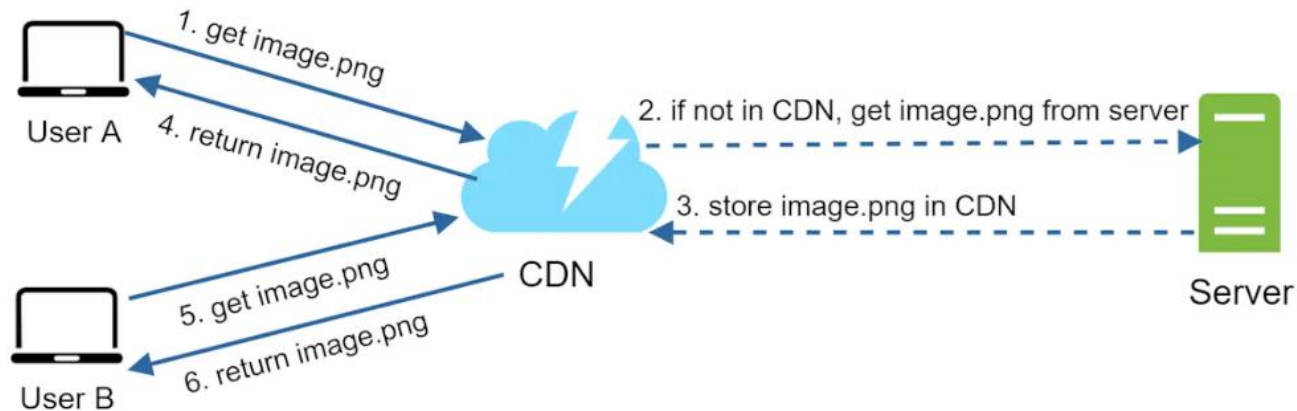
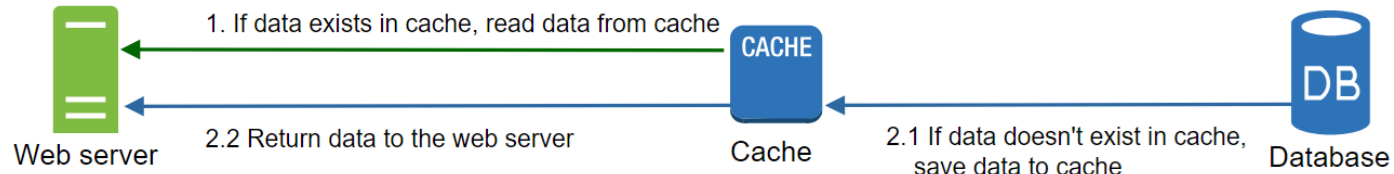
# Scaling system

- 데이터 베이스 확장
- 카탈로그 읽기만 하는 것은 상관 없음
- 읽기 vs 쓰기
- 그런데 완판 된 걸 재고 있다고 올리면 ..?



