

Portfolio

Derby - 문지현

목차

I. 실시간 데이터 분석 시스템 구축

1. 개요
2. 아키텍처
3. 모니터링 및 성능개선

II. Data lineage Pipeline prototype 개발

1. 개요
2. 아키텍처
3. Pipeline 구성 작업

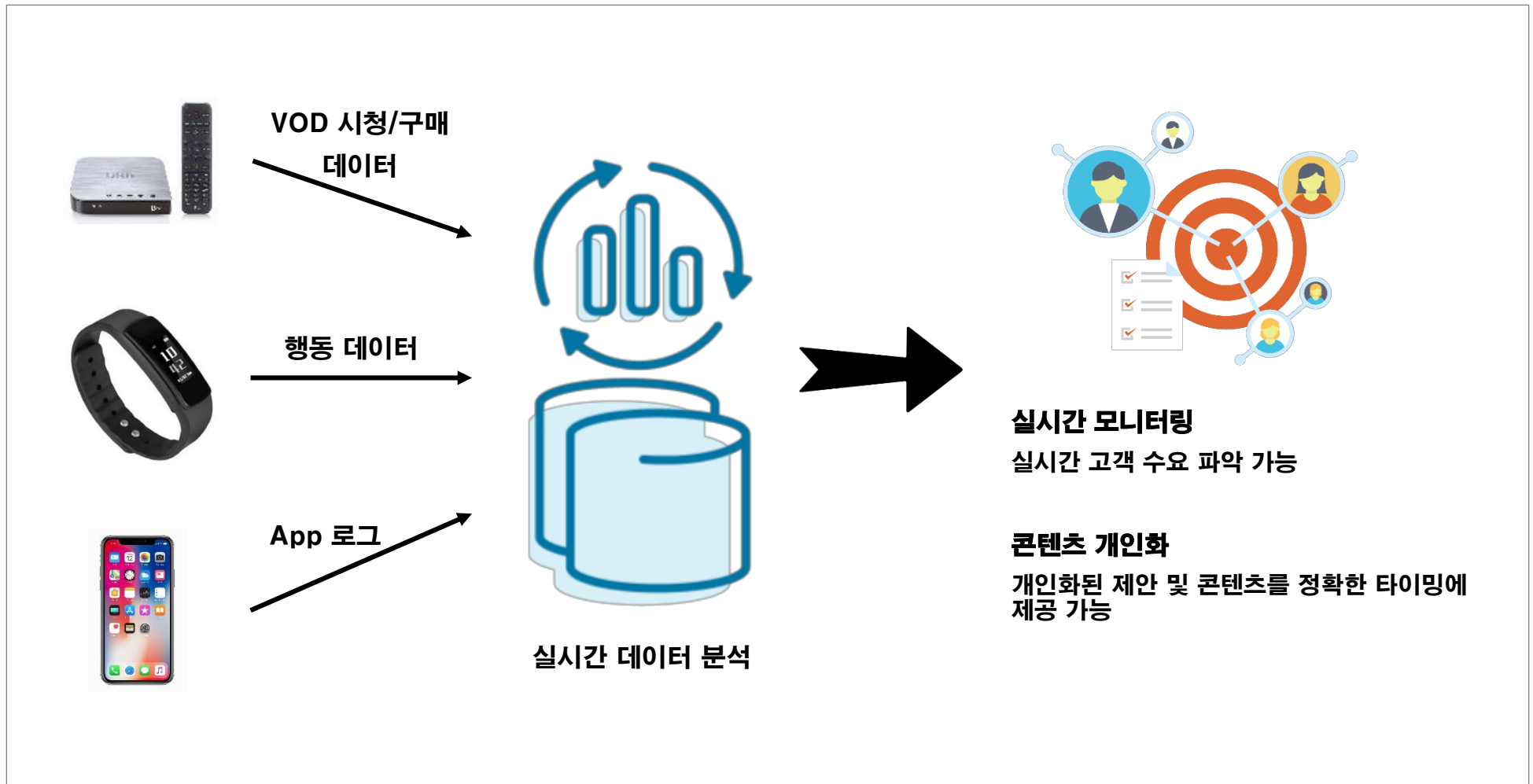
III. 블록체인 서비스 데이터 Pipeline 개발 및 운영

1. 개요
2. 아키텍처
3. Pipeline 구성 작업

1. 개요

I. 실시간 데이터 분석 시스템 구축

타겟 마케팅을 위해 고객의 서비스 활용 및 행동 데이터를 실시간으로 분석하여 고객이 원하는 요구사항에 빠르게 대응할 수 있는 실시간 데이터 분석시스템을 구축했습니다.

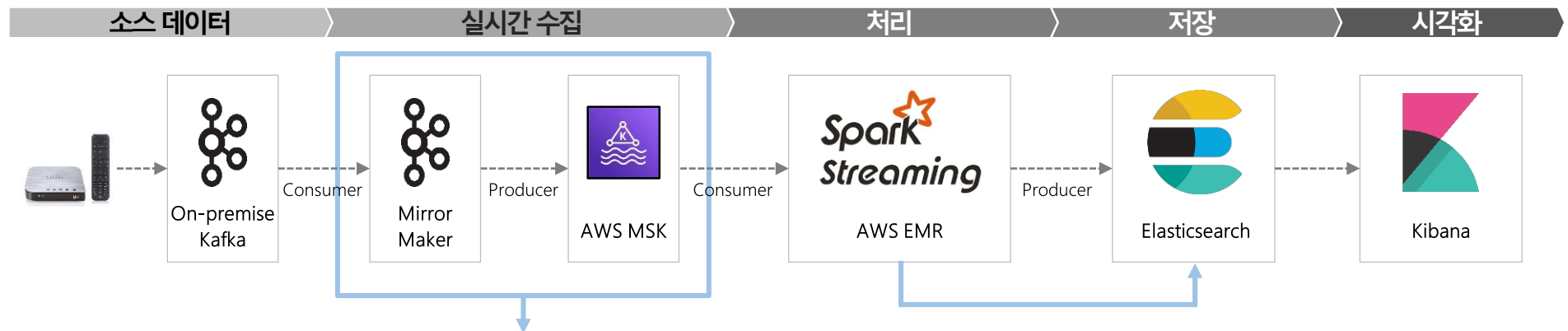


2. 아키텍처

실시간 데이터 분석시스템 구축 시 고려해야할 주요 사항을 정하고, 이에 맞는 아키텍처에 대한 구축을 수행했습니다.

✓ 주요 고려 사항

- ① 50초 미만의 지연시간
- ② 실시간으로 유입되는 대용량의 데이터에 대한 저장
- ③ 특정 이벤트 기간 동안의 분석에 적합한 환경



- ① 수집 단계에 해당하는 솔루션 튜닝 진행
→ 50초 미만의 지연시간 & 데이터 유실방지

- ② 데이터 집계/필터 처리 및 분석한 결과 데이터 마트 저장
→ 시간당 150GB로 유입되는 데이터의 저장 용량 줄여
저장공간 효율성 증대

- ③ 전체 분석 시스템의 on/off 자동화 프로그램 개발
→ 특정한 이벤트기간에만 분석하는 환경으로,
필요 없는 기간에는 전체 분석시스템을 종료하고, 필요시에 기동하여 AWS 요금 절약

