데이터 분산처리 끝내기

- Part1 데이터 엔지니어링과 분산처리 기본 개념

이 수업에서 다루고자 하는 것

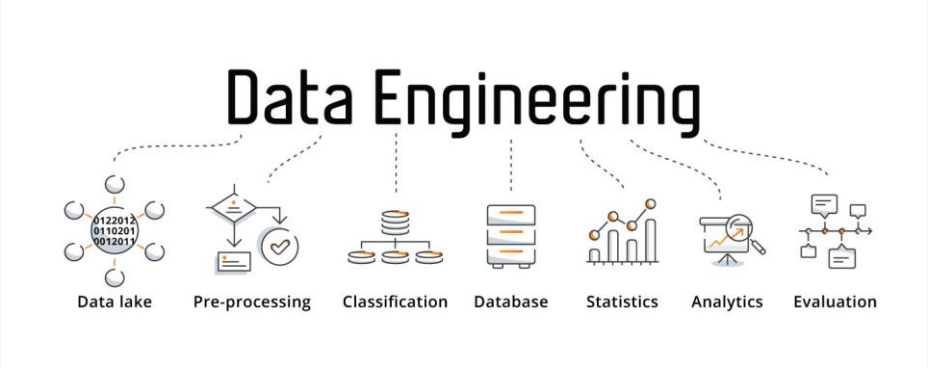
이론

- 데이터 엔지니어링의 이해
- 데이터 엔지니어가 하는 일
- 분산 시스템에 대한 이해1,2
- 데이터 파이프라인의 종류
- Data Observability

데이터 엔지니어링과 분산처리 기본 개념

데이터 엔지니어링의 이해

01. 데이터 엔지니어링의 이해



Data Engineering in Wikepedia

- 데이터 엔지니어링은 데이터를 **수집**하고 **사용**할 수 있도록 시스템 을 **구축**하는 것을 말합니다 .
- 이 데이터는 일반적으로 **후속 분석** 및 **데이터 과학** 을 가능하게 하는 데 사용됩니다 . 데이터를 사용 가능하게 만드는 것은 일반적으로 상당한 컴퓨팅 자원이 필요하고 이에 따른 데이터 처리 및 정리를 포함 합니다.

=> 수집, 사용할 수 있는 시스템 구축, 분석과 데이터 사이언스, 데이터 처리

01. 데이터 엔지니어링의 이해

수집

- 다양한 서비스에서 생성된 데이터들을 모으는 일
- 데이터 파이프라인
- 배치 또는 스트리밍

시스템 구축

- 파이프라인을 어떻게 구성할 것인가?
- 적절한 기술의 선택, 의사 결정
- 배치 처리, 스트리밍 처리

DATA ENGINEERING

01. 데이터 엔지니어링의 이해

분석과 데이터 사이언스

- 쿼리를 통한 분석, 분석 시스템
- 어떻게 전달할 것인가 – Visualization, Dashboard
- Feature Engineering

데이터 처리

- 모델을 위한 데이터 전처리
- 생성일자, 업데이트 일자 등등의 Timestamp처리

DATA ENGINEERING

01. 데이터 엔지니어링의 이해

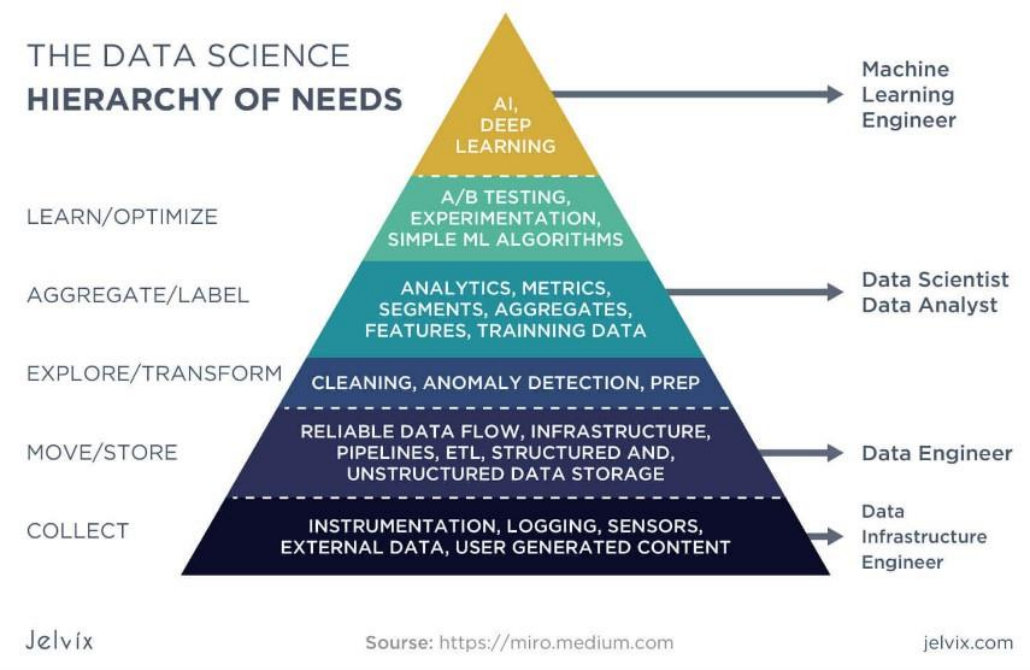
10 Most In-Demand Jobs in AI, ML, and Big Data

	2022	2021
1	Data Engineer	Data Engineer
2	Data Analyst	Data Scientist
3	Data Scientist	Data Analyst
4	Machine Learning Engineer	Machine Learning Engineer
5	Data Architect	Big Data Engineer
6	Staff Data Engineer	Principal Data Scientist
7	Data Engineering Manager	Staff Data Engineer
8	Lead Data Scientist	Marketing Data Analyst
9	Data Science Manager	Data Engineering Manager
10	Lead Data Engineer	Lead Data Scientist

데이터 엔지니어 전망

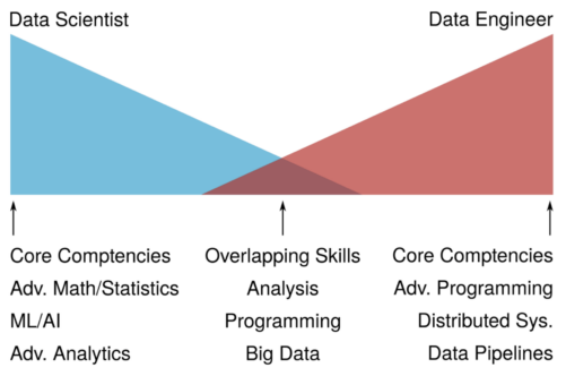
- 21년에 이어 22년에도 1위를 차지함
- Data직군들이 많아지고 있음에도 불구하고 수요가 계속 증가함

