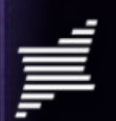
 Fast campus × *upstage* 

Upstage AI Lab

Machine Learning Project | 2025. 06.10(화)

목차

01. 팀 소개

02. 프로젝트 개요

03. 프로젝트 수행 절차 및 방법

04. 회고

01

팀 소개

팀장/팀원 소개
협업 방식

* ML_2조 [GPTeam5]



팀장

송규현

RAG / 경영정보

ML Engineer



팀원

김효석

비트코인 / 재료공학

ML Engineer



팀원

박진섭

RAG, LLM / 해킹보안

ML Engineer



팀원

안재윤

TDA/수학

ML Engineer



팀원

이진식

음성인식, LLM / 비전공

ML Engineer

프로젝트 협업 방식

: MLOps Project

🎯경진대회 이후 우리 팀의 협업 방식 변화

- 협업 방식 전환
 - 기존 단순 분업 → **MLOps** 중심 체계적 협업으로 전환
 - Git + Notion을 활용한 실시간 피드백 & 문서화
- 역할 분담 & 실시간 공유
 - 모델링 / 데이터 파이프라인 / API / 배포 역할 분담
 - 어려운 부분은 즉시 공유 → 병목 최소화

📅 July 17 협업 일정 & 문제 해결 경험

- 협업 일정
 - 매일 1시간 이상의 정기미팅
 - Notion으로 회의록 관리, 이슈 정리 및 우선순위 관리
 - Pull Request 기반 코드 리뷰 및 병합
- 문제 발생 & 해결
 - 문제점 1) MLOps 프레임워크에 대한 이해도 부족 -> 멘토 자문 활용
 - 문제점 2) Docker 환경 불일치 -> 공통 Dockerfile 사용으로 베이스 이미지 통합
- 변화된 점
 - 운영 가능성 중심 사고로 전환
 - 실무에 가까운 MLOps 경험
 - 팀원 간 기술 공유 문화 자리잡음



강사_이인우 오후 6:21

Q. 에어플로우에서 함수별로 태스크로 나누어 파이썬 오퍼레이터로 워크플로우 처리를 할 수 있는데, 파이어 라이브러리를 사용하여 배쉬 오퍼레이터로 워크플로우 처리를 하는 방법이랑 어떤 점이 다른지 알고 싶습니다.

A. 우선 Airflow의 DAG에서 비즈니스 로직을 작성하거나 ml 코드를 import해서 사용하는 방식은 여러가지 문제를 발생시킵니다.

- 의존성 충돌 문제 : Airflow DAG 내에서 코드를 직접 수행시키면 해당 코드를 수행하기 위한 의존성들이 Airflow에 설치되어야 합니다. DAG가 많아질 수록 해당 의존성들끼리 버전 문제나 충돌 문제가 발생할 확률이 높아집니다.
- 소스코드 누적 문제 : DAG 내에서 직접 ML 코드를 수행하기 위해서는 소스코드 패키지를 Airflow 각 컴포넌트(스케줄러, 워커 등)에서 가지고 있어야 합니다. 이는 ML DAG가 늘어날 수록 유지해야 하는 소스코드 양이 지속적으로 누적되고 이는 용량과 성능 문제를 일으킬 수 있습니다.

따라서 해당 소스코드 패키지는 Docker Image로 빌드하고 해당 이미지를 DockerOperator를 통해 컨테이너로 실행시킨 뒤 Taks 실행 명령을 전달하는 방식으로 구현하면 DAG 별로 독립된 태스크 실행 환경을 구현할 수 있습니다.(권장)

PythonOperator와 BashOperator의 차이는 말씀해주신 상황에서는 동일한 동작으로 동작합니다. 다만 BashOperator를 사용하면 필요 시 Bash 쉘의 다양한 기능들을 활용할 수 있습니다.



02

프로젝트 개요

목표 수립
기술 스택 / 아키텍처 설계

프로젝트 목표 수립

: MLOps Project | 목표 및 주요 작업

주제

MLOps 프로젝트 | 아파트 가격 예측 MLOps 프로젝트

기간

2025. 05. 26 ~ 2025. 06.10

목표

목적

주요내용

주요작업

- 실제 서울시 아파트 거래 데이터를 활용한 가격 예측 진행
- 데이터 수집부터 모델링, 평가, 배포까지 전체 MLOps 파이프라인 구축 및 자동화
- 학습 및 실무 활용이 가능한 구조의 설계 및 확장성 확보

