AIRFLOW SUMMIT | ONLINE RECONNECT

Orchestrating the AI Revolution



# Configuration-Driven Data Pipelines

## Amazon MWAA with dbt and Astronomer Cosmos


Suba Palanisamy
Sr. Technical Account Manager
Amazon Web Services (AWS)

Sushmita Barthakur
Sr. Data Solutions Architect
Amazon Web Services (AWS)

Sneha Rao
Sr. Solutions Architect
Amazon Web Services (AWS)

# Agenda




- ETL Complexities
- Building Blocks: Amazon MWAA, dbt, Cosmos and OpenLineage
- Use-case for config-driven data pipeline on AWS
- Solutions Architecture with End-to-End flow
- Demo
- Operational Excellence

# Challenges with ETL at scale

As pipelines grow, data modeling teams slow down and struggle to keep data quality high because of

**DAG sprawl**
Every new model means more DAG code, wired by hand

**Late quality checks**
Bad data is found after it reaches a dashboard

**No traceability**
Hard to tell where a table's data came from

**Slow onboarding**
Analysts wait on engineers to ship new models

## Building Blocks

**Amazon Managed Workflows for Apache Airflow**



**dbt**



**Cosmos**

ASTRONOMER

**OpenLineage + Amazon SageMaker Unified Studio**



# Building blocks: Amazon MWAA

## Amazon MWAA



- AWS-managed Apache Airflow service.
- Easy Airflow deployment and AWS-managed security updates.
- Multi-AZ deployment and native AWS integration.
- Autoscaling prevents performance bottlenecks.
- Best for complex DAGs and consistently high-volume workloads.

## Key Features

### Config-driven DAG factory

One YAML file builds every pipeline's DAG, so a new pipeline means adding a YAML block, not writing a new DAG.

### Asset-Aware Scheduling

Pipelines trigger when their upstream input data is ready, and not on a fixed schedule.

# Building blocks: dbt

## dbt (data build tool)



- Open-source framework for transforming data that's already in a data lake or warehouse.
- Each transformation is a SQL SELECT statement, called a model.
- Builds models in dependency order and runs data-quality tests.
- Auto creates documentation and a dependency graph.
- Doesn't load data or schedule runs, so an orchestrator like Airflow handles scheduling.

## Key Features

**Models & ref()**
(the dependency graph)

**Tests with severity**

**Tags**

**Manifest**

# Building blocks: Cosmos

ASTRONOMER

## Cosmos (astronomer-cosmos)

- Open-source Python library from Astronomer for running dbt Core projects in Airflow
- DbtDag: turns a dbt project, or part of it, into an Airflow DAG
- DbtTaskGroup: places dbt models inside a larger DAG, next to other tasks
- No hand-written task wiring; the dbt project is the source of truth

## Key Features

**Per-model tasks and tests**

**Flexible execution modes**

**Data-aware scheduling**

**Model Selection**

# Building blocks: OpenLineage with Amazon SageMaker Unified Studio

## OpenLineage

- Open platform for collection and analysis of data lineage.
- Tracks metadata about datasets, jobs, and runs, enabling users to identify root cause of complex issues and understand impact of changes.
- Open Standard for lineage metadata collection designed to record metadata for a job in execution.
- As an extension of the OpenLineage project, a custom transport is available to send lineage events directly to Amazon SageMaker Unified Studio's endpoint.

## Amazon SageMaker Unified Studio



- Single data and AI development environment to find and access data in an organization and act on it using the best tools across any use case.
- Admins/ Users can configure lineage in projects while setting up connections for data sources to enable lineage capture from data source runs.

### OpenLineage-compatible Features

| | | | |
|---|---|---|---|
| Open event standard | Extensible facets | Visual lineage graph | OpenLineage ingestion |

# Use-Case: Config-driven data pipelines on AWS

## Use-Case

**A retail company's data pipeline includes -**

- **Raw "Orders" data files**
  Lands in object storage every day
- **Operations team**
  Cleans them into orders, sales and revenue tables
- **Marketing team**
  Builds customer segments, but only once fresh order data is ready

