On-prem 데이터 플랫폼 설계 고려사항

I. 서론

온프레미스 환경에서 데이터 플랫폼 상의 데이터 파이프라인 아키텍처(Data Pipeline Architecture)는 오픈소스 기반의 서비스를 기반으로 특정 기술이나 국내외 제조사 들에 의존하지 않는 독립적인 구성으로 연구자들이 ML/DL/AI을 위해 기존 사용 중인 서비스와 도구들을 이용하며 적용하는 체계로 구축하여 연구자들의 노력을 최소화 한다.

- 협업과 융합 연구활성화를 위한 플랫폼·인프라 운영체계
  - 다수에 지속적인 서비스 제공 가능한 인프라
- 아래 '그림 1. 인프라 계층과 데이터 과학 프로젝트의 관계'와 같이 데이터 과학 프로젝트는 인프라 서비스 계층에 관련한 분야의 서비스가 필요하다.
  - 실험과 반복적인 작업의 '데이터 사이언스'
  - 통합 연동의 '소프트웨어 아키텍처'
  - 데이터와 컴퓨팅의 '기능적 인프라스트럭처'

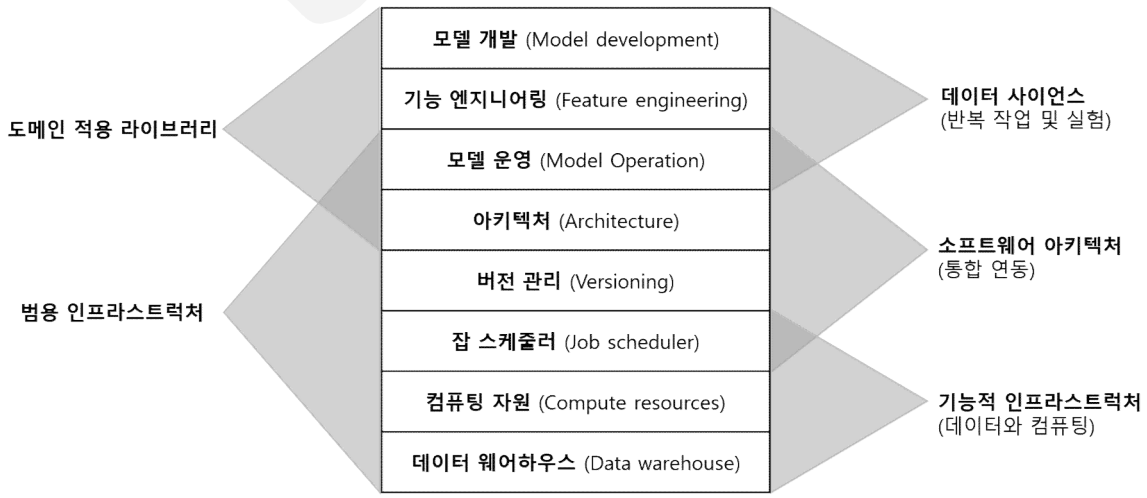

그림 1. 인프라 계층과 데이터 과학 프로젝트의 관계

아래의 '그림 2. 데이터 과학 스택과 데이터 인프라의 관계'와 같이 데이터 플랫폼 상의 데이터와 스토리지는 데이터웨어하우스와 컴퓨팅 자원에 영향을 받으며, 데이터의 전처리를 위한 작업과 버전 관리 등은 소프트웨어 아키텍처의 영향을 받는다.

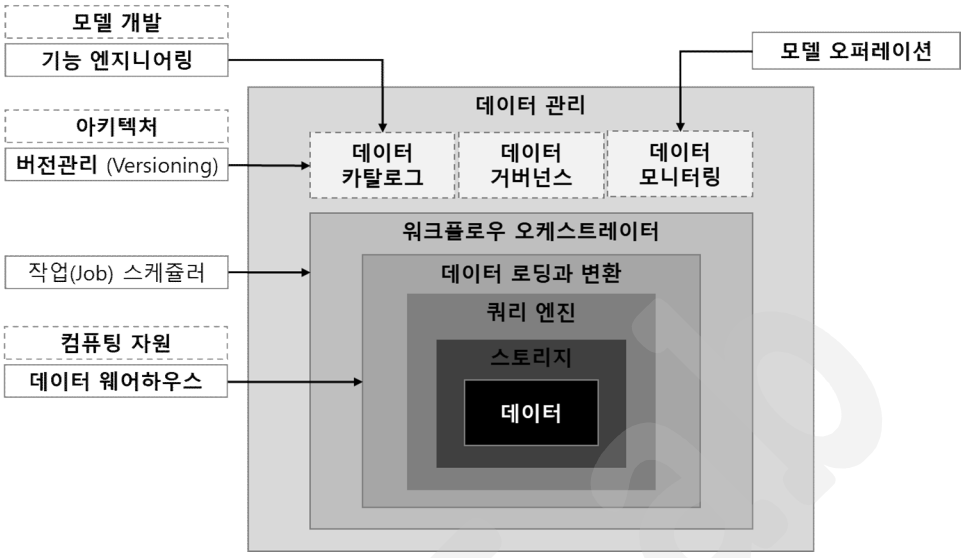

그림 2. 데이터 과학 스택과 데이터 인프라의 관계

데이터 획득부터 분석을 위한 사용까지의 과정은 아래의 '그림 3. AI를 위한 데이터 파이프라인'과 같이 스트리밍 데이터와 배치(Batch) 데이터를 모두 이용 가능하고 융합 처리 구성이 가능하다.

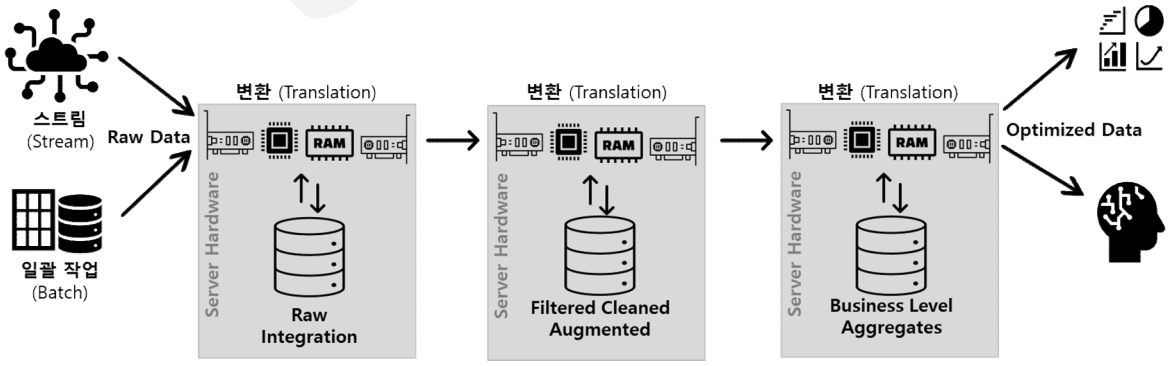

그림 3. AI를 위한 데이터 파이프라인

II. 데이터 플랫폼 인프라 아키텍처

1. 아키텍처 개요

데이터플랫폼 인프라는 서비스를 지원하기 하기 위해 하드웨어와 소프트웨어 등의 주요기술 간에 추상화로 계층 구조를 도입하여 특정 기술이나 제조사에 의존하지 않는 아키텍처 기반의 설계로 클라우드 네이티브 기술의 성능 확장과 오픈소스 기반 기술 도입을 할 수 있다.

