

**SK네트웍스 Family AI 과정 10기  
 모델링 및 평가 시스템 아키텍처**



| **산출물 단계** | 모델링 및 평가 |
| --- | --- |
| **평가 산출물** | 시스템 아키텍처 |
| **제출 일자** | 2025.09.09 |
| **깃허브 경로** | https://github.com/skn-ai14-250409/SKN14-Final-3Team-Web |
| **작성 팀원** | 이수미, 김의령 |

1. **컴포넌트 다이어그램**

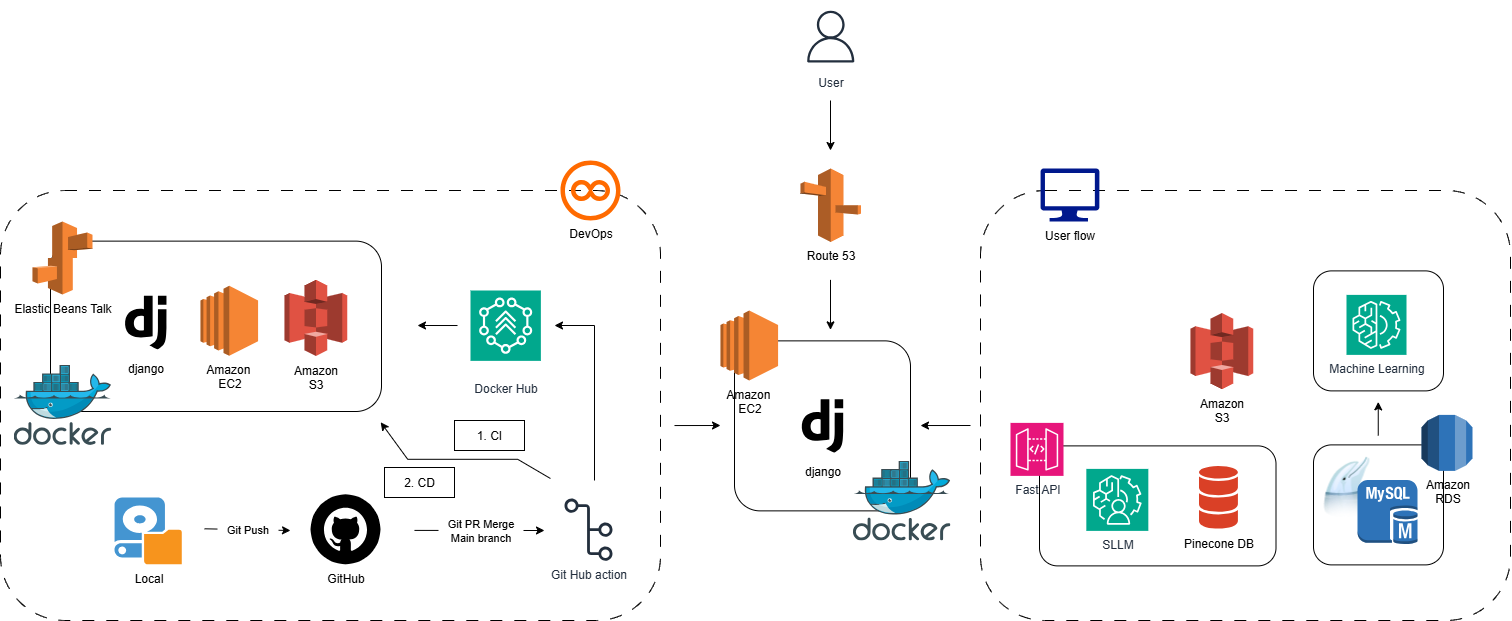


Figure1. 전체 컴포넌트 다이어그램

### 구성 요소

## 클라이언트 (Client)

## 컴포넌트

## Chrome 브라우저

## 인터페이스

## Google Cloud OAuth 2.0 (인증 처리)

## JSON 데이터 전송 (HTTPS 기반 통신)

## AWS Cloud

## EC2 인스턴스

## Nginx: 리버스 프록시 및 정적 파일 제공

## Gunicorn: WSGI 서버로 FastAPI 실행 관리

## Django + FastAPI: 백엔드 애플리케이션, 사용자 요청 처리 및 비즈니스 로직 실행

## LangGraph 모델

## Pinecone: 벡터 데이터베이스 기반 검색

## Prompt Engineering: AI 프롬프트 최적화 및 응답 품질 향상

## 데이터베이스

## Amazon DB (MySQL)

## 구조적 데이터 저장 및 조회

## 외부 서비스

## RunPod

## sLLM 실행 환경 제공 (대규모 언어모델 추론)

## 2) User Flow 동작 설명

## 

Figure2. User Flow 컴포넌트 다이어그램

## 본 시스템의 사용자 요청 처리 흐름은 다음과 같다:

## 사용자 인증

## 사용자는 Google OAuth 2.0을 통해 인증 후 HTTPS 요청을 전송한다.

## 요청 전달

## AWS Route 53이 사용자의 요청을 받아 Amazon EC2 인스턴스로 전달한다.

