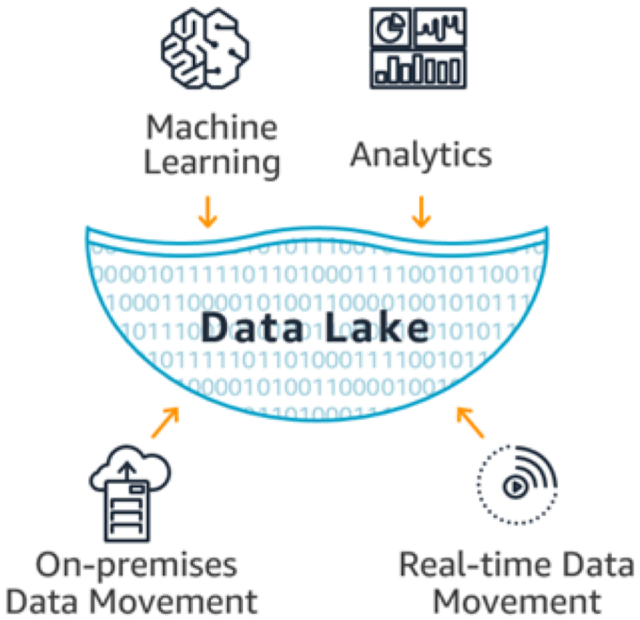
데이터 레이크 at kakao

데이터 레이크: 정의



모든 데이터를 저장 하는 Repository

- Structured data
- Unstructured data

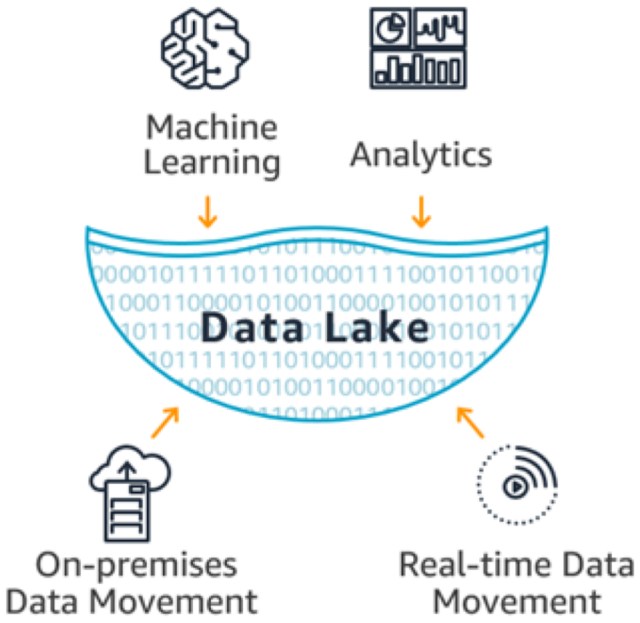
Schema-on-read

모든 종류의 데이터를 별도로 structure data로 변환 하지 않고 모두 그대로 저장

- 일단 저장하고, 활용할 계획이 나오면 알아서 잘 활용한다.
- 일단 데이터는 모여 있으니, 빅데이터 분석 + 머신러닝등 필요한 컴포넌트에서 활용할수 있다.

Reference: https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/

데이터 레이크: 현실



모든 데이터를 저장 하는 Repository
- Structured data
- Unstructured data

Schema-on-read

모든 종류의 데이터를 별도로 structure data로 변환 하지 않고 모두 그대로 저장

- 일단 저장, 근데 어디(Where)에 어떤 데이터(What)가 어떤 형태(How)로 저장 되는지 아무도 모름.
- 저장 되는 데이터가 추가 되거나 변경이 있어도 아무도 모름.
- Client입장에서 활용계획이 생길 때마다 기존 위키(or 엑셀)을 참조 -> 담당자와 논의 -> 개인정보 승인 -> POC -> 결과가 별로니 drop = 모두에게 소모적

Reference: https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/

데이터 레이크: Why?

아무도 어떤 데이터(What)이 어디에(Where) 어떻게(How) 관리 되는지 알 수 없다면, 굳이 필요할까?

