AWS Lambda를 이용한 Serverless 챗봇 개발

– 관련 아티스트 추천 –

이마태

2020년 4월 ~ 2020년 7월

목차

- 개요
- Spotify API 데이터 소개
- 관련 아티스트 추천 알고리즘
- 데이터 파이프라인
- 예시 화면
- 추후 발전시키고 싶은 것들

개요

- Spotify(음원 서비스) API의 아티스트/음원 데이터를 이용하여, 아티스트를 입력하면 관련 아티스트를 추천해 주는 카카오톡 챗봇. 입력받은 아티스트와 **유사도가 가장 큰 아티스트를 추천**해 줌
- **리눅스 crontab**을 이용하여 데이터 처리 자동화
- 물리적인 서버를 구동하지 않고, 사용자의 메시지 request가 있을 때에만 작동하는 Serverless 방식
- 데이터 처리 – **배치 처리**
  1. API에서 가져온 raw data(아티스트별 인기 트랙, 트랙별 음원 특성)를 Amazon S3에 저장하여 Data Lake 구성
  2. S3 데이터를 Athena를 통해 쿼리. 트랙별 음원 특성 벡터를 이용하여 아티스트들 사이의 유사도(Euclidean distance)를 계산한 후, MySQL에 저장하여 Data Mart로 사용
- 데이터 처리 – **실시간 처리**
  - 사용자가 카카오톡에서 아티스트를 입력하면, 해당 아티스트 및 유사도가 가장 큰 3개 아티스트의 인기 트랙을 응답해 줌

- **사용 기술**
  - **Python, AWS 서비스** (API Gateway, Lambda, EC2, S3, Athena, MySQL, DynamoDB)

Spotify API 데이터 소개

1. Artists
   - **아티스트** 이름, ID(Spotify 지정)
2. Top_tracks
   - **아티스트**의 인기 **트랙**(곡)
3. Audio_features
   - **트랙**의 특성이나 분위기를 수치화
   - ex: loudness(곡 세기, dB 단위), energy(빠르고 시끄러운 정도)

관련 아티스트 추천 알고리즘

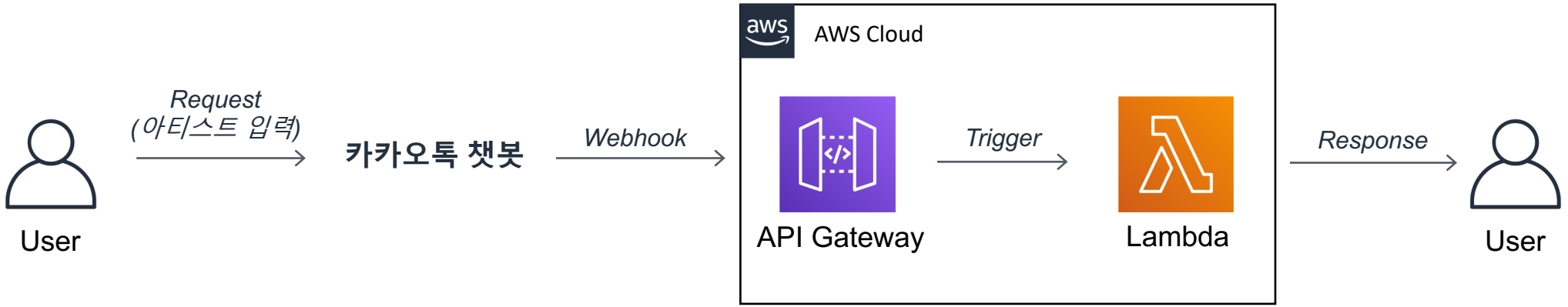
1. Artist별 audio_features 벡터 생성
   - **Artist**별로 10개 정도의 **Top_tracks**가 있음
   - A 아티스트가 있다고 하면, 이 아티스트의 **Top_tracks**에 있는 트랙들의 **Audio_features** 수치별 **평균** 사용
     → *mean(loudness) = mean(track1의 loudness, track2의 loudness, …, 마지막 track의 loudness)*
   - **Audio_features**의 수치 6개 사용. 6개의 원소를 가지는 벡터 → *(mean(loudness), mean(energy), …)*
   - 수치별로 scale이 다름. 특정 수치의 영향을 줄이기 위해 **정규화**하여 scale을 0~1 사이로 맞춰 줌