01. 데이터 엔지니어링의 이해



데이터 사이언스의 기반이 되는 직무

01. 데이터 엔지니어링의 이해



https://www.oreilly.com/radar/data-engineers-vs-data-scientists/

Data Engineer와 Data Scientist

- Machine Learning 모델을 주 서비스로 하는 경우 업무가 중첩되는 경우가 발생함
- Data Engineer는 보통 데이터 파이프라인 설계, 분산 시스템에 대한 부분을 주로 다루게 됨
- ML에 관련된 부분을 데이터 엔지니어가 맡게 되는 경우
- 다양한 데이터 엔지니어 파생 직군들이 발생하게 됨

데이터 엔지니어가 하는 일

02. 데이터 엔지니어가 하는 일

회사마다 데이터 엔지니어를 정의하는 것에는 차이가 있음

- 데이터 파이프라인 설계, 구축, 운영
- Kubernetes 등 인프라 관리
- 클라우드 인프라 관리
- 네트워크 관리
- 데이터 웨어하우스 구축, 관리
- 대시 보드 생성, 운영
- Feature Engineering
- 모델 서빙
- GPU 등 리소스 관리
- 데이터 마트 및 View 제공
- ML 플랫폼 구축
- 다양한 부서와의 협업

02. 데이터 엔지니어가 하는 일

데이터 엔지니어

- 데이터 파이프라인 설계, 구축, 운영
- 인프라 관리
- 데이터 가공
- 데이터 웨어하우스 구축, 운영
- 데이터 연동
- ETL/ELT 처리
- 데이터 분석가, ML업무 지원
- 장애 대응 Monitoring, Alert

02. 데이터 엔지니어가 하는 일

Analytics Engineer

- 최종사용자를 위한 데이터 가공
- 분석과 비즈니스를 위한 데이터 설계
- 데이터 마트 사용 교육
- 데이터 정제
- 데이터에 대한 문서
- 툴을 통한 데이터 시각화, 이용자 훈련

02. 데이터 엔지니어가 하는 일

MLOps Engineer

- ML서비스를 위한 인프라 관리
- ML 플랫폼 구축
- GPU 등 리소스 관리
- 모델 성능 최적화
- CI / CD / CT (Continuous Integration, Delivery, Training)
- 학습 자동화
- 서빙 자동화
- 서빙 및 학습 최적화

계속 변화하고 있는 데이터 엔지니어

- 기본적인 개념이나 베이스는 크게 변화가 없을 것
- 분산처리나 기타 개념들을 잘 잡아놓는 것이 중요
- 각 특징들에 대한 이해
- 활용을 어떻게 할 것인가?
- 어떤 변화가 있는지 지속적인 확인

  Ex) Airflow의 버전

  Ex) Kubernetes 버전

새로운 툴, 더 좋은 툴에 대한 확인

Major Features in Airflow 2.0

Airflow 2.0 includes hundreds of features and bug fixes both large and small. Many of the significant improvements were influenced and inspired by feedback from Airflow's 2019 Community Survey, which garnered over 300 responses.

A New Scheduler: Low-Latency + High-Availability

The Airflow Scheduler as a core component has been key to the growth and success of the project since its creation in 2014. As Airflow matured and the number of users running hundreds of thousands of tasks grew,

Before, with **Airflow 1.10**:	**After**, with **Airflow 2.0:**
Can't use Subdags for a clean UX	✔ **Task groups replace Subdags**
API isn't officially supported	✔ **Full API support for triggering DAG**
Uses community based and custom operator	✔ **Independently versioned community operators**
Scheduler performance causes data downtime	✔ **HA Scheduler with 20x performance improvements**
UI is slow and unstable because the Web Server needs to parse DAG files	✔ **Stateless UI with serialized DAGs in webserver**
Can only pass limited data between tasks	✔ **Uses can use an external store (S3) as an XCOM backend**
Need to upgrade entire Airflow environment to upgrade community supported operators	✔ **Can independently version operators without upgrading Airflow**

계속 변화하고 있는 데이터 엔지니어

- 기본적인 개념이나 베이스는 크게 변화가
- 분산처리나 기타 개념들을 잘 잡아놓는 것
- 각 특징들에 대한 이해
- 활용을 어떻게 할 것인가?
- 어떤 변화가 있는지 지속적인 확인
  Ex) Airflow의 버전
  Ex) Kubernetes 버전

새로운 툴, 더 좋은 툴에 대한 확인

Amazon EKS Kubernetes release calendar

Note
Dates with only a month and a year are approximate and are updated with an exact date when it's known.

Kubernetes version	Upstream release	Amazon EKS release	Amazon EKS end of support
1.25	August 23, 2022	March 2023	May 2024
1.24	May 3, 2022	November 15, 2022	January 2024
1.23	December 7, 2021	August 11, 2022	October 2023
1.22	August 4, 2021	April 4, 2022	May 2023
1.21	April 8, 2021	July 19, 2021	February 15, 2023
1.20	December 8, 2020	May 18, 2021	November 1, 2022
1.19	August 26, 2020	February 16, 2021	August 1, 2022

Dependency	Validated or Included Version(s)	Notes
Kubernetes	1.21	*Kubernetes 1.22 is NOT supported by Kubeflow 1.5, see kubeflow/kubeflow#6353 for more information.*

분산 시스템에 대한 이해1

03-1. 분산 시스템에 대한 이해

분산시스템이란?

- 분산되어 있는 여러 서버를 하나의 서버처럼 다루는 시스템

- 왜 분산 시스템이 필요한가?
- **고 가용성 :** 서버와 네트워크, 프로그램 등의 정보 시스템이 상당히 오랜 기간 동안 지속적으로 정상 운영이 가능한 성질 (wikipedia)

- **한 머신(서버)는 장애가 날 것이라고 가정. 장애가 없는 서버는 없다**
- 장애에 대한 대응 시스템 - 장애를 방어할 수 있는 시스템이 바로 분산 시스템
- 한 서버가 고장 나면 다른 서버에서 처리할 수 있도록

- 대표적인 방식이 **복제**

03-1. 분산 시스템에 대한 이해

복제의 용도

- 고가용성

  한 서버(또는 여러 장비나 전체 데이터센터가 다운될 때도 시스템이 계속 동작하게 한다.