개별 서비스의 데이터만으로는 충분한 가치를 만들기 어려움.

- 사용자의 행태 정보들을 사용자 기준으로 연결 하지 않으면, 사용자에 대해 파편적인 정보만을 활용하게 됨.
- 데이터의 파편화
  - 서로 다른 종류의 행태 정보들을 유기적으로 연결 하기 어려움.
  - 아무리 좋은 알고리즘, 모델링 기술이 있어도, 편협적인 곳에만 활용됨.

어떻게 관련된 데이터들을, 수작업을 줄이면서 연결 해 놓고, 이걸 필요한 곳에 잘 전달하고, 전달 받을 수 있게 하면?

- 데이터를 하나의 repository에 모으는 것이 의미를 가지게 됨.

데이터 레이크: What?

어떻게 관련된 데이터들을, 수작업을 줄이면서 연결 해 놓고, 이걸 필요한 곳에 잘 전달하고, 전달 받을 수 있게 하려면?

1. 데이터 소스의 일반화
2. 메타 데이터(Schema) 관리
3. 메타 데이터를 활용한 데이터 처리 자동화

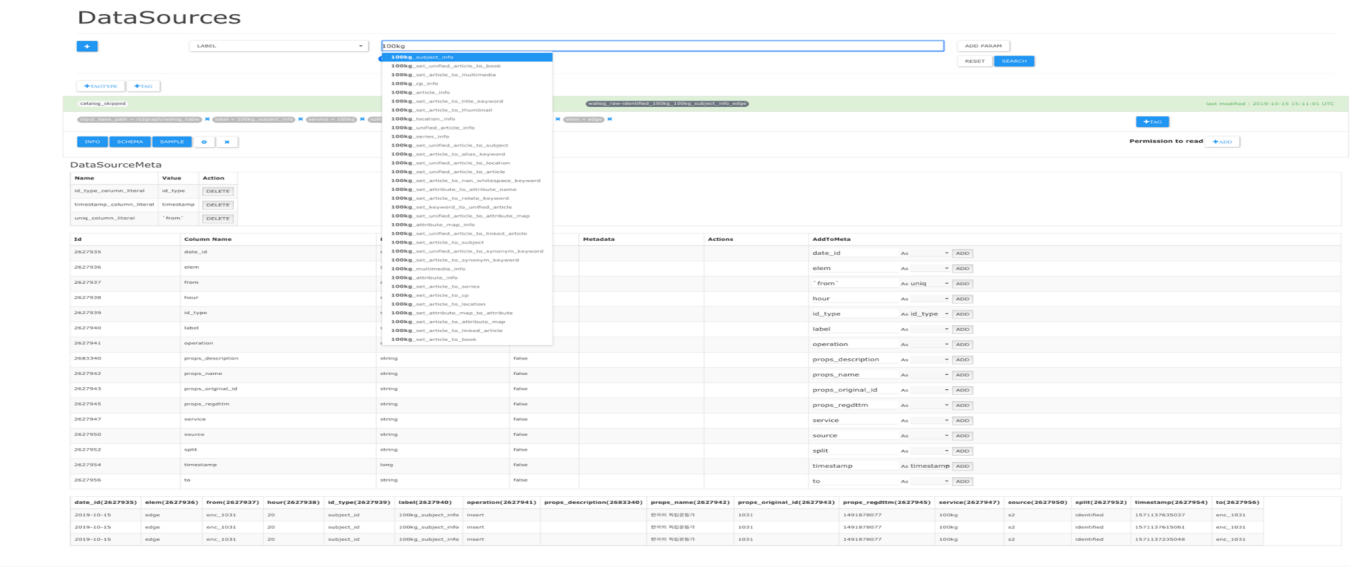
데이터 레이크(Before)

1. 결제 관련해서 어떤 데이터가, 어떤 형태로, 어디에 관리 되나요?
2. 데이터를 가져가서 새로운 프로젝트에 활용해 보고 싶은데 권한은 어떻게 신청 하나요?