2. 벡터를 이용하여 Artist들 간의 유사도 계산
   - Artist별 벡터를 이용하여 Euclidean Distance 계산
   - Artist가 n개이면, **nC2** = $\frac{n\times(n-1)}{2}$ 회 계산

3. Artist별로 거리가 가장 가까운 Artist 3개를 MySQL에 저장
   - 최종적으로 챗봇에 응답해 줄 데이터가 됨

데이터 파이프라인 – 1 (챗봇 연결)

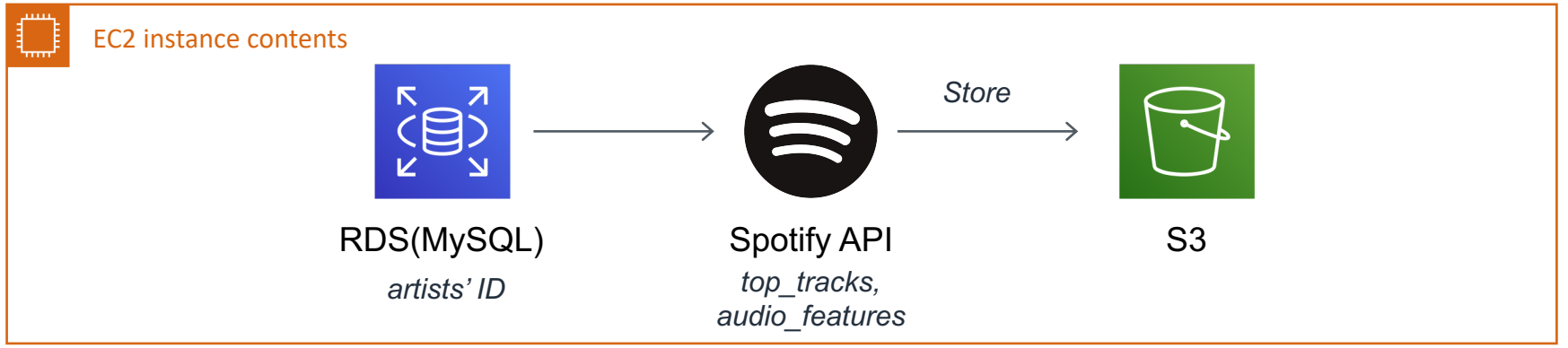
AWS Lambda와 연결



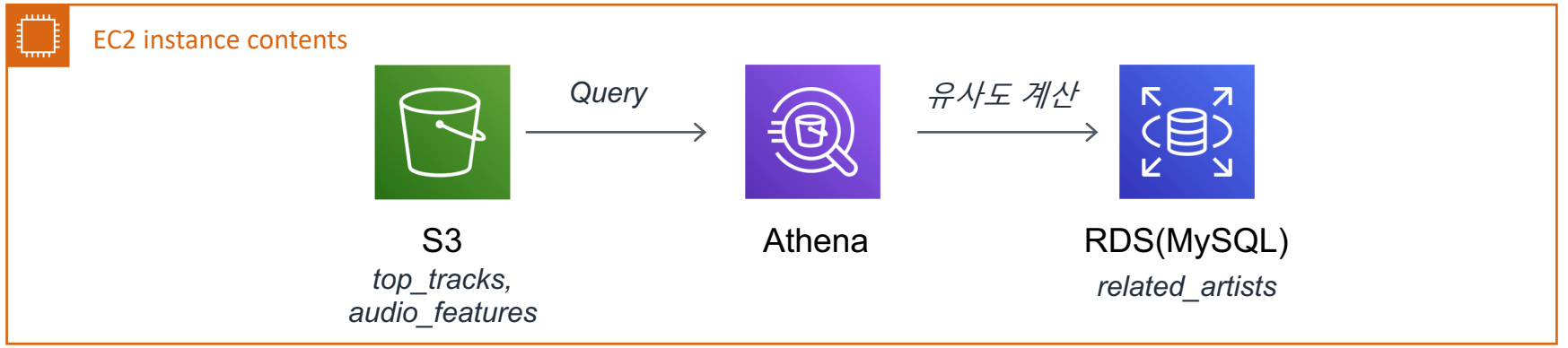
데이터 파이프라인 – 2 (배치 처리)