3. 모니터링 및 성능개선

I. 실시간 데이터 분석 시스템 구축

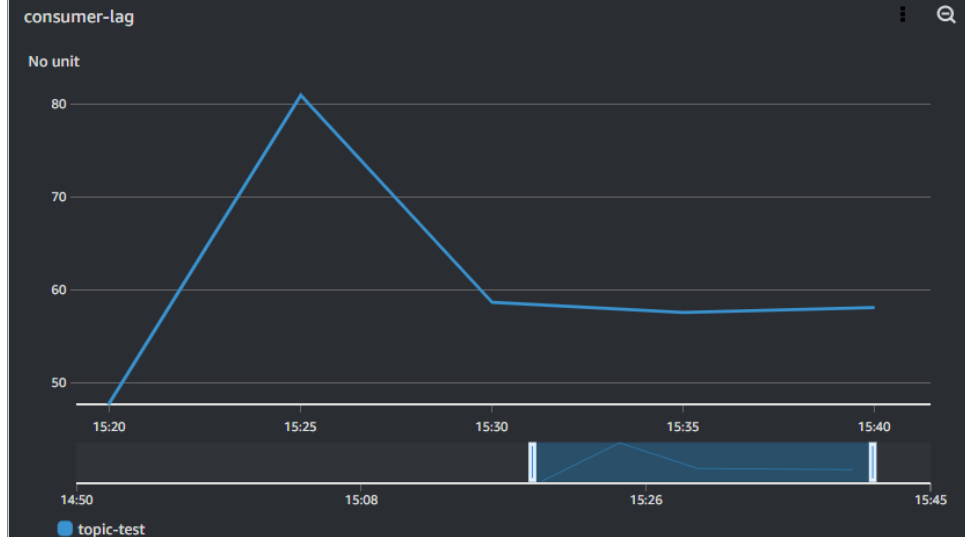
수집 단계 솔루션에 대한 성능확인 및 장애 대응을 위해 모니터링용 지표를 수집하여 대시보드를 구성했습니다.

MirrorMaker 모니터링을 위한 지표 수집

```
[ec2-user@ip-10-0-0-208 ~]$ sh test_cloudwatch.sh
Wed Sep 14 06:53:34 UTC 2022    topic:test    consumer-lag    116
Wed Sep 14 06:53:38 UTC 2022    topic:test    consumer-lag     6
Wed Sep 14 06:53:42 UTC 2022    topic:test    consumer-lag    14
Wed Sep 14 06:53:46 UTC 2022    topic:test    consumer-lag    21
Wed Sep 14 06:53:50 UTC 2022    topic:test    consumer-lag    28
Wed Sep 14 06:53:53 UTC 2022    topic:test    consumer-lag    36
Wed Sep 14 06:53:57 UTC 2022    topic:test    consumer-lag    43
Wed Sep 14 06:54:01 UTC 2022    topic:test    consumer-lag    51
Wed Sep 14 06:54:05 UTC 2022    topic:test    consumer-lag    59
Wed Sep 14 06:54:09 UTC 2022    topic:test    consumer-lag    66
Wed Sep 14 06:54:13 UTC 2022    topic:test    consumer-lag    73
Wed Sep 14 06:54:17 UTC 2022    topic:test    consumer-lag    81
Wed Sep 14 06:54:21 UTC 2022    topic:test    consumer-lag    89
Wed Sep 14 06:54:24 UTC 2022    topic:test    consumer-lag    96
Wed Sep 14 06:54:28 UTC 2022    topic:test    consumer-lag   103
Wed Sep 14 06:54:32 UTC 2022    topic:test    consumer-lag   111
Wed Sep 14 06:54:36 UTC 2022    topic:test    consumer-lag     1
```

- MirrorMaker 성능을 나타내는 지표 수집 프로그램 개발

MirrorMaker 모니터링 지표 시각화



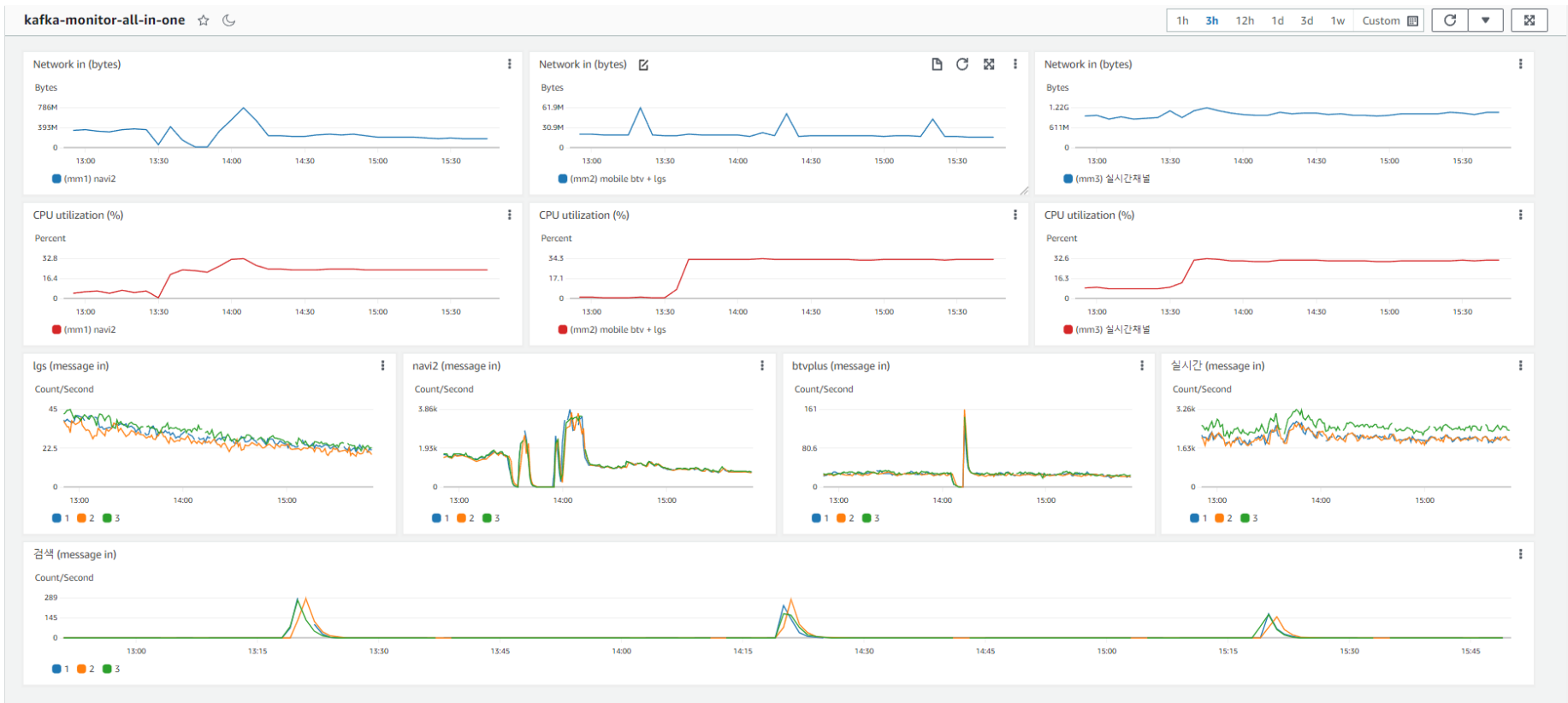
3. 모니터링 및 성능개선

I. 실시간 데이터 분석 시스템 구축

수집 단계 솔루션에 대한 성능확인 및 장애 대응을 위해 모니터링용 지표를 수집하여 대시보드를 구성했습니다.

