

ETL_Pipeline 구축 프로젝트

AI 16기 – CP1 DE 과정

이승훈

목차

- 01. 프로젝트 개요
- 02. 프로젝트 구성
- 03. 프로젝트 수행 절차 및 방법
- 04. 프로젝트 수행 결과
- 05. 자체 평가 의견

1. 프로젝트 개요

ETL Pipeline Basemodel

프로젝트 개요

API 서버에서 주기적으로 생성되고 있는 암호화된 로그데이터를 추출하여 복호화 하고, 효율적으로 압축 변환을 거친 뒤 AWS S3에 파티셔닝하여 저장하는 ETL 파이프라인 구축 프로젝트

ETL_Pipeline

추출(Extraction) - 변환(Transformation) - 로드(Load)의 절차를 따르는 데이터의 수집 및 가공·정비·구축의 데이터 처리 프로세스

장점

- 다양하고 복잡한 변환 수행
- 데이터 통제 유리
- 시각화 편리

단점

- 실시간 처리 부적합 (대용량 파일)
- 불필요한 대기 시간 발생

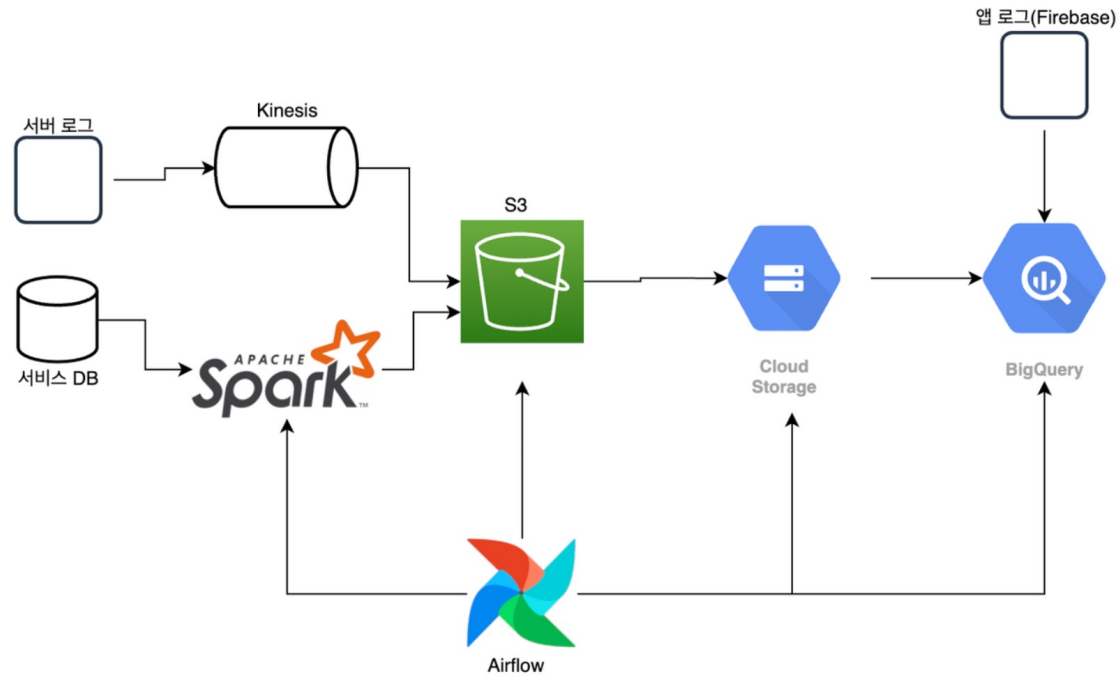
활용

- 배치

사례

- DW구축, 데이터 집중화, 표준화
- Data Silo간 이동

2. 프로젝트 구성



Json log Data → Extract & Transform & Load → AWS S3 upload → AWS Athena

데이터 추출 (Extract)

```
decrypted_data, str_compressed_data = json_gen(API_URL, FERNET_KEY)
json_save(ORIGIN_JSON_PATH, decrypted_data)
json_save(COMPRESSION_JSON_PATH, str_compressed_data)
print('Extract(데이터 추출) Complete!')
```

- Json 데이터 추출 후 Json파일 저장(문자열 압축 적용)

데이터 변환 (Transform)

```
GZIP(ORIGIN_JSON_PATH, ORIGIN_GZIP_PATH) # 모든 원본 데이터를 쌓아놓은 압축파일
GZIP(COMPRESSION_JSON_PATH, COMPRESSION_GZIP_PATH)
print('Transform(데이터 변환) Complete!!')
```