- 오픈소스는 물론 상용 서비스 연계 가능한 계층화 인프라스트럭쳐
- 하드웨어와 가상화등 계층별 추상화하여 데이터 플랫폼 서비스의 극대화
- 서비스와 테이터의 인프라 노출의 최소화로 준수(Compliance)를 위한 보안 강화
- 성능 개선과 연결 기관/서비스 호환성을 위한 계층 오프로드(Offload) 지원
- 데이터 플랫폼 서비스는 가상화나 클라우드 네이티브 기반으로 제공하며, 사용자들의 윈도우등 Non-Linux 환경을 위해 두가지 모두 가능한 서비스가 일반적

그림 4. 데이터 파이프라인 가속 인프라 아키텍처 (예)

데이터 과학을 위한 워크플로우 실행을 위한 아키텍처는 서비스 목적에 적절한 구성과 이를 위한 오픈소스 등의 방법을 선택하는 것을 용이하게 하며, 데이터 플랫폼은 구동의 위치를 온프레미스에 베어메탈 하드웨어 기반으로 고려한다.

아래 '그림 5. 워크플로우 실행의 세 가지 고려 사항'과 같이 아키텍처는 워크플로우가 컴퓨팅 자원(Where)과 사용 소프트웨어(How) 등을 고려하여 목적(What)에 필요한 구성을 할 수 있다.

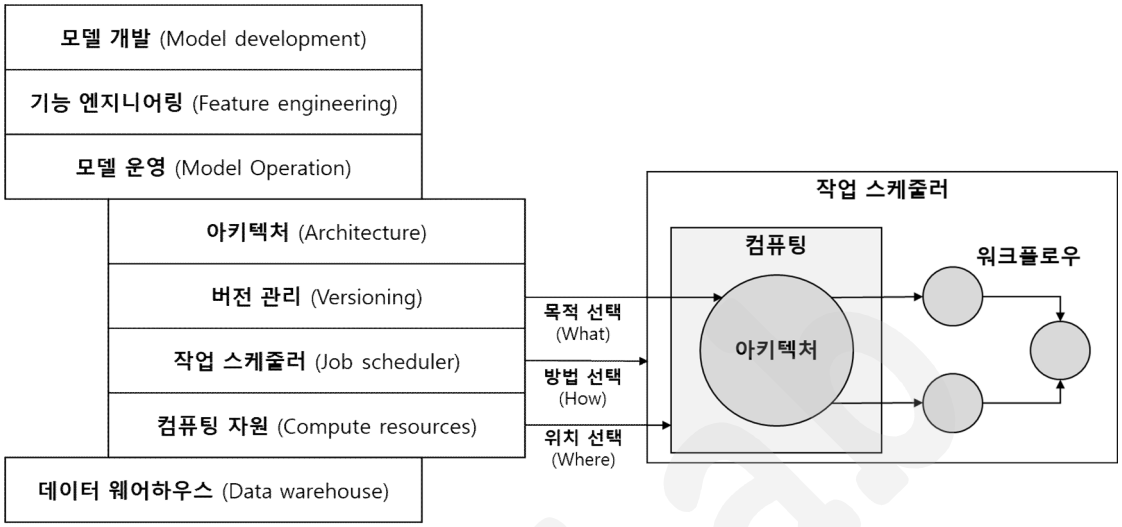

그림 5. 워크플로우 실행의 세 가지 고려 사항

데이터 플랫폼은 일반적으로 아래의 기능을 제공한다.

- 검색 바: 사용자는 키워드, 자료 이름 또는 기타 기준으로 특정 자료 데이터를 검색할 수 있다.
- 데이터 개요: 여기에는 자료 수, 데이터 포인트 수, 데이터 소스 등 사용 가능한 자료 데이터에 대한 개요를 표시한다.
- 데이터 시각화: 산점도, 히스토그램, 히트 맵과 같은 데이터의 시각적 표현을 표시합니다. 사용자는 시각화하려는 데이터를 선택하고 원하는 대로 시각화를 사용자 지정할 수 있다.
- 머신 러닝 모델: 사용 가능한 데이터를 기반으로 재료 속성을 예측할 수 있는 머신 러닝 모델에 액세스할 수 있다. 사용자는 사용하려는 모델을 선택하고 자체 데이터를 입력하여 예측 결과를 얻을 수 있다.
- 데이터 필터: 이를 통해 사용자는 재료 유형, 속성 또는 소스와 같은 특정 기준에 따라 재료 데이터를 필터링할 수 있다.
- 데이터 내보내기: 사용자는 추가 분석을 위해 데이터와 시각화를 내보내 자신의

연구 또는 프로젝트에 사용할 수 있다.

사용자 관점의 데이터 플랫폼 서비스는 편리한 서비스와 데이터의 활용을 위한 체계가 중요하며, 아래 '그림 6. Data Product Canvas'[1] 와 같은 요구를 설계에 반영하는 것이 필요하다.

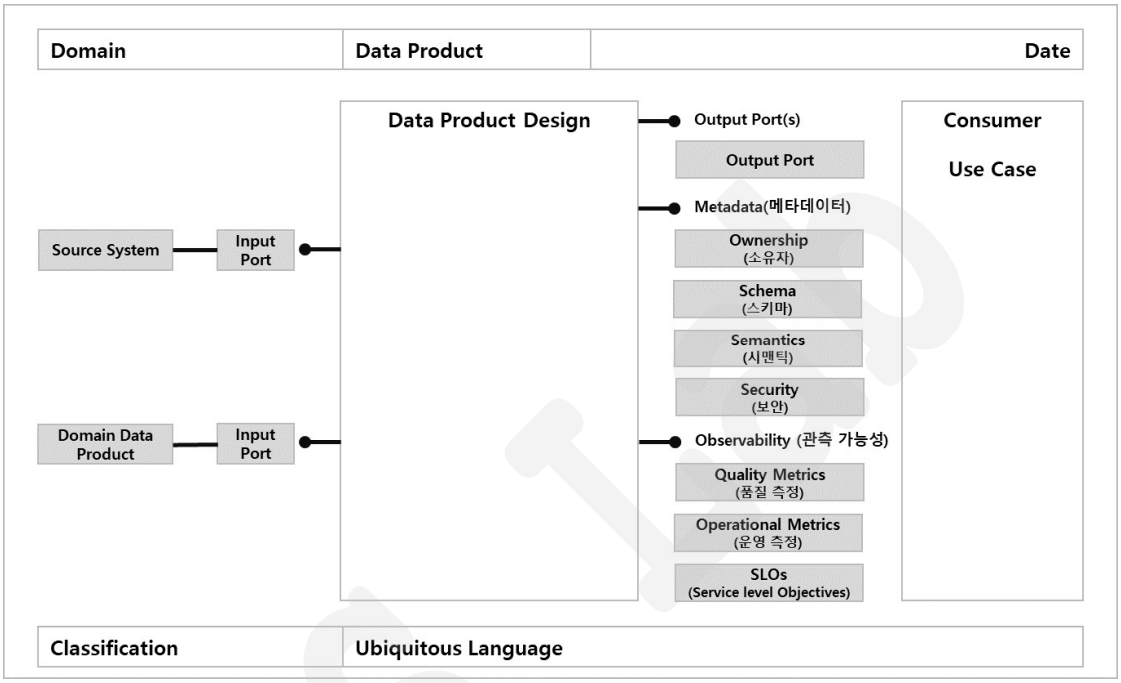

그림 6. Data Product Canvas

위 '그림 6. Data Product Canvas'의 각 항목별 설명은 아래와 같다.