AWS MSK 모니터링을 위한 대시보드

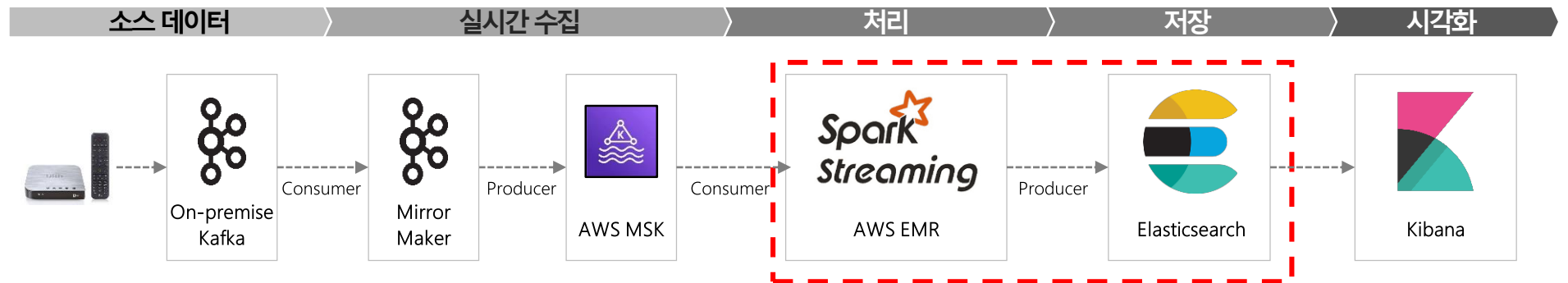
실시간으로 유입되는 데이터의 양, MSK의 CPU 사용량/잔여 저장용량 확인



3. 모니터링 및 성능개선

I. 실시간 데이터 분석 시스템 구축

실시간 데이터 분석시스템에서 저장용으로 사용한 Elasticsearch의 분석 쿼리 성능의 문제가 있어, 해당 문제의 개선안으로 Druid 솔루션 활용을 고려했습니다.



■ Elasticsearch의 이슈 사항

1. Elasticsearch의 집계 쿼리 성능 문제
2. 랭킹 분석 시, 서브 랭킹 분석에 대한 성능 문제

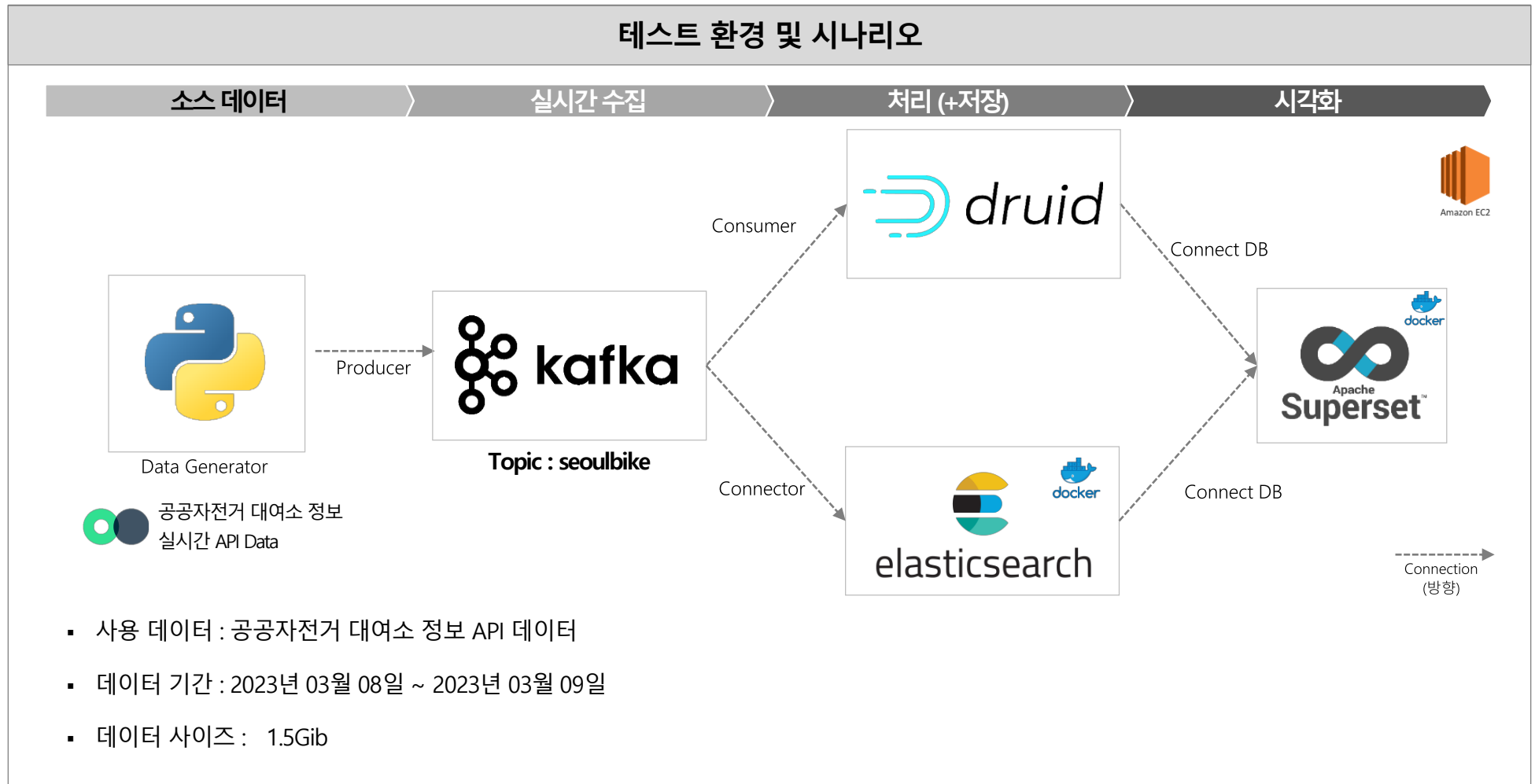


- ✓ 온라인 분석처리 작업에 중점을 둔 솔루션
- ✓ 데이터 스캔 및 집계에 최적화
- ✓ 반정형, 정형 데이터 처리

3. 모니터링 및 성능개선

I. 실시간 데이터 분석 시스템 구축

Elasticsearch 와 Druid의 분석 쿼리 성능 비교 테스트를 위한 환경을 구성했으며, 수집된 실시간 데이터를 Druid와 Elasticsearch에 각각 동시에 적재하여 처리 단계에만 차이를 두어 비교 테스트를 진행했습니다.



3. 모니터링 및 성능개선

I. 실시간 데이터 분석 시스템 구축

성능 비교 테스트에 사용한 분석 쿼리는 Random Access와 Group by, Ranking 분석 쿼리를 이용했습니다.

성능분석 쿼리			
	구분	Druid	Elasticsearch
1	Random Access -a	SELECT ISOCD, __time, parkingBikeTotCnt, rackTotCnt, shared, stationId, stationLatitude, stationLongitude, stationName FROM seoulbike_raw WHERE __time >= '2023-03-08 00:00:00' AND stationId = 'ST-292'	SELECT ISOCD, parkingBikeTotCnt, rackTotCnt, shared, stationId, stationLatitude, stationLongitude, stationName, timestamp FROM seoulbike WHERE timestamp.keyword >= '2023-03-08 00:00' AND stationId = 'ST-292'
2	Random Access -b	SELECT count(*) FROM seoulbike_raw WHERE __time >= '2023-03-08 00:00:00' AND stationId = 'ST-292'	SELECT count(*) FROM seoulbike WHERE timestamp.keyword >= '2023-03-08 00:00' AND stationId = 'ST-292'
3	Group By	SELECT stationId , COUNT(*) FROM seoulbike_raw WHERE __time >= '2023-03-08 00:00:00' GROUP BY stationId	SELECT stationId , COUNT(*) FROM seoulbike WHERE timestamp.keyword >= '2023-03-08 00:00' GROUP BY stationId
4	Ranking	SELECT stationId, count(*) FROM seoulbike_raw WHERE __time >= '2023-03-08 00:00:00' GROUP BY stationId ORDER BY COUNT(*) DESC LIMIT 20	SELECT stationId, count(*) FROM seoulbike WHERE timestamp.keyword >= '2023-03-08 00:00' GROUP BY stationId ORDER BY COUNT(*) DESC LIMIT 20

3. 모니터링 및 성능개선

I. 실시간 데이터 분석 시스템 구축

성능 테스트 결과 Druid 가 전반적으로 우세한 쿼리 성능을 보였으며,
Elasticsearch는 Group By 와 Ranking 부문에서 쿼리결과가 Timeout될 정도로 성능이 미흡했습니다.

