

2024-07-25

인프라 스펙 및 아키텍처 관련

내용

- 인프라: 쿠버네티스로 **AWS** 인스턴스 + 온 프레미스 환경
- **airflow**: 제어관련 노드들을 **master** 단일에 통일 복사본 **X**, **worker** 노드 3개
- **spark**: 온 프레미스로 작동
- **DB**: VIP 활용 이중 **DB**를 사용하여 항상성 유지

TODO:

- **port** 인바운드, 아웃바운드 규칙 확정
- 추출 데이터 스키마, **DB** 데이터 스키마 확정(이를 위해서 목표 사이트 분석 필요)

2024-07-31

데이터 스키마 관련 1차 회의

참여자: 유정연, 김대건, 이서연

내용:

- 원티드, link-in, 로켓 펀치 크롤링 시도 및 사이트 분석 완료
- **Link-in:**
 - 필터: 지역(전국), 직업 검색어, 월급(하한선), 근무시간 및 형태, 경력
 - 필터 조건에 대한 결과를 최대 1000건만 보여줌
 - 검색 쿼리를 보낸 타임 스탬프 기준으로 보여줌
 - 추출 방법: 공고리스트 검색 쿼리 + 세부 내용은 Selenium
 - 관련: 사이트 내 검색 쿼리에서 공고 게시물의 id값을 추출할 수 있는지
 - 공고글 별 id 값 추출 가능
 - 계정 로그인 해야 기술 스택 가능
 - 회사 id값 있을 가능성 있음
 - 추출 가능한 항목: 공고 제목, 게시일, 필요스킬?, 회사, 공고 내용(형식이 없고, 전문성 등급은 있음), 직무 분야, task 유형
- **Wanted:**
 - 공고글, 회사 id값 존재
 - id값 기반 검색쿼리 api를 get으로 호출 가능
 - api는 limit와 offset으로 제어되며, offset을 시작으로 limit 만큼 결과 보여줌
 - 필터 값에 직무-개발이 있지만 몇몇 포지션은 다른 직무쪽으로 산재되어 있음 찾아볼 필요 있음
 - 3500건 정도 있음
 - 회사 id 기반 회사 정보 접근 가능
 - 추출 가능한 항목: 공고 제목(포지션 내용), 주요업무, 요구사항, 우대사항, 마감일, 회사 정보
- 로켓펀치:
 - 필터 조건이 큰 의미가 없음
 - 데이터 자체가 별루 없음(IT 개발쪽 필터 걸었을시 대략 1200건)
 - 추출방법: request get으로 접근
 - 회사별 id값 존재, 공고글 별 id값 존재 가능성 있음
 - 추출 가능한 항목: 공고 제목, 게시일&마감일, 업무 내용(detail, 형식이 없고, 내용이 천차만별임), 경력사항
- 프로그래머스:
 -

2024-08-12

데이터 스키마 재정립 논의

참여자: 유정연, 김대건

- 논의 내용

- 1. 각 사이트별 공고 중복은 당장은 확인 불가(변수가 너무 많음)
- 2. 우리가 아무리 정제를 해도 채용 사이트 템플릿에서 보여주는 공고보다 못할 가능성이 높음
- 3. 트렌드 분석시 채용공고 중복될 때 신뢰도가 떨어질 가능성이 있으므로 중복 처리가 필요하다
- 4. 데이터 생명 주기; 같은 공고라도 모집기간 or 공고 날짜가 다른 경우; 시간에 따라 기술 트렌드 변화로 내용 변화
 - 공고 기간이 없는 경우(상시 채용)
- 5. 직무 유형이 각 사이트마다 다름 (BE 개발자, IT 서비스가 같은 직업군으로 취급 되는 경우)

- 요구 사항

- 1, 2: 사용자 필요사항에 대해서 사용자가 원하는 정보를 필터를 제공하고 세부 정보나 공고 지원을 위해 해당 링크로 가는 바로가기 버튼 제공
- 3: 단위 시간 당 채용공고를 중복처리 or 공고글 기준이 아닌 회사 기준으로 기술 스택 정리
 - 3-1: 정제 단계를 여러번 걸치고 실버~골드 정도 수진일 때 중복 처리 가능할 것으로 보임
 - 3-2: 회사내 여러 포지션마다 요구하는 기술이 다르므로, '회사-포지션' 기준으로 기술스택 정리 필요
- 4: 채용 중 **feature** 값 추가하여 최신 공고글 제공
- 5: EDA 작업을 통해 직무 유형 분류를 확인할 필요 있음

