Beyond Execution Dates: Empowering inference execution and hyper-parameter tuning with Airflow 3

Ankit Chaurasia

Rahul Vats

Problem in Airflow 2.x with execution_date

Each DAG run tied to a unique execution_date.

Event-driven workflow triggers

Inflexible for on-demand ML inference or hyperparameter tuning

Workarounds such as hacked timestamps used by data engineers

ASTRONOMER

Solution – AIP 83

AIP-83 – The Proposal

Rename execution_date to logical_date for clearer data interval semantics and run timing clarity

Remove uniqueness constraint on logical_date, allowing multiple DAG runs per interval and null values

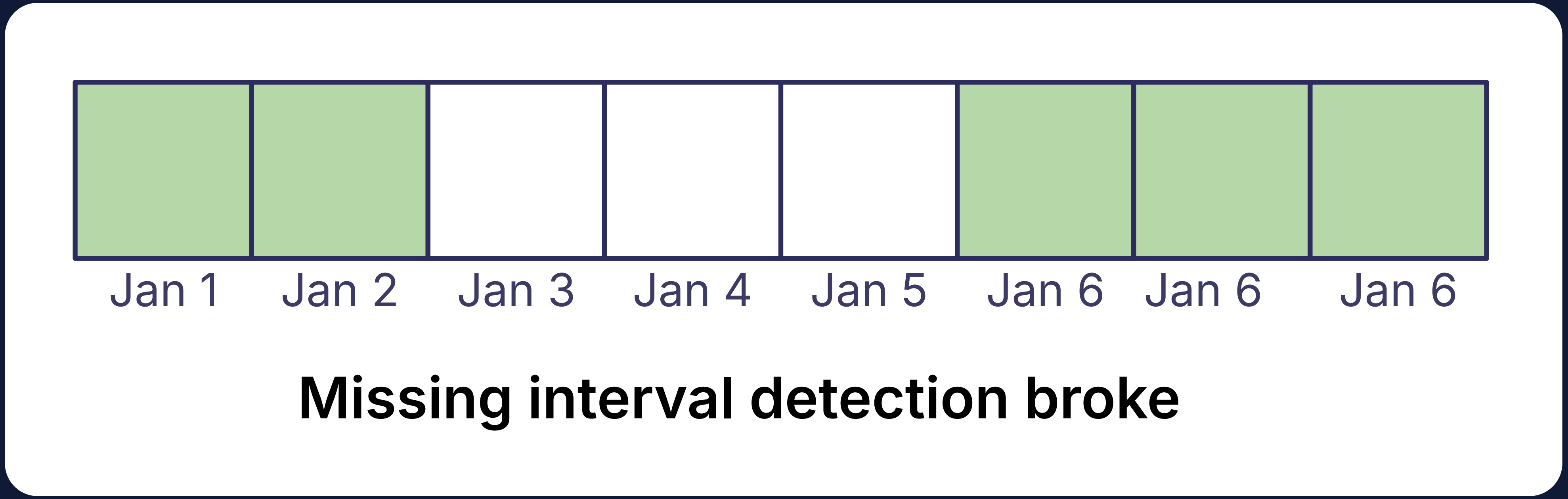
ASTRONOMER

Why Rename Execution Date?

- Execution date ≠ actual run time
- It represented data interval start
- Caused confusion for new users
- Logical Date = better reflects purpose

Challenges After Removing Uniqueness Constraint

- Backfill & catch up



- Depends_on_past

Challenges After Removing Uniqueness Constraint Contd.

- If you try to retrieve xcom of the task from the prior run, which run should it choose?
- Airflow UI Grid View struggled to visualize multiple runs with identical logical_date, causing user confusion

AIP-83 Amendment

Restored uniqueness for logical_date while allowing logical_date=NULL

Introduced run_after attribute to explicitly order DAG runs chronologically, independent of logical_date

Preserved data interval-based scheduling behavior alongside support for event-driven workflows

Technical Implementation & Migration Overview

1. Rename execution_date to logical_date
2. Run IDs as unique DAG run identifiers
3. Allow null logical_date
4. Update logging, API, CLI, and UI
5. Balance backward compatibility and flexibility

AIP 83 impact on Airflow features

- Asset triggered DAG now has null logical date
- From UI datepicker picks datetime now but we can send null in logical date here
- TriggerDagRunOperator defaults to null logical_date, supporting parallel and event-driven runs
- API and CLI require explicit logical_date or None to avoid ambiguity in runs
- These changes improve intuitive operation while enabling advanced ML and event-driven pipelines
- For runs with no logical date / data interval, there is a key error if these vars are accessed from template / context.

Real-World Use Case - Hyperparameter Tuning with Airflow 3

Launch N parallel DAG runs with different hyperparameters using `logical_date=None`

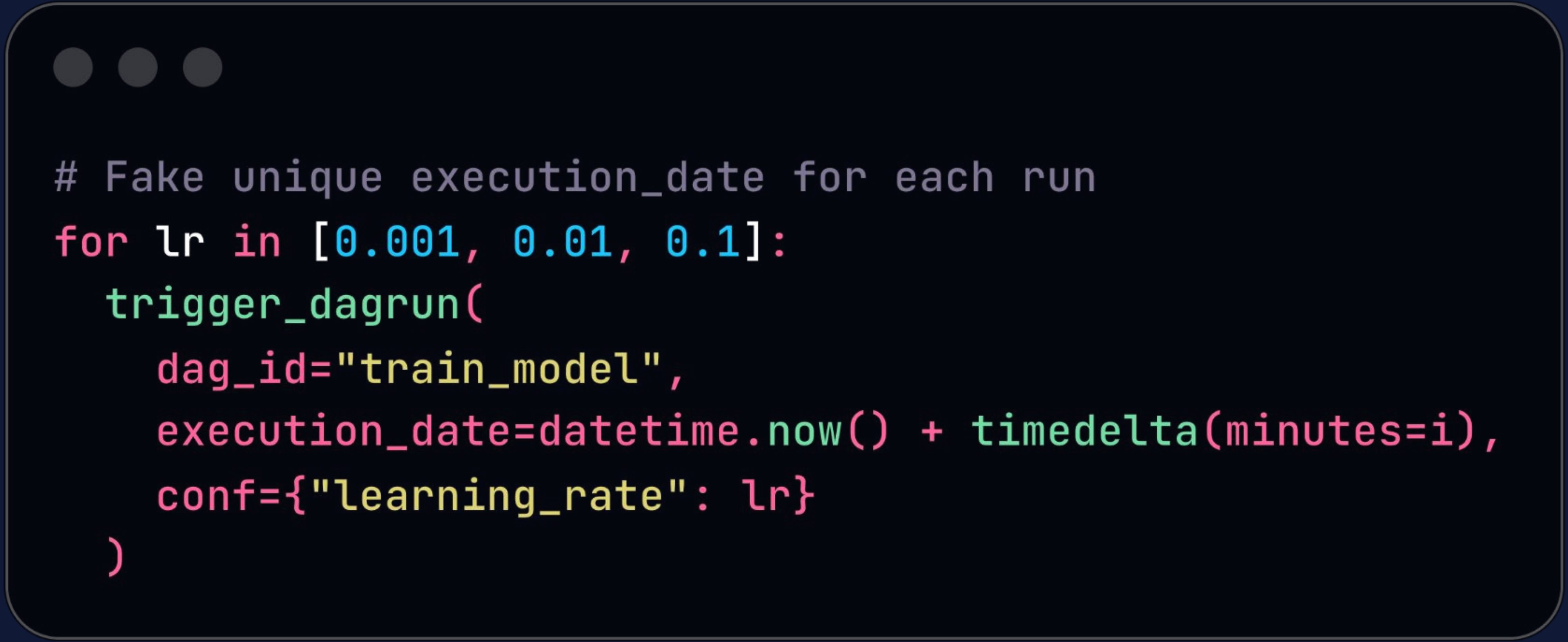
Runs execute independently without timestamp collisions, simplifying orchestration

Aggregate results externally or through downstream tasks for seamless analysis

Native support removes complex workarounds, speeding up model experimentation

Improves resource utilization and efficiency in ML pipeline orchestration

Hyperparameter Tuning – Before (Airflow 2.x)



```
# Fake unique execution_date for each run
for lr in [0.001, 0.01, 0.1]:
  trigger_dagrun(
    dag_id="train_model",
    execution_date=datetime.now() + timedelta(minutes=i),
    conf={"learning_rate": lr}
  )
```

- Hack: fake timestamps to bypass uniqueness.
- Confusing + error-prone.

ASTRONOMER

Enable Parallel Hyperparameter Tuning Runs in Airflow 3.0

```
TriggerDagRunOperator(
  task_id="trigger_training",
  trigger_dag_id="model_training_dag",
  conf={"lr": 0.01},
  logical_date=None # Allows parallel ad-hoc runs
)
```

Using `TriggerDagRunOperator` with `logical_date=None` allows launching multiple parallel DAG runs without requiring unique execution dates. This simplifies hyperparameter tuning by enabling concurrent ad-hoc runs with different configurations, accelerating ML workflows and experimentation.

Real-World Use Case – Inference Execution

Before Airflow 3, executing on-demand inference pipelines required complex workarounds to avoid conflicts from unique execution_date constraints. Airflow 3 eliminates this by allowing multiple ad-hoc DAG runs with logical_date=None to run concurrently, ordered by run_after. This enhancement supports real-time inference pipelines that dynamically handle incoming requests and scale horizontally without conflicts. Additionally, the improved UI distinctly separates these ad-hoc runs, enhancing monitoring and operational visibility.

Inference Execution – Before (Airflow 2.x)

```
trigger_dagrun(
  dag_id="inference_pipeline",
  execution_date=datetime.utcnow(), # required, but meaningless
  conf={"input_path": "s3://bucket/batch_123.csv"}
)
```

- Forced to assign fake execution_date.
- UI cluttered with meaningless timestamps.

ASTRONOMER

Inference Execution – After (Airflow 3.0)

```
trigger_dagrun(
  dag_id="inference_pipeline",
  conf={"input_path": "s3://bucket/batch_123.csv"},
  logical_date=None
)
```

- Clean ad-hoc runs
- Natural for batch inference, GenAI pipelines

ASTRONOMER

Inference Execution – Patterns

Push Pattern: Trigger via REST API

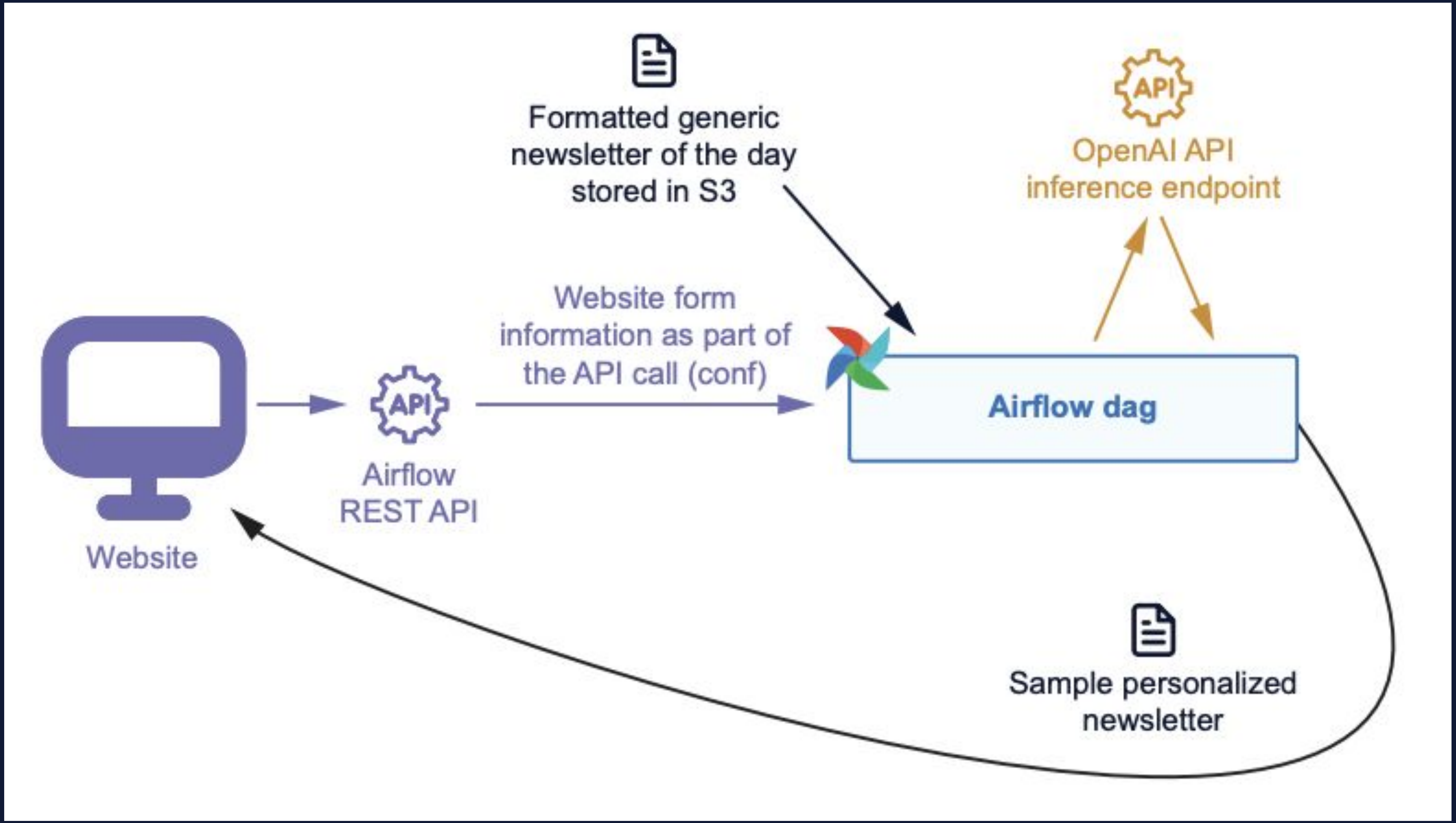
→ Website/API sends POST request with user input

Poll Pattern: Event-driven via message queue

→ AssetWatcher listens for events (e.g., SQS message)

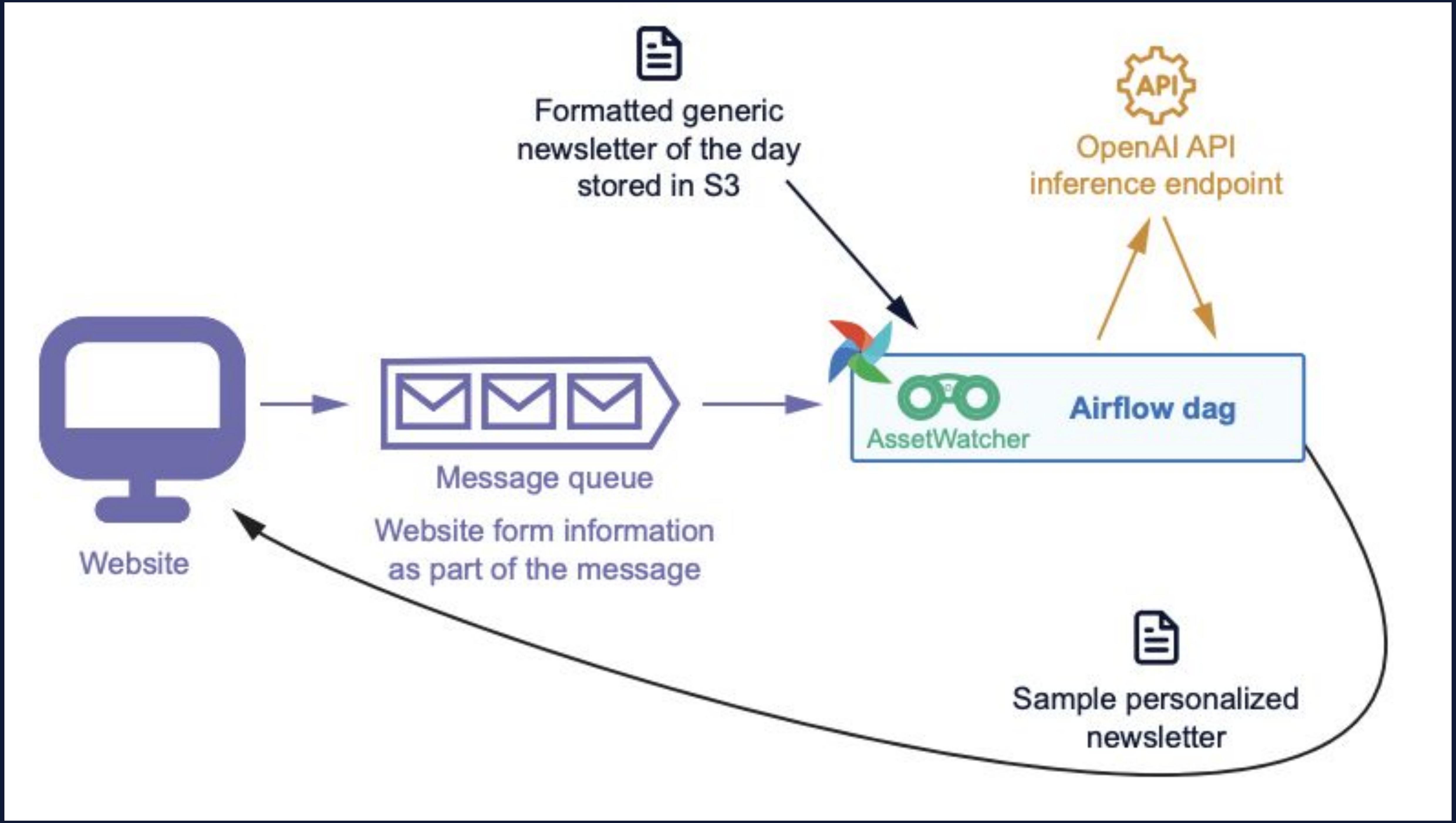
ASTRONOMER

Architecture Example – Push Pattern


Formatted generic newsletter of the day stored in S3
OpenAI API inference endpoint
Website form information as part of the API call (conf)
Airflow dag
Website
Airflow REST API
Sample personalized newsletter

ASTRONOMER

Architecture Example – Poll Pattern


Formatted generic newsletter of the day stored in S3
API
OpenAI API inference endpoint
Website
Message queue
Website form information as part of the message
AssetWatcher
Airflow dag
Sample personalized newsletter

ASTRONOMER

QUESTION?

ASTRONOMER

The 2025 Apache Airflow® Survey is here!

Fill it out to for a free Airflow 3 Fundamentals or DAG Authoring in Airflow 3 certification code

ASTRONOMER