성능분석 결과

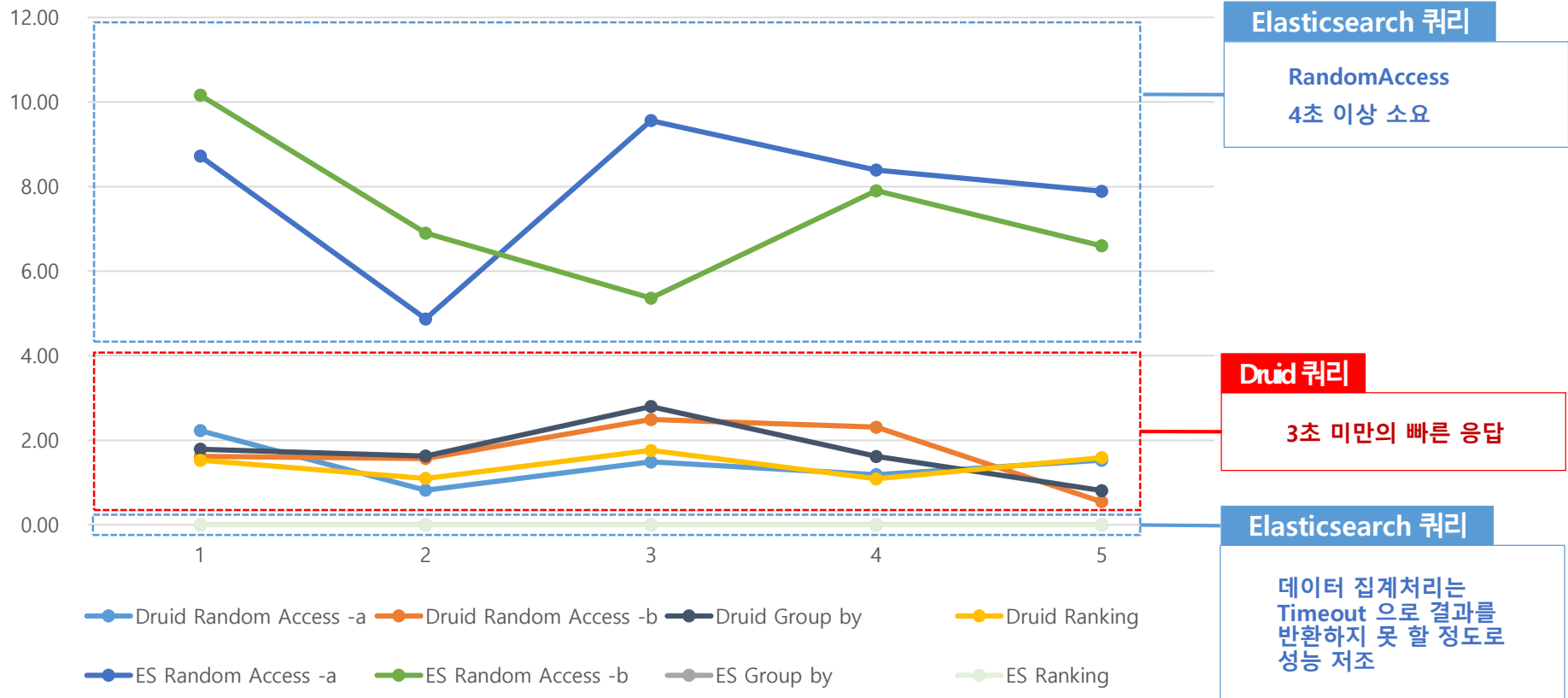
	구분	Druid					Elasticsearch				
		1 st	2 nd	3 rd	4 th	5 th	1 st	2 nd	3 rd	4 th	5 th
1	Random Access -a	2.23	0.82	1.49	1.19	1.53	8.72	4.87	9.56	8.39	7.89
2	Random Access -b	1.62	1.57	2.49	2.31	0.55	10.16	6.90	5.36	7.90	6.60
3	Group By	1.79	1.63	2.80	1.62	0.81	error (issue-1002)	error (issue-1002)	error (issue-1002)	error (issue-1002)	error (issue-1002)
4	Ranking	1.53	1.10	1.76	1.09	1.59	error (issue-1002)	error (issue-1002)	error (issue-1002)	error (issue-1002)	error (issue-1002)

* 단위 : 초(s)

3. 모니터링 및 성능개선

Druid는 분석용 DB로 데이터 수집과 집계에 특화되고 광범위하고 빠른 질의기능을 지원하는 반면, Elasticsearch는 분석에 필요한 기본 기능들에 약하여 분석 용도로는 Druid가 더 특화됨을 확인했습니다.

Druid vs ElasticSearch 분석 쿼리 성능 비교



목차

I. 실시간 데이터 분석 시스템 구축

1. 개요
2. 아키텍처
3. 모니터링 및 성능개선

II. Data lineage Pipeline prototype 개발

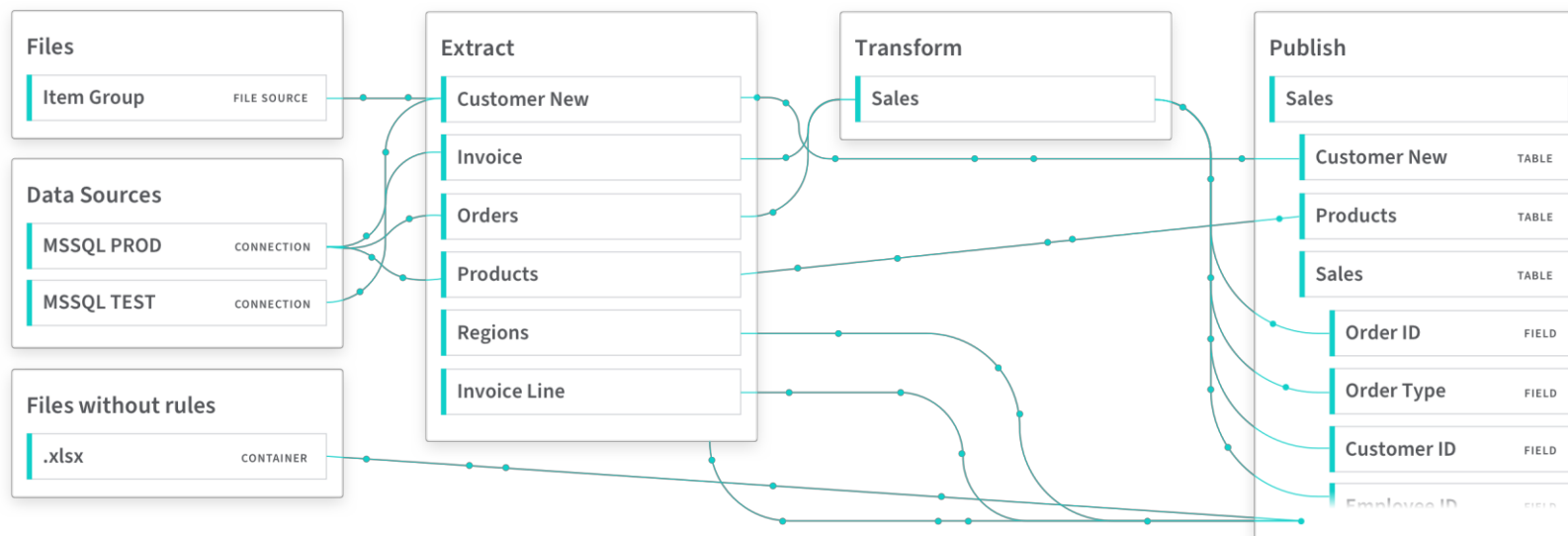
1. 개요
2. 아키텍처
3. Pipeline 구성 작업

III. 블록체인 서비스 데이터 Pipeline 개발 및 운영

1. 개요
2. 아키텍처
3. Pipeline 구성 작업

1. 개요

Data lineage(데이터 계보)란, 데이터 흐름을 시각화하여 특정 테이블이 어떤 테이블을 참조하는지, 데이터가 어디에서 와서 어디로 흘러가는지 편리하게 알 수 있는 데이터 관리 방법론입니다.