- 도메인: 각 데이터 제품은 하나의 도메인 팀에서만 구현, 발전 및 유지 관리해야 한다. 따라서 각 데이터 제품은 정확히 하나의 도메인에 속한다. 이러한 구성 요소(Building Block)와 관련된 질문은 다음과 같다.
  - 데이터 제품에 대한 책임은 누구에게 있는가?
  - 누가 데이터 제품의 요구 사항을 지정하나?
  - 데이터 제품에 대한 질문에 누가 답변할 것인가?
  - 데이터 제품이 오류가 있으면 누가 수정하나?

1) 참고: https://www.datamesh-architecture.com/data-product-canvas

- 데이터 제품 이름: 각 데이터 제품에는 조직 내에서 식별하고 액세스할 수 있는 고유한 이름이 있다. 일반적으로 제품 이름은 일반적인 명명 전략을 따라야 한다.

- 소비자 및 사용 사례: 이 빌딩 블록은 데이터 제품의 존재 이유를 설명한다. 데이터 제품 설계는 '제품 사고' 철학을 따른다. 항상 사용자의 요구에서 시작한다.

  ▫ 데이터 제품의 목적을 파악하기 위해 분석 사용 사례와 조직의 목표를 설명한다. 사용 사례를 이해하는 것은 사용 사례를 구현하는 데 필요한 데이터를 지정하는 데 필수적이다. 향후 데이터 제품의 소비자는 자체 도메인 팀 또는 다른 도메인 팀일 수 있다.

- 출력 포트: 출력 포트는 데이터가 노출될 수 있는 형식과 소비 프로토콜을 정의한다. (예)로 출력 포트는 데이터베이스 테이블, 파일, API 또는 시각화일 수 있다.

- 메타데이터: 데이터 제품은 출력 포트 사양 외에도 메타데이터를 정의해야 한다.

  ▫ 소유권 부분은 데이터 제품의 도메인 이름, 제품 소유자, 조직 단위, 라이선스, 버전 및 예상 만료일을 설명
  ▫ 데이터 스키마 부분은 속성, 데이터 유형, 제약 조건 및 데이터 단위의 다른 요소와의 관계(예: 기능 종속성)를 설명
  ▫ 의미론 부분은 데이터 단위의 논리 모델에 대한 설명을 제공
  ▫ 보안 블록은 데이터 제품 사용에 적용되는 보안 규칙(예: 공개, 조직, 내부 및 PII 속성)을 설명
  ▫ 필수 데이터 집합 속성과 그 의미 대한 논의는 데이터 자체에 대한 팀의 이해에 기여하며 추가 공간을 사용하는 (예)로 브레인스토밍과 토론

- 입력 포트: 이 빌딩 블록은 향후 데이터 제품에 대한 입력 데이터를 설명이며, 입력 포트는 데이터 제품을 구성할 데이터의 수신 메커니즘 (입력 포트는 데이터를 읽을 수 있는 형식과 프로토콜을 정의, 여기서는 운영 소스 시스템과 내부 또는 다른 도메인에서 가져올 수 있는 기타 데이터 제품을 구분)

- 분류: 이 블록에서는 노출된 데이터의 특성을 지정 (데이터 제품을 소스 정렬, 집계 또는 소비자 정렬 중 하나로 분류)

- 데이터 제품 설계: 데이터 제품의 내부를 설계하는 핵심 빌딩 블록이며, 입력 포트와 출력 포트 사이의 모든 것을 지정하고 데이터 제품을 설계하는 데 필요한 모든 것을 개념적인 수준에서 설명 (예: 데이터 수집, 저장, 전송, 랭글링[2], 정리, 변환, 강화, 보강, 분석, SQL 문 또는 데이터 플랫폼 서비스)

- 관찰 가능성: 데이터 제품 설계는 메타데이터 사양 외에도 통합 가시성을 위한 정보를 의미

  ▫ 품질 메트릭은 정확성, 완전성, 무결성, 데이터 거버넌스 정책 준수와 같은 데이터 품질 요구 사항과 메트릭을 설명

  ▫ 운영 메트릭에는 변경 주기, 최신성, 사용 통계, 가용성, 사용자 수, 데이터 버전 관리 등이 포함

  ▫ 데이터 제품에 대한 SLO는 각 데이터 제품에 대한 신뢰성을 구축하는 규율을 가능하게 한다. 여기에서는 경보를 트리거하는 메트릭의 임계값을 지정

  ▫ 이것은 메타데이터 및 통합 가시성 포인트의 광범위한 목록이 아니다. 데이터 메시 아키텍처를 구축하기 위해 조직에서 필요한 모든 정보를 정의

- 유비쿼터스 언어: 프로젝트에 관련된 모든 사람이 공유하는 공통 언어를 여기에 설명한다. 일반적으로 운영 시스템 및 데이터 제품과 관련된 컨텍스트별 도메인 용어

2) 랭글링: 데이터 랭글링 은 다양한 데이터 소스의 데이터를 통합하고 쉽게 액세스하고 분석할 수 있도록 정리하는 프로세스

2. 데이터 수집과 저장

데이터 수집 계층은 데이터를 수집하는데 사용하는 계층이며, 다양한 데이터 소스에서 데이터를 수집하고, 표준화 및 정규화하여 데이터 처리 계층에 제공한다.

데이터 저장 계층은 데이터를 저장하는데 사용되는 계층이며, 대규모 데이터를 저장하기 위해 클라우드 스토리지 또는 빅 데이터 저장소를 사용한다.

데이터 변환을 포함한 ETL[3]이 필요한 데이터 스토리지 프레임워크(Data Storage Framesworks)는 아래와 같이 3개의 기술이 시장에 적용되고 있으며, 최근에는 자동화와 데이터 인프라를 융합하는 데이터 레이크하우스(Data Lakehouse) 기술이 주목을 받고 있어 데이터 플랫폼의 장기적인 서비스 발전 방향으로 고려하는 것이 바람직하다.

- 데이터 웨어하우스 (Data Warehouse)
- 데이터 레이크 (Data Lake)
- 데이터 레이크하우스 (Data Lakehouse)

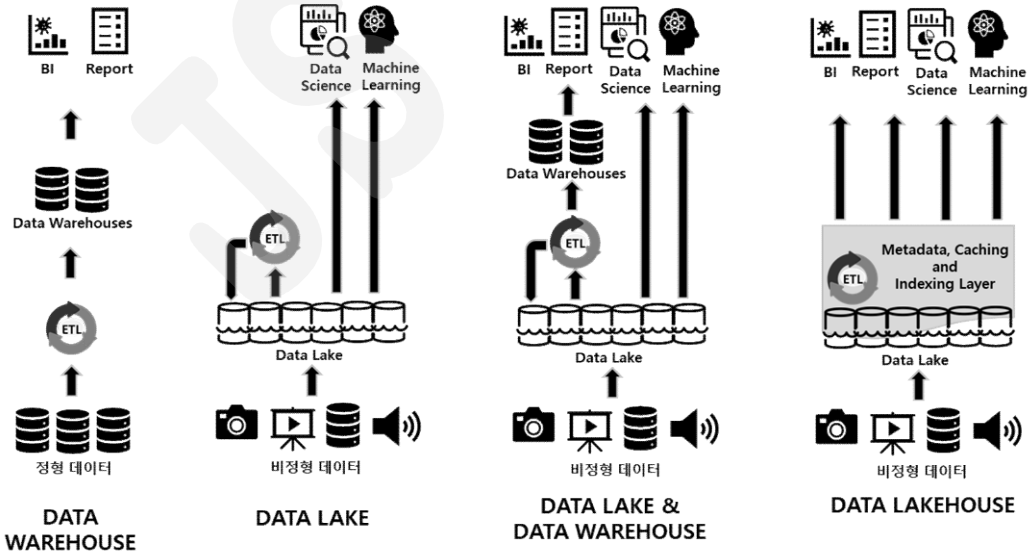

그림 7. 데이터 스토리지 프레임워크 (Data Storage Framesworks)

3) ETL: 추출(Extract), 변환(Transform), 로드(Load)

데이터 표준화를 위해 목표에 필요한 공통 구성 항목들을 체계화하는 것이 필요하다. 개발 단계에서는 큰 카테고리로 하여 데이터를 정리하고, 자세한 하부 정보를 구체화하여 데이터를 체계화해야 한다.