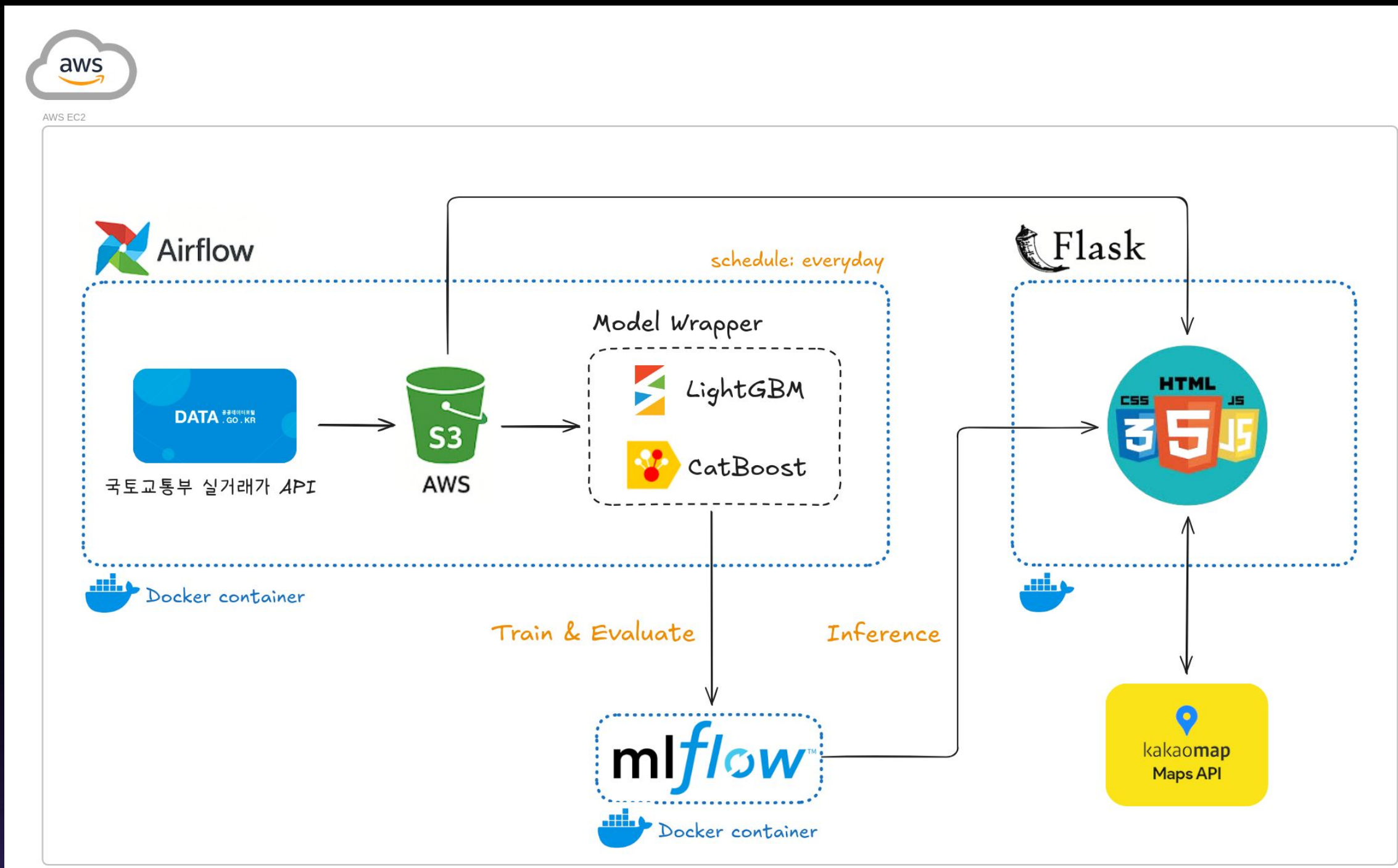
- 실제 서울시 아파트 거래 데이터를 이용한 가격 예측
- End-to-End MLOps 파이프라인 구축 및 운영

- 데이터 수집 → 전처리 → 학습 → 예측 → 웹 서비스 제공
- 자동화 및 운영 효율화 (Airflow, Docker, AWS S3 등)
- 재현성 있는 구조 및 코드 제공

- 국토교통부 API를 활용하여 서울시 아파트 실거래 데이터 수집
- 수집된 데이터의 정제 및 주소, 위치 정보 통합
- CatBoost, LGBM모델을 통해 학습수행 및 Wrapper 설계
- Hyperopt를 활용한 모델 파라미터 튜닝
- RMSE 기반의 Cross Validation 수행
- 모델 로드 및 추론 기능 제공
- Flask를 통해 예측 결과 사용자에게 제공
- Airflow DAG구성으로 전체 프로세스 자동화
- Docker, .env, requirements를 통한 환경설정 및 자동화

