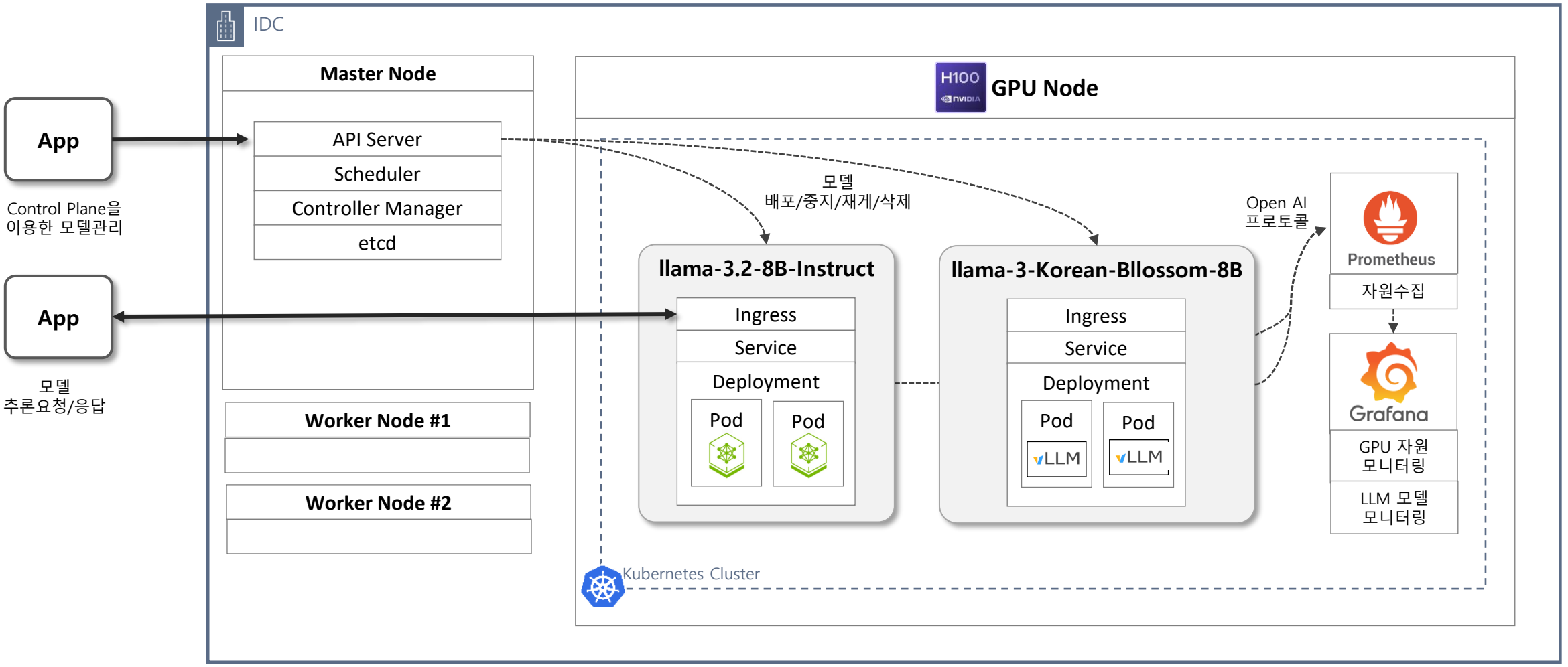
LLM 서비스 구축 관련자료

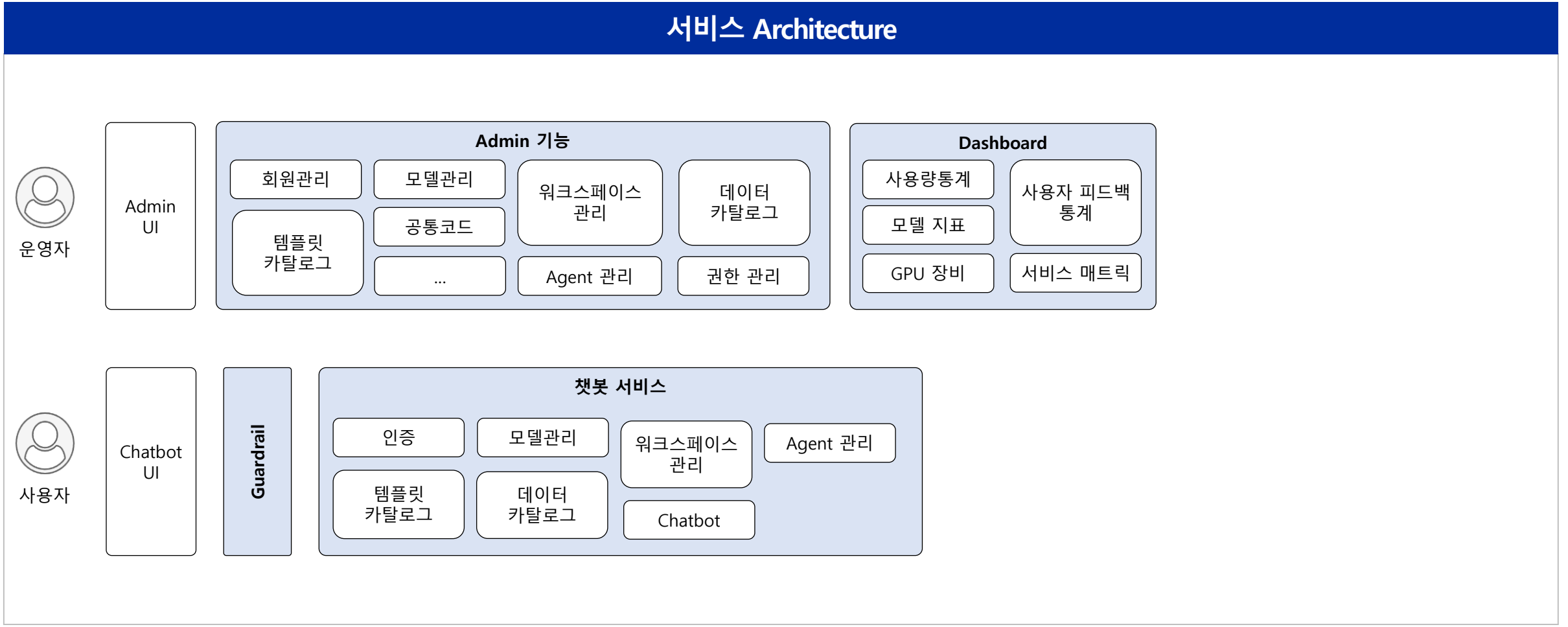
Control Plane Service Architecture

Control Plane은 K8S API를 활용해 모델 관리, 모델 배포, 모델 추론요청을 처리한다. NIM 또는 vLLM으로 생성된 모델은 Ingress를 통해 추론 요청을 처리하고 Serving된 모델의 매트릭스와 GPU 관련 사용정보를 Prometheus에서 수집해서 Grafana Dashboard로 모니터링 한다.

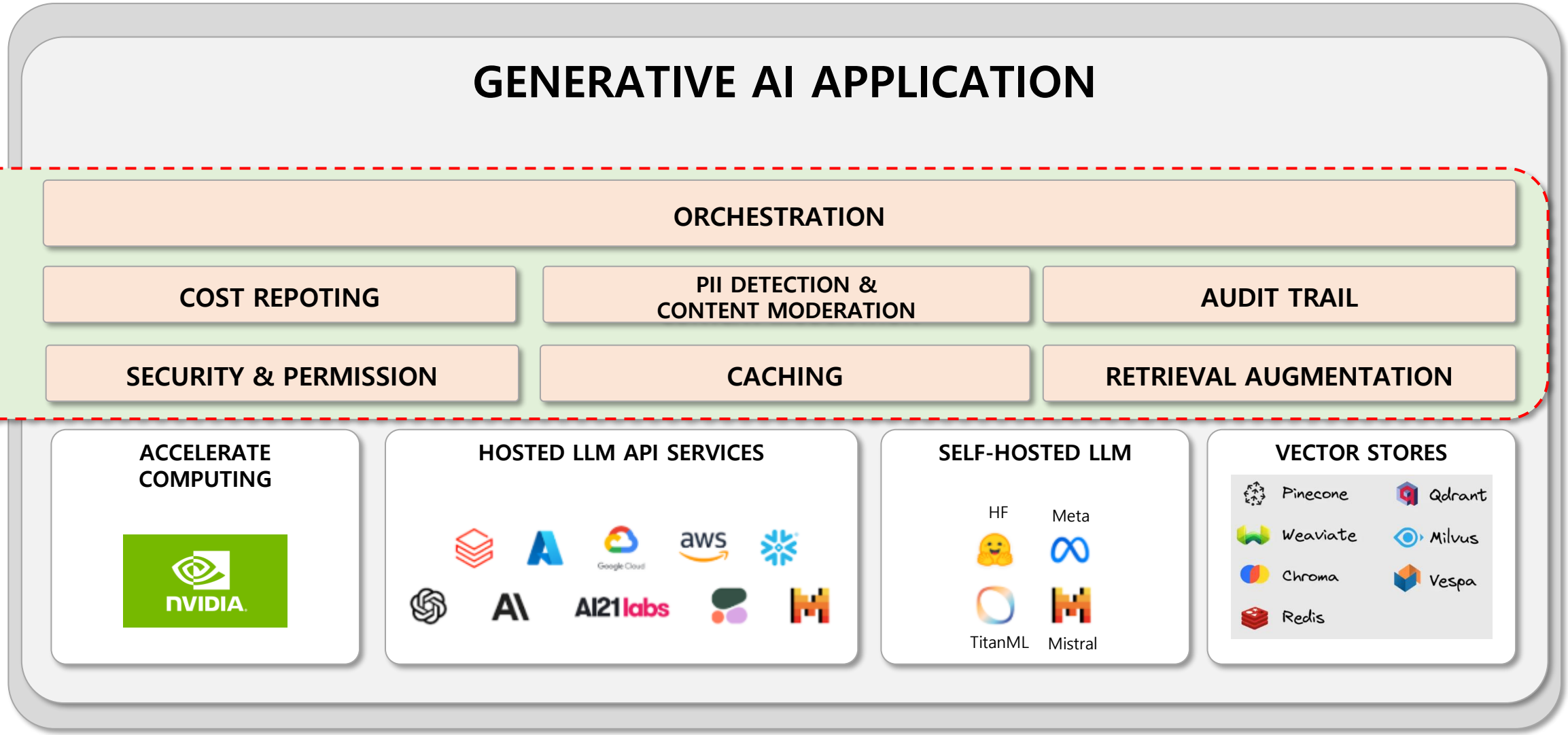


AI 챗봇 서비스 아키텍처

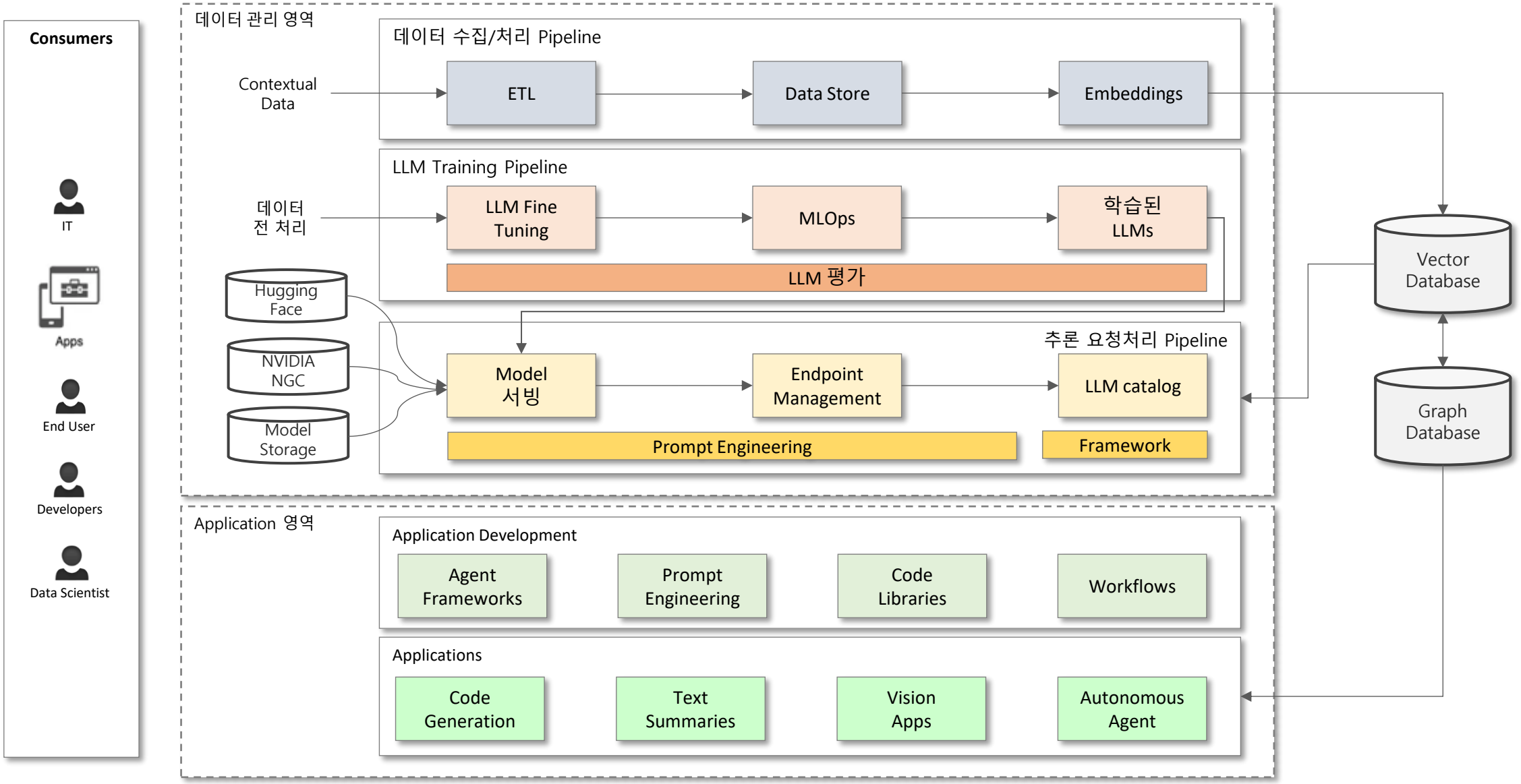
챗봇 서비스 아키텍처 설명...