## 애플리케이션 서버 처리

## EC2 내부에서 Django 서버(Nginx + Gunicorn)가 요청을 수신한다.

## Django는 내부적으로 FastAPI 모듈을 호출하여 후속 처리를 진행한다.

## 데이터 및 AI 모듈 연동

## FastAPI는 Chroma DB를 조회하고, sLLM / Machine Learning 모듈을 활용해 질의응답 및 분석을 수행한다.

## 필요한 경우 Amazon DB(MySQL)와 Amazon S3를 통해 데이터를 조회하거나 저장한다.

## 응답 반환

## 최종 결과를 JSON 형태로 변환하여 사용자에게 반환한다.

## 3) DevOps Flow 동작 설명

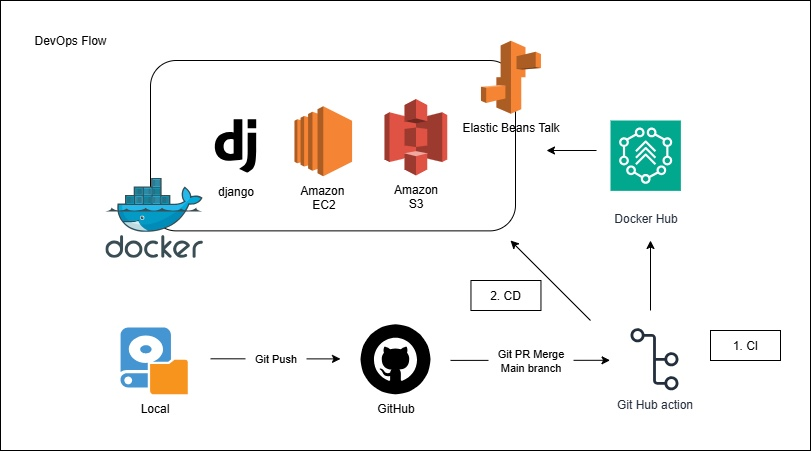
* 

Figure3. DevOps Flow 컴포넌트 다이어그램

## 본 프로젝트의 CI/CD 파이프라인은 다음과 같은 절차를 따른다.

## Local

## 개발자가 로컬 환경에서 코드를 작성한 뒤 GitHub 저장소에 Push한다.

## GitHub

## Pull Request(PR) → Main Branch Merge 과정을 통해 배포 준비가 완료된다.

## CI (Continuous Integration)

## GitHub Actions가 자동 실행되어 다음 절차가 수행된다.

## 코드 빌드 및 유닛 테스트 실행

## Docker 이미지 생성 및 Docker Hub 업로드

## CD (Continuous Deployment)

## Elastic Beanstalk이 Docker Hub에서 최신 이미지를 가져와 자동 배포한다.

## 배포된 컨테이너는 Amazon EC2에서 실행되며, 정적/비정형 데이터는 Amazon S3와 연동된다.

## 자동화 동작 요약

## 개발자가 GitHub에 코드 Push

## GitHub Actions(CI) 실행 → 빌드 & 테스트 → Docker 이미지 생성

## Docker Hub에 이미지 업로드

## Elastic Beanstalk이 이미지를 가져와 EC2/S3에 자동 배포

## Django 서비스가 