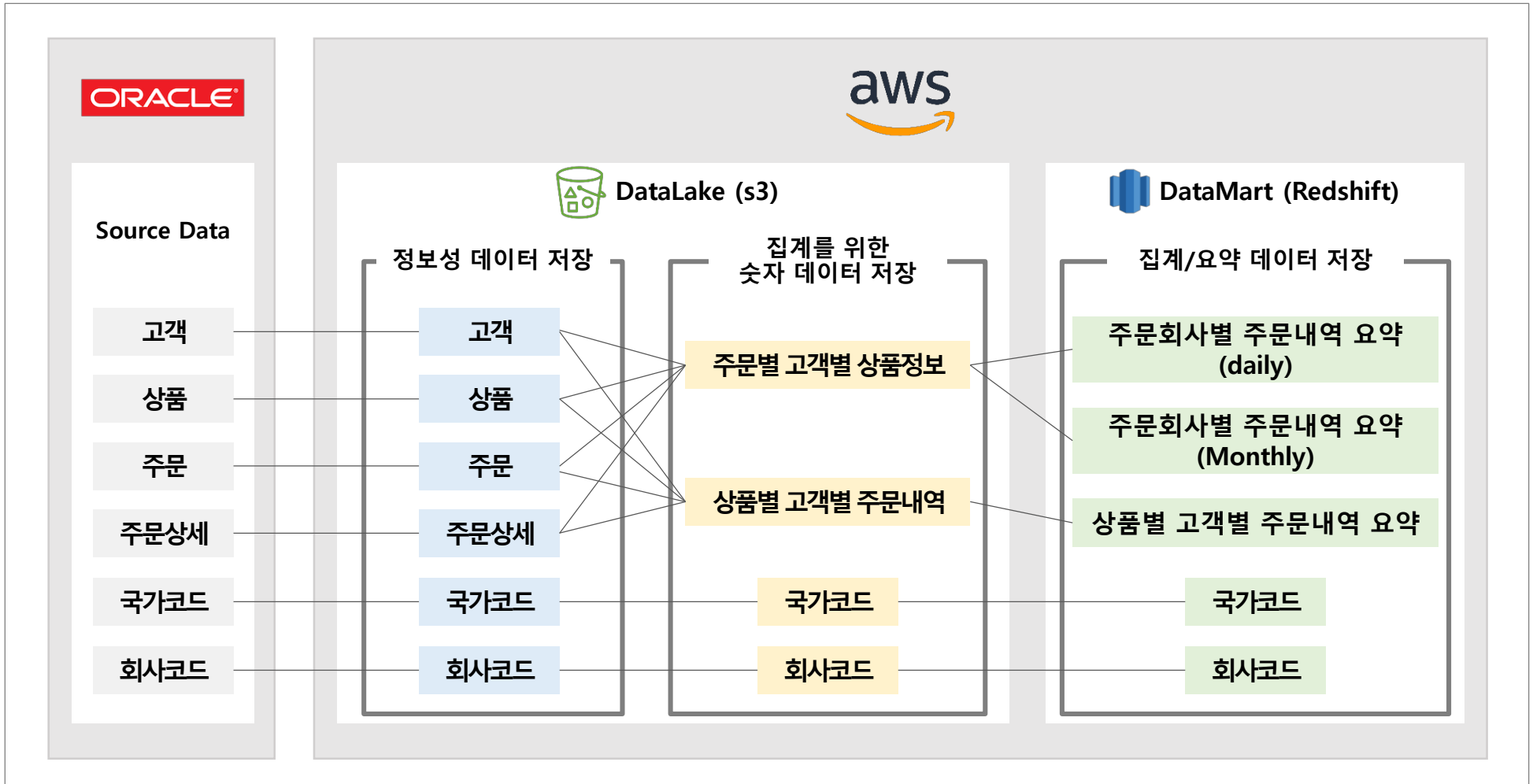


Data lineage 주요 이점

1. Big Data 및 클라우드 환경에 대한 데이터 흐름 가시화
2. 데이터 생성, 변경, 이동 등 전체 데이터의 생명주기 관리 가능
3. 데이터 프로세스 이상을 발견, 추적 및 수정 가능
4. 데이터 변경 시 영향도 파악을 통한 변경에 따른 위험도 감소
5. 조직 전체 데이터의 신뢰성 제고

2. 아키텍처

Data lineage 기능 구현을 위해, 데이터 테이블들의 변환과정을 파악할 수 있는 Pipeline Prototype을 개발했습니다.



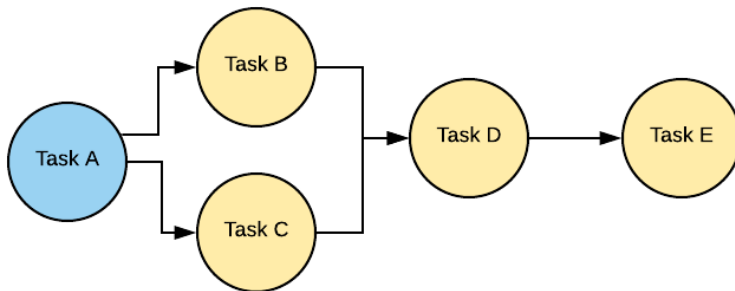
3. Pipeline 구성 작업

II. Data lineage Pipeline prototype 구성

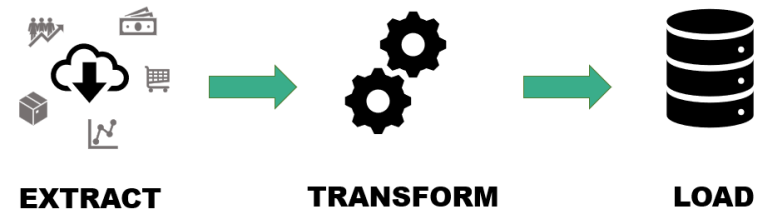
데이터의 추출, 변환, 적재 작업(ETL JOB)은 Spark 솔루션을 활용하고, 데이터 처리 작업에 대한 스케줄링은 Airflow 솔루션을 이용해 Pipeline을 구성했습니다.



Workflow



ETL JOB



* Workflow :
Airflow에서 실행할 데이터 처리와 관련한 Task들에 대해
실행순서, Task간의 dependency 와 Task 실행 스케줄링의 정보를 담고 있다.