# Scaling system

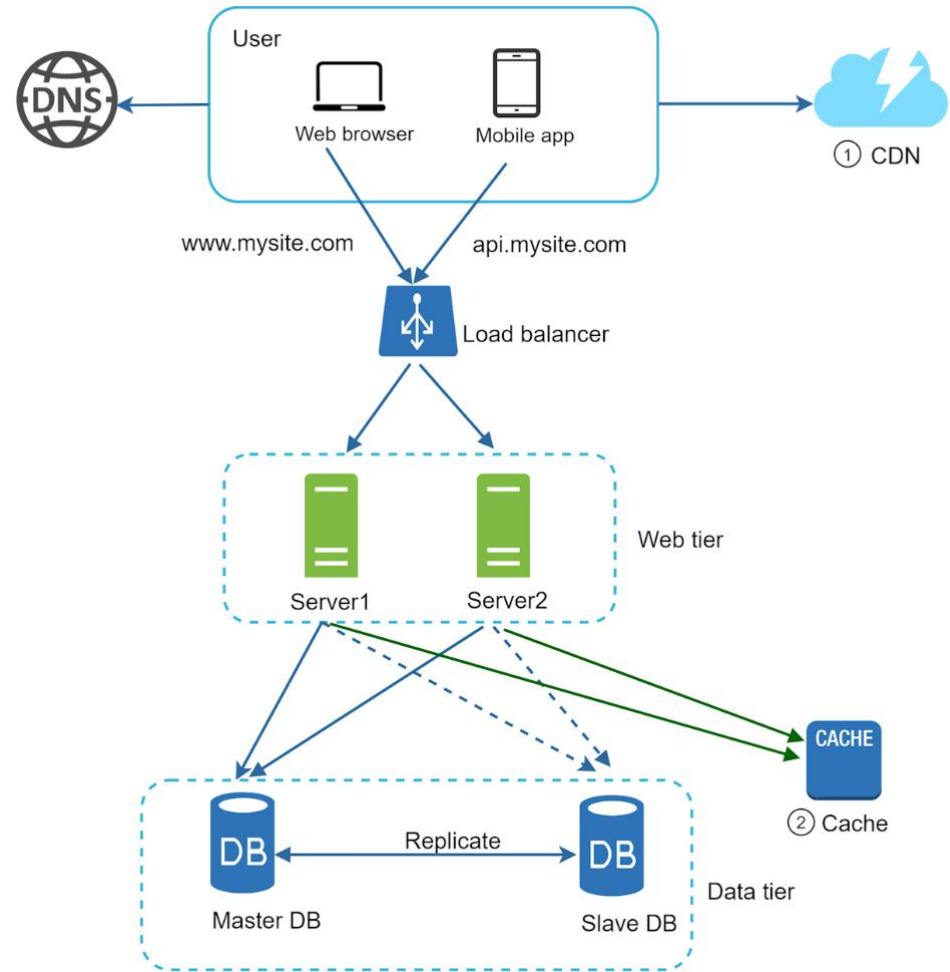
## cache, CDN



- FAQ 저장
- 매일 쓰는 것 저장

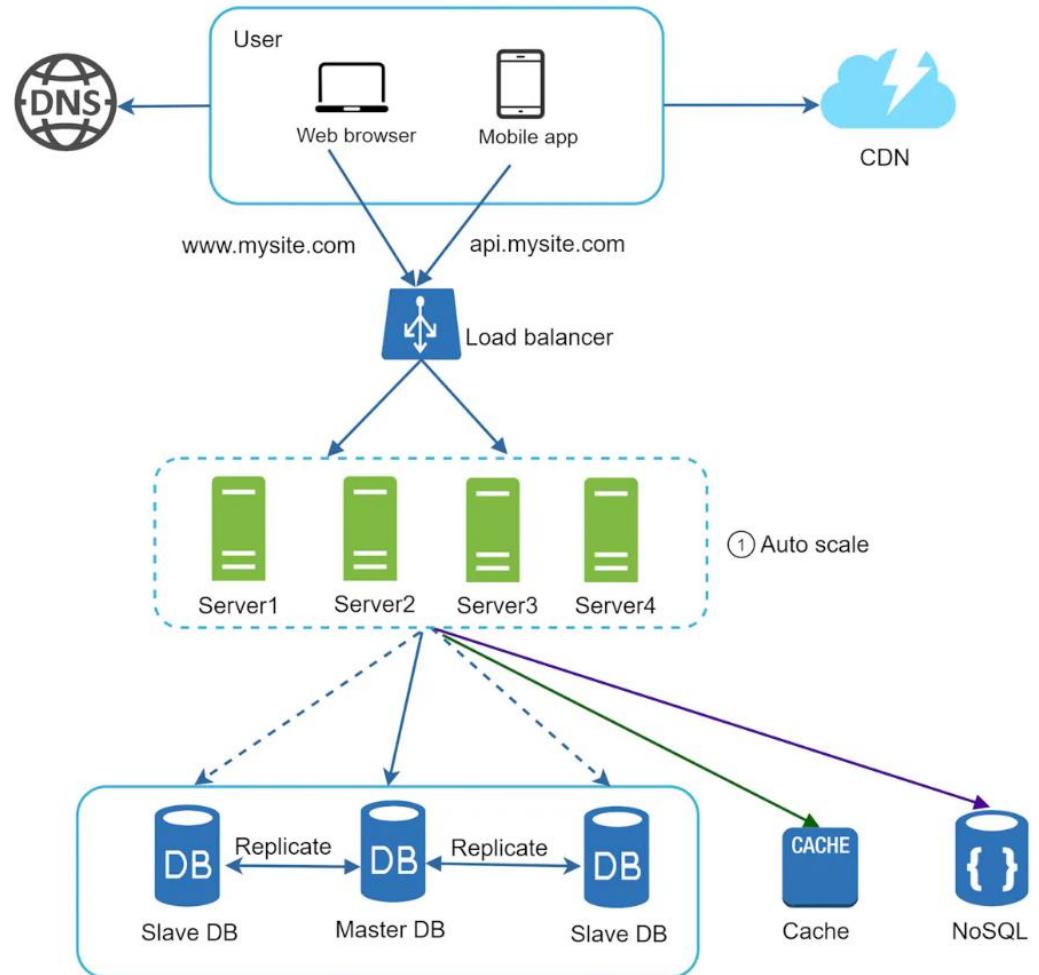
# Scaling system

- CDN 통합



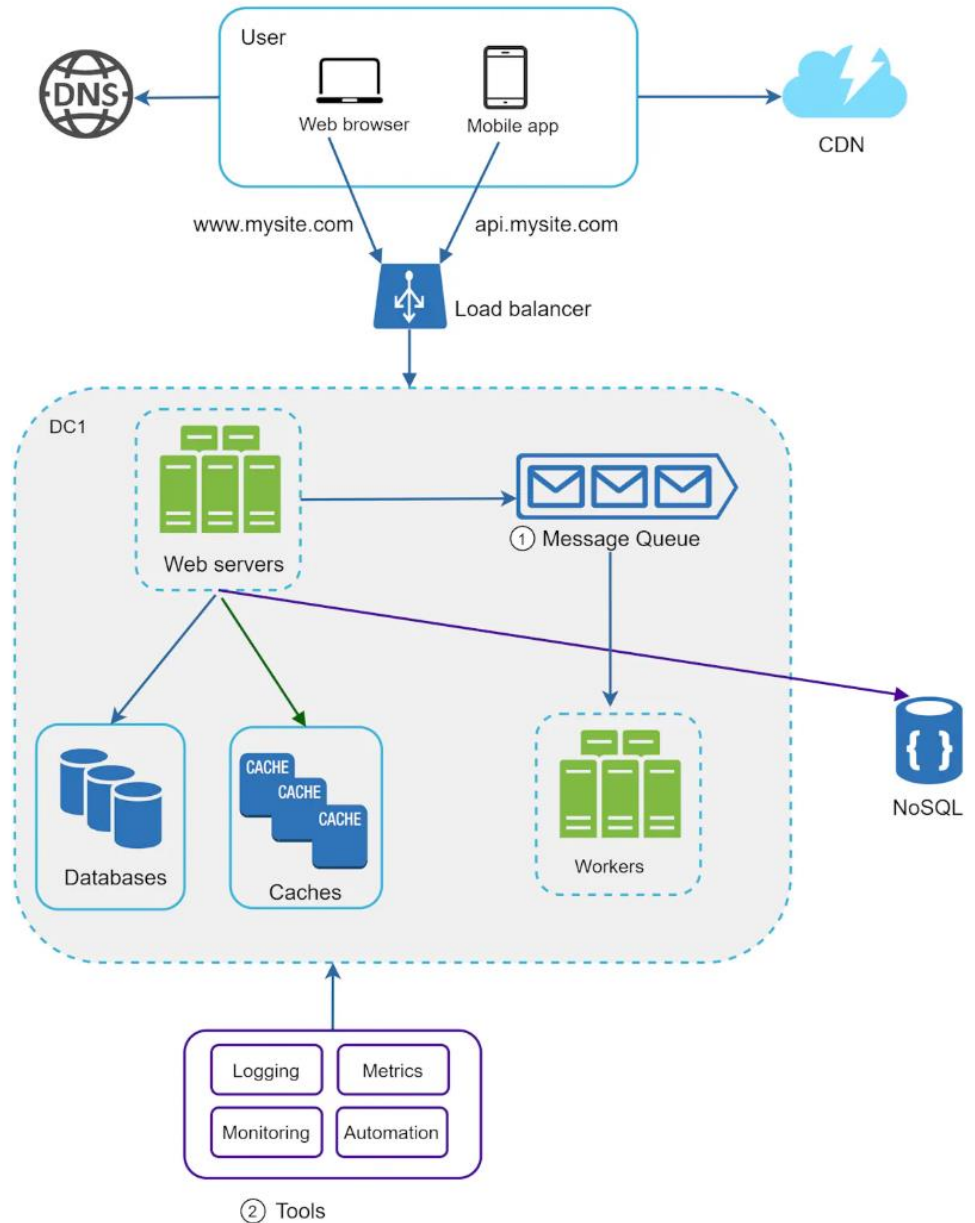
# Scaling system

- Stateful / stateless
- Stateless architecture



# Scaling system

- Logging, metrics, monitoring
- Message queue



# LLM 서비스 - SkyNet

- 왜 One Evil AI 서비스가 아닌가
  - 하나의 수퍼 서비스가 아님
  - 예: API 콜과 텍스트 리턴
- 너 내가 누군지 몰라??
  - 종합병원 예
  - 내 데이터를 가지고 훈련시키지 않아??
- 훈련 시키기는 합니다만...
  - A lot of testing - what is actually useful?
  - Personal data often not useful
- 이야기했던 내용을 기억을 하긴 합니다만...;
  - 콜센터를 기억하자..



# Colab 노트북을 서비스로 못 만드는 이유

Colab 노트북은 매번 다시 시작해야 하죠...