새로운 버전으로 업데이트

## 4) 통합 특징

### 보안

## Google OAuth 2.0 기반 사용자 인증

## AWS 보안 계층(VPC, IAM, 보안 그룹)을 통한 안정적 운영

### 확장성

## FastAPI, sLLM, ML 모듈을 컨테이너 기반으로 독립 실행

## 모듈 단위 확장 및 교체 용이

### 데이터 관리

## Amazon DB(MySQL): 금융 데이터 및 메타데이터와 같은 구조적 데이터 관리

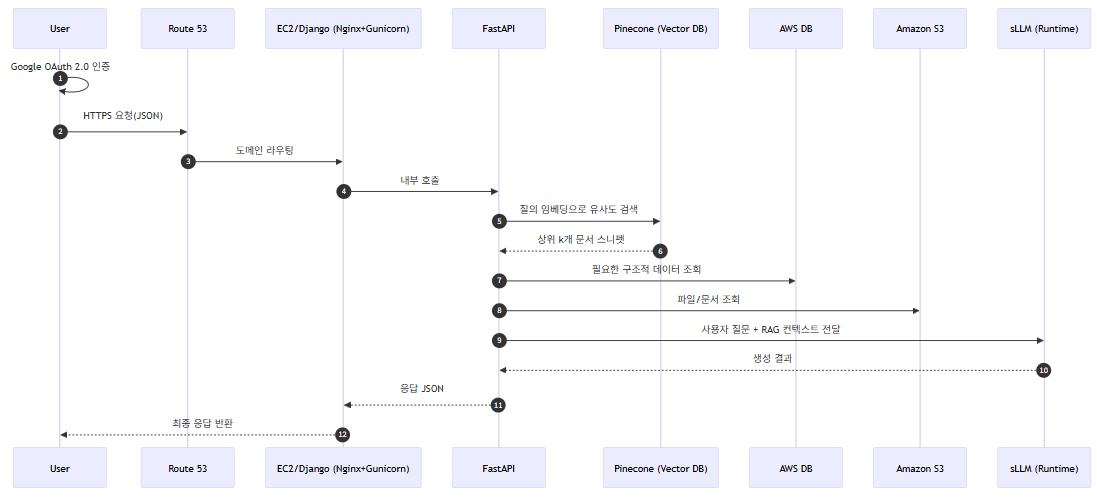
## Amazon S3: 문서, 모델, 로그 등 비정형 데이터 저장

## Pinecone DB: 벡터 기반 검색 및 RAG(Retrieval-Augmented Generation) 지원

### 5) 자동화

## GitHub Actions → Docker Hub → Elastic Beanstalk으로 이어지는 완전 자동화 CI/CD 파이프라인 구축

**2. 시퀀스 다이어그램**



## 참여자 (Actors)

* **User**: 클라이언트(Chrome 브라우저, Docs, Chrome Extensions)를 통해 요청을 시작하는 사용자
* **Google Cloud OAuth 2.0**: 사용자 인증 및 권한 부여 처리
* **Route 53**: 사용자 요청을 AWS 서버(EC2)로 라우팅
* **EC2/Django (Nginx + Gunicorn)**
  + **Nginx**: 요청 수신 및 리버스 프록시 역할
  + **Gunicorn**: FastAPI 애플리케이션 실행
  + **Django**: 엔드포인트 제어 및 FastAPI 호출 관리
* **FastAPI**: 요청 처리 및 비즈니스 로직 실행
* **Pinecone (Vector DB)**: 쿼리 임베딩 기반 유사도 검색 수행
* **AWS DB**: 구조적 데이터 조회 및 저장 처리
* **Amazon S3**: 비정형 파일 및 문서 저장소 조회
* **sLLM (Runtime, RunPod 환경)**: RAG 컨텍스트와 사용자 질문을 받아 답변 생성