- 지연 시간

  지리적으로 사용자에게 가까이 데이터를 배치해 사용자가 더 빠르게 작업할 수 있게 한다.

- 확장성

  복제본에서 읽기를 수행해 단일 장비에서 다룰 수 있는 양보다 많은 양의 읽기 작업을 처리할 수 있다.

03-1. 분산 시스템에 대한 이해

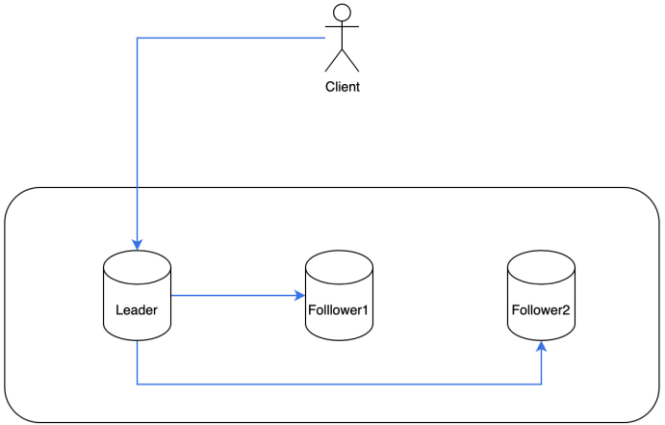
복제

- 복제란 네트워크로 연결된 여러 장비에 동일한 데이터의 복사본을 유지한다는 의미
- **리더 팔로워 방식**을 사용
  복제 서버 중 한대를 리더로 사용하고 나머지 서버를 팔로워 서버로 사용한다

- 복제된 서버에도 같은 데이터가 있어야 같은 처리를 할 수 있다
- 클라이언트가 쓰기를 원할 때는 리더에게 요청을 해야한다
- 요청을 받은 리더가 다른 팔로워들에게 동일한 요청을 보낸다

- 만약에 리더 서버에 장애가 난다면 팔로워 중 하나를 리더로 선출한다
- 새로 선출된 리더가 이전처럼 클라이언트의 요청을 처리한다

- 읽을 때는 리더 혹은 팔로워에게 질의 가능, 쓰기는 only 리더만

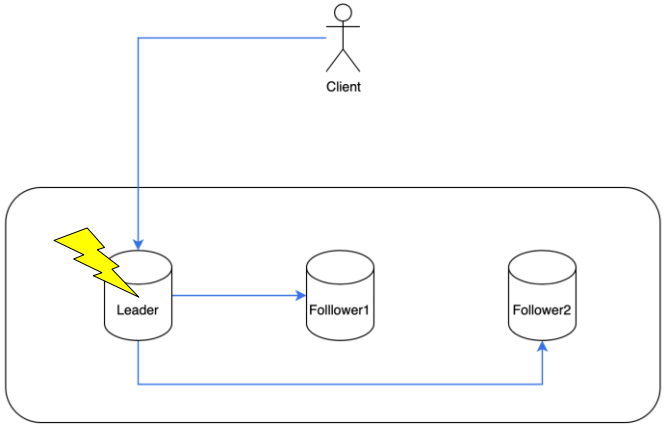


03-1. 분산 시스템에 대한 이해

리더의 장애상황, Fail Over 과정

- 팔로워 중 하나를 리더를 승격해야 한다
- 클라이언트에도 새 리더로 쓰기를 요청하기 위한 재처리가 필요
- 다른 팔로워들은 새 리더로부터 달라진 데이터를 읽어야 한다

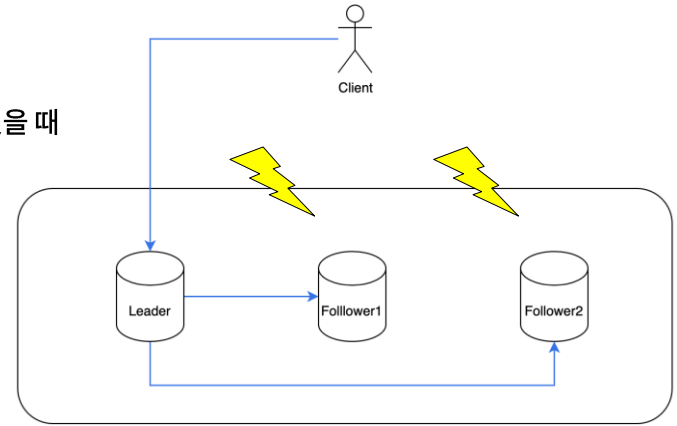
1. 리더가 장애 상황인지 확인한다 (health check)
   대부분 타임아웃을 사용한다
2. 선출과정을 통해 새로운 리더를 선택한다 (적합한 후보를 골라놓는다)
   가장 적합한 후보는 가장 최신의 데이터를 갖고 있는 팔로워
3. 새로운 리더를 위한 시스템을 설정한다
   이전 리더가 팔로워가 되었다는 것을 알린다



03-1. 분산 시스템에 대한 이해

팔로워의 장애상황, 따라잡기

- 팔로워가 죽더라도 리더를 통해 매우 쉽게 복구할 수 있다.
- 마지막 트랜잭션을 확인
- 리더에 연결하고 연결이 끊어진 동안의 데이터 변경을 모두 요청할 수 있음
- 스레드 읽는 처리량은 고정이라 큰 장애는 없음
- 변경된 데이터의 보관 기간이 넘어가면 로그가 사라져서 연결 안 될 수 있다
  하둡의 SNN 장애의 경우, Airflow DAG가 실패한지 오래됐다가 한번에 실행됐을 때
- 리더의 데이터 상황을 다 따라잡으면 장애 상황 종료



03-1. 분산 시스템에 대한 이해

* 변경 데이터란?