알아서 답을 하지 못하죠..

사람이 shift+enter 눌러야 하죠..

1:1로 집밥 해먹이기 vs 편의점에 도시락으로 뿌리기

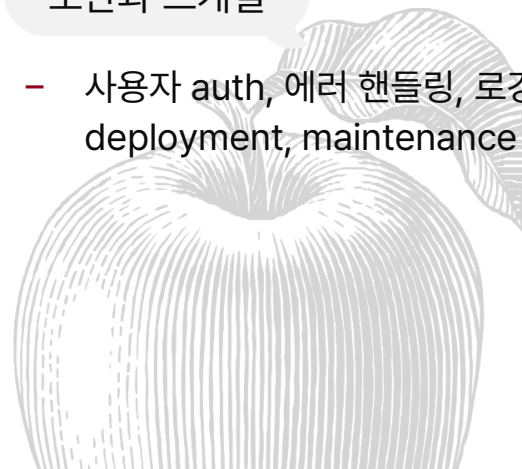
내 컴퓨터에서 홈페이지 돌리기 vs 전세계에서 쓸 수 있는 웹사이트 만들기

환경을 셋업하고 설치하고 모니터 해야 하  
죠

- 창구 (API endpoint) 디자인과 설계
- 웹 프레임워크 / 모바일 UI

보안과 스케일

- 사용자 auth, 에러 핸들링, 로깅, deployment, maintenance



## Part 5-3.

# 기존 시스템에 LLM을 더 하는 패턴





## 타입 - 쉬운 도입(1~2개월)

- 간단한 모바일 앱
- 간단한 웹사이트 - 현존 시스템에 통합되지 않음
- 사용자가 별로 없음
- Async 가 괜찮음 (실시간이 아님)
- 전문 용어가 그리 많지 않거나,  
아주 좋은 데이터가 준비되어 있음
- 데이터 소스가 간단함 (특히 텍스트 파일!)
- 한 번 결과를 만들어 놓으면 똑같은 것을  
여러사람들이 다운로드 해도 됨  
예: 어려운 복약지도를 쉬운 말로 풀어둔 것, 이미지 다운로드
- 좀 틀려도 됨
- Single-turn
- 세션 저장 안 해도 됨
- Authentication 필요 없음
- Scaling 필요 없음
- 공짜 서비스 아님



