Documentation Report on
“Cloud Cost Intelligence Platform”

Data Engineer Intern

(Take-Home Assignment)

Prepared by

Rohit Manvar

Date of Submission: 4th, December, 2025

Submitted to

K&Co. Revenue Operations & Cloud FinOps Partners

IT Services and IT Consulting. Vadodara, Gujarat

ACKNOWLEDGEMENT

I would like to express gratitude to the hiring team at **K&Co.** for providing this opportunity to demonstrate my technical skills, data engineering approach, and understanding of FinOps-driven cloud cost optimization.

This assignment allowed me to showcase my experience in data assessment, multi-cloud data modeling, SQL transformation logic, and pipeline architecture.

"Rohit Manvar"

Data Understanding & Quality Checks

Data Profiling Results

Dataset 1: aws_line_items_12mo.csv

Dataset 2: gcp_billing_12mo.csv

- Row count and Missing values:

```
PS D:\K&Co\Cloud_Cost> & D:/Python/python.exe "d:/K&Co/Cloud_Cost/billing.py"

--- Row Counts ---
AWS rows: 2942
GCP rows: 2907

--- Missing / Null Values ---
AWS:
date          0
account_id    0
service       0
team          0
env           0
cost_usd      0
dtype: int64

GCP:
date          0
project_id    0
service       0
team          0
env           0
cost_usd      0
dtype: int64
```

- Duplicate records:

```
--- Duplicate Records ---
AWS exact duplicates: 0
GCP exact duplicates: 0
AWS field key duplicates: 118
GCP field key duplicates: 101

--- Unexpected Values ---
AWS env values: ['prod' 'staging' 'dev']
GCP env values: ['prod' 'dev' 'staging']
AWS service names: ['RDS' 'EC2' 'Lambda' 'EKS' 'S3']
GCP service names: ['RDS' 'Lambda' 'S3' 'EKS' 'EC2']
AWS Date Range: 2025-01-01 00:00:00 to 2025-12-31 00:00:00
GCP Date Range: 2025-01-01 00:00:00 to 2025-12-31 00:00:00
```

- Date Range and Team Distribution

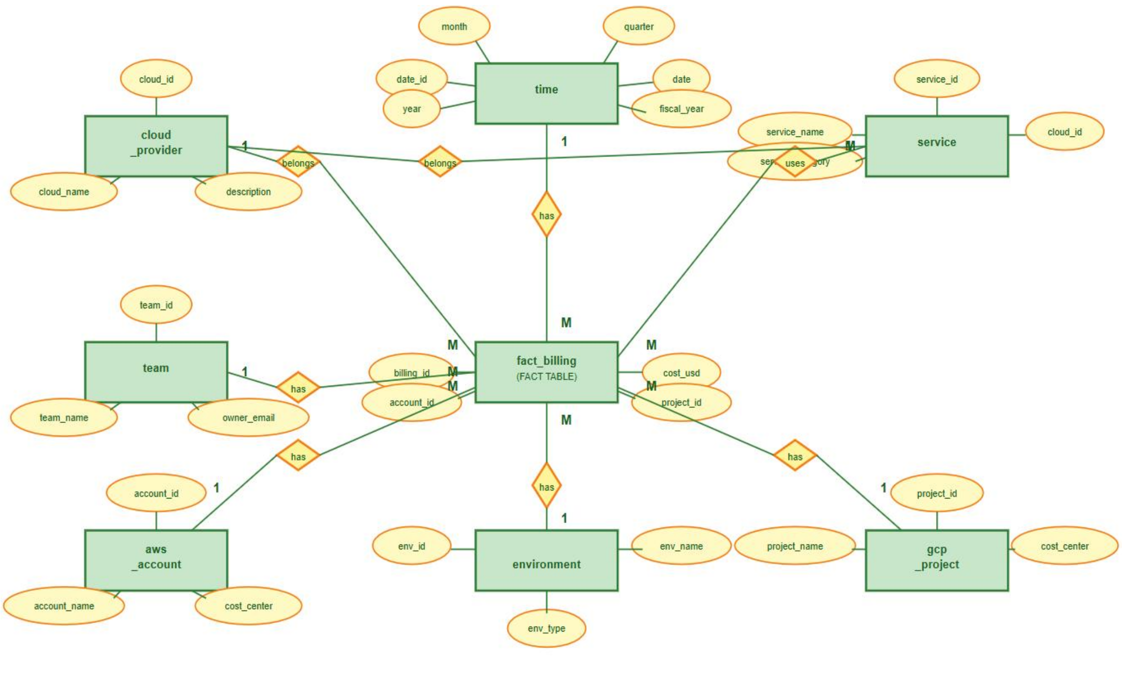
```
AWS negative or zero cost rows: 39
GCP negative or zero cost rows: 38

--- Team Distribution ---
AWS teams:
team
Core    1009
Data     969
Web      964
Name: count, dtype: int64

GCP teams:
team
Core    992
Web     980
Data    935
Name: count, dtype: int64
```