이러한 표준화 항목을 토대로 하여 표준 템플릿을 구축하는 것이 중요하며, 실제 데이터를 생성하고 제공하는 도메인 전문가와 인공지능 전문가가 팀을 이루어 연구 개발을 수행할 수 있도록 해야 한다. 이를 위해 해외의 사례를 벤치마킹하고 데이터를 비교 분석하며 템플릿을 구성해야 할 것이다. 용어 및 분류체계 등에 대한 표준화 방안 마련도 또한 필요하다.

3. 데이터 처리

데이터 처리 계층은 수집된 데이터를 처리하고 저장하는데 사용되는 계층이다. 이 계층은 대량의 데이터를 처리하고 저장하기 위한 기술을 포함하며, 대규모 데이터 분석 및 머신 러닝 모델링을 수행하기 위한 도구를 제공한다.

아래 '그림 8. 분석과 ML 애플리케이션의 데이터 흐름 비교'와 같이 목적에 따라 데이터의 흐름에서 처리하는 방법이 다를 수 있고, 데이터의 일반적인 특성을 대시보드에 표시하여 워크플로우 설계를 위한 서비스를 개선하는데 사용하여 병행 할 수 있다.

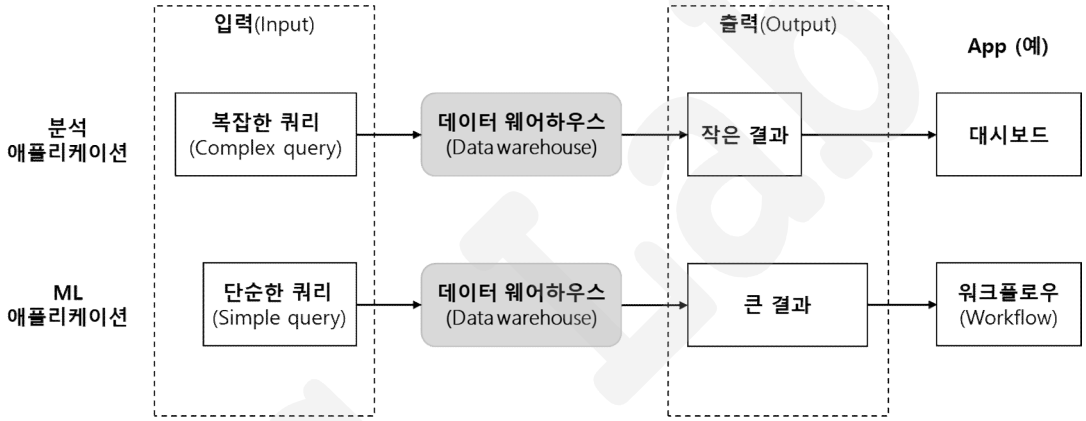

그림 8. 분석과 ML 애플리케이션의 데이터 흐름 비교

여러 데이터 소스를 사용하면서 두번 이상의 변환이 필요한 경우가 있에 이를 위한 데이터 파이프라인을 자동화하여 아래 '그림 9. 데이터 처리를 위한 분석 처리 엔진'과 같은 처리를 오케스트레이션 하는 것도 고려 하며 서비스를 개선 할 수 있다.

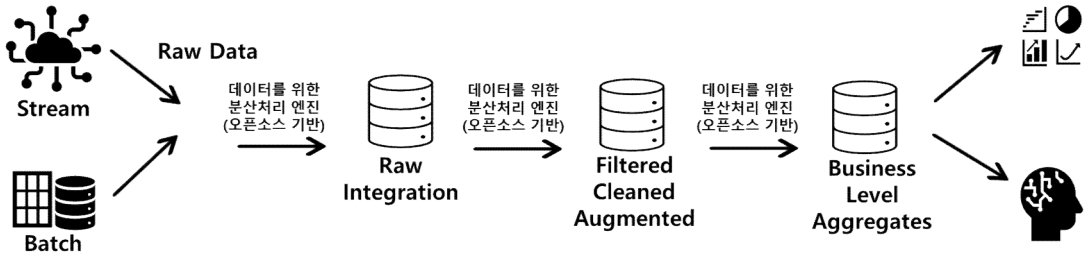

그림 9. 데이터 처리를 위한 분석 처리 엔진

4. 데이터 검색

데이터 카탈로그는 사용자가 필요한 정보를 빠르게 찾을 수 있도록 하는 데이터 자산 목록이다. 카탈로그는 대부분 다른 데이터에 대한 기본 정보를 제공하고 그것이 무엇인지 설명하는 메타데이터이다. 사용자는 데이터 관리 및 검색 도구와 결합한 데이터 카탈로그를 가진다.[4)]

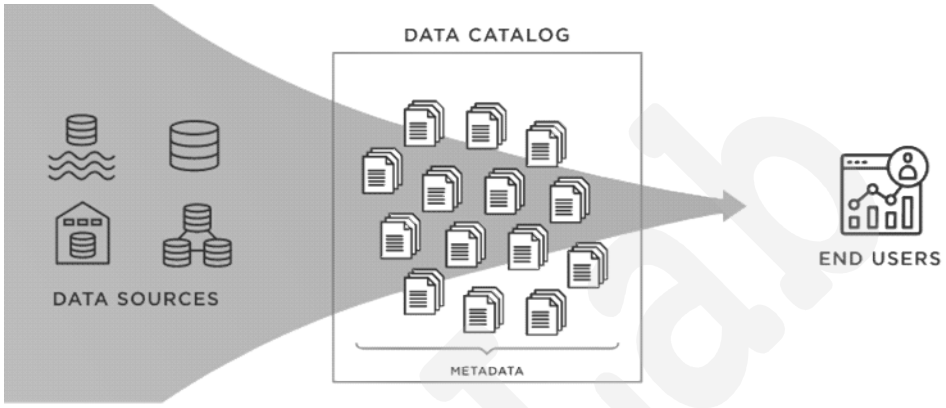

그림 10. 데이터 카탈로그

데이터 카탈로그는 다음을 포함하여 데이터 자산에 대한 모든 데이터를 제공한다.[5)]

- 상위 데이터 자산 그룹 또는 데이터 자산 그룹의 일부를 형성하는지 여부
- 데이터 자산을 소유자
- 데이터 자산이 데이터 수명 주기에 있는 위치
- 데이터 소유자가 관리하는 자산에 대한 설명
- 데이터 자산에서 파생된 메타데이터 (예: 자산 유형, 형식, 변경 내역 등)
- 데이터 자산 보안
- 데이터 자산에 대한 감사 정보
- 적용 가능한 모든 데이터 표준 또는 정책
- 데이터 자산의 데이터 계보
- 데이터 미리보기

4) 참고: https://www.tibco.com/ko/reference-center/what-is-a-data-catalog
5) 참고: https://www.risual.com/2019/03/modern-data-platform-data-catalogue/

검색은 이를 위한 카탈로그 설계에서 개선의 방향을 구성 할 수 있다. 데이터 플랫폼은 서비스의 활성화를 위해 여러 시스템을 하나인 것 같이 통합하는 것을 최종 목표로 할 수 있고, 다음 '그림 11. 오픈 페더레이션 카탈로그'[6]와 같이 검색을 위한 데이터 카탈로그 관리가 하나 이상의 체계로 분산되어 있는 것을 수용하는 방안을 보여준다.