Linux(EC2) Crontab을 통해 매일 밤 자동화 스크립트 실행

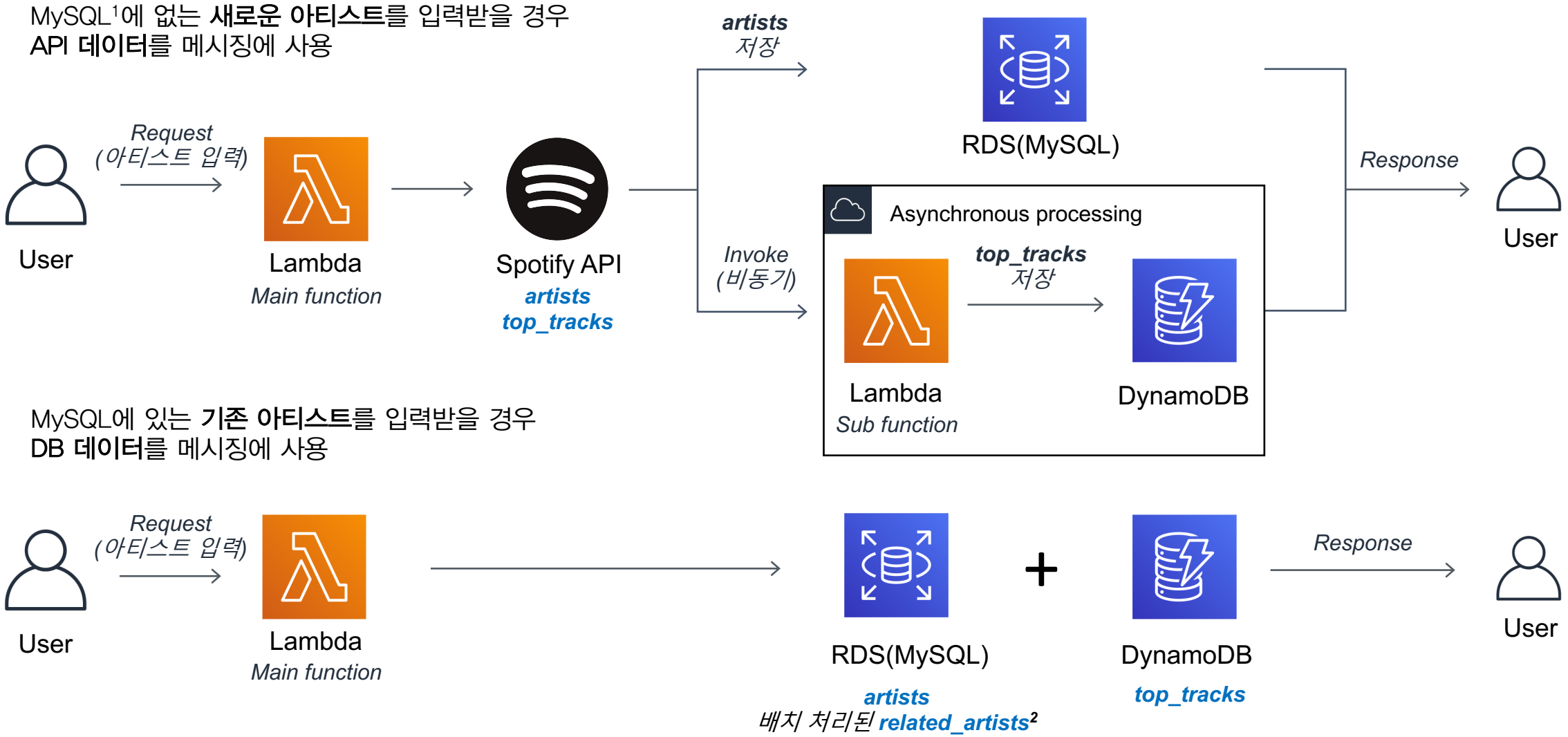
1. Artist들의 ID로 Spotify API 데이터 쿼리, S3에 저장하여 Data Lake 구성 *(ttandaudio_to_s3.py)*