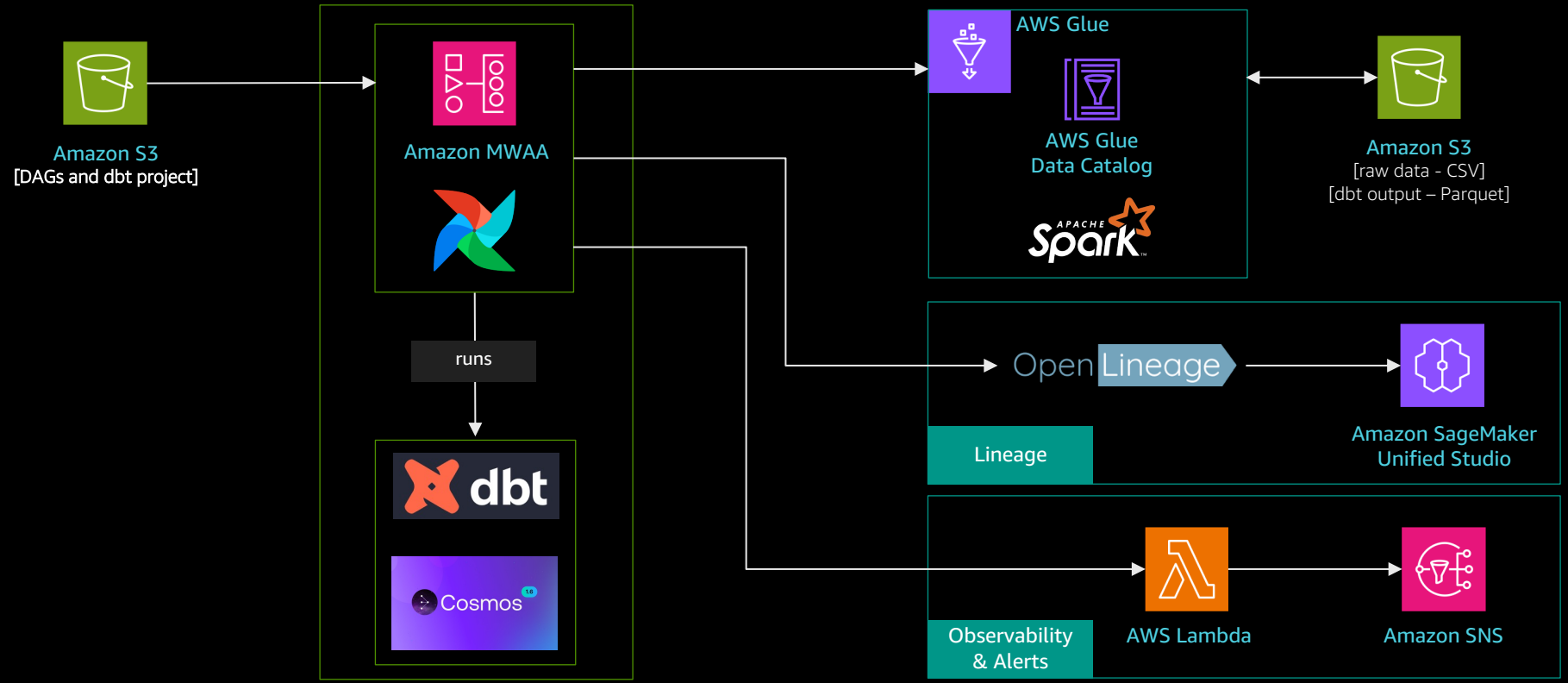
## Challenges

- **Slow to scale**
- **Late quality checks**
- **Lack of traceability**

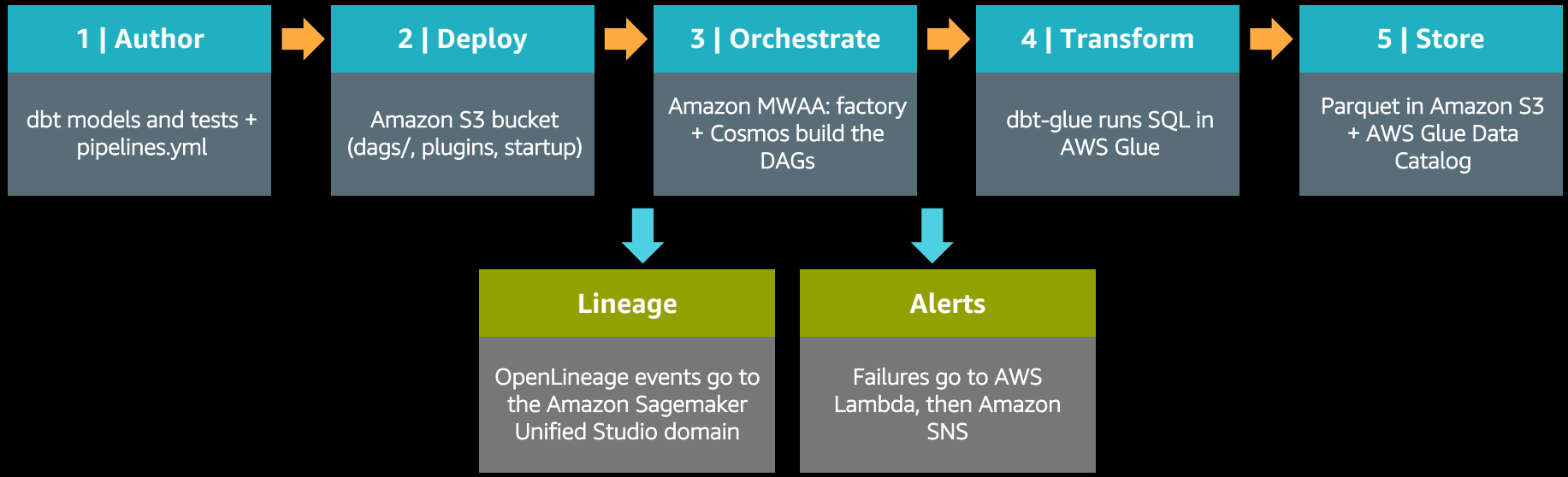
## Solution

| dbt on AWS Glue | Cosmos | Amazon MWAA | OpenLineage |
|---|---|---|---|
| Write transformations in SQL with dbt; process serverlessly on Spark using AWS Glue | Turns the dbt project into Airflow tasks, so a new pipeline is a few lines of YAML | Runs and schedules pipeline; executes on a schedule, or when their input data is ready | Sends lineage to Amazon SageMaker Unified Studio to trace every table back to source |

# Solution Architecture



# End-to-end Data Pipeline flow




**A new model file or a new YAML block reaches production as tasks, tests, lineage and alerts, with no DAG code written**

# Workflow Deep Dive

Team :Operations
Dag : sample_orders_dag
Schedule : @daily

→ **publishes customer_orders** →

Team : Marketing
Dag : customer_insights_dag

Input : a tagged SQL file and a YAML block. No Python.

**models/staging/stg_orders.sql**

```
