PlantPulse® **Industrial DataLake**

TECHNICAL WHITEPAPER

기술 백서 · REV. 2026.09

산업 데이터를 위해 조립된 데이터 인프라.

Aster 엔진 셋 **TSDB · CEP · MQTT** · 3계층 데이터 수명주기 · Iceberg 레이크하우스 · 시계열 엔진 · 단일 이미지 16 컴포넌트 · `pd` 한 줄 기동. 구조 · 수치 · 설정 · 실측을 저장소와 가동 중인 인스턴스에서 그대로 가져왔습니다.



16
컴포넌트 · 한 이미지

7.1×
시계열 저장 축소(실측)

Aster
엔진 셋 — TSDB · CEP · MQTT

4
질의 언어 · 1 게이트웨이

2026년 9월

PlantPulse DataLake · 온프레미스 / 폐쇄망 · 기술 검토자용

kopens®

00 · 요약

요약 — 산업 데이터레이크는 범용 데이터레이크와 무엇이 다른가

공장 데이터는 **초당 수만 건의 시계열**, 몇 년의 보존, 설비 계층이라는 맥락, 폐쇄망이라는 조건을 동시에 갖습니다. 범용 레이크하우스는 이 중 «보존»만 잘합니다. PlantPulse Industrial DataLake 는 **수집 → 핫/웜/콜드 저장 → 시계열 계산 → 표준 질의 → 운영**을 산업 데이터 기준으로 하나의 이미지에 조립한 인프라입니다. 심장부의 **Aster 엔진 셋** — 시계열 저장소 **TSDB** · 실시간 판정 **CEP** · 설비가 들어오는 문 **MQTT** — 은 KOPENS 가 소스째 개발 · 유지합니다. 시계열 엔진(TSE)과 데이터 게이트웨이도 자체 개발이고, 성격이 달라 따로 다룹니다(10 · 15–16쪽).

16
컴포넌트, 이미지 하나, `pd start` 한 번

7.1×
시계열 저장 축소 — 11.9 → 1.7 B/row (20M 행 실측)

233k
rows/s 인제스트 (Aster TSDB 단일 노드 벤치)

SQL:2016
Aster CEP — `MATCH_RECOGNIZE` 를 엔진에 구현, Flink 와 대조

3
계층 — 인메모리 · Aster TSDB 링 · Iceberg

RF 3
토큰 링 복제, `pd node add` 가 자동 조정

4
질의 언어 PQL · CQL · SQL · TQL, 게이트웨이 하나

12
콘솔 화면 · `--json` 계약 · 종료 코드 9종

목차

01 문제 — 세 벽 3
02 설계 원칙 다섯 4
03 아키텍처 — 배치 · 포트 · 기동 순서 5
04 이미지 — 출처 · 검증 · 설정 렌더 6
05 데이터 수명주기 — 핫 · 웜 · 콜드 7
06 Aster TSDB — 시계열 데이터베이스 8
07 스키마 — 키스페이스 · 파티션 · 압축 9
08 시계열 엔진 TSE — 집계 · TQL · Prometheus 10
09 시계열 DB 비교 — InfluxDB · TimescaleDB 11
10 레이크하우스 — Iceberg · Spark · 아카이브 12
11 메시징 — Kafka · MQTT · 두 버퍼 13
12 Aster CEP — 실시간 판정 · EQL 14
13 질의 계층 — Data Gateway 15
14 통제 · 감사 — 상한 · 오류 · 드라이버 16
15 확장 · 복제 — 토큰 링 · 워커 17
16 운영 — 콘솔 · pd · JSON 계약 18
17 보호 — 백업 · 복구 · 업그레이드 19
18 실측 — 가동 중인 인스턴스 20
19 범용 데이터레이크와의 비교 21
20 규모 산정 · 출처 · 문의 22

독자. 데이터 아키텍트 · 인프라/DB 운영자 · 보안 검토자. 영업 문서가 아닙니다 — 수치는 저장소의 설정 · 스키마 · 벤치마크 문서와 2026-09-08 가동 인스턴스(dev 41) 실측에서만 가져왔고, 출처는 22쪽에 있습니다.

01 · 문제

세 벽 — 히스토리안, 범용 레이크, 직접 조립

공장에서 데이터 인프라를 세우는 세 가지 길이 있고, 각각 다른 벽에 부딪힙니다.

벽 1
태그 히스토리안

저장과 조회는 훌륭하지만 **이벤트 · 명령 · BLOB · 관계**는 밖에 둡니다. 장기 보관은 «외부 레이크로 내보내기» — 데이터가 두 벌, 스키마가 두 벌. 태그 수가 과금 단위라 «덜 모으는» 유인이 생깁니다.

벽 2
범용 클라우드 레이크하우스

Parquet 위의 SQL 은 잘하지만 **초당 수만 건 시계열 인제스트 · 갭필 · 시간가중 평균 · 설비 계층**은 앱 몫입니다. 사용량 과금이라 «분석을 더 할수록» 비용이 늘고, 폐쇄망에는 들어오지 못합니다.

벽 3
오픈소스 직접 조립

Kafka + TSDB + Spark + 브로커를 잇는 PoC 는 빠릅니다. 3년 뒤 남는 것은 **버전 조합 · 설정 드리프트 · 사람에 묶인 운영 지식**. 표준 모델이 없어 사이트마다 스키마가 갈립니다.

산업 데이터가 요구하는 것 — 여섯 가지

요구	왜 범용으로 안 되는가	PlantPulse DataLake 의 답 (해당 장)
초당 수만 건, 수년 보존	행 단위 저장은 디스크가 선형으로 늘고, 삭제·TTL 이 컴팩션 부하가 된다	시간 윈도우 컴팩션 + 컬럼형 청크 7.1× 축소, 윈도우 통째 만료 (06)
시계열 계산은 저장소 안에서	앱이 원시 행을 끌어와 계산하면 네트워크와 메모리가 병목	21 개 CQL 함수 · 갭필 · TSE 30여 집계기 (06 · 08)
설비 맥락(사이트→라인→설비→태그)	범용 스키마에는 계층이 없고 매핑이 앱마다 다르다	static 컬럼 계층 · site_id 파티션 · 자산 모델 공유 (07)
이벤트 · 명령 · 이미지도 같은 곳에	TSDB 는 숫자만, 오브젝트 스토리지는 따로	Aster TSDB + RustFS 한 클러스터, BLOB 아카이버 (10 · 11)
폐쇄망 · 온프레미스 완결	SaaS 는 «Private» 옵션이 있어도 앱 · 관리면이 클라우드에 남는다	단일 이미지 · 서명 검증 · 오프라인 번들 (04)
전담 조직 없이 운영	분산 DB 는 보통 DBA 팀을 요구한다	콘솔 12 화면 · `pd` · `--json` · 미측정≠정상 (14)

이 백서가 «자랑»하는 방식. 형용사 대신 설정 키 · 스키마 · 명령 출력 · 벤치마크 표를 싣습니다. 검토자가 같은 저장소와 같은 인스턴스에서 **재현할 수 있는 것**만 적었습니다.

02 · 설계 원칙

다섯 가지 설계 원칙

아래 원칙은 구호가 아니라 코드와 테스트로 강제됩니다. 각 원칙 옆에 «어디서 강제되는가»를 적었습니다.

원칙 1

하나의 이미지, 하나의 기동 순서

16 컴포넌트가 `pd/plantpulse-datalake` 이미지 하나로 나오고, 기동 순서의 정본은 `services/order.txt` 파일 하나입니다. «다른 곳에는 순서를 정의하지 않는다»가 파일 첫 줄의 주석입니다. 저장 · 분석이 문지기 서비스로, 이들이 안 뜨면 exit 5 로 멈춥니다.

원칙 2

설정은 템플릿에서, 매 기동 때 렌더

27 개 설정 템플릿(`config/templates/*.template`)이 `render.map` 에 따라 런타임 경로로 렌더됩니다. 치환 안 된 `${PP_*}` 가 하나라도 남으면 exit 4 로 거부 — 반쯤 렌더된 설정으로 뜨는 일이 없습니다. 컨테이너 안에서 손으로 고친 것은 다음 기동에 사라집니다(드리프트 0).

원칙 3

오픈 포맷 · 표준 인터페이스

웜 계층은 upstream 호환 Cassandra 6.0 온디스크 포맷(Aster TSDB 는 «opt-in»만 추가), 콜드 계층은 Iceberg v2 + Parquet(zstd). 접근은 JDBC(Aster TSDB · Spark SQL · TQL 드라이버) · REST · MQTT · Kafka · Prometheus remote_write. 어느 계층도 전용 클라이언트를 요구하지 않습니다.

원칙 4

노드가 곧 확장 단위

워커 컨테이너를 띄우고 `pd node add` 를 치면 살아 있는 노드 수에 맞춰 복제 계수를 올리고 리페어합니다. 데이터센터 이름 · 노드 수를 읽지 못하면 «추측하지 않고 exit 1» — 잘못된 ALTER 가 복제를 조용히 없애기 때문입니다.

원칙 5

미측정은 정상이 아니다

`pd --json` 문서의 모든 검사 항목은 `pass` / `fail` / `na` 세 칸을 갖고, `na` 는 반드시 `reason` 을 동반합니다. 콘솔은 못 잰 값을 회색으로 남기고 0 이나 초록으로 채우지 않습니다. «우아하게 실패하는 파서는 못 잰 값을 정상으로 그리는 바로 그 결함»(json-contract.md) — 이 문장이 콘솔 · CLI · 테스트(`test-json-output.sh`)를 관통합니다.

원칙이 만드는 운영상의 결과

상황	일반적인 조립 스택	PlantPulse DataLake
업그레이드	구성요소별 버전 조합 검증 · 설정 병합	이미지 태그 교체 → `pd start` . 설정은 템플릿에서 다시 렌더
노드 추가	DB 별 절차 · 수동 RF 변경 · repair 일정	`pd node add` 1회 (RF 자동 · repair 포함)
장애 판정	대시보드마다 다른 임계 · 빈 그래프가 정상처럼 보임	16 컴포넌트 × 임계 · 24h 타임라인 · `na` 는 회색
감사	어느 커밋이 어느 사이트에 갔는지 추적 어려움	이미지 라벨 `io.kopens.dist-inputs` · 빌드 커밋 · 콘솔 «업그레이드» 화면

03 · 아키텍처

배치 · 포트 · 기동 순서

한 이미지가 **MASTER** 또는 **WORKER** 모드로 뜹니다. 마스터는 전 서비스를, 워커는 저장 · 분석 · 관리 API 만 올립니다. 기동 순서는 `services/order.txt` 그대로입니다.

서비스	MASTER	WORKER	구성요소 (기동 순)	포트
storage	●	●	valkey → postgres → cassandra → minio(MASTER 만)	REDIS 6379 · POSTGRESQL 5432 · CASSANDRA 9042 · MINIO 9000
analytics	●	●	spark-master → hive → gravitino → kyuubi	SPARK-MASTER 7077 · SPARK-UI 4440 · HIVE 9083 · KYUUBI 10000 · GRAVITINO 19001
messaging	●	–	kafka(KRaft), mqtt(Aster MQTT)	KAFKA 9092 · MQTT 1883
timeseries	●	–	engine(TSE), dashboard(Grafana)	ENGINE 7800/7801 · UI 3000
cep	●	–	Tomcat (CEP 엔진 13 jar + war)	CEP 7400
workflow	●	–	temporal → kestra → warehouse 워커	ENGINE 7233 · ENGINE-UI 8233 · BATCH 8380
data-gateway	●	–	Tomcat	DATA_GATEWAY 5500
sql	●	–	Tomcat (SQL 콘솔)	SQL 4000
admin-api	●	●	운영 콘솔 API + 콘솔 정적 번들	ADMIN_API 4949 · 콘솔 4950

문지기: `storage` 와 `analytics` 의 `health.sh` 가 통과해야 다음 서비스로 넘어갑니다(옛 Java 기동기의 «datalake ready» 검사를 셸로 옮긴 것). 시계열 엔진이 MASTER 전용인 이유는 문서에 적혀 있습니다 — 질의 연합이 없는 엔진을 워커에 띄우면 «마스터가 이미 쓴 행을 같은 링에 다시 복사하는 4.4 GB 서비스»가 됐기 때문입니다(2026-09-05 실측 후 결정).

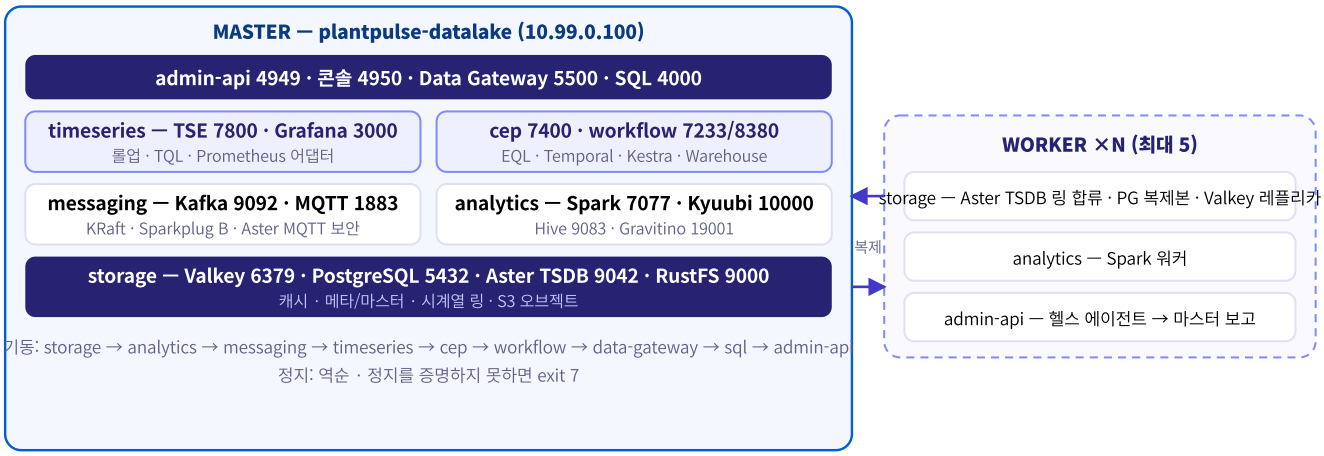


그림 1. 마스터 · 워커 배치와 포트. 워커는 storage · analytics · admin-api 만 올리고, 마스터의 Aster TSDB 링 · PostgreSQL 스트리밍 복제 · Valkey 레플리카에 합류합니다(13장).

04 · 이미지

이미지 — 출처 넷, 검증 셋, 렌더 하나

데이터레이크 이미지는 `plantpulse-datalake-docker` 가 조립합니다. 무엇이 어디서 오는지가 파일로 선언돼 있고, 구운 이미지 안에서 다시 검증합니다.

출처 넷 — 각각 정본이 하나

출처	나르는 것	선언 파일
tree/ (git)	업스트림 배포판 — Kafka · Aster TSDB · Tomcat · Spark · Temporal … 2026-09-01 스냅샷	`modules.txt` (10 모듈)
Nexus	PlantPulse 산출물 — war 3종 · `-lib.zip` · 고정 jar · 콘솔 zip. 못 받으면 **빌드가 죽는다**(건너뛰기 플래그 없음)	`nexus-overlay.txt`
src/ (모듈 git)	스크립트 · 템플릿 · systemd 유닛 — `pd` 전체, 스토리지 기동기, 시계열 런처	`src-overlay.txt`
dist (모듈 CI 아티팩트)	소스뿐인 모듈의 배포판 — `plantpulse-backup` . 아티팩트 커밋 ≠ clone 한 main HEAD 면 **거부**	`dist-inputs.txt`

검증 셋

- **이미지 안** `sha256sum -c` — src 오버레이 173 파일과 dist 매니페스트가 소스와 같은지 구운 뒤 확인. 다르면 CI 가 태그를 밀지 않습니다.
- **서명 · 체크섬** — 이미지에 cosign 서명, 폐쇄망 번들은 반입 전 체크섬 + 서명 검증. 실패하면 설치가 멈춥니다.
- **출처 라벨** — `io.kopens.dist-inputs` (예: `plantpulse-backup=<sha>`), 빌드 커밋 · 굽기 시각이 콘솔 «업그레이드» 화면에 표시됩니다.

설정 렌더 — 27 템플릿, render.map 한 장

```
# config/render.map — <template>|<target relative to PP_HOME>|<modes>|<cluster>
cassandra.yaml|plantpulse-storage/db/cassandra/conf/cassandra.yaml|MASTER,WORKER|cluster
jvm-server.options|plantpulse-storage/db/cassandra/conf/jvm-server.options|MASTER,WORKER|cluster
postgresql.conf|plantpulse-storage/db/postgres/conf/postgresql.conf|MASTER,WORKER|cluster
spark-defaults.conf|plantpulse-analytics/spark/conf/spark-defaults.conf|MASTER,WORKER|cluster
kafka.properties|plantpulse-messaging/kafka/config/kafka.properties|MASTER|-
gravitino-iceberg-rest-server.conf|plantpulse-analytics/gravitino/conf/...|MASTER|-
```

WORKER 는 `config/templates/cluster/` 의 같은 이름 템플릿을 렌더합니다(`cluster` 열). 렌더는 `pd start` 마다 일어나며, 광고 주소가 루프백(`127.0.0.1`)이면 exit 8 로 거부합니다 — 브로커가 자기 주소를 127.0.0.1 로 광고해 밖에서 못 붙는 사고를 막기 위한 것입니다.

운영자가 손대는 곳은 하나

호스트 `/etc/kopens/conf` 가 `$PP_HOME/plantpulse-datalake-cli/config/templates` 위에 마운트됩니다. 여기서 고친 템플릿이 다음 렌더에 반영되고, `pd config diff` 가 «지금 도는 설정»과 «렌더될 설정»의 차이를 보여줍니다(차이가 있으면 exit 1).

05 · 데이터 수명주기

핫 · 웜 · 콜드 — 데이터가 나이를 먹는 길

같은 태그 ID · 같은 자산 모델로 세 층을 관통합니다. 층 사이의 이동은 스케줄러(Temporal)와 저장 엔진(Aster TSDB)의 티어링이 맡고, 사람은 보존 기간만 정합니다.



웜 계층 보존 기간 — 출하 기본값 (`plantpulse-storage.properties`)

테이블	TTL	용도
tm_asset_data (실시간 설비 스냅샷)	10일	화면 · 규칙이 읽는 현재값 계열
tm_tag_point (원시 태그 포인트)	**3650일**	10년치 원본 즉시 조회 — 저장량이 가장 큰 축이므로 현장에서는 대개 줄여 씁니다
tm_tag_point_aggregation_* · sampling_* · snapshot	93일	1분 ~ 1시간 집계 · 10초 ~ 1시간 샘플링 · 스냅샷
tm_tag_point_archive	1년	콜드로 넘어간 뒤에도 링에서 바로 읽는 완충
tm_system_log · tm_monitor_* · tm_tag_point_validation	10일 · 1일 · 1일	운영 진단
tm_batch_job_run · tm_batch_audit · tm_warehouse_audit	31일	배치 실행 이력 · 감사

엔진 안의 티어링 — Aster TSDB 의 tiering 정책 (06장)

웜 계층 «안에서도» 나이가 듭니다. 테이블별 JSON 정책 `hot_window` (행으로 두는 기간, 예 7d) → 재인코더가 `interval` (예 1h)마다 지난 윈도우를 `<table>__chunks` 로 컬럼형 청크화 → `cold_window` 만료 시 청크를 통째로 드롭. **SELECT 는 바뀌지 않습니다** — 기본 테이블 조회가 핫 행과 청크를 자동으로 병합합니다. 청크화 때 셀 TTL 이 떨어지므로 `default_time_to_live` 를 쓰는 테이블은 `cold_window` 를 같은 값으로 두는 것이 규칙입니다.

06 · Aster TSDB

Aster TSDB — KOPENS 가 만든 산업 시계열 데이터베이스

웜 계층의 저장 엔진은 KOPENS 가 직접 개발 · 유지하는 **Aster TSDB**(Aster Timeseries Database, `github.com/kopens/aster-tsdb`, Apache-2.0)입니다. Apache Cassandra 6.0 위에 **압축 · 보존 · 시계열 계산을 데이터베이스 안으로** 넣었습니다. 온디스크 포맷과 CQL 문법은 업스트림 그대로라 기존 6.0 데이터를 그대로 읽고, 시계열 기능은 전부 **테이블별 opt-in** 입니다.

+5.4만	7.1×	21	93 + 49
줄 — 엔진 2.6만 · 테스트 2.8만 (업스트림 대비 169 파일)	저장 축소 — 11.9 → 1.7 B/row (20M 행 실측)	서버측 CQL 시계열 함수 · 갭필 · 회귀	릴리즈 게이트 통합 단언 · 3노드 클러스터 단언

컴팩션

TimeSeriesCompactionStrategy (TSCS)

윈도우 순서 정렬 · 현재 윈도우 안에서는 UCS 에 위임 · 지난 윈도우는 **SSTable 하나로 동결** · 보존 경계를 넘은 윈도우는 컴팩션 없이 **통째로 드롭**. 엣지가 밀어 넣는 늦은 데이터(백필)는 flush · 스트리밍이 윈도우 경계에서 갈라 자기 과거 윈도우에 격리합니다.

청크 포맷 V4

컬럼형 청크 코덱

윈도우 하나 = 공유 타임스탬프 축 + 컬럼별 독립 섹션. `double` 은 ALP/ALP-RD, 정수는 FOR/델타 비트패킹, boolean 1비트, 텍스트는 사전/원문. 변하지 않는 컬럼은 `CONSTANT`, 전부 null 은 `ALL_NULL` 로 O(1). 코덱은 Gorilla · Chimp128 과 **실측 대결**로 골랐습니다.

티어드 스토리지

투명 읽기 — SELECT 는 바뀌지 않는다

재인코더가 `hot_window` 지난 윈도우를 청크로 옮기고, 기본 테이블 `SELECT` 가 핫 행과 청크를 **자동 병합**합니다 — 범위 · 점 조회 · 집계 · 갭필 · `LIMIT` / `DESC` 모두. `nodetool retier` · 가상 테이블로 운영.

MEMTABLE · 검색

TimeSeriesMemtable · SAI 전문 검색

기본 자료구조 배열로 행당 힙 **5.5× 절감**(511 → 92 B), 쓰기 시점에 윈도우 배정. SAI `LIKE` + `index_analyzer` 로 `ALLOW FILTERING` 없이 **한국어 포함 부분 문자열 검색**.

21 개 서버측 CQL 시계열 함수 — 앱이 원시 행을 끌어와 계산하던 왕복이 사라진다

`time_bucket` · `first` / `last` · `delta` / `rate` / `derivative` · `counter_delta` / `counter_rate` (리셋 인지) · `percentile` · `time_weighted_average` · `integral` · `variance` / `stddev` · `histogram` · `approx_count_distinct` · 이변량 회귀(`regr_slope`, `regr_r2` …) · 갭필 `time_bucket_gapfill` + `locf()` / `interpolate()`

```
SELECT time_bucket_gapfill(1h, timestamp, '2026-07-01', '2026-07-02'), locf(avg(latency))
  FROM pp.tm_tag_point WHERE tag_id='TAG-001' AND timestamp >= '2026-07-01' AND timestamp < '2026-07-02'
 GROUP BY tag_id, time_bucket_gapfill(...) ORDER BY timestamp ASC;   -- 빈 버킷을 마지막 관측값으로 채운다
```

벤치마크 — 운영 스키마 20M 행, 티어링 끔 vs 켬 (host 234 · Xeon Silver 4114T · 2026-08-04)

지표	티어링 전	티어링 후 (v4)	효과
저장 용량	237.8 MB	33.3 MB	**7.1× 축소**
count(*) (4만 행 파티션)	303 ms	50 ms	**6.1×**
time_bucket 집계	150–270 ms	31–64 ms	**3–6×**
갭필 (locf / interpolate)	162–284 ms	30–53 ms	**5.4×**
90 태그 p95 · 시간 평균 (360만 행)	14.2 s · 14.9 s	2.6 s · 4.2 s	**5.4× · 3.5×**
처리량	인제스트 **233k rows/s**(단일 노드 · 12 로더 × 100행 배치) · 청크 재인코딩 **108k rows/s** · 청크 전체 스캔 740 µs / 3,600 행		

측정한 모든 질의가 티어링 뒤에 같거나 빨랐습니다. 드롭인 · 업스트림 추종. 빌드 산출물은 업스트림 파일명 그대로라 jar 교체가 설치의 전부이고, 정체는 매니페스트(`Implementation-Title: Aster TSDB` · `Aster-Upstream-SHA`)가 밝힙니다 — 빠지면 CI 가 실패합니다. **제약도 문서에 있습니다:** 파티션 키 없는 전체 스캔 집계는 티어링 테이블에서 쓰지 않음, 청크된 데이터에 `DELETE` / `SET null` 거부(`cold_window` 만료로만 제거), `counter` · 비동결 컬렉션 불가. 처리량은 RF 1 조건입니다(RF 3 에서는 낮아집니다).

07 · 스키마

키스페이스 · 파티션 · 압축 · 인덱스

스키마 정본은 `plantpulse-database-model` — Aster TSDB DDL(**104 테이블**) · PostgreSQL DDL(43 테이블, Flyway) · Iceberg DDL(Spark SQL). Polyglot persistence 로 데이터 성격마다 저장소가 다릅니다.

키스페이스와 노드 설정

```
CREATE KEYSPACE pp WITH replication = {
  'class': 'NetworkTopologyStrategy', 'datacenter1': 3 };

# cassandra.yaml.template (렌더 후)
cluster_name: 'PP_CLUSTER'
num_tokens: 16
allocate_tokens_for_local_replication_factor: 3
endpoint_snitch: GossipingPropertyFileSnitch
authenticator: PasswordAuthenticator
authorizer: CassandraAuthorizer
concurrent_reads: 256 / concurrent_writes: 256
memtable_heap_space: 2GiB / memtable_offheap_space: 2GiB
memtable_allocation_type: offheap_objects
memtable_flush_writers: 8
compaction_throughput: 320MiB/s
commitlog_sync: periodic (10000ms)
storage_compatibility_mode: NONE
# jvm-server.options.template
-Xms16G -Xmx16G -XX:+UseZGC
```

테이블 접두사

접두사	용도
tm_asset_*	자산 단위 시계열(초 · 분 · 시 · 일 롤업)
tm_tag_*	태그 포인트 · 집계 · 샘플링 · 카운트 · 알람
tm_opc_* · tm_site_*	연결 상태 · 사이트 요약
tm_system_* · tm_monitor_*	시스템 로그 · 클러스터 메트릭
ts_*	시계열 엔진(TSE) 저장
tw_*	Iceberg 아카이브 대상(콜드)

파티션 키 전략

패턴	파티션 키	용도
고빈도	(id, date, hour)	실시간 센서
중빈도	(id, date)	분 단위 집계
저빈도	(id)	시 · 일 집계
아카이브	(site_id, year, month, day, hour)	장기 보관
감사	(yyyymmdd) + timestamp + id	append-only

자주 바뀌지 않는 계층 메타(`site_id · area_id · line_id`)는 `static` 컬럼으로 파티션당 한 번만 저장합니다.

롤업 계열 — 원시 → 집계 → 샘플링

테이블 군	간격
tm_tag_point_aggregation_*	1 · 5 · 10 · 30분 · 1시간 — avg min max sum count stddev median first last
tm_tag_point_sampling_*	10 · 30초 · 1 · 5 · 10 · 30분 · 1시간
tm_asset_data_based_*	timestamp · second · minute · hour · day

컴팩션 · 압축 · 인덱스 (ddl.cql 실측)

- **UnifiedCompactionStrategy** 두 프로파일 — 고빈도: `T8 · base_shard 8 · target 512MiB · min 128MiB` , 일반: `T4 · base_shard 4 · target 1GiB` , 만료 SSTable 검사 300s
- **압축** — 핫 테이블 16개 `LZ4 (4 KB chunk)` , 콜드 성격 8개 `Zstd level 2-3 (16-64 KB chunk)`
- **SAI(Storage Attached Index)** 54개 — `ALLOW FILTERING` 없는 보조 조건 검색

08 · 시계열 엔진

TSE — 집계 · 롤업 · TQL · Prometheus 를 한 프로세스에

PlantPulse TimeSeries Engine 은 KairosDB 계열의 자체 시계열 엔진입니다. Aster TSDB 를 백엔드로 쓰고, **REST · TQL/JDBC · Grafana 데이터소스 · Prometheus 호환** 네 얼굴을 한 JVM 에서 제공합니다.

런타임

언어 · GC	Java 21 · 세대별 ZGC (`-XX:+UseZGC -XX:+ZGenerational`)
서버	Jetty 9.4 — HTTP 7800 / HTTPS 7801 동시
인제스트	파일 기반 **영속 큐(bigqueue)** → 배치 200 · 메모리 큐 100,000
인증	JAAS 파일/LDAP · Basic Auth · SSL/TLS · OAuth(옵션)
저장소	Aster TSDB (java-driver 4.19) · H2(개발) · 원격 데이터스토어 전송
UI	React 19 시계열 뷰어 — CDN 없음, 폰트 self-host (에어갭)
클라이언트	Java SDK — **store-and-forward** 버퍼링, 연결풀 1024

집계기 — 30여 종 (소스 실측)

Avg · Sum · Min · Max · Count · First · Last · Percentile · Std · Range · Rate · Diff · Divide · Scale · Sma · Ema · Integral · LeastSquares(회귀) · Sampler · Filter · Trim · Limit · Drop · **Lttb**(다운샘플링) · **Hampel**(이상치 필터) · **Anomaly** · Score · DataGapsMarking · SaveAs 등. 롤업 정의는 `/api/v1/rollups` 로 저장 · 자동 실행.

Prometheus 호환 어댑터

TSE 의 Jetty 안에서 in-process — 별도 Prometheus 서버 없이 `remote_write` (snappy+protobuf 1.0)를 받고, Grafana 가 «Prometheus 데이터소스»로 붙어 **PromQL** 을 그대로 씁니다. 라벨 ↔ 태그 무손실 매핑, `/prometheus/*` 경로에 같은 Basic Auth.

TQL — SQL 을 아는 사람이 바로 쓰는 시계열 질의

```
SELECT AVG('1 MINUTES') >> MAX('1 DAYS')     -- 집계 체이닝
  FROM CHANNEL1.PUMP1.TEMP_0001
 WHERE timestamp WITHIN '7 DAYS'
   AND tag['site_id'] IN ('SITE_A','SITE_B')
   AND value > 80
 GROUP BY _HOST, MM:SITE_ID
 ORDER BY timestamp DESC LIMIT 1000;

SELECT PERCENTILE('1 HOURS','START','0.95')
  FROM M WHERE timestamp BETWEEN ? AND ?;   -- PreparedStatement
```

시간 조건	**WITHIN '30 MINUTES' · SINCE '2026-01-01 00:00:00.000' · BETWEEN … AND …**
집계	AVG MIN MAX SUM COUNT FIRST LAST DEV PERCENTILE GAPS LEAST_SQUARES
변환	RATE DIFF SAMPLER DIV SCALE FILTER TRIM
기타	INSERT … TAGS · DELETE · DESCRIBE · METRICS · ROLLUPS · PING
JDBC	`jdbc:ts://host:7800` · `jdbc:tss://` (TLS) · SPI 자동 등록 · 결과는 JSON 한 컬럼

```
Connection c = DriverManager.getConnection(
    "jdbc:ts://192.168.0.41:7800", "tse", "…");
ResultSet rs = c.createStatement().executeQuery(
    "SELECT AVG('1 MINUTES') FROM CHANNEL1.PUMP1.TEMP_0001 "
  + "WHERE timestamp WITHIN '1 HOURS'");
while (rs.next()) json = rs.getString(1);
// {"metric":..,"timestamp":..,"value":..,"tags":{..}}
```

> **왜 두 시계열 계층인가.** 저장 엔진(Aster TSDB)은 **CQL 안에서** 버킷 · 갭필 · 회귀를, TSE 는 **메트릭 API 위에서** 롤업 · 이상치 · 다운샘플링 · Prometheus 호환을 맡습니다. 원시 데이터는 한 링에 한 번만 저장됩니다.

09 · 시계열 DB 비교

InfluxDB · TimescaleDB 와 같은 기준으로

둘 다 훌륭한 시계열 DB 입니다. 차이는 «산업 데이터 인프라의 원 계층»으로 놓았을 때 드러납니다. 경쟁 제품 항목은 2026-09-09 공개 문서 기준(출처 20쪽).

항목	PlantPulse (Aster TSDB + TSE)	InfluxDB 3 (Core / Enterprise)	TimescaleDB (PostgreSQL 확장)
저장 모델	Aster TSDB(Cassandra 6.0 기반 자체 엔진) — 행(핫) + 컬럼형 청크(웜) 투명 병합, 토큰 링 분산	Arrow · DataFusion · Parquet 컬럼형, 오브젝트 스토리지(S3) 기반 «디스크리스»	PostgreSQL 하이퍼테이블 · 청크, 컬럼형 압축
확장 · HA	● 토큰 링 · RF 3 · 노드 추가 = 선형, 워커 최대 5, `pd node add` 자동	◐ **Core 는 단일 노드**. 멀티노드(인제스트 · 질의 · 컴팩터 분리)는 **Enterprise**	◐ **멀티노드는 2.14 부터 지원 종료**(2.13 이 마지막). 수직 확장 + PG 스트리밍 복제본
압축	● 청크 코덱 v4(ALP · FOR/델타) **7.1×**(11.9 → 1.7 B/row, 20M 행 실측) + SSTable LZ4/Zstd	● Parquet 컬럼형(수치 미공개)	● 컬럼형 압축 «90–95% 절감»(공개 자료)
장기 질의 · 보존	● 테이블별 TTL, 윈도우 통째 만료, 콜드는 Iceberg 로 자동 아카이브 — 질의 범위 제한 없음	◐ **Core 는 질의 범위 72시간 제한**(432 Parquet 파일 = 10분 블록 × 72h). 컴팩터로 해제되는 것은 Enterprise	● 보존 정책 · 청크 드롭, 계층화(Tiered Storage)는 Timescale Cloud
시계열 함수	● CQL 21종(갭필 · 회귀 · 시간가중평균 · 카운터) + TSE 30여 집계기(LTTB · Hampel · 이상치)	● SQL · InfluxQL (Flux 는 3.x 에서 제외)	● `time_bucket` · 연속 집계(자동 갱신 MV) · 갭필 · 하이퍼펑션
설비 맥락 · 이벤트 · BLOB	● 자산 계층 static 컬럼 · 알람 · 이벤트 · 명령 · BLOB 이 같은 링/오브젝트 스토리지	○ 숫자 · 문자열 시계열, 오브젝트는 별도	◐ 관계형이라 이벤트는 가능, BLOB 은 PG 밖 권장
수집 표준	● Kafka · MQTT/Sparkplug B · REST · Prometheus remote_write · 산업 프로토콜은 Edge 42종	◐ Telegraf(플러그인) · 라인 프로토콜	◐ PG 클라이언트 · 외부 수집기
라이선스	Aster TSDB Apache-2.0(공개) + 제품 라이선스(구성 기준)	Core: MIT/Apache-2 · Enterprise: 상용(홈 사용 무료 예외)	Apache-2 코어 + **TSL**(압축 · 연속 집계 · 보존은 커뮤니티 라이선스, 관리형 서비스 제공 금지)
운영면	● 콘솔 12화면 · `pd` · 백업(Medusa) · 임계 기반 상태 · 24h 타임라인	◐ CLI · UI(Enterprise), 컴팩터 운영	● PostgreSQL 운영 도구 그대로(pgBackRest 등)

PlantPulse 를 고르는 경우

수년 보존과 수만 태그를 **한 링에서** 선형 확장해야 하고, 이벤트 · 명령 · 이미지가 시계열 옆에 있어야 하며, 폐쇄망에서 콘솔까지 완결돼야 할 때. 시계열 함수를 저장소 안에서 돌려 앱을 가볍게 하고 싶을 때.

Influx · Timescale 이 맞는 경우

단일 노드 규모의 메트릭 · 모니터링(Influx Core), 이미 PostgreSQL 중심이고 관계형 조인이 핵심인 팀(Timescale). 두 제품 모두 PlantPulse 와 **Kafka · Prometheus remote_write · JDBC** 로 연동 가능합니다.

10 · 레이크하우스

Iceberg · Spark · 아카이브 — 콜드 계층의 실제 모양

테이블 정의 (spark/ddl.sql 발췌)

```
CREATE TABLE spark.plantpulse.tm_tag_point_warehouse (
  tag_id STRING, tag_name STRING, value STRING, quality INT,
  site_id STRING, area_id STRING, line_id STRING, asset_id STRI
NG,
  opc_id STRING, timestamp TIMESTAMP, tuuid STRING, type STRIN
G,
  attribute MAP<STRING,STRING>, error_code INT, latency INT,
  year INT, month INT, day INT, hour INT, minute INT, second IN
T)
USING iceberg
PARTITIONED BY (site_id, year, month, day)
TBLPROPERTIES (
  'format-version'='2',                          -- row-level del
ete
  'write.target-file-size-bytes'='268435456',  -- 256 MB
  'write.data.compression-codec'='zstd',
  'write.metadata.previous-versions-max'='5',
  'write.metadata.delete-after-commit.enabled'='true',
  'read.split.target-size'='134217728');       -- 128 MB
```

왜 Iceberg 인가

- **ACID** — 아카이브 중 읽어도 반쯤 쓰인 파일이 보이지 않음
- **스키마 · 파티션 진화** — 컬럼 추가/변경에 기존 파일 재작성 없음
- **타임 트래블** — 스냅샷 시점의 테이블 재조회 · **히든 파티셔닝** — 질의에 파티션 컬럼 불필요
- **엔진 중립** — Spark · Kyuubi 외에 Trino · Flink 등 Iceberg 리더가 그대로 읽음

분석 계층 버전 (UPSTREAM-LOCK 실측)

Spark	4.1.3 (Scala 2.13) — 세대별 ZGC, `spark.sql.shuffle.partitions=128`, executor 8 cores
Kyuubi	1.12.0 — 다중 사용자 SQL 서빙 · JDBC 10000
Gravitino	1.3.0 — Iceberg REST 카탈로그 19001
Hive · Hadoop	2.3.9 메타스토어 9083 · 3.3.4
RustFS	1.0.0 — S3 API 9000 · 콘솔 9001 (Apache-2.0)

아카이브 잡 8종 — SPI 플러그인 (plantpulse-archive-job)

jobType	Aster TSDB → Iceberg
ARCHIVE_TAG_POINT	tm_tag_point_archive → tw_tag_point
ARCHIVE_ASSET_DATA	tm_asset_data → tw_asset_data
ARCHIVE_ASSET_AGGREGATION	tm_asset_aggregation → tw_asset_aggregation
ARCHIVE_ASSET_ALARM	tm_asset_alarm → tw_asset_alarm
ARCHIVE_ASSET_EVENT	tm_asset_event → tw_asset_event
ARCHIVE_ASSET_CONTEXT	tm_asset_context → tw_asset_context
ARCHIVE_ASSET_COMMAND	tm_asset_command → tw_asset_command
ARCHIVE_BLOB	tm_blob / tm_blob_object → tw_blob + S3 원본

```
// plantpulse.warehouse.s3.spi.ArchiveTableJob — ServiceLoader
로 발견
String jobType();  String sourceTable();  String targetTable();
long archive(SparkSession spark, ArchiveRequest request);
```

jar 는 이미지에 굽지 않고 `$PP_HOME/plantpulse-workflow/warehouse/jobs` 또는 `$PP_DATA_DIR/jobs` 에 놓입니다 — 사이트가 이미지 재빌드 없이 잡을 더할 수 있습니다. 제공자를 하나도 못 찾으면 «어디를 봤는지 말하며 실패»합니다(0행 성공이 아니라). 실행 로그: `[ARCHIVE-JOB] loaded ARCHIVE_ASSET_ALARM … tm_asset_alarm -> tw_asset_alarm from file:/…/jobs/plantpulse-archive-job.jar`

스케줄 — Temporal 워크플로

워크플로	스케줄 (Asia/Seoul)	수행
ArchiverWorkflow	매일 02:20	`archive.sh` → `spark-submit`, 8 잡
OptimizerWorkflow	일요일 03:00	`rewrite_data_files` (min 10 파일 · 128 MB) · `expire_snapshots` (3일, 최소 1) · `remove_orphan_files`

Task Queue `PP-S3-QUEUE`. 최적화 대상은 «발견된 아카이브 잡이 쓰는 테이블»을 그대로 씁니다 — 목록을 두 벌 두면 «아카이브는 되는데 최적화만 안 되는 테이블»이 조용히 생기기 때문입니다.

11 · 메시징

Kafka · MQTT — 들어오는 네 개의 문

Kafka 4.3.1 (KRaft — ZooKeeper 없음)

```
# kafka.properties.template (발췌)
num.partitions=4
log.retention.ms=3600000          # 1h — 버퍼이지 저장소가 아니다
log.retention.check.interval.ms=60000
log.segment.bytes=10485760         # 10 MB 세그먼트
message.max.bytes=157286400        # 150 MB — BLOB(이미지 · 도면) 통과
num.network.threads=3 / num.io.threads=8
default.replication.factor=1
```

Kafka 는 **수집 버퍼**입니다. 보존 1시간은 «저장은 Aster TSDB 가 한다»는 설계의 표현이고, 콘솔 «스토리지» 화면이 Kafka 디스크와 보존을 따로 보여 줍니다(실측 191 MB).

BLOB 전송 계약 (plantpulse-event-client-blob)

```
topic: blob   key = JSON, value = byte[]
{ "_event_type":"BLOB", "blob_id":"BLOB_ID_001",
  "timestamp":1772041960266,
  "attribute":"{\"site_id\":\"SITE_00001\",\"asset_id\":\"ASSET_DJ_M_01_1\"}",
  "_file_path":"/data/images/vision.jpg",
  "_file_size":102400, "_mime_type":"image/jpeg" }
```

비전 이미지 · 도면이 태그와 같은 자산 ID 로 들어와 `tm_blob` → RustFS → `tw_blob` 아카이브까지 한 경로를 탑니다.

MQTT — Aster MQTT

설비가 들어오는 문입니다. 표준 MQTT 위에서 자체 인증/ACL 로 플랫폼 계정에 접속을 묶고, Sparkplug B 로 자산 트리를 토픽에 미러링(UNS)합니다.

운영에서 고친 것 — 🔴 **인증자가 없으면 기동을 거부합니다**(보안 익스텐션이 안 뜨면 브로커가 **열린 채로 «정상»처럼** 보였습니다). **드롭 사유별 카운터 일곱**이 용량 · 백프레셔 · 정책을 가르고, 네트워크 라이브러리의 **공개 취약점 15건**을 닫았습니다.

버퍼가 둘인 이유

버퍼	맡는 것
Kafka	대량 수집 · BLOB · 외부 DB 동기화. 파티션 단위 병렬, 디스크 보존 1시간
Valkey 스트림	**실시간 판정용 이벤트**. 접두사 `PP:CEP-IN-STREAM`, **16 샤드**, 명시적 위치(스트림 ID)로 읽는다

둘은 경쟁하지 않습니다 — 저장 경로는 Kafka 또는 Redis 를 `batch.data.source` 로 고르고, 실시간 판정(12장)은 Valkey 스트림을 읽습니다.

수집 경로 요약

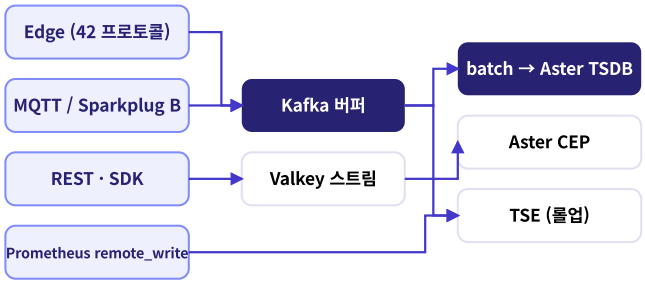


그림 2. 네 입구, 두 버퍼, 세 소비자. 저장(batch-web → Aster TSDB)은 Kafka 또는 Redis 를 `batch.data.source` 로 고를 수 있고, 외부 DB 동기화(TimescaleDB · InfluxDB · QuestDB · M3DB)도 같은 컨슈머가 맡습니다.

> **보존은 버퍼의 일이 아닙니다.** Kafka 1시간 · Valkey 스트림 최대 길이는 «따라잡지 못한 소비자» 를 위한 여유이지 아카이브가 아닙니다. 영구 보존은 Aster TSDB 링과 Iceberg(10장)가 맡습니다.

실시간 버퍼의 계약 — 잃지 않는 대신 두 번 올 수 있다

at-least-once

「위치를 옮기는 것」과 「체크포인트를 커밋하는 것」을 원자적으로 묶을 방법이 없습니다. 체크포인트 **뒤**에 옮기면 그 사이 크래시가 재전달하고, **앞**에 옮기면 유실합니다. **앞쪽을 골랐습니다** — 이벤트를 잃지 않는 대신, 마지막 체크포인트와 실패 사이의 이벤트가 두 번 옵니다. 트랜잭션 없는 Kafka 소비자가 갖는 천장과 같습니다.

위치는 클라이언트가 든다

소비자 그룹의 서버측 목록이 아니라 **명시적 스트림 ID** 로 읽습니다. 서버가 위치를 들면 체크포인트가 거기로 복원할 수 없기 때문입니다. 명시적 ID 는 클라이언트 측 상태라 **Kafka 오프셋과 똑같이 체크포인트**됩니다.

잘린 구간은 «센다»

스트림은 최대 길이로 잘리므로, 소비자가 오래 내려가 있었으면 **남아 있는 어떤 것보다 오래된 위치**로 돌아옵니다. 그 위치에서 읽으면 구멍 뒤의 항목이 **아무 오류 없이** 옵니다. 그래서 조용히 재개하지 않고 **구멍 건수를 세어 보고**합니다 — 0 이 아니면 소비자가 보기도 전에 사라진 것입니다.

결과 쪽도 계약이 있습니다. 판정 결과는 문장마다 하나인 토픽 `PP:CEP-OUT-TOPIC:<이름>` 으로 나가고, **빈 flush 는 아무것도 발행하지 않습니다** — 빈 배열은 「매치했고 행이 없다」는 주장이고, 아무것도 없을 때 정직한 메시지는 침묵입니다. 구독자가 0이어도 발행은 «성공» 하므로 거기가 결과가 조용히 사라지는 자리이고, 그래서 `published with no subscriber` 경고를 남깁니다.

12 · 실시간 판정

Aster CEP — 실시간 판정 엔진

저장한 뒤에 물어보는 것과, 흘러가는 동안 판정하는 것은 다른 문제입니다. **CEP(복합 이벤트 처리)** 는 이벤트가 지나가는 동안 시간 · 논리 조건을 맞춰 보고 알람 · 트리거를 냅니다. 규칙은 **EQL** 로 쓰고, 판정은 KOPENS 의 **Aster CEP** 엔진이 CEP 서버(`:7400`) 안에서 **in-process** 로 합니다 — 별도 스트리밍 클러스터가 필요 없습니다.

0
별도 스트리밍 클러스터 — CEP 서버 «안»에서 판정한다

2,127
엔진 테스트 + CEP 계층 2,501

152
골든 EQL 코퍼스 — 엔진별 능력 행렬 자동 생성

SQL:2016
`MATCH_RECOGNIZE` 연속 수량자 — 엔진에 직접 구현

구조 — 언어가 먼저, 엔진은 그것을 실행한다

언어	**EQL 1.0** — SQL-92 코어 + 스트리밍 확장(`WINDOW` · `OUTPUT … EVERY` · 패턴 · 부재 패턴 · `MATCH_RECOGNIZE`)
중간표현	엔진 독립 공통 IR · 판정 셋 `SUPPORTED` · `UNSUPPORTED` · `UNPARSEABLE`
실행 엔진	**Aster CEP** — CEP 서버 안에서 문장 하나 = 앱 하나
오라클	Apache Flink 2.3.0 — 같은 문장의 답을 **대조하는 데만** 씁니다. 운영에서는 돌지 않습니다
입력 · 출력	Valkey 스트림 16 샤드 → 문장마다 토픽 `PP:CEP-OUT-TOPIC:<이름>`

엔진이 우리 것이라는 말의 뜻

Aster CEP 는 KOPENS 가 **소스째 개발 · 유지**하는 엔진입니다. 모듈 넷(`aster-core` · `query-api` · `query-compiler` · `annotations`), 패키지 `plantpulse.cep.aster.*` , 플랫폼과 같은 캘린더 버전(`2026.09`)으로 함께 발행합니다. **현장에서 필요해진 구문과 결함 수정을 남의 로드맵을 기다리지 않고 넣습니다** — 아래 행 패턴 인식이 그렇게 들어갔습니다.

엔진을 고칠 때 지키는 것 — CI 게이트

- **층을 먼저 가른다.** 함수로 되는 것은 classpath 확장으로 얹고, **문법 · 상태기계를 바꿔야만 하는 것**만 엔진 코어에 넣습니다 — 엔진 변경이 적을수록 회귀가 적습니다
- **공개 API 게이트** — 클래스 집합과 공개 API 가 예고 없이 달라지면 빌드가 실패합니다. 엔진 위의 어댑터가 조용히 깨지지 않습니다
- **회귀 게이트** — 엔진 시험 **2,127** 건과 골든 EQL 코퍼스가 매 빌드에 함께 돕니다

못 돌리는 구성은 «이름을 대고 거부» 합니다. 건너뛰고 배포하면 문장이 **돌고 · 내보내고 · 틀린** 상태가 됩니다 — 아무것도 빨갛지 않은 채로. 엔진은 부팅당 하나이고 **폴백이 없습니다** — 둘이 함께 돌면 같은 입력이 두 번 발행됩니다.

행 패턴 인식 — SQL:2016 `MATCH_RECOGNIZE`

설비의 「시작–종료 구간」 판정은 **사이의 «모든» 행**이 조건을 만족할 것을 요구합니다. 즉 **연속** 수량자 `A{2,5}` 가 필요하고, 틈을 허용하는 반복은 다른 집합입니다. 이것은 함수나 플러그인으로 흉내 낼 수 없어 **문법과 상태기계를 엔진 안에서 직접** 구현했습니다 — 엔진이 우리 것이라 할 수 있는 일입니다.

```
-- 골든 코퍼스 230: 지연 100 초과가 «연속» 3건이면
SELECT tag, n FROM Point MATCH_RECOGNIZE (
  PARTITION BY tag_id
  MEASURES first(h.tag_id) AS tag, count(h.latency) AS n
  AFTER MATCH SKIP TO NEXT ROW
  PATTERN (h{3})
  DEFINE h AS h.latency > 100
)
```

한다	이름을 대고 거절한다
수량자 `* + ? {n} {n,m} {n,}` · greedy/reluctant · `DEFINE` · `MEASURES FIRST/LAST/COUNT` · `ONE ROW PER MATCH` · `SKIP TO NEXT ROW` · `PARTITION BY`	`ALL ROWS PER MATCH` · `SKIP PAST LAST ROW` · 교대 `|` · `PERMUTE` · MEASURES 안의 산술

증명 — 답을 «다른 구현» 과 대조했다

픽스처	Aster CEP	Flink 2.3.0
구간 `s m* e`	1매치 (1,5,2,4,3)	같다
greedy · reluctant `m*?`	mid 2 · mid 0	같다
겹치는 입력 · `{1,2}`	4매치 · 2매치	같다
greedy 가 백트랙해야 하는 입력	**1매치 (SQL:2016)**	0매치

마지막 줄은 오라클과의 **측정된 불일치**이고, 표준을 따른 쪽이 Aster CEP 입니다. 픽스처는 모드가 **실제로 갈라지는 것**을 오라클에서 먼저 확인해 골랐습니다.

능력은 선언이 아니라 «생성물» 이다

골든 EQL 152건을 엔진마다 실제 컴파일러에 넣어 능력 행렬을 **생성**하고, CI 가 커밋본과 대조합니다. IR 에 실리는 127건 중 **Aster CEP 106** · Esper 78 · Flink 53. 못 받는 21건은 전부 `pattern:until` 같은 **거절 토큰**으로 이름이 붙어 있습니다. 차등 하네스는 판정이 `AGREEMENT` · `DIVERGENCE` · `NOTHING_COMPARED` 셋이고, 비교할 것이 없으면 통과가 아니라 실패합니다.

13 · 질의 계층

Data Gateway — 네 언어를 한 REST 로

분석가 · 외부 시스템 · BI · AI 에이전트는 저장소에 직접 붙지 않습니다. **Data Gateway(:5500 · TLS :5501)** 가 네 언어를 한 엔드포인트로 받고, 스키마를 바꾸는 문장을 막고, 결과를 제한하고, **모든 질의를 기록**합니다.

네 언어, 네 저장소

언어	붙는 곳	무엇을 보나
PQL	PostgreSQL	메타데이터 — 자산 · 태그 · 사용자 · 설정
CQL	Aster TSDB	센서 원시값 · 집계 · BLOB 메타
SQL	Kyuubi (Spark)	Iceberg 콜드 계층 · 웨어하우스
TQL	시계열 엔진 (TSE)	메트릭 · 롤업

엔드포인트

POST /api/v1/query	전체 결과 + `truncated` 플래그(상한 100,000행)
POST /api/v1/query/stream	NDJSON — 행마다 한 줄, 마지막 줄이 `end`
POST /api/v1/update	갱신 (TQL 은 거부)
POST /api/v1/format	질의문 정렬 · 들여쓰기
POST /api/v1/cache	인메모리 현재값 조회
/api/v1/keys	API 키 발행 · 목록 · 폐기
GET /api/health	readiness (인증 불필요) — DB · 인메모리 · 질의로그
GET /admin/*	운영 지표 · 스레드 덤프 (인증 필요)

인증은 Spring Security 체인 하나가 정하고 **fail-closed** 입니다 — 목록에 없는 경로는 401 이지 익명이 아닙니다. 세션도 쿠키도 만들지 않습니다 (STATELESS).

왜 직접 붙지 못하게 하나

- **감사** — 「누가 무엇을 언제 조회했나」가 저장소마다 흩어지지 않고 한 표에 남습니다
- **상한** — 행 수 · 요청 크기 · 초당 요청을 **한 자리에서** 겁니다. 저장소마다 다른 방식으로 거는 것이 아닙니다
- **차단** — 스키마를 바꾸는 문장을 언어와 무관하게 같은 규칙으로 막습니다
- **교체** — 저장소의 주소 · 계정 · 드라이버가 바뀌어도 **소비자 코드는 그대로**입니다

그래서 BI 도구처럼 **JDBC 로 직접 붙어야만 하는** 경우에는 드라이버 셋(14장)을 따로 내주되, 그 경로는 저장소의 계정 체계로 통제합니다. **「열지 않는다」가 아니라 「어느 문으로 열었는지 안다」** 가 이 계층의 목적입니다.

스트리밍 질의

```
curl -X POST -H "X-API-Key: <key>" -H "X-Lang: PQL" \
  -H "Content-Type: application/json" \
  -d "SELECT tablename FROM pg_tables WHERE schemaname='pp'" \
  https://host:5501/api/v1/query/stream

{"row":{"tablename":"mm_asset"}}
{"row":{"tablename":"mm_tag"}}
{"end":{"count":43,"exec_ms":42,"truncated":false}}
```

서버가 전체를 메모리에 올리지 않고 JDBC 커서로 흘려보냅니다. 인증은 헤더 단계에서 끝나므로 **응답은 언제나 200 으로 시작합니다** — 그 뒤의 실패는 본문의 마지막 줄 `{"error":{…}}` 로만 보입니다.

파라미터 바인딩

```
X-Params: <URL-encoded JSON>
{ "site_id": "SITE_00001",
  "from:ts-long": 1772041960266,
  "limit:int": 500 }

SELECT * FROM pp.mm_tag
 WHERE site_id = :site_id AND created_at >= :from LIMIT :limit
```

키에 `:타입` 을 붙여 변환을 지정합니다 — `int long float double string epoch ts-long ts-string` . 모르는 타입 접미는 **400** 이지 무시가 아닙니다. 헤더가 깨져 있어도(`%zz` · JSON 아님) 400 입니다 — 클라이언트 오류를 500 으로 내면 «서버가 고장났다»로 읽힙니다.

> **AI 에이전트도 같은 문으로.** 플랫폼의 MCP 도구는 내부적으로 이 게이트웨이와 **같은 권한 · 같은 로그 · 같은 상한** 아래에서 실행됩니다. 에이전트라고 더 볼 수 있는 길이 없습니다.

연결 설정 하나 (plantpulse-jdbc.properties)

```
jdbc.cassandra.url = jdbc:cassandra://${PP_HOST_IP}:9042/pp
jdbc.postgres.url  = jdbc:postgresql://${PP_HOST_IP}:5432/plant
pulse
jdbc.spark.url     = jdbc:hive2://${PP_HOST_IP}:10000/
jdbc.tse.url       = jdbc:ts://${PP_HOST_IP}:7800
cache.host         = ${PP_HOST_IP}
```

14 · 통제 · 감사

열되 다치지 않게 — 무엇을 막고 무엇을 남기나

스키마를 바꾸는 문장은 막는다

```
DROP · TRUNCATE · ALTER · CREATE · GRANT · REVOKE

/* 안녕 */ DROP TABLE pp.mm_tag   → 막힘
SELECT 1; DROP TABLE pp.mm_tag    → 막힘 (다중 구문)
```

블록 · 줄 주석을 **먼저 걷어낸 뒤** 첫 토큰을 봅니다. 그리고 첫 토큰만 보는 방식은 «한 요청에 문장 하나»일 때만 성립하므로, 세미콜론이 보이면 **내용을 보기 전에** 거부합니다.

상한 — 값과 함께

설정	기본값
query.max.rows	**100,000** 행 (0 = 무제한)
rate.limit	**1,000** req/s (클라이언트별, 고정 윈도)
rate.limit.global	0 = 무제한 — 버킷 폭증에 대한 천장
max.body.bytes	1 MB — 초과 시 **413 조기 거부**
max.params.bytes	64 KB
async.timeout.ms	60,000
async.concurrency.limit	**256** — 가상 스레드, 상한에 닿으면 백프레셔
slow.query / slow.update	10,000 ms / 2,000 ms — 경고 로그

> **잘린 결과는 «성공» 으로 돌아옵니다.** HTTP 는 200 이고 행도 정상입니다 — `truncated` 를 보지 않으면 **잘린 결과를 전체로 착각한 채** 집계하게 됩니다. 그래서 이 플래그를 두 경로 모두에 **항상** 싣습니다. 그리고 상한에 닿으면 거기서 **읽기를 끊습니다** — 끝까지 읽은 뒤 자르는 것이 아닙니다.

`X-Forwarded-For` 는 기본적으로 무시합니다. 직속 peer 가 신뢰 프록시 목록에 있을 때만 첫 hop 을 클라이언트 IP 로 씁니다 — 그렇지 않으면 헤더 하나를 지어내 한도를 우회할 수 있습니다. `/api/*` 와 `/admin/*` 은 **같은 카운터**를 공유합니다.

오류 계약 — 4xx 와 5xx 를 가른다

error_code	HTTP	뜻
AUTH_MISSING / AUTH_INVALID	401	키 누락 · 불일치
PARAM_MISSING	400	헤더 · 언어 · 타입 변환 실패
QUERY_SYNTAX	400	DDL · 다중 구문 · 문법
QUERY_CONSTRAINT / _DEADLOCK	409	무결성 · 교착
QUERY_CONNECTION	503	백엔드에 연결 실패
QUERY_TIMEOUT	504	백엔드 응답 지연

```
{"result":"ERROR","error_code":"QUERY_SYNTAX",
 "message":"DDL statements are not allowed: DROP"}
```

문법 · 제약 · 데이터 · 인증은 **클라이언트 원인(4xx)**, 연결 · 타임아웃은 **인프라 원인(5xx)** 입니다. 하나로 뭉개면 「내 질의가 틀렸나 저장소가 죽었나」를 호출자가 구별할 수 없습니다. 서버 로그도 같은 기준으로 4xx 는 WARN, 5xx 는 ERROR 로 갈라 남깁니다.

모든 질의가 남는다

테이블	`tm_data_gateway_log` (Aster TSDB, timeuuid PK)
남는 것	언어 · 질의문 · 행 수 · 실행 시간 · 실패 여부 · 오류
쓰기	큐 + 배치 플러시 — 질의 경로를 막지 않는다

`/api/health` 는 liveness 가 아니라 **readiness** 입니다 — `database` · `inmemory` · `query_log` 셋이 모두 참이어야 UP 이고, 아니면 `STARTING` · `DEGRADED` 입니다. **감사 기록이 초기화되지 않은 게이트웨이는 UP 이 아닙니다.**

JDBC 드라이버 셋 — 저장소에 «직접» 붙는 길

드라이버	URL · 비고
Aster TSDB JDBC	`jdbc:cassandra://host:9042/pp` — Aster TSDB CQL 함수 그대로, 불변 좌표 `YYYY.MM-YYYYMMDD` 발행
Spark SQL JDBC	Kyuubi Thrift(10000) — thin + standalone jar, BI 도구에서 Iceberg 콜드 계층
TQL JDBC	`jdbc:ts://host:7800` · `jdbc:tss://` — SPI 자동 등록

세 드라이버 모두 고객이 컴파일하는 SDK 라 **이동 포인터(YYYY.MM)와 불변 좌표를 함께** 발행하고, 공개 API 는 `abiCheck` (직전 발행본 대비 ABI 비교)를 통과해야 나갑니다. 자바 애플리케이션용으로는 게이트웨이 SDK(`plantpulse-data-gateway-client`)가 따로 있습니다.

15 · 확장 · 복제

토큰 링 — 워커를 꽂고 `pd node add`



그림 3. 콘솔 «토폴로지» — 마스터 10.99.0.100 + 워커 10.99.0.101, 빈 슬롯 4. 노드 상세: 링 정상 · datacenter1/rack1 · 부하 8.37 GiB · 토큰 16 · 키스페이스 `pp` 유효 소유 48.8%. *2026-09-08 실측.*

절차 — 셋

1. 워커 컨테이너 기동(`PP_MODE=WORKER` , 같은 이미지). `pd status` 로 storage · analytics · admin-api RUNNING 확인.
2. 마스터에서 `pd node status` — 새 노드가 `UN` (Up·Normal). `UJ` 면 아직 합류 중.
3. 마스터에서 `pd node add` — 살아 있는 노드 수에 맞춰 RF 조정 + `nodetool repair` . 셋째 노드가 붙으면 한 번 더.

```
$ pd node add
[INFO] [node] cassandra: change replication factor + repair ...
[INFO] [node] datacenter=datacenter1 live_nodes=2 -> replicatio
n_factor=2
[WARN] [node] the ring has 2 node(s), so RF is 2. It rises to 3
when
       the third node joins and pd node add runs again - ...
[INFO] [node] cassandra: nodetool repair ...
```

규칙: 노드 2대 → RF 2, 3대 이상 → RF 3(그 이상 올리지 않음 — RF > 노드 수면 QUORUM 질의가 전부 실패). 데이터센터 이름이나 노드 수를 못 읽으면 **exit 1**.

무엇이 어떻게 복제되나

구성요소	방식
Aster TSDB	토큰 링(`num_tokens: 16` , RF 인지 토큰 배정) · NetworkTopologyStrategy · GossipingPropertyFileSnitch · hinted handoff
PostgreSQL	스트리밍 복제 — `wal_level = logical` , 슬롯 `replica_worker_N` , `max_slot_wal_keep_size = 8GB` (슬롯이 WAL 을 무한히 붙잡지 못하게), `max_wal_size = 4GB`
Valkey	프라이머리 → 레플리카, 뒤처짐(bytes) 콘솔 표시
Spark	워커 노드가 Spark 워커로 합류(7077)
명부	`/etc/kopens/workers.roster` — 콘솔 «워커» 화면이 명부 · 링 · PG 슬롯 · Redis · 헬스를 한 줄로 판정

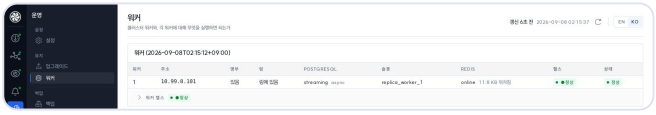


그림 4. 워커 화면 — 1번 워커 10.99.0.101: 명부 있음 · 링에 있음 · PostgreSQL streaming(async) · 슬롯 `replica_worker_1` · Redis online(11.8 KB 뒤처짐) · 헬스 정상.

> **마스터 1 + 워커 최대 5.** 서비스를 세우지 않고, 이관 없이 늘리는 것이 목적입니다. 늘릴 시점은 콘솔 «스토리지»(볼륨 사용률) · «메트릭»(CPU · 메모리 · 디스크 I/O)이 미리 보여줍니다.

16 · 운영

콘솔 · pd · JSON 계약

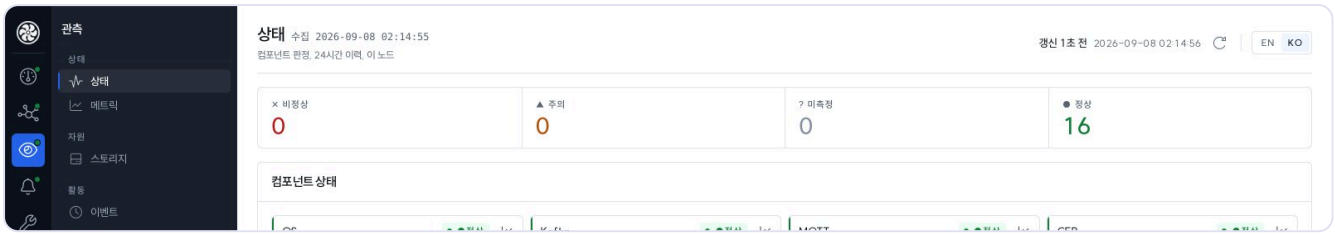


그림 5. 콘솔 «상태» — 16 컴포넌트 각각 «무엇을 쟀고 임계가 얼마인지»(Kafka 오프라인 파티션 0/0 · PG 최장 문장 26 ms/5000 · Aster TSDB 대기 컴팩션 0/200 · TSE 큐 626/40000).

콘솔 열두 화면 · 다섯 묶음

개요	대시보드 — 한 줄 판정 · 활성 알람(연속 3회 임계) · 컴포넌트 헬스 · 24/48h 타임라인
관측	상태 · 메트릭(1h~7d) · 스토리지 · 이벤트 · 로그 · 진단 · 작업
운영	설정(렌더 · diff) · 업그레이드 · 워커 · 백업
도구 · 기록	8 웹 화면 바로가기(상태 포함) · 감사

레일 아이콘의 점은 «그 묶음의 최악 상태». 못 잰 묶음은 점을 그리지 않습니다.

메트릭 출처 — JMX 와 넷

컴포넌트	경로
Kafka · Aster TSDB · PG · MQTT · CEP · TSE · …	JMX (admin-api `jmx/impl`)
Spark	`:4440/metrics/master/prometheus/` (슬래시 필수 — 없으면 302)
Temporal · Kestra	`127.0.0.1:8244/metrics` · `:8391/prometheus` (루프백)
Valkey	RESP `INFO`

`pd` — 동사 하나에 파일 하나

```
pd start | stop | restart | status [service]
pd config render | diff [--templates] | list
pd secret rotate VAR=value ... | probe-mqtt
pd node add | status | repair | info
pd backup [list|schedule|status|run|restore]
pd recover | logs | storage | retention | flow | doctor | env
```

`--json` 계약 (json-contract.md)

```
{ "schema":"pd.doctor/1", "verb":"doctor",
  "generated_at":"2026-09-08T02:10:58+09:00",
  "status":"ok", "reason":null, "exit_code":0,
  "mode":"MASTER",
  "checks":[ {"id":"core ports","result":"pass","reason":null},
             {"id":"worker volumes","result":"na",
              "reason":"docker socket not mounted"} ],
  "summary":{"pass":31,"fail":0,"na":2} }
```

- stdout 은 JSON 하나뿐 · 시크릿 `***` 마스킹 · `na` 는 `reason` 필수 · `pass+fail+na == len(checks)` · `exit_code` == 종료 코드

종료 코드

0 / 1 / 2	성공 / 실패·차이 있음 / 사용법 오류
3 / 4	필수 시크릿 누락(이름 전부 출력) / 렌더 실패(치환 안 된 `${PP_*}` 전부 출력)
5 / 6 / 7 / 8	문지기 서비스 미기동 / 일부 서비스 미준비 / 정지 미증명 / 광고 주소 루프백

17 · 보호

백업 · 복구 · 업그레이드



그림 6. 콘솔 «백업» — PostgreSQL(pgBackRest) 마지막 full 2026-09-06 01:00, differential 매일 01:30, 세트 8개(33.6–36.4 MB), 세트마다 «복원» 버튼. Aster TSDB(Medusa)가 같은 모양으로 이어집니다.

스케줄 — systemd 타이머 10 유닛, 컨테이너 안

잡	주기
PostgreSQL diff / full	매일 01:30 / 일요일 01:00 (pgBackRest)
Aster TSDB diff / full	매일 01:20 / 일요일 00:10 (Medusa)
purge	일요일 03:00 — `repo1-retention-full` · Medusa `max_backup_age`

```
sudo CQL_PASS='…' backup.sh --install     # /etc/pgbackrest/pgbackrest.conf · /etc/medusa/medusa.ini (600)
backup.sh --schedule-install | --schedule-show   # JSON: 유닛 · 스위치
backup.sh --postgres diff | --cassandra full | --purge
pd backup run --engine postgres --type full --json
pd backup restore --engine cassandra --set full_20260906_0010 --dry-run --json
```

- `stanza-create` 와 `check` 는 게이트 — 하나라도 실패하면 exit 1, 재실행은 멱등
- 스위치가 꺼져 있으면 도구를 건드리지 않고 `SKIPPED BY CONFIGURATION` · `result=skipped` 를 **기록**
- 백업 시점에는 «도구가 있는가»만 묻고 없으면 `install-deps.sh` 를 가리키며 실패 — 새벽 백업이 `dnf install` 을 돌리던 경로는 제거
- 백업 루트 기본 `/data1/pp-backup` (`BACKUP_ROOT`) — **별도 디스크 · NAS 권장**, 외부 S3 동기화는 스토리지 분리 시점의 설계 항목

부분 유실 — 데이터 복구 도구

```
java -jar plantpulse-data-recovery.jar \
  -asset_id=ASSET001,ASSET002 -start=20250101 -end=20250105
# tm_tag_point 의 태그 포인트로 tm_asset_data 를 재구성
# 일 → 시/분/초 → 태그 3단계 병렬, 시점별 최신값 조회
```

업그레이드 — 이미지 교체 한 번

- 콘솔 «업그레이드»가 **이미지 태그 · build.version · 굽기 커밋 · 시각**과 바꾸는 명령을 보여줌
- 설정은 템플릿에서 매 기동 렌더 → 컨테이너 안 수정은 남지 않음 (드리프트 0)
- 데이터는 볼륨(`PP_DATA_DIR · PP_BACKUP_DIR …`), 설정은 `/etc/kopens/conf` — 이미지와 분리돼 있어 **이전 태그로 되돌리면 끝**
- DB 스키마는 Flyway 가 기동 시 적용

공급망

이미지 서명	cosign — 레지스트리 반영 후 다이제스트 서명
폐쇄망 번들	반입 전 체크섬 + 서명 검증, 실패 시 설치 중단
이미지 내부 검증	src 오버레이 173 파일 + dist 매니페스트 `sha256sum -c`
입력 커밋 추적	dist 아티팩트 커밋 ≠ clone HEAD 면 빌드 거부 (최대 300초 대기 후)

> **운영 사고에서 배운 규칙이 코드가 됐습니다.** «클래스패스가 모자란 채 exit 0 으로 성공 보고»(2026-09-04)를 막는 회귀 테스트, «없는 잡»과 «0행 아카이브»를 로그에서 갈라 적는 아카이버 — 이 백서의 여러 «반드시 실패한다»는 그런 사고의 기록입니다.

18 · 실측

가동 중인 인스턴스 — 2026-09-08 02:14 KST, dev 41

아래 수치는 콘솔이 그 시각에 보여준 값 그대로입니다. 부하 시험이 아니라 «문서상이 아니라 돌고 있다»는 증거입니다.

16 / 16
컴포넌트 정상 · 서비스 19/19 · 알람 0

2 / 2
노드 — 마스터 + 워커, 연동 정상

48.8%
키스페이스 pp 유효 소유(마스터) · 토큰 16

2.8 / 4.7 TB
볼륨 사용(59%) — PP_HOME · DATA · TEMP · BACKUP

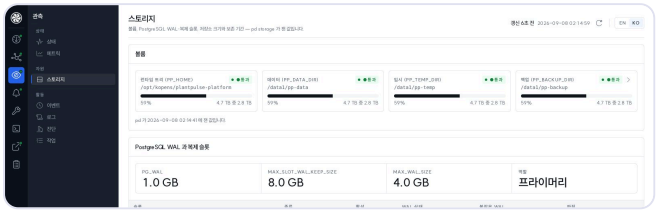


그림 7. «스토리지» — `pd storage` 실측: PG_WAL 1.0 GB · MAX_SLOT_WAL_KEEP_SIZE 8.0 GB · MAX_WAL_SIZE 4.0 GB · 역할 프라이머리 · 슬롯 `replica_worker_1` physical/활성/통과.

Aster TSDB 디스크	크기
데이터 (키스페이스 pp)	12.3 GB
커밋 로그	75.1 MB
스냅샷	3.9 GB
system · system_auth · schema …	7.8 MB · 154.9 KB · 1.2 MB

Kafka · 모니터 보존	값
kraft_combined_logs	191.2 MB · 보존 1시간
콘솔 모니터 보존	메트릭 7일 · 이벤트 30일 · 작업 30일 · 헬스 1일

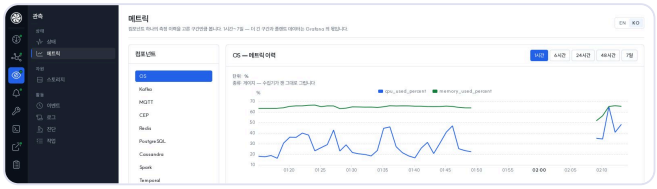


그림 8. «메트릭» — OS 1시간: CPU 15–47%, 메모리 ~65%, 디스크 쓰기 피크 ~135 MB/s, 디스크 총 5 TB / 사용 3 TB. 회색 구간은 «그릴 것이 없다»고 서버가 답한 곳 — 짐작해 잇지 않습니다.

노드 부하 (마스터)	값
CPU / MEM / DISK	36% / 65% / 59%
Aster TSDB 부하 · 대기 컴팩션	8.37 GiB · 0 (임계 200)
PostgreSQL 최장 문장	26 ms (임계 5000)
TSE 인제스트 큐	626 (임계 40,000)
Kafka 오프라인 파티션 · CEP 드롭	0 · 0
Temporal 최장 대기 · Kestra 힙	0 s · 52–61%

백업 (콘솔)	값
pgBackRest 리포지터리	3.4 GB · 세트 8개 · 통과
Medusa 리포지터리	26.8 GB · 최근 세트 «미완료» 표시 — 콘솔이 숨기지 않는 실패

정직하게. 이 인스턴스는 개발 · 스테이징 겸용이라 Medusa 최근 세트가 미완료로 잡혀 있었습니다. 그 사실이 콘솔에 빨간 ✕ 로 떠 있는 것 자체가 «미측정 · 실패를 초록으로 칠하지 않는다»는 원칙의 증거라 그대로 실었습니다.

19 · 범용 데이터레이크와 비교

범용 레이크하우스 · 클라우드 IoT 시계열과 어디가 다른가

Databricks · Snowflake 류의 범용 레이크하우스와 AWS IoT SiteWise / Timestream 류의 클라우드 IoT 서비스는 훌륭한 «분석 목적지»입니다. 산업 «데이터 인프라»로 놓았을 때의 차이만 적습니다.

축	PlantPulse Industrial DataLake	범용 레이크하우스 (Databricks · Snowflake 등)	클라우드 IoT 시계열 (AWS SiteWise · Timestream 등)
배포	● 온프레미스 · 완전 폐쇄망 단일 이미지, 오프라인 번들 · 서명 검증	◐ 클라우드 SaaS(퍼블릭 클라우드 위), 온프레미스 옵션은 제한적	○ 해당 클라우드 전용
산업 프로토콜 수집	● Edge 42종 + Kafka · MQTT/Sparkplug B · Prometheus remote_write	○ 없음 — 별도 수집 계층 필요	◐ 게이트웨이(OPC UA 등) 제공, 클라우드로 업로드
시계열 저장 · 계산	● 시계열 전용 저장 엔진(7.1×) + CQL 21 함수 + TSE 롤업/이상치	◐ Parquet 위 SQL — 시계열 함수는 일부, 인제스트는 배치 지향	● 시계열 특화 저장 · 집계
설비 맥락 · 이벤트 · BLOB	● 자산 계층 · 6종 데이터 클래스 · 알람/이벤트/명령/BLOB 한 클러스터	◐ 테이블로 모델링은 가능, 표준 모델은 앱 몫	◐ 자산 모델 제공, 이미지 · 문서는 별도 스토리지
장기 보관 포맷	● Iceberg v2 · Parquet — 어떤 엔진도 읽음, 반출에 «내보내기» 불필요	● Delta / Iceberg 오픈 포맷	◐ 서비스 내부 저장, 내보내기로 반출
과금 구조	● 구성(노드 · 사이트) 기준 — 저장량 · 질의 횟수에 비례하지 않음	◐ 컴퓨트 · 저장 사용량 과금(크레딧)	◐ 수집 건수 · 저장량 · 질의 스캔량 과금
실시간 판정	● 자체 엔진 Aster CEP — EQL(SQL-92 + 스트리밍 확장) · SQL:2016 행 패턴 인식, 별도 클러스터 없음	◐ 스트리밍은 별도 제품 · 잡	◐ 알람 규칙(단순 임계 중심)
운영 주체	● 고객 인프라, 콘솔 · `pd` · 백업 포함	● 벤더 관리형(운영 부담 낮음, 통제 낮음)	● 벤더 관리형

함께 쓰는 구성

사이트에는 PlantPulse DataLake 를 두고, 전사 분석은 콜드 계층의 Iceberg 테이블을 **그대로** 범용 레이크하우스가 읽게 하는 구성이 가능합니다 — 카탈로그(Gravitino Iceberg REST)를 공유하면 복사도 필요 없습니다. 클라우드로는 집계 · 지표만 올려 사용량 과금을 통제합니다.

범용 제품이 맞는 경우

공정 원시 데이터가 이미 클라우드에 있고 보안 정책이 반출을 허용하며, 시계열보다 **관계형 · 대규모 조인 · ML 파이프라인**이 중심인 조직. 이 경우 PlantPulse 는 «현장 수집 · 표준화 · 콜드 아카이브 생산자»로 앞단에 놓입니다.

경쟁 · 범용 제품 항목은 각 사 공개 문서의 일반적 특성을 요약한 것이며(2026-09-09), 특정 버전 · 요금제에 따라 다를 수 있습니다. ● 기본 제공 ◐ 옵션 · 별도 · 제한 ○ 해당 없음

20 · 규모 산정 · 출처

규모 산정 절차와 출처

입력 넷 → 노드 구성

입력	무엇을 정하는가
태그 수 × 수집 주기	초당 행 수 → 인제스트 · memtable · 컴팩션 부하, 노드 수의 1차 기준
보존 기간 (핫 행 · 청크 · 콜드)	웜 디스크(`hot_window` 행 + 청크) · 콜드 오브젝트 스토리지 용량
동시 사용자 · 대시보드	조회 부하 → TSE · Kyuubi · Data Gateway 동시성(256)
CEP 규칙 수 · 이벤트율	실시간 판정 부하 → CEP 힙 · Valkey 스트림

디스크 산정 예시 (Aster TSDB 벤치 B/row 기준)

```
10만 태그 × 1초 = 8.64 × 10⁹ 행/일
행 저장(hot_window)  ≈ 11.9 B/row → ≈ 103 GB/일
청크 후(warm)        ≈  1.7 B/row → ≈  15 GB/일

보존 90일   : 7일 핫 + 83일 청크 ≈ 0.72 + 1.2  ≈ 1.9 TB  (RF 3 → ×3)
보존 365일  : 7일 핫 + 358일 청크 ≈ 0.72 + 5.4  ≈ 6.1 TB  (RF 3 → ×3)
보존 3650일 : 7일 핫 + 3643일 청크 ≈ 0.72 + 54.6 ≈ 55 TB   (RF 3 → ×3)  ← 출하 기본값
콜드(Iceberg zstd)   별도 산정 — Parquet 압축률은 값 분포에 좌우
```

위 수치는 Aster TSDB 벤치(20M 행 운영 스키마)의 바이트/행을 단순 곱한 **예시**이며, 실제는 태그 값 분포 · 컬럼 수 · 복제 계수에 따라 달라집니다. **보존 기간이 용량을 결정하는 가장 큰 손잡이**입니다 — 출하 기본값 3650일을 그대로 두면 위 표의 마지막 줄이 되므로, 현장 요구에 맞춰 `storage.tag.point.ttl` 을 먼저 정하십시오. PoC 에서 `pd storage --json` 으로 일주일 실측한 뒤 산정하는 것을 권합니다.

런타임 기본값에서 읽는 최소 사양

Aster TSDB 힙 16 GB(`-Xms16G -Xmx16G`) + memtable off-heap 2 GiB, PostgreSQL `shared_buffers 4GB` (≈RAM 25% 가정), Spark driver 2 GB · executor 코어당 메모리 자동, TSE 최소 4 GB. 마스터 노드의 최소 사양은 **16 vCPU · 128 GB RAM · 1 TB SSD** 이며, 정확한 사양은 위 입력 넷으로 산정합니다.

출처

- **kopens/aster-tsdb** README · `doc/timeseries/` (tiering-benchmark · codec-bakeoff · production-rollout) — TSCS · 청크 코덱 v4 · 21 함수 · 벤치마크 · 제약
- **plantpulse-cep-aster-engine** — 엔진 소스 · `MATCH_RECOGNIZE` 시험(Flink 2.3.0 대조) / **plantpulse-cep** `docs/capability-matrix.md` (생성물) · `differential-harness.md`
- **plantpulse-database-model** — `cassandra/CASSANDRA.md` · `ddl.cql` (104 테이블 · UCS · 압축 · SAI) · `spark/ICEBERG.md` · `ddl.sql`
- **plantpulse-datalake-cli** — `services/order.txt` · `config/render.map` · `config/templates/*` · `docs/PD-USAGE.md` · `docs/json-contract.md` · `docs/metrics-endpoints.md`
- **plantpulse-datalake-docker** README — 출처 넷 · 오버레이 · 이미지 내 sha256 검증 · dist 입력
- **plantpulse-storage · timeseries · analytics · messaging · workflow** README · `UPSTREAM-LOCK.md` — 버전 실측
- **plantpulse-timeseries-engine-server** docs — TQL JDBC · Prometheus 어댑터 · 클라이언트 · Web UI
- **plantpulse-data-gateway-web** README — 엔드포인트 · 통제 · 오류 계약
- **plantpulse-archive-job · warehouse-web** README — 8잡 · SPI · Temporal 스케줄 · Optimizer
- **plantpulse-backup** README · `backup.sh` — 스케줄 · 설정 경로 · 게이트
- **plantpulse-datalake-admin-web** README — 콘솔 화면 구성 · 원칙 / 콘솔 실화면 2026-09-08 dev 41
- InfluxData 블로그(InfluxDB 3 Core 72h 제한 · Enterprise) · TimescaleDB 2.14 릴리즈 노트(멀티노드 지원 종료) · Timescale 라이선스 문서 — 2026-09-09 조회

> **읽는 법.** 경쟁 · 범용 제품 항목 외의 모든 수치는 위 저장소의 특정 커밋 시점 값입니다. 제품은 월 단위 캘린더 버전으로 바뀌므로 검토 시점의 `UPSTREAM-LOCK.md` 와 콘솔 «업그레이드» 화면으로 재확인하십시오.

문의

전화 · 이메일	1588-5268 · webmaster@kopens.com
문의 · 데모	kopens.com/ko/contact
제품 · 문서	plantpulse.io · docs.plantpulse.io
저장 엔진 소스	github.com/kopens/aster-tsdb (Apache-2.0)

㈜한국오픈솔루션 · KOPENS, Inc. © 2026 KOPENS. 플랜트펄스®는 ㈜한국오픈솔루션의 등록상표입니다. 본 문서 Rev. 2026.09.