## 타입 - 덜 쉬운 도입(3~6개월)

- 틀리면 안 됨
- 동시 접속 사용자가 10명 이상이고 async 아님
- 기존 앱이나 사이트에 연계되는 앱/사이트임
- 사용하고 있는 기존 데이터베이스가 있고, 그 데이터베이스와 통합해야 함
- 데이터 사용 규칙이 있음
- Multi-turn 지원하고 싶음
- 사용자 세션 지원하고 싶음
- Auth 가 있음
- Permission 레벨이 다양함
- CI 와 그 외 파이프라인, 빌드 시스템이 있고 거기에 통합하고 싶음
- 모델 관리를 하고 싶고 퀄리티도 모니터 하고 싶음
- 신경 써야 하는 정부 규제 등이 있음



## 타입 - 어려운 도입(1년+)

- 틀리면 안 됨
- 동시 접속 사용자가 100명 이상이고 async 아님
- 데이터 소스가 structured/unstructured 여러가지로 다섯개 이상
- 대기업이고 고객들이 쓸 프로덕트임
- 국제적인 시스템임
- 파인 튜닝 필요함
- Internal audit 필요함
- CI 와 그 외 파이프라인, 빌드 시스템이 있고 거기에 통합하고 싶음
- 모델 관리를 하고 싶고 퀄리티도 모니터 하고 싶음
- 현존하는 복잡한 시스템에 통합될 것임 (예: 아래아 한글에 LLM 도입)
- DSL 이나 그 외 custom codegen 이 필요함
- 아웃풋 퀄리티에 따라 손해배상이 필요할 수도 있음
- 보안 문제 고려해야 함



## 쉬운 도전

- OpenAI API 를 바로 콜
- Langchain 바로 쓰기
- 돈 걱정 안해도 됨
- Greenfield project, POC
- Feature parity 필요 없음



## 미뚱 레벨 도전

- RAG 시스템이나 파인 튜닝
- 적절한 LLM 과 sLLM 혹은 on-prem 시스템도 섞어서 씀
- Langchain 혹은 사내 시스템과 통합
- Evaluation pipeline 설정
- 테스트 만들기
- 법무부, 보안팀, UI 팀 들과 연계해서 작업



## 하드 레벨 도전

- 여러가지 모델과 파인 튜닝
- 여러가지 데이터 소스 플러그인 쓰고 테스트하여 컴포넌트 통합
- Authentication, security, auditing, monitoring 시스템과 통합
- 법무부팀, 국제 리스크 관리팀과 함께 데이터 privacy 완벽 통제
- LLM 비용과 사용자 관리 시스템 도입
- 모든 프로덕트 릴리즈 전 리뷰 프로세스 도입 - 보안, 데이터, RAI, UI 등등
- SLA, Quality agreement 미리 동의함



**Part 5-4.**

**LLM 디자인의 기본**



# LLM 디자인의 기본

- deterministic
- 그냥 프로그래밍 언어가 천 배 만 배로 싸고 빠름
- Cosine similarity 와 GPT 사용 예 - 노트북
- Scale, 그리고 reliability.  
얘는 느리고 비싸고 답이 매번 다름. 이걸 해결하지 않고 내놓을 수 없음.
- 답변의 질 확실하게 테스트
- 계속적으로 유저들 모니터하면서 업데이트  
: Data privacy, security, RAI..
- Single-turn, multi-turn  
: Save user session?
- Data retrieval 스텝별로 테스트





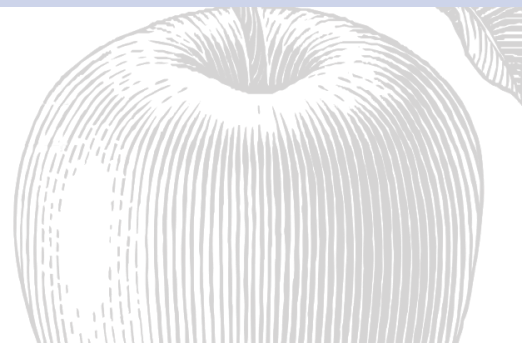
# LLM 디자인의 기본

## 1. LLM 사용 최소화

- 최대한 LLM 사용을 줄이고, 가능한 경우 결정론적 프로그래밍 언어 사용

이유

비용 절감	LLM은 계산 비용이 많이 드는 반면, 결정론적 언어는 효율적
속도 향상	LLM 추론 속도는 느릴 수 있지만, 결정론적 코드는 빠르게 실행
테스트 용이성	결정론적 코드는 테스트하기 쉽고, 디버깅도 용이
유지 보수 용이성	결정론적 코드는 이해하고 유지 관리하기 쉬움