{{ config(materialized='table',
          tags=['operations']) }}

with typed as (
  select cast(order_id as int) as order_id,
         customer, amount, status, order_date
  from {{ source('raw', 'raw_orders') }}
),
deduplicated as (
  select *, row_number() over (
      partition by order_id
      order by order_date desc) as _row_num
  from typed
)
select order_id, customer, amount, status, order_date
from deduplicated where _row_num = 1
```

**dags/config/pipelines.yml**

```
- name: sample_orders
  owner: data-platform
  schedule: "@daily"
  stages:
    - name: audit_dbt_task
      select: ["tag:audit"]
    - name: transform_data
      select: ["tag:operations"]
  runtime_vars:
    max_daily_revenue: 1000000
  produces:
    - .../dbt_demo/customer_orders
    - .../dbt_demo/daily_sales
    - .../dbt_demo/revenue_by_region
```

# How the YAML becomes a DAG



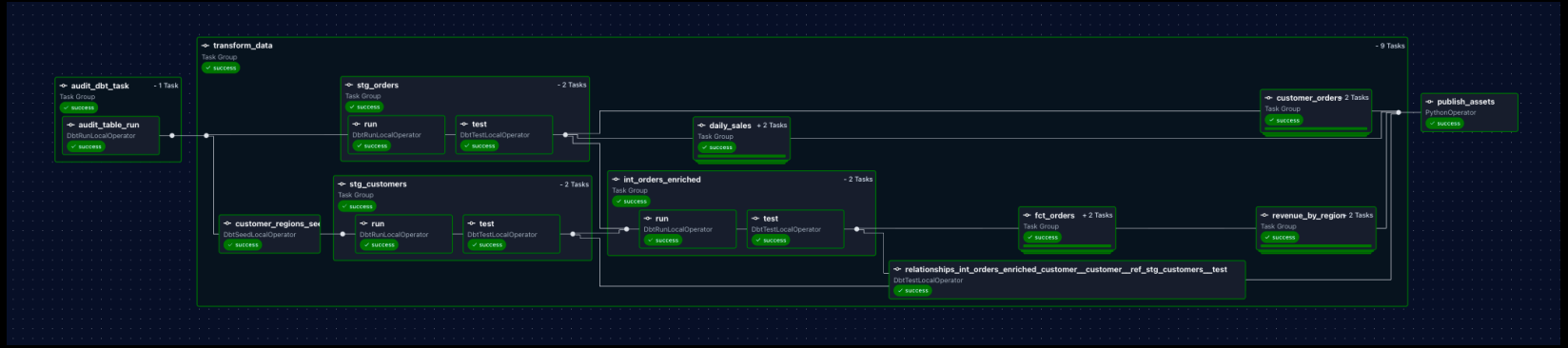

Two inputs meet in Cosmos: the dbt manifest says **what exists**, the YAML says **what to run and when**