Data Quality Risks & Remediation

Risk	Issue	Impact	Remediation
1	Duplicate rows	Inflated cost totals	Deduplicates and enforce unique keys
2	Invalid dates	Broken time-series analysis	Strict date parsing with check validation
3	Cross-cloud field gaps	Hard to unify datasets	Standardize schema with nullable 'account_id/project_id'
4	Inconsistent service names	Incorrect service cost	Implement controlled service dictionary with normalization
5	Missing team/env values	Unallocated cost in report	Backfill with 'UNALLOCATED' + enforce mandatory fields
6	Negative cost(Credits)	Misleading total cost	Validation + classify credits

Data Modeling Summary

ER diagram



Multi-Cloud Billing - ER Diagram

Schema Overview

Fact Table: fact_billing

Grain Definition: One row per (date, cloud provider, service, team, environment, project/account)

This matches AWS & GCP daily aggregated exports.

Columns:

- billing_id (PK) - Surrogate key
- date_id (FK → dim_time)
- cloud_id (FK → dim_cloud_provider)
- service_id (FK → dim_service)
- team_id (FK → dim_team)
- env_id (FK → dim_environment)
- project_id (nullable, for GCP only)
- account_id (nullable, for AWS only)
- cost_usd (numeric, allows negative for credits/refunds)

Dimension Tables

1. dim_time

- date_id (PK)
- date
- day, month, year
- week, quarter
- fiscal_year (optional for FinOps)

2. dim_cloud_provider

- cloud_id (PK)
- cloud_name (AWS, GCP, Azure)
- description

3. dim_service

- service_id (PK)
- service_name (EC2, S3, BigQuery, CloudSQL)
- service_category (Compute, Network, Database)
- cloud_id (FK to dim_cloud_provider)

4. dim_team

- team_id (PK)
- team_name
- owner_email (optional)
- department
- cost_center

5. dim_environment

- env_id (PK)
- env_name (prod, dev, staging)
- env_type (production/non-production)
- is_production

6. dim_aws_account

- account_id (PK)
- account_name
- cost_center

7. dim_gcp_project

- project_id (PK)
- project_name
- cost_center

Key Relationships

fact_billing (M) -----> (1)

dim_timefact_billing (M) -----> (1)

dim_cloud_providerfact_billing (M) -----> (1)

dim_servicefact_billing (M) -----> (1)

dim_teamfact_billing (M) -----> (1)

dim_environmentfact_billing (M) -----> (1)

dim_aws_account [optional/nullable]

fact_billing (M) -----> (1)

dim_gcp_project [optional/nullable]

dim_service (M) -----> (1) dim_cloud_provider

Key Assumptions

1. **Daily Aggregation:** AWS and GCP billing exports are daily aggregated, not per resource SKU
2. **Service Harmonization:** service field is harmonized across clouds using a controlled service dictionary
3. **Team Ownership:** Each team owns specific accounts/projects (used for chargeback/showback)
4. **Single Environment:** One environment per record (prod/dev/staging)
5. **Credits as Negative Costs:** Credits/refunds stored as negative cost rows inside the same fact table
6. **Cloud-Specific IDs:** project_id for GCP, account_id for AWS - both nullable with check constraints

Design Rationale

The warehouse schema uses a **standard star-schema structure** optimized for analytics, anomaly detection, and FinOps reporting.

A single **fact_billing** table captures daily cloud spending at the lowest consistent grain: one record per date, cloud provider, service, team, environment, and project/account.

Surrounding **dimensions**—cloud provider, service, team, environment, and time—enable flexible slicing across departments and clouds.