# LLM 디자인의 기본

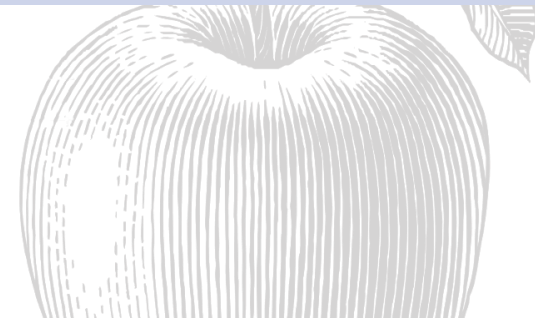
## 2. 처음부터 확장성 및 안정성 고려

- 시스템 설계 단계부터 확장성과 안정성을 고려해야 함

이유

느린 속도	확장성이 없는 시스템은 사용자 증가 시 속도 저하
높은 비용	안정적이지 않은 시스템은 유지 관리 비용 증가
예측 불가능한 결과	시스템 출력 결과가 매번 다를 경우 신뢰도 저하
서비스 제공 시 중요	상용 서비스로 제공하려면 확장성과 안정성 필수

- 안정적인 API 디자인



# LLM 디자인의 기본

## 3. 응답 품질 확실한 테스트

- 특히 민감한 분야(의료, 정부, 법률, 금융) 시스템 응답 품질 엄격하게 테스트해야 함

이유

**부정확한 정보**

잘못된 정보 제공 시 심각한 결과 초래 가능

**편향된 결과**

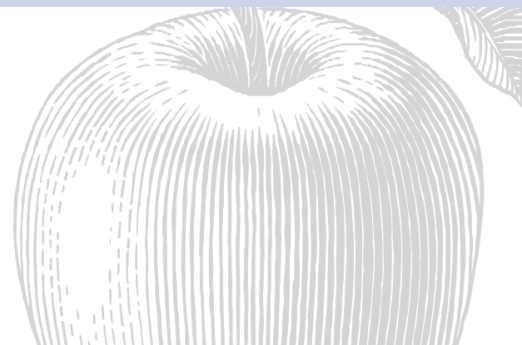
특정 그룹에 대한 편향된 결과는 차별로 이어질 수 있음

**비윤리적 결과**

비윤리적 결과는 사회적 문제 야기 가능성

**법적 책임**

잘못된 정보로 인해 법적 책임 발생 가능성



# LLM 디자인의 기본

## 4. RAI, 데이터 개인 정보 보호, 보안 고려

- 책임 있는 AI(RAI), 데이터 개인 정보 보호, 보안은 LLM 시스템 설계 필수 요소

이유

윤리적 문제

편향, 차별, 악용 등 윤리적 문제 발생 가능성

개인 정보 침해

개인 정보 유출 시 심각한 결과 초래 가능

시스템 해킹

보안 취약점 악용 시 시스템 손상 및 데이터 유출 가능성



# LLM 디자인의 기본

## 5. 운용 비용 고려

- 파인 튜닝, RAG 시스템, 그 외 모든 LLM 관련 비용이 상상이상으로 비쌈
- 수익 모델을 먼저 고려하고 도입



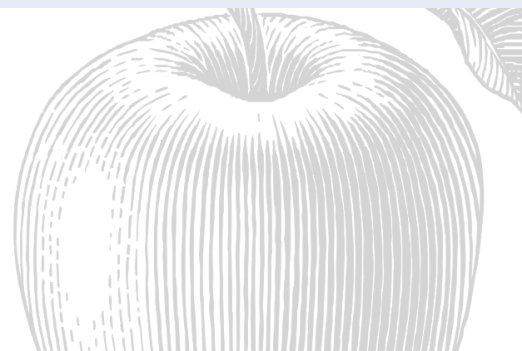
# LLM 디자인 간소화하기

## 싱글턴 vs 멀티턴

- LLM 시스템 설계 시 싱글턴 또는 멀티턴 방식 선택 고려해야 함
  - **싱글턴** : 한 번의 요청으로 응답 생성
  - **멀티턴** : 여러 번의 요청과 응답 교환을 통해 응답 생성

### 선택 기준

작업 유형	싱글턴은 간단한 작업, 멀티턴은 복잡한 대화
데이터 양	싱글턴은 적은 양의 데이터, 멀티턴은 많은 양의 데이터
사용자 경험	싱글턴은 빠른 응답, 멀티턴은 자연스러운 대화



# LLM 디자인 간소화하기

## 데이터 소스 숫자 줄이기

### 사용자 세션 서포트

- 시스템에서 대화 내용을 저장할지 여부 결정해야 함

#### LLM 저장 이점

##### 사용자 경험 개선

이전 대화 기반 개인 맞춤형 응답 제공 가능

##### 모델 학습 데이터 활용

저장된 대화를 모델 학습 데이터로 활용하여 성능 향상

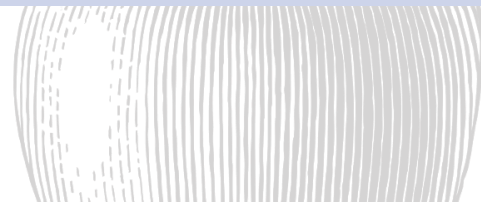
#### LLM 저장 단점

##### 개인 정보 보호 문제

대화 내용에 개인 정보 포함될 경우 문제 발생 가능성

##### 저장 공간 및 처리 비용 증가

대화 내용 저장으로 인해 저장 공간 및 처리 비용 증가



# LLM 디자인 간소화하기

Caching, async, data  
simplification...