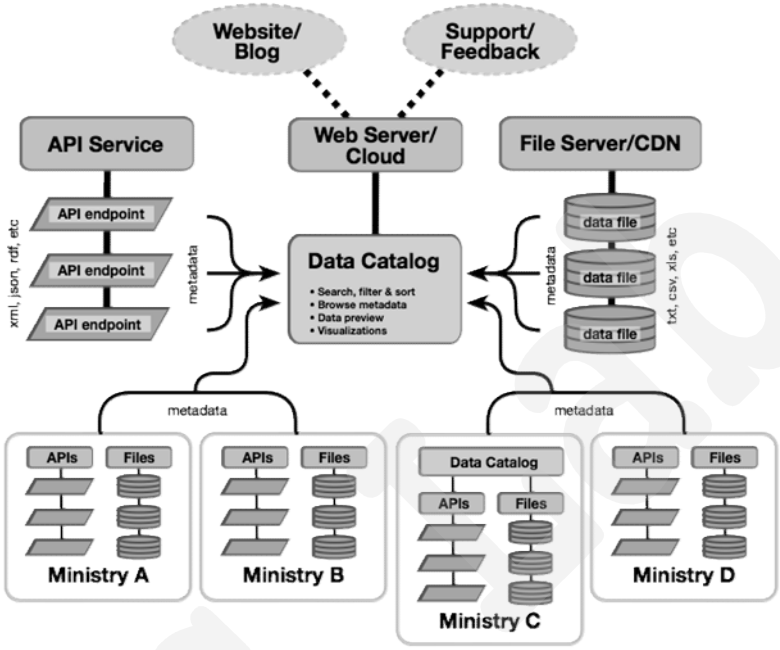

그림 11. 오픈 페더레이션 카탈로그

이 접근 방식에서는 일부 데이터 파일 또는 API 서비스는 개별 데이처 체계에서 관리하지만, 메타데이터는 중앙 카탈로그에 제공되어 체계간 검색 및 액세스가 가능하다. 한 제계는 중앙 카탈로그를 지원하면서 자체 카탈로그(예: 지리공간 데이터 또는 교육 통계)를 운영할 수 있다.

검색을 위한 오픈 데이터 카탈로그의 구성하는 방법은 아래와 같다.

- Easy access (쉬운 접근): 개방형 데이터 카탈로그를 사용하면 사용자가 빠르고 자유롭고 직관적으로 데이터에 매우 쉽게 액세스할 수 있다. 오픈 데이터 카탈로그에 액세스하려면 등록이나 로그인이 필요하지 않다. 이러한 요구 사항은 탐색 및 사용을 방해하기 때문이다.

6) 참고: http://opendatatoolkit.worldbank.org/en/technology.html

- Search (검색): 개방형 데이터 카탈로그를 사용하면 데이터를 쉽게 찾을 수 있다. 대부분의 데이터 카탈로그는 주제, 조직 또는 유형별로 데이터를 정렬하고 카탈로그 콘텐츠의 전체 텍스트 검색을 지원한다. 많은 오픈 데이터 카탈로그는 데이터를 기존 검색 엔진에 노출하기 위해 검색 엔진 최적화를 구현한다.
- Machine-readable data access(기계 판독 가능 데이터 액세스): 데이터는 컴퓨터가 읽을 수 있는 비독점 전자 형식으로 다운로드할 수 있다. 가능하면 데이터셋의 모든 데이터를 단일 다운로드 파일로 사용할 수 있도록 하는 것을 권장한다.
- Metadata (메타데이터): 게시 날짜 및 속성과 같은 주요 메타데이터는 각 데이터 세트에 대해 눈에 띄게 표시된다. 많은 오픈 데이터 카탈로그는 더블린 코어 메타데이터 표준을 구현하고 메타데이터를 기계가 읽을 수 있는 형식으로 제공한다.
- Clear data licenses (데이터 라이선스 지우기): 데이터 라이센스는 각 데이터 세트에 대해 명확하고 눈에 띄게 표시된다. 크리에이티브 커먼즈, 오픈 데이터 라이선스 또는 기타 표준에 따라 데이터 라이선스가 부여된 경우 이러한 라이선스에 대한 투명한 링크가 포함된다.
- Data preview/visualization (데이터 미리보기/시각화): 많은 개방형 데이터 카탈로그에는 내장 그래프 또는 매핑 도구를 사용하여 데이터를 다운로드하거나 시각화하기 전에 데이터를 미리 볼 수 있는 기능이 포함되어 있다.
- Standards compliance (표준 준수): 대부분의 오픈 데이터 카탈로그에는 데이터 형식(예: CSV, XML, JSON) 및 메타데이터(예: 더블린 코어)와 같은 다양한 표준에 대한 지원 기능이 내장되어 있다. 개방형 데이터 카탈로그는 대부분 각 데이터셋을 고유하고 영구적인 URL로 제공하므로 데이터를 인용하고 연결할 수 있다.
- Application Programming Interface (애플리케이션 프로그래밍 인터페이스, API): API를 사용하면 소프트웨어 개발자가 소프트웨어를 통해 오픈 데이터 카탈로그(종종 데이터 자체)에 액세스할 수 있다. API는 데이터 검색, 분석, 카탈로그 통합, 외부 사이트 및 애플리케이션 호스트에서 메타데이터 수집을 용이하게 한다.
- Security (보안): 개방형 데이터 카탈로그는 권한이 없는 사용자가 데이터와 메타데이터를 변경하지 못하도록 보호하는 보안 조치를 구현한다.

5. 데이터 분석 및 시각화

데이터 분석 및 시각화는 데이터를 시각적으로 표현하여 쉽게 이해할 수 있도록 돕는 기술이다. 데이터 분석 결과를 시각화하면 데이터의 패턴이나 추세를 빠르게 파악할 수 있다. 파이썬에서는 Matplotlib[7], Seaborn[8], Plotly, Bokeh 등 다양한 라이브러리를 활용하여 데이터 시각화를 할 수 있다.

데이터 분석은 데이터를 수집하고 정리하여 의미 있는 정보를 추출하는 과정이다. 데이터 분석을 통해 데이터의 특징이나 패턴을 파악하고, 이를 바탕으로 의사결정을 내릴 수 있다.

아래 '그림 12. 일반적인 데이터 분석의 패턴'[9] 은 분석의 성격에 따라 다른 방법으로 처리하는 구성을 보여 준다.