LLM Mesh Component 구성도

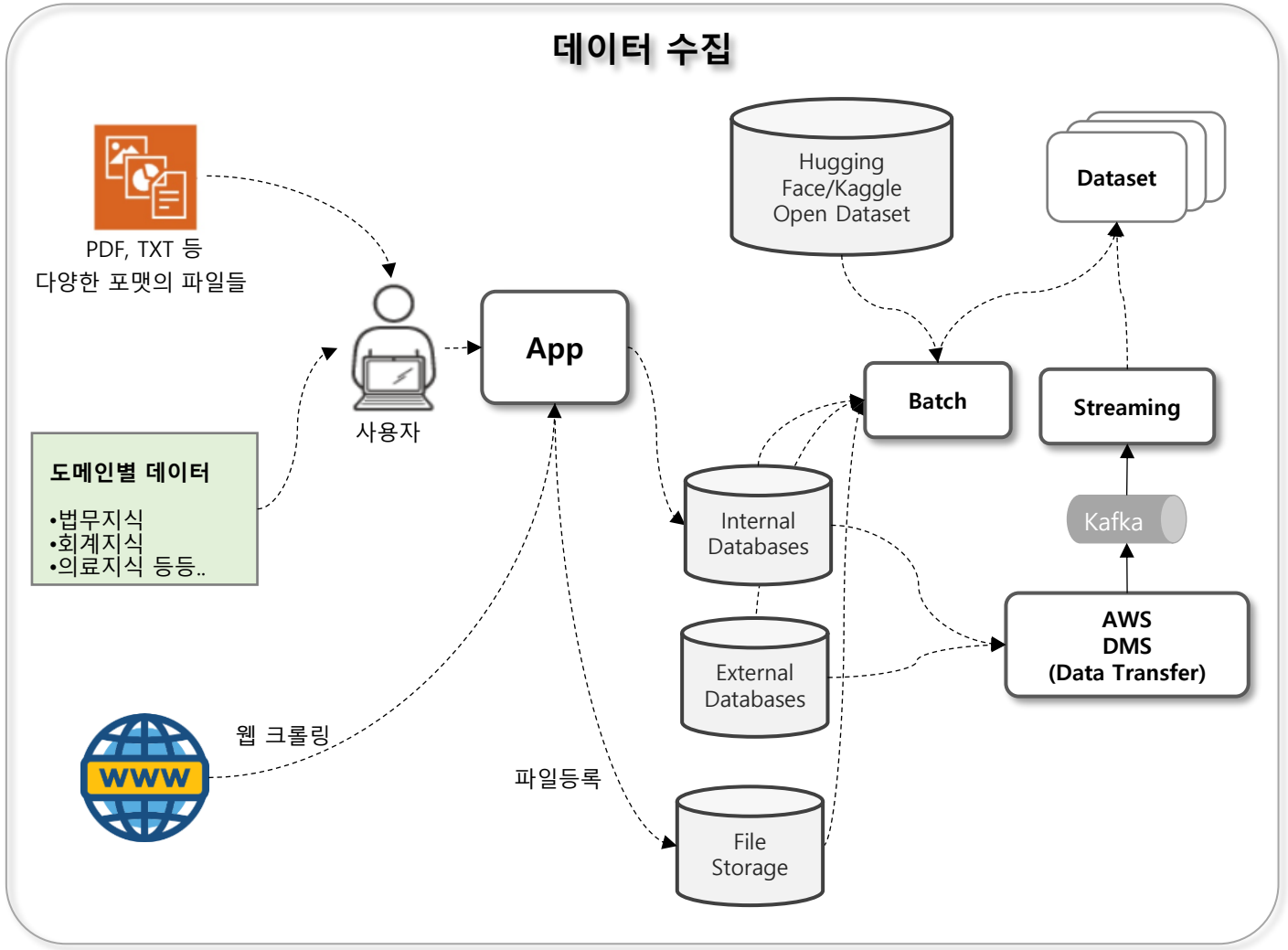


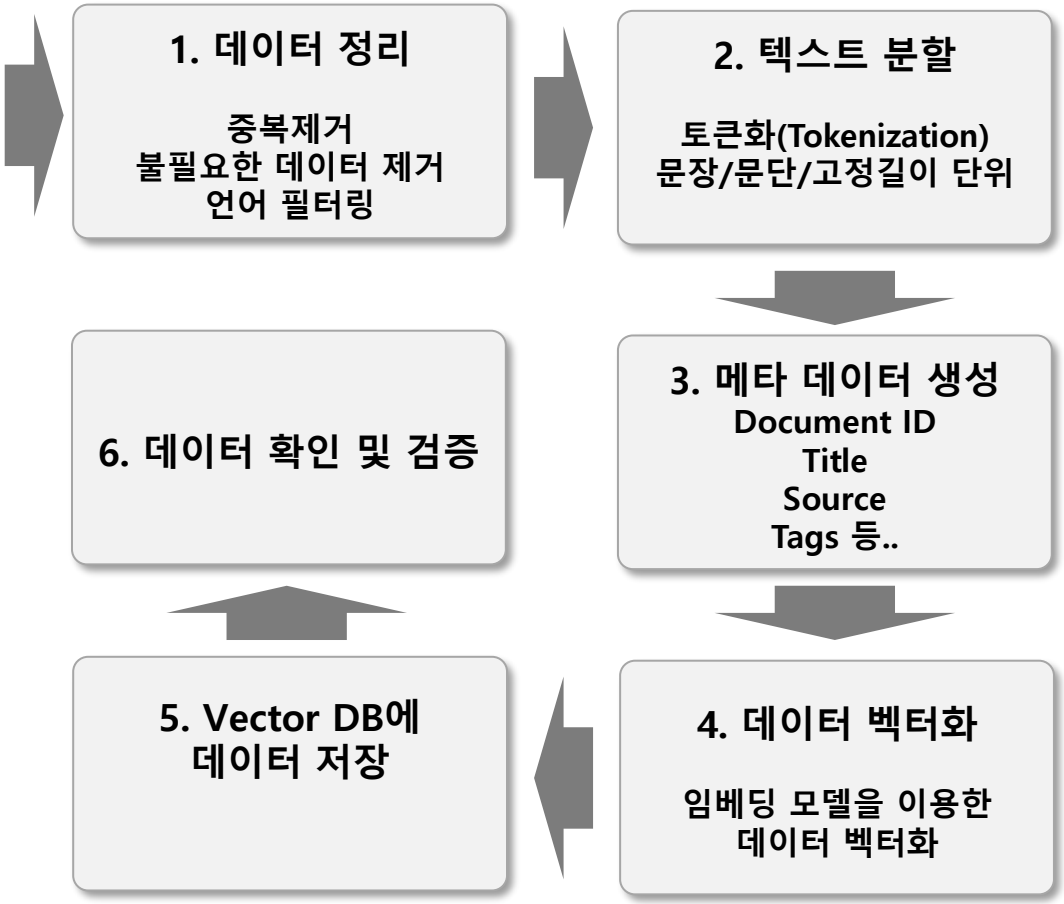
LLM Overall Architecture



RAG 데이터 수집방법 및 단계별 처리과정

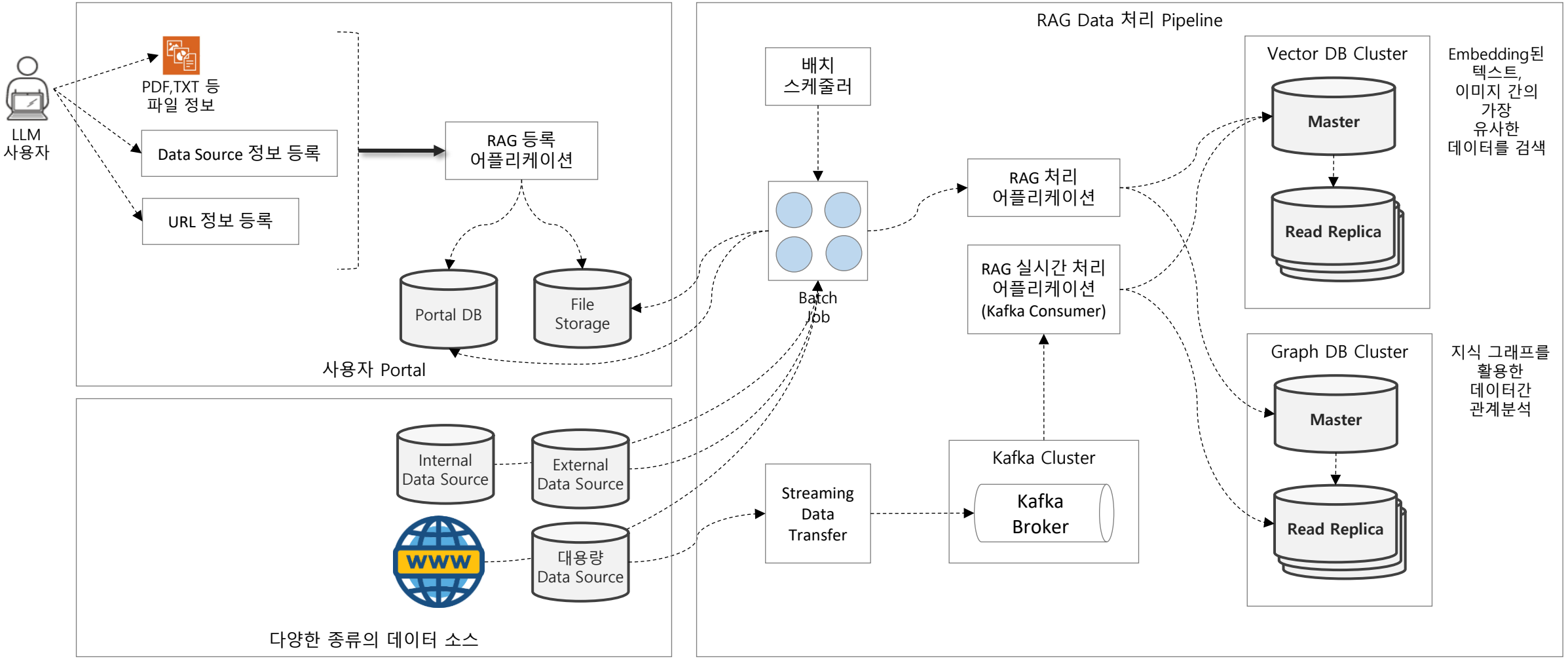
RAG 데이터 처리를 위해서는 데이터를 수집, 변경, 저장하는 전 처리 과정이 필요하다.