데이터 레이크(After): 메타 데이터 서비스

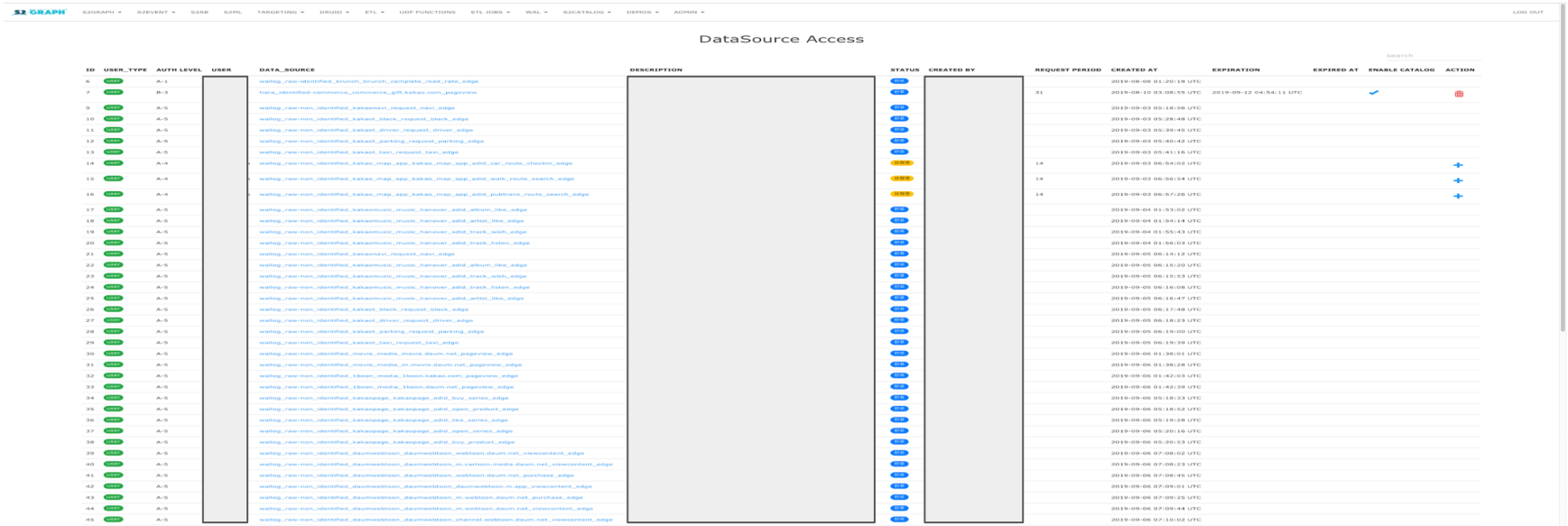
레이크에 추가 되는 데이터들에 대해 자동으로

- Schema + Sample 생성 -> Structured Data -> Hive / Spark Catalog
- 메타 데이터를 검색 인덱스에 추가.
- 목표: ‘Select * from xxx limit 10’, ‘describe xxx’ 더이상 하지 맙시다.



데이터 레이크(After): Governance

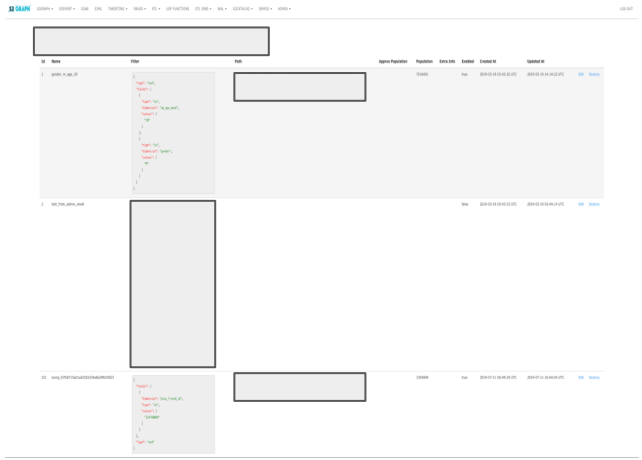
데이터가 필요한 사람들은 필요한 데이터를 찾아보고, 권한신청
- predefined 몇개의 권한으로 관리
- 데이터 소스가 변경 되거나, 장애가 생겼을 때 누가 어디에 활용하고 있는지 파악