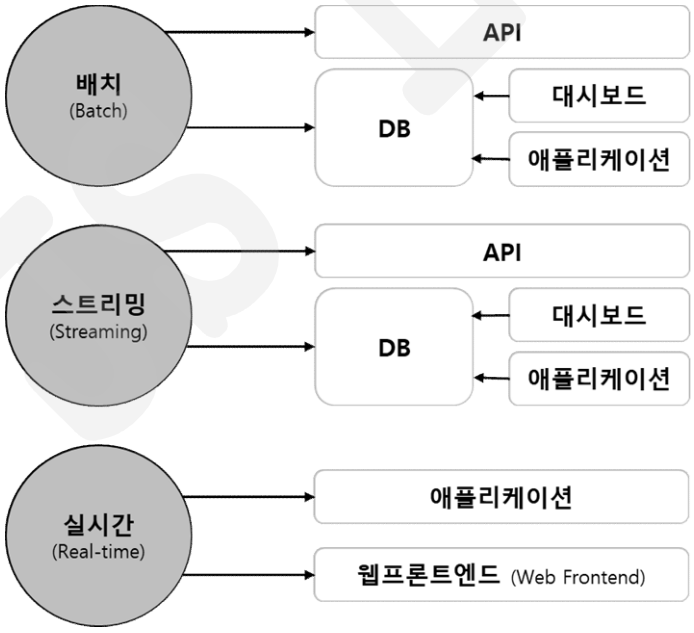

그림 12. 일반적인 데이터 분석의 패턴

7) Matplotlib: 파이썬에서 데이터 시각화와 2D 그래프 플롯에 사용하는 라이브러리
8) Seaborn: 파이썬 데이터 시각화 라이브러리로, matplotlib 기반으로 만들어졌다
9) Source: Tuulos, Ville. Effective Data Science Infrastructure: How to make data scientists productive

아래 '그림 13. 데이터 파이프라인 아키텍처' 기반의 데이터 분석 파이프라인을 구축하는 방법은 아래와 같다.

- 데이터 수집: 데이터베이스, API, 로그 파일 등과 같은 다양한 소스에서 데이터를 수집한다.
- 데이터 로드: 데이터 레이크 또는 데이터 웨어하우스와 같은 중앙 집중식 데이터 스토리지 시스템에 데이터를 로드한다.
- 데이터 준비: 분석에 사용할 수 있도록 데이터를 정리, 변환 및 정규화합니다.
- 데이터 분석: SQL, Python 또는 R과 같은 도구를 사용하여 탐색적 데이터 분석을 수행하고 시각화를 생성하며 인사이트를 생성한다.
- 데이터 모델링: 기계 학습 알고리즘을 사용하여 예측 모델을 생성하여 예측하고 미래 추세에 대한 통찰력을 얻는다.
- 데이터 시각화: 이해관계자가 이해하기 쉬운 방식으로 분석 및 모델링 결과를 제시한다.
- 데이터 인프라: 파이프라인이 클라우드 기반 솔루션 및 분산 컴퓨팅 시스템과 같은 확장 가능한 인프라를 사용하여 증가하는 데이터 양을 처리할 수 있는지 확인한다.
- 모니터링 및 유지 관리: 파이프라인이 올바르게 작동하는지 지속적으로 모니터링하고 필요에 따라 변경하여 원활하게 실행되도록 한다.

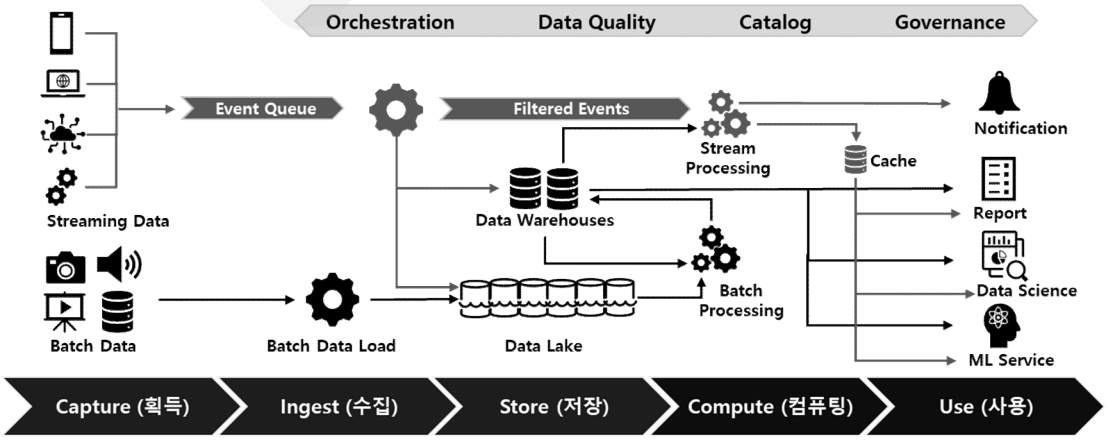

그림 13. 데이터 파이프라인 아키텍처

6. 데이터 보안

데이터의 기밀성, 무결성, 가용성을 보장해야 하는 것은 새로운 일은 아니지만, 데이터의 양이 증가하고 한 시스템에서 다른 시스템으로 데이터가 복사됨에 따라 더욱 까다로워질 수 있다. 이러한 시스템 간의 정보 보안 관행의 차이로 인해 데이터가 침해에 취약해질 수 있다. 아래 '그림 14. 일반 보안 모델'[10]과 같이 데이터 스토리지와 연계하며 고려 할 사항들이 있다.

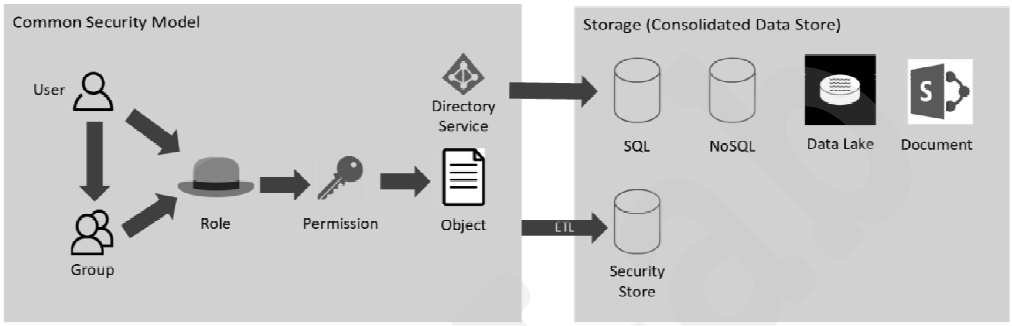

그림 14. 일반 보안 모델

- 누가 특정 개체에 액세스할 수 있나?
- 사용자가 액세스할 수 있는 개체는 무엇인가?
- 특정 사용자가 특정 날짜에 이 개체에 액세스할 수 있나?

보안을 위한 감사(Audit)에 대한 고려가 필요하며 아래는 '그림 15. 일반 감사 모델(Common Audit Model)'이다.