- 데이터 질적 수준 등급

등급	수준	설명
동 등급	Lv.1	완전 Low; 간단한 전처리라도 하지 않으면 아예 쓸수 없는 단계(공백, 줄바꿈 등)
	Lv.2	간단한 전처리를 하지 않아도 쓸 여지가 있는 상태인 Low data
	Lv.3	간단한 전처리 및 원본을 기반으로 파싱 진행할 경우
은 등급	Lv.4	데이터 값에 대한 전처리 진행(불필요한 특수기호 제거, 동의어 일괄 변경)
	Lv.5	원본 데이터에서 추출한 변수가 아닌 파생 변수 생성할 경우
	Lv.6	OLDP(row 기준; 공고 목록) vs OLAP(col 기준; 트렌드 분석) 분리
금 등급	Lv.7	- 공고 목록: 채용 중, 마감율 나타낼 값과 id기반 사이트 식별 코드 부여(hash or 채용사이트 심볼) - 트렌드 분석: ML에게 던져서 주요 기술 스택 추출

	Lv.8	<ul style="list-style-type: none"> - 공고 목록: 필터값으로 보여줄 데이터 지정(어떤 값을 공개할 것인가?) - 트렌드 분석: 직무 유형(or 회사 산업 분류)를 정제 or ML로 추출
	Lv.9	<ul style="list-style-type: none"> - 공고 목록: 미정 - 트렌드 분석: 분기별 조회를 위한 회사이름과 공고 채용 기간 올린 시점 추출 및 중복 처리

2024-08-19

주제: 마일스톤 제작 및 **WBS progress** 설정

참여자: 유정연, 이서연, 김대건, 최성현, 박성우

진행 및 결정 사항:

1. ERD 제작 완료

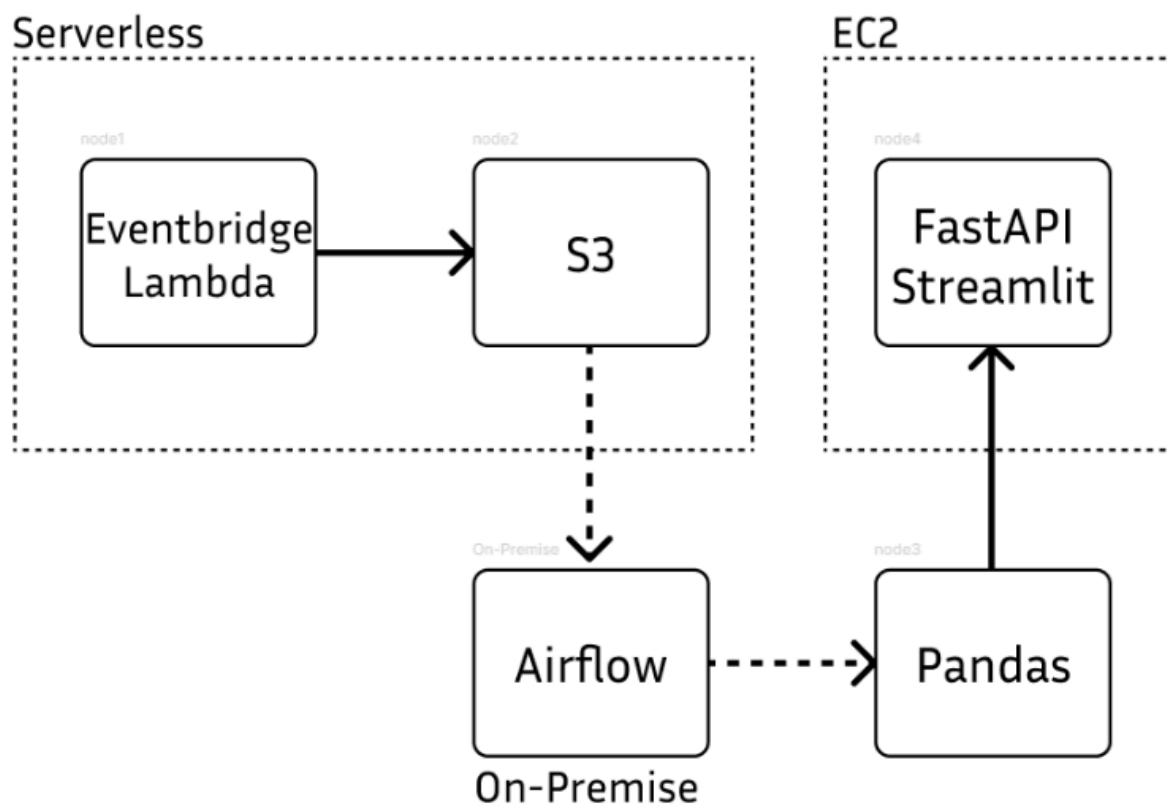
2. FE는 Streamlit을 사용하여 제작할 예정

3. github issues 마일스톤 등록

- <https://github.com/S0rrow/FPT5/milestones>

4. spark 드랍 결정

- 데이터 정제시 사용 예정이었으나 데이터양에 비해 부하가 많이 걸려 불필요할 것으로 판단
- **Pandas** 로 대체



2024-08-23

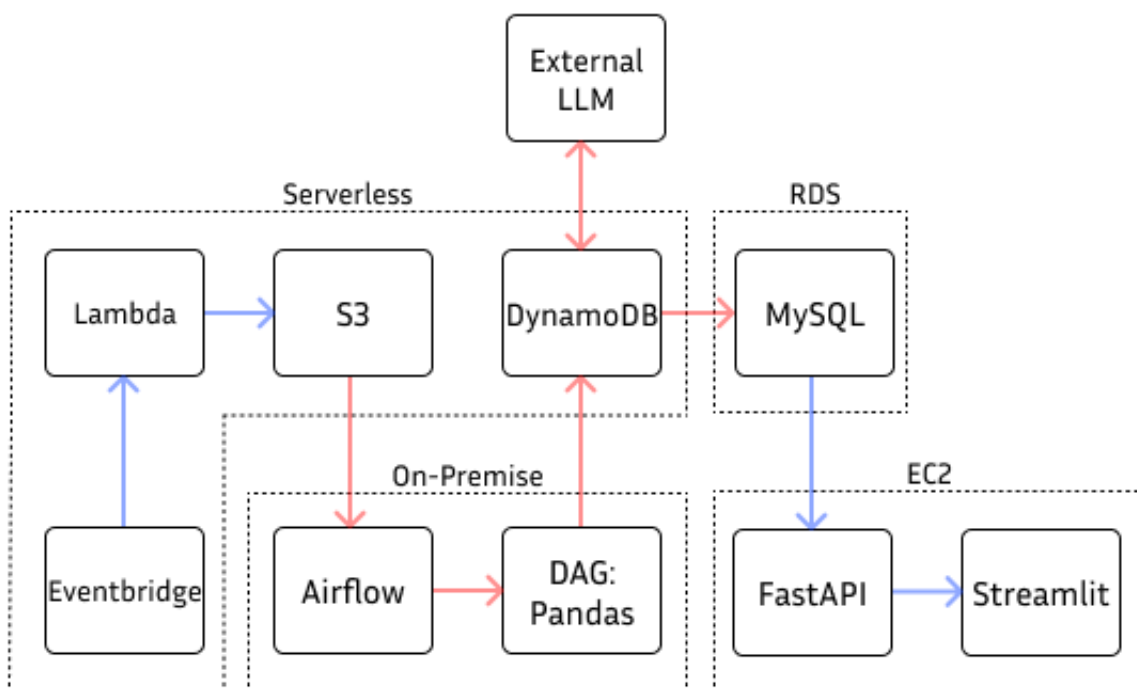
멘토링

참여자: 유정연, 이서연, 김대건, 최성현, 박성우

이번 주 진행 상황

0. Overview Image

- 여러 프레임워크의 사용 PoC를 검토하고 테스트해본 결과 다음과 같은 아키텍처로 시스템을 구성하는 방향으로 작업 중입니다.