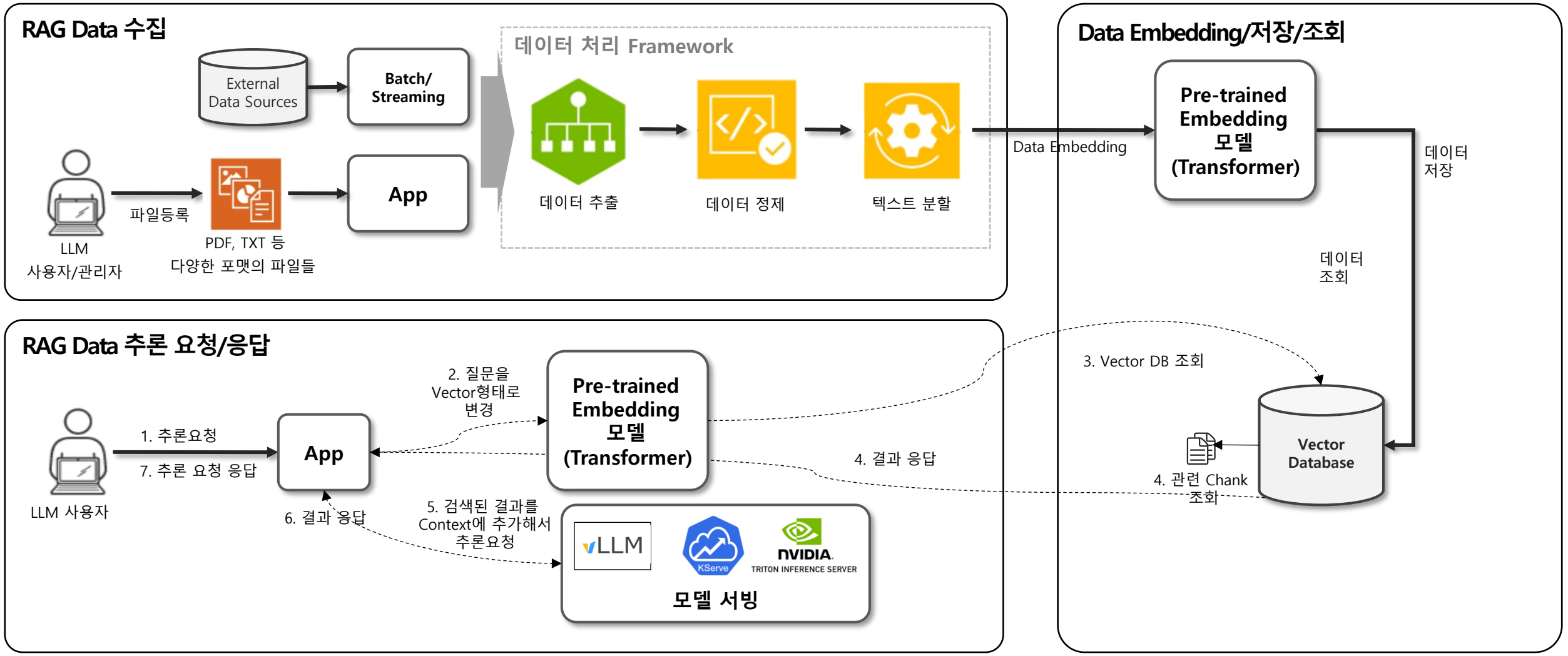
RAG Data 수집/저장 서비스 Architecture

다양한 종류의 데이터 소스를 이용해서 데이터를 수집하고 수집된 데이터를 배치 또는 대용량 데이터 실시간 처리를 통해 Vector DB에 저장합니다.
Vector DB Cluster를 이용해 대용량 실시간 데이터는 Master Node에 저장하고 데이터 수집에는 Read Replica를 활용해서 처리 성능을 향상시킬 수 있습니다.



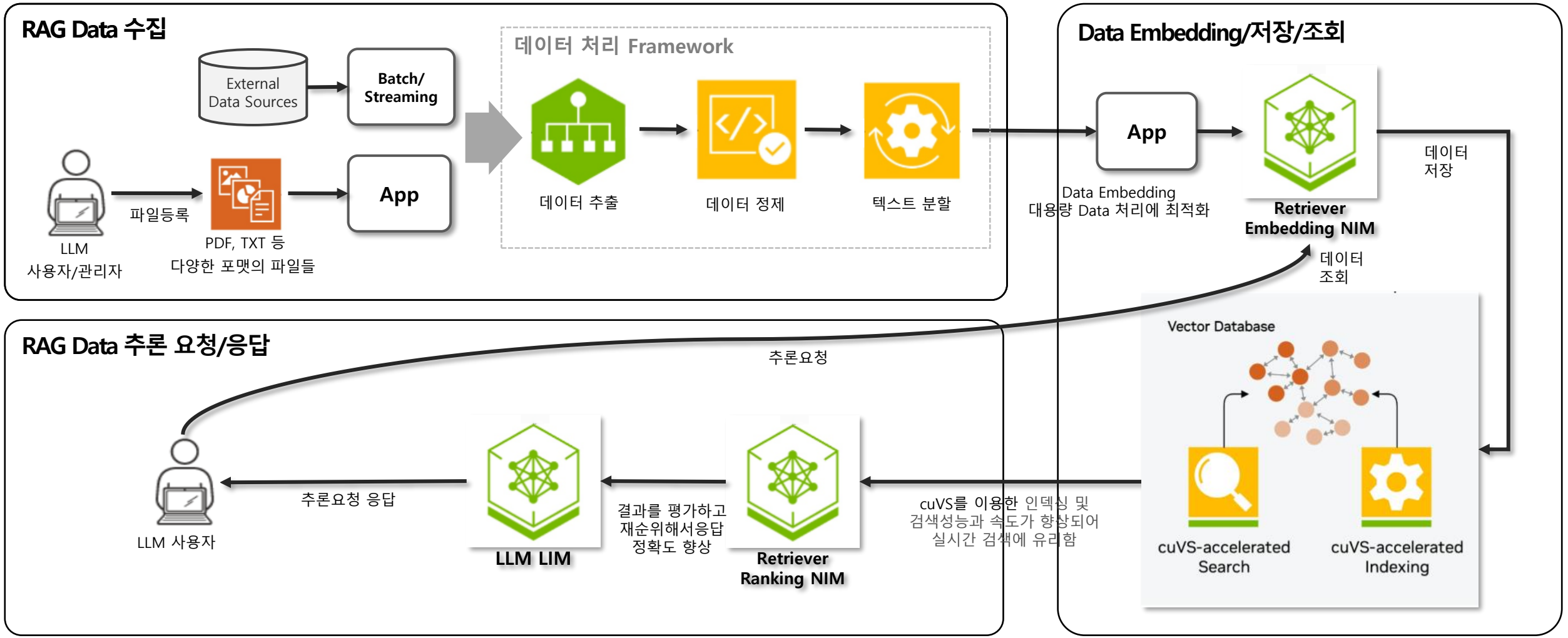
vLLM을 활용한 일반적인 RAG 처리 프로세스

사용자가 등록한 파일이나 외부 저장소에 등록된 데이터를 데이터 처리 Framework을 이용해서 처리하고 Embedding된 모델을 이용해 벡터화해서 Vector DB에 저장한다. RAG된 데이터는 사용자의 추론 요청시에 Vector DB에서 조회해서 추론요청 Context에 추가한 뒤 추론 요청/응답한다.



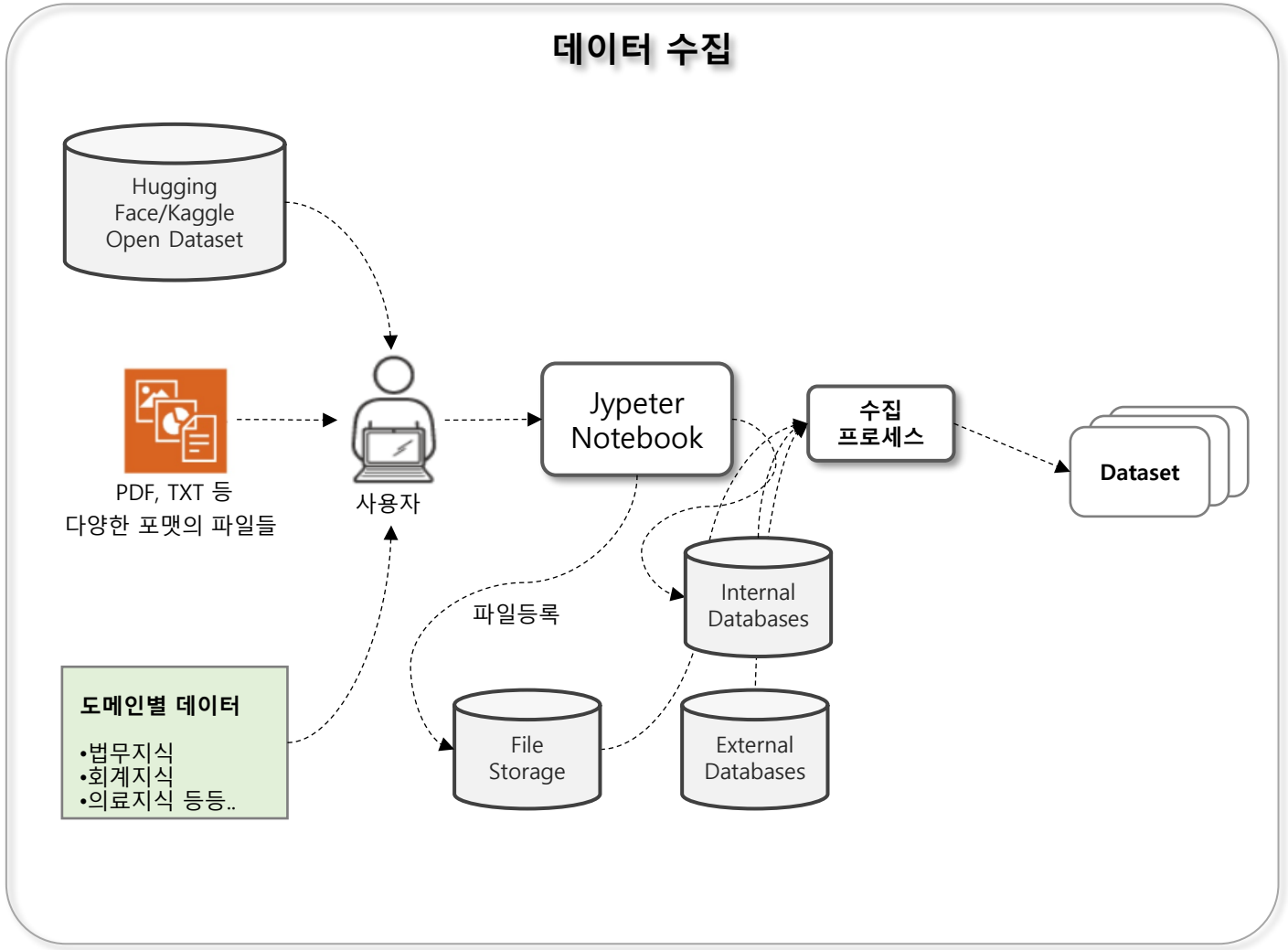
NeMo Retriever NIM Container를 활용한 Advanced RAG 처리 프로세스

입력된 데이터를 기반으로 Nemo Framework을 이용해 데이터를 처리한다. Retriever NIM Container를 이용해 Vector DB에 Data 저장 및 cuVS를 이용해 데이터를 조회하고 사용자의 추론 요청에 응답한다. Retriever NIM Container를 이용하면 대용량 데이터 처리에 최적화 되고 추론 요청에 대한 정확도가 향상된다.



Fine-Tuning 데이터 수집방법 및 단계별 전 처리 과정

Fine-Tuning을 위해서는 데이터를 수집, 변경, 저장하는 전 처리 과정이 필요하다.



1. 데이터 클렌징

중복제거
불필요한 데이터 제거
언어 필터링

2. 데이터 정규화

토큰화(Tokenization)
텍스트 정규화
특수문자처리

3. 데이터 Formatting

Text, Json, CSV
형식으로 변경

4. 데이터 검증

품질검사
예제 샘플링

5. 저장 및 로드 (내용수정!!!)