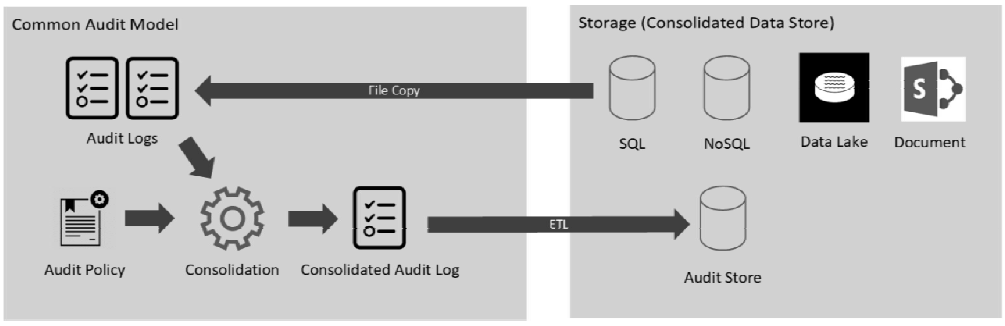

그림 15. 일반 감사 모델(Common Audit Model)

10) 참조: https://www.risual.com/2019/03/modern-data-platform-security/

- 이 객체는 언제 누가 마지막으로 변경했는지 확인
- 이 개체에 어떤 변경이 있었는지 확인
- 사용자가 액세스한 개체는 언제 무엇인지 확인

데이터 수명 주기 관리의 일부가 되어야 하는 데이터 보호를 위해 잘 정립된 관행에는 다음과 같은 것들이 있다.[11)]

- 미사용 데이터와 전송 중인 데이터의 기밀성을 유지하기 위한 암호화 해당 데이터에 대한 특정 역할이 필요한 사용자 및 서비스에만 데이터 액세스를 제공하는 액세스 제어 자원에 대한 접속을 관리하는 규칙을 정의하는 정책 암호화 키 생성, 관리 및 폐기를 위한 키 관리 데이터 규모가 증가함에 따라 이러한 관행의 구현 및 관리 자동화 한다.
- 데이터의 양이 증가함에 따라 분석 및 머신 러닝 사용자의 요구를 더 잘 충족하기 위해 스토리지 시스템에 변화를 고려한다.
- 오브젝트 스토리지가 블록 및 파일 시스템 스토리지에서 사용할 수 있는 일부 기능을 제공하는 것을 고려한다.
- 스토리지 시스템은 자주 액세스하지 않는 데이터를 더 저렴한 스토리지로 마이그레이션하고 데이터베이스가 데이터에 액세스하는 방식을 최적화하는 등 데이터 수명 주기 관리 작업을 자동화하는 방법을 고려한다.

동일한 노트북을 데이터 과학 외에도 여러 용도로 사용하기 때문에 많은 문제가 발생하며, 노트북을 물리적으로 분실할 수도 있다. 다른 클라우드 인스턴스와 유사하게 보안 및 모니터링이 쉽다. 클라우드 서비스의 경우는 AWS IAM과 같은 표준 클라우드 기반 인증 및 권한 부여 시스템을 사용할 수 있다.

인프라 팀이 외부 클라우드 워크스테이션과 로컬 노트북 사이에 VPN(가상 사설망)과 같은 보안 네트워크 연결을 포함하여 워크스테이션을 설정해야 한다. 초기 구성 비용을 지불하고 나면 이 설정은 데이터 과학에 매우 생산적일 수 있다.

---

11) Sullivan Ph.D., Dan. The Gorilla Guide to...® Navigating Data, Analytics, and AI/ML in 2021

7. API 게이트웨이

API(Application Programming Interface)를 사용하면 소프트웨어 개발자가 소프트웨어를 통해 오픈 데이터 카탈로그(또는 데이터 자체)에 액세스할 수 있다. API는 데이터 검색, 분석, 카탈로그 통합, 외부 사이트 및 애플리케이션 호스트에서 메타데이터 수집을 용이하게 한다. 일반적인 API 구성은 아래 '그림 16. API 플랫폼과 인프라 구성'[12]과 같다.

그림 16. API 플랫폼과 인프라 구성

12) 참고: Biehl, Matthias. API Architecture: The Big Picture for Building APIs (API-University Series Book 2)

다음은 데이터 플랫폼에 대한 오픈소스 소프트웨어를 사용하는 하이레벨 API 아키텍처(예)이다.

- API 게이트웨이: Kong[13]이나 Tyk[14]와 같은 API 게이트웨이를 사용하여 API 엔드포인트를 관리하고 보호한다. API 게이트웨이는 API 엔드포인트에 대한 역방향 프록시 역할을 하며 인증, 속도 제한 및 기타 보안 관련 작업을 처리한다.
- 마이크로 서비스: 각 API 엔드포인트를 별도의 마이크로서비스로 구현하는 것이 바람직 하다. 마이크로서비스를 관리하려면 Kubernetes 또는 Docker Swarm과 같은 컨테이너 오케스트레이션 플랫폼을 사용한다.
- 서비스 레지스트리 및 검색: Consul[15] 또는 Eureka[16]와 같은 서비스 레지스트리 및 검색 도구를 사용하여 마이크로서비스의 위치 및 상태를 관리한다. API 게이트웨이는 서비스 레지스트리를 쿼리하여 마이크로서비스의 위치를 확인한다.
- API 문서: Swagger[17] 또는 OpenAPI와 같은 도구를 사용하여 API 엔드포인트에 대한 문서를 생성한다. 문서에는 API 엔드포인트, 요청 및 응답 형식, 인증 및 권한 부여 요구 사항에 대한 정보가 포함되어야 한다.
- API 테스트: Postman[18] 또는 Newman[19]과 같은 도구를 사용하여 API 엔드포인트를 테스트한다. 자동화된 테스트 모음을 만들어 API 엔드포인트가 올바르게 작동하고 예상 응답을 반환하는지 확인한다.

전반적으로 이 API 아키텍처는 재료 데이터 플랫폼을 위한 확장 가능하고 신뢰할 수 있는 솔루션을 제공하는 것을 목표로 한다. 마이크로서비스를 사용하면 각 API 엔드포인트를 독립적으로 개발 및 배포할 수 있으며, API 게이트웨이와 서비스 레지스트리는 엔드포인트를 관리하고 검색하는 중앙 집중식 보안 방법을 제공한다. API 문서 및 테스트 도구는 API 엔드포인트가 잘 문서화되어 있고 올바르게 작동하는지 확인하는 데 도움이 된다.

---

13) Kong 참조: https://github.com/Kong/kong
14) Tyk 참조: https://tyk.io/open-source/
15) Consul 참조: https://www.consul.io/
16) Eureka 참조: https://coe.gitbook.io/guide/service-discovery/eureka
17) Swagger 참조: https://swagger.io/
18) Postman 참조: https://www.postman.com/product/api-client/
19) Newman 참조: https://jokerkwu.tistory.com/195?category=922316

데이터 플랫폼에 대한 하이레벨 API 설계(예)는 아래와 같다.

- 데이터 수집 API (엔드포인트 /데이터/인제스트, POST): 데이터를 데이터 플랫폼으로 수집할 수 있도록 합니다. API는 JSON 또는 CSV 형식의 데이터를 수락하고 데이터 플랫폼 내에 적절한 형식으로 저장한다.
- 데이터 검색 API (엔드포인트 /데이터/검색, GET): 사용자가 전체 텍스트 검색, 패싯 검색 및 자연어 쿼리를 사용하여 데이터 플랫폼을 검색할 수 있습니다. 사용자는 검색 필터와 페이지 매김 매개변수를 지정할 수도 있다.
- 데이터 분석 API (엔드포인트 /데이터/분석, POST): 사용자가 Apache Spark를 사용하여 데이터 플랫폼에서 데이터 분석을 수행할 수 있도록 한다. 사용자는 JSON 또는 YAML 구성 파일을 사용하여 수행할 분석을 지정할 수 있다.
- 메타데이터 관리 API (엔드포인트 /metadata, GET/POST/PUT/삭제): 사용자가 데이터 플랫폼과 관련된 메타데이터를 관리할 수 있다. 사용자는 데이터 세트, 파일, 테이블 등 다양한 데이터 개체에 대한 메타데이터를 보고, 추가하고, 수정하고, 삭제할 수 있다.
- 사용자 관리 API (엔드포인트 /users, GET/POST/PUT/삭제): 관리자가 데이터 플랫폼과 관련된 사용자, 역할, 권한을 관리할 수 있다. 사용자는 사용자 계정을 보고, 추가, 수정, 삭제하고, 역할과 권한을 할당하고, 인증 및 권한 부여 메커니즘을 관리할 수 있다.
- 시스템 상태 API (엔드포인트 /health, GET): 데이터 플랫폼의 상태에 대한 정보를 제공한다. 여기에는 데이터 플랫폼의 가용성, 다양한 데이터 서비스의 상태 및 기타 시스템 관련 메트릭에 대한 정보를 포함한다.