2. Athena에서 쿼리 및 유사도 계산, MySQL에 저장 *(related_artists.py)*

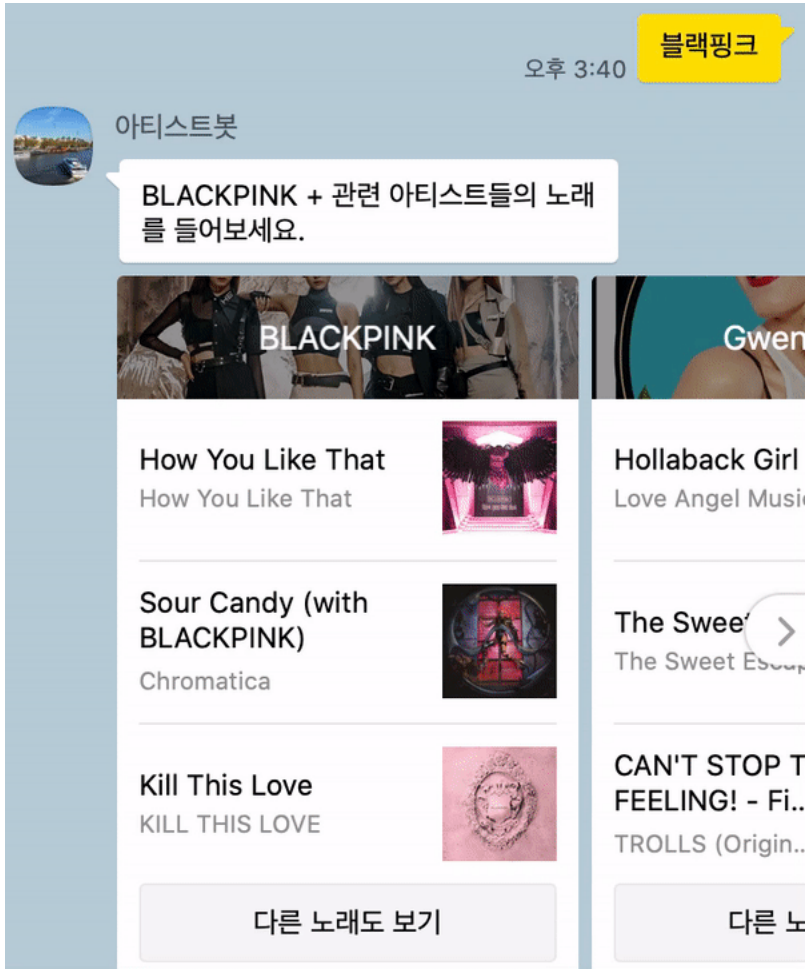


데이터 파이프라인 – 3 (실시간 처리)



1. artists 테이블에 해당 아티스트가 있는지 조회
2. 관련 아티스트(related_artists)는 기존 아티스트 대상으로만 제공

예시 화면



‘블랙핑크’를 입력할 경우

- 카드를 넘기면서 블랙핑크의 관련 아티스트인 ‘*Gwen Stefani, Red Velvet, miss A*’ 확인할 수 있음
- 트랙을 클릭하면 유튜브에서 ‘가수 이름 + 트랙 이름’을 검색한 화면으로 넘어감
- ‘다른 노래도 보기’ 버튼을 클릭하면 유튜브에서 가수 이름을 검색한 화면으로 넘어감

추후 발전시키고 싶은 것들

1. 추천 알고리즘

- 유사도로 **Cosine similarity** 사용 → 특정 수치가 가깝지 않고, 전체적인 수치가 유사한 아티스트 추천 가능
- **가사** 기반 추천 → 가사에 등장하는 단어 임베딩 후 TF-IDF 이용
- 유저의 사용 로그 수집 후 **Collaborative Filtering** 적용 → 노래의 특성뿐만 아니라 나와 유사한 유저의 취향까지 반영할 수 있음

2. Airflow 등 ETL Tool 사용

- 현재는 각 태스크의 실행 시간을 고려하여, Linux Crontab의 실행 시점 결정
  - Ex: 02시 20분에 *ttandaudio_to_s3.py*, 02시 30분에 *related_artists.py* 실행
- ETL Tool을 사용하여, 이전 Task가 완료되면 다음 Task 진행

3. 아티스트별로 트랙의 audio_features 수치 평균이 아닌, 각 트랙의 수치 사용

- 현재는 **평균**을 사용하기 때문에 **정보 손실**이 생김
- 각 트랙의 고유 수치 사용

기타

- 자세한 내용 및 코드는 블로그와 GitHub에 올려 놓았습니다

  - 블로그: https://sulmasulma.github.io
  - GitHub: https://github.com/sulmasulma/kakao-chatbot
  - 챗봇 사용해 보기: https://pf.kakao.com/_xgubvxb

감사합니다