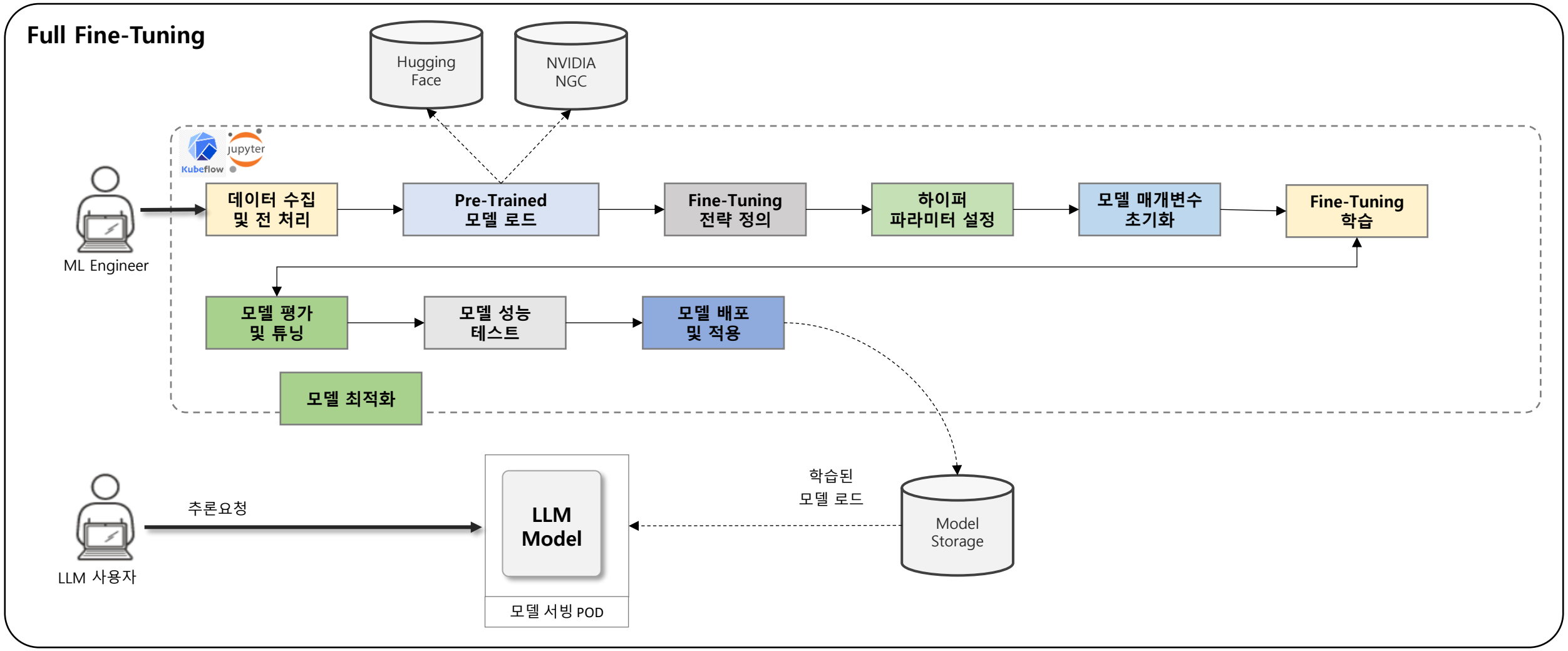
Text, Json, CSV
형식으로 저장

6. 모델과 데이터 호환성 검증

샘플 입력 테스트

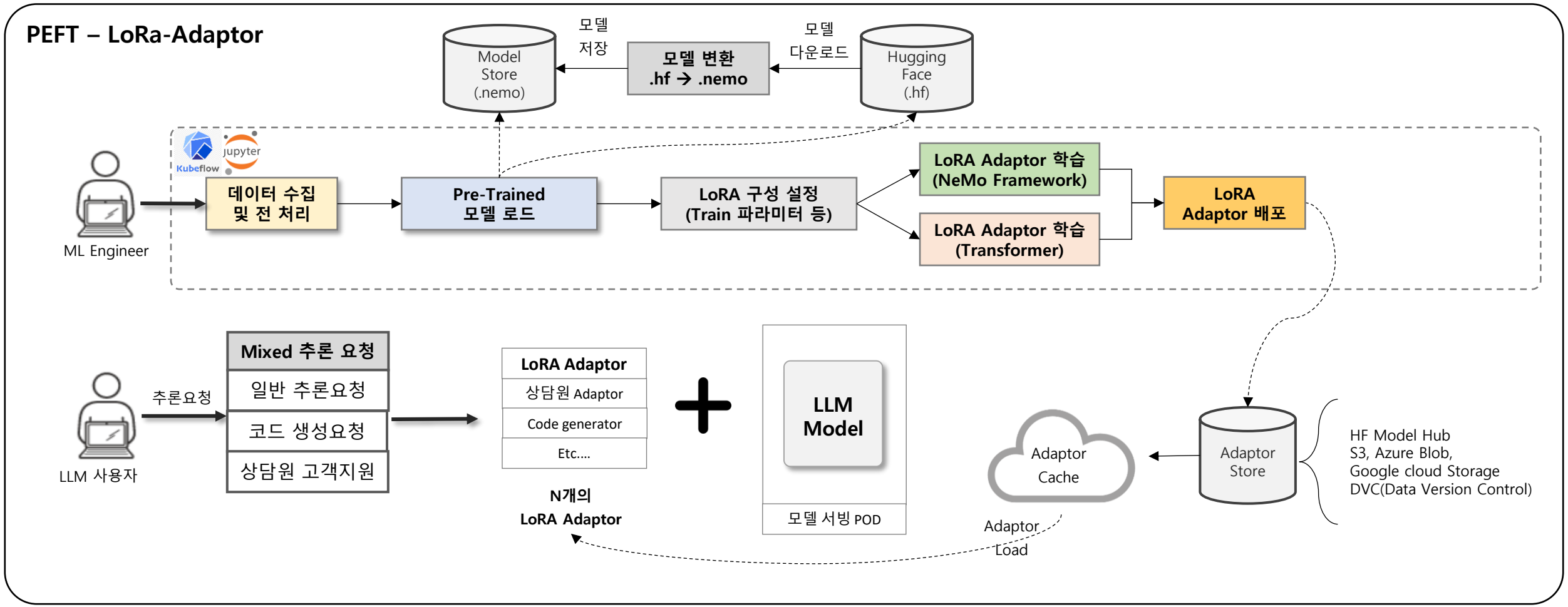
Full Fine-Tuning 처리 프로세스

Kubeflow의 Jypeter Notebook을 이용한 Fine-Tuning 데이터 처리 및 학습과정을 Kubeflow Pipeline으로 구성하고 모델 평가과정을 거쳐 배포한다. Full Fine-Tuning의 경우 많은 양의 메모리와 시간이 필요하다.



Transformers와 PEFT 이용한 LoRA Adaptor 학습 프로세스

LoRA Adaptor를 이용한 학습은 기존 모델의 가중치를 변경하지 않고 Kubeflow의 Jypeter Notebook과 NeMo Framework을 이용해 학습시키고 평가 과정을 거쳐서 배포한다. 사용자는 Mixed 추론 요청을 사용해서 하나의 모델에 여러가지 Adaptor를 결합해서 처리하고 결과를 응답 받을 수 있다. PEFT를 이용한 모델 학습은 Full Fine Tuning에 비해서 적은 자원과 시간을 이용해 학습이 가능하고 한가지 모델에 여러 Adaptor를 포함시켜서 여러가지 추론 요청 처리가 가능하게 해준다.



* PEFT: Parameter-Efficient Fine-Tuning

- 필요에 따라.nemo 변환 없이 Hugging Face Transformers 등 기타 라이브러리를 이용해 LoRA 학습 가능함.

모델 평가 방법 - Perplexity

Perplexity (PPL) 를 활용하는 방법

Perplexity (PPL)란?

Perplexity (혼란도) 는 모델이 테스트 데이터의 단어 시퀀스를 얼마나 "예측하기 어려운지"를 측정하는 지표입니다.
값이 낮을수록 모델이 더 좋은 성능을 보인다는 의미입니다.
사용 예: Hugging Face의 evaluate 라이브러리와 transformers 라이브러리의 logits_processor를 사용

Perplexity 평가의 한계 및 보완 방법

→ Perplexity만으로는 문장의 품질을 완전히 평가하기 어렵다.

→ Perplexity가 낮다고 해서 반드시 인간이 보기에도 자연스러운 문장이라는 보장은 없음.
이를 보완하기 위해 BLEU, ROUGE, METEOR 같은 지표와 함께 평가하는 것이 중요함.
Fine-tuning된 모델이 특정 도메인에서는 Perplexity가 낮지만, 일반성은 떨어질 수 있음.

→ 특정 데이터셋에서만 성능이 좋은 지 확인하려면 다운스트림 태스크 (QA, 요약 등) 테스트
Perplexity는 모델의 분포를 측정하는 것이므로, 직접적인 의미 평가가 어렵다.
사람이 직접 평가하는 Human Evaluation이 필요할 수도 있음.

```python
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer
import evaluate

# 평가할 모델 및 토크나이저 불러오기
model_name = "[평가 대상 모델 명명]"
model = AutoModelForCausalLM.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)

# Perplexity 평가를 위한 라이브러리 로드
perplexity = evaluate.load("perplexity", module_type="metric")

# 평가할 문장 (샘플 텍스트)
text = "Hugging Face models are great for NLP tasks."

# 토큰화 수행
inputs = tokenizer(text, return_tensors="pt")

# 모델을 사용해 Perplexity 계산
with torch.no_grad():
    ppl_score = perplexity.compute(model_id=model_name, input_texts=[text], batch_size=1)

print(f"Perplexity: {ppl_score['perplexities'][0]}")

#출력예시
Perplexity: 32.5
```

모델 평가 방법 – BLEU, ROUGE, METEOR

BLEU, ROUGE, METEOR를 활용하는 방법

BLEU란?

BLEU(Bilingual Evaluation Understudy)는 기계 번역 및 텍스트 생성 모델을 평가할 때 사용되며, 정답 문장(reference)과 모델이 생성한 문장(candidate) 간의 N-그램(N-gram) 유사도를 기반으로 계산됩니다.
주로 1-그램(BLEU-1)부터 4-그램(BLEU-4)까지 사용합니다.
완전 일치할 경우 BLEU 점수는 1.0 (100%)이고, 완전히 다를 경우 0에 가까운 값이 나옵니다.

ROUGE란?

ROUGE(Recall-Oriented Understudy for Gisting Evaluation)는 요약 모델 평가에 자주 사용되는 지표입니다.

주요 ROUGE 지표:
ROUGE-N: N-그램 기반 유사도 측정 (ex: ROUGE-1, ROUGE-2)
ROUGE-L: 문장 내 최장 공통 부분(LCS, Longest Common Subsequence) 기반 평가
ROUGE-W: 가중치를 적용한 ROUGE-L

METEOR란?

METEOR(Metric for Evaluation of Translation with Explicit ORdering)는 BLEU보다 더 정교한 방식으로 평가하며, 동의어(synonyms) 및 형태소 분석(lemma)을 고려하여 점수를 계산합니다.
문장 구조가 다르더라도 유사한 의미를 가질 경우 BLEU보다 더 정확한 평가를 제공합니다.

```python
import evaluate

# BLEU metric 로드
bleu = evaluate.load("bleu")

# 모델이 생성한 문장 (Candidate)
candidate = ["The cat is on the mat"]

# 정답 문장 (Reference)
reference = [["The cat is sitting on the mat"]]

# BLEU 점수 계산
result = bleu.compute(predictions=candidate, references=reference)

print(f"BLEU Score: {result['bleu']:.4f}")

#출력예시
BLEU Score: 0.7598
```

모델 평가 방법 – LLM 기반 자동 평가(LLM-based Evaluation)

LLM 기반 자동 평가 모델에는 여러 가지가 있으며, 대표적인 모델은 다음과 같습니다.

LLM 기반 자동 채점 모델을 활용하면 단순한 N-그램 기반 평가(BLEU, ROUGE, METEOR)보다 문맥 이해, 의미 유사성, 창의성 평가가 가능해집니다.

→ 추천 모델 별 활용 용도

GPT-4 → 일반적인 자동 채점, 코드 평가, 논리적 분석

Claude 2/3 → 문서 및 법률 평가, 대화 평가

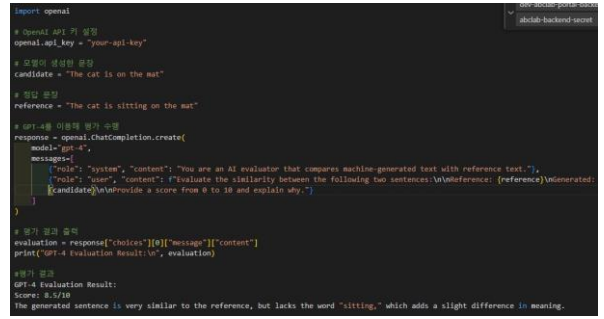
Gemini → 다중 모달(이미지 포함) 데이터 평가

LLaMA 3 → 오픈소스 기반 커스텀 평가 가능

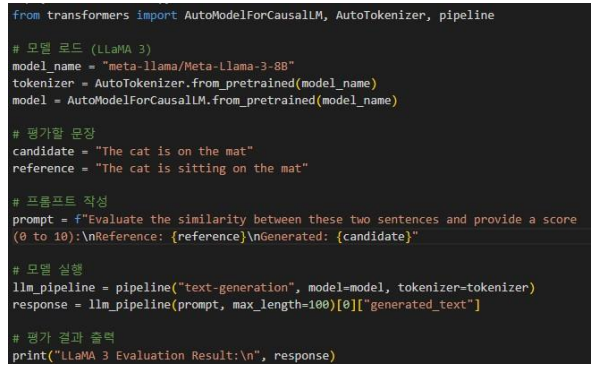
Mistral 7B → 가벼운 모델 평가 수행

모델	제공업체	주요 특징	사용 가능 여부
GPT-4	OpenAI	강력한 이해력 및 논리력, 코딩/논문 평가 가능	API 사용 가능 (유료)
Claude 2/Claude 3	Anthropic	법률, 문서 이해력 강함, 안정적 답변 제공	API 사용 가능 (유료)
Gemini (Bard)	Google DeepMind	다중 모달 처리(텍스트, 이미지) 가능	API 사용 가능 (유료)
LLaMA 3	Meta	오픈소스, 커뮤니티 지원	오픈소스 사용 가능
Mistral 7B/ Mixtral	Mistral AI	경제적이면서 강력한 성능, 오픈소스	오픈소스 사용 가능
Command R+	Cohere	지식 기반 문서 요약 및 평가 강점	API 사용 가능 (유료)

모델	문맥 이해	창의적 평가	코드 평가	데이터 분석	속도	비용
GPT-4	✅ 매우 강함	✅ 창의적	✅ 우수	✅ 가능	중간	비쌈
Claude 2/3	✅ 매우 강함	✅ 논리적	✅ 우수	✅ 가능	빠름	비쌈
Gemini	✅ 강함	✅ 다중 모달	✅ 가능	✅ 가능	빠름	비쌈
LLaMA 3	✅ 좋음	❌ 제한적	❌ 제한적	❌ 제한적	중간	무료
Mistral 7B/Mixtral	✅ 좋음	❌ 제한적	❌ 제한적	❌ 제한적	빠름	무료
Command R+	✅ 강함	✅ 논리적	✅ 가능	✅ 가능	중간	중간



GPT4 평가 예제 소스



LLaMA 3 평가 예제 소스

모델 평가 방법 – Human-in-the-loop 기반 LLM 평가

설명: 사람이 직접 생성된 텍스트의 품질을 평가하는 방식

평가 기준:
Fluency (유창성): 문법적으로 자연스러운 문장인가?
Relevance (적합성): 질문이나 맥락에 맞는 적절한 답변인가?
Consistency (일관성): 이전 내용과 논리적으로 연결되는가?
Bias & Fairness (편향성): 사회적 편향을 포함하고 있는가?

