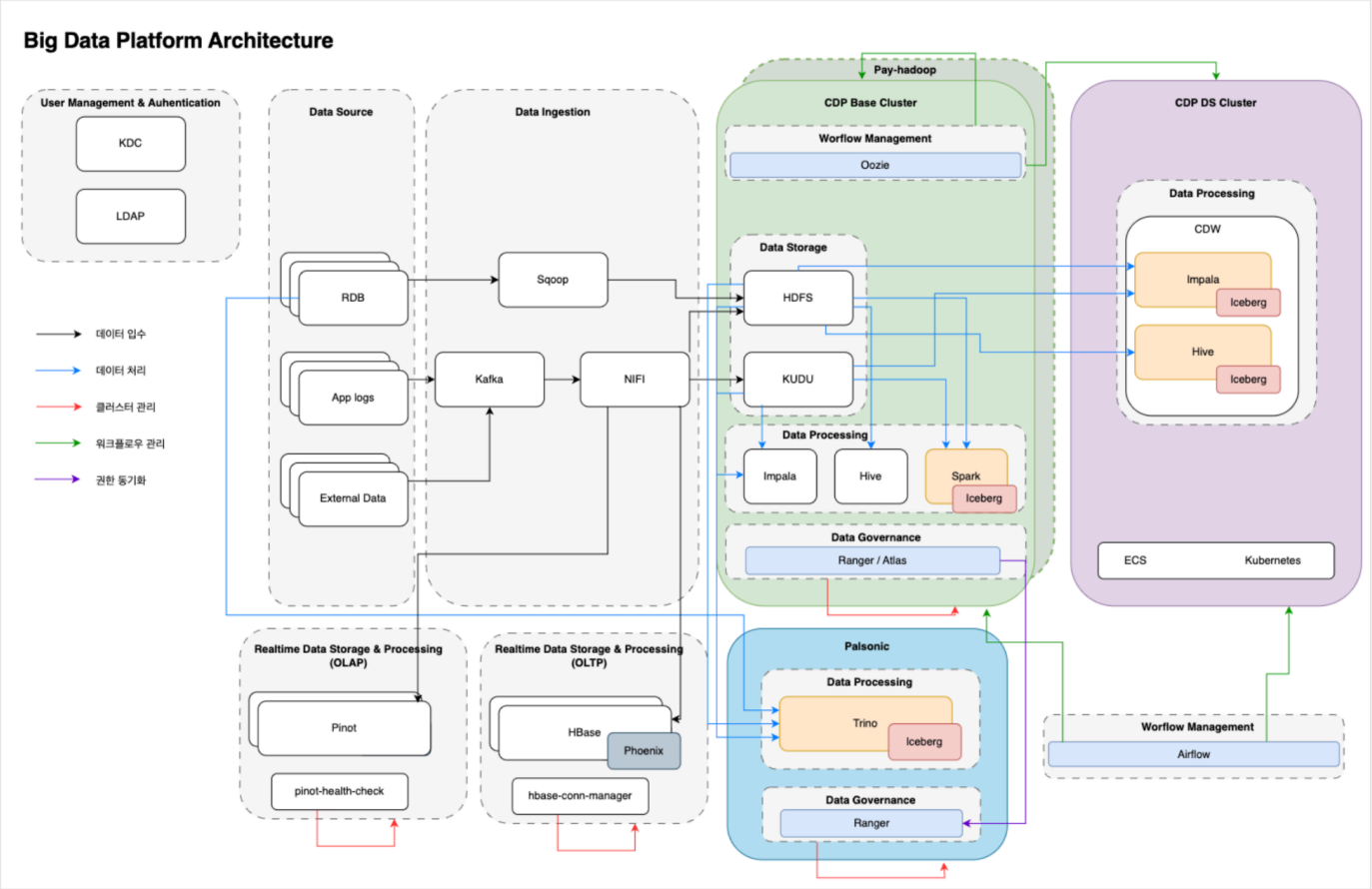
카카오페이 데이터플랫폼 아키텍처



하둡플랫폼은 크게 하둡 클러스터, HBase/Phoenix 클러스터, Trino(트리노) 클러스터, Pinot(피노) 클러스터를 제공하고 있습니다.

클러스터	용도	사용기술
하둡 클러스터 (CDP Base, CDS)	대용량 데이터 분산저장, 배치 처리	HDFS, Impala, Kudu, Spark, Hive, Iceberg 등
HBase/Phoenix 클러스터	컬럼 기반 데이터 분산저장 (NoSQL), 준실시간	HBase, Phoenix, HDFS 등
Trino 클러스터	다양한 데이터 소스(하둡, Mysql, NoSQL 등)에 분산된 데이터를 SQL 기반으로 분석	Trino
Pinot 클러스터	실시간 데이터를 SQL 기반으로 빠르게 분석 (실시간 OLAP)	Pinot

영역	기술 스택	역할
수집	Kafka, NiFi, Sqoop	실시간/배치 혼합 수집을 안정화, 오류 복구·재시도·스루풋 보장
저장소(배치/준실시간/OLTP)	HDFS, Kudu, HBase/Phoenix, Iceberg	워크로드 특성별 저장: 대용량·업서트·키밸류·테이블 포맷 분업
처리/쿼리	Spark, Impala, Hive, Trino, Pinot	배치/대화형/페더레이션/실시간을 엔진 특화로 최적화
오케스트레이션	Airflow, Oozie	DAG·SLA·재시도·알람 표준화, 운영 안정성
분석/DS	Jupyter, Zeppelin, Hue, Livy	탐색·실험·협업, 셀프서비스 분석
컨테이너 실행	ECS, Kubernetes	학습/서빙/ETL 마이크로서비스 탄력 스케일
가용성/운영	LB, 모니터링/APM	장애 전가·성능 관측, 운영 리스크 감소
보안/인증	Knox, KDC/LDAP, Ranger, Atlas, Hue Proxy	단일 진입점·감사·정책 중앙화 → 컴플라이언스/감사 대응

기술 스택

수집(데이터 인제스트)

Apache Kafka
분산 메시징·스트리밍 플랫폼.
앱 로그/이벤트의 실시간 수집·버퍼링·전달. 이후 Spark Streaming/Trino/Pinot/HBase로 분기.

Apache NiFi
UI 기반 데이터 플로우 오케스트레이션/ETL 도구.
외부 API/파일/DB 등 다양한 소스의 배치·준실시간 취합, 포맷 변환 및 라우팅, 오류 재시도·백프레셔·관측성 확보.

Sqoop
RDBMS ↔ HDFS/Hive 간 대용량 배치 전송.
RDB 스냅샷/증분 적재(배치) 파이프라인.

저장소

HDFS (Hadoop Distributed File System)
분산 파일 시스템(대용량 배치/분석 저장소).
비식별 분석 데이터의 주 저장소. 장기간 보관·대용량 스캔 워크로드에 최적.