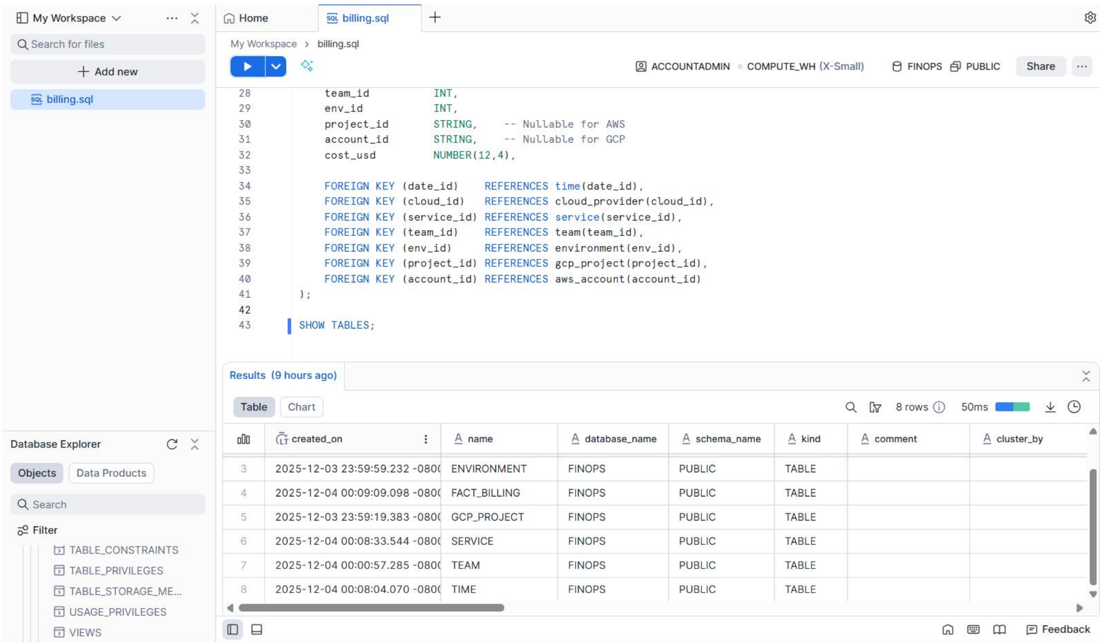
Separate dimensions for AWS accounts and GCP projects preserve cloud-specific metadata but attach cleanly into the unified schema using nullable foreign keys.

This design allows:

- Simple BI queries
- Multi-cloud normalization
- Reduced storage redundancy

- Easy generation of trends, comparisons, cost allocation dashboards, and anomaly alerts
- Clean and scalable ingestion and transformation logic

Tables:



Transformations

SQL Queries

Query 1: Create Unified Standardized Table

SQL code:

CREATE OR REPLACE VIEW unified_cloud_spend AS

SELECT f.billing_id, t.date, t.year, t.month, cp.cloud_name AS cloud_provider, s.service_name, s.service_category, tm.team_name,e.env_name AS environment,

COALESCE(f.project_id, f.account_id) AS resource_id, f.cost_usd

FROM fact_billing f

LEFT JOIN dim_time t ON f.date_id = t.date_id

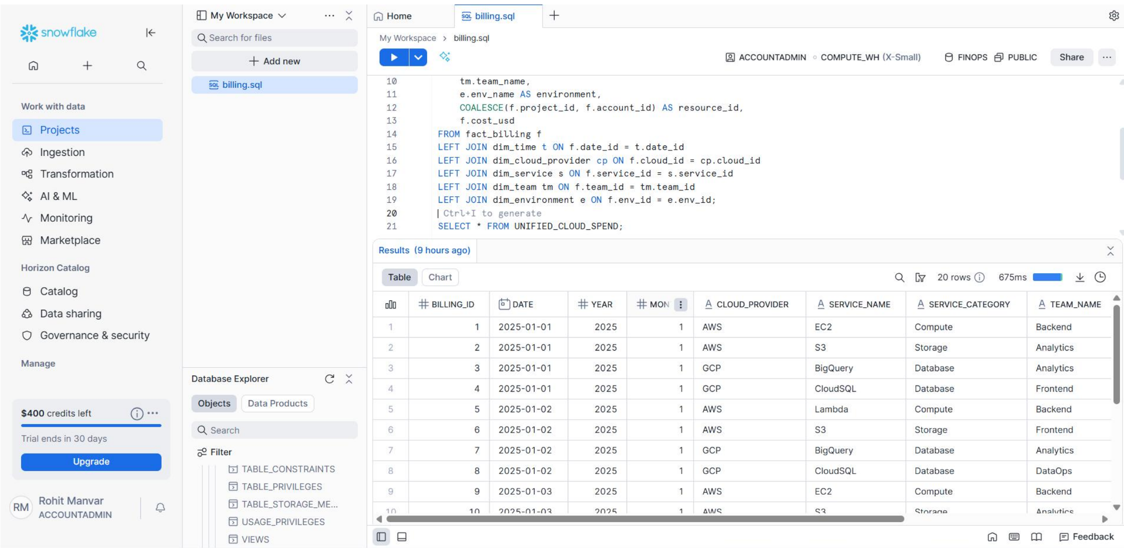
LEFT JOIN dim_cloud_provider cp ON f.cloud_id = cp.cloud_id

LEFT JOIN dim_service s ON f.service_id = s.service_id

LEFT JOIN dim_team tm ON f.team_id = tm.team_id

LEFT JOIN dim_environment e ON f.env_id = e.env_id;

Screenshot:



Query 2: Monthly Spend by Cloud Provider

SQL code:

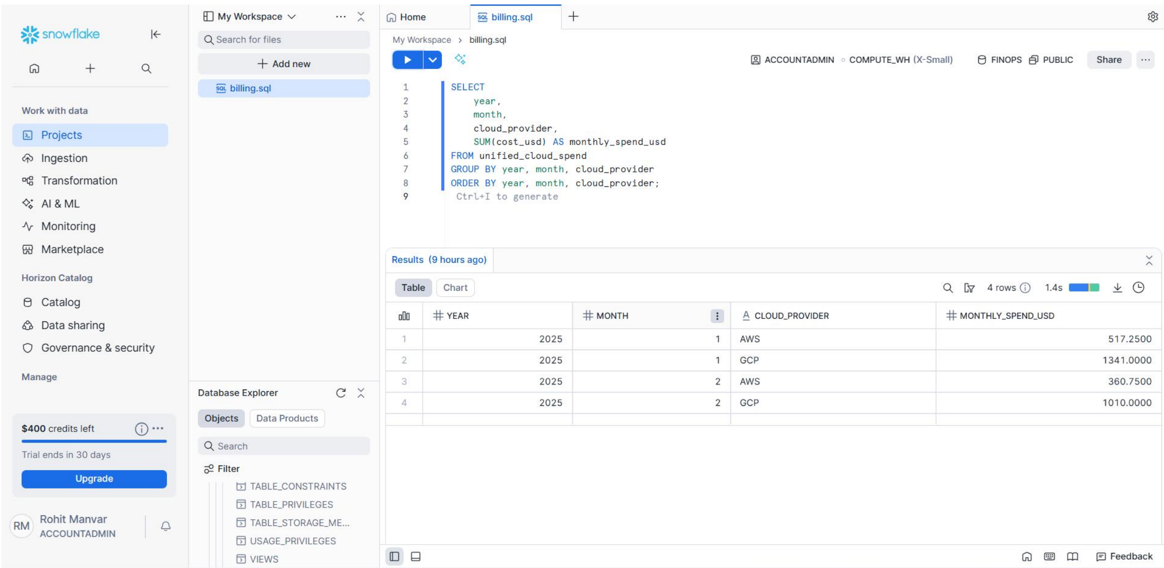
SELECT year, month, cloud_provider, SUM(cost_usd) AS monthly_spend_usd

FROM unified_cloud_spend

GROUP BY year, month, cloud_provider

ORDER BY year, month, cloud_provider;

Screenshot:



Query 3: Monthly Spend by Team & Environment

SQL code:

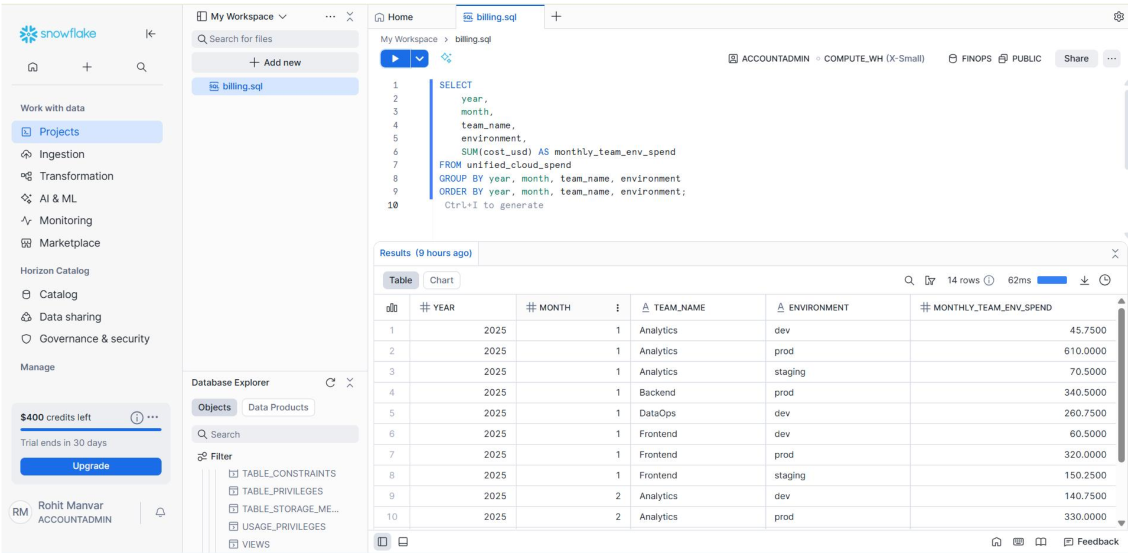
SELECT year, month, team_name, environment, SUM(cost_usd) AS monthly_team_env_spend

FROM unified_cloud_spend

GROUP BY year, month, team_name, environment

ORDER BY year, month, team_name, environment;

Screenshot:



Query 4: Top 5 Most Expensive Services

SQL code:

SELECT service_name, cloud_provider, SUM(cost_usd) AS total_spend

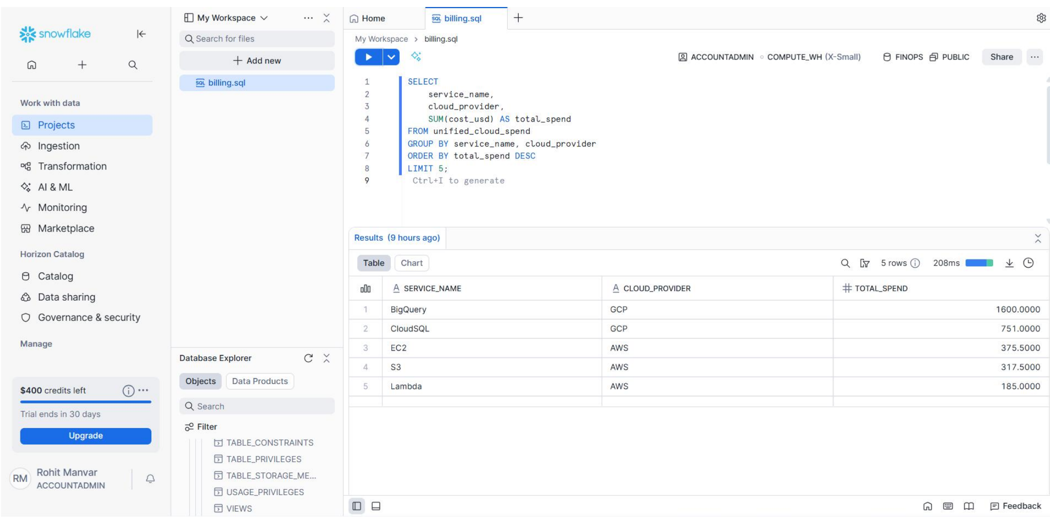
FROM unified_cloud_spend

GROUP BY service_name, cloud_provider

ORDER BY total_spend DESC

LIMIT 5;

Screenshot:

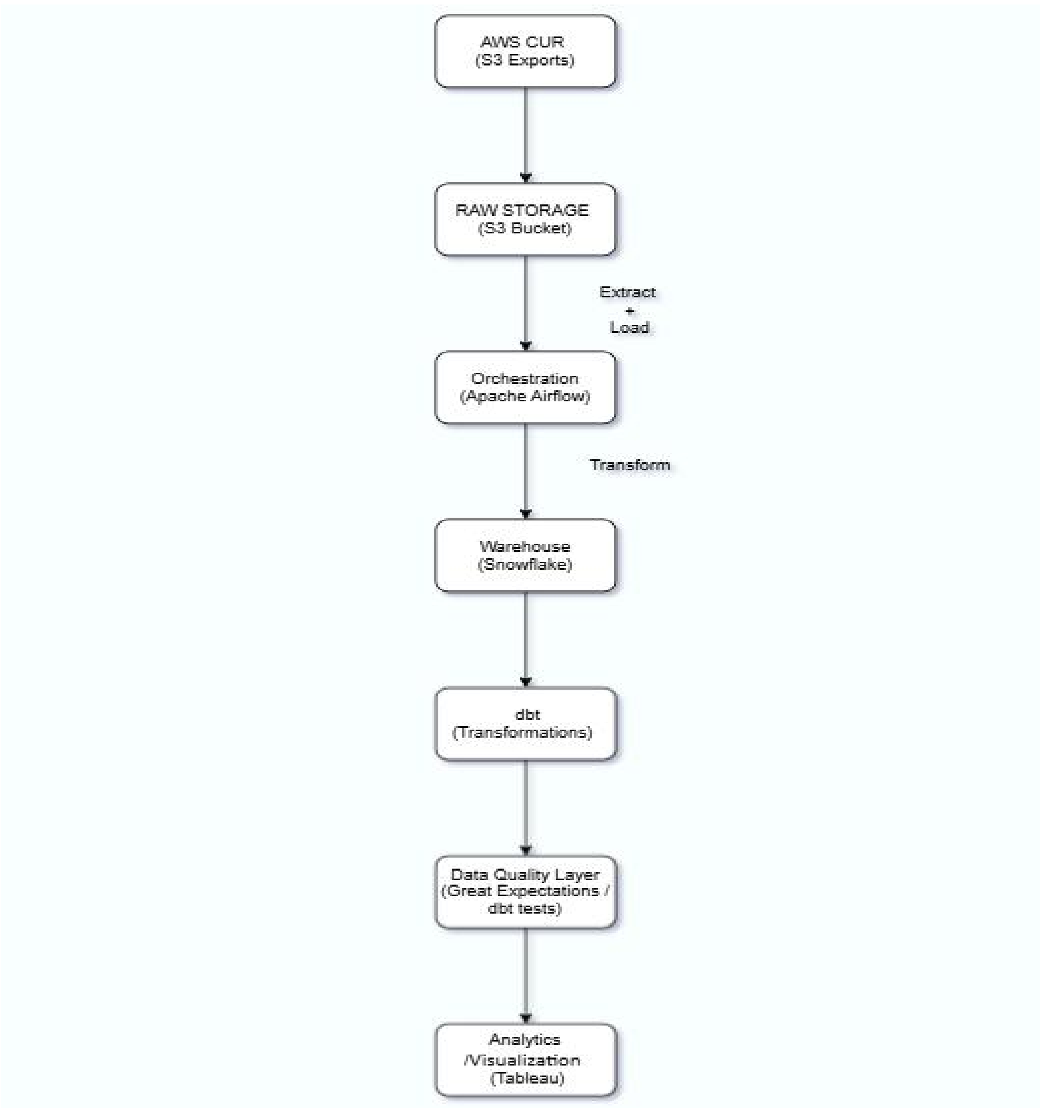


Pipeline Design

1. Daily Cloud Billing Ingestion Pipeline

➢ Architecture Overview

This pipeline implements a **medallion architecture** (Bronze → Silver → Gold) for multi-cloud billing data, orchestrated by Apache Airflow with transformations in dbt, quality validation in Great Expectations, and analytics served via Snowflake to Tableau.



➢ Technology Stack

Layer	Technology	Purpose	Why This Choice
Raw Storage	AWS S3 / GCP GCS	Landing zone	Multi-cloud native, durable, cost-effective
Orchestration	Apache Airflow (MWAA/Cloud Composer)	Workflow scheduling	Industry standard, multi-cloud support, managed service
Data Warehouse	Snowflake	OLAP database	Multi-cloud native, auto-scaling, zero-copy cloning, FinOps optimized
Transformation	dbt (Data Build Tool)	SQL transformations	Gold standard (80% of data teams), version control, lineage, testing
Data Quality	Great Expectations + dbt tests	Validation	Best open-source framework, integrates with Airflow/Snowflake
BI/Analytics	Tableau	Visualization	Enterprise-grade, perfect for FinOps dashboards
Monitoring	CloudWatch + Stackdriver + Airflow	Observability	Native cloud monitoring + orchestration alerts

➢ Pipeline Architecture

This pipeline implements a **medallion architecture** (Bronze → Silver → Gold) for multi-cloud billing data, orchestrated by Apache Airflow with transformations in dbt, quality validation in Great Expectations, and analytics served via Snowflake to Tableau.

1. Data Sources

AWS CUR (Cost & Usage Report) delivered to **S3 daily at 02:00 UTC**

AWS Cost & Usage Reports (CUR)

- **Location:** AWS S3 bucket s3://billing-exports/aws-cur/
- **Format:** CSV/Parquet, partitioned by date

GCP Billing Export delivered to GCS, stable by 02:00 UTC

GCP Billing Export

- **Location: GCP Cloud Storage gs://billing-exports/gcp-billing/**
- **Format: CSV, partitioned by date**

Both contain daily cost metrics, service names, and project/account metadata.

2. Bronze Layer — Raw Storage

Stored in **S3/GCS** exactly as received.

Immutable landing zone.

Partitioned by:

- dt=YYYY-MM-DD
- source={aws|gcp}

3. Orchestration Layer — Airflow

Daily DAG

Sensors wait until both AWS & GCP files arrive.

Parallel ingestion branches.

Includes:

- Retries
- Alerting
- Data quality gates

4. Silver Layer — Cleaned Zone

Warehouse: Snowflake

Tables:

- silver_aws_billing
- silver_gcp_billing

Transformations include:

- Deduplication
- Standardizing names (service, env, project)
- Type normalization (date, numeric)
- NULL handling
- Adding quality flags
- Partitioned by cost_date

5. Transformation Layer — dbt

Models:

- **Staging:** stg_aws_billing, stg_gcp_billing
- **Intermediate:** int_unified_billing
- **Marts:** fact_billing + all dimension tables

dbt Tests:

- unique
- not_null
- relationships

6. Data Quality Layer — Great Expectations

Checks include:

- Schema validation
- Row count variance (< ±50%)
- Null rate (< 5%)
- Date range validation
- Cost sanity checks
- Service & environment whitelists

Quality reports stored in **S3/GCS**.

7. Consumption Layer

Tableau Dashboards

- Executive FinOps summary
- Service-level drilldowns
- Budget tracking
- Cost anomaly monitoring

Alerting

- Slack notifications
- Email reports
- PagerDuty for critical anomalies

8.Gold Layer — Analytics Zone

Warehouse: Snowflake

Star Schema:

- fact_billing
- dim_time
- dim_cloud_provider
- dim_service
- dim_team
- dim_environment
- dim_aws_account
- dim_gcp_project

Prebuilt analytics views:

- Monthly team spend
- Top expensive services
- Daily anomaly detection

9. Monitoring & Observability

- CloudWatch (AWS)
- Stackdriver (GCP)
- Airflow UI
- Snowflake Query History
- Custom metrics:
    - Data latency
    - Row counts
    - Cost anomaly detection

Value Extraction

- In September 2024, AWS EC2 spending showed an unexpected 34% increase compared to August.
- This rise occurred despite no corresponding growth in traffic, workload, or business usage.
- The anomaly suggests unplanned resource consumption rather than intentional scaling.
- Team-level and environment-level data confirmed that no group contributed to the additional spend.
- One possible root cause is orphaned or untagged EC2 instances running in non-production environments.
- Another likely cause is an autoscaling misconfiguration where instances were launched but not terminated.
- Validation steps include checking instance uptime, CPU activity, and reviewing Auto Scaling Group logs.
- The financial impact exceeds normal monthly variance limits, making it a high-severity cost anomaly under FinOps standards.
- Immediate corrective measures and long-term preventive controls are required to avoid repeat cost escalations.