장점:
가장 신뢰할 수 있는 평가 방법
실제 사용자 경험과 가장 유사

단점:
비용과 시간이 많이 듦
주관성이 개입될 가능성이 높음

사용 가능한 솔루션
→ Amazon Mechanical Turk (MTurk)
→ Scale AI
→ Prolific (사용자 대상 피드백 수집)

1. Amazon Mechanical Turk (MTurk) 를 활용한 평가
Amazon Mechanical Turk (MTurk)는 클라우드 소싱 기반 평가 시스템입니다.
수천 명의 실제 작업자를 활용하여 LLM 모델의 출력을 평가할 수 있습니다.

Step 1. MTurk에서 Human Intelligence Task (HIT) 생성
MTurk 작업자들이 수행할 수 있는 평가 태스크를 정의합니다.
예를 들어, "LLM이 생성한 문장을 1~5점으로 평가하세요."

Step 2. 평가 폼 설계
Amazon MTurk의 웹 UI 또는 API를 이용해 평가 인터페이스를 구성합니다.
HTML 기반 설문지 또는 Python으로 API를 활용하여 데이터를 업로드할 수 있습니다.

Step 3. 평가 수행
MTurk 작업자들이 생성 된 문장을 보고 점수를 매기고 피드백을 작성합니다.

Step 4. 데이터 분석
MTurk에서 수집한 평가 데이터를 Python 또는 AWS 데이터 서비스에서 분석하여 점수를 계산합니다.

2. Scale AI 를 활용한 평가
Scale AI는 AI 학습 데이터를 수집, 정제 및 평가하는 엔터프라이즈 레벨 플랫폼입니다.
특히 대규모 모델 평가 및 Fine-Tuning 데이터 생성에 많이 사용됩니다.

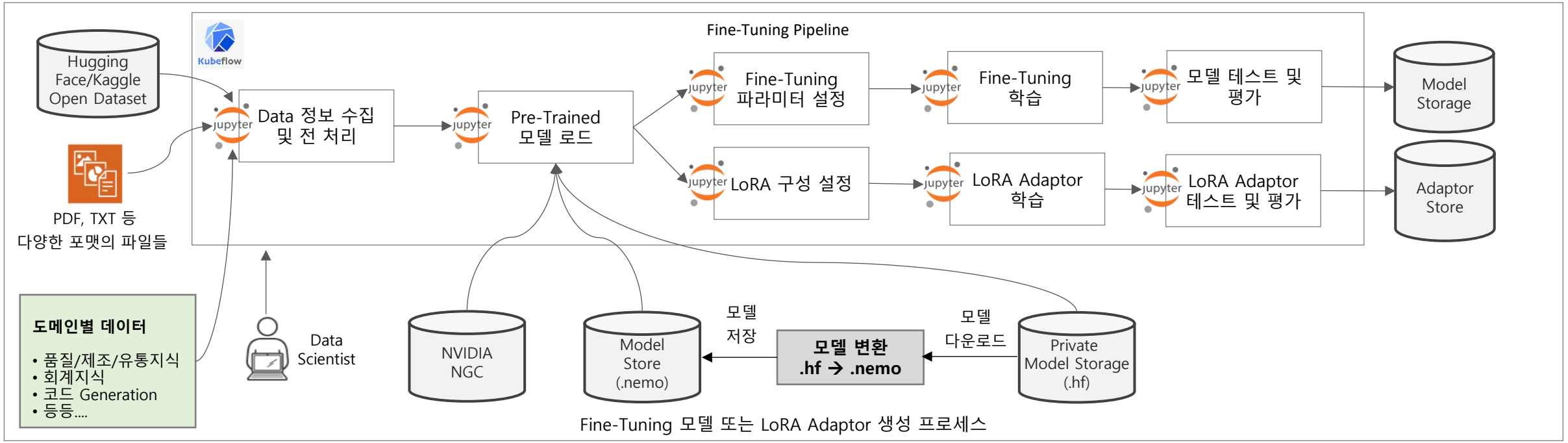
2.1 평가 방식
LLM 응답 품질 검증
모델이 생성한 문장이 의미론적으로 타당한지 평가
인공지능 감지(AI Detection) 및 편향 분석
모델이 사실적으로 정확한지 확인
데이터셋 구축 및 자동 평가
AI 훈련을 위한 고품질 라벨링 데이터 생성

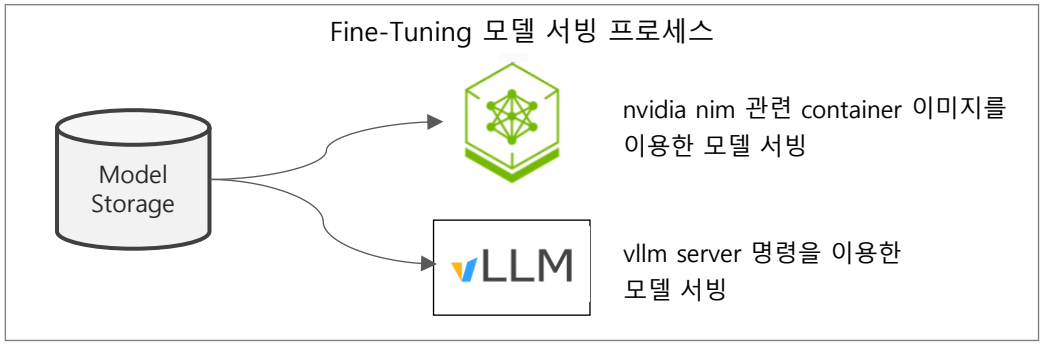
3. Prolific (사용자 대상 피드백 수집) 를 활용한 평가
Prolific은 일반 사용자를 대상으로 AI 평가를 수행하는 클라우드 소싱 플랫폼입니다.
실제 사용자들의 평가를 반영할 수 있기 때문에 제품 개발과 LLM 모델 튜닝에 매우 유용합니다.

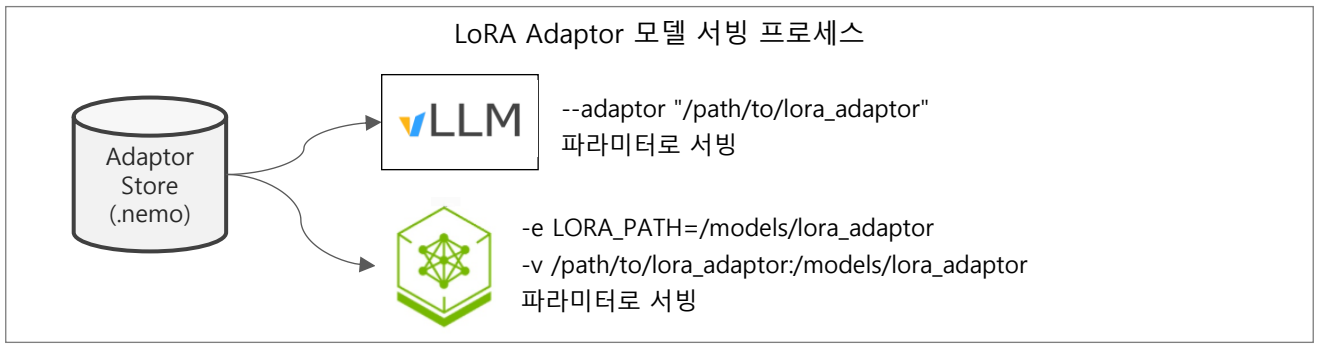
3.1 평가 방식
LLM이 생성한 문장을 사용자들에게 제공
실제 사람들이 응답을 평가 (자연스러움, 편향, 이해 가능성 등)
설문조사를 통해 피드백을 수집
평가 데이터를 모델 개선에 활용

Fine-Tuning 서비스 아키텍처

Jupyter Notebook을 이용해 단계마다 필요한 로직을 작성해서 Kubeflow Pipeline 생성해서 Fine-Tuning 절차를 생성합니다.
생성된 모델은 Model Storage에 저장하고 vLLM이나 NIM 형태로 모델을 서빙해서 서비스 합니다.







정형/비정형 데이터 수집 프로세스

정형/비정형 데이터 수집 프로세스는 요구사항 정의에서 시작해, 데이터 소스 식별, 수집 방법 설계, 수집, 전 처리, 저장 및 관리, 검증 및 품질 관리, 활용 준비의 단계를 거칩니다. 각각의 단계는 데이터를 효과적으로 수집하고 저장하며 활용하기 위한 체계적인 과정으로 구성되어 있습니다.

데이터 수집

다양한 데이터 소스에서 데이터를 안정적으로 수집하고 적재 데이터는 배치 방식(ETL)과 실시간 스트리밍 방식(CDC)으로 가져옵니다.

데이터 전송

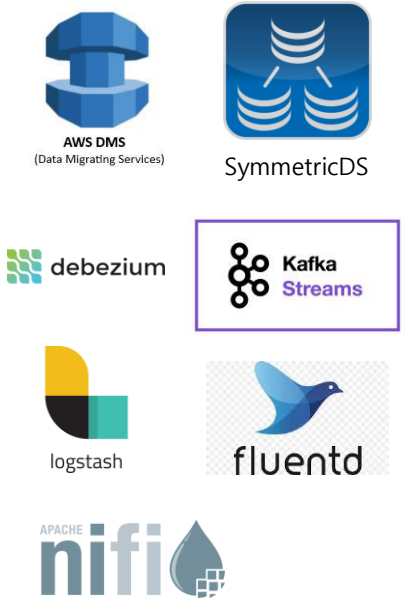
수집한 데이터를 중앙화 된 메시징 시스템을 통해 저장소나 Down Stream 서비스로 안정적으로 전송합니다.

데이터 저장

데이터의 유형(로그, DB TEXT, JSON 등)과 크기, 접근 패턴에 따라 다양한 저장소를 선택합니다. 비정형 데이터를 효율적으로 저장하고 관리합니다.

데이터 처리/분석

저장된 데이터를 기반으로 배치, 실시간 스트리밍 분석, 머신 러닝 등 다양한 분석 작업을 수행



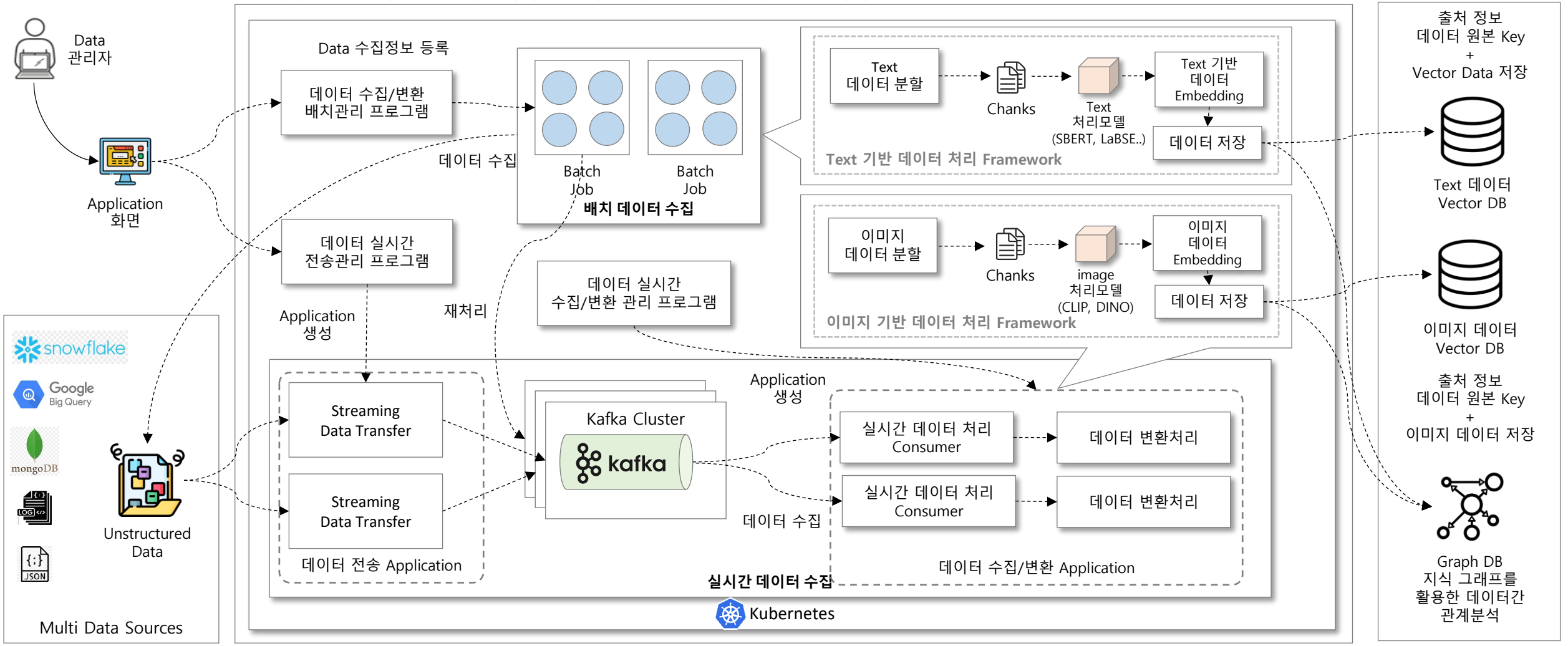
CDC를 이용한 실시간 데이터 수집 솔루션

실시간 데이터 처리를 위한 방법 중 하나인 CDC(Change Data Capture)를 통해 데이터 변경 사항(Insert, Update, Delete)을 실시간으로 대상에 반영할 수 있게 해주는 솔루션입니다.