# LLM 디자인의 기본

“뭘 할 수 있을까” 묻기 전에,  
“어떤 모델을 돌릴까” 묻기 전에,

**우리는 무슨 데이터가 있는가 돌아보기  
데이터 서빙부터 시작**



# LLM 디자인의 기본

“할 수 있으니깐 한다” 를 지양하기  
“꼭 해야 하는 부분부터”

**그러나 전체 시스템에 미칠 영향과, 앞으로 변해갈 방향을 고려하여..!**



**Part 5-5.**  
**가장 단순한 패턴**  
**- API Calling**



## LLM 제일 간단한 방법 - API 콜

샘플 시나리오	<ul style="list-style-type: none"> <li>• 검사 결과를 언어 모델로 친절한 안내로 바꾼 다음에 보냄 현재 보내는 내용이 나오는 흐름에서 API call 하나 더 함</li> <li>• 이미지를 저장할 때 내용을 "읽어서" 텍스트까지 저장</li> <li>• 상담 내용 저장할 때 요약해서 저장</li> <li>• Async</li> </ul>
Authentication	시스템 내의 세팅 그대로?
Cost Control	Token cost
Data goes to OpenAI	"김와와 고객님의 피검사 결과는 다음과 같습니다. 이 부분은 상당히 염려되오니 꼭 정밀 검진 받으시기 바랍니다."
밖으로 나가는 콜 - 인프라넷이라면?	



# LLM 제일 간단한 방법 - API 콜

## 장점

- 간단함
- 언어 제한 없음

## 단점

- 어차피 시스템이 커지면 일 다시 다 해야 함
- 세부적인 부분 컨트롤이 쉽지 않음 (데이터 보안 등)
- 데이터 수집과 정리를 우선하지 않을 수 있음 (좋은 모델에 의지)
- 가격, 외부로 데이터 보내기
- Fine tuning?



**Part 5-6.**

**다른 모델을 쓰겠다!**



# LLM 다른 모델들

- 다른 모델 써보기
- HuggingFace
- LeaderBoard
- Evaluation metrics
- 한국말은??
- 우리 분야 전문용어는??



## LLM 다른 모델들

- 백날 천날 오토바이 모델 고르고 있을 수 없음.  
언젠가는 콜 받고 배달 나가야 함.
- 한 번 도입하면 쉽게 바꿀 수 없음을 고려해야 함.
- 최대한 바꿀 수 있도록 시스템 셋업






**Part 5-7.**  
**파인 튜닝한**  
**Cloud /on-prem 모델**



# On-prem? Cloud?

## 집에서 지지고 볶고 다 하기 vs 우리집 전용 식당시설 빌리기

집 모델	필요할 때마다 빌리기 (cloud)
<ul style="list-style-type: none"> <li>• 우리집 양념, 우리집 재료. 누가 손댔을까 걱정 안 해도 됨. 빠름.</li> <li>• 요리를 못하면...? 우리집이 과연 더 안전할까??</li> <li>• 요리 도구랑 양념 다 사놔야 함. 냉장고 업글 필요.</li> </ul>	<ul style="list-style-type: none"> <li>• 보안...?</li> <li>• 집에서 재료 들고 나가야 함 (아니면 재료를 그곳에 보관)</li> <li>• 계속 쓰자면 더 비쌌.</li> <li>• 어디서 뭘 가지고 어떻게 만들었는지...?</li> <li>• 업그레이드는 쉬움</li> <li>• 땀 식당으로 옮기기는 어려움</li> </ul> 

## On-prem? Cloud? 비교분석표

	사내 LLM	클라우드 호스팅 LLM
비용	하드웨어, 인프라, 유지 관리에 따른 초기 비용이 높음. (A100 하나에 만불!!)	초기 비용이 낮고 주로 사용량에 따른 운영 비용 발생.
확장성	추가 하드웨어가 필요하며 확장이 느리고 비용이 많이 듦.	요구에 따라 리소스를 쉽게 조정할 수 있어 물리적 인프라 변경 없이도 확장이 용이함.
제어	하드웨어, 소프트웨어, 데이터 보안에 대한 완전한 제어.	물리적 인프라에 대한 제어는 적지만 구성과 데이터에 대해 일정 정도 제어 가능.
유지 관리	유지 관리 및 업데이트를 위해 전담 IT 직원이 필요함. 하지만 과연 아마존보다 잘 할까 ...?	유지 관리 및 업데이트는 일반적으로 서비스 제공업체가 처리함.
보안, sovereignty	데이터 풀 컨트롤. 관련 법규 준수하기가 더 쉬움. 데이터 보안 걱정 덜 해도 됨.	데이터 보안 위험이 더 크고 관련 법규 준수하기 힘들 수 있음. 데이터가 인터넷으로 이동함.

**Part 5-8.**  
**파인 튜닝 실습**



# OpenAI 파인 튜닝

## 실습

- Fine tuning UI
- Fine tuning API



# 파인 튜닝 코드 보려면 이해해야 하는 단어 정리

**Base model** : 공부 전의 학생

**Train** : 공부!

**Tokenizer** : 책 내용을 작은 단위로 나누어 정리하는 도구

**Epochs (학습 반복 횟수)** : 학생이 교재를 처음부터 끝까지 읽는 횟수.

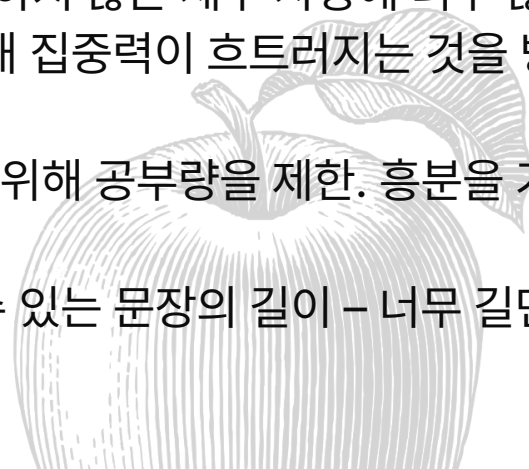
**Batch size (묶음 크기)** : 학생이 한 번에 공부하는 양 - 한 번에 한 장? 한 단원? 한 문단?