## 주요 흐름

1. **사용자 요청 시작**
   * 사용자가 클라이언트(브라우저/확장 프로그램)를 통해 요청을 전송
   * 요청 전 Google Cloud OAuth 2.0에서 인증 및 토큰 획득
2. **라우팅**
   * Route 53이 HTTPS 요청(JSON)을 받아 AWS EC2 인스턴스로 전달
3. **애플리케이션 서버 처리**
   * EC2에서 Nginx가 요청을 수신하고 Gunicorn을 통해 Django + FastAPI로 전달
   * Django는 FastAPI 모듈을 호출하여 로직 실행 시작
4. **데이터 검색**
   * FastAPI가 Pinecone(Vector DB)을 호출하여 유사도 검색 수행 후 상위 k개 문서 스니펫 획득
   * 필요 시 AWS DB에서 구조적 데이터 조회
   * Amazon S3에서 비정형 문서 및 파일 조회
5. **AI 생성 단계**
   * FastAPI가 수집된 컨텍스트와 사용자 질문을 sLLM(Runtime)에 전달
   * sLLM이 답변을 생성하여 FastAPI로 반환
6. **응답 반환**
   * FastAPI가 결과를 JSON으로 Django에 반환
   * Django → EC2 → 사용자 브라우저로 최종 응답 전송
   * 사용자는 브라우저에서 결과 확인

**3. 액티비티 다이어그램**



## 구성

1. **시작 지점**
   * 사용자가 클라이언트에서 작업을 시작합니다.
2. **클라이언트 활동**
   * Chrome 브라우저, Google Docs, 또는 Chrome 확장 프로그램을 통해 요청이 생성됩니다.
   * Google Cloud OAuth 2.0을 통해 사용자 인증이 수행됩니다.
3. **데이터 전송**
   * Route 53(DuckDNS 역할 포함)을 통해 클라이언트 요청이 AWS 클라우드로 전달됩니다.
   * 요청은 JSON 형식으로 HTTPS를 통해 안전하게 전송됩니다.
4. **AWS 클라우드 내부 활동**
   * **Nginx**가 요청을 수신하고 리버스 프록시 역할을 수행합니다.
   * **Gunicorn**이 요청을 FastAPI 애플리케이션으로 전달하여 처리합니다.
   * **Django**는 엔드포인트를 제어하고 FastAPI와의 연동을 담당합니다.
5. **LangGraph 모델 호출**
   * FastAPI는 LangGraph 모델을 호출하여 외부 데이터 소스(PyKRX, Open DART, Yahoo Finance 등)와 연동하거나 AI 프롬프트 최적화(Chroma, Prompt Engineering)를 수행합니다.
6. **데이터베이스 작업**
   * Amazon DB(MySQL)을 통해 구조적 데이터를 조회하거나 저장합니다.
   * **Amazon S3**에서 비정형 파일이나 문서를 불러올 수 있습니다.
7. **외부 서비스 호출**
   * RunPod(sLLM) 또는 ChatGPT와 같은 AI 서비스를 호출하여 생성형 응답을 획득합니다.
8. **결과 반환**
   * FastAPI는 응답을 JSON 형태로 변환하여 Django를 통해 클라이언트로 반환합니다.
   * 클라이언트는 최종 결과를 확인합니다.
9. **종료 지점**
   * 작업이 완료됩니다.

## 주요 액션 노드

* **클라이언트 활동**: 사용자가 Chrome에서 요청을 시작하고 인증을 수행합니다.
* **데이터 전송**: Route 53(DuckDNS)과 HTTPS를 통해 요청이 안전하게 AWS로 전달됩니다.
* **AWS 클라우드 활동**: 서버에서 요청을 처리하며 Nginx, Gunicorn, Django, FastAPI가 함께 동작합니다.
* **LangGraph/데이터 연동**: 외부 데이터 소스 호출, AWS DB(MySQL) 및 Amazon S3를 통한 데이터 처리 수행.
* **외부 서비스 호출**: RunPod(sLLM)을 통해 응답을 생성합니다.
* **응답 반환**: 최종 결과가 JSON 형태로 클라이언트에 반환되어 사용자가 확인합니다.