기능	AWS DMS	Debezium	Maxwell	SymmetricDS	Apache Kafka Connect
지원 DB	RDS, Aurora, DynamoDB 등	MySQL, PostgreSQL 등	MySQL	다양한 DB 지원	다양한 DB 지원
CDC 방식	자동화된 CDC	트랜잭션 로그 기반	Binlog 기반	트리거 기반	트랜잭션 로그 기반
실시간 지원	가능	가능	가능	가능	가능
구현 복잡성	낮음	중간	낮음	중간	중간~높음
Kafka 의존성	없음	있음	없음	없음	필수
주요 특징	AWS 서비스와 통합 최적화	Kafka와의 통합 최적	가벼운 CDC	분산 환경 적합	대규모 스트리밍 처리

비정형 데이터 수집, 처리 및 저장 Architecture

각 서비스 시스템에서 수집된 데이터들은 데이터의 성격과 종류에 따라 실시간 또는 배치 서비스를 이용해 수집되고 데이터 변환 과정을 거쳐서 Vector화 되어져 Vector Database에 저장됩니다. Text 기반 데이터는 Text 처리 모델을 이용한 Framework으로 Image 기반 데이터는 Image 처리 모델을 이용한 Framework을 통해 변환, 저장됩니다.
저장된 데이터는 LLM 모델의 추론 요청에 RAG로 사용되어 응답의 정확도를 높이는데 활용됩니다.



haystack by deepset LangChain LlamaIndex

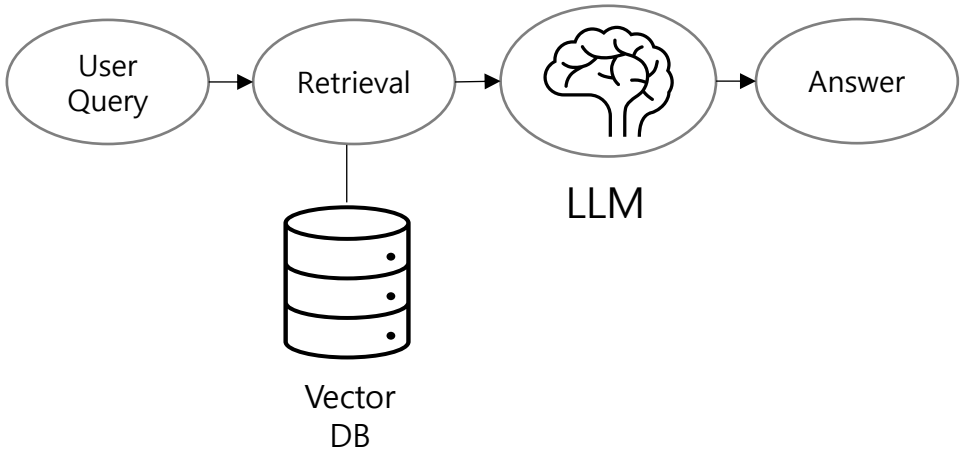
Vector DB 솔루션 비교분석

Vector DB	주요 특징	클러스터 지원	읽기 복제본	온프레미스/클라우드	멀티모달 데이터 지원	장점	단점
Neo4J	• 그래프 데이터베이스로, 관계형 데이터를 분석하는 데 강력하며 벡터 데이터 검색 기능을 통해 멀티 모달 데이터 처리가 가능. • AI/ML 워크플로우를 지원하며 텍스트 임베딩, 이미지 벡터 등과 통합 가능. • Graph Data Science (GDS) 라이브러리를 통해 머신러닝과의 통합 및 벡터 연산 제공.	Yes	Yes	온프레미스/클라우드	O	• **그래프와 벡터 통합:** 관계형 데이터와 벡터 데이터를 함께 처리하여 강력한 연결 기반 분석 가능.추천 시스템, 소셜 네트워크 분석, 지식 그래프 등에 적합. • **확장성 및 통합:** Neo4j는 다양한 클라이언트 라이브러리와 API를 제공하며, Python, Java, GraphQL 등과 쉽게 통합 가능. Neo4j의 Cypher 쿼리 언어는 직관적이고 강력한 그래프 및 벡터 쿼리 처리 가능. • **성숙한 생태계:** 오래된 커뮤니티와 풍부한 문서 및 플러그인.	**벡터 데이터에 초점 부족:** 그래프 데이터 처리가 주력이며, 벡터 데이터 검색 성능은 Milvus, Weaviate와 같은 전용 벡터 DB만큼 최적화되어 있지 않음. **복잡성 증가:** 그래프와 벡터 데이터를 함께 사용할 경우 설계가 복잡해질 수 있음. **비교적 낮은 성능:** 벡터 검색 전용으로 설계된 Milvus, Pinecone 등의 성능에는 미치지 못함. **라이센스 제한:** 오픈소스 버전에서 사용할 수 있는 기능에 제약이 있을 수 있으며, GDS 라이브러리는 상용 라이센스가 필요한 경우가 있음.
Milvus	• AI 워크로드 최적화, 다양한 인덱싱 및 확장성. • 분산형 벡터 검색 엔진으로, 고성능의 대규모 벡터 데이터 검색 및 분석을 지원. • 텍스트, 이미지, 오디오 등 다양한 멀티 모달 데이터 지원. • Kubernetes 등 프라이빗 클라우드 및 온프레미스 환경에서 쉽게 배포 가능.	Yes	Yes	온프레미스/클라우드	O	• **확장성**: 분산 아키텍처를 통해 수십억 개의 벡터를 효율적으로 처리 가능. • **멀티 모달 지원**: 다양한 데이터 형식에 대한 기본 지원 제공. • **커뮤니티 지원**: 활발한 커뮤니티와 풍부한 문서. • **플러그인 통합**: PyTorch, TensorFlow, Hugging Face 등과 통합 가능.	• 학습 곡선이 다소 가파름. • 저장소 및 검색 성능 최적화를 위해 일부 추가적인 설정이 필요.
Weaviate	• Kubernetes 기반, 유연한 스키마 및 하이브리드 검색. • 오픈소스 벡터 검색 엔진으로, 텍스트와 이미지의 통합 검색 지원. • 내장된 RESTful API와 GraphQL API 제공. • 온프레미스 및 클라우드 환경에서 모두 동작 가능.	Yes	Yes	온프레미스/클라우드	O	• 데이터 유형 지원: 텍스트와 이미지를 포함한 멀티 모달 데이터 지원. • 쉬운 API 사용: RESTful 및 GraphQL API를 통해 빠르게 통합 가능. • 확장성: 분산 시스템에서 클러스터를 구성해 대규모 데이터 처리 가능. • AI 통합: OpenAI 및 Cohere와 같은 AI 서비스와 원활한 연동.	• 일부 고급 기능은 초기 설정 및 튜닝이 필요. • 비교적 적은 커뮤니티 기반
Pinecone	• 완전 관리형, API 중심, 대규모 데이터 최적화. • 벡터 검색 및 대규모 인덱싱에 최적화된 상용 및 오픈소스 친화형 솔루션. • 클라우드 기반 솔루션으로 설계되었으나 온프레미스에서 • 구동 가능한 옵션 제공.	Yes	Yes	제한적 온프레미스/클라우드	제한적	• 운영 최적화: 벡터 인덱싱 및 검색 작업에 특화. • 스케일 아웃 지원: 클러스터 기반으로 성능 향상 가능. • 자동 관리: 복잡한 설정 없이 간단한 배포 및 운영 가능.	• 프라이빗 클라우드 배포 옵션이 제한적일 수 있음. • 오픈소스 버전에 제한된 기능 (상용 서비스에 의존하는 경향).

Agentic AI 소개

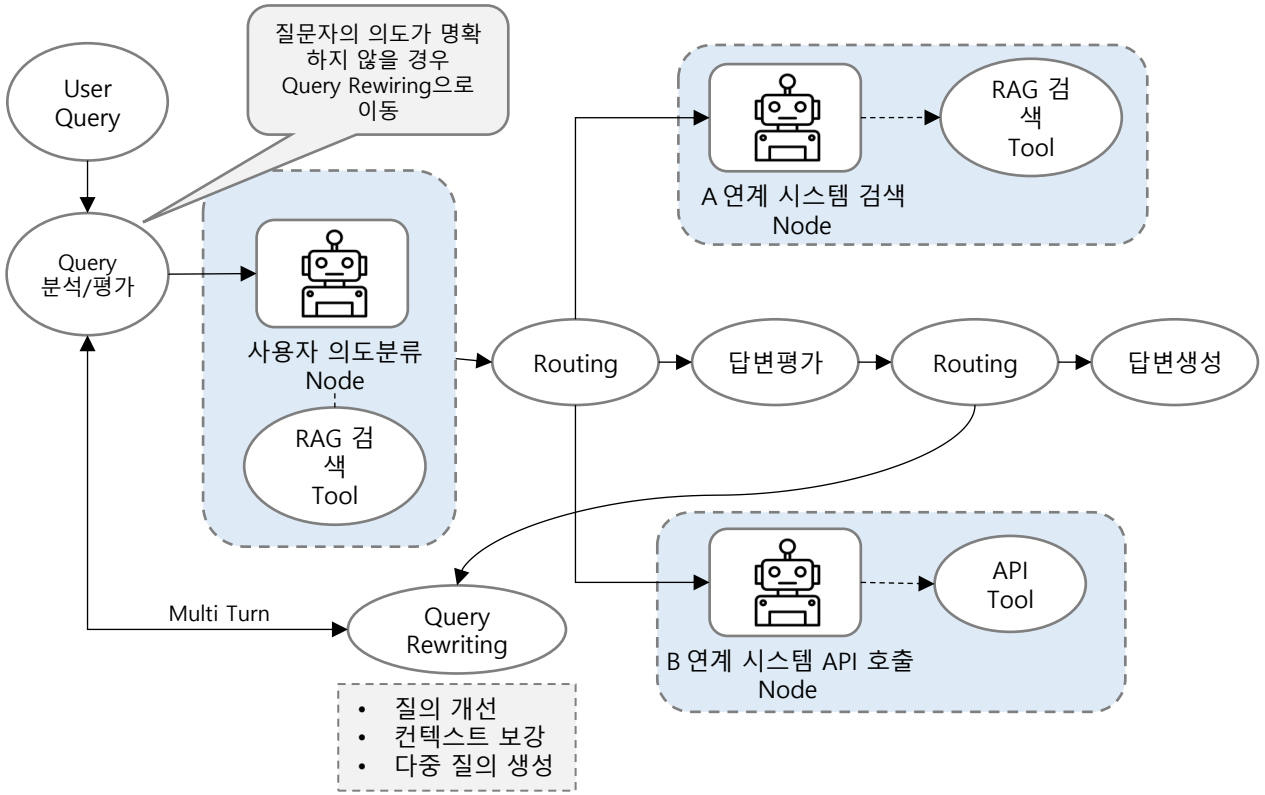
다양한 AI 모듈을 조합한 Agentic AI 방식을 도입해, AI가 스스로 목표를 설정하고 상황을 인식하며 지속적으로 의사결정을 수행함으로써, 단순 기능을 넘어 자율적으로 문제를 해결하고 사용자 요구에 유연하게 대응하는 시스템을 구현함

Modular RAG Pattern



- RAG 성능 최적화를 위한 전략적 노력
  - 질의 재해석
  - Indexing 최적화
  - Re-Raking

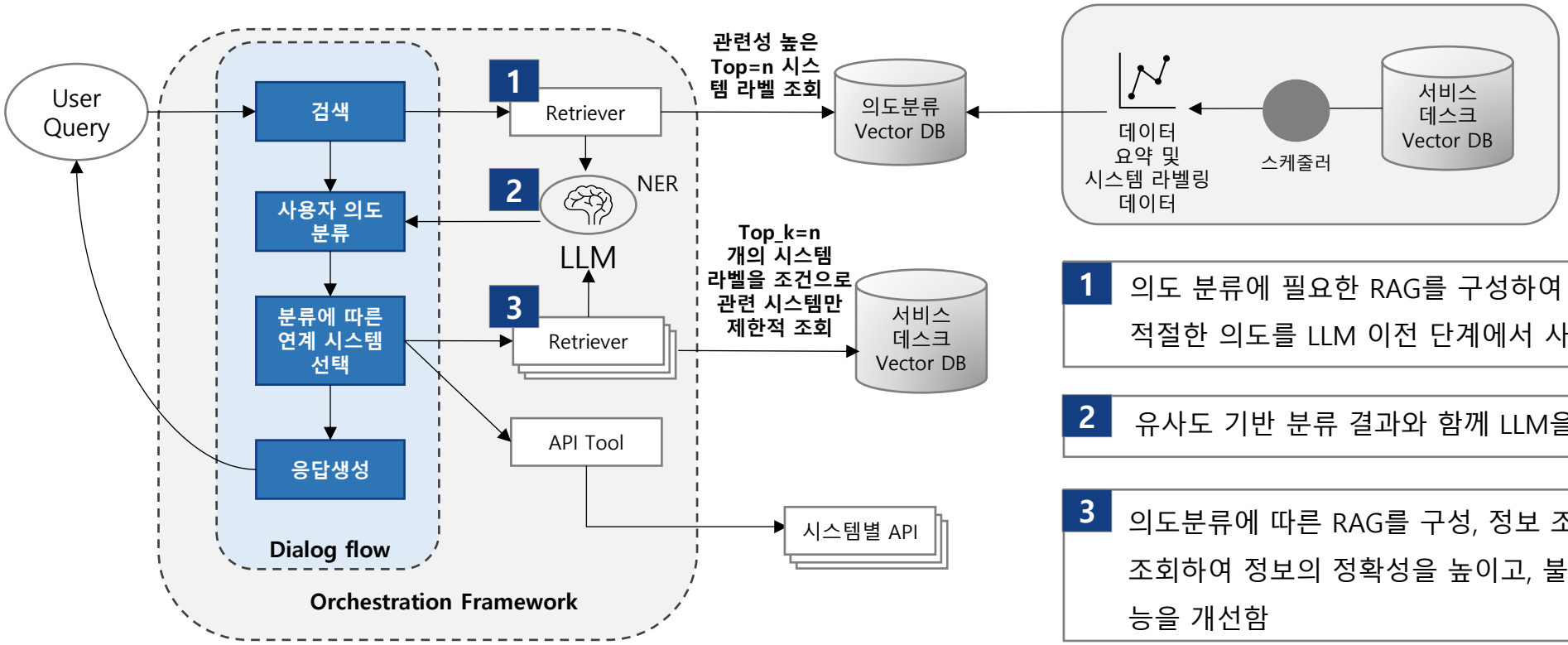
Multi-Agent RAG Flow



사용자 의도분류를 통한 정확도 개선

의도분류의 정확도를 개선하기 위해 의도분류를 위한 RAG를 구성하여, 이를 통해 사용자의 쿼리를 정확히 이해하고 적절한 응답을 제공하는 데 핵심적인 역할을 수행함