프로젝트 개요

: MLOps Project | 기술 스택 및 아키텍처 설계



03

프로젝트 수행 절차 및 방법

데이터셋 및 데이터 처리 / 모델 개발 /
모델 배포 / MLOps 워크 플로우 /
모니터링

프로젝트 수행 절차 및 방법

: MLOps Project | 데이터셋

국토교통부_아파트 매매 실거래가 상세 자료 1.0.0

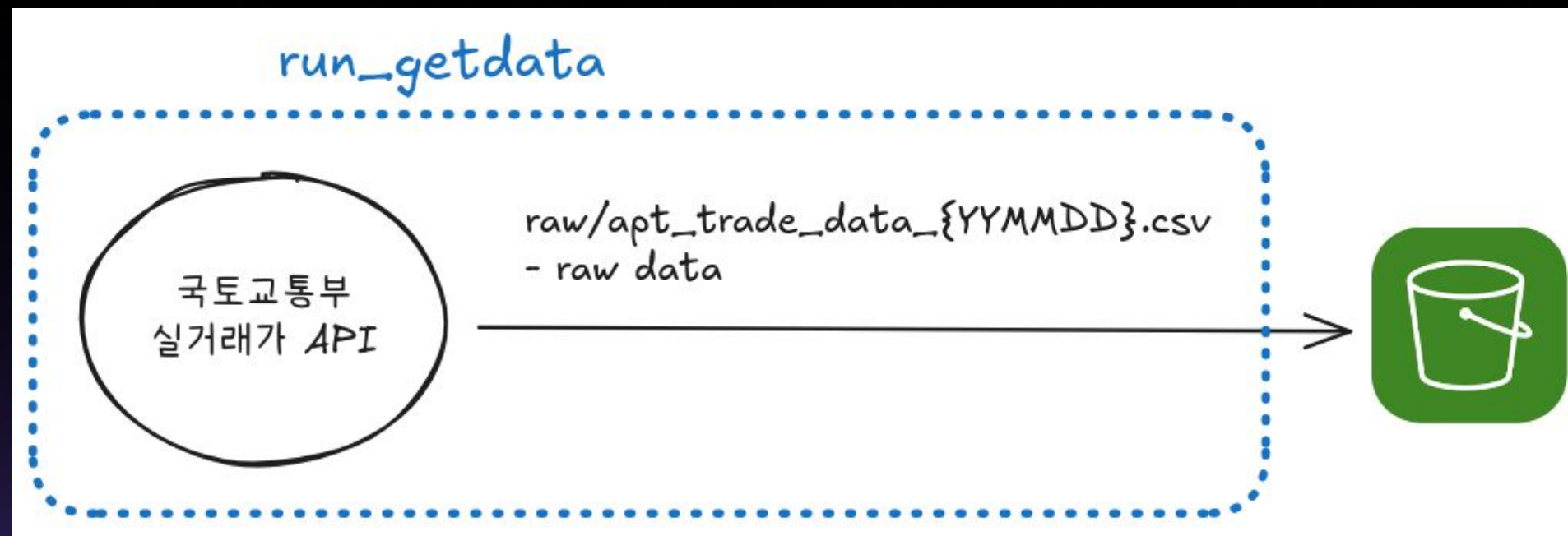
[Base URL: apis.data.go.kr/1613000/RTMSDataSvcAptTradeDev]

행정표준코드관리시스템(www.code.go.kr)의 법정동 코드 중 앞5자리(예시: 서울 종로구 - 11110), 계약년월(예시: 201801)로 해당 지역, 해당 기간의 아파트 매매 신고 상세정보를 조회

API 목록

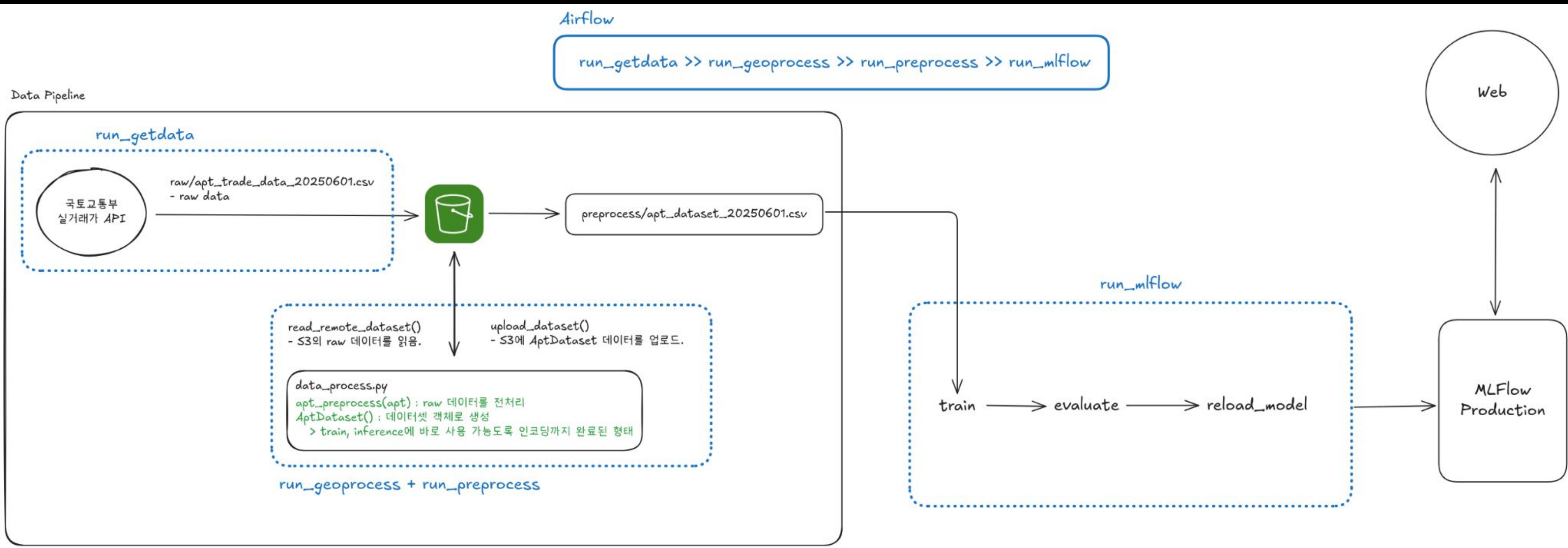
GET

/getRTMSDataSvcAptTradeDev 아파트 매매 실거래가 공개 자료(상세)