- MySQL에서는 Binary Log
- PosgtreSQL에서는 Log Sequence Number

- CREATE, DROP, ALTER와 같은 DDL과 DML을 통해 DB에 변경사항이 생길 때 그 변화된 이벤트를 기록
- SELECT 등 조회 문법은 제외됨
- 리더의 복제 로그의 정확한 위치를 알려주는 파일
- 서버 내에서 발생되는 모든 변경내역이 기록되는 파일

```
18
19 SHOW BINARY LOGS;
```

Query Favorites Query History

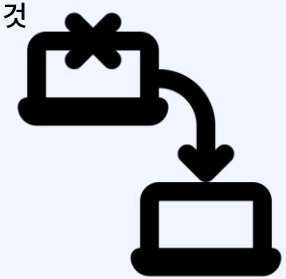
Log_name	File_size
mysql-bin-changelog.011677	46437018
mysql-bin-changelog.011678	48331067
mysql-bin-changelog.011679	44924243
mysql-bin-changelog.011680	47819694
mysql-bin-changelog.011681	45790728
mysql-bin-changelog.011682	47919160
mysql-bin-changelog.011683	47404641

https://medium.com/@mahadir.ahmad/reading-mysql-binlog-with-python-c85a8ece3b28

03-1. 분산 시스템에 대한 이해

분산 시스템의 문제점

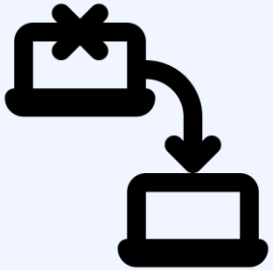
- 분산 시스템은 완전하지 않다
- 결함이 발견됐을 때 시스템이 이를 견딜 수 있게 만들기도 쉽지 않다
- 한 노드에서 다른 노드로 정보가 흐를 수 있는 유일한 방법은 신뢰성이 없는 네트워크로 보내는 것
- 시스템이 커질 수록 구성요소 하나가 고장 날 가능성이 높아짐
- 결함이 생겼다면, 잘못된 결과를 내라? 아니 차라리 완전히 동작하지 마라!
- 예측할 수 없는 방식으로 고장나는 것 ⇒ **부분 장애**
- 어떨 때는 동작하지만 어떨 때는 예측되지 않는 방식으로 실패 => **비 결정성**



03-1. 분산 시스템에 대한 이해

해결법 맛보기

- 결함이 있더라도 서비스는 올바르게 동작하게 만들기
- 부분에 장애가 발생하면 전체를 Kill (운영에는 적용되기 힘들 수 있다)
- 보장을 해주는 범용적인 추상화를 찾아 구현한다
  애플리케이션단에서 이 보장에 의존하게 한다 (ex, 데이터베이스의 트랜잭션)
- 신뢰성 없는 구성 요소를 사용해 신뢰성 있는 시스템 구축하기 (완벽한 신뢰성은 없다)
- 그 외 선형성 보장, 순서화 보장 등등… 너무 깊은 내용
- (Zookeeper)

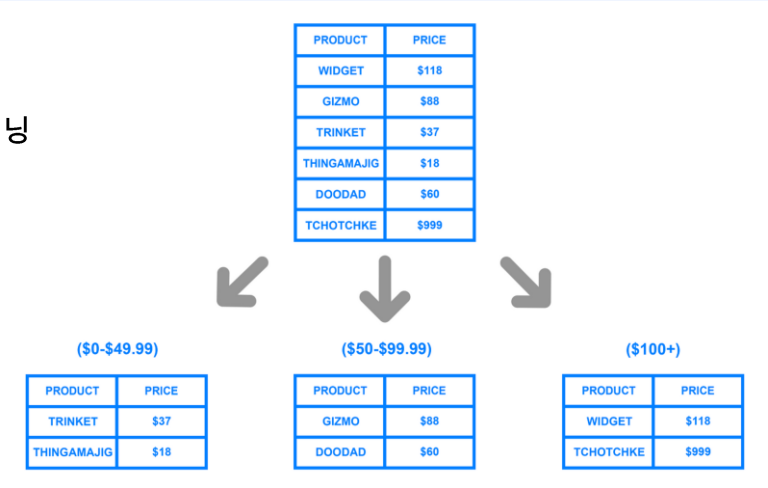


분산 시스템에 대한 이해2

파티셔닝

- 파티셔닝과 샤딩 사실 상 같은 용어
- 파티션은 MongoDB, ElasticSearch, SolR의 샤드와 같다
- 데이터셋이 매우 클 때 또는 쿼리 처리량이 매우 높을 때 데이터를 파티셔닝
- 데이터 파티셔닝을 하는 주된 이유는 확장성
- 대용량 데이터셋이 여러 디스크에 분산될 수 있고,
  쿼리에 대한 부하를 여러 프로세서에 분산시킬 수 있게된다

- 노드 사이에 쿼리 부하를 고르게 분산시키는 것이 중요
- 고르게 분산이 되지 않아 한쪽으로 쏠린 것을 **Skewed** 되었다고 부름
- 불균형하게 높은 쿼리 부하가 걸리는 파티션을 **핫스팟**이라고 한다
  노드를 무작위로 고르면 해소가 되지만, 어느 노드에 저장됐을지 알 수 없으므로
  모든 노드에 병렬적으로 쿼리를 실행한다



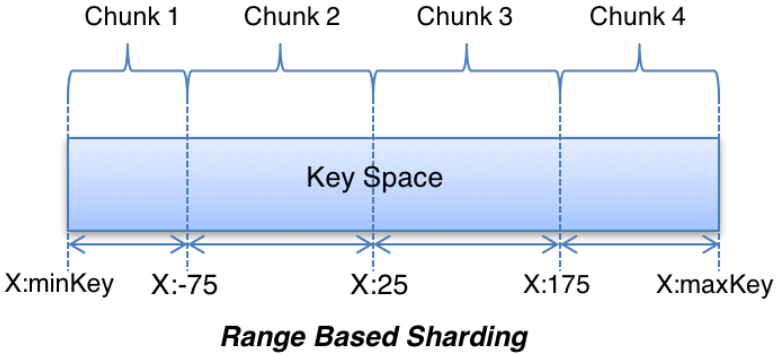
분리하기 용이한 기준을 잡고 데이터를 분산저장하거나
Row의 행수 등을 기준으로 분산 저장

Key Value 데이터 파티셔닝

- 키 범위 파티셔닝, 연속된 범위(최소값 ~ 최대값)의 키를 할당
- 데이터셋이 매우 클 때 또는 쿼리 처리량이 매우 높을 때 데이터를 파티셔닝
- 파티션 경계는 수동으로 선택 OR 데이터베이스에서 자동으로 선택
- 각 파티션 내에서는 키를 정렬된 순서로 저장할 수 있다