**Learning rate (학습률)** : 학생이 새로운 정보를 배우는 속도 - 너무 빠르면 이해가 부족하고, 너무 느리면 비효율적

**Weight decay (가중치 감소)** : 학생이 공부하면서 중요하지 않은 세부 사항에 너무 많은 신경을 쓰지 않도록 하기 - 이는 과도한 정보 축적으로 인해 집중력이 흐트러지는 것을 방지

**Gradient clipping** : 과도한 스트레스나 혼란을 피하기 위해 공부량을 제한. 흥분을 가라앉히고 꾸준히 학습할 수 있도록 도와줌.

**Sequence length** : 학생이 한 번에 집중해서 공부할 수 있는 문장의 길이 - 너무 길면 이해하기 힘들고 너무 짧으면 문맥 파악이 어려움.



## 파인 튜닝 코드 보려면 이해해야 하는 단어 정리

**PEFT** : 중요한 부분이나 자신에게 필요한 부분만 선택적으로 공부하는 방법입니다. 예를 들어, 시험에 나올 가능성이 높은 부분만 집중적으로 공부하거나, 이미 알고 있는 부분은 복습을 생략하고 새로운 부분만 학습하는 것

**LoRA** : 학생이 주요 개념을 이해한 후, 세부 사항을 간단하게 추가하여 이해를 확장하는 것

**QLoRA** : 학생이 중요한 개념을 기억하기 위해 핵심 내용을 요약하고 압축하는 방법. "중요한 포인트만 간단히 정리"



# 파인 튜닝 코드 보려면 이해해야 하는 단어 정리

**Base model** : 기본 모델

**Train** : 학습

**Tokenizer** : 텍스트를 의미 단위로 나누는 도구

**epochs** : 전체 학습 데이터셋이 신경망을 통해 앞뒤로 통과되는 횟수

**Batch size** : 한 번의 반복에서 사용되는 학습 예제의 수

**Learning rate** : 최적화 과정에서의 스텝 크기를 제어

**Weight decay** : 큰 가중치를 페널티하여 과적합을 방지하는 정규화 기법

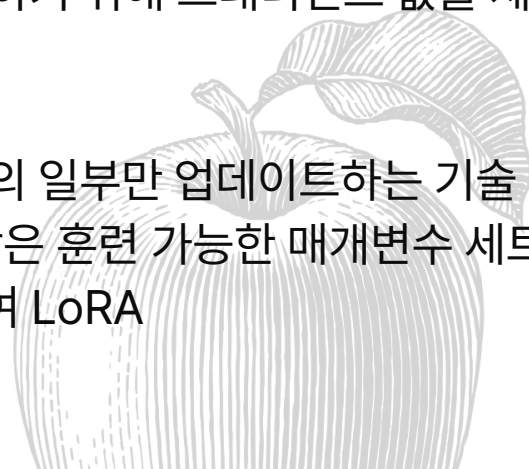
**Gradient clipping** : 그래디언트 값이 폭발하는 것을 방지하기 위해 그래디언트 값을 제한

**Sequence length** : 입력 시퀀스의 최대 길이

**PEFT (Parameter-Efficient Fine-Tuning)**: 매개변수의 일부만 업데이트하는 기술

**LoRA** : 사전 학습된 모델의 가중치를 고정하고 각 계층에 작은 훈련 가능한 매개변수 세트를 주입

**QLoRA** : QLoRA는 모델 가중치를 양자화(정밀도 감소)하여 LoRA





## 다른 모델 파인 튜닝

- LLM 튜토리얼 중 아마도 가장 흔함
- 모델 로딩 – 데이터 로딩 – 파라미터 세팅 – 훈련 – 테스트 – 설치

## 실습

- [Deeplearning.ai](#)
- [Llama2](#)
- [Llama3\\_1, Alpaca + Llama 3](#)
- [Amazon Jumpstart, 요약버전](#)
- [Unsloth-QLoRA](#)



**Part 5-9.**

**직접 foundation model 만  
들기**



# LLM 모델을 직접 만든다면?

만들 사정은 안 되지만 직접 만든다면??

## 1. 데이터 수집 및 전 처리

- 교육에서 제일 중요한 것은 교육!! 퀄리티 교과서!! 그리고 12년 간의 교육 과정!!
- 교과서 편찬도 쉽지 않은데 아무것이나 먹일 수는 없음
- 데이터 수집, 정리, 표준화 등등

## 2. 어떤 "학생"을 고를 것인가? 여러가지 모델 아키텍처가 존재함

- (Transformer, GPT-3, Megatron-Turing NLG 등)
- 학습 방법 결정
- 학습을 시키자! \$\$\$\$\$
- NVidia 의 주가가 높은 이유...