프로젝트 수행 절차 및 방법

: MLOps Project | 데이터 전처리



프로젝트 수행 절차 및 방법

: MLOps Project | 모델 개발

* 모델 선택 및 이유 (사용한 모델 ex. BERT, LSTM 등)

사용한 모델

모델 선택 이유



CatBoost



LightGBM

[LightGBM]

빠른 학습 속도와 낮은 메모리 사용량
→ 대용량 데이터 처리에 효율적

Leaf-wise 트리 성장 방식
→ 정확도(성능) 향상 가능

[CatBoost]

과적합 방지를 위한 Ordered Boosting
→ 안정적인 성능

적은 튜닝으로도 좋은 성능
→ 기본 설정만으로도 강력함

프로젝트 수행 절차 및 방법

: MLOps Project | 모델 배포

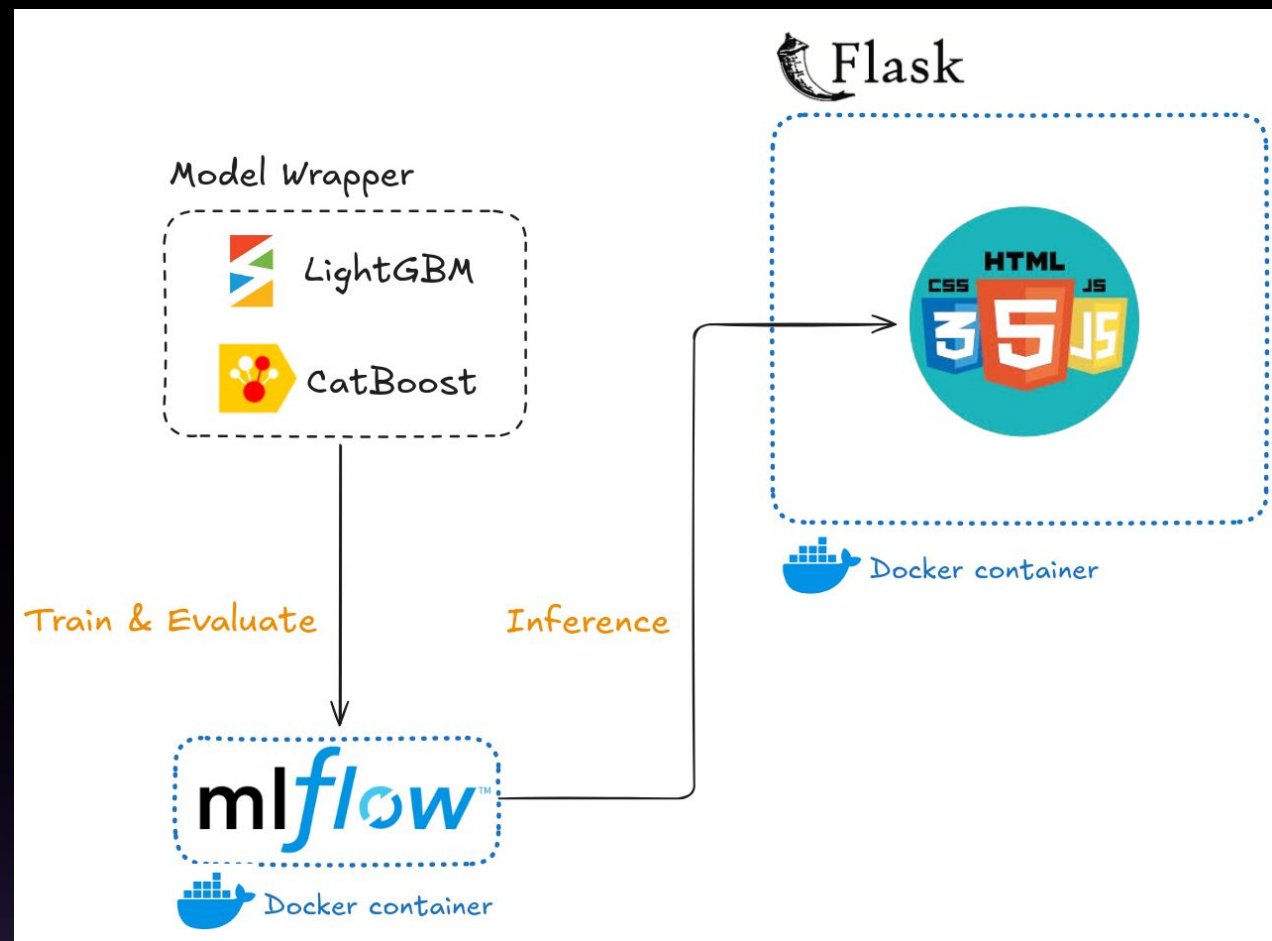
* 배포 방법 및 모델 서빙 공유

배포 방법

배포 과정 :
Airflow 에서 매일 학습한 모델을 mlflow에 reload 합니다.
업데이트된 모델을 Flask 웹앱에서 로드하여 추론에 사용합니다.