전반적으로 API 설계는 사용자가 재료 데이터 플랫폼에서 데이터를 수집, 검색, 분석 및 관리할 수 있는 포괄적인 엔드포인트 집합을 제공하는 것을 목표로 한다. 이 API는 표준 HTTP와 상태 코드를 사용하여 RESTful로 설계하며, 다른 애플리케이션 및 도구와 쉽게 통합할 수 있다.

III. 데이터 인프라 구성

데이터 플랫폼과 연계하는 데이터 인프라 구성은 하드웨어 기반의 가속과 함께 소프트웨어 기반의 아키텍처로 데이터 분석 파이프라인을 최적화하는 서비스가 가능하다.

관리도구를 통합 가능한 HCI[20]와 같이 최단의 물리적 스토리지 경로 제공하여 데이터 파이프라인을 빠르게 처리하고, GPU Direct와 같이 CPU를 오프로드하여 AI/ML 처리를 가속 가능한 환경을 제공한다. 아래 '그림 17. HCI 기반의 데이터 처리'와 같이 HCI 기반의 데이터 플랫폼 상의 배치(Batch) 데이터는 물론 스트리밍 데이터의 파이프라인을 가속 할 수 있다.

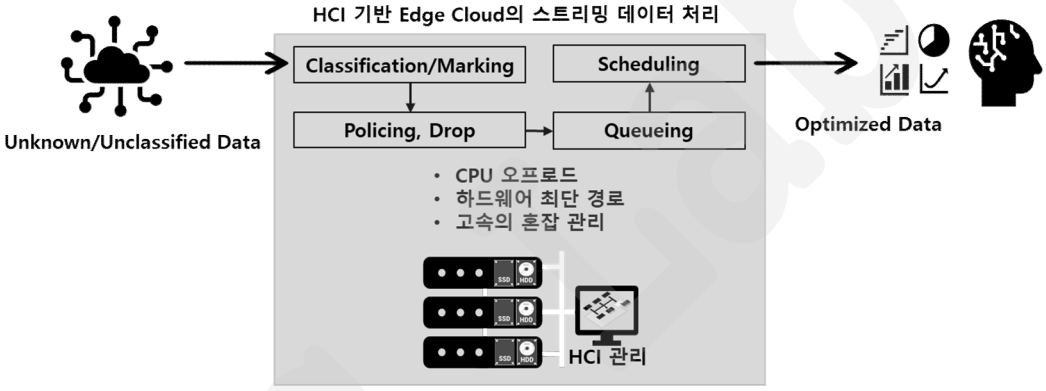

그림 17. HCI 기반의 데이터 처리

데이터는 장기적으로 이미지 데이터 처리와 AI/ML 등의 분석이 증가하여 GPU나 TPU 기반의 처리가 증가하는 것에 대응 할 수 있다.

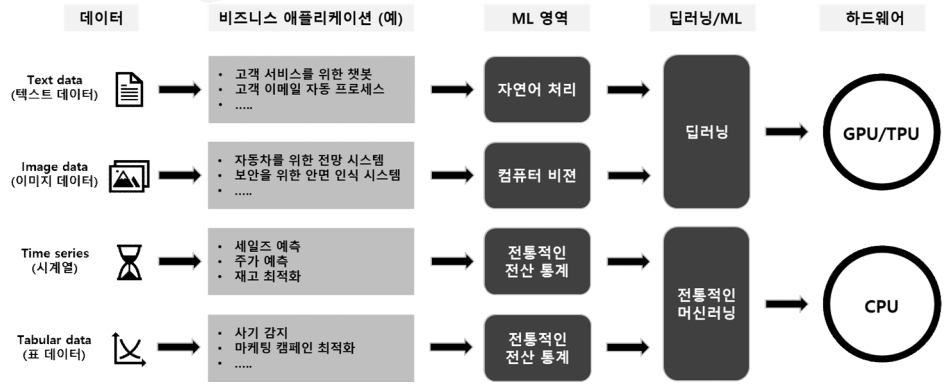

그림 18. From Data to Machine

20) HCI (Hyper Converged Infrastructure)

데이터 플랫폼 인프라의 가상화와 함께 데이터의 가상화도 장기적으로 도움이 될 수 있다. 데이터 가상화는 서로 다른 기술의 많은 소스 시스템이 있지만 모두 응답 시간이 빠르고 운영 애플리케이션을 많이 실행하지 않는 경우에 도움이 된다. 그런 식으로 데이터를 이동 및 복사하고 사전 집계하지 않지만 아래 '그림 19. 데이터 가상화를 위한 Cube 구성'[21]과 같이 비즈니스 모델을 생성하는 시맨틱 계층이 있으며, 이 데이터 가상화 계층을 쿼리하는 경우에만 데이터 소스를 쿼리한다. Dremio[22] 경우 오픈소스 Apache Arrow[23] 기술을 사용하여 많은 인메모리를 캐시하고 최적화할 뿐만 아니라 빠른 응답 시간을 단축한다.

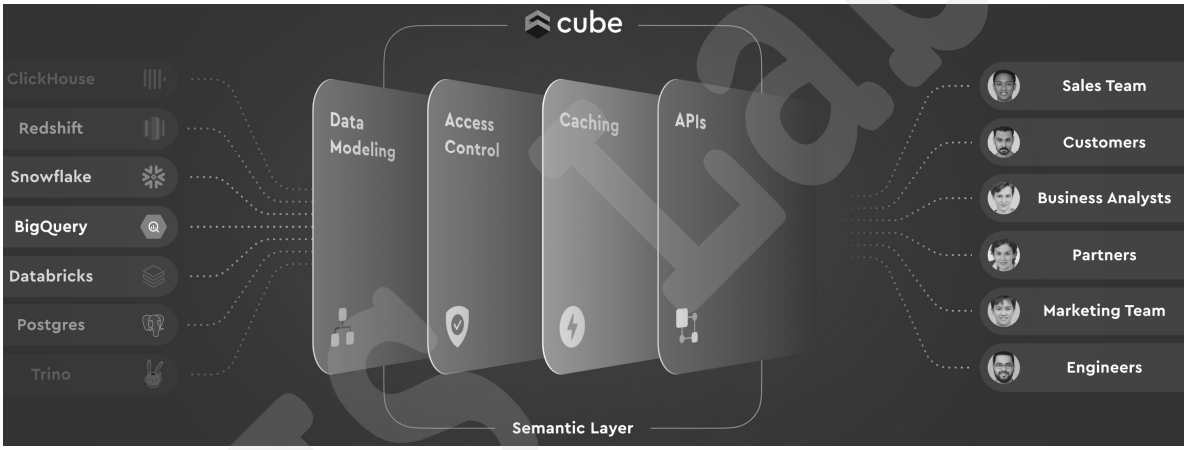

그림 19. 데이터 가상화를 위한 Cube 구성

- 끝 -

21) 참조: https://cube.dev/use-cases/semantic-layer
22) 참조: https://www.dremio.com/
23) 참조: https://arrow.apache.org/