Apache Kudu
고속 컬럼 저장소(업서트/저지연 스캔) – HDFS와 HBase의 중간 지점.
준실시간 분석(업데이트/증분이 잦고 지연 민감) 테이블 저장. 대시보드/운영지표/서빙형 분석에 적합.

Apache HBase (+ Phoenix)
분산 키-밸류 스토어(HBase)와 SQL on HBase(Phoenix).
OLTP 성격의 실시간 조회/쓰기, 세그먼트 캐시, 피처 스토어 성격의 접근에도 활용.

Apache Iceberg
오픈 테이블 포맷(스냅샷/스키마 진화/파티션 진화/타임트래블).
HDFS/오브젝트스토리지의 테이블 단위 트랜잭션·메타 관리를 제공, Spark/Trino/Impala와 상호운용.

HDFS, Kudu, Hbase, iceberg 설명

HDFS 외에 카카오페이가 사용하는 **Kudu, HBase, Iceberg**는 모두 HDFS와 깊은 관련이 있지만, 그 관계와 역할은 각기 다릅니다.

세 가지 모두 HDFS와 직접적인 데이터 저장 역할은 구분되지만, **HBase**와 **Iceberg**는 HDFS를 기반 스토리지로 활용하는 소프트웨어 레이어에 가깝고, **Kudu**는 HDFS와 독립적인 분산 저장소입니다.

---

1. HBase (Hadoop Database)

HBase는 HDFS 위에 구축된 **(on top of HDFS)** 분산형 NoSQL 데이터베이스입니다.

- **HDFS**와의 관계**:** 기반 소프트웨어 레이어 **(On top of HDFS)**
  - HBase는 데이터를 **HDFS**의 파일**(HFile)** 형태로 저장합니다. 즉, HBase는 자체적으로 물리적인 스토리지를 관리하지 않고, 데이터의 영구 저장소로 HDFS를 사용합니다.
  - HDFS가 제공하는 고가용성 및 대용량 저장 능력을 활용합니다.
- 주요 역할**:** HDFS의 일괄 처리**(Batch Processing)** 중심의 단점을 보완하여, HDFS에 저장된 데이터에 대해 실시간 무작위 읽기**/**쓰기**(Random Read/Write)** 접근을 제공합니다. 이는 HBase가 구글의 **BigTable**을 모델로 한 키-값(Key-Value) 기반의 컬럼 지향 데이터베이스이기 때문에 가능합니다.

---

2. Apache Iceberg (Table Format)

Iceberg는 HDFS와 같은 분산 파일 시스템 위에 테이블 구조를 정의하는 개방형 **테이블 포맷(Table Format)**입니다.

- **HDFS**와의 관계**:** 기반 소프트웨어 레이어 **(Metadata Layer on Storage)**
  - Iceberg는 HDFS를 포함한 S3, ADLS 등 객체 스토리지 위에 데이터 파일(Parquet, ORC 등)을 저장합니다.
  - Iceberg는 데이터 파일 자체를 관리하는 것이 아니라, 파일들의 위치, 스키마, 파티셔닝 변경 이력 등을 담은 메타데이터를 관리합니다. 이 메타데이터를 통해 ACID 트랜잭션과 시간 여행**(Time Travel)** 같은 데이터 웨어하우스의 기능을 데이터 레이크 상에 구현합니다.
- 주요 역할**:** HDFS와 같은 단순 파일 저장소에 저장된 데이터를 마치 전통적인 SQL 테이블처럼 안정적으로 관리할 수 있게 해주는 최상위 관리 계층입니다.

---

3. Apache Kudu

Kudu는 HDFS와 독립적인**(Separate)** 분산 스토리지 엔진입니다.

- **HDFS**와의 관계**:** 별개의 저장소 **(Separate Storage Engine)**
  - Kudu는 HDFS처럼 자체적인 데이터 저장 및 복제 메커니즘을 가지고 있습니다. 즉, HDFS에 의존하지 않고 별도의 **Master Server**와 **Tablet Server**로 클러스터를 구성합니다.
  - 다만, Kudu는 Hadoop 생태계의 일부로, HDFS 클러스터와 동일한 물리적 노드에서 함께 구동될 수 있어 상호 보완적으로 사용되는 경우가 많습니다.
- 주요 역할**:** HDFS의 장점인 **고처리량 순차 읽기(High-throughput Sequential Read)**와 HBase의 장점인 **빠른 무작위 접근(Fast Random Access/Mutations)**을 동시에 제공하는 컬럼 기반의 저장소입니다. 빈번한 업데이트가 필요한 시계열 데이터나 OLAP 워크로드에 적합합니다.

저장소	**HDFS** 기반 여부	저장 방식	주요 역할
HBase	**HDFS** 기반 (파일 저장소로 사용)	키-값 기반 NoSQL DB	HDFS 데이터에 실시간 무작위 읽기**/**쓰기 제공
Iceberg	**HDFS** 기반 (객체**/**파일 저장소로 사용)	파일 시스템 위에 테이블 포맷 관리	데이터 레이크에 **ACID** 트랜잭션 및 메타데이터 관리 기능 부여
Kudu	**HDFS**와 별개 (자체 분산 저장소)	컬럼 지향 저장소	빠른 업데이트와 효율적인 분석을 동시에 제공

처리 엔진(배치/대화형/실시간)

Apache Spark

범용 분산 처리 엔진(배치·스트리밍·ML).
대규모 ETL, 모델 학습/서빙 파이프라인, Iceberg/Delta 스타일 레이크 테이블 관리. Yarn/Standalone/K8s에 배치됨.

Apache Hive

SQL-on-Hadoop 생태의 원조(메타스토어/쿼리 레이어).
메타스토어(카탈로그)로 테이블 스키마 관리, 배치성 쿼리/ETL의 오케스트레이션 축.

Apache Impala (CDW 포함)

MPP 기반의 저지연 대화형 SQL 엔진.
BI/애드혹 분석에 초저지연 응답 제공(특히 Parquet/Kudu/ORC). 피크 타임 쿼리 급증 시 리소스 보호 정책 필요.

Trino

페더레이티드 쿼리 엔진(다중 소스 조인/쿼리).
Hive/Iceberg/HBase/MySQL/NoSQL 등 이기종 소스를 단일 SQL로 통합 분석. 탐색/데이터 고객사 POC에 적합.

Apache Pinot

실시간 OLAP(초저지연) 분석 DB.
대시보드·서비스 실시간 지표(초·수초 단위 레이턴시) 제공. Kafka/Spark로 적재.

YARN (Spark on YARN)

하둡 리소스 스케줄러.
Spark/MapReduce/Kudu 등 클러스터 자원(메모리/CPU) 할당·격리·큐 정책.

오케스트레이션 / 워크플로

Apache Airflow / Oozie

ETL/ML 파이프라인 스케줄러(의존성·재시도·알람).
배치/스트리밍 잡의 엔드투엔드 DAG 관리, 데이터 품질 점검과 SLA 모니터링 연계.

데이터 사이언스·분석 툴

Jupyter / Zeppelin

노트북형 분석/탐색 환경.
인터랙티브 SQL/파이썬/스칼라 분석, 실험/시각화, 모델 프로토타이핑.

Hue

SQL 에디터(Hue)
데이터 탐색·파일 브라우징을 브라우저에서 수행.

인프라

ECS / Kubernetes(DS Cluster)

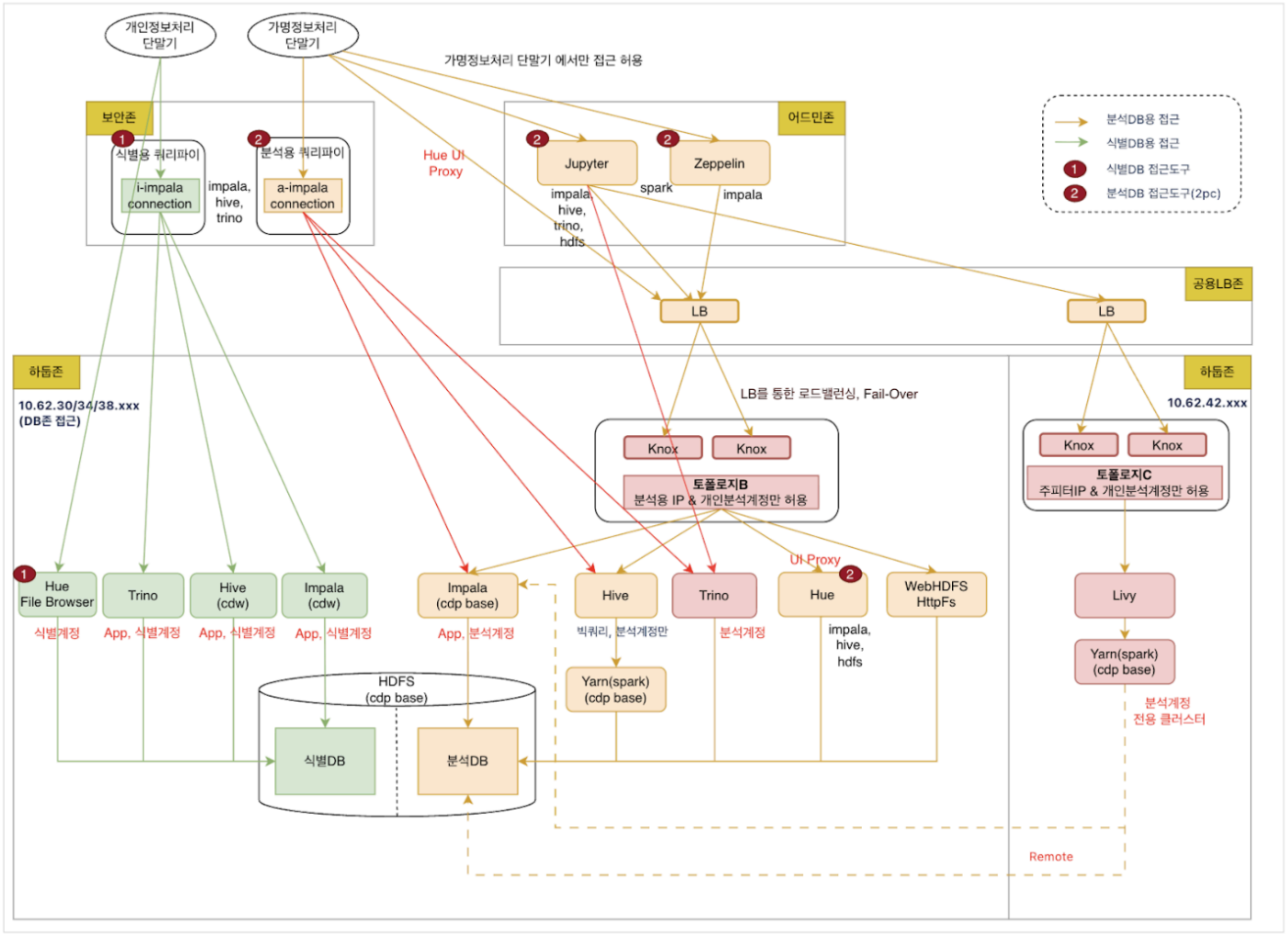
컨테이너 실행/오토스케일 인프라.
모델 학습/서빙 마이크로서비스, 파이프라인 러너, 피처/배치 잡의 탄력적 스케일링.

보안

아키텍처

가명데이터의 저장소, 처리/조회와 관련된 모든 환경을 식별DB와 분리된 환경으로 구축하는것이 권장되나

- 비용을 최소화하기 위해 Apache Knox 게이트웨이를 도입하여 식별DB 접근을 차단한다.
  - 저장소 분리는 신규 HW 및 데이터 마이그레이션 비용이 많이 들기 때문에, 논리적 분리로 사용한다. (김앤장 컨설팅을 통해 저장소는 논리적 분리 가능함을 확인 완료)
  - 분석DB용 임팔라 비용 최소화를 위해 배치용, 개인 쿼리용으로 함께 사용하고 개인접근 환경에서는 배치용 계정으로 접근할 수 없도록 Knox로 차단한다.
- 쿼리파이는 Apache Knox를 지원하지 않아 자체 정책으로 식별DB 접근을 차단한다.
- 클라우데라 Knox에서 Trino를 지원하지 않아 분석DB용 Trino를 신규 구축하고 자체 정책으로 식별DB 접근을 차단한다.
- 하둡존 Yarn(Spark) 환경은 DB존 ACL이 허용되어 있어, 분석DB용 Yarn(Spark) 클러스터를 신규 구축하여 개인신용정보 접근 경로를 차단한다.
- Hue는 어드민존에서 제공되어야 하나, 클라우데라 솔루션이 같은 네트워크존을 요구함에 따라 하둡존에 위치시키고 Knox 를 이용하여 UI 를 Proxy한다.



보안·접근 제어 / 인증

Apache Knox

하둡 에코시스템(HTTP 기반 서비스)에 대한 경계 보안 게이트웨이(Reverse Proxy).
외부/내부 사용자의 WebHDFS, Hive/Impala/Trino UI, Hue, Zeppelin, Jupyter, Livy 접근을 단일 진입점으로 묶고 인증/권한/감사를 중앙화. 네트워크 경계와 감사 추적을 강제해 개인정보 영역/분석 영역 분리를 보장.

Hue Proxy

Hue 및 관련 웹툴 접근 트래픽을 Knox 앞단/옆단에서 프록시.
데이터 이동 경로를 통제하고 로깅/정책 연계를 단순화(특히 식별/비식별 영역 경계 통제).

KDC / LDAP

Kerberos(KDC)와 조직 계정 관리(LDAP).
하둡/스파크/툴들의 SSO·서버 간 신뢰 기반 인증. Knox, Ranger, Atlas, Hive/Impala 등과 연계해 사용자·서비스 프린시펄을 관리.

Apache Ranger

데이터 접근 제어 정책(테이블/컬럼/로우/오브젝트 수준) 관리.
Hive/Impala/Trino/HBase/Kafka 등 엔진별 ABAC/RBAC 정책을 중앙에서 집행하고 감사 로그(Who/What/When)를 수집.

Apache Atlas

메타데이터/데이터 계보(Lineage) 관리.
데이터 흐름 추적(원천→ETL→테이블/뷰→리포트/모델), 데이터 카탈로그 제공, 거버넌스/컴플라이언스 증거화.

카카오페이 데이터플랫폼 운영 분석

전체 개요 : 보안과 분석을 동시에 만족하는 하이브리드 데이터 플랫폼

카카오페이의 데이터 플랫폼은 금융 서비스의 특성상, 보안·컴플라이언스·확장성을 모두 충족시키기 위해 설계된
보안 분리형 하둡(Hadoop) 기반 빅데이터 플랫폼입니다.

핵심 설계 철학은 다음 세 가지입니다:
보안 분리 (Security Isolation) : 식별정보(DB)와 비식별 분석(DB)을 논리적으로 완전 분리
이중 게이트웨이 구조 (Knox 기반 Access Control) : 모든 접근은 Knox Proxy를 통한 중앙 통제
하이브리드 처리 구조 (Batch + Realtime + ML) : Hadoop 기반 대규모 배치와 Spark/Trino 기반 실시간 분석 병행

플랫폼 구성 요소 요약

계층	주요 구성	설명
데이터 수집 (Ingestion)	Kafka, NIFI, Sqoop	RDB, 로그, 외부데이터 실시간/배치 수집

데이터 저장 (Storage)	HDFS, Kudu, HBase	대용량 배치 / 준실시간 / OLTP 데이터 저장
데이터 처리 (Processing)	Spark, Hive, Impala, Trino, Iceberg	분석·가공 및 모델링, 다양한 데이터 소스 통합 분석
데이터 거버넌스	Ranger, Atlas	접근제어 및 메타데이터 관리
워크플로/오케스트레이션	Oozie, Airflow	ETL 및 ML 파이프라인 스케줄링
보안·접근제어	Apache Knox, Hue Proxy	사용자 인증, 접근 통제, 감사 로그 일원화
시각화/분석 도구	Zeppelin, Jupyter, Hue	분석가, 엔지니어, 데이터 사이언티스트용 인터페이스

보안 및 접근 통제 아키텍처

Apache Knox 기반 이중 경로 구조로 외부 접근을 차단하고, 내부 접근을 Proxy 기반으로 제어합니다.

- Hue, Zeppelin, Jupyter 등은 모두 Knox를 통해서만 접근 가능
- 개인정보 영역(식별DB) 과 분석 영역(비식별DB) 은 완전히 분리
- Spark ACL, Yarn ACL, Impala ACL 정책으로 세분화된 접근 통제 수행
- Hue Proxy가 데이터 이동 경로를 중앙집중식으로 제어

즉, 데이터 저장소 분리 + 접근 경로 통제 + ACL 기반 Role Control의 삼중 보안 구조로 운영

저장소·**CPU**·메모리 운영 현황 및 증가 추이

HDFS 저장소 용량

항목	2024.12	2025.03	2025.06	2025.09
HDFS 사용량 (TB)	3,316	6,120	8,059	8,837
전체 용량 (TB)	11,151	12,083	14,991	21,268
사용률	51% → 79%	97.8%	70.6% (증설 후 안정화)	
1년 예상 사용률	316% → 103%	(추가 증설 필요 예상)		

분석 :
9개월간 약 5.5PB 증가 (3,316 → 8,837TB)
월 평균 증가량: 약 615TB
일 평균 증가량: 약 20.5TB/day
→ 데이터 증가 속도가 매우 빠르며, 2025년 말에는 HDFS가 한계에 도달할 가능성이 높습니다.
→ HDFS의 “70% 사용 기준”에 근접 중이므로 노드 증설 또는 압축/아카이빙 정책이 필요합니다.

데이터 증가 트렌드 요약

항목	증감 구간	증가량	일 평균 증가량
HDFS 데이터	3,316TB → 8,837TB	▲5,521TB	20.5TB/day
Kudu 데이터	724TB → 2,058TB	▲1,334TB	5.0TB/day
총 데이터 증가량	약 ▲6.8PB	25~26TB/day (평균 기준)	

요약 :
하루 약 25TB 수준의 데이터가 신규 적재되고 있으며,
이 중 약 80%는 비식별 분석 데이터(HDFS),
20%는 실시간 처리 데이터(Kudu, Kafka) 로 분류됩니다.
향후 AI/ML 워크로드 확대 시 1일 30TB 이상 증가 추세 예상.

클러스터		빅플3.0 분석									
날짜		2024. 12. 1	2025. 1. 1	2025. 1. 16	2025. 2. 25	2025. 3. 23	2025. 4. 18	2025. 6. 18	2025. 8. 7	2025. 8. 21	2025. 9. 12
cpu	peak (7D)						88%				
	avg (7D)						25%				
	peak (1D)						82%				100%
	avg (1D)						43%				46%
	평균 사용율			40%	40%		43%				
memory	impala peak (7D)										
	impala avg (7D)										
	impala peak (1D)						70				19.4
	impala avg (1D)						40				
	impala 전체 메모리 (TB)						128				19.8
	impala 메모리 peak사용율			50%	50%		55%				98%
	impala 메모리 30% 확보										
	내년 30% 확보										
	yarn 메모리 peak (7d)			2.5	2.5		5.7				7
	yarn 메모리 avg (7d)						4.3				
	yarn 메모리 전체 용량			4.7	4.7		6.17				7.75
	~~pay-hadoop 사용량 이동 (90% usage 가정)~~			7.8			9.3				90%
	증설 필요 서버 대수 (54G/1대)			**58.4**			**58.9**				
disk	hdfs 사용량	3,316	4,674	5,632	6,646	8,120	6,093	6,871	8,059	8,294	8,837
	kudu 사용량	724	1,020	1,229	1,352	1,495	1,352	1,700	1,987	1,935	2,058
	사용량 합계(TB)	4,040	5,693	6,861	7,997	9,615	7,444	8,571	10,045	10,230	10,895
	일 증가량(TB)		**53.3**	**77.8**	**28.4**	**62.2**	**-83.5**	**18.5**	**29.5**	**13.2**	**30.3**
	사용율		51.1%	61.5%	71.1%	79.6%	49.7%	57.2%	47.2%	48.1%	51.2%
	전체 capa(TB)		11,151	11,151	11,254	12,083	14,991	14,991	21,268	21,268	21,268
	1년 후 예상 용량			**35,270**	**18,369**	**32,328**	**9,639**		**20,810**	**15,035**	**21,938**
	1년 후 예상 사용율			316.28%	163.23%	267.55%	64.30%		97.84%	70.69%	103.15%
	pay-hadoop 사용량 2배 (70% usage 기준)			**16354.7**			**17027.7**				
	pay-hadoop 필요용량 (70% usage 기준)			**16060.2**			**15883.9**				
	필요한 용량 (70% usage 기준)			16060.2			15883.9		29728.2	21478.9	31340.4
	증설 필요 용량			4908.8			892.6		8459.7	210.4	10071.9
	증설 필요 서버 대수 (145T/1대)			**33.9**			**6.2**		**58.3**	**1.5**	**69.5**
	pay-hadoop 재활용 수량			40.0			33.0				
	pay-hadoop재활용 용량 (116T/1대)			4640.0			3828.0				
	클러스터 워커노드 수			90			117				
	증설필요 워커노드 수			46.1			32.5				
	총 워커노드 수			136			**150**				

카카오페이 데이터 플랫폼에서 **Databricks Lakehouse** 전환 전략

기술 구성 / 운영 효율 / 보안 대응 / 비용 절감의 4대 관점

구분	카카오페이 데이터 플랫폼 (현재)	Databricks 데이터 인텔리전스 플랫폼 (Lakehouse)	전환 전략 및 기대 효과
기술 구성	Hadoop/CDP 기반: HDFS, Hive, Impala, Spark, Kudu, Trino, HBase, Pinot 식별/비식별 영역 분리형 설계 (보안 중심 구조) 데이터 수집(NiFi, Kafka), 배치 처리(Airflow, Oozie) 등 이원화 Ranger/Atlas로 정책 및 메타 관리	Lakehouse 통합 아키텍처 (Unity Catalog + Delta Lake/Iceberg 기반) LakeFlow로 배치·스트리밍 파이프라인 통합 Lakebase 트랜잭션 DB: Postgres 기반 초저지연 구조 DB SQL + AI/BI (Genie) 로 BI/애널리틱스 통합 Agent Bricks: 도메인별 AI 서비스 자동화	Hadoop 클러스터 복잡도 제거 데이터/AI/BI/거버넌스 완전 통합 단일 스택에서 데이터 엔지니어링 · AI · 분석 · 거버넌스 구현 가능
운영 효율	100여 노드의 온프레미스 클러스터 관리 필요 Impala/Spark 자원 경쟁, 큐 설정 수동 관리 배치 중심의 느린 워크로드 (수시간 단위) 로그·모니터링·리소스 관리 체계 분리	Serverless/Auto-scaling 컴퓨팅 (DB SQL, Jobs Compute) LakeFlow Designer: 코드 없는 ETL 설계 + 운영 자동화 Photon Engine: Spark 대비 최대 5배 빠른 SQL 처리 통합 리소스 관제 (워크로드별 격리 + 자동 최적화)	운영 리소스 30~50% 절감 ETL 관리 자동화로 DevOps 인력 의존도 감소 실시간 워크로드에도 동일 플랫폼 재활용 가능
보안 대응	Apache Knox + Ranger + LDAP/Kerberos 중심의 분리형 인증체계 식별/비식별 이중 DB 운영으로 보안은 강하나 유연성 낮음 보안 로그 및 감사 정보 시스템별 분리 관리	Unity Catalog 기반 단일 거버넌스 계층 데이터·모델·대시보드 단위 정책, 계보(Lineage), 감사(Audit) 통합 Row/Column-level Policy, PII Masking 지원 Delta Sharing 기반 외부 협업도 보안 유지	분리된 보안 시스템을 단일 정책으로 통합 규제 준수 + 협업 유연성 확보 감사·리니지 자동화로 보안관리 효율 상승
비용 절감	온프레미스 인프라 유지비용 및 HW 교체 주기 부담 HDFS 용량 증가속도 하루 20~25TB (1년 내 증설 필요) Idle 자원 및 중복 클러스터로 인한 낭비	컴퓨팅-스토리지 분리형 아키텍처 (Object Storage 기반) Serverless SQL + Spot Instance로 Idle 비용 제거 Lakebridge: 레거시 웨어하우스 자동 마이그레이션 도구 -업계 최고 수준의 TCO 절감 (최대 60%)	HDFS→Object Storage 전환으로 스토리지 비용 절감 서버리스로 Idle Compute 제거 관리·전력·라이선스 비용 포함 총비용(TCO) 절감

종합 요약

항목	카카오페이 데이터 플랫폼	Databricks Lakehouse 전환 후 기대 효과
구조 복잡도	다중 클러스터, 다계층 인증 구조	단일 통합 플랫폼 (Data + AI + Governance)
운영 효율성	수동 리소스 관리, 배치 위주	Serverless 자동화, 실시간/배치 통합

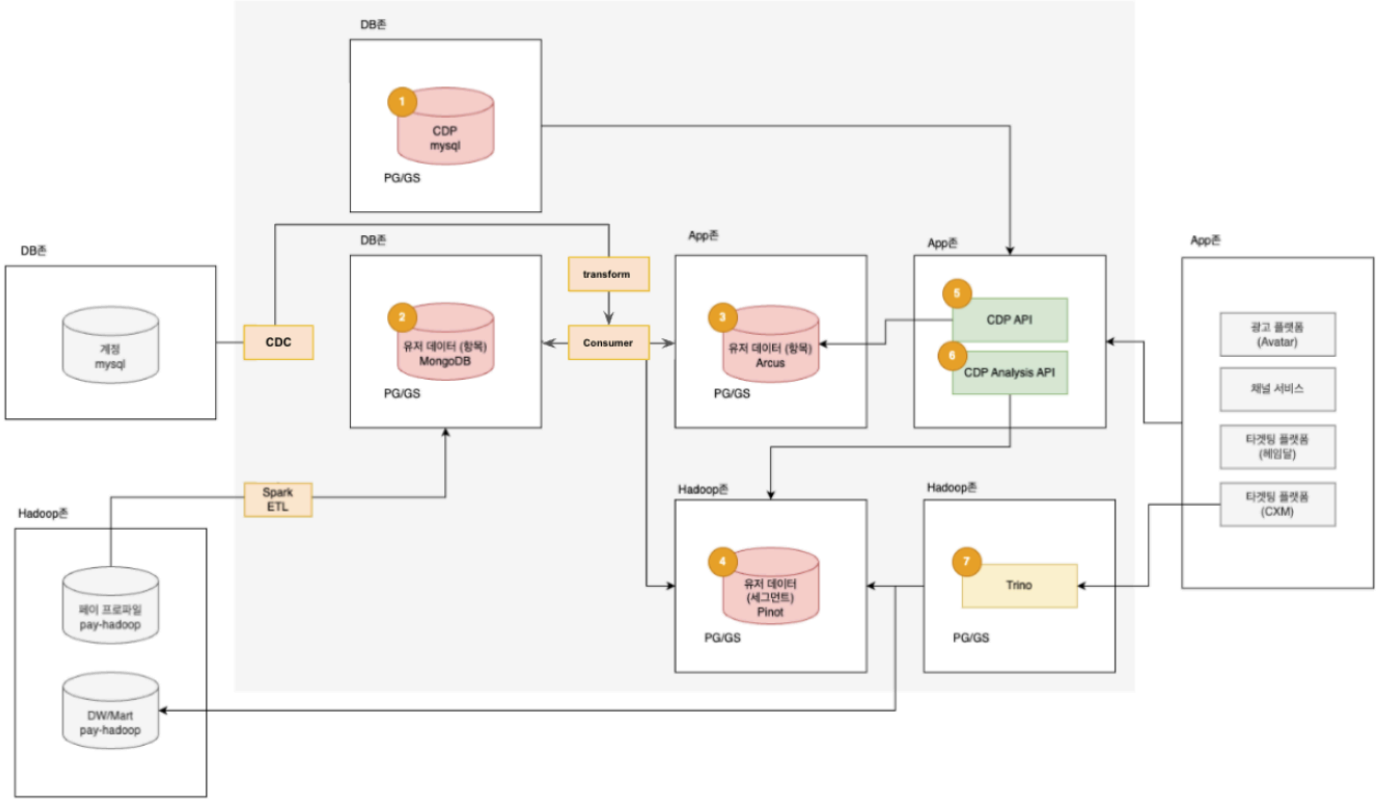
보안·컴플라이언스	Knox + Ranger 분리형	Unity Catalog 기반 중앙 거버넌스
확장성	Scale-up 중심, 용량 한계 도래	Scale-out / Cloud-native 구조
비용 효율성	HDFS·노드 유지비 과다	Object Storage + Auto-scaling으로 40~60% 절감
AI 연계성	Spark 중심 ML 한정	MLflow + Agent Bricks + Genie 기반 AI 확장

단계별 전환 전략

단계	주요 목표	실행 방안
Phase 1 – 준비 단계 (3~6개월)	아키텍처 진단 및 호환성 검증	HDFS/Kudu 데이터 분석 Delta/Iceberg 포맷 호환성 테스트 Airflow → LakeFlow PoC
Phase 2 – 마이그레이션 (6~12개월)	핵심 워크로드 이전 및 검증	Impala/Spark Job → Databricks SQL 변환 Lakebridge 활용해 SQL 코드 자동 변환 Ranger 정책 → Unity Catalog 전환
Phase 3 – 최적화/확장 (12개월 이후)	운영 효율화 및 AI 통합	Serverless SQL 활성화, AI/BI 연동 Agent Bricks 통한 AI 서비스 구축 Unity Catalog 기반 데이터 품질/보안 관리 강화

CDP

데이터플로우



플랫폼 설명

구축 배경

- 광고, 마케팅 및 채널에서 데이터를 사용할 수 있는 데이터 플랫폼 필요
  - 페이 프로파일 데이터를 즉시 연동하고 사용할 수 있는 체계
  - 하나의 API에서 다양한 조건으로 데이터를 가져갈 수 있도록 구현
  - 약관 등 실시간 데이터를 연동할 수 있도록 환경 구현
  - 유저 항목, 세그먼트 데이터 서비스 제공
  - 데이터를 파악할 수 있는 데이터카탈로그 제공
  - https://wiki.kakaopaycorp.com/pages/viewpage.action?pageId=253329699

목적

- CDP(고객 데이터 플랫폼)를 개발하고, CDP 기반 데이터 활용 프로세스 구축
- 정교화된 분석 데이터인 페이 프로파일 데이터와 실시간 고객 데이터를 제공하는 CDP 구축
- 광고, 유저/콘텐츠 타겟팅, 개인화 추천 플랫폼에 CDP를 연동하여, 광고, 마케팅 및 채널에서 즉시 사용 가능하게 제공
- 이를 통해 광고, 마케팅 및 채널, 비즈니스에 필요한 개인화 역량을 증대

플랫폼 목표 지표

데이터 통합	계정 실시간 데이터 수집 → 가공 → 서빙 10s - lag 관점으로 표현 (장애)	준실시간(near-realtime) 기준 챌린지 - 일반적으로 수 초, 수 분으로 표현.
데이터 서빙	서빙 API tps 28,214 서빙 API 응답 시간 p90 <10ms	max tps 14,107 * 2배 고려 - heimdal max tps 6,000 - cxm max tps 2,000 - avatar max tps 4,500 - home max tps 155 - payment max tps 192 - benefit max tps 1,260 latency 10ms - heimdal 평균 10ms - avatar 평균 56ms

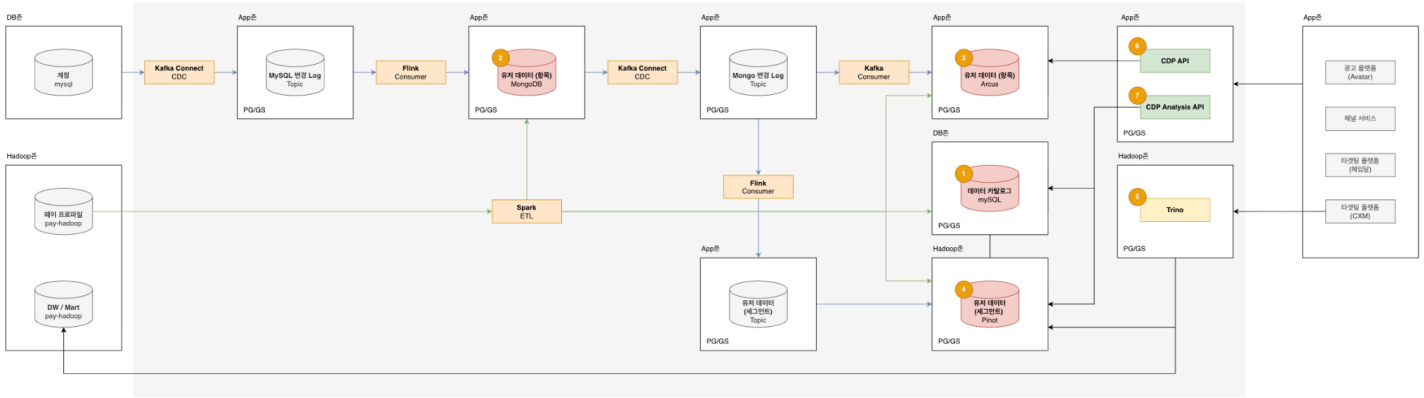
아키텍처

기본 방향

- DR 구성, Scale-out, 유지보수
- tps 28,214 / latency p90 <10ms

전체 아키텍처

Mongo CDC 활용



1	CDP mysql	데이터 카탈로그 관리 DB	mysql • hubble mysql 활용	
2	CDP MongoDB	데이터 스토리지 • 계정, 페이프로파일데	MongoDB • config/mongos ◦ A1 타입 서버 x 3 대	• MongoDB 서버 표준 ◦ https://wiki.kakaopaycorp.com/pages/viewpage.action?pageId=165679405 • 성능 테스트 ◦ 샌박 기준 pay_account_id 기준 50ms 측정 → 헤임달 기준의 평균 10ms 빠른 응답을 하기 위해서는 대용량 캐시 필요 ◦ https://docs.google.com/spreadsheets/d/1z804-rKJcjaciLsp7pMESyhiuOGKH9hsis6B5HpR8AY/edit?gid=0#gid=0

		이터1, 451 여개 항목 적재	• mongod (2 샤드) ◦ D2 타입 서버 x 3 대	

3	CDP Arcus	데이터 캐시
		• 계정, 페이프로파일 데이터 1, 451 여개 항목 적재
		• 특정 노드가 죽으면 m
			Arcus
			• zookeeper
			  ◦ A1 타입 5 대 (pg2, gs3)
			• memcached
			  ◦ B2 타입 8 대 (pg4, gs4
				https://kakaopay.agit.in/g/300042480/wall/413867130#comment_panel_414250984
				1. 특정 노드 불능상태일 경우
				노드가 빠질 경우엔 아커스 서비스 상으로는 특이사항 없으나, 데이터가 모두 사라지기 때문에 다시 적재해야 함
				• 위와 같은 사유로 서비스 영향도에 따른 노드 당 적정 데이터 용량 정책 필요
				• 노드 재투입할 경우 리밸런싱이 발생하는데, 전체 데이터 양에 따라 1분이상(500GB 기준)의 시간이 소요될 수 있음
				리밸런싱 시간 동안 Arcus 서비스 불능
				• 위와 같은 사유로 클러스터 2개를 Active-Standby 형태로 운영하는 방법이 필요함
				2. Arcus 서비스 제공의 기본 Rule
				현재 페이에서 사용하는 Arcus 의 경우 단순 Cache 용도로만 사용하기 때문에, 데이터 복구가 불가능
				3. 대규모 트래픽에 대한 Arcus Cluster 운영
				현재 페이에서 사용되고 있는 Arcus 서비스는 전체 데이터 기준으로 500GB 가 가장 큰 클러스터
				4. Zookeeper DR
				가산 3대, 판교 2대로 나눴을 경우 가산이 무너지면 쿼럼유지가 안되서 서비스가 내려갑니다. 아커스에서 주키퍼는 client 입장에서 멤캐시드와의 연결을 위해 서비스됩니다.
				최초 주키퍼에게 멤캐시드 서버에 대한 정보를 받고 한 번 연결하면 해당 맴캐시드와만 통신을 시작하게 되는데요.
				위와 같은 사유로 주키퍼 서비스가 내려간다고 당장의 문제는 발생하지 않습니다만, 주키퍼가 내려간 상태에서 새로 아커스에 접근 시도할 경우엔 맴캐시드 정보를 확인할 수 없어 아커스 접근에 문제가 발생합니다.

		ongoDB로 fall-back • arcuscluster를 이중화하여 구성	) → 12대 (pg6 , gs6) 으로 증설	위와 같은 장애도 없애기 위해서는 AWS 에 AZ 별로 1대씩 더 추가하는 방식으로 DR 구성을 진행할 수 있으며, 해당 구성을 진행할 경우엔 아래 스펙의 인스턴스가 추가로 필요하오니 참고부탁드립니다. #참고 AWS 스펙 c5.xlarge / vcpu : 4, memory :8 / 시간 당 : $0.192 / 월 간 약 $138.24 * 2대

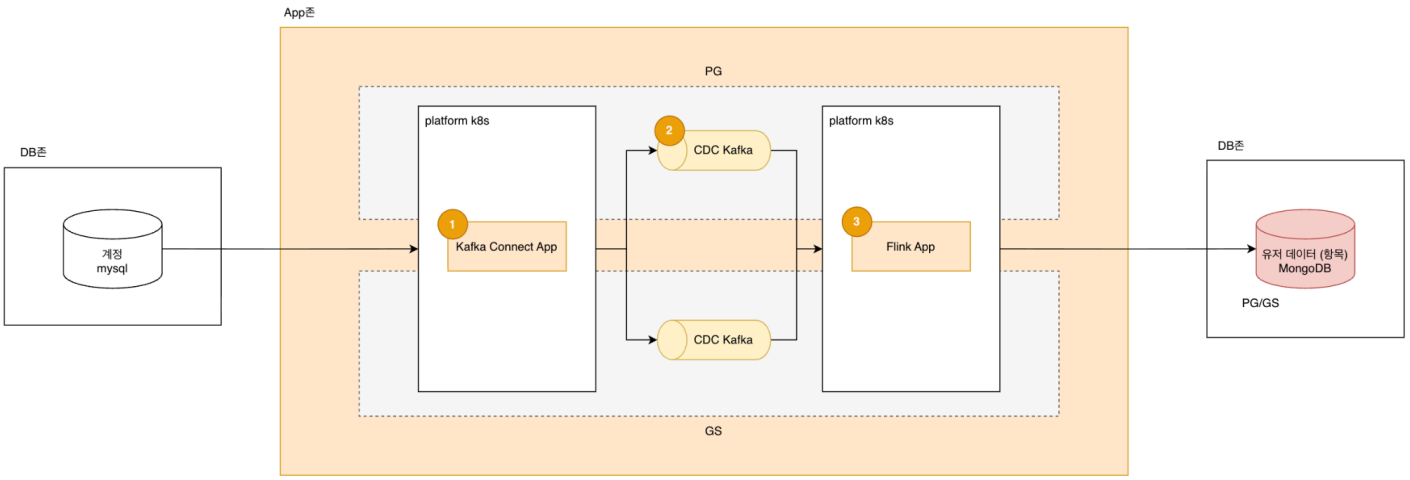
		할 필요 존재		
4	CDP Pinot	세그먼트 데이터 처리 (데이터 카탈로그 기반) • 계정, 페이프로파일데이터 1, 451 여개 항목	Pinot • zookeeper/controller ○ B2 타입 6 대 (pg3, gs3) • broker ○ B2 타입 6 대 (pg	row 단위 90억건, 세그먼트 및 수집 데이터 지속 증가, scale-out, 유지보수 고려하여 pinot 선정 ■ aurora (과도한 비용) vs. mysql shard (신규 스택) vs. elasticsearch (높은 라이선스 비용) vs. opensearch (신규 스택) vs. pinot (엔지니어링 지원)

		적재 • 저장소 + 실시간 OLAP 분석 기능	3, gs3) • worker ◦ D2 타입 (+128GB) 12대 (pg6, gs6)	

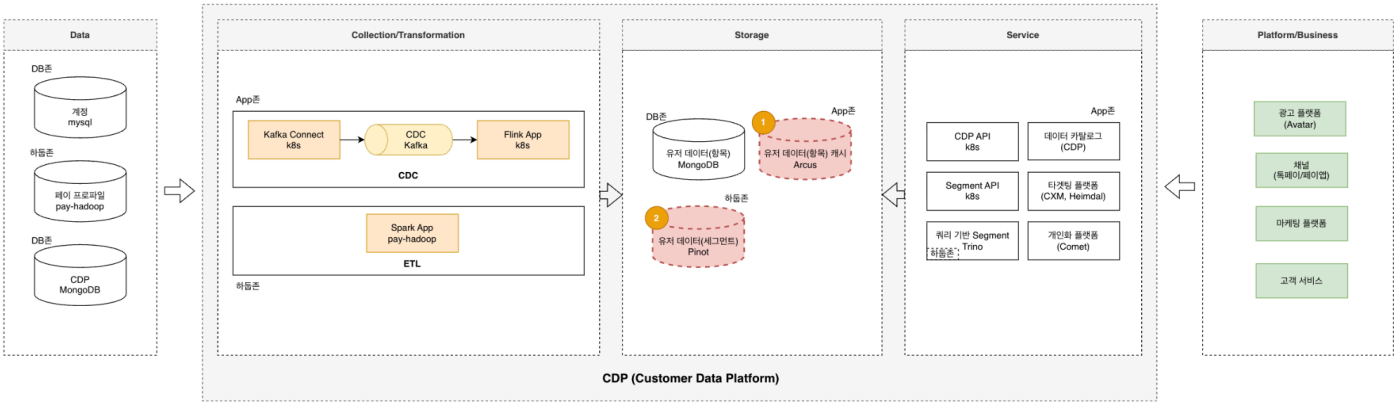
5	CDP Trino	세그먼트 데이터 처리 (Query 기반) • 하둡 + CDP 데이터 결합 추출 기능	trino • D2타입(+ 32db*4) 서버 x 8대 (pg 4대, gs 4대) → pg 12대, gs 4대로 증설	• Trino MongoDB 연동 ◦ https://jira.kakaopaycorp.com/browse/PAYDATA-14154 ◦ https://wiki.kakaopaycorp.com/pages/viewpage.action?pageId=274514985 • 공용 팔소닉(trino) 부하 포인트 ◦ https://wiki.kakaopaycorp.com/pages/viewpage.action?pageId=104050353
6	cdpick-api	유저 단위 데이터 서빙 api	k8s	
7	cdpick-analysis-api	데이터 카탈로그 서빙, 모수 추출 api	k8s	

CDC 아키텍처

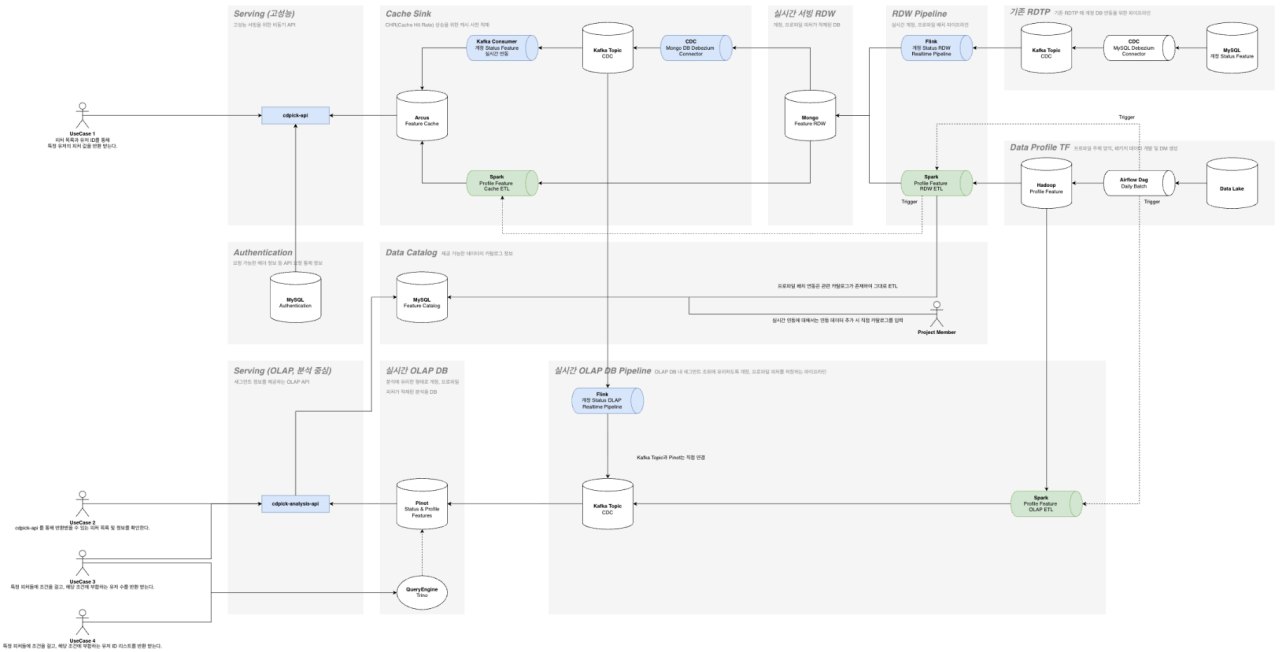
실시간 데이터의 일 평균 반영 시간 10s



데이터 플랫폼 구성



데이터 서빙 흐름



Kafka	분산형 메시징 시스템	• 높은 처리량과 낮은 지연 시간 • 안정적인 메시지 전달 보장 • 확장성 및 내 fault tolerance	• 메시지 broker에 대한 의존성 • 복잡한 설정 및 관리	실시간 스트리밍 데이터 파이프라인, 로그 집계, 메시지 기반 아키텍처
Flink	스트림 데이터 처리 프레임워크	• 실시간 데이터 처리 및 분석 • 다양한 데이터 소스 및 싱크 지원 • 강력한 fault tolerance	• 상대적으로 높은 학습 곡선 • 배치 처리 성능은 Spark에 비해 떨어짐	실시간 스트리밍 분석, 이벤트 기반 애플리케이션, 데이터 파이프라인

Spark	분산형 데이터 처리 엔진	• 대규모 데이터 처리 및 분석 • 다양한 프로그래밍 언어 지원 • 배치 처리 및 스트림 처리 가능	• 메모리 기반 처리로 인해 메모리 부족 문제 발생 가능 • 실시간 처리 성능은 Flink에 비해 떨어짐	배치 데이터 처리, ETL, 머신러닝, SQL 분석
Pinot	실시간 분석 데이터베이스	• 낮은 지연 시간으로 실시간 분석 가능 • 대규모 데이터 처리 및 분석 • SQL 기반 쿼리 지원	• 데이터 모델링 및 설계 필요 • 상대적으로 높은 비용	실시간 대시보드, 사용자 행동 분석, 광고 분석
Arcus	분산형 캐시 시스템	• 빠른 데이터 접근 속도 • 높은 확장성 • Memcached 프로토콜 지원	• 데이터 영속성 보장 X • 캐시 miss 발생 시 성능 저하	웹 서비스 성능 향상, 데이터 캐싱
MongoDB	NoSQL 데이터베이스	• 유연한 스키마 • 대규모 데이터 처리	• ACID 속성 미지원 • 관계형 데이터베	웹 애플리케이션, 모바일 애플리케이션,

		• 다양한 데이터 타입 지원	이스에 비해 성능 이슈 발생 가능성	IoT 데이터 저장