1. 데이터 수집 기능 구현

- 각 사이트별 데이터 추출을 위한 크롤러 코드 작성 및 테스트 완료
- 각 사이트별 크롤러 코드 AWS Lambda 코드로 변환 및 테스트 완료
- 크롤러 결과물들의 결과 데이터 적재를 위한 S3 버킷 생성 및 테스트 완료
- AWS Eventbridge를 통한 AWS Lambda 코드 실행 스케줄 생성 및 테스트 완료
- 데이터 수집 파이프라인 최종 검수 및 수정 완료

2. 데이터 정제 기능 구현

- 정제된 데이터를 적재하기 위한 DynamoDB 생성 및 연결 테스트 완료
- S3에 적재된 데이터를 가져와 정제하는 코드 작성 진행 중
- 데이터 로드, 정제, 적재 테스트

3. 데이터 제공 기능 구현

- FastAPI를 통해 MySQL 서버로부터 RESTful API 방식으로 Dataframe을 가져오는 서버 구현 및 테스트 완료
- Streamlit을 통해 FastAPI에서 Dataframe을 가져와 여러 종류의 차트로 시각화하는 기능 구현 및 테스트 완료
- Google API를 통해 Streamlit에서 로그인 및 계정 정보 제공 기능 구현 및 테스트 완료

다음 주 구현 목표

1. 데이터 가공 기능 구현

- S3에 적재된 데이터를 DAG 및 Pandas를 통해 가공
- DynamoDB에 적재된 데이터를 RDS에 2차로 가공 및 적재
- DB로부터 웹 인터페이스에게 데이터 가공 및 제공

2. 데이터 제공 기능 구현

- Streamlit을 통한 웹 인터페이스 제작
 - 사용자 입력에 따라 데이터를 필터링하고, 이를 그래프 또는 차트로 시각화
 - 사용자가 데이터 범위, 파라미터를 선택할 수 있는 입력 폼 제공

질문 사항

Q1. 2차 데이터 가공 때 LLM을 사용할 예정인데, Prompt Engineer를 할 때 참조할 만한 자료가 있는지?

A1. 클라우드 또는 대량의 데이터를 처리하는 경우 Gemini AI 활용하는 것을 추천.

프롬프트 엔지니어링의 경우 다음의 사이트 참조해보는 것도 좋을듯

- <https://www.promptingguide.ai/kr>
- <https://console.cloud.google.com/vertex-ai/generative/prompt-gallery?project=wev-prod-artis-t-analysis>
- <https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/design-multimodal-prompts?hl=ko>
- <https://cloud.google.com/vertex-ai/generative-ai/docs/learn/introduction-prompt-design?hl=ko>

LLM 모델을 활용하는 경우 어디에 활용할 생각인지?

2차 가공하는 단계에서 데이터 전처리할때 사용할 예정

Q2. 온 프레미스로 구축한 시스템 구성요소들을 클라우드로 옮길 생각인데 이걸 단순히 EC2 인스턴스에 마이그레이션하는 것과 아니면 AWS 서비스 기능을 사용해서 재구축하는 것중 무엇이 좋을지?

A2. 기존에 있는 서비스를 쓰는 경우 HA나 추가적인 유지보수를 신경쓰지 않아도 된다는 점이 장점임. EC2에 별개의 서비스를 구축하는 경우에는 직접 구축해야한다는 점이 장점이자 단점.

Q3. 모니터링 서버의 필요 유무 질문; 필요하다면 어떤 정보에 대해서 시각화하는 것이 좋을지?

A3. 다음과 같은 항목들에 대해서 시각화하면 될듯

...

네, 이해했습니다. 채용공고 웹사이트의 데이터를 크롤링하여 **AWS** 환경에서 처리하고 시각화하는 서비스를 모니터링하기 위한 그래파나 대시보드를 만드는 것에 대해 조언드리겠습니다. 다음과 같은 주요 지표들을 고려해볼 수 있습니다:

1. 데이터 수집 관련 지표:

- 크롤링 성공률
- 수집된 채용공고 수 (일별/주별/월별)
- 크롤링 소요 시간
- 크롤링 에러 발생 횟수 및 유형

2. ETL 프로세스 관련 지표:

- **Airflow DAG** 실행 성공률
- 태스크별 실행 시간
- **RDS** 데이터 적재 속도
- 데이터 처리량 (records/second)

3. RDS 성능 지표:

- **CPU** 사용률
- 메모리 사용률
- 디스크 I/O
- 쿼리 실행 시간
- 연결 수

4. Streamlit 애플리케이션 성능:

- 페이지 로드 시간
- 사용자 세션 수
- 에러 발생률
- 리소스 사용률 (CPU, 메모리)

...