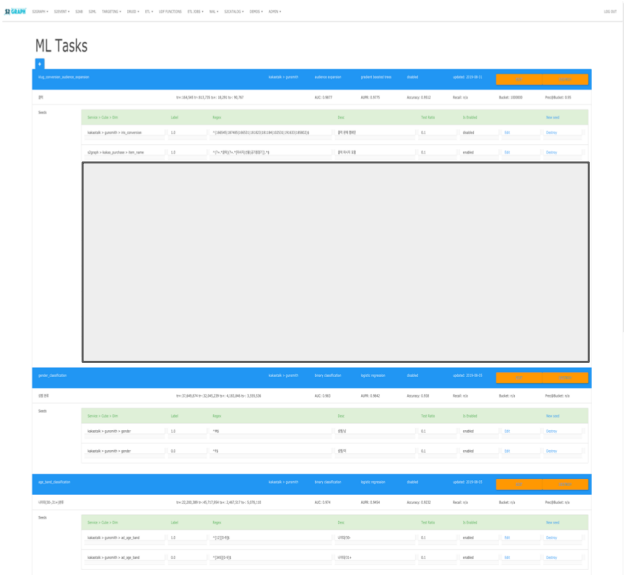


데이터 레이크(After): 활용

중요한 feature/label 데이터를 한 곳에서 생성 + 관리 => 활용하고자 하는 Client에게 일원화 된 스펙으로 제공 가능

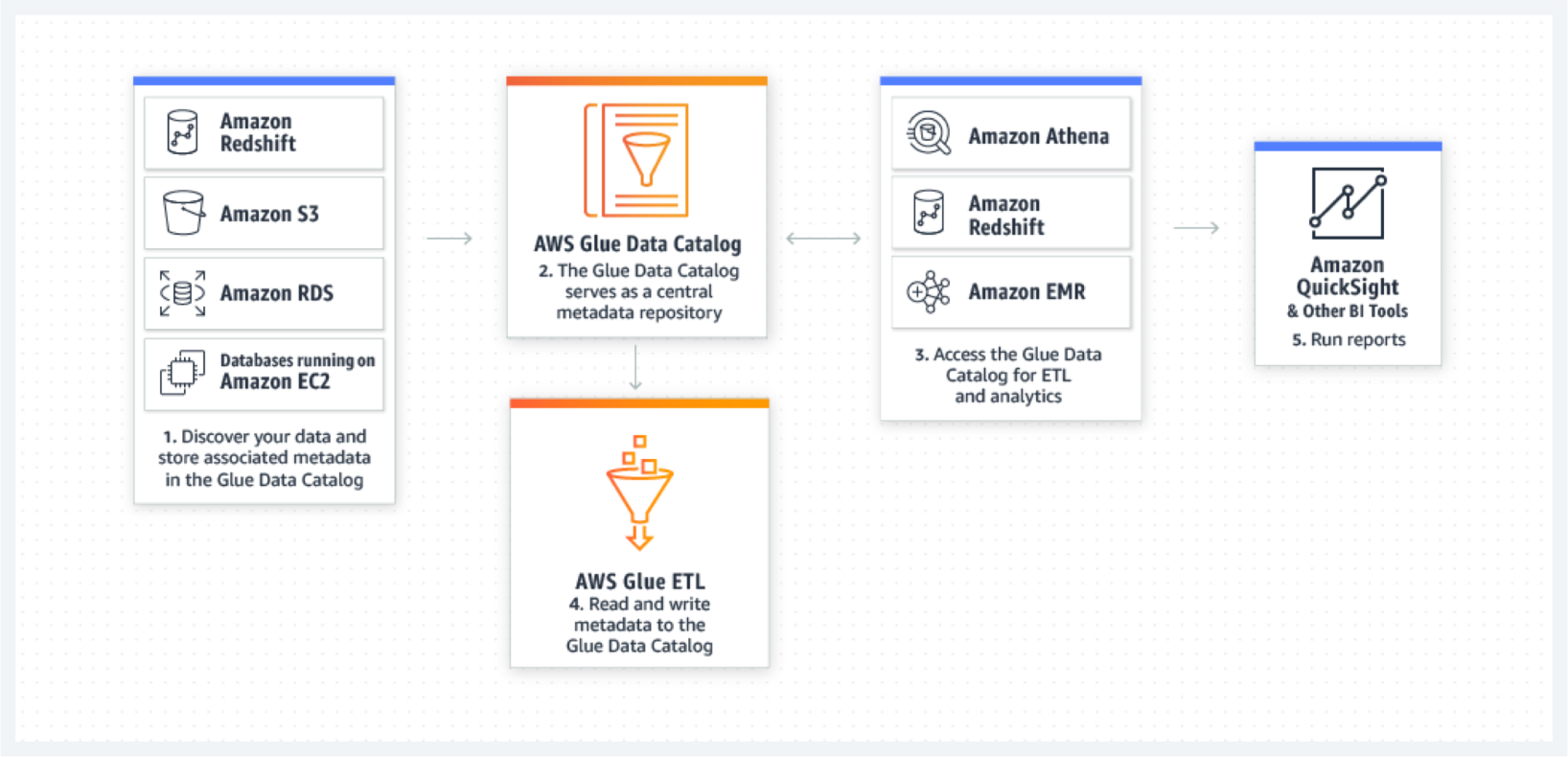
- DMP: 특정 조건을 만족하는 사용자 수와 리스트 생성
- ML: Label 데이터를 선정하여 쉽게 최적화된 prediction 결과들을 생성





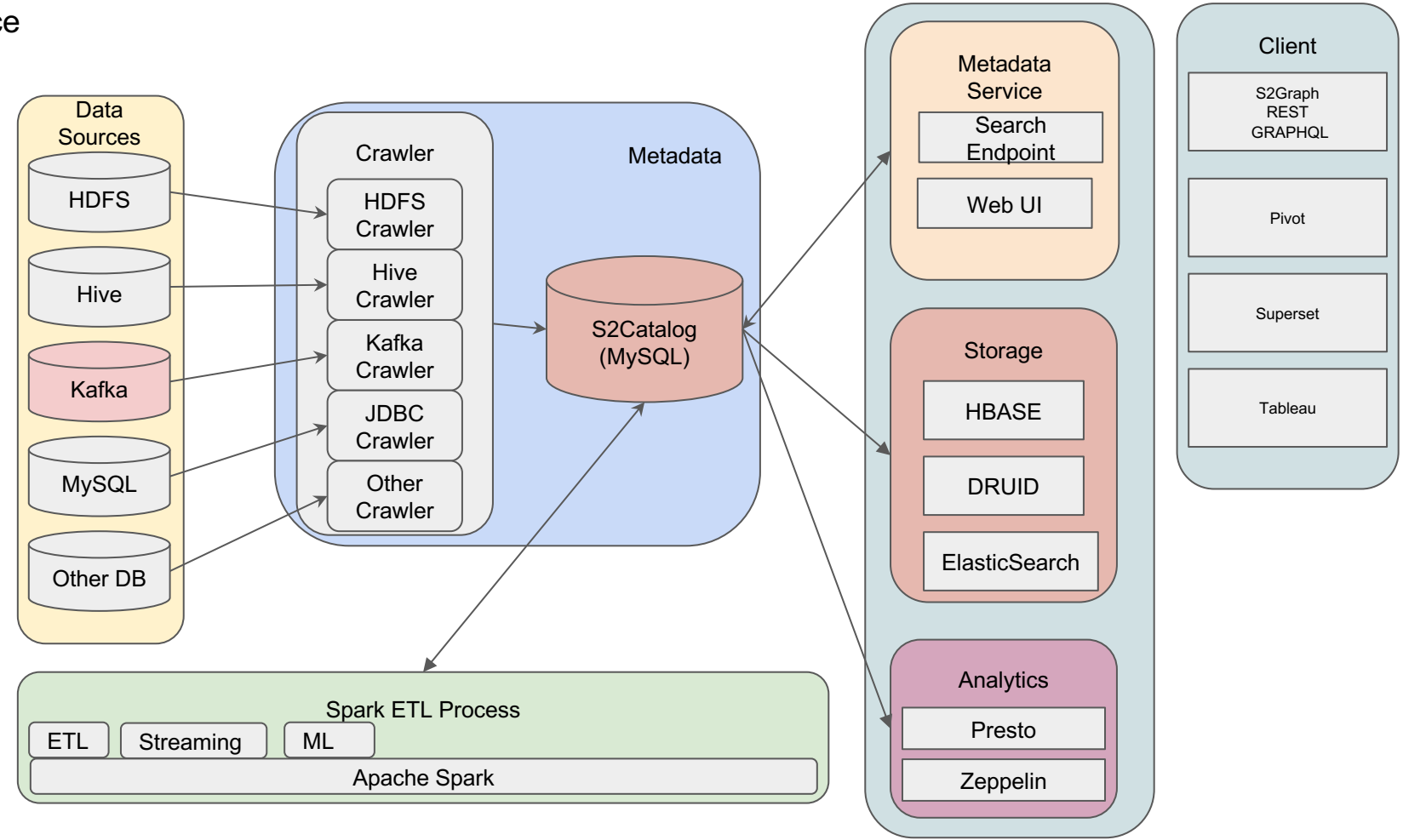
시스템 구성

AWS



Reference: https://aws.amazon.com/glue/

Open Source


Data Sources
HDFS
Hive
Kafka
MySQL
Other DB
Crawler
HDFS Crawler
Hive Crawler
Kafka Crawler
JDBC Crawler
Other Crawler
Metadata
S2Catalog (MySQL)
Spark ETL Process
ETL
Streaming
ML
Apache Spark
Metadata Service
Search Endpoint
Web UI
Storage
HBASE
DRUID
ElasticSearch
Analytics
Presto
Zeppelin
Client
S2Graph REST GRAPHQL
Pivot
Superset
Tableau

핵심 컴포넌트

다 중요하지만 일단 데이터 레이크를 만들기 위해서는

1. Apache Kafka: streaming data source.

- 안정성, 성능이 검증된, 데이터 레이크에서 ingest layer를 담당.
- 가능하면 publish 시점 부터 Schema를 registry 하는게 유리.

핵심 컴포넌트

2. S2 Catalog: (AWS Glue): Metadata

- Apache Spark의 DataFrame: Schema discrepancies.
  - Two pass: InferSchema -> Load the data.
  - 같은 field가 record별로 type이 다르면 -> as string or corrupted record.

- Incremental하게 쌓이는 DataFrame에 field가 추가 되거나, 특정 시점에는 field가 없어 지기도 함.
  - 12시 00분: {“k1”: “abc”, “k2”: true}
  - 12시 05분: {“k2”: true, “k3”: “100”}
  - 12시 10분: {“k1”: [“def”, “ghi”], “k2”: false, “k3”: 200}
  - incremental하게 현재의 DataFrame과 catalog에 schema 정보 merge한 후 다시 catalog update

- Incremental하게 데이터 처리시 bookeeping
  - 이전 작업의 File의 modification 시간, DB record의 id, timestamp등을 관리.
  - 다시 작업을 실행 해도 자동으로 어떤 데이터들을 처리해야 할지 판단.

핵심 컴포넌트

2. S2 Catalog: (AWS Glue): Metadata(cont)

- 참고한 reference들
  - https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-crawler-pyspark-extensions-dynamic-frame.html
  - https://github.com/lyft/amundsenfrontendlibrary
  - https://github.com/linkedin/WhereHows
- 오픈소스 화 진행 중