  \+ : 범위 스캔이 쉬워짐

  \- : 자주 사용하는 특정한 접근 패턴이 핫스팟을 유발

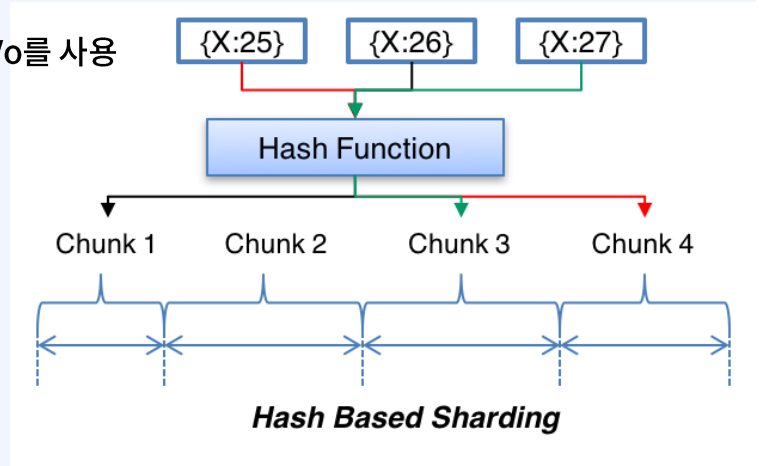


Range Based Sharding

https://www.oreilly.com/library/view/architecting-data-intensive-applications/9781786465092/132886d4-3507-4cb4-97fe-c3661cdffc3f.xhtml

Key Value 데이터 파티셔닝

- 키의 해시값 기준 파티셔닝, 쏠림과 핫스팟의 위험 때문
- Cassandra와 MongoDB는 MD5, Voldemort는 Fowler-Noll-Vo를 사용
- 키에 적합한 해시 함수를 구하고 각 파티션에 해시값 범위를 할당한다
- 해시값이 파티션의 범위에 속하는 모든 키를 그 파티션에 할당한다



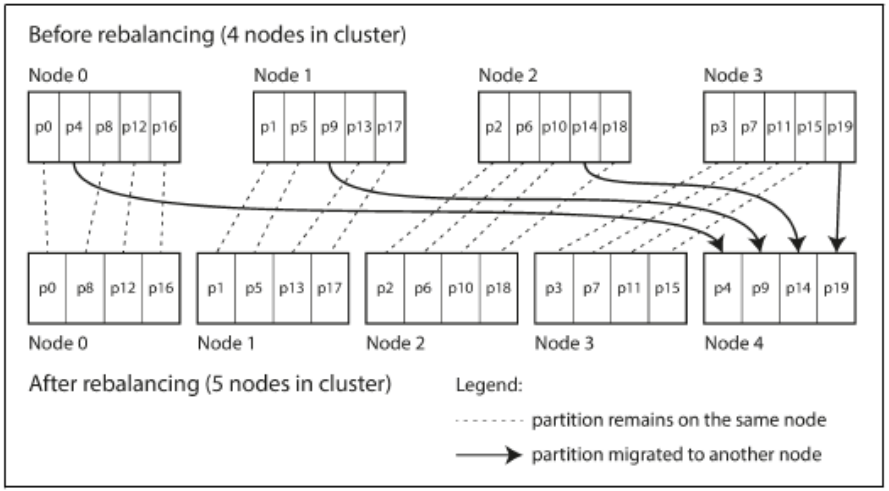
Hash Based Sharding

- 키를 파티션 사이에 균일하게 분산시키는 데 좋음
- 파티션 경계는 크기가 동일하도록 나누거나 Random하게 선택가능
- 해당 기법을 **일관성 해싱**이라고 함

- 키 범위 파티셔닝의 좋은 속성을 잃어 버리는 단점이 있음
  범위 질의를 효율적으로 실행할 수 없고 모든 파티션의 정렬 순서가 유지 되지 않음

https://www.oreilly.com/library/view/architecting-data-intensive-applications/9781786465092/97f65e4f-7b95-467f-b347-6f88c730a93b.xhtml

03-2. 분산 시스템에 대한 이해



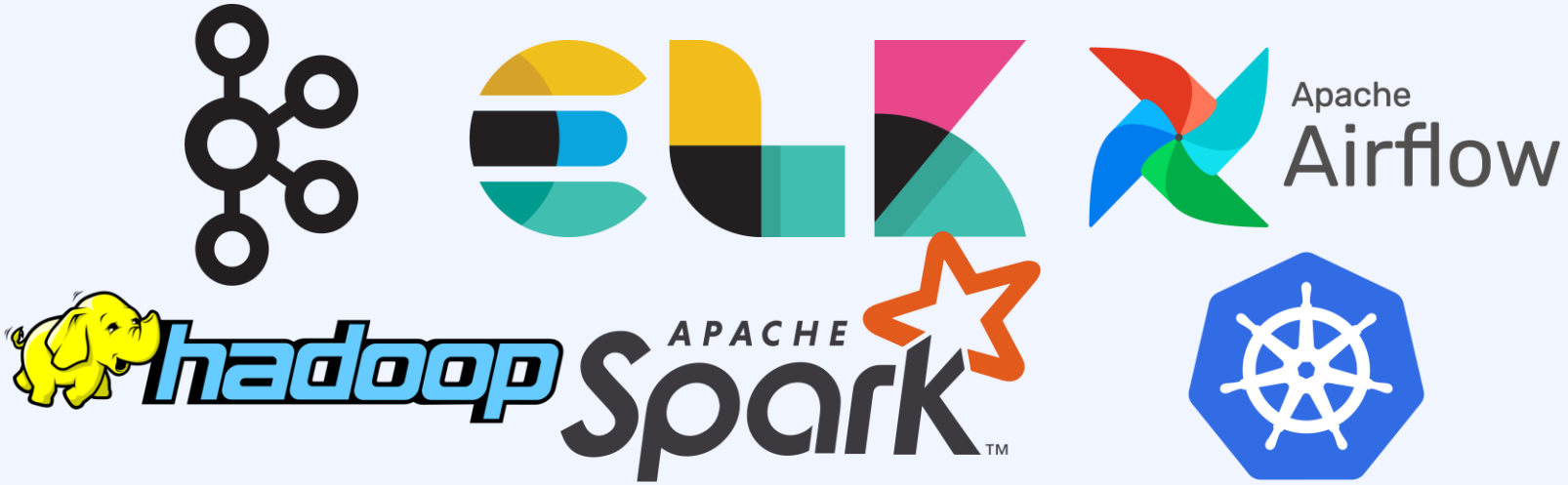
data-intensive-application

파티션 리밸런싱

- 추가 리소스가 필요해서 노드를 추가할 경우,
  클러스터에서 한 노드가 담당하던 부하를 다른 노드를 옮기는 과정이 **Rebalancing**