멘토링 결과

1. 프롬프트 엔지니어링

LLM 추천 모델

제미나이: 무료로 사용 가능하여 멘토님 추천.

클로드: 추가 검토 필요.

가이드: **Prompting Guide** 참고.

url: <https://www.promptingguide.ai/kr>

구글 플랫폼 추천: 다양한 AI 모델들이 제공되는 플랫폼.

사례별 추천: Google Vertex AI Prompt Gallery에서 멀티모달 프롬프트 설계 사례 참고 가능.
url:
<https://console.cloud.google.com/vertex-ai/generative/prompt-gallery?project=wev-prod-artist-analysis&pli=1>

2. 데이터 정제

항목 정리 필요성

추출하려는 데이터 항목에 대한 명확한 정리가 필요.
에어플로우(온프레임) vs 클라우드 매니지드 서비스

클라우드 매니지드 서비스가 편리하며, 구축 경험이 있어 추천됨.
간단한 스케줄링에는 이벤트 브릿지 사용 가능.
종합적인 오케스트레이션에는 에어플로우가 적합하며, 예를 들어 실패 시 알림 메일 전송 가능.
에어플로우 사용 시 고려사항

API 호출을 통해 에어플로우를 사용하는 방안 검토. 마이그레이션 시 이슈를 최소화할 수 있음.
코드 변경 시 배포 방안을 고민해야 함.
데이터 인제스천

데이터 인제스천의 개념에 대한 이해가 부족하다는 의견.
에어플로우를 단순한 스케줄러로만 생각하면 서비스 구성에서 중요한 요소들을 놓칠 수 있음.
데이터 흐름을 정의하는 것이 중요하며, 에어플로우가 다른 서비스와 연계될 때의 이점 고려 필요.
진행 방법

한 번에 완벽하게 하려 하지 말고, 시행착오를 거치며 수정해 나가는 자세가 필요.
LLM 사용 시 데이터 정제

다양한 LLM에 데이터를 입력하여 정제 성능을 평가하고, 성능에 따라 적합한 모델을 선택.
POE 서비스 추천: 여러 LLM을 비교 및 테스트할 수 있음.

3. 모니터링

그라파나+프로메테우스 대시보드

대시보드에 추가할 항목에 대한 논의 필요.
멘토님의 답변: 함께 찾아보며 최적의 항목을 결정하자.

[클로드 답변 참고]

데이터 수집 관련 지표:

크롤링 성공률

수집된 채용공고 수 (일별/주별/월별)

크롤링 소요 시간

크롤링 에러 발생 횟수 및 유형

ETL 프로세스 관련 지표:

Airflow DAG 실행 성공률

태스크별 실행 시간

RDS 데이터 적재 속도

데이터 처리량 (records/second)

RDS 성능 지표:

CPU 사용률

메모리 사용률

디스크 I/O

쿼리 실행 시간

연결 수

Streamlit 애플리케이션 성능:

페이지 로드 시간

사용자 세션 수

에러 발생률

리소스 사용률 (CPU, 메모리)

4. 개발 툴 추천

Cursor: 비주얼 스튜디오 대신 추천. 프롬프트를 생성할 수 있으며 코파일럿과 유사한 기능 제공.