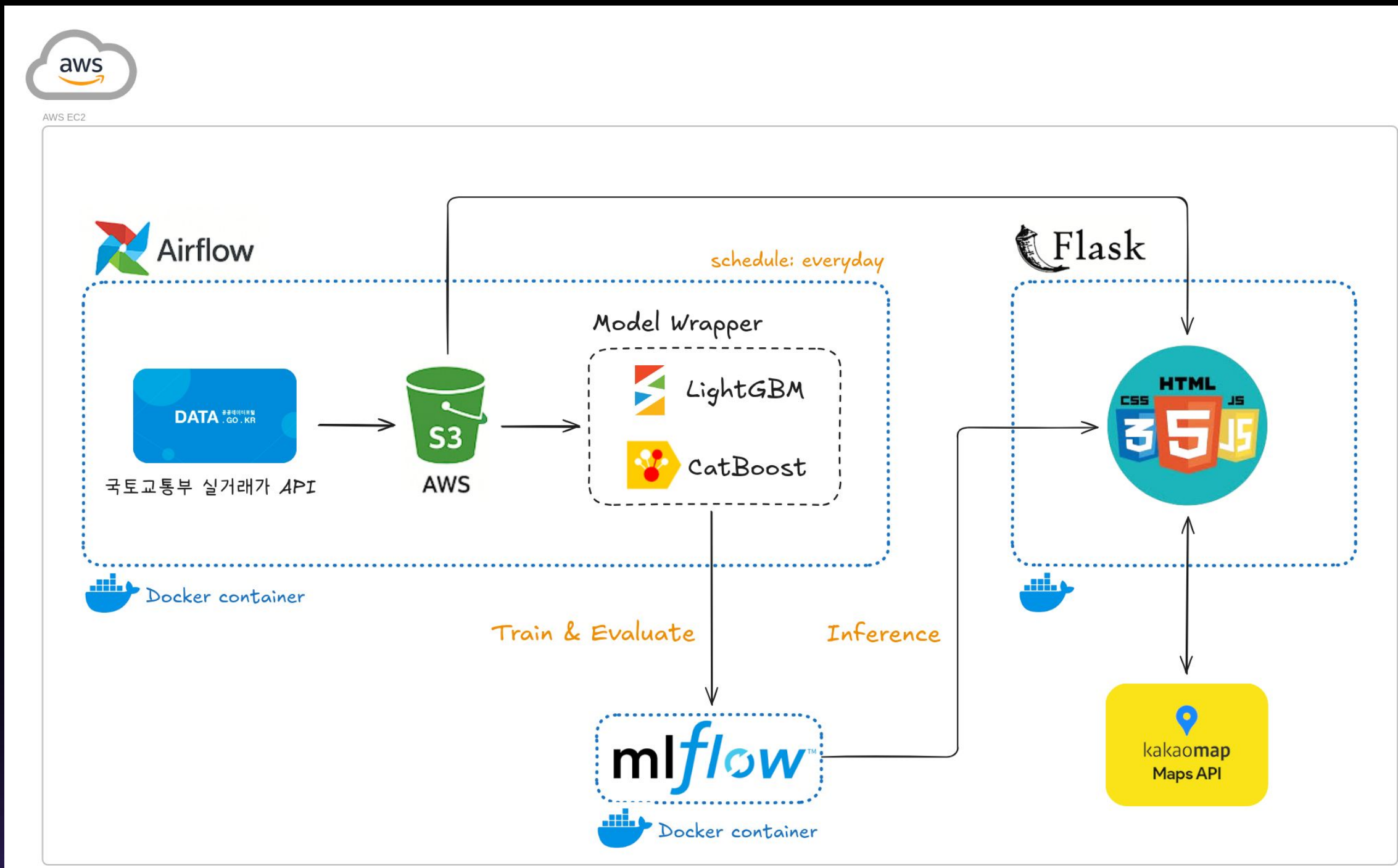
모델 서빙

- 모델 업데이트: Airflow에서 주기적으로 모델을 학습시킵니다.
- 사용자 요청 예측: 사용자가 요청한 아파트에 대해 3개월치 실거래가 예측을 수행합니다.