의도 분류 정확도 개선을 위한 구현 방안



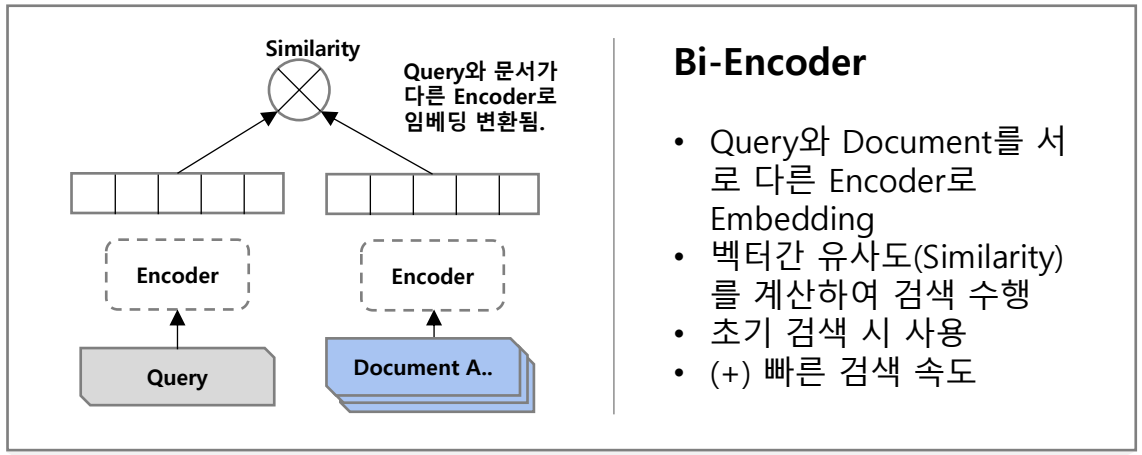
1 의도 분류에 필요한 RAG를 구성하여 사용자의 발화 의도에 맞는 적절한 의도를 LLM 이전 단계에서 사전 분류하여 정확도를 보완

2 유사도 기반 분류 결과와 함께 LLM을 통해 사용자 의도를 재해석

3 의도분류에 따른 RAG를 구성, 정보 조회 시 해당 의도 RAG에서 조회하여 정보의 정확성을 높이고, 불필요한 검색 비용을 낮춰 성능을 개선함

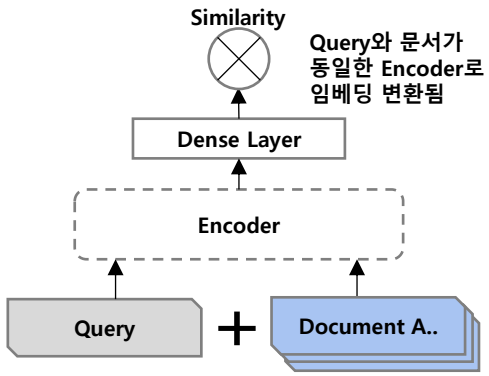
Embedding 및 Re-Ranking 튜닝

Bi-Encoder와 Cross-Encoder 기반의 Two-Stage Retrieval System을 활용한 Re-Ranking 처리 체계를 통해 AI Chatbot의 응답 시간을 고려한 대화 품질 개선을 기대할 수 있습니다.



Bi-Encoder

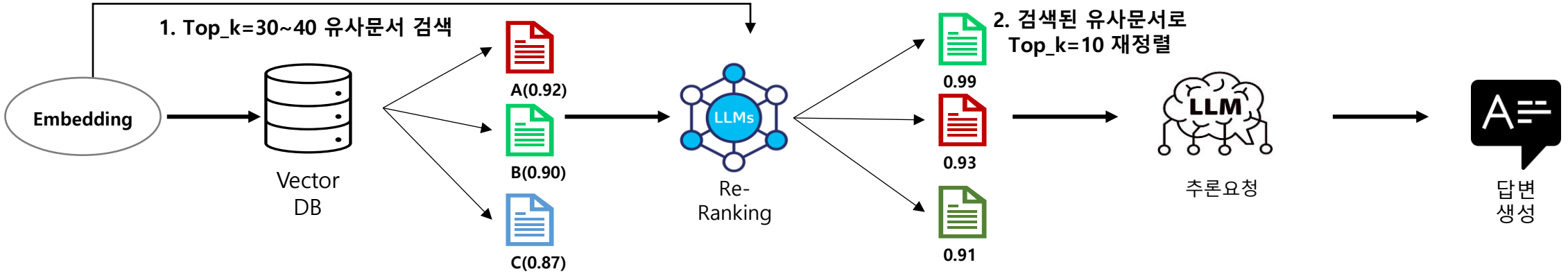
- Query와 Document를 서로 다른 Encoder로 Embedding
- 벡터간 유사도(Similarity)를 계산하여 검색 수행
- 초기 검색 시 사용
- (+) 빠른 검색 속도



Cross-Encoder

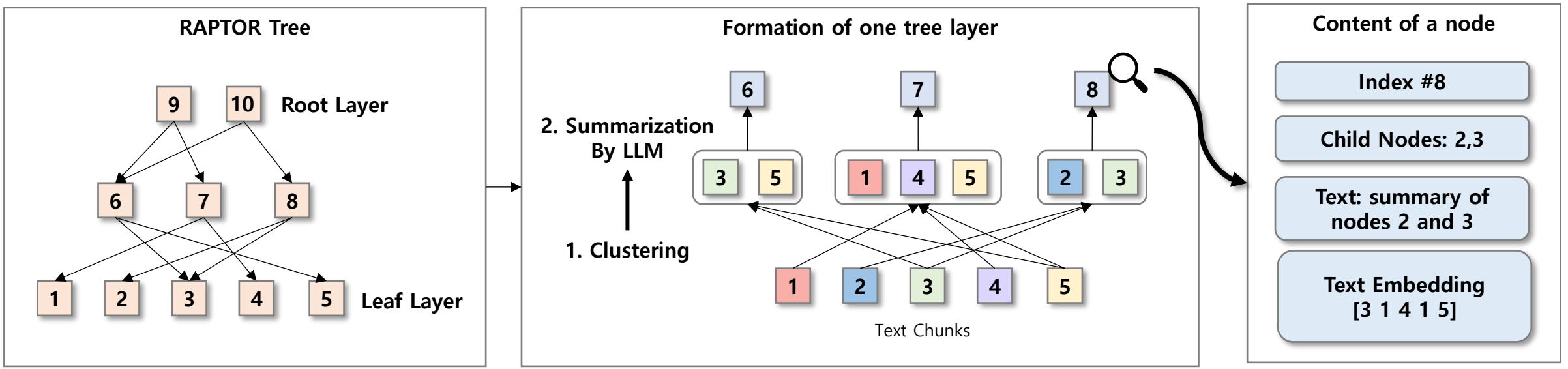
- Query와 Document를 하나의 Encoder에 함께 입력
- 문장 간의 상호작용 정보를 포함한 Embedding 생성
- Re-Ranking에 적합
- (+) 높은 정확도

Embedding & Re-Ranking 처리



RAPTOR를 이용한 성능 & 정확도 개선

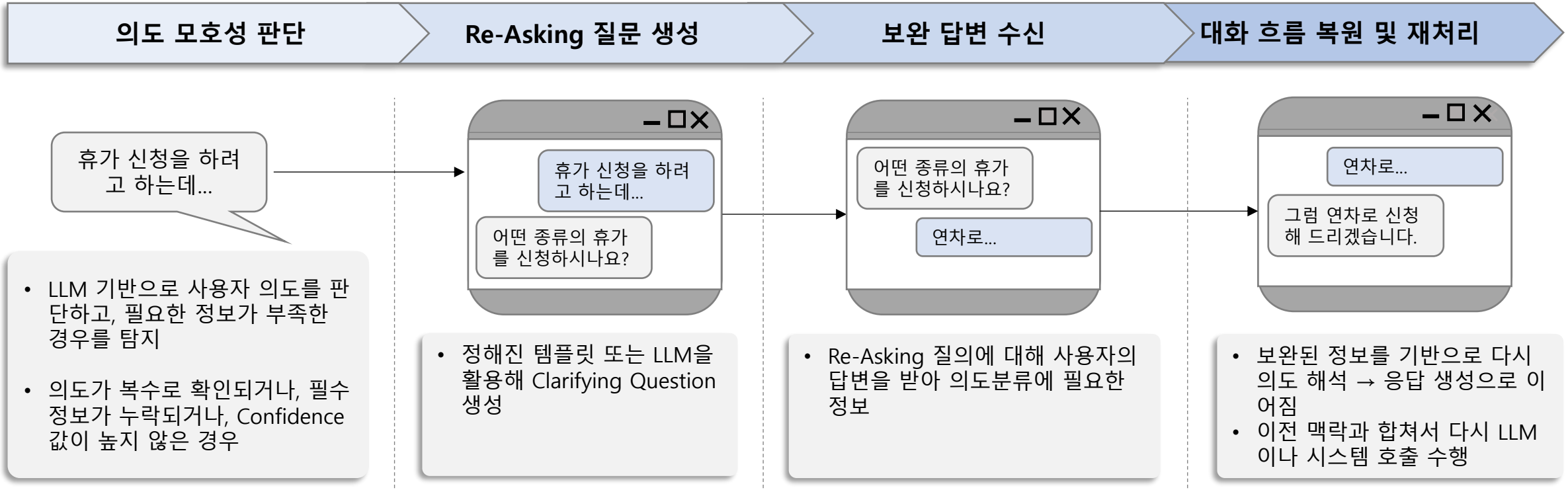
RAPTOR를 이용한 임베딩 전략 : 기본 임베딩 방식(BERT)을 이용한 RAG 시스템 구축 방식은 다음과 같은 차이점이 있음.
기존 RAG: 문서 전체를 대상으로 검색하며, 계층적 구조를 고려하지 않고 단순히 질문과 관련 있는 텍스트 조각을 찾아 응답에 활용합니다. 책의 전반적인 줄거리를 요약해야 하는 경우, Native RAG는 단편적인 retrieval만 수행하여 문서의 일부 정보만 제공하고, 전체적인 맥락이나 구조를 잘 전달하지 못할 수 있음. 모든 내용을 검색하려면 비용과 속도 이슈가 발생할 수 있습니다.
RAPTOR RAG : 문서의 계층적 구조를 활용해 단계별 요약을 생성하고, 질문에 따라 적절한 계층의 정보를 검색함. 이는 복잡하거나 긴 문서에서 더 정밀한 결과를 제공할 수 있음. 다양한 레벨의 쿼리에 대해 세부적인 정보와 전체적인 요약 모두를 제공할 수 있어 사용자 경험을 향상시킴
또한, Raptor RAG는 요약된 정보와 세부 정보를 동시에 처리하여 검색 품질과 속도를 개선할 수 있음.