Poe: 모델 테스트 사이트

2024-08-29

멘토링

참여자: 유정연, 이서연, 김대건, 최성현, 박성우

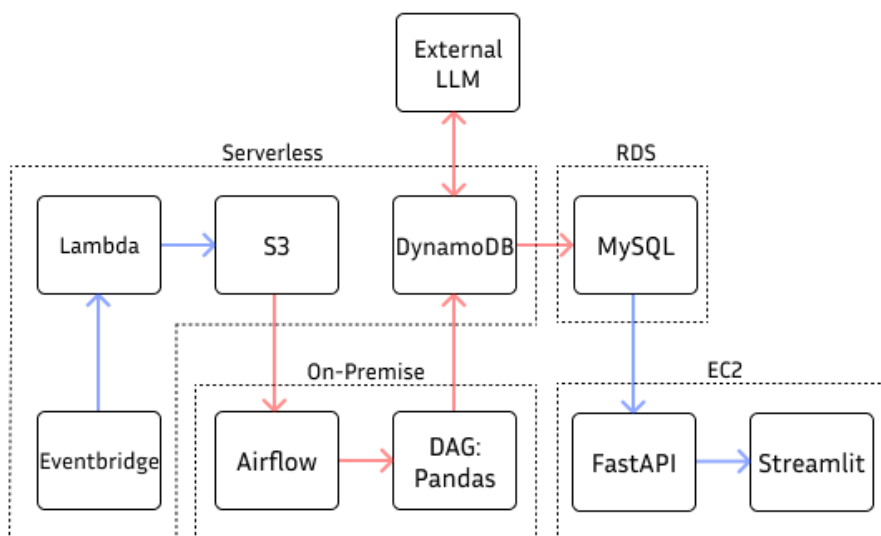
프로젝트 목표 Remind

이 프로젝트는 여러 채용 사이트들에서 여러 공고들을 수집하고 분석해 특정 업무 분야에 대해서 다양한 업무 분야에서 실제로 각광받거나 많이 사용되는 기술 스택의 목록을 추천하는 시스템을 구성하기 위한 것입니다.

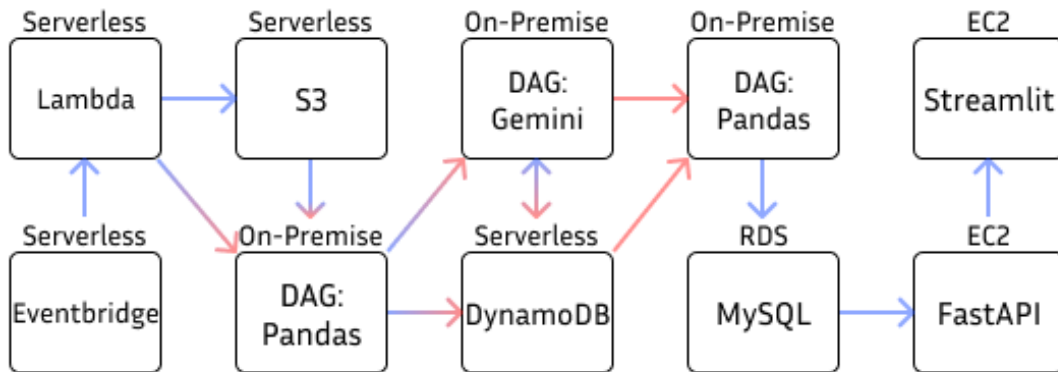
이번 주 진행 상황

0. Overview Image

기존의 아키텍처



현재의 아키텍처



(파란 화살표= 완료, 붉은 화살표= 준비중)

진행 순서

1. **AWS Eventbridge**의 scheduler 기능을 활용해 지정한 시간에 **Lambda**를 통해 크롤러 시동.
2. 결과가 나오면 **AWS S3**에 데이터를 적재하고 **Lambda**를 통해 **Airflow** 서버에 **API** 요청, DAG 시작.
3. DAG를 통해 S3에서 데이터를 끌어와 1차 전처리(특수문자 제거, Hash id값 추가, 각 datetime을 Timestamp로 통일)를 진행하고, 그 결과를 **DynamoDB**에 적재.
4. DAG가 완료되면 추가적으로 다시 **Airflow** 서버에 API를 요청해 **Gemini**에 프롬프트 엔지니어링을 거친 DAG를 시작.
5. DAG의 결과물을 동일하게 **DynamoDB**에 적재하고, 다시 **Airflow** 서버에 API를 요청해 DAG 시작, 최종적으로 **RDS**상에 구축된 **MySQL**서버에 적재하기 위해 데이터 처리.
6. **MySQL**에 적재된 데이터를 기반으로 **FastAPI**와 **Streamlit**으로 이루어진 웹 인터페이스를 클라이언트에게 제공.