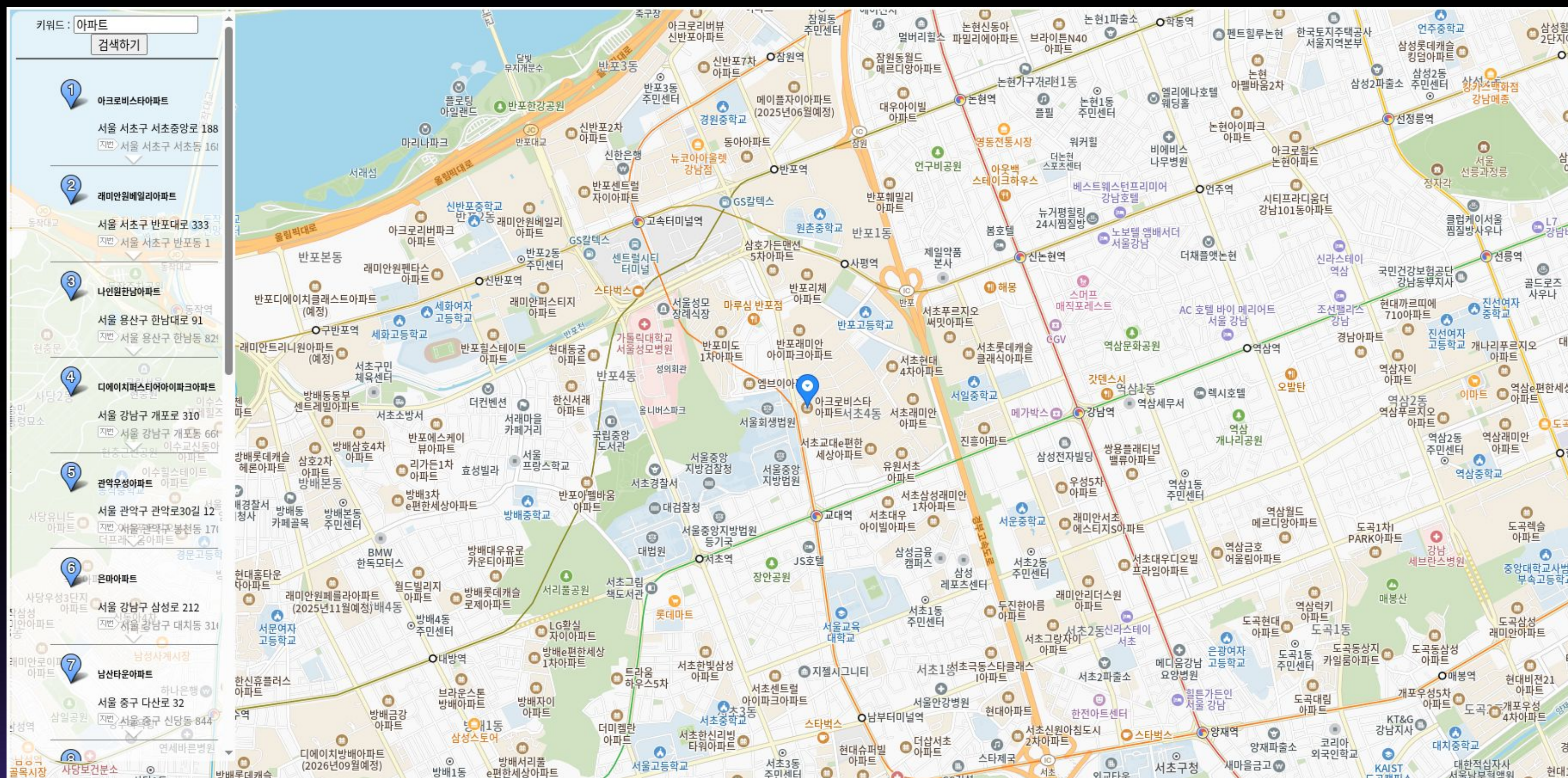
프로젝트 수행 절차 및 방법

: MLOps Project | MLOps 워크 플로우



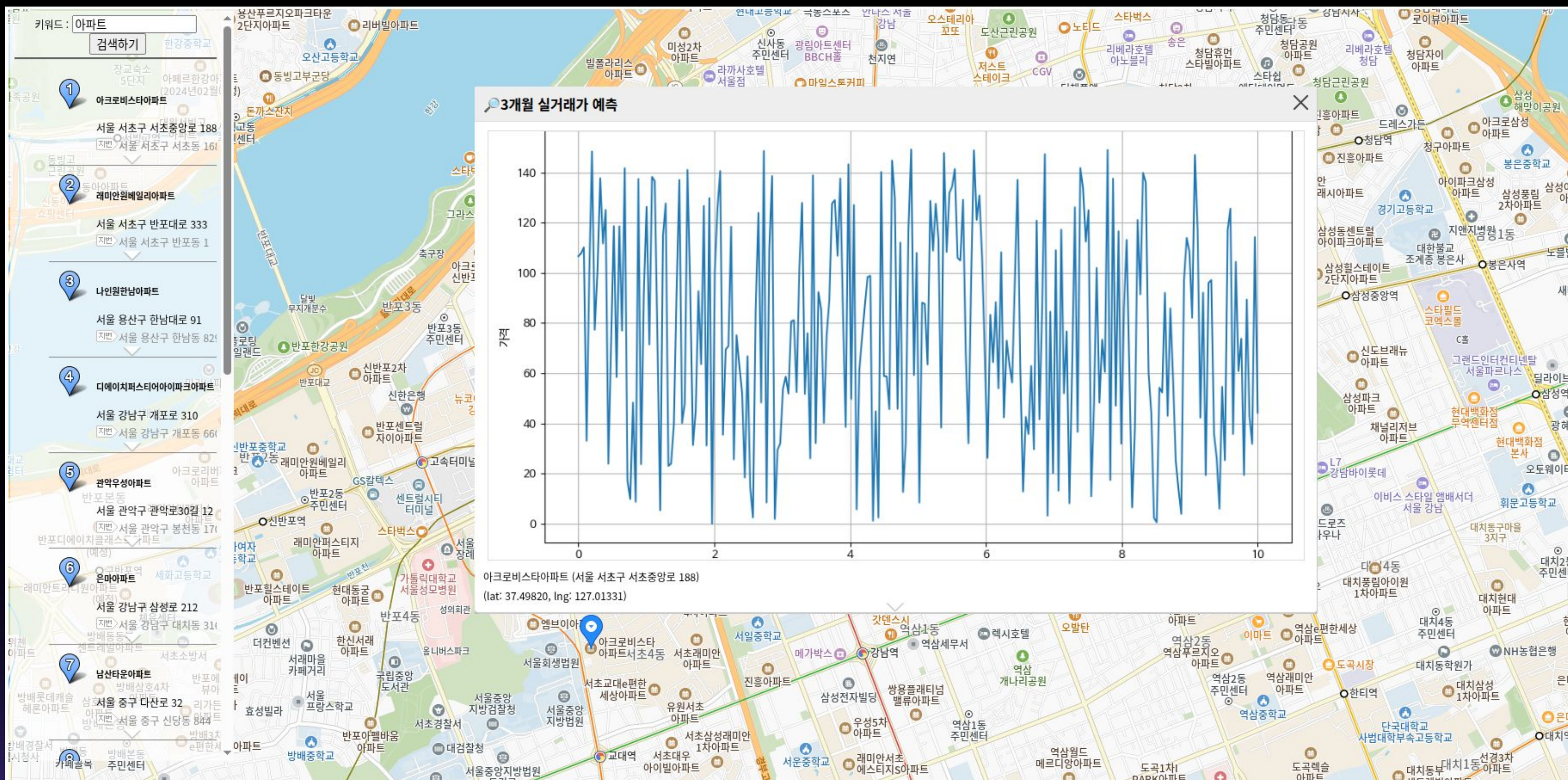
프로젝트 수행 절차 및 방법

: MLOps Project | Web app



프로젝트 수행 절차 및 방법

: MLOps Project | Web app



04

회고

결과 / 인사이트 도출 / 향후 계획
느낀점

프로젝트 회고

: MLOps Project | 결과 및 향후 계획

1

* 최종 결과



- mlflow는 미완성
- Flask webapp 개발 완료
- Airflow 데이터 파이프라인 개발 완료

2

* 도전 과제 및 해결



- Docker에 대한 이해도 부족으로 개발환경 통합에 많은 시간 소요
- Airflow DockerOperator에 많은 시행착오 경험

3

* 인사이트 도출



- Docker Contianer 활용 시 개발환경 통합, 서비스 자원 관리, 유지보수 측면에서 효과적임을 체감할 수 있었습니다.

