Data-Intensive Distributed Computing

CS 451/651 (Fall 2025)

UNIVERSITY OF WATERLOO

Rubber, Meet Road: Orchestrators

(v1.0)

Week 5: September 30

Khaled Ammar

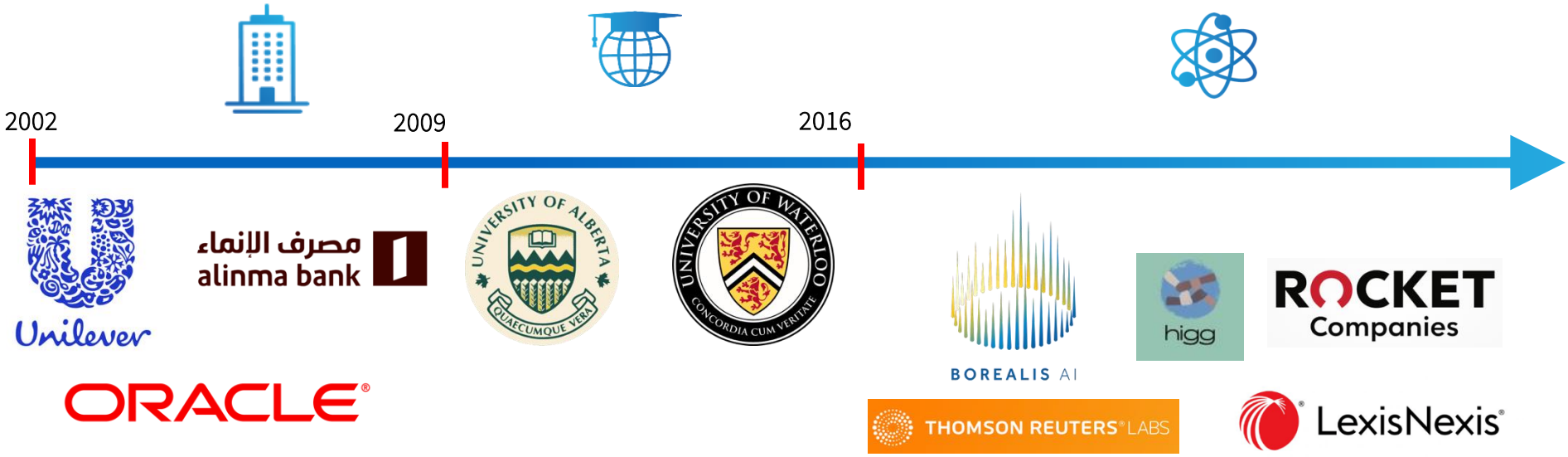
Rocket Innovation Studio

These slides are available at `https://lintool.github.io/cs451-2025f/`

Khaled Ammar

Director of Data Science – Applied Research

Rocket Innovation Studio



This Week

Now: Orchestrator

DAGs, ML pipelines, Airflow, etc.

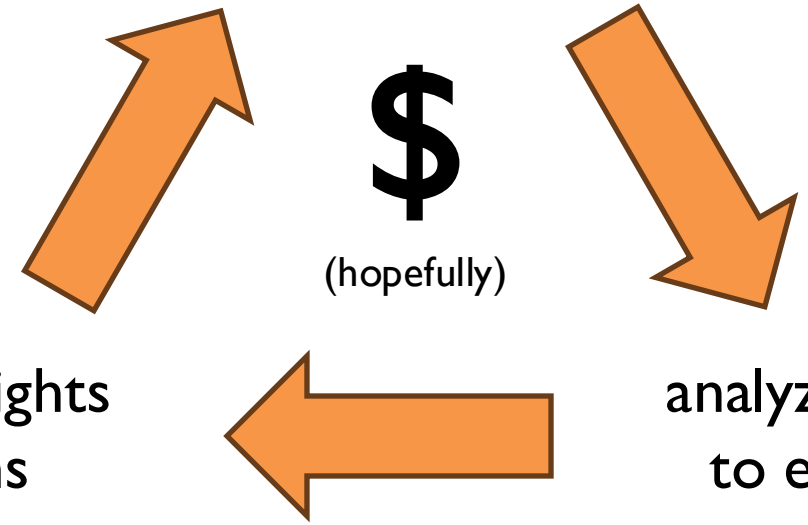
Next: Data Management in Production

Data Governance, Metadata, Feature Stores

Context…

The Data Flywheel

(a virtuous cycle)





Context…

What's this course about?

The *infrastructure* that supports the data flywheel.

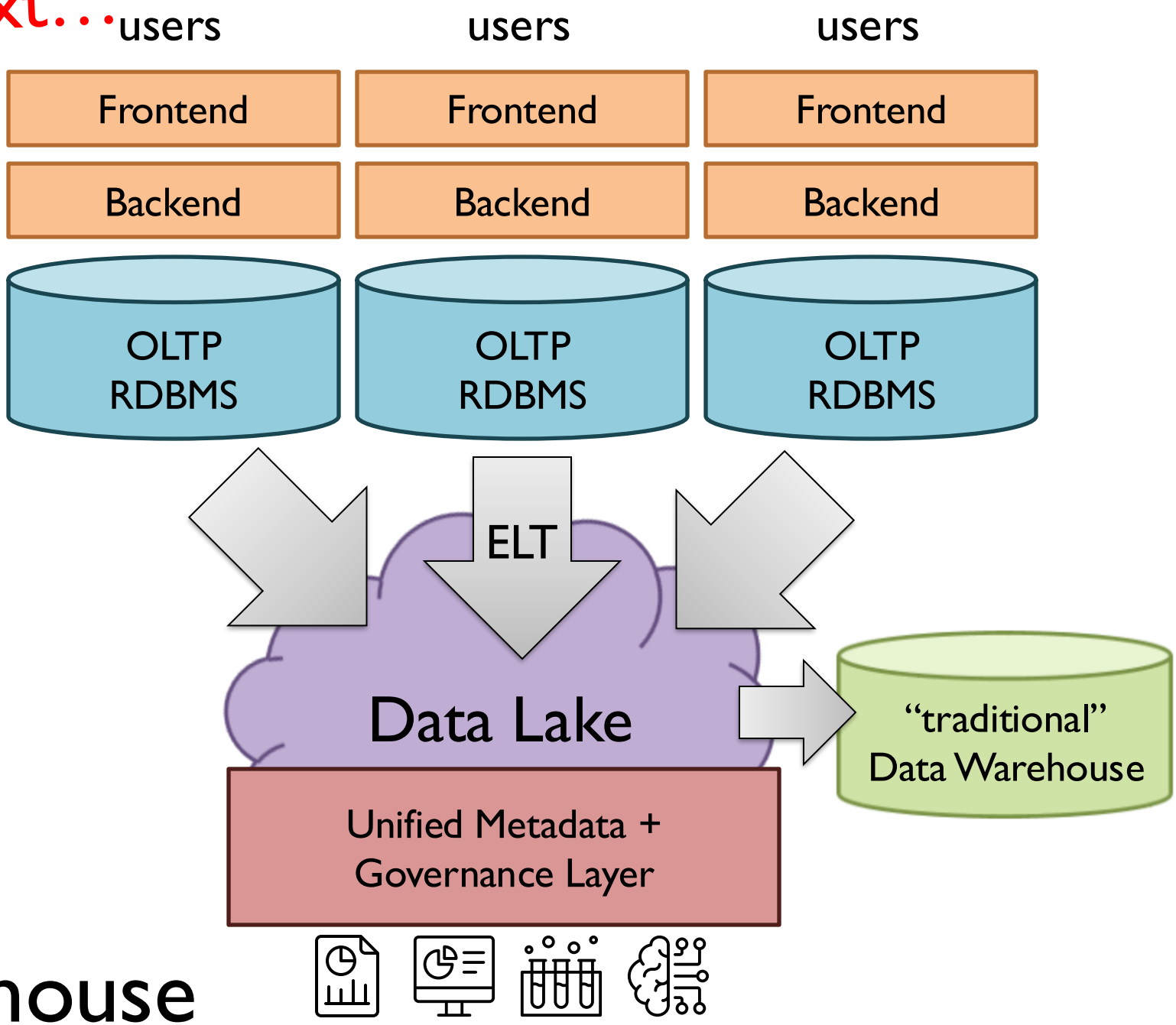
data platforms + data engineering

Context…

What problems do data platforms solve?

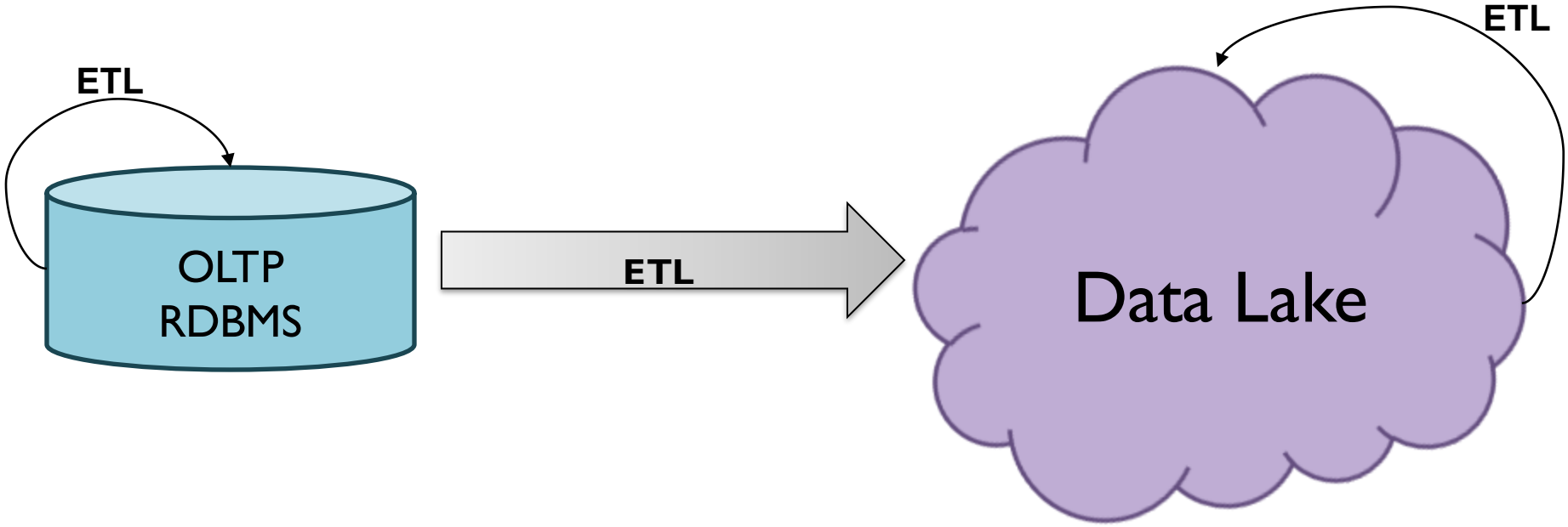
Ingesting, storing, manipulating, maintaining, serving…
the data that supports the data flywheel.

Context…



Lakehouse


ETL everywhere;
How to run them?
SPACE RANGER
LIGHTYEAR


ETL
OLTP
RDBMS
ETL
Data Lake
ETL

Creating Spark jobs is not enough!

How are you going to run it?

Creating Spark jobs is not enough!

How are you going to run it?

Script

Run multiple ETLs

Wake up in mid-night to run it

Manual retry

Multiple scripts are not easy to handle!

Cron Jobs

Run multiple ETLs

Scheduled to run automatically

No retry

Multiple jobs can run over each other

Workflow orchestrators

Declare a set of tasks, specify dependencies, retry policies, schedules, triggers, and monitor the whole pipeline.

Run multiple ETLs

Scheduled to run automatically – not only based on time but also using triggers!

Flexible retry policies

Clear dependencies between jobs

What is Workflow orchestration?

It is the process of managing the execution of independent tasks (e.g. Spark jobs) across multiple systems and time.

Common features:

- **Dependency management:** specify that task B depends on task A's success.
- **Scheduling:** specify when a task can run and how often
- **Retry policies:** specify how many times to retry a failed job, any delay policies.
- **Backfills / catchup:** specify how to handle missed past intervals.
- **Monitoring & Alerting:** view the overall status of the system including failures and latency.

Airflow DAG Example

```
12
13 default_args = {
14     "owner": "Khaled",
15     "depends_on_past": False,
16     "retries": 2,
17     "retry_delay": timedelta(seconds=10),
18 }
19
20 with DAG(
21     dag_id="etl_daily",
22     default_args=default_args,
23     start_date=datetime(2025, 9, 1),
24     schedule_interval="@daily",
25     catchup=False,
26     tags=["demo","etl"],
27 ) as dag:
28
29     extract = PythonOperator(
30         task_id="extract",
31         python_callable=extract_run,
32         op_kwargs={"day": "2025-09-03"},
33     )
34
35     transform = PythonOperator(
36         task_id="transform",
37         python_callable=transform_run,
38         op_kwargs={"day": "2025-09-03"},
39     )
40
41     load = PythonOperator(
42         task_id="load",
43         python_callable=load_run,
44     )
45
46     extract >> transform >> load
```

Stateless DAG

Retry policy

Define DAG details

Define Extract step

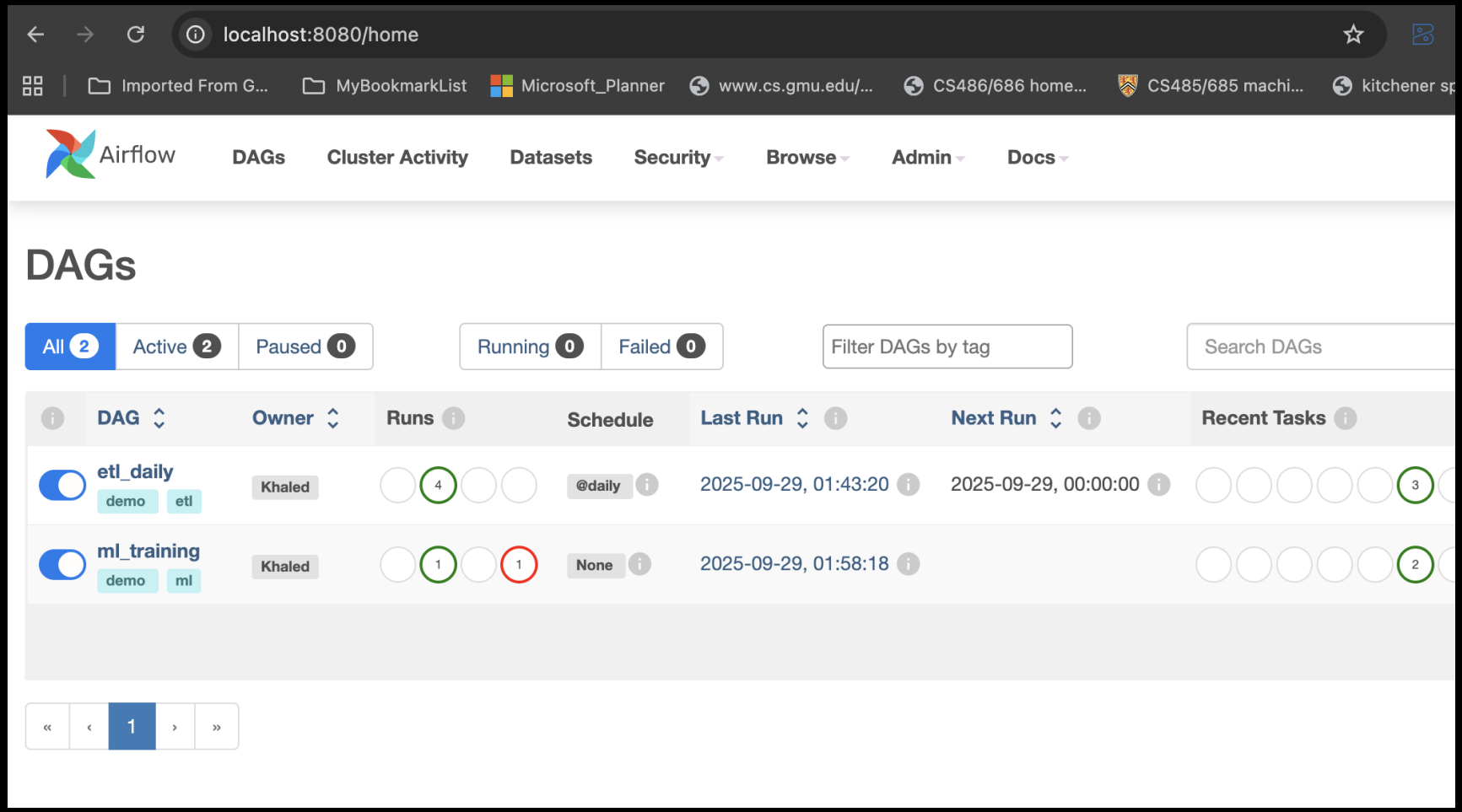
Define Transform step

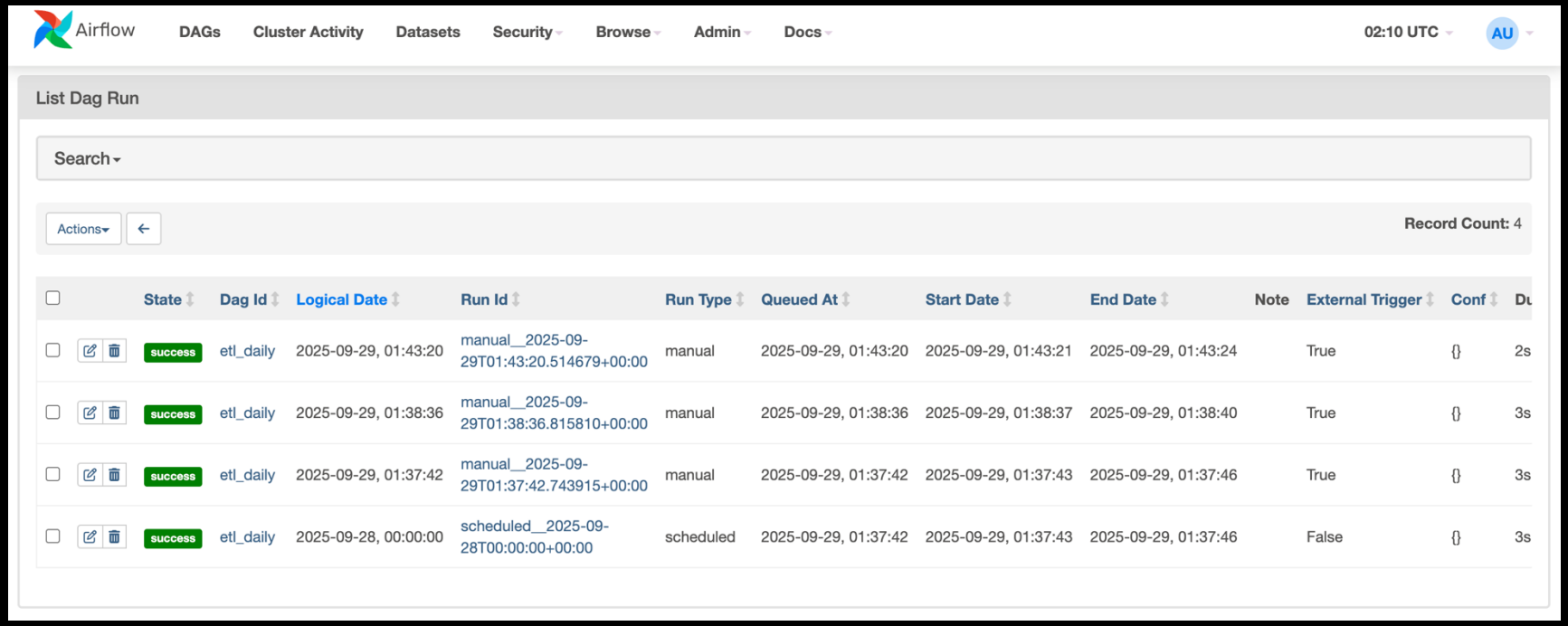
Define Load step

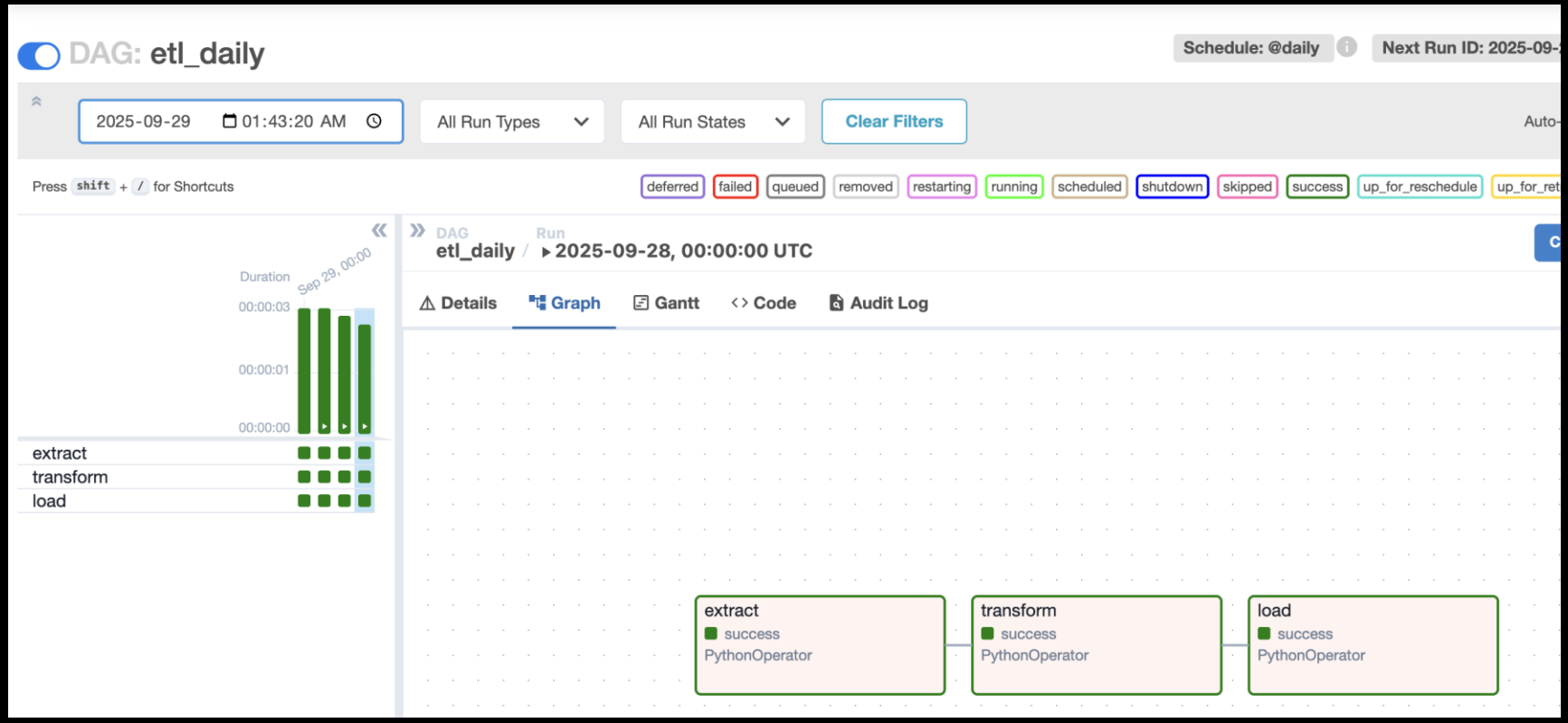
Order to run these steps

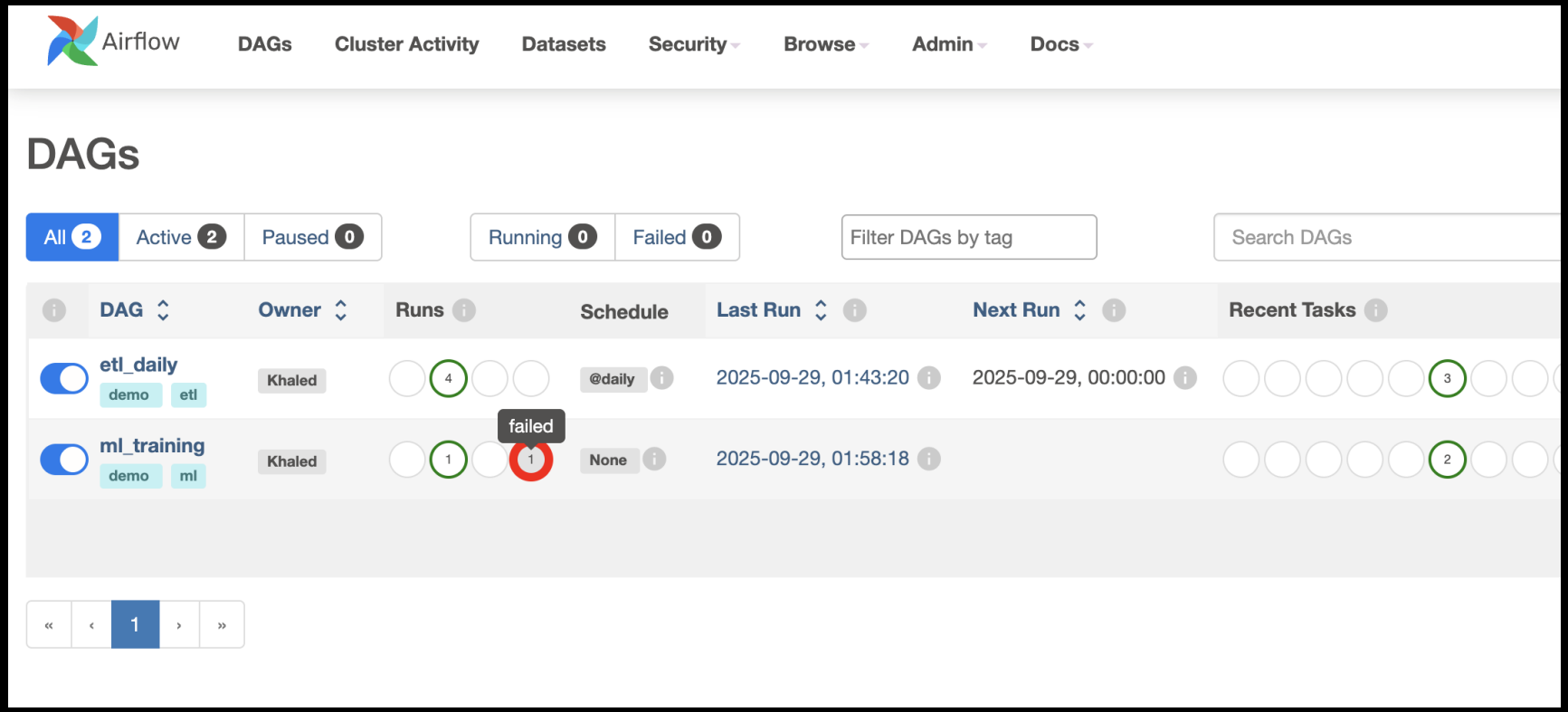
Airflow DAG Example

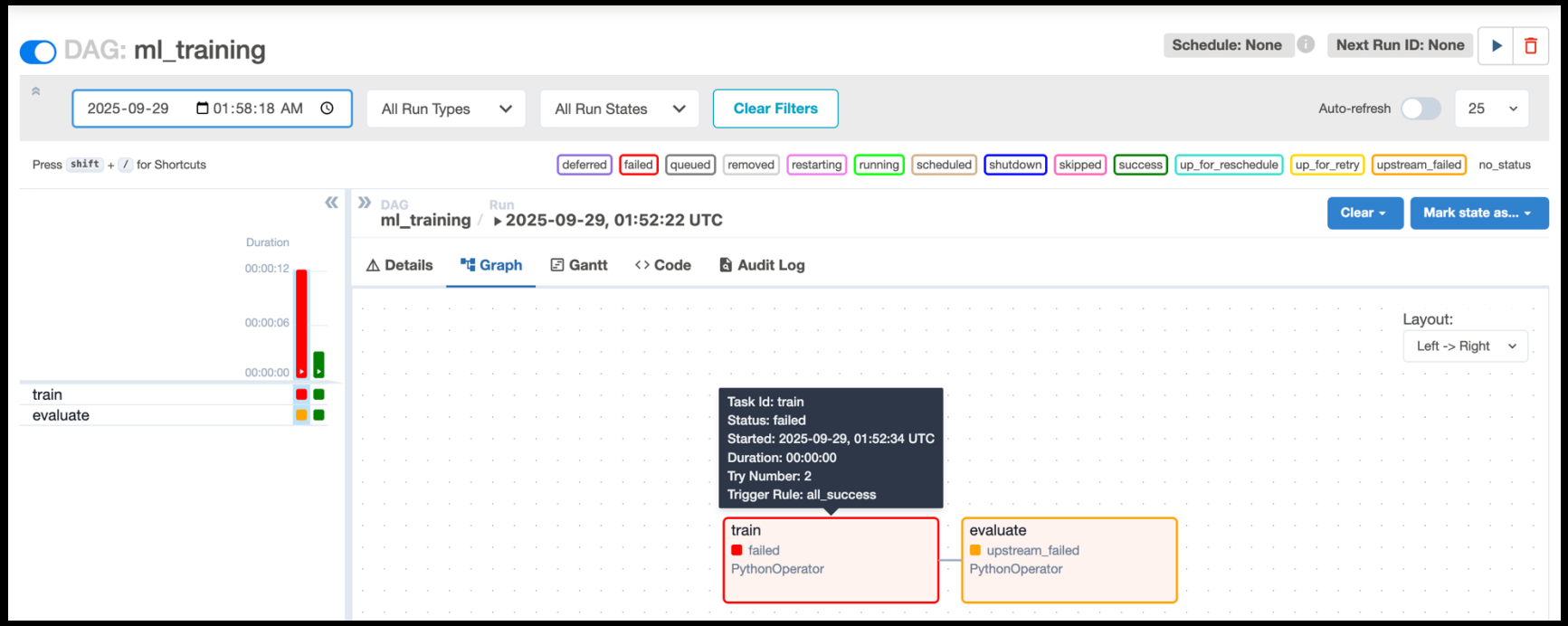
Live Demo


localhost:8080/home
Imported From G...
MyBookmarkList
Microsoft_Planner
www.cs.gmu.edu/...
CS486/686 home...
CS485/685 machi...
kitchener sp
Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
DAGs
All 2
Active 2
Paused 0
Running 0
Failed 0
Filter DAGs by tag
Search DAGs
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
etl_daily
demo
etl
Khaled
@daily
2025-09-29, 01:43:20
2025-09-29, 00:00:00
ml_training
demo
ml
Khaled
None
2025-09-29, 01:58:18
« ‹ 1 › »

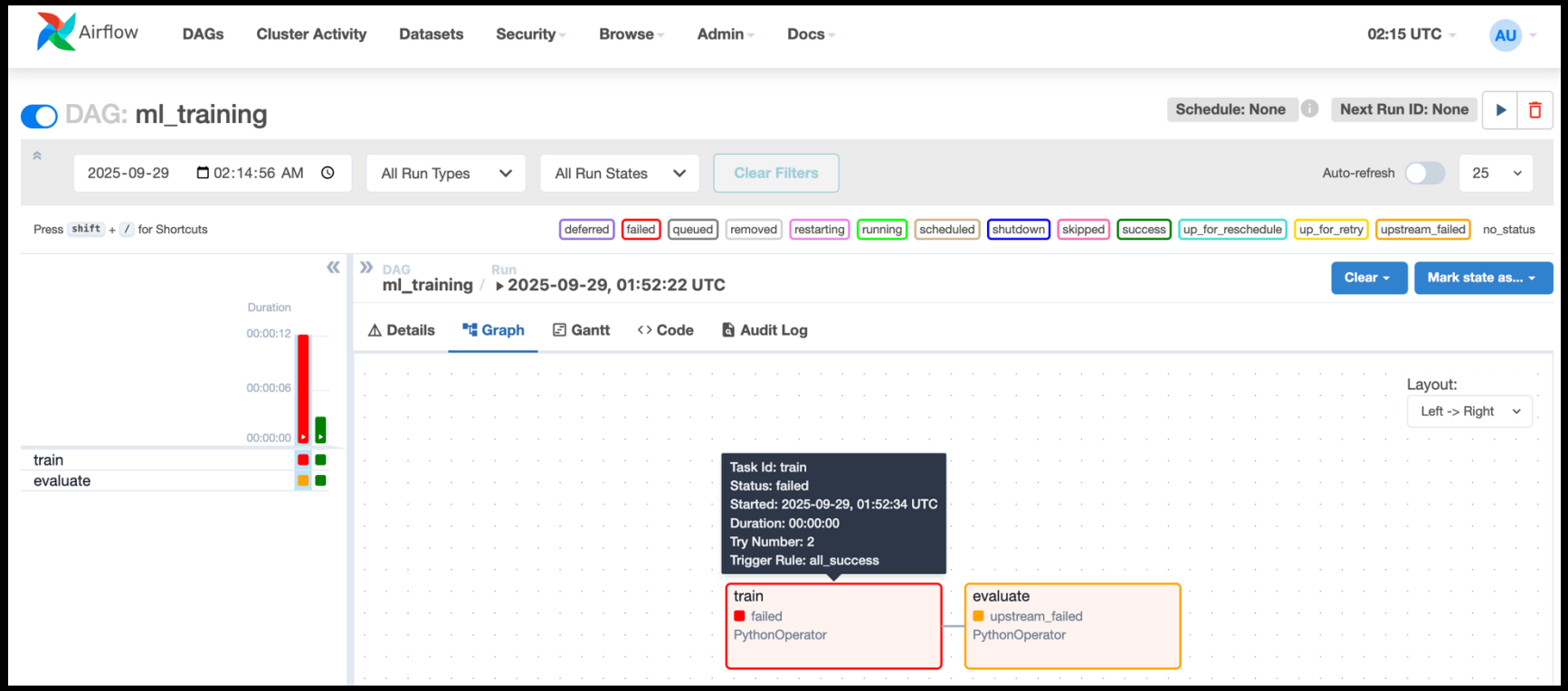

Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
02:10 UTC
AU
List Dag Run
Search
Actions
Record Count: 4
State
Dag Id
Logical Date
Run Id
Run Type
Queued At
Start Date
End Date
Note
External Trigger
Conf
Du
success
etl_daily
2025-09-29, 01:43:20
manual__2025-09-29T01:43:20.514679+00:00
manual
2025-09-29, 01:43:20
2025-09-29, 01:43:21
2025-09-29, 01:43:24
True
{}
2s
success
etl_daily
2025-09-29, 01:38:36
manual__2025-09-29T01:38:36.815810+00:00
manual
2025-09-29, 01:38:36
2025-09-29, 01:38:37
2025-09-29, 01:38:40
True
{}
3s
success
etl_daily
2025-09-29, 01:37:42
manual__2025-09-29T01:37:42.743915+00:00
manual
2025-09-29, 01:37:42
2025-09-29, 01:37:43
2025-09-29, 01:37:46
True
{}
3s
success
etl_daily
2025-09-28, 00:00:00
scheduled__2025-09-28T00:00:00+00:00
scheduled
2025-09-29, 01:37:42
2025-09-29, 01:37:43
2025-09-29, 01:37:46
False
{}
3s

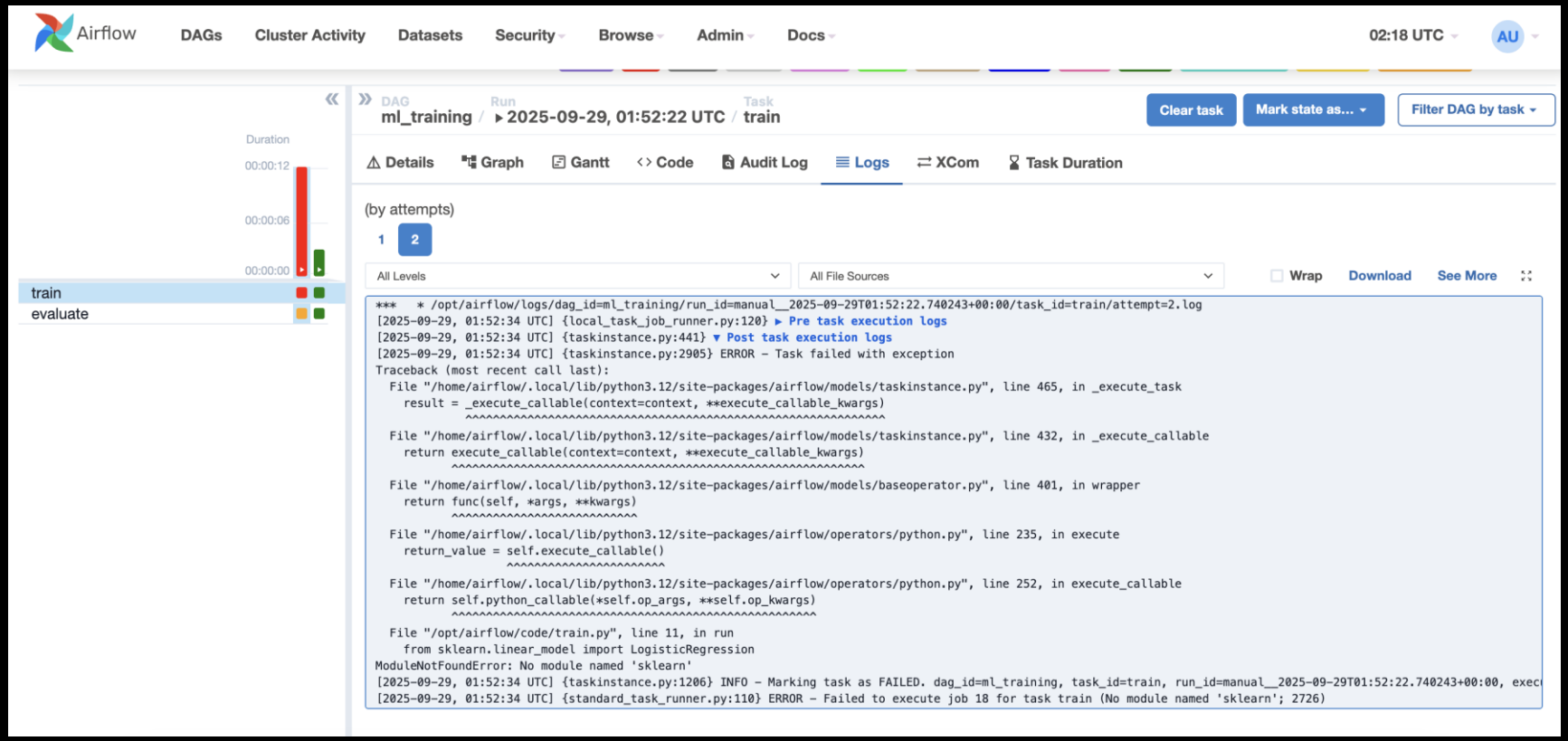

DAG: etl_daily
Schedule: @daily
Next Run ID: 2025-09-
2025-09-29
01:43:20 AM
All Run Types
All Run States
Clear Filters
Auto-
Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
shutdown
skipped
success
up_for_reschedule
up_for_ret
Duration
Sep 29, 00:00
00:00:03
00:00:01
00:00:00
extract
transform
load
DAG
etl_daily
Run
2025-09-28, 00:00:00 UTC
Details
Graph
Gantt
Code
Audit Log
extract
success
PythonOperator
transform
success
PythonOperator
load
success
PythonOperator


Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
DAGs
All 2
Active 2
Paused 0
Running 0
Failed 0
Filter DAGs by tag
Search DAGs
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
etl_daily
demo
etl
Khaled
@daily
2025-09-29, 01:43:20
2025-09-29, 00:00:00
ml_training
demo
ml
Khaled
failed
None
2025-09-29, 01:58:18
1

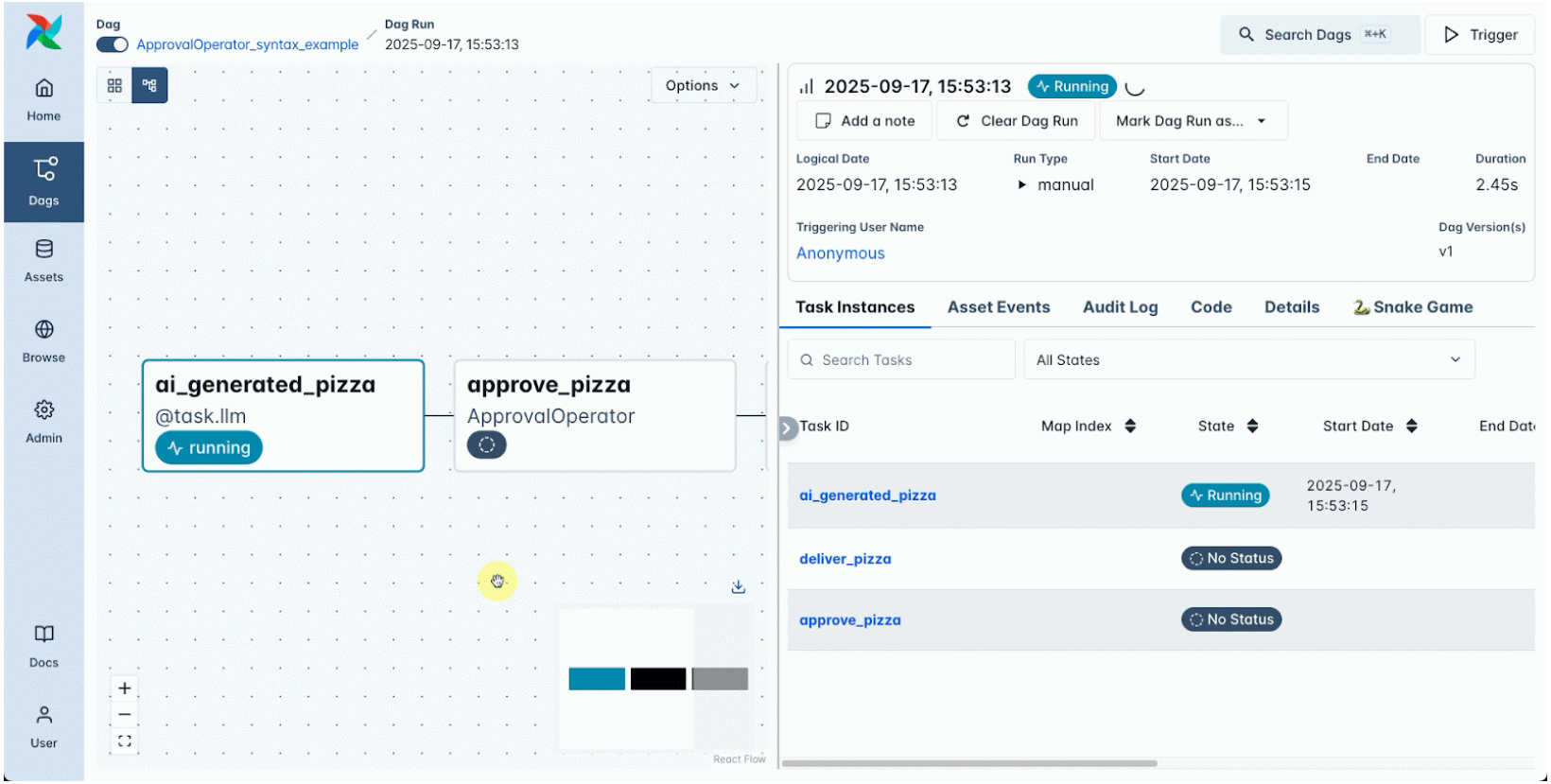

DAG: ml_training
Schedule: None
Next Run ID: None
2025-09-29
01:58:18 AM
All Run Types
All Run States
Clear Filters
Auto-refresh
25
Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
shutdown
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Duration
00:00:12
00:00:06
00:00:00
train
evaluate
DAG
ml_training
Run
2025-09-29, 01:52:22 UTC
Clear
Mark state as...
Details
Graph
Gantt
Code
Audit Log
Layout:
Left -> Right
Task Id: train
Status: failed
Started: 2025-09-29, 01:52:34 UTC
Duration: 00:00:00
Try Number: 2
Trigger Rule: all_success
train
failed
PythonOperator
evaluate
upstream_failed
PythonOperator

```python
default_args = {
    "owner": "Khaled",
    "depends_on_past": False,
    "retries": 1,
    "retry_delay": timedelta(seconds=10),
}

with DAG(
    dag_id="ml_training",
    default_args=default_args,
    start_date=datetime(2025, 9, 1),
    schedule_interval=None,
    catchup=False,
    tags=["demo","ml"],
) as dag:

    train = PythonOperator(
        task_id="train",
        python_callable=train_run,
    )

    evaluate = PythonOperator(
        task_id="evaluate",
        python_callable=evaluate_run,
        op_kwargs={"threshold": 0.6},
    )

    train >> evaluate

```


Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
02:15 UTC
AU
DAG: ml_training
Schedule: None
Next Run ID: None
2025-09-29
02:14:56 AM
All Run Types
All Run States
Clear Filters
Auto-refresh
25
Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
shutdown
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Duration
00:00:12
00:00:06
00:00:00
train
evaluate
DAG
ml_training
Run
2025-09-29, 01:52:22 UTC
Clear
Mark state as...
Details
Graph
Gantt
Code
Audit Log
Layout:
Left -> Right
Task Id: train
Status: failed
Started: 2025-09-29, 01:52:34 UTC
Duration: 00:00:00
Try Number: 2
Trigger Rule: all_success
train
failed
PythonOperator
evaluate
upstream_failed
PythonOperator


Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
02:18 UTC
AU
Duration
00:00:12
00:00:06
00:00:00
train
evaluate
DAG
ml_training
Run
2025-09-29, 01:52:22 UTC
Task
train
Clear task
Mark state as...
Filter DAG by task
Details
Graph
Gantt
Code
Audit Log
Logs
XCom
Task Duration
(by attempts)
1
2
All Levels
All File Sources
Wrap
Download
See More
*** * /opt/airflow/logs/dag_id=ml_training/run_id=manual__2025-09-29T01:52:22.740243+00:00/task_id=train/attempt=2.log
[2025-09-29, 01:52:34 UTC] {local_task_job_runner.py:120} ▶ Pre task execution logs
[2025-09-29, 01:52:34 UTC] {taskinstance.py:441} ▼ Post task execution logs
[2025-09-29, 01:52:34 UTC] {taskinstance.py:2905} ERROR - Task failed with exception
Traceback (most recent call last):
File "/home/airflow/.local/lib/python3.12/site-packages/airflow/models/taskinstance.py", line 465, in _execute_task
result = _execute_callable(context=context, **execute_callable_kwargs)
File "/home/airflow/.local/