Scale and Security : How Autodesk Securely Develops and Tests PII Pipelines with Airflow

Bhavesh Jaisinghani

AUTODESK

Introduction

Name

Bhavesh Jaisinghani

Role

Data Engineering Manager

Based in

San Francisco Bay Area




Agenda

1. Data Tech Stack
2. Background, Challenges and Impact (Why)
3. Testing Strategies
4. Ideal Environment (What)
5. Laying the foundation (How)
6. Development flow
7. Metrics and Outcomes
8. Q&A

Tech Stack : Keeping our Jenga Tower Steady

Data Tech Stack

Ingest

Fivetran

MATILLION

amazon Kinesis

Transform

Apache Spark

dbt

amazon EMR

Storage

Warehouse

snowflake

Lake

HIVE

Amazon S3

BI / Reporting

Looker

Power BI

Catalog

atlan

Orchestration

ASTRONOMER

Apache Airflow

Workspace

POPSQL

jupyterhub

Query Engine

trino

Monitoring

pepperdata

Databand

Rules of the Game :

Assumptions and Guardrails




Background

- Product Data Engineering Team
  - Focuses on data transformation to BI layer
  - Evolving business rules
  - Scale & optimize spark scripts for growing data volumes
  - High Emphasis on data accuracy

- Dedicated Data Platform team
  - Implemented Multi-Tenancy Model
  - Dev, Stg and Prod environment setup on different AWS Accounts

Environment Challenges

Security Concerns on Data copy / mounting to non prod environments

Data copy process requires multiple approvals for PII and sensitive data

Limited AWS access to Data Engineering Personas

Restrictions on local data copy for development and testing due to its sensitivity

Development Impact

Long development and testing cycles

Fragmented developer experience

Rise of Cowboy Coders

Trust, but Validate

ETL Testing Strategies

Code Testing

- Unit Testing
- Syntax Testing

Data Testing

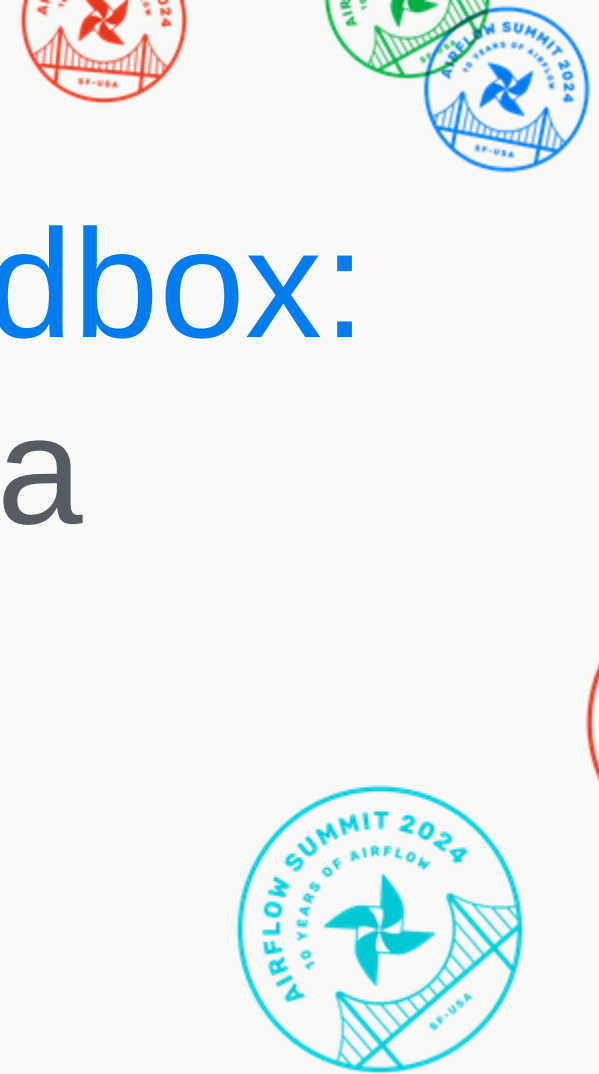
- Smoke Testing (DDL, Duplicate, Null, Count)
- Data Dependency Checks
- Functional Testing / Data Quality Checks
- Data Regression Testing

Pipeline Testing

- Workflow Testing
- ETL Performance / Data Load Testing

The Matrix Sandbox:

Where your data dodges bullet



Ideal UAT Environment (What)

- Production-Like Environment
- Helps with historical loads (minimizes production overload)
- Supports regression, pipeline, performance, and stress testing
- No impact on production tables
- Supports multiple developers working on a monorepo

Laying the foundation (How)

- Database
  - Mirror production Hive schemas for UAT

- Infrastructure
  - Separate EMR clusters for UAT
  - Astronomer deployments for each developer (Airflow UI servers)

- CI/CD and Developer Utilities
  - Common functions to sync tables and data from production to newly created schemas
  - Standard CI pipeline that can deploy on any deployment node for any engineer
  - Auto-load UAT environment variables on deployment
  - Framework for regression testing of tables in UAT and PROD

Development Flow

Before UAT

1. Develop and Test locally
2. Validate data logic with functional test results
3. Promote to release and deploy to Production
4. **Apply hotfix for data quality issues or address spark performance issues**
5. Redeploy

After UAT

1. Develop and Test locally
2. **Sync required tables from Production to UAT**
3. **Deploy feature branch on UAT Airflow node**
4. **Perform stress test and validate end to end outcomes**
5. Promote to release and deploy to Production

Endgame, Data Assemble






Metrics and Outcome of UAT environment

Improved product discovery and Supports experimentation of new data processing logic.

Helped identify performance bottlenecks and optimize spark jobs. Average spark job optimization **tech debt reduced by 60%.**

Improved Error Identification and data quality of our data outputs by **90%**

Code deployed to production **33% faster**.

Improved testing of infrastructure upgrades (Eg: Spark, EMR version upgrades, package management)

Feedback from Astronomer

Recommendations

Continue

UAT in Prod Development Pattern - The ability of teams to test code changes against production data safely is somewhat of a "holy grail" in data ops practices. This group has shown its value and appears to have the capabilities required to implement and maintain it as a practice. It is a phenomenal capability for a mature QA practice - especially as pipelines evolve to handle edge cases, increase performance, and add capabilities in the future.

Questions?

Find me on Linkedin

https://www.linkedin.com/in/bjaisinghani