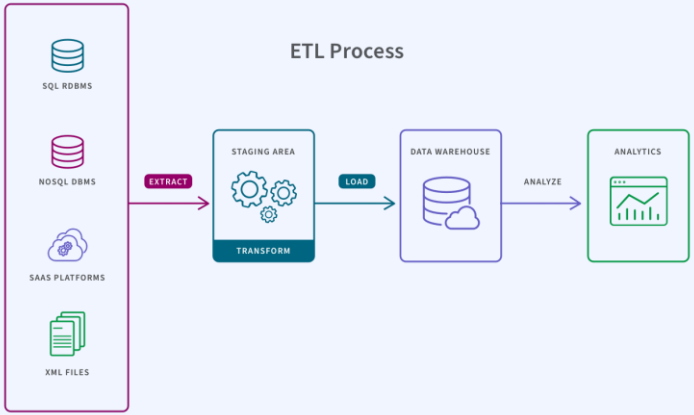
03-2. 분산 시스템에 대한 이해

분산 시스템을 활용한 도구들



데이터 파이프라인의 종류

04. 데이터 파이프라인의 종류



https://www.qlik.com/us/data-integration/data-pipeline

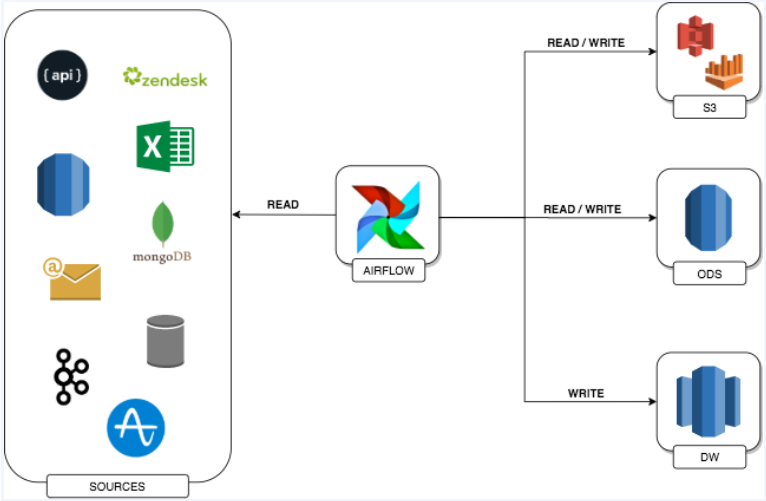
데이터 파이프라인이란?

- 다양한 데이터 소스에서 RAW 데이터를 수집한 후 분석을 위해 Datalake 또는 Data warehouse 같은 데이터 저장소로 이전하는 방법
- 연결되는 구성이 파이프라인 같다고 해서 데이터 파이프라인이라고 부름

- 서로 다른 여러 소스 시스템에서 데이터를 추출하고 해당 데이터를 변환, 결합 및 검증하고 대상 리포지토리에 로드하는 프로세스를 자동화

- 배치 파이프라인과 스트리밍 파이프라인이 존재함

04. 데이터 파이프라인의 종류



https://medium.com/quintoandar-tech-blog/how-apache-airflow-is-helping-us-to-evolve-our-data-pipeline-at-quintoandar-7d157e9f9773

배치 데이터 파이프라인

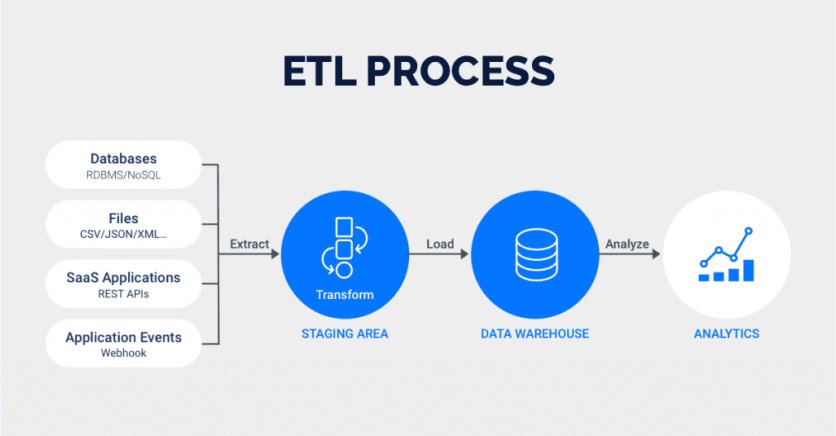
- 배치 데이터란(Bounded Data)? 기간이 정해져 있는 유한한 데이터
  ex) 22년 12월 접속 유저 수
- 배치 처리(일괄 처리): 일정기간(일,월단위) 또는 한정된(bounded) 데이터를 한 시점에 순서적으로 처리하는 방식
- 맵리듀스(MapReduce)기법인 하둡또한 배치처리 방식
- 배치 처리방식은 일정기간의 데이터를 일괄적으로 처리
- 스케줄링이 필수적!
- 대표적으로 Airflow가 배치 데이터 파이프라인에 많이 활용되고 있음
- 데이터를 모으고 읽어서 처리한 후 다시 적재하기 (ETL)

04. 데이터 파이프라인의 종류

* ETL과 ELT의 차이

- ETL (Extract Transform Load)
  Extract : Source Data로부터 추출
  Transform : 추출된 데이터 변형
  Load : DW, Data Lake로의 데이터 적재