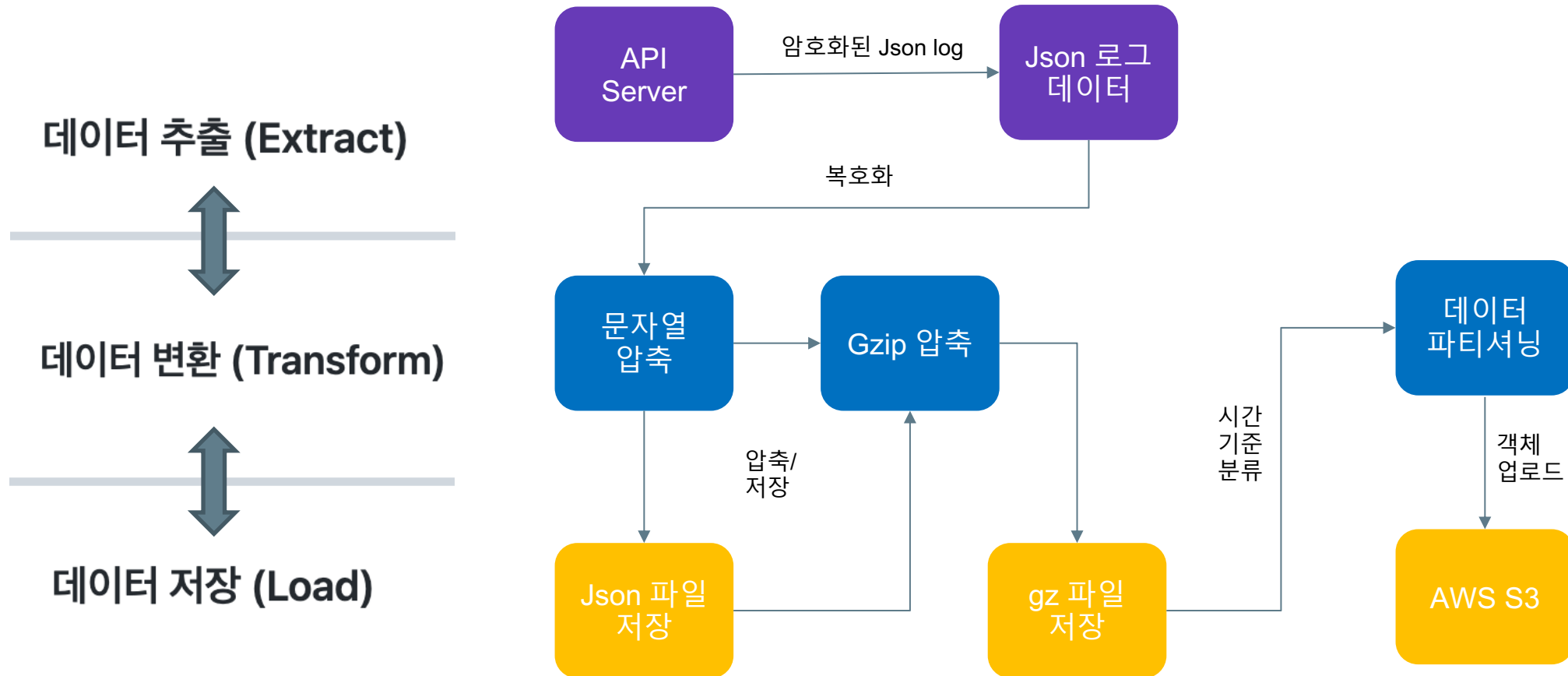
- 저장한 Json 파일을 gzip 압축하여 변환.

데이터 저장 (Load)

```
s3_upload_partitioning()
print('Load(데이터 저장) Complete!!!\n\n')
```

- .gz 압축파일을 파티셔닝하여 AWS S3 객체 업로드

3. 프로젝트 수행 절차 및 방법




4. 프로젝트 수행 결과


아래 이미지를 클릭하면 Github 페이지로 이동합니다.

GitHub - badro97/ETL_Pipeline


Contribute to badro97/ETL_Pipeline development by creating an account on GitHub.

 https://github.com/badro97/ETL_Pipeline

**badro97/
ETL_Pipeline**



1 Contributor 0 Issues 0 Stars 0 Forks



Github Link

https://github.com/badro97/ETL_Pipeline

개발문서

개발과정 정리

5. 자체 평가 의견

ETL_Pipeline 베이스 모델 구축 완료

느낀 점

1. 일반적인 파일 입출력 프로세스로는 대용량 파일 실시간 처리에 한계가 있다.
2. 다양한 변환 과정을 실습하며 공간 복잡도, 리소스에 대한 심화개념 학습의 필요성을 느꼈다.

추후 개선사항

1. AWS Athena 조회, AWS Glue 크롤러 사용법 숙지하고 적용하기
2. 대용량 파일 프로세스 처리 최적화