4

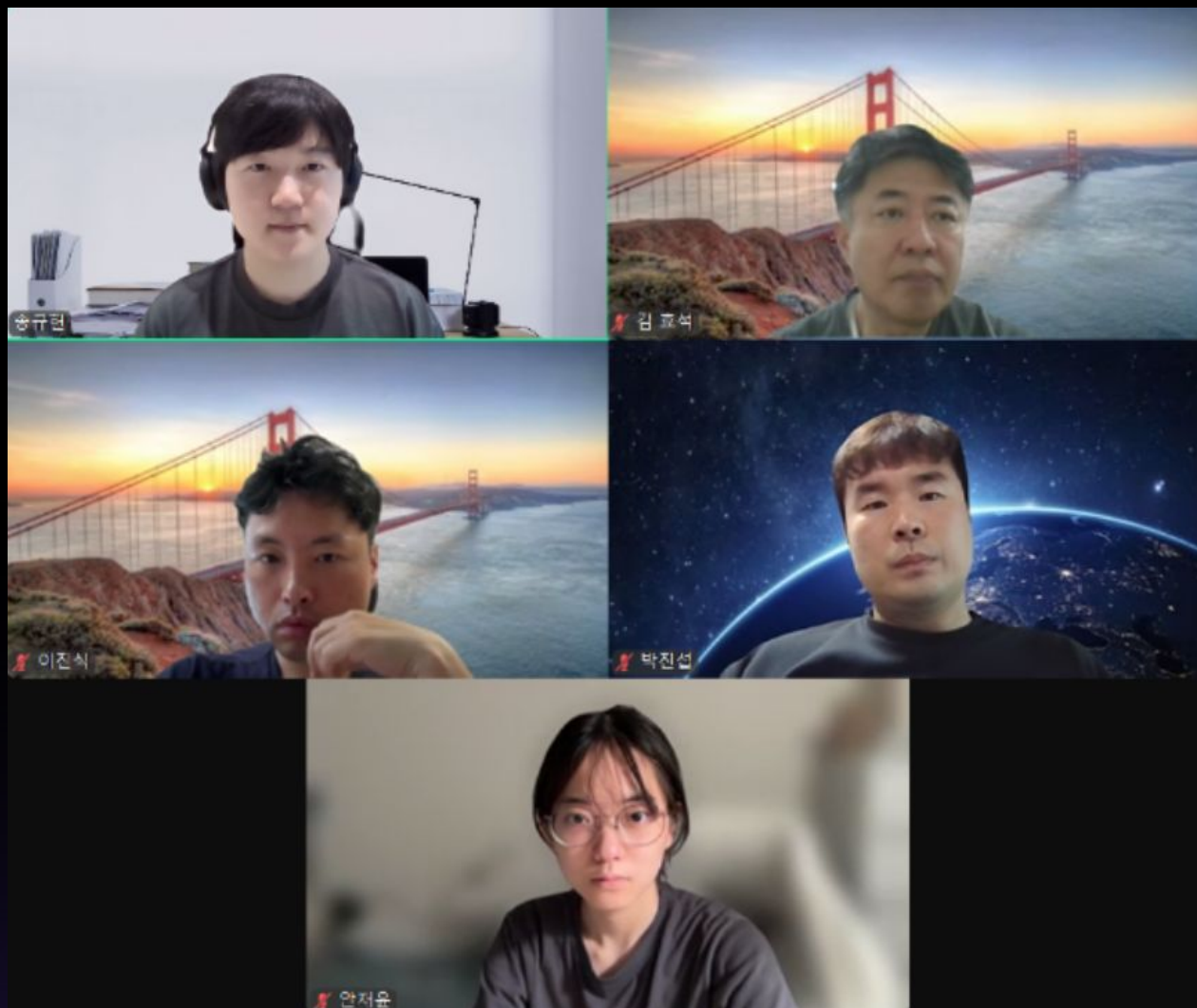
* 개선 방향



- mlflow 완성하여 webapp에 연계
- 모니터링 기능 추가

프로젝트 진행 소감

: MLOps Project | 느낀점



* 송규현

Docker를 활용하여 개발 환경을 통합하고 효율적으로 자원을 관리하는 방법을 직접 체험할 수 있었습니다. 데이터 전처리부터 모델 학습, 추론 API 개발 및 배포까지의 전 과정을 직접 구성하면서 end-to-end MLOps 파이프라인을 구축해볼 수 있었습니다. 이 과정에서 MLOps의 핵심 개념들을 자연스럽게 익힐 수 있었습니다.

* 김효석

Docker와 Airflow를 활용해 end-to-end MLOps 파이프라인을 직접 구축하며 모델 개발을 넘어 자동화와 배포까지 경험할 수 있었습니다. 개념 이해와 구현 과정은 어려웠지만, 반복적인 트러블슈팅을 통해 실전 감각을 키울 수 있는 값진 시간이었습니다.

* 박진섭

이번 프로젝트를 통해 단순한 모델 개발을 넘어, 데이터 수집부터 자동화, 예측 서비스까지 전체 MLOps 파이프라인을 구현해보는 경험을 쌓을 수 있었습니다. 특히 Airflow와 Docker를 활용한 자동화 구성, Hyperopt를 활용한 하이퍼파라미터 튜닝 과정에서 많은 실전 감각을 익혔습니다. 이후에는 MLflow 기반의 실험 추적과 API 확장 등 운영 관점에서의 보완을 해보고 싶습니다.

* 안재윤

모델의 버전별 관리와 실험 관리를 위해 MLflow를 도입하여 Docker 기반의 Tracking 서버 및 모델 레지스트리 환경을 구축하였습니다. 실험 자동 로깅과 모델 버전 관리 파이프라인을 구현하면서, 도구 도입 시 팀 커뮤니케이션과 교육, 그리고 DB/저장소 백엔드 설계의 중요성을 깊이 체감할 수 있었습니다. 특히 팀원들과 실험 관리 프로세스를 표준화하고 협업의 효율성을 높인 경험이 가장 인상 깊게 남았습니다. 추후에는 실시간 서빙 자동화를 도입하여 ML프로젝트의 모든 생애주기에 대한 경험을 확장해나가고 싶습니다.

* 이진식

어려웠다. 좋은 강의와 중간중간 멘토링, 그리고 약 2주의 시간이 있었지만, 도커와 도커 컴포즈를 이용해서 컨테이너를 관리하고 airflow나 mlflow를 이용해서 workflow 관리 하는 것을 익혀서 프로젝트를 완성하는 것은 힘들었다. 개념의 이해도 힘들었고, 개념을 적용시켜 구현을 할때마다 트러블샷 해야할 것들이 생겨서 LLM에게 계속 물어볼 수 밖에 없었다.

Life-Changing Education

감사합니다.