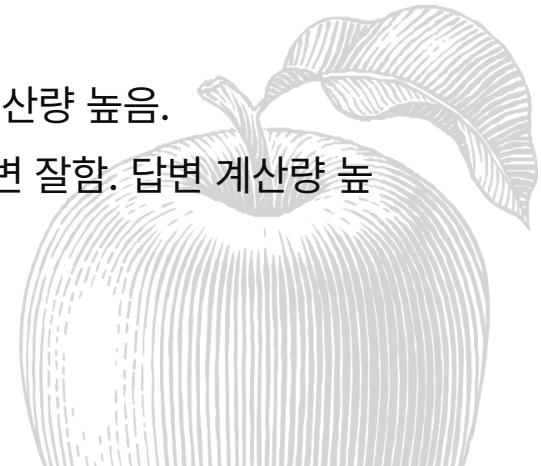
## (자주 나오는 단어 정리)

### Transformer

- 구글 출신! 2017년 출신이고 가장 대표적인 모델 아키텍처
- 앞뒤만 보는게 아니라 조금 더 길게 봐서 문맥이해에 월등함
- 인코더+디코더: 입력 텍스트 처리는 인코더, 출력 텍스트는 디코더
- 다양한 파생 모델
  - 학습 데이터 다다익선
  - 아니 그래서 그걸 어디서 배워왔냐고...? 블랙박스 모델

### 그 외

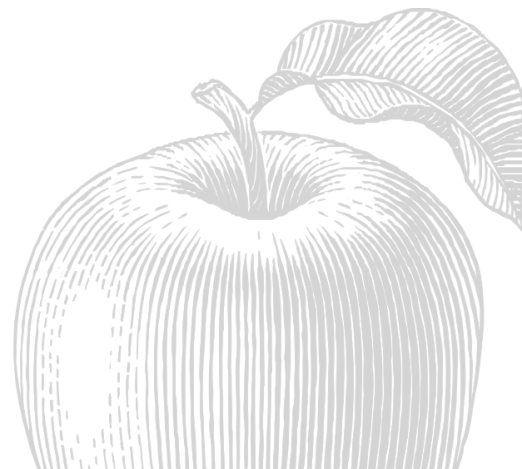
- BERT (구글 2018): 인코더 집중. 이해에 집중. 모델 크고 계산량 높음.
- GPT-3 (OpenAI): 디코더 집중. 생성에 초점. 창의적임. 답변 잘함. 답변 계산량 높음.



# LLM 모델을 직접 만든다면?

## 3. 모델 평가 및 개선

- 여러가지 평가 방법 - BLEU, ROUGE, CIDEr
- 모델 개선, 추가 학습, 하이퍼파라미터 조정  
(학습률, 배치 크기, 에포크 수, hidden layer 수, 뉴런 수 etc)
- 반복



# LLM 모델을 직접 만든다면? 4

## 4. 배포 및 사용

- 아무리 훌륭한 직원이라도 창고에 가둬 놓으면...?
- 효율적으로 일할 수 있는 환경과 시스템이 필요함
  - API 서버 구축, 모델 최적화, 관시 시스템, 유저 인터페이스, 튜토리얼, LLMOps 툴 etc

## 5. 홍보!

- 투자금액 회수...?
- 언어 모델 벤치마크 테스트 도전
- 킬러 앱?



**Part 5-10.**

**LoRA, Quantization**



# LoRA

- 아 제발 다시 학습한다고 하지 마세요  $\pi.\pi$
- 파인 튜닝 해야 하긴 하는데...
  - LoRA: Low Rank Latent Representation - 저차원, 나중에 표현
  - 모델 크기와 학습 시간을 크게 줄이면서 성능은 향상
- 파인 튜닝: 다시 재교육
- LoRA: 어댑터 위에 끼우는 느낌
  - 저차원 행렬 (matrices) 사용해서 가중치를 수정 (나중에)





# Quantization 양자화

- 양자화가 뭘 말아야  $\pi.\pi$ :
  - 물리학의 “양자화”
  - 계속되는 값이 아니라 뚝뚝 떨어지는 불연속적인 값
  - 근사치로 표현
- 단순화 해서 모델 크기 / 메모리 공간을 줄여 속도 높이고 좋은 하드웨어 없어도 돌아감. 전력 소비도 줄임.
- 모바일 폰, 임베디드 시스템.. 작은 모델 많이 필요함
- 정확도 손실, 하드웨어 호환성



## QLoRA – Quantized LoRA

- 양자화된 LoRA
- 모델 가중치를 양자화하고 LoRA 를 기반으로 함.
- 메모리 사용량과 컴퓨팅 요구사항을 더욱 줄임
- 일반 소비자용 하드웨어에서도 모델 미세조정 가능
- LoRA 보다 훨씬 효율적임



# QLoRA – Quantized LoRA 실습

- QLoRA 노트북



**Part 5-11.**

**내 컴퓨터에서 LLM을  
돌려보자 - Ollama**



# Local LLM - Ollama

- API key 없이, 인터넷 연결 안하고, 그냥 내 컴퓨터에서 돌릴 수 없나요?
- Ollama - <https://ollama.com/>
- 소개, 다운로드, 설치
  - 모델 선택
  - <https://github.com/ollama/ollama>
- Cmd 에서 돌리기
- Custom instruction, 모델 세이브
- 주피터 노트북에서 돌리기
- Phi3 파인튜닝 샘플:

