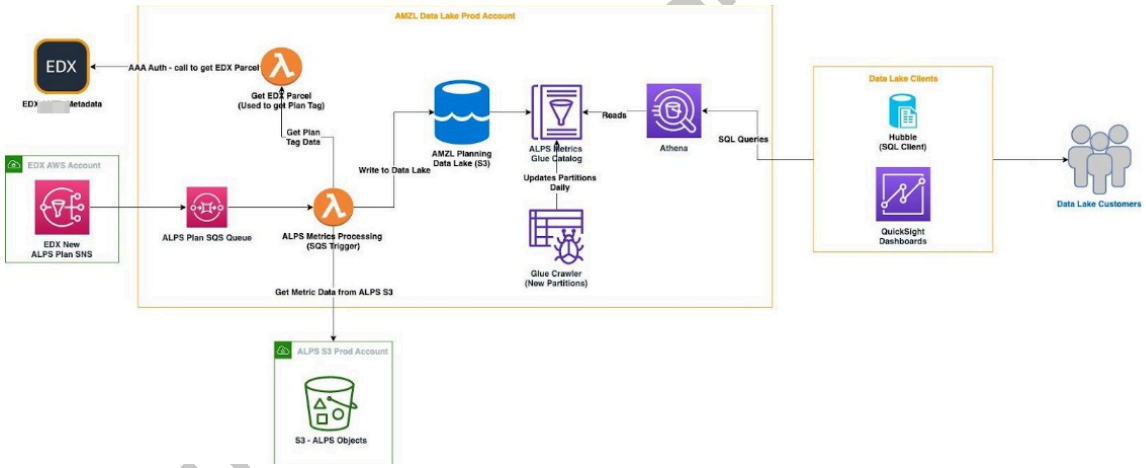
案例：物流人力规划数据中台建设（英文原版请下拉）

1. 项目背景

本项目是物流人力规划系统 SPLA 在三年内进行的第三次重大基础设施重构。此次重构的核心目标是解决多个报表之间数据来源分散、指标口径不一致、刷新频率不统一等问题，同时满足业务对准实时（15 分钟）指标更新能力的迫切需求。

为更好理解这些挑战的根源，需先了解此前的数据摄取流程架构。当新的劳动力计划发布后，EDX 服务会通过 SNS 消息发送元数据，进而触发 SQS 队列和一个下游的 Lambda 函数。该 Lambda 会解析出 `plan_id`，并从 SPLA 的 S3 存储中拉取 80 余个原始 CSV 文件，将其转为分区 Parquet 格式并保存至基于 S3 的数据湖中。随后，Glue Crawler 会扫描这些分区并更新 Schema，使数据可通过 Athena 查询。最终，分析师及业务用户通过 SQL 工具或 QuickSight 报表消费这些数据。



但这一架构存在以下核心痛点：

- 不同报表依赖不同数据源、刷新频率和指标定义，导致指标不一致、KPI 不对齐；
- 缺乏统一的中间层来标准化数据转换及指标聚合逻辑；
- 基于事件触发的摄取机制使调试难度大、成本高、运维复杂。

为了解决上述问题，我们对数据平台进行全面重构，设计了具有明确分层、支持 15 分钟定时刷新、可维护性强且性能更优的通用型平台架构。

2. 解决方案架构

以下展示该可扩展数据平台从设计到落地的完整实现流程：

[上游数据源]
├── SPLA 模型（计划数据）
├── Orbit 系统（实际数据）

核心三层架构：

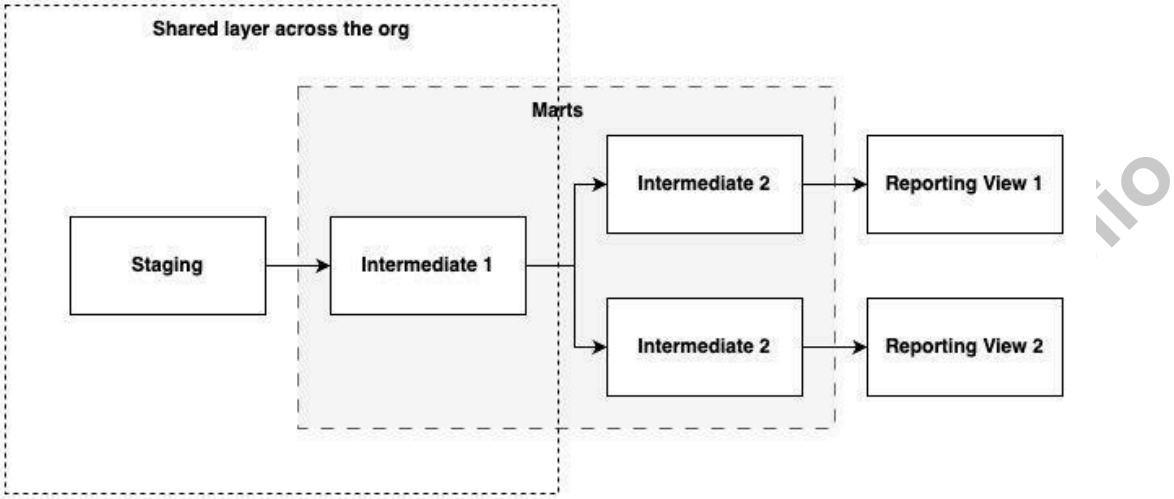
[Staging 层]
├── 在 S3 数据湖中按分区存储原始计划数据与实际数据

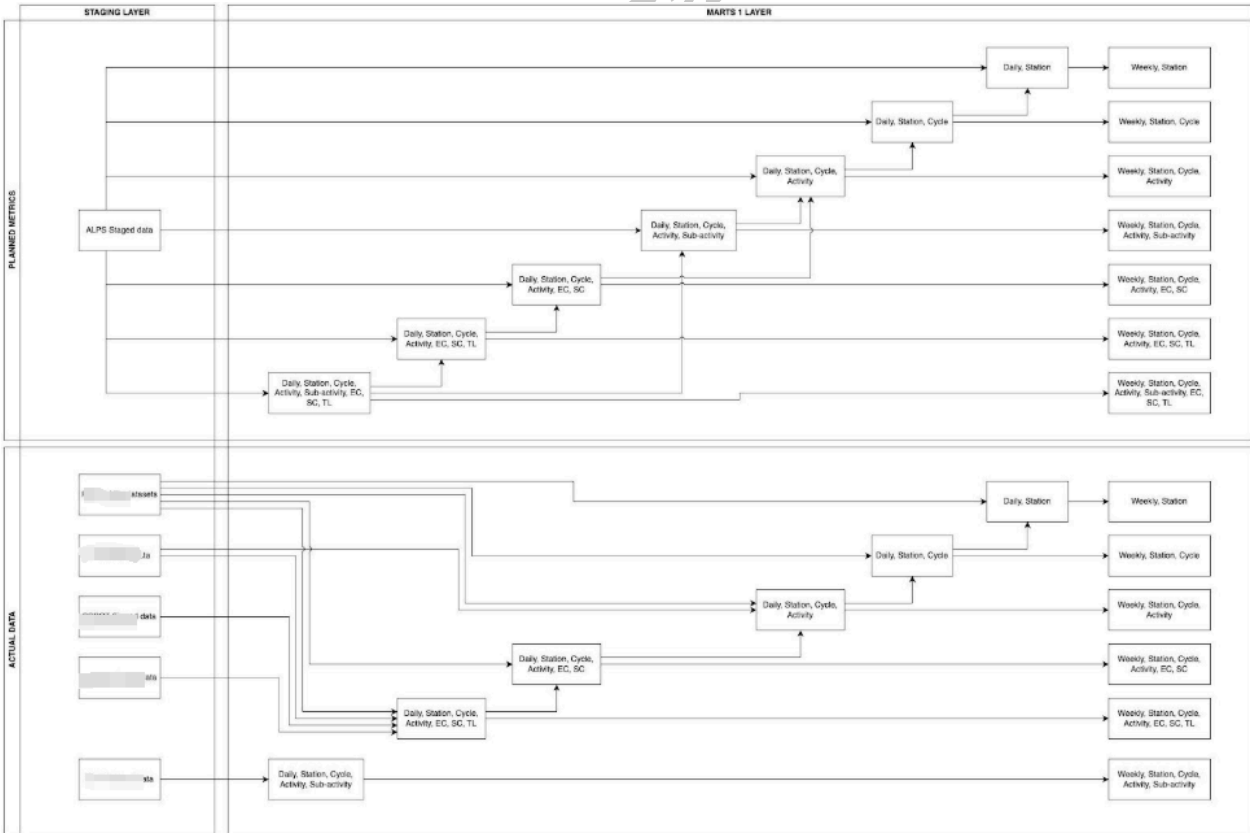
[Mart 层]
├── 实现统一的指标转换与聚合逻辑

[Reporting 层]
├── QuickSight 报表
└── API 接入 / 分析师查询端

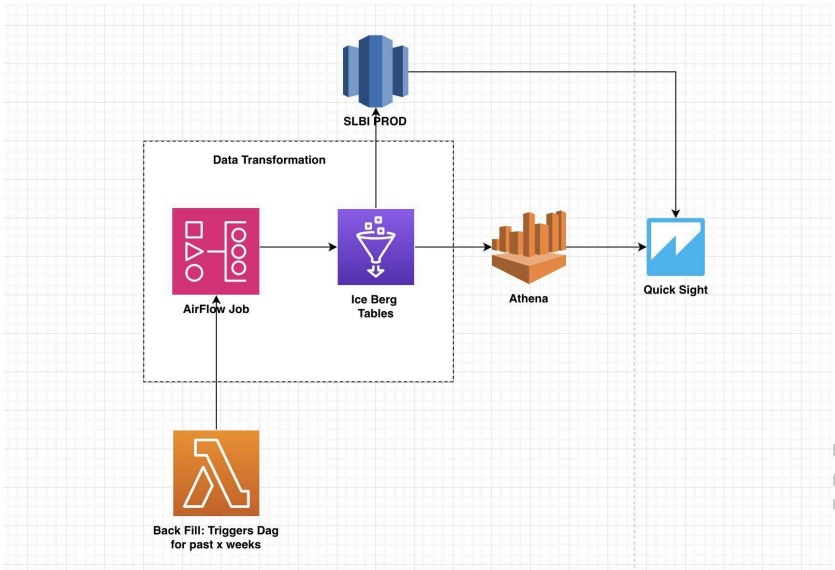
执行引擎：
[Airflow DAG（每 10 分钟运行一次）]
├── 捕捉过去 20 分钟内的新增计划数据
├── 创建 CTAS 临时快照表（Athena SQL）
├── 将新增数据合并进 Iceberg 数据集市
├── 删除过期 plan_id 数据
├── 生成下游可消费的汇总报表表
└── 通过 Glue API 清理 CTAS 临时表

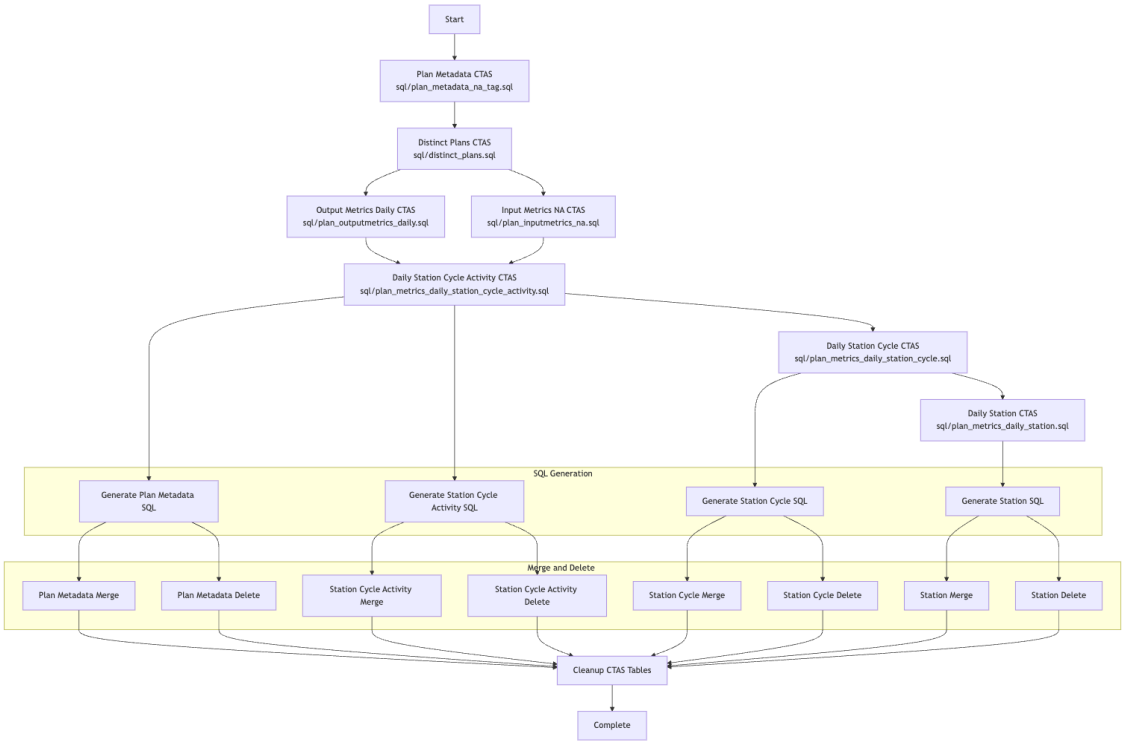
核心三层架构


Shared layer across the org
Marts
Staging
Intermediate 1
Intermediate 2
Reporting View 1
Intermediate 2
Reporting View 2


STAGING LAYER
MARTS 1 LAYER
PLANNED METRICS
ACTUAL DATA
ALPS Staged data
Daily, Station
Weekly, Station
Daily, Station, Cycle
Weekly, Station, Cycle
Daily, Station, Cycle, Activity
Weekly, Station, Cycle, Activity
Daily, Station, Cycle, Activity, Sub-activity
Weekly, Station, Cycle, Activity, Sub-activity
Daily, Station, Cycle, Activity, EC, SC
Weekly, Station, Cycle, Activity, EC, SC
Daily, Station, Cycle, Activity, EC, SC, TL
Weekly, Station, Cycle, Activity, EC, SC, TL
Daily, Station, Cycle, Activity, Sub-activity, EC, SC, TL
Weekly, Station, Cycle, Activity, Sub-activity, EC, SC, TL

執行引擎


SLBI PROD
Data Transformation
AirFlow Job
Ice Berg Tables
Athena
Quick Sight
Back Fill: Triggers Dag for past x weeks


Start
Plan Metadata CTAS sql/plan_metadata_na_tag.sql
Distinct Plans CTAS sql/distinct_plans.sql
Output Metrics Daily CTAS sql/plan_outputmetrics_daily.sql
Input Metrics NA CTAS sql/plan_inputmetrics_na.sql
Daily Station Cycle Activity CTAS sql/plan_metrics_daily_station_cycle_activity.sql
Daily Station Cycle CTAS sql/plan_metrics_daily_station_cycle.sql
Daily Station CTAS sql/plan_metrics_daily_station.sql
SQL Generation
Generate Plan Metadata SQL
Generate Station Cycle Activity SQL
Generate Station Cycle SQL
Generate Station SQL
Merge and Delete
Plan Metadata Merge
Plan Metadata Delete
Station Cycle Activity Merge
Station Cycle Activity Delete
Station Cycle Merge
Station Cycle Delete
Station Merge
Station Delete
Cleanup CTAS Tables
Complete

- CTAS 快照表：创建临时表（`Create Table As`）捕捉当前数据快照；
- 动态 SQL 合并与删除逻辑：Python 自动解析 Glue 中的 Schema，生成字段感知的 `MERGE` 和 `DELETE` 语句；
- 合并逻辑：主键匹配时更新数据，不匹配则插入新行（UPSERT 模式）；
- 删除逻辑：清理 mart 层中过期 `plan_id` 数据，仅保留最新版本；
- CTAS 表清理：每轮 DAG 运行结束后，调用 Glue API 自动删除临时表，防止 Catalog 冗余过重。

架构更新对比

本次重构的核心变更之一，是从传统的事件触发（EDX）机制迁移至基于 Airflow 的定时调度（每 10 分钟）方案。每轮任务处理过去 20 分钟的计划数据，带来了显著的运维与性能优化，具体对比如下：

对比项	EDX 事件触发（每个文件触发一次，约 8 万次/周）	Airflow 定时调度（每 10 分钟一次，约 1 千次/周）
执行次数	❌ 极多，难以追踪单次失败	✅ 明确可控，分批处理更易排查
调试难度	❌ 多次触发带来调试难度	✅ 批量失败更易定位问题
异常处理	❌ 每个计划需配置 DLQ	✅ 批量回滚逻辑更高效
运维复杂度	❌ 高频执行需频繁监控	✅ 固定调度降低运维负担
成本效率	❌ 频繁 Lambda / Athena 查询带来高开销	✅ 批量处理显著节约查询成本
Athena 查询负载	❌ 单计划一查询，查询次数多	✅ 批量化查询大幅提升效率
系统稳定性	❌ 高并发易引发超载和失败	✅ 调度稳定，压力控制更强
延迟	✅ 实时触发，延迟最小	❌ 固定调度，约 10 分钟延迟
运维维护性	❌ 逻辑复杂，依赖繁重	✅ 更简洁，便于管理

3. 项目难点总结

- 频繁的 Schema 变动：SPLA 系统经常新增字段，需具备 Schema 自适应能力；
- 指标定义频繁调整：业务需求常变化，ETL 逻辑必须配置化、模块化；
- 15 分钟刷新约束：全流程必须在 15 分钟内完成，满足准实时看板需求；
- 非可加指标聚合：部分指标（如 capacity）需用 min/max 聚合策略，不能简单累加；

4. 项目管理方式

本项目采用 Scrum 式敏捷管理，使用如下结构化层级追踪：

	A	B	C	D	E	F	G	H
1	Milestone	Epic	Tasks	Sub-tasks	Sizing (*S = 1 day, M = 2-3 day, L = 1 Week, XL = 2-3 Week*)	Status	ECD	Owner
2	Data layer development	Scoping	Identify the exhaustive metric list	Identify list of metrics that are currently reported in the dashboard along with the granularity	L			
3			Identify data sources	Identify plan and actual data sources	S			
4			Onboard data sources to the data lake	Onboard data tables on the data lake (preferably not via data lake) but by Andes or data share	M			
5		Building	Build dimension tables	Date dimension	S			
6				Site list dimension	S			
7				Shift code dimension (if required)	S			
8				Other dimension tables as required	M			
9			Build staging tables	Build staging tables per the source (not required for ALPS)	L			
10			Build marts 1 planned tables	Daily, Station, Cycle, Activity, Sub-activity, EC, SC, TL	M			
11				Daily, Station, Cycle, Activity, EC, SC, TL	M			
12				Daily, Station, Cycle, Activity, EC, SC	M			
13				Daily, Station, Cycle, Activity, Sub-activity	M			
14				Daily, Station, Cycle, Activity	M			
15				Daily, Station, Cycle	M			
16				Daily, Station	M			
17				Weekly, Station, Cycle, Activity, Sub-activity, EC, SC, TL	S			
18				Weekly, Station, Cycle, Activity, EC, SC, TL	S			
19				Weekly, Station, Cycle, Activity, EC, SC	S			
20				Weekly, Station, Cycle, Activity, Sub-activity	S			
21				Weekly, Station, Cycle, Activity	S			
22				Weekly, Station, Cycle	S			
23				Weekly, Station	S			
24			Build marts 1 actual tables	Daily, Station, Cycle, Activity, Sub-activity	M			
25				Daily, Station, Cycle, Activity, EC, SC, TL	M			
26				Daily, Station, Cycle, Activity, EC, SC	M			
27				Daily, Station, Cycle, Activity	M			
28				Daily, Station, Cycle	M			
29				Daily, Station	M			
30				Weekly, Station, Cycle, Activity, Sub-activity	S			
31				Weekly, Station, Cycle, Activity, EC, SC, TL	S			
32				Weekly, Station, Cycle, Activity, EC, SC	S			
33				Weekly, Station, Cycle, Activity	S			
34				Weekly, Station, Cycle	S			
35				Weekly, Station	S			
36	Adoption	Replace data sources for existing dashboards	ALPS	Code change	M			
37				Validation	S			
38			G.A.P.	Code change	M			
39				Validation	S			
40			Follow with other dashboards	Code change	M			
41				Validation	S			
42	Deprecation	Deprecate old tables	Internal		M			
43			External		M			

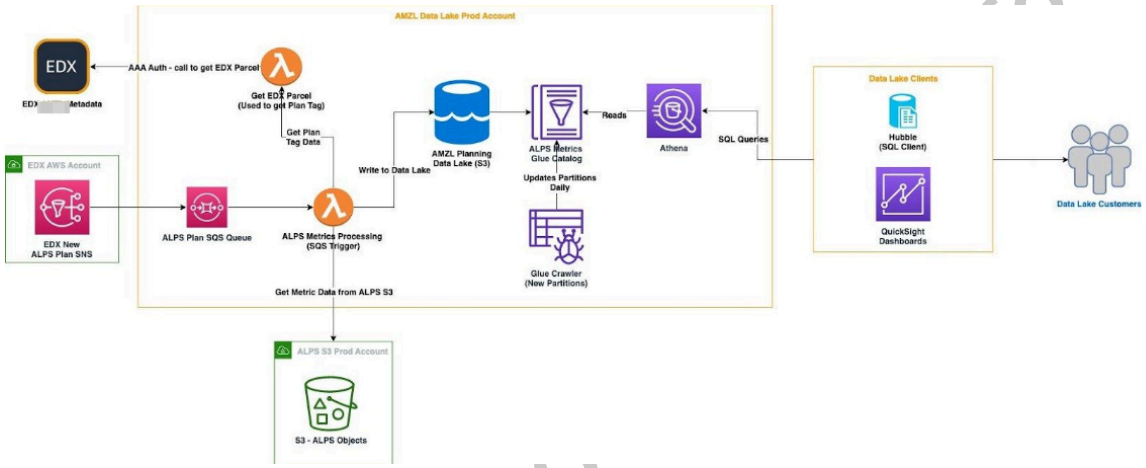
5. 关键成果总结

- 所有报表实现 15 分钟刷新频率一致性；
- 25+ 下游用户通过统一指标集提升分析效率，数据混乱问题减少 80%；
- 优化8 个重复报表被下线；
- ETL 与查询逻辑优化后，Athena 查询成本降低 30%。

Case Study: Labor Planning Data Platform

1. Background

This project marked the third major infrastructure revamp in three years for the workforce planning system, SPLA. The goal was to address fragmented data sources and inconsistent reporting logic across dashboards, while meeting the growing need for near real-time (15-minute) metric updates.



To better understand the source of these challenges, it's helpful to first examine the previous ingestion pipeline architecture. The diagram below illustrates the end-to-end data ingestion and processing flow prior to the revamp. When a workforce plan was published, the EDX service emitted metadata via an SNS message, which triggered an SQS queue and a downstream Lambda function. This Lambda extracted the `plan_id` and pulled over 80 raw CSV files from SPLA S3 storage. The data was then converted into partitioned Parquet files and saved in an S3-based data lake. Glue Crawlers scanned these partitions and registered schema updates, making the data queryable via Athena. Finally, analysts and business users consumed the data through SQL tools and QuickSight dashboards.

However, this architecture posed several issues:

- Multiple dashboards relied on inconsistent sources, metric definitions, and refresh cadences, leading to discrepancies and misaligned KPIs;
- There was no unified layers to standardize transformation and metric aggregation logic
- The event-driven nature of EDX-based ingestion created complexity in debugging, cost inefficiency, and high operational overhead.

To address these pain points, we embarked on a full redesign of the data platform to enable a more maintainable, performant, and near real-time system with structured data layers and a scheduled 15-minute refresh cadence.

2. Solution Design

Below shows the end-to-end design and implementation of a scalable central data platform.

[Upstream Sources]

├── SPLA Model (Plan Data)

├── Orbit (Actual Data)

Core Layers:

[Staging Layer]

├── Store raw planned and actuals data on a Data Lake (S3-based)

[Mart Layer]

├── Apply unified transformation and aggregation

[Reporting Layer]

├── QuickSight Dashboards

└── APIs / Analyst Access

Execution Engine:

[Airflow DAG (Runs every 10 mins)]

├── Capture past 20 mins of new plan data

├── CTAS temp snapshot (Athena SQL)

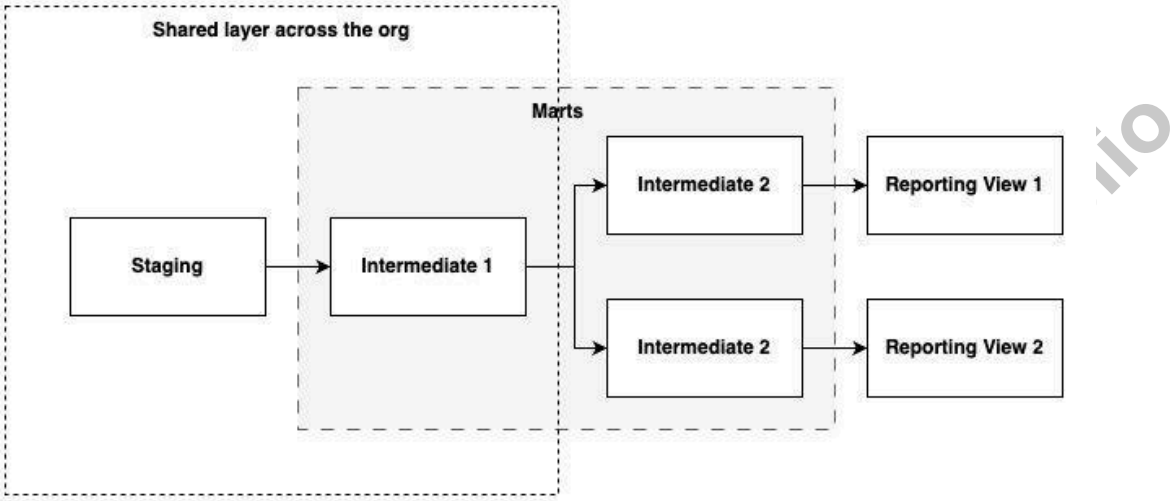
├── Merge new rows into Iceberg mart

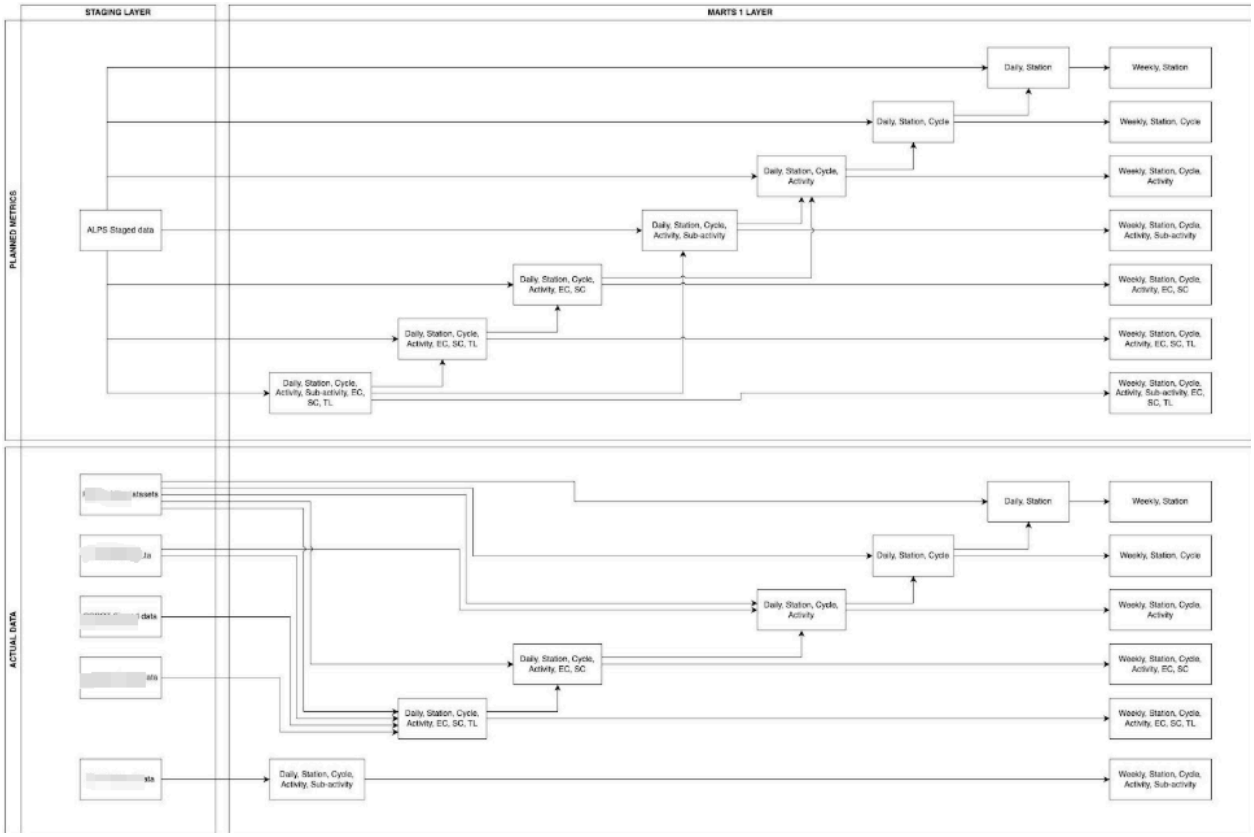
├── Delete outdated plan_id records

├── Materialize reporting tables for downstream consumption

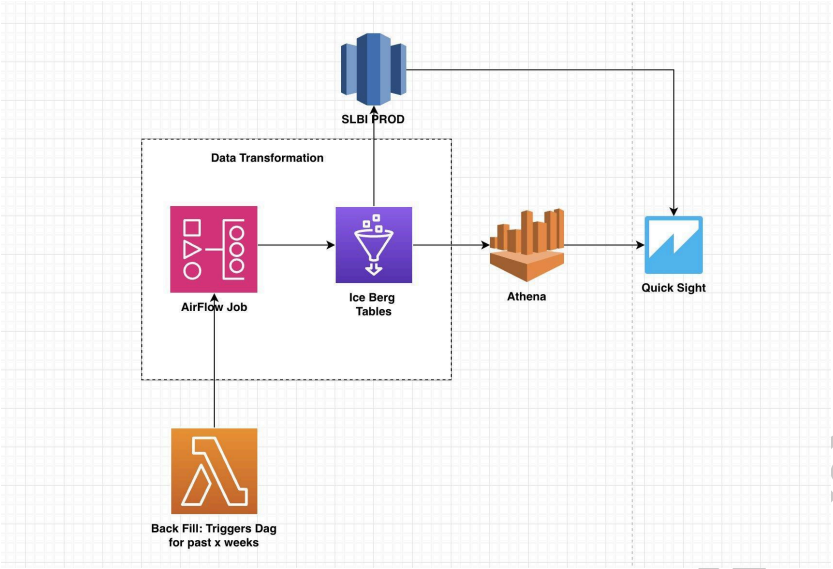
└── Purge CTAS temp tables via Glue API

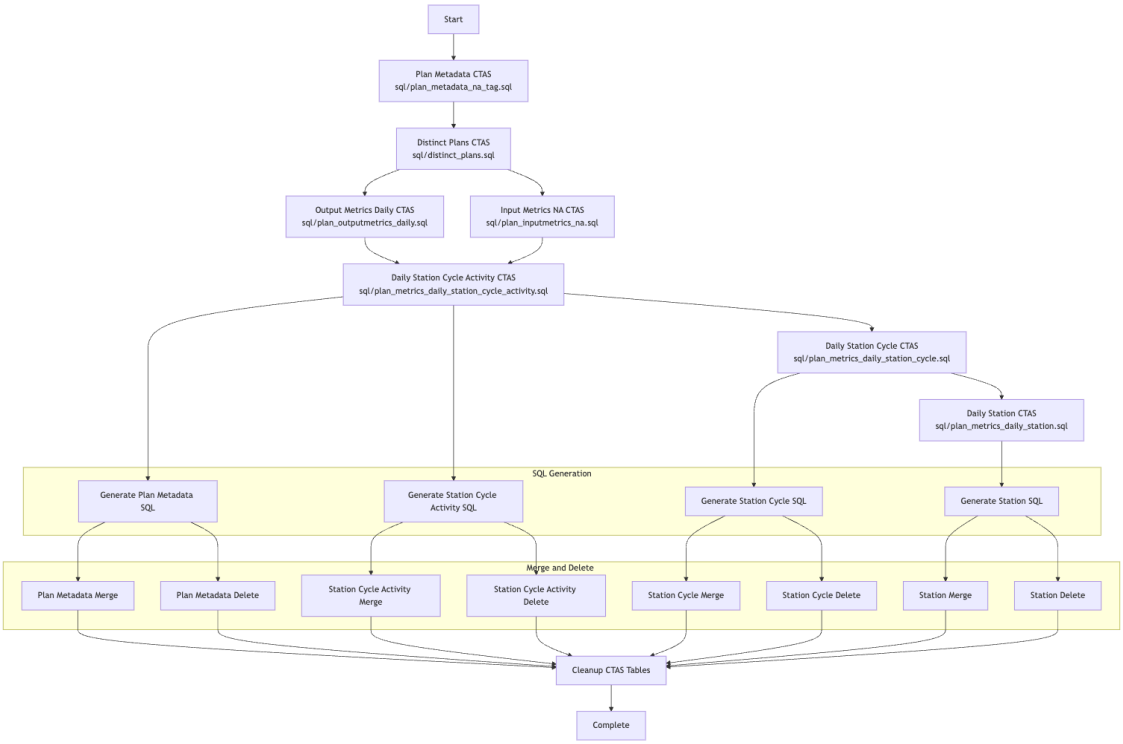
Core Layers


Shared layer across the org
Marts
Staging
Intermediate 1
Intermediate 2
Reporting View 1
Intermediate 2
Reporting View 2


STAGING LAYER
MARTS 1 LAYER
PLANNED METRICS
ACTUAL DATA
ALPS Staged data
Daily, Station
Weekly, Station
Daily, Station, Cycle
Weekly, Station, Cycle
Daily, Station, Cycle, Activity
Weekly, Station, Cycle, Activity
Daily, Station, Cycle, Activity, Sub-activity
Weekly, Station, Cycle, Activity, Sub-activity
Daily, Station, Cycle, Activity, EC, SC
Weekly, Station, Cycle, Activity, EC, SC
Daily, Station, Cycle, Activity, EC, SC, TL
Weekly, Station, Cycle, Activity, EC, SC, TL
Daily, Station, Cycle, Activity, Sub-activity, EC, SC, TL
Weekly, Station, Cycle, Activity, Sub-activity, EC, SC, TL
Daily, Station, Cycle, Activity, Sub-activity

Execution Engine





CTAS Intermediate Table: Create Table AS, capturing snapshot of current data;

- **Merge Logic**: If keys matched, data was updated; if not, new rows were inserted (UPSERT pattern);
- **Delete Logic**: Removed outdated `plan_id`s from the mart to retain only current ones;
- **CTAS Cleanup**: After each run, purge CTAS temp tables and prevent catalog overflow.

A key architectural decision was to shift from a legacy event-driven pipeline to a scheduled 10-minute Airflow DAG. This scheduled approach processed plan data from the past 20 minutes in each run. The change brought major operational benefits, as detailed in the table below:

	A	B	C
1	**Criteria**	**Event-Driven (Triggered per File - 80K/Week)**	**Scheduled (Every 10 Minutes - 10K/Week)**
2	**Number of Executions**	~80,000 per week	~1,080 per week
3	**Ease of Debugging**	❌ Harder → Too many executions, harder to track individual failures	✅ Easier → Fewer runs, failures grouped in batches
4	**Failure Handling**	❌ Requires DLQ for each plan	✅ Easier, failures affect batches, fewer retries needed
5	**Operational Overhead**	❌ High → Frequent executions need more monitoring	✅ Low → Predictable, scheduled runs
6	**Cost Efficiency**	❌ Expensive → More Lambda calls, frequent Athena queries	✅ Cheaper → Batches reduce query & processing costs
7	**Athena Query Load**	❌ High → Many small queries (one per plan)	✅ Low → Batching improves query efficiency
8	**System Stability**	❌ Risk of throttling & failures due to high concurrency	✅ More stable with controlled execution load
9	**Latency**	✅ Real-time processing (Minimal delay)	❌ 10-minute delay in processing
10	**Overall Maintenance**	❌ Harder	✅ Easier

3. Key Challenges

- **Frequent schema changes**: Schema-flexible logic was required to add or modify fields;
- **Business-driven metric volatility**: New metrics or formula adjustments occur frequently, requiring configurable and flexible ETL logic;
- **15-minute refresh**: All must complete within 15 mins to support near real-time reporting;
- **Non-additive metrics**: Many metrics require min/max logic to roll up correctly.

4. Project Management Approach

Scrum-style agile tracking was used with clearly defined layers:

	A	B	C	D	E	F	G	H
1	Milestone	Epic	Tasks	Sub-tasks	Sizing (*S = 1 day, M = 2-3 day, L = 1 Week, XL = 2-3 Week*)	Status	ECD	Owner
2	Data layer development	Scoping	Identify the exhaustive metric list	Identify list of metrics that are currently reported in the dashboard along with the granularity	L			
3			Identify data sources	Identify plan and actual data sources	S			
4			Onboard data sources to the data lake	Onboard data tables on the data lake (preferably not via data lake) but by Andes or data share	M			
5		Building	Build dimension tables	Date dimension	S			
6				Site list dimension	S			
7				Shift code dimension (if required)	S			
8				Other dimension tables as required	M			
9			Build staging tables	Build staging tables per the source (not required for ALPS)	L			
10			Build marts 1 planned tables	Daily, Station, Cycle, Activity, Sub-activity, EC, SC, TL	M			
11				Daily, Station, Cycle, Activity, EC, SC, TL	M			
12				Daily, Station, Cycle, Activity, EC, SC	M			
13				Daily, Station, Cycle, Activity, Sub-activity	M			
14				Daily, Station, Cycle, Activity	M			
15				Daily, Station, Cycle	M			
16				Daily, Station	M			
17				Weekly, Station, Cycle, Activity, Sub-activity, EC, SC, TL	S			
18				Weekly, Station, Cycle, Activity, EC, SC, TL	S			
19				Weekly, Station, Cycle, Activity, EC, SC	S			
20				Weekly, Station, Cycle, Activity, Sub-activity	S			
21				Weekly, Station, Cycle, Activity	S			
22				Weekly, Station, Cycle	S			
23				Weekly, Station	S			
24			Build marts 1 actual tables	Daily, Station, Cycle, Activity, Sub-activity	M			
25				Daily, Station, Cycle, Activity, EC, SC, TL	M			
26				Daily, Station, Cycle, Activity, EC, SC	M			
27				Daily, Station, Cycle, Activity	M			
28				Daily, Station, Cycle	M			
29				Daily, Station	M			
30				Weekly, Station, Cycle, Activity, Sub-activity	S			
31				Weekly, Station, Cycle, Activity, EC, SC, TL	S			
32				Weekly, Station, Cycle, Activity, EC, SC	S			
33				Weekly, Station, Cycle, Activity	S			
34				Weekly, Station, Cycle	S			
35				Weekly, Station	S			
36	Adoption	Replace data sources for existing dashboards	ALPS	Code change	M			
37				Validation	S			
38			G.A.P.	Code change	M			
39				Validation	S			
40			Follow with other dashboards	Code change	M			
41				Validation	S			
42	Deprecation	Deprecate old tables	Internal		M			
43			External		M			

5. Key Impact

- Ensured all dashboards maintained a consistent 15-minute refresh cadence.
- Enabled 25+ downstream users to access unified, reliable metrics, reducing data-related churn by **80%**.
- Allowed BI team to deprecate **8 redundant dashboards**, simplifying reporting surface.
- Optimized ETL and query logic, resulting in **30% reduction** in Athena query costs.