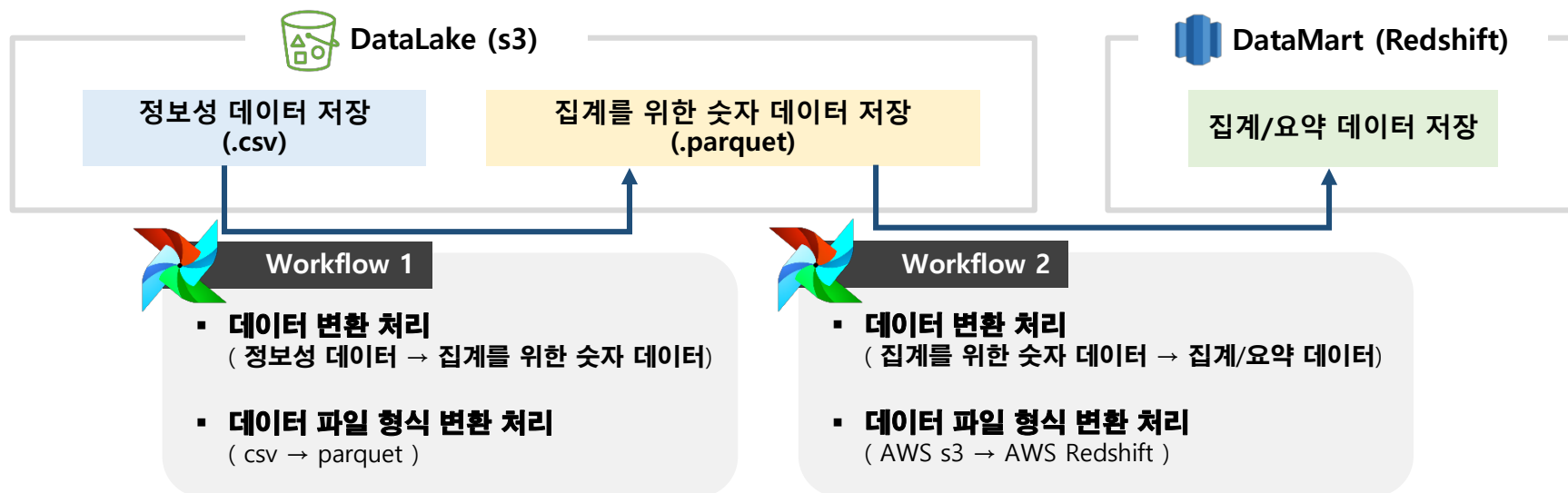
3. Pipeline 구성 작업

데이터 흐름 추적에 대한 상세한 시각화를 위해, Airflow의 Workflow를 데이터 저장위치에 따라 나누어 구성했습니다.

✓ 주요 고려 사항

① 데이터 흐름에 대한 상세한 시각화가 가능하도록 할 것

→ 데이터 처리 작업을 실행할 순서에 맞게 구성된 **Workflow를 데이터 저장위치 변화에 따라 나누어 구성**



3. Pipeline 구성 작업

각 데이터 테이블별 원천 데이터 및 참조 데이터 추적을 용이하게 하기위해 데이터 변환 처리와 관련한 작업을 세분화하여 만들었습니다.

