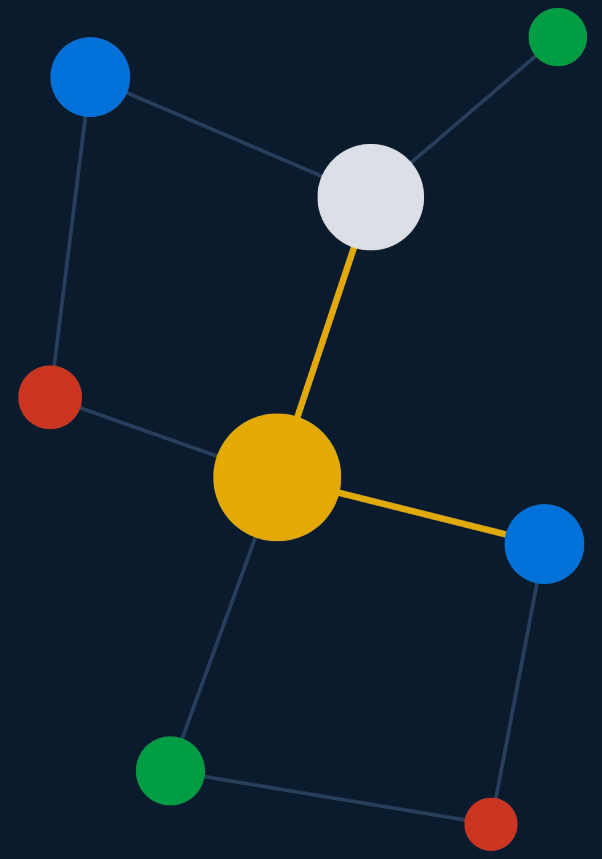
AIRFLOW SUMMIT ONLINE RECONNECT 2026

Ontology-First Pipelines

Preparing data for knowledge graphs and AI agents with Airflow 3



Cesar Mendez

Data engineering and AI · in/devcsar · csar.dev

ABOUT ME

Cesar Mendez

I design agentic AI architectures and build enterprise semantic layers.



Industries

Smart cities · pharma · fintech · automotive

Communities

Anthropic Builders Council · Claude Ambassador, Mexico City · Founder, AWS AI User Group Mexico

Since 2017

Building communities and advocating for open science

Find me

in/devcsar · csar.dev

THE PROBLEM

An agent knows the words. It doesn’t know what they mean here.

It has read most of the public internet. It has never read your CRM, your contracts, or the way your team says “active customer”.

ONE WORD, THREE MEANINGS

What does “customer” mean?

Sales · CRM

Anyone with a signed contract

Finance · ERP

An account that paid at least one invoice

Support · help desk

Anyone who has opened a ticket

Ask an agent “How many customers do we have?” and it quietly picks one of these, without telling you which.

FROM PHILOSOPHY

Ontology: an account of what exists

ὄντος
ontos · “of being”

+

λόγος
logos · “word, account, reason”

What exists, what kinds of things there are, and how they relate.

~350 BC
Aristotle’s Categories sorts everything that exists into kinds

1606
The Latin word “ontologia” appears in print

1993
Tom Gruber brings it to computing: “an explicit specification of a conceptualization”

2004
OWL becomes the W3C standard for writing ontologies

2012
Google launches its Knowledge Graph: “things, not strings”

2026
Ontologies become the context layer for AI agents

THE WORD HAS POWER

Grammar class was data modeling

Subject, verb, object. A thing, a relation, another thing.

FROM LITERATURE TO ENGINEERING

Six steps from a word to code

01

Word

A pointer to something in the world

Agent guesses

02

Dictionary

Each word gets a meaning

Agent can explain

03

Taxonomy

Things get a family: an Artist is a Person

Agent knows the types

04

Ontology

Rules: what can connect to what, what must be true

Agent refuses impossible questions

05

Knowledge graph

Facts with stable IDs that follow the rules

Agent follows the links

06

Code

Turtle, Cypher and pipelines that load and check it

Agent queries through tools

A dictionary is for people. An ontology is a dictionary with rules a machine can check.

SEEING THE GRAPH

How a graph is built

Nouns become nodes

Café Luna, Order 1003

Verbs become edges

PLACED, ABOUT, CONTAINS

Many names become one ID

customer:cafe_luna

Rules decide what gets in

a ticket with no order is held back

WORDS PEOPLE MIX UP

Rules, facts and math

Layer	Answers	Holds	Example
Taxonomy	What kind of thing is it?	A family tree of types	An Artist is a Person
Ontology	What can exist, and how can it connect?	Classes, relationships, rules	Every Ticket is ABOUT exactly one Order
Knowledge graph	What is true right now?	Facts with IDs and sources	Ticket T-77 is ABOUT Order 1003
Semantic layer	How do we measure it?	Metrics, dimensions, joins	Revenue is the sum of paid invoice lines

The ontology is the rules, the graph is the facts, the semantic layer is the math. Agents need all three.

WHY AGENTS NEED IT

A map instead of a pile of pages

Grounding

Every answer points to a node ID and a source, not to a paragraph that sounded right.

Multi-hop

Customer to ticket to order to product is a walk along edges, not a lucky text match.

Determinism

The agent writes a query you can read, rerun and audit. Same question, same answer.

Guardrails

The rules say what can’t exist, so the agent can say “this data can’t answer that” instead of guessing.

Vector search hands an agent pages that look similar. An ontology hands it the objects it can read, the links it can follow and the actions it’s allowed to take.

THE EVIDENCE

Same model, same questions, more context

16.7%

GPT-4 answering enterprise questions with SQL over the raw database schema

54.2%

The same questions asked over a knowledge graph built on an ontology

72%

Adding an ontology-based check that repairs bad queries, including 8% honest “I don’t know”

⚠ **Not magic.** On simple one-fact questions, plain vector RAG often matches or beats GraphRAG. The gain comes from a curated ontology and a pipeline that keeps it correct.

Sequeda, Allemang & Jacob 2023 · Allemang & Sequeda 2024 · Han et al. 2025 · GraphRAG-Bench 2025

AN INDUSTRY TREND

Everyone is building the same layer

Palantir

Ontology

An operational layer of objects, links and actions on top of the data. A digital twin of the organization.

`Core of Foundry and AIP`

Databricks

Genie Ontology

One context layer for Genie: Unity Catalog semantics plus context it infers from queries and dashboards.

`Public Preview`

Snowflake Labs

Ontology on Snowflake

Graph tables, ontology metadata, generated views and semantic models, used by a Cortex Agent.

`Open reference build`

NVIDIA

Auto-Ontology

Compiles database metadata into an ontology people certify, then serves it to agents through MCP.

`Open source, Apache-2.0`

And a shared standard: Apache Ossie (Incubating) is an open spec for exchanging semantic models and ontologies between tools.

palantir.com/docs · docs.databricks.com · github.com/Snowflake-Labs · github.com/NVIDIA

NOW THE ENGINEERING

The graph is the easy part.

The hard part is the data engineering that happens before it. That’s where Airflow becomes an actor.

ONTOLOGY-FIRST PIPELINE

Six stages before the graph

01

Ingest

Land each source as it is, timestamped and replayable

Prevents losing the original when a rule changes

02

Conform

Normalize names, accents, dates, units and IDs

Prevents duplicates that are only formatting

03

Extract

An LLM reads free text, limited to the ontology's types

Prevents invented types and relations

04

Resolve

Decide which records are the same real thing

Prevents three copies of every customer

05

Validate

Check every fact against the ontology's rules

Prevents publishing data that makes no sense

06

Publish

Idempotent MERGE with the source and run ID on every fact

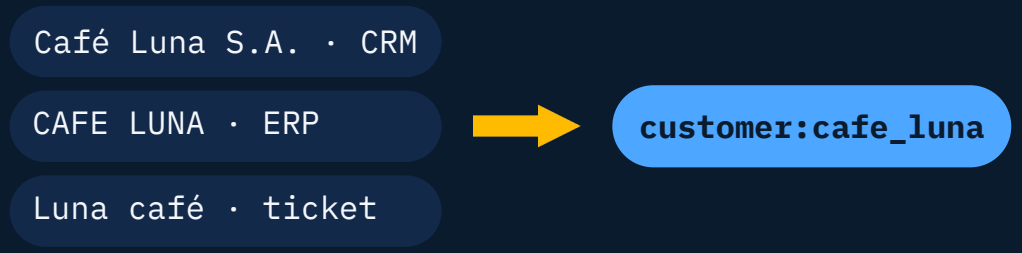
Prevents doubled graphs and untraceable answers

Agents only ever read what came out of stage six.

WHERE GRAPHS GO WRONG

The two steps most teams skip

Entity resolution



Match on hard IDs first, then normalized names, then embeddings with a threshold. Ambiguous pairs go to a person.

Validation as a gate

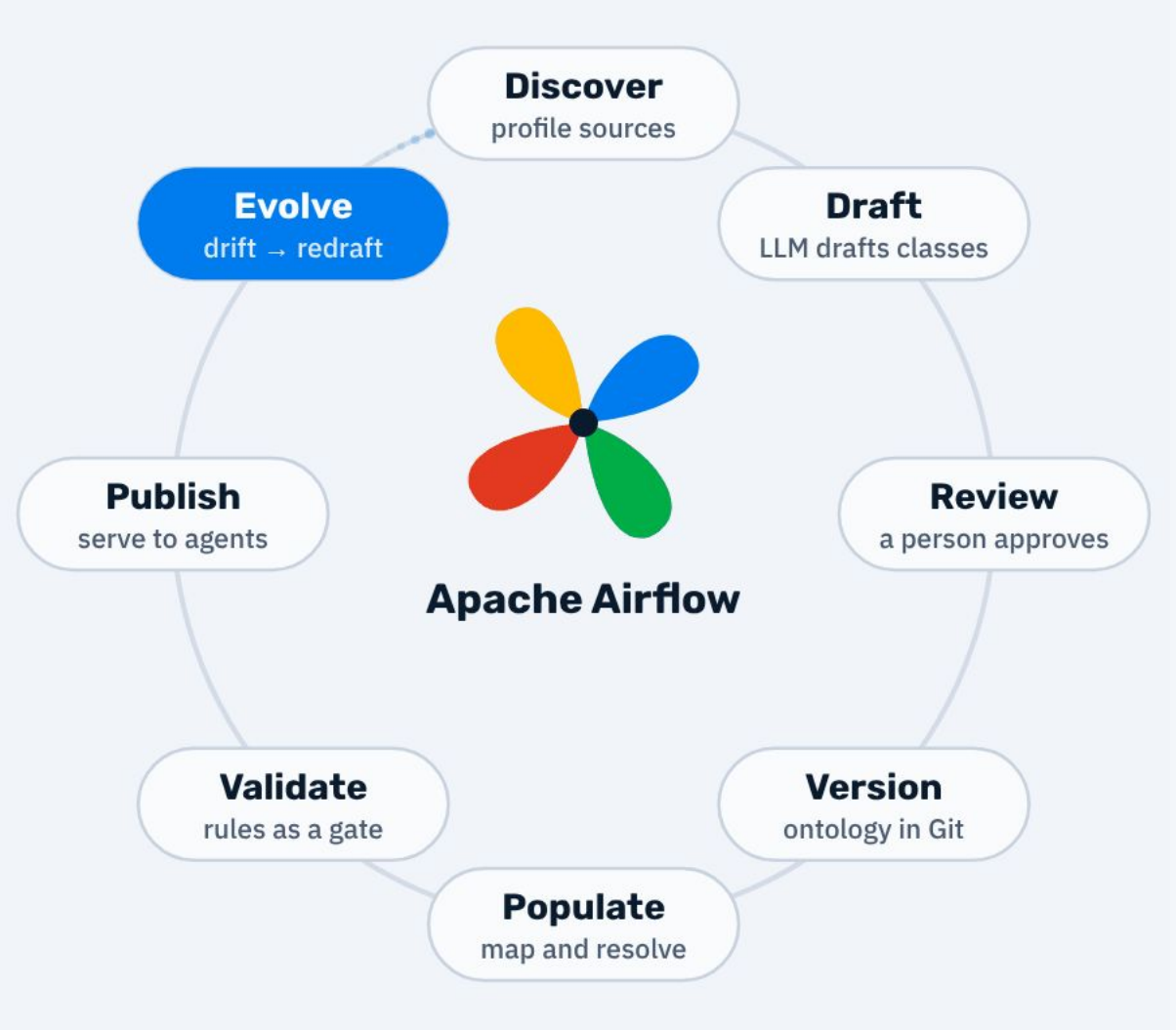
```
Rule: every Ticket is ABOUT exactly one Order
✕ T-78 has no order · quarantined
graph/published stays at the last good version
```

Loading proves the data arrived. Validation proves it still makes sense, and keeps agents on the last version that passed.

AIRFLOW AS AN ACTOR

Airflow keeps the ontology and the graph alive

People define the ontology. Airflow runs its life cycle: it drafts with an LLM, waits for approval, versions it, fills the graph, checks it and publishes it. When a source changes, it starts again.



WHY AIRFLOW

Airflow 3 already has the pieces

What the pipeline needs	Airflow 3 feature
One step per semantic layer: raw, conformed, graph	Assets and data-aware scheduling
React when a file or a message arrives	AssetWatchers, event-driven runs
One task per source and per document	Dynamic task mapping
LLM extraction limited to the ontology	@task.llm with a Pydantic output_type
Know which ontology version built the graph	DAG versioning, asset event metadata
Trace every fact back to its source	OpenLineage provider
Agents that read the graph through tools	@task.agent with MCPToolset
A person approves ambiguous merges	Human-in-the-loop review

REFERENCE ARCHITECTURE

Each task produces an asset

Sources: CRM · Postgres | ERP · warehouse | Tickets · queue | Contracts · PDFs | Ontology · Git

01
ingest_sources
@task · mapped per source
raw/*

→

02
conform_entities
@task · Polars
conformed/tables

→

03
extract_from_docs
@task.llm · mapped per doc
conformed/mentions

→

04
resolve_entities
@task · rules + review
conformed/entities

05
build_graph
@task · mapping files
graph/staging

→

06 · quality gate
validate_ontology
@task · pySHACL rules
graph/validated

→

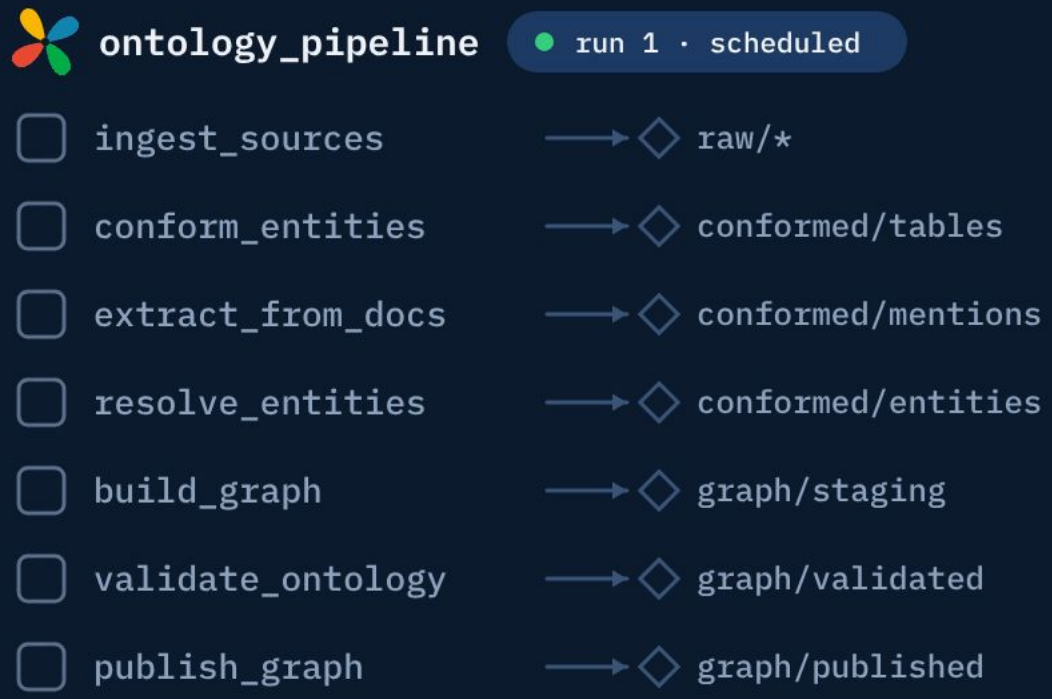
07
publish_graph
Neo4jOperator · MERGE
graph/published

→

Consumers
answer_question
@task.agent · MCPToolset
reads published

AIRFLOW IN ACTION

Airflow decides what agents see





THE QUALITY GATE IN CODE

Validation is just another asset

```python
from airflow.sdk import dag, task, Asset
from airflow.exceptions import AirflowFailException

@dag(schedule=[Asset("graph/staging")])      # runs when the staging graph changes
def validate_ontology():
    @task(outlets=[Asset("graph/validated")])
    def validate(*, outlet_events):
        ok, _, report = pyshacl.validate("staging.ttl", shacl_graph="shapes/v1.4.ttl")
        if not ok:
            raise AirflowFailException(report)     # nothing downstream runs
        outlet_events[Asset("graph/validated")].extra = {"ontology": "1.4"}
    validate()
validate_ontology()
```

LESSONS LEARNED

Five mistakes that break a graph

Mistake	Do this instead
Modeling the whole business at once	Start from 5 to 10 questions the agent must answer
Using an LLM on data that is already a table	Map tables with code, use the LLM only for free text
Skipping entity resolution	Resolve by ID, then name, then embeddings, then a person
Checking it loaded, not that it makes sense	Validate every run against the ontology’s rules
Rebuilding everything, keeping no provenance	Idempotent MERGE with source and run ID on every fact

And test your rules with bad data on purpose. A rule that points at the wrong class passes everything.

TAKE HOME

Three ideas to keep

Words become contracts

An ontology turns your business vocabulary into rules that data has to follow and agents can rely on.

The pipeline enforces the contract

Conform, resolve and validate on every run. Without that, agents read whatever arrived last.

Airflow is an actor, not a pipe

It keeps the ontology and the graph alive. Every layer is an asset, and agents read only the published one.

AIRFLOW SUMMIT ONLINE RECONNECT 2026

Thank you

Questions, and the hardest entity resolution case you have seen



Cesar Mendez

LinkedIn in/devcsar · csar.dev