**YAML key → what it becomes in the DAG**

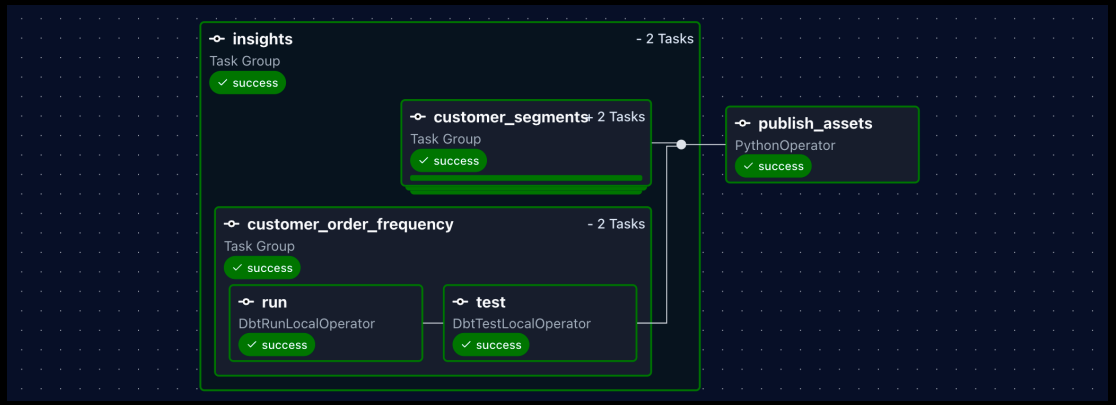
| schedule | stages.select | on_test_failure | alerting | produces |
|---|---|---|---|---|
| @daily → DAG schedule | tag:operations → task group of every tagged model | block → failed test stops downstream | → Lambda callbacks, one route per alert kind | → publish_assets, starts downstream DAGs |

# The generated DAGs

**Operations Team - sample_orders_dag**




**Marketing Team -** customer_insights_dag



**New AWS Console dualstack experience available**
AWS Console is now available internally on a new dualstack console.aws.com domain. If you already installed Tampermonkey script for testing IPv6, use this link to switch the domain. Otherwise, install the script first following these instructions. Please notice that only accounts attached to internal Service Principal can be used to access dualstack experience.

Analytics

# Amazon SageMaker
## The center for data, analytics, and AI

The next generation of Amazon SageMaker delivers an integrated experience for analytics and AI with unified access to all of your data. Now with a new serverless notebook with a built-in AI Agent.

Amazon DataZone is now part of Amazon SageMaker. You can continue to use your existing domains using the links on this page.

### Get started with Amazon SageMaker Unified Studio

Analyze and build with data in this AWS account using IAM roles and permissions

Open

View existing domains

### Resources

What is Amazon SageMaker Unified Studio?

Getting started with Amazon SageMaker Unified Studio

Documentation

API documentation

Demo videos

FAQ

### Unified Studio

Build with tools from AWS Analytics and AI/ML services in a single data and AI development environment. This includes a new serverless notebook with a built-in AI Agent.

### Artificial Intelligence & Machine Learning

Build, train, and deploy ML models, including foundation models, for any use case with fully managed infrastructure, tools, and workflows.

### Governance

Securely discover, govern, and collaborate on data and AI. Bring in data in minutes, using existing AWS Identity and Access Management (IAM) roles and permissions.

### Lakehouse Architecture

Unify data across Amazon S3 data lakes, Amazon Redshift data warehouses, and third-party and federated data sources.


© 2026, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
2026-11-04 09:09:16

# Operational Excellence

AIRFLOW SUMMIT | ONLINE RECONNECT

## Lineage & Observability

## Testing Patterns

## Scaling

- Scaling means adding models and pipelines without adding DAG code or breaking existing ones.
- Guardrails keep many teams from colliding.

## Migration

| 1 Legacy | 2 Shadow | 3 Cutover | 4 Retire |
|---|---|---|---|
| Hand-written DAG with BashOperator tasks runs the models | Same models in a separate database, alerts off, tests warn-only | Remove the overrides, pause the legacy DAG. Rollback: revert and unpause | Delete the legacy file after enough clean run |

AIRFLOW SUMMIT | ONLINE RECONNECT

# Thank You




Suba Palanisamy
Sr. Technical Account Manager
Amazon Web Services (AWS)

linkedin.com/in/subapalanisamy

Sushmita Barthakur
Sr. Data Solutions Architect
Amazon Web Services (AWS)

linkedin.com/in/sushmita-barthakur

Sneha Rao
Sr. Solutions Architect
Amazon Web Services (AWS)

linkedin.com/in/sneha-rao-1b4b1396