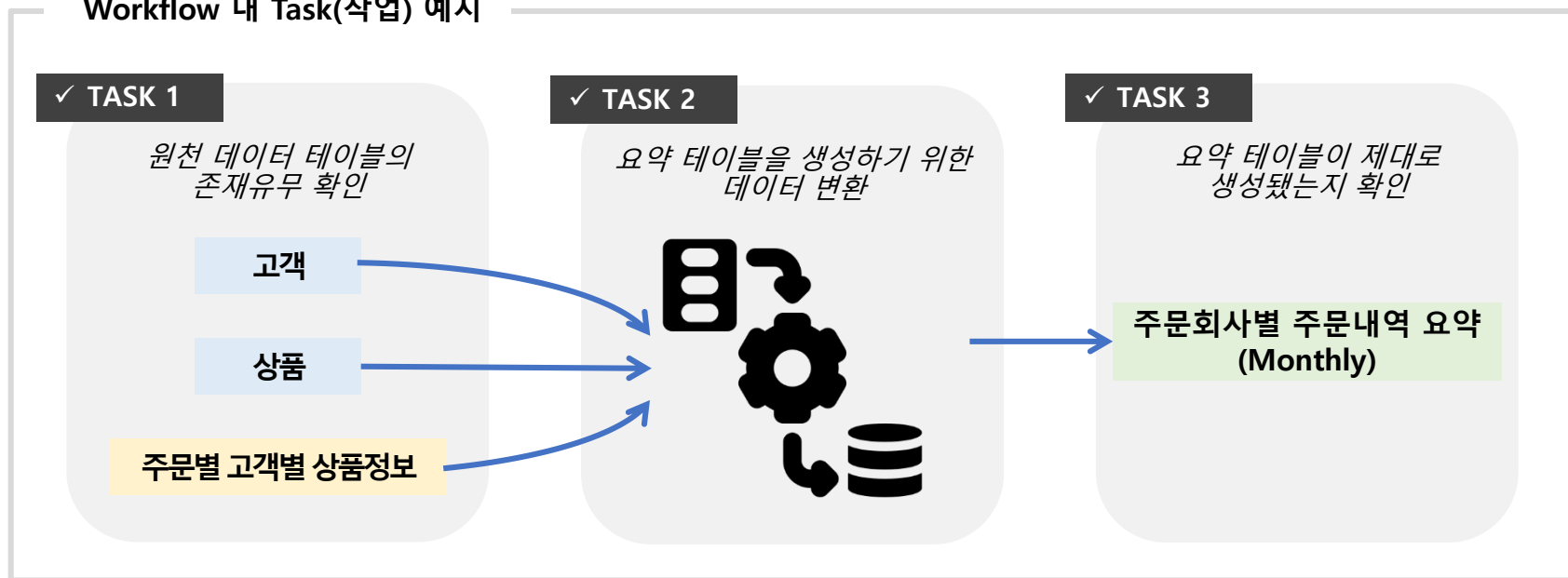
✓ 주요 고려 사항

- ② 집계를 위한 숫자데이터 테이블이나 요약 테이블에 대한 원천 데이터 추적이 용이할 것

→ 데이터 변환 처리 Task(작업)을

원천 데이터 존재 확인, 데이터 변환, 타겟 데이터 생성확인 순으로 세분화하여 만들

Workflow 내 Task(작업) 예시



3. Pipeline 구성 작업

▪ Airflow DAG Task 구성코드 예시

```
## 1) COUNTRY_DIM : SoR -> EDW
with TaskGroup(group_id='grp_COUNTRY_DIM') as grp_COUNTRY_DIM:

    ##1. SoR COUNTRY_DIM Data wait
    t_wait_SoR_COUNTRY_DIM = S3KeySensor(
        task_id='t_wait_SoR_COUNTRY_DIM',
        bucket_key=f'{SOR_DATA_DIR}/COUNTRY_DIM/*.csv',
        wildcard_match=True,
        mode='reschedule',
        poke_interval=30,
        timeout=60*60*2,
    )

    ##2. EMR add step country dim SoR -> EDW
    t_run_wl_country_dim_edw = EmrAddStepsOperator(
        task_id='t_run_wl_country_dim_edw',
        job_flow_id=EMR_ID,
        steps=[{
            'Name': f'wl_country_dim_edw > {STRD_DT}',
            'ActionOnFailure': 'CONTINUE',
            'HadoopJarStep': {
                'Jar': 'command-runner.jar',
                'Args': ['spark-submit', f'{WL_SOURCE_DIR}/wl_country_dim_edw.py']
            }
        }],
    ),

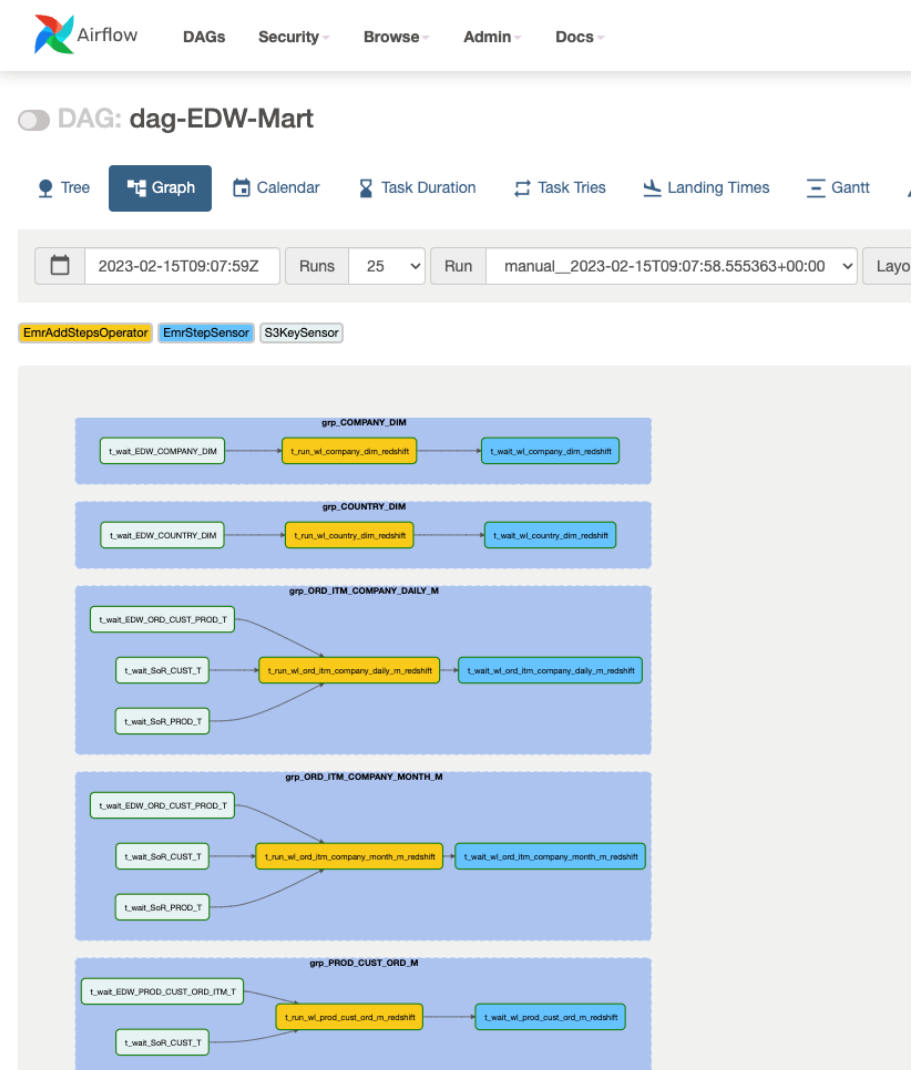
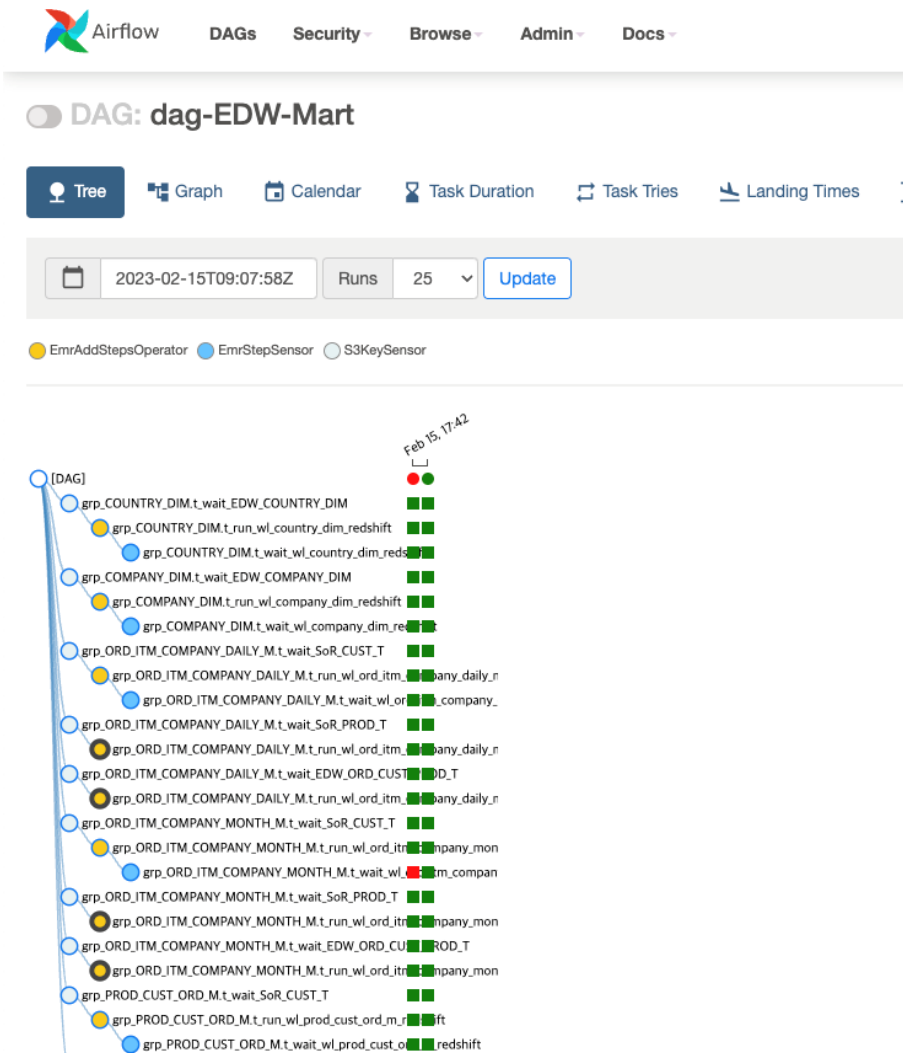
    ##3. EMR step sensor country dim SoR -> EDW
    t_wait_wl_country_dim_edw = EmrStepSensor(
        task_id='t_wait_wl_country_dim_edw',
        job_flow_id=EMR_ID,
        step_id="{{ task_instance.xcom_pull(task_ids='grp_COUNTRY_DIM.t_run_wl_country_dim_edw', key='return_value')[0] }}",
        mode='reschedule',
        poke_interval=30,
        timeout=60*30,
    )

t_wait_SoR_COUNTRY_DIM >> t_run_wl_country_dim_edw >> t_wait_wl_country_dim_edw
```