1. 데이터 정제 기능 구현

- 데이터 1차 전처리 완료,
- 2차 전처리를 LLM 프롬프트 엔지니어링 완료.
- 2차 전처리 및 적재 python 코드화
- Airflow DAG 진행 중.
- Lambda 코드 완료.

2. 데이터 제공 기능 구현

- FastAPI와 Streamlit을 통한 multiselect 기능 구현 완료
- RDS에서 SQLAlchemy를 통해 FastAPI 서버로 데이터 가져오는 기능 구현 완료
- 사용자 입력에 따라 데이터를 필터링하고, 이를 그래프 또는 차트로 시각화하는 기능 구현 완료
- 사용자가 데이터 범위, 파라미터를 선택할 수 있는 입력 폼을 제공하는 기능 구현 완료

이번 주 구현 목표

1. 데이터 정제 기능

- 1차 전처리 마무리 및 **DAG** 작성.
- 2차 전처리 코드 예외 처리 및 최적화
- 2차 전처리 코드 **DAG** 작성
- 2차 전처리 결과 저장용 **DynamoDB Table** 생성

다음 주 구현 목표

1. 데이터 정제 기능 마무리

- 2차 전처리까지 마무리하고 **RDS**에 데이터를 적재하는 것까지 마무리

2. 데이터 제공 기능 추가 기능 구현

- 구글 **API** 로그인 기능 구현한 것을 기반으로 유저별로 검색 데이터를 저장해 추가적인 기능 구현.

3. **EC2** 구축 및 마이그레이션

- **Streamlit**과 **FastAPI** 서버 **EC2**로 마이그레이션 진행

4. **FastAPI**로 전처리 완료된 데이터 가져와서 웹 **UI**에 보여주기

- **S3 > DynamoDB > RDS**로 이어지는 데이터 파이프라인의 뼈대 구축 완료해서 **RDS**까지 전달해 웹 **UI**에 보여주기

질문 사항

1. 시스템에 관한 질문(모니터링)

- **Q:** 모니터링 서버를 구축할때, **Grafana** 또는 **CloudWatch**를 사용하는데, 현재 2가지를 전부 사용하고 있는데 하나로 모니터링 플랫폼을 통일하는게 좋을지, 또는 그냥 현재 시스템의 2가지를 그대로 사용하는게 좋을지 궁금합니다.
- **A:** 시스템이 기본적으로 **AWS** 상에서 구성되어 있다는 점에서 **CloudWatch**를 쓰면 처리하기 간편하고 **AWS** 측에서 관리해준다는 점에서 유용할 것. **Grafana**의 경우 구축과 설정이 좀 더 어렵지만 세부적인 시각화 설정을 건드릴 수 있다는 점과 현역에서 자주 사용된다는 점에서 경험을 쌓는다면 유용할 것임. 모니터링 플랫폼의 경우 일장일단이 있을테니 팀원들과 논의해서 잘 선택하길

2. 사용자가 웹 인터페이스를 사용할 때 로그 데이터를 수집하는 방법 관련 질문

- Q: **Streamlit**을 사용한 웹 인터페이스를 제공할 때, 클릭 단계에서 데이터를 수집하는 것까지는 파이썬으로도 구현이 가능한데, 그 이외에 행동(스크롤을 통해 추가적인 **element**들이 렌더링 되는 경우 등)에 대해서는 실시간으로 감시하기 어려운 상태. 혹시 방법이 있을지 질문하고 싶음. 결과적으로 **JS**를 쓰는 것 외에 순수하게 파이썬의 범주 안에서 해결하기에는 어려운 문제일지.
- A. **Streamlit**에 **js** 코드를 삽입하는 방법이 존재함. 애초에 **Client Side**에서 발생하는 이벤트의 경우 **js** 외에는 추가적으로 감시할 방법이 제한적임. 근데 애초에 렌더링이나 버튼 클릭과 같은 이벤트의 경우 **Streamlit** 자체적으로 로그 데이터를 수집할 방법이 존재함. 굳이 **JS**를 삽입할 필요가 있을지?