Re-Asking 처리

화자의 의도가 모호하여 의도를 명확하게 하기 위해서 , Chatbot은 대화 이력을 저장하고 이전 대화와 관련된 정보를 검색해서 이전 대화의 맥락을 유지하여 재질의를 구성하고, 사용자 질문의도를 완성함

Re-Asking 처리 프로세스



Next Step 안내

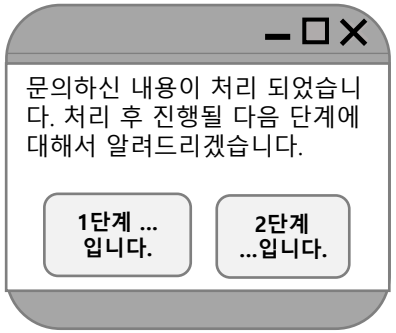
대화 품질 개선을 위해서, 지속적으로 화자의 의도에 맞춘 대화를 이어나가기 위해서, 초기 응답 후 대화 흐름을 분석하여 사용자가 다음에 무엇을 질문할 것인지 예측하고, 관련 정보를 제안하거나 버튼 식 UI를 제공해 제안을 선택할 수 있음

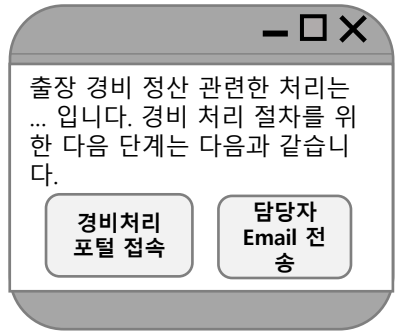
Next Step 안내 프로세스

사용자 상황 분석 → **상황 분류** → **다양한 답변 시나리오 제공**

- 사용자가 현재 어떤 단계에 있는지, 목표가 무엇인지를 파악하기 위해 대화 흐름 및 이전 질문/응답 내용을 분석.
- 사용자의 목표 달성을 위해 어떤 조치가 필요한지를 결정해서 버튼 식 UI를 이용한 제안 선택 기능 제공.

- 사용자가 업무 프로세스, 학습, 문제 해결 등 어떤 상황에서 대화를 진행하는지에 따라 분류.



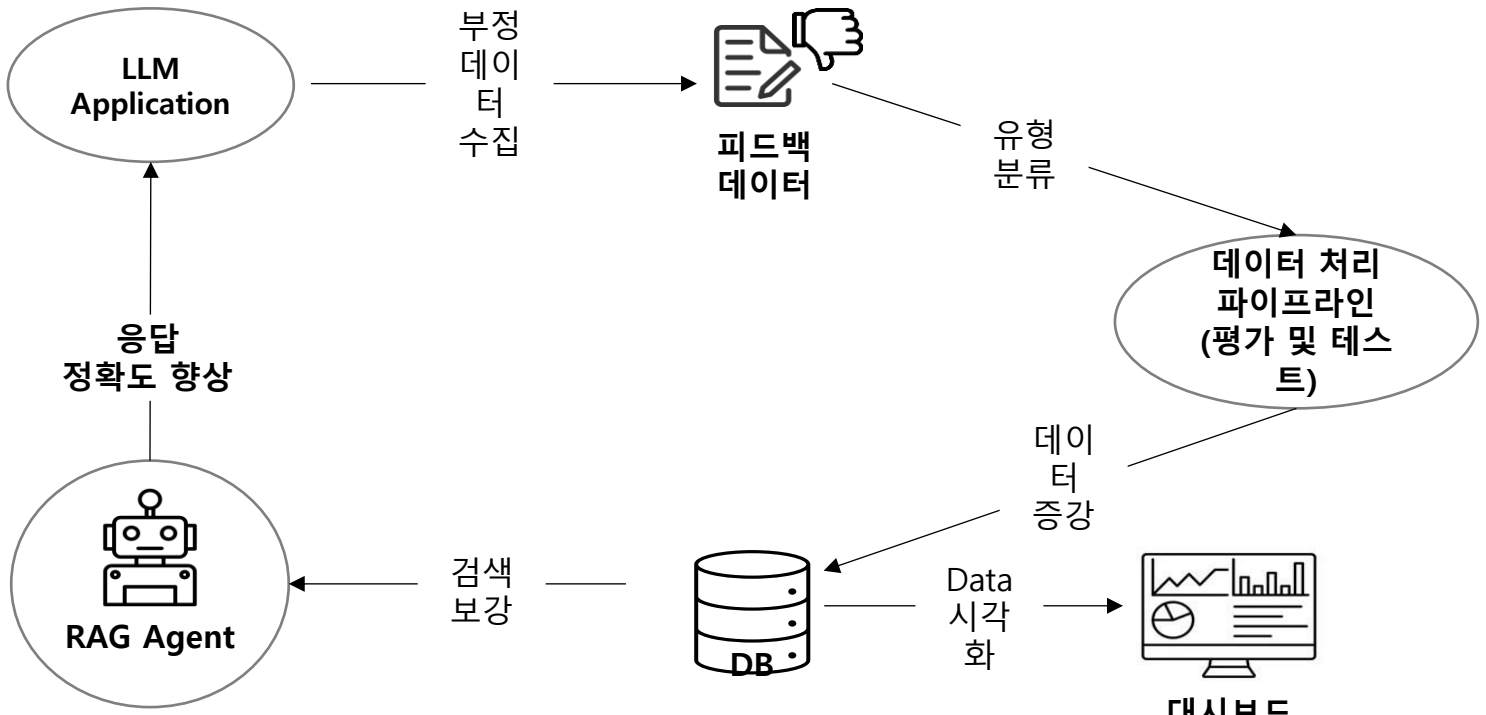


- 각 상황별로 사전 정의된 "다음 단계" 템플릿이나 가이드라인을 마련해두고, 이를 기반으로 사용자에게 맞춤형 안내를 제공.
- 현재까지의 결과를 요약하고 "이후 해야 할 작업은 A, B, C입니다."와 같이 구체적인 단계를 안내

사용자 Feedback 데이터 기반 강화학습

어플리케이션을 통해 부정 피드백 데이터를 저장하고 데이터 처리 파이프라인을 통한 데이터 유형을 분류, 평가, 테스트한 후 다시 저장함. 또한 수집된 데이터를 기반으로 RAG Agent의 응답 정확도를 향상하고 사용자 대시보드를 통한 피드백 데이터를 시각화 합니다.

사용자 Feedback Flow



Key Point

- 부정 피드백 데이터 저장 및 관리
  - 부정적 피드백 수집 및 유형 분류
- 데이터 처리 파이프라인 구축
  - 다양한 데이터 유형을 분류하는 자동화 파이프라인 구현
- 대시보드를 통한 데이터 모니터링
  - 수집된 데이터를 시각화해서 대시보드에서 조회 및 활용
- RAG Agent 응답 정확도 향상
  - 평가된 데이터 기반으로 RAG Agent의 응답 정확도 개선

Guardrail을 이용한 요청/응답 검증

LLM(Large Language Model) 애플리케이션의 안전성과 신뢰성을 높이기 위해서 Guardrails을 통한 Input, Output Guard를 활용해 LM의 입력과 출력을 정책 기반으로 검증하고 제어할 수 있습니다.

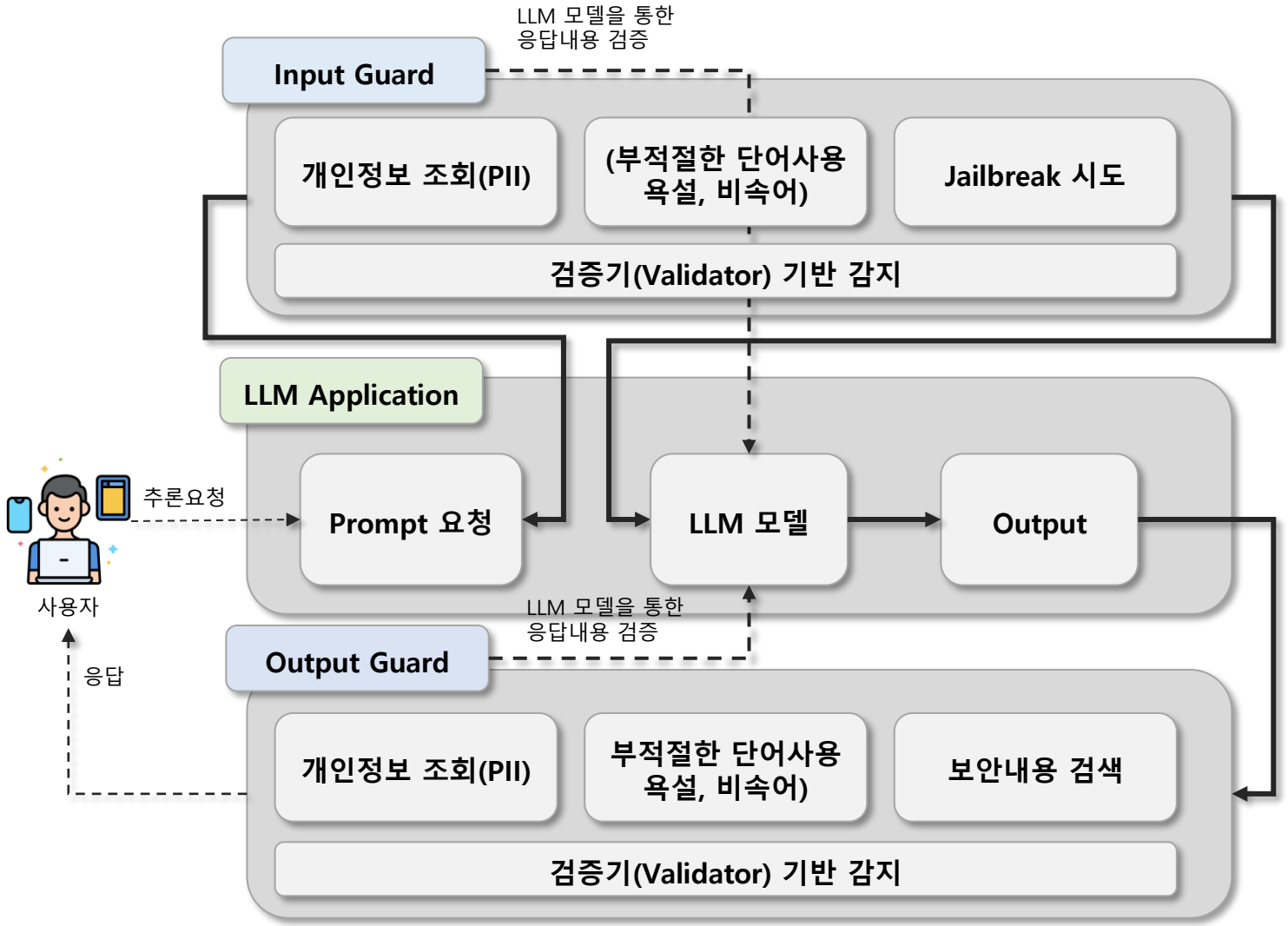
개인정보(PII) 탐지, 부적절한 단어사용, Jailbreak 시도와 같은 잘못된
요청에 대한 검증기(Validator)를 생성하고 LLM 추론 요청 및 응답 시
검증기를 기반으로 한 필터링을 가능하게 합니다.

추가적으로 처리가 필요한 정책이 생길 경우 검증기 추가를 통한
확장성을 가질 수 있습니다.

사용사례:
개인정보 보호: 금융, 의료, 법률 분야에서 개인정보(예: 주민등록번호, 전화번호)를 자동으로 감지하고 마스킹하여 데이터 유출을 방지합니다.

온라인 플랫폼 관리: 소셜 미디어, 채팅 앱, 커뮤니티에서 욕설 및 비속어를 실시간으로 필터링하여 사용자 경험을 개선합니다.

보안 위협 차단: LLM을 사용하는 서비스에서 Jailbreak 시도와 같은 보안 위협을 사전에 감지하고 차단하여 시스템 안정성을 유지합니다.



LLM 서비스 기능별 컴포넌트

사용자

- 일반 사용자
- 업무별 담당자

관리자

- Super 관리자
- LLM 관리자

사용자 Portal

Landing Page
- 로그인
- 회원가입
- 소개 및 Insights
- FAQ

공지사항
- 공지사항 목록
- 공지사항 상세

내 스페이스
- 스페이스 목록조회
- 스페이스 상세조회
- 스페이스 평가
- 스페이스 추가
- 챗봇 히스토리 조회

모델 관리
- 모델 목록조회
- 모델 상세조회
- 모델 평가

RAG Agent
- Agent 목록조회
- Agent 상세조회
- Agent 평가
- 내 Agent 등록

데이터 카탈로그
- 폴더 생성
- RAG 요청

템플릿 카탈로그
- 템플릿 관리
- 템플릿 등록

마이페이지
- 마이페이지
- Dashboard
- API Key 관리

Admin Portal

공통
- 로그인
- 회원가입

회원관리
- 계정관리
- 가입승인관리

스페이스 관리
- 스페이스 목록조회
- 스페이스 상세조회
- 챗봇 히스토리 조회

모델 관리
- 모델 목록조회
- 모델 등록
- 모델 상세조회
- 모델 배포
- 모델 설정
- APK Key 관리
- 모델평가 조회

RAG Agent 관리
- Agent 목록조회
- Agent 상세조회
- 기본 Agent 등록

템플릿 카탈로그
- 템플릿 조회
- 기본 템플릿 등록

시스템 관리
- 공통코드 관리
- 메뉴관리
- URL 관리

관리자 관리
- 권한관리
- 계정관리
- 승인대기 관리

GPU 관리
- 리소스 관리
- GPU 드라이버 관리
- GPU 스케줄러
- GPU 라이브러리 관리
- MIG 관리

데이터 카탈로그
- 폴더 조회
- RAG 설정

모니터링
- 시스템 Resource
- GPU Node
- 모델 별 Resource
- Token 모니터링

개발완료
개발 필요

LLM 서비스 전체 Architecture 구성도