3. Pipeline 구성 작업

II. Data lineage Pipeline prototype 구성

■ Airflow DAG 예시



목차

I. 실시간 데이터 분석 시스템 구축

1. 개요
2. 아키텍처
3. 모니터링 및 성능개선

II. Data lineage Pipeline prototype 개발

1. 개요
2. 아키텍처
3. Pipeline 구성 작업

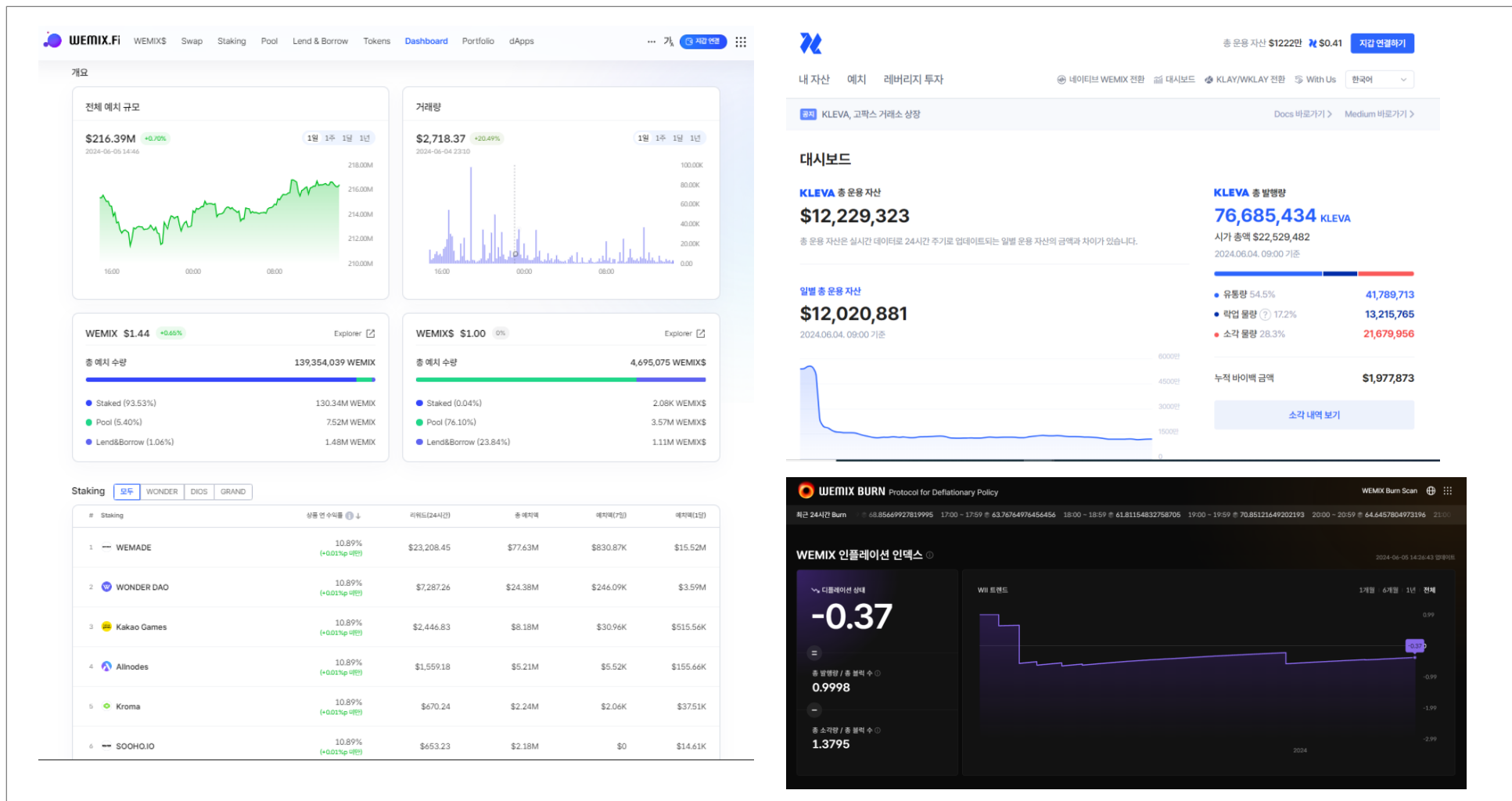
III. 블록체인 서비스 데이터 Pipeline 개발 및 운영

1. 개요
2. 아키텍처
3. AWS -> Azure 이관

1. 개요

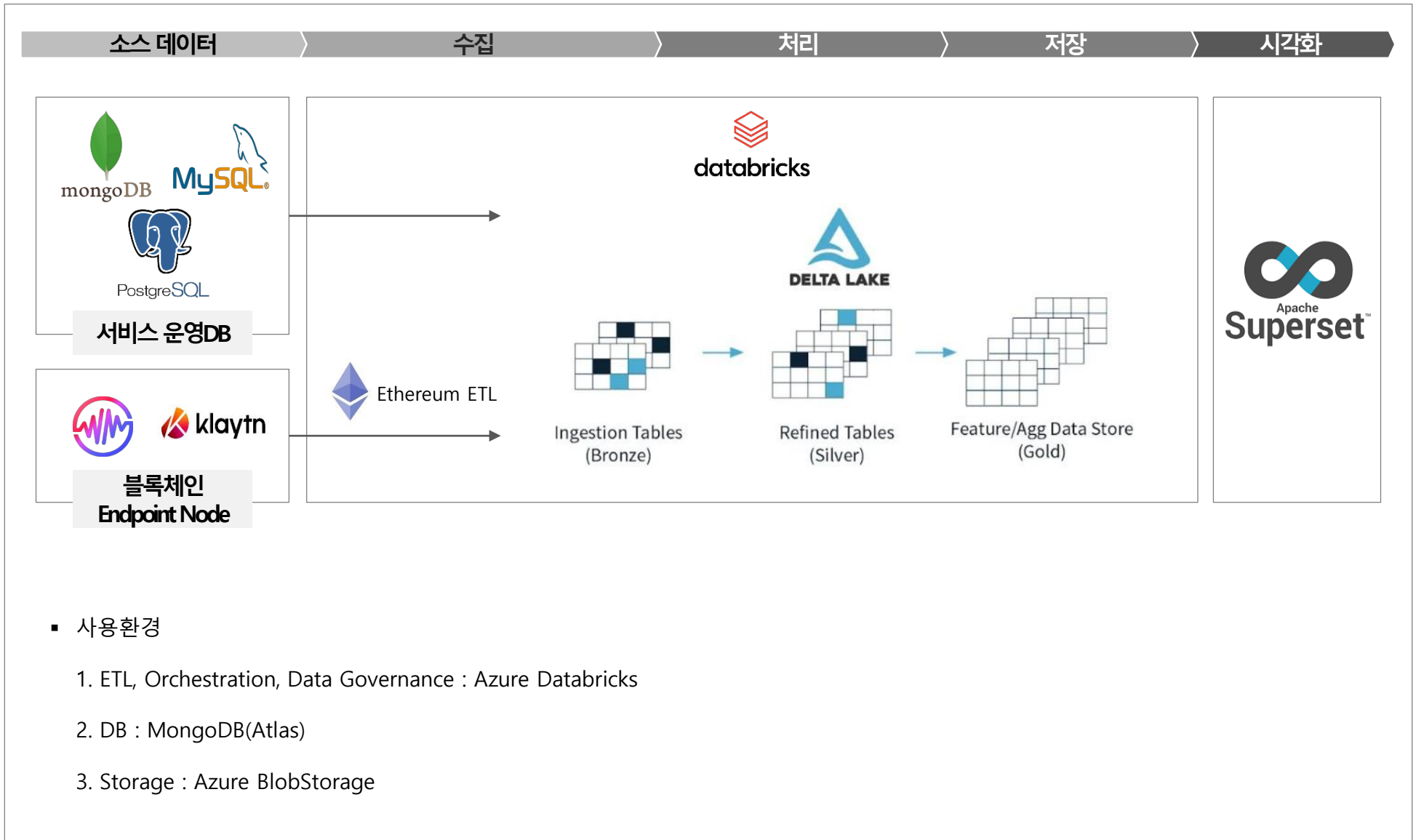
Ⅲ. 블록체인 서비스 데이터 Pipeline 개발 및 운영

Wemix 블록체인 디파이 서비스 KPI 대시보드 데이터 서빙을 위한 파이프라인 개발 및 운영업무 진행



2. 아키텍처

Ⅲ. 블록체인 서비스 데이터 Pipeline 개발 및 운영










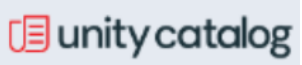







3. AWS -> Azure 이관

Ⅲ. 블록체인 서비스 데이터 Pipeline 개발 및 운영

✓ 주요 고려 사항

- ① 기존 파이프라인 환경 기술 대응
- ② 운영 관리 용이 (운영 서비스에 비해 관리 인원이 적기 때문)
- ③ 데이터 품질 이슈 해소

구분	AS-IS (AWS)	TO-BE (Azure)
ETL	 	  
Orchestration	 	
Data Catalog		
DW/DM		 
Storage		

THANK YOU