- ELT (Extract Load Transform) 왜 순서가 다를까?
- ETL은 Transform에서 시간이 상당히 소요됨
- Big Data, DataLake의 등장
- 데이터의 중요성을 깨닫게 됨, 하드웨어 비용 감소
- 일단 모아놓고 필요할 때 나중에 활용하자
- BI, 분석 팀들이 빠르고 유연하게 움직일 수 있음



https://rivery.io/blog/etl-vs-elt/

04. 데이터 파이프라인의 종류

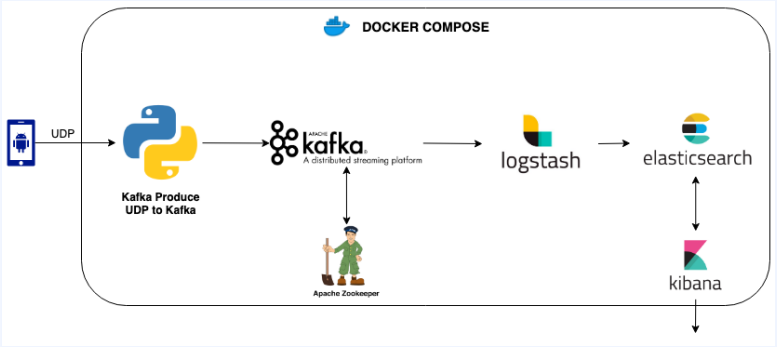
* ETL과 ELT의 차이

- ETL은 시대에 뒤떨어진 걸까?
- ETL은 데이터 웨어하우스가 사전 구성되어있음
- 데이터가 구조화된 다음 변환(T)되는 상황이라면,
  ETL을 통해 빠르고 효율적 안정적으로 데이터를 분석할 수 있음
- ELT는 빠른 분석을 요구하는 작업에는 적합하지 않을 수 있음
- 데이터 크기가 중요함

- 아직도 많이 사용되고 있음
  ex) Nifi로 처리, Jolt

https://technology.amis.nl/wp-content/uploads/2022/04/jolt-json-to-json-transformation.png

04. 데이터 파이프라인의 종류



https://itnext.io/real-time-iot-data-streaming-from-old-smartphone-e9fbee6ecc91

스트리밍 데이터 파이프라인

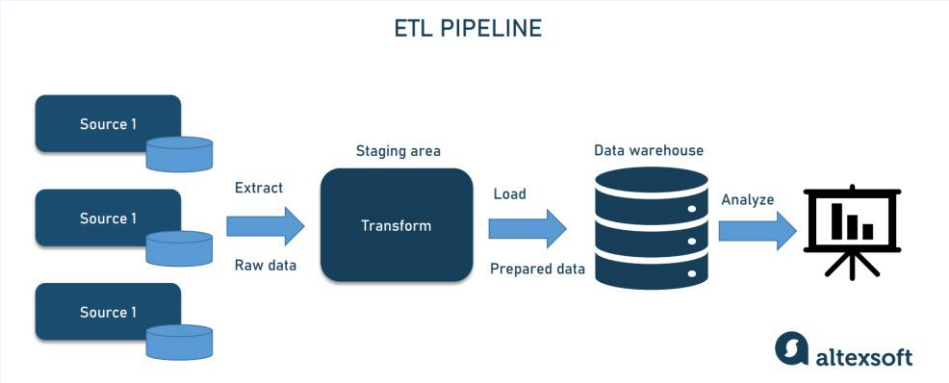
- 스트리밍 데이터(Unbound Data)? 기간 제한이 없는 무한한 데이터
  ex) 실시간 주문 데이터
- 끝없이 흐르는 데이터를 처리하기 위한 파이프라인
- 분산 스토리지에 저장하지 않고 처리를 계속함
- 실시간성이 중요한 경우에 사용이 되는 파이프라인
  ex) 물류 시스템 대시보드, 실시간 예측성 모델
- Near Real Time (Mini Batch) : 데이터를 조금씩 모아서 처리하는 방식
  즉, 배치 프로세싱을 작게 쪼개서 하는 방식 ex) Spark Streaming
- Real Time : 데이터가 들어올 때마다
- 대표적으로 ELK, Flink, Nifi

04. 데이터 파이프라인의 종류

데이터 연동을 위한 파이프라인 (데이터 분석 및 시각화)

- 활용 목적, Transform, 타겟 정의
- ELT OR ETL
- 원본 소스에서 데이터를 추출
- 연동 주기 설정 (5분?, 15분? 하루? 1주일…)
- 타겟 쪽에 데이터 저장
- 데이터 확인

  정합성 검증
- Visualization 또는 데이터 분석 활용

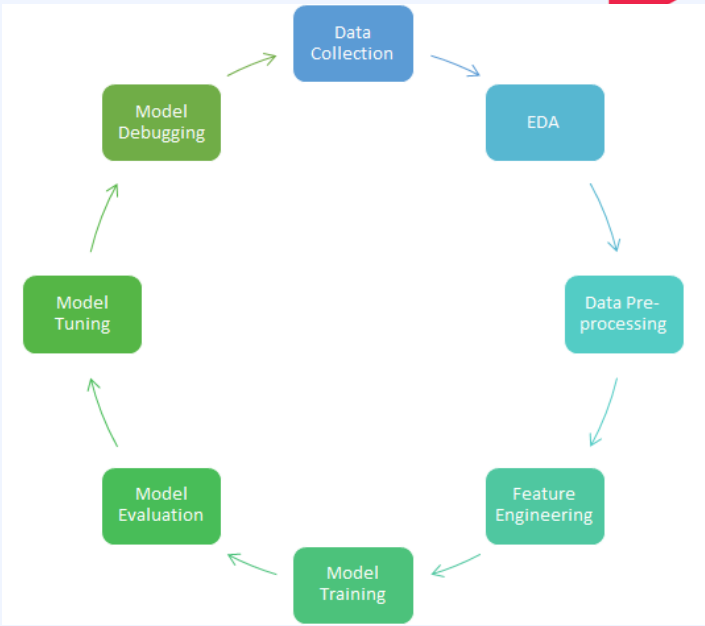


https://www.altexsoft.com/blog/data-pipeline-components-and-types/

04. 데이터 파이프라인의 종류

ML모델을 위한 파이프라인
- 실시간성 모델? 배치성 모델?
- 모델에 따른 파이프라인 설정

1. Feature 정의
2. Feature Engineering
3. 학습 데이터 / 검증 데이터 / 테스트 데이터
4. 모델 생성
5. 모델 배포
6. 모니터링 및 자동화



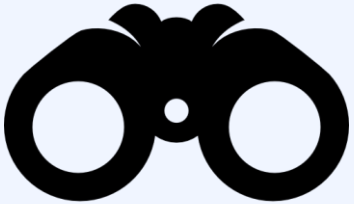
https://medium.com/analytics-vidhya/machine-learning-why-it-is-an-iterative-process-bf709e3b69f2

Data Observability

05. Data Observability

Data Observability?

- Data Observability란 조직 내의 시스템에 있는 데이터 상태를 온전히 알고 있는 것
- 자동화된 모니터링, 알람, 트리거 등을 통해 데이터 품질과 찾을 가능성이 있는 문제를 찾을 수 있다.



Data Observability 5 Pillars (5개의 요소)

- Freshness : 데이터의 최신 업데이트 시간을 확인한다.
- Distribution : 데이터가 허용되는 범위 안에 들어 왔는지 확인한다.
- Volume : 테이블의 데이터 건수가 일정하게 유지되는지를 확인한다.
- Schema : 테이블이 왜 변경되었는지 모니터링 해야한다.
- Lineage : 특정 데이터가 이상하다면 어디서부터 잘못되었는지 알아야한다.
  데이터 마트가 이상하다면 원본 테이블에 이상이 있을 수 있다

05. Data Observability

https://www.montecarlodata.com/blog-what-is-data-observability/

Data Observability vs. Monitoring

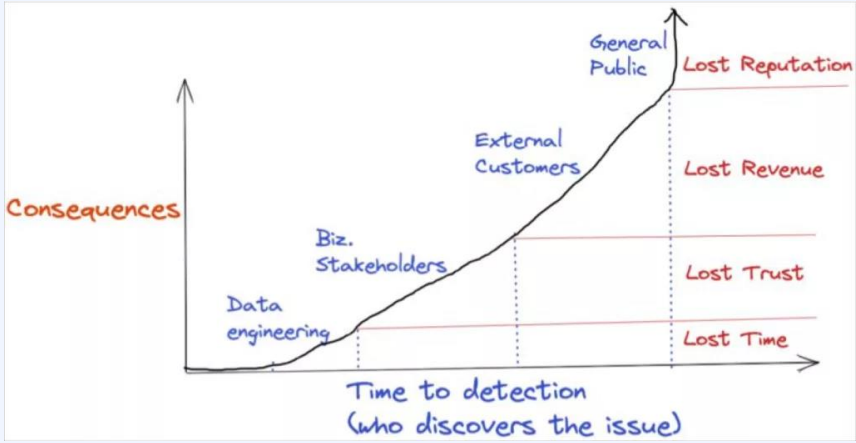
- Data observability는 기존 메트릭과 파라미터가 기존과 다르면 팀에 alert를 한다.
- Data monitoring은 미리 정해놓은 문제의 값에 대해서만 alert를 한다.
  즉, Data monitoring은 Data observability에 비하면 빙산의 일각이다.

05. Data Observability

Data Observability

신뢰할 수 있는 데이터를 만드는 방법

- 데이터 중단 시간을 줄일 수 있음
- 데이터 다운타임의 결과가 심각할 수 있기 때문에 데이터 관찰 가능성이 중요
- 데이터 엔지니어와 개발자에게는 데이터 가동 중지 시간이 곧 시간과 리소스 낭비를 의미
- 데이터 소비자의 경우 의사 결정에 대한 자신감이 약화됨



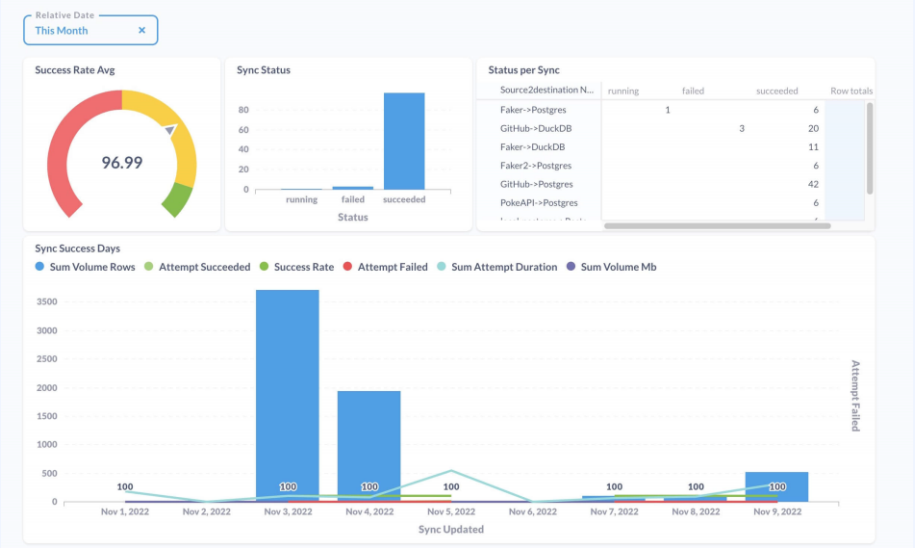
https://www.montecarlodata.com/wp-content/uploads/2022/12/why-data-observability-is-important-1536x792.webp

05. Data Observability

Data Quality

파이프라인 상에서 가장 중요한 이슈일 것

- Missing Data
- Invalid Data
- Noisy Data
- Duplicated Data
- Type Mismatch …
- 좋은 상황에서의 기대되는 값을 갖고 파악
- 모니터링 설정과 Threshold를 걸어놓을 수 있음
- Data Quality Monitoring

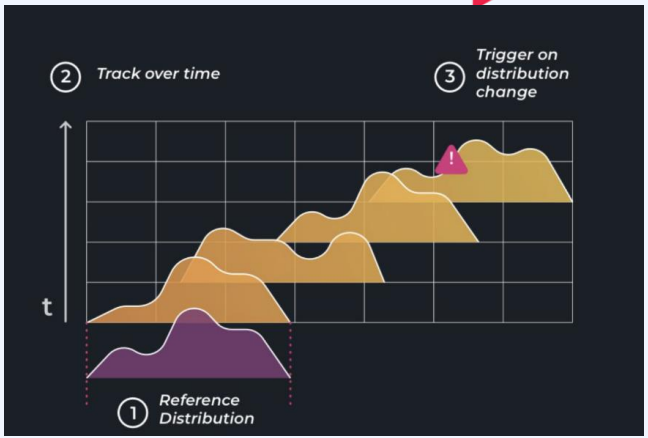


https://airbyte.com/blog/airbyte-monitoring-with-dbt-and-metabase

05. Data Observability

Data Drift

- 시간이 지나면서 점점 변화하는 데이터
- 점점 변화하는 데이터는 이상을 감지하기가 어렵다
- Reference distribution과 현재의 distribution 과 다른 것
- Data Quality보다 한 단계 깊은 내용
- 어떻게 이 Drift를 Quantify 정량화 할 것인지가 어렵다
- Feature Engineering에 특화된 내용?
- 데이터 관리에 필수적인 항목
- 데이터마다의 통계적인 정의가 필요함



https://content.hopsworks.ai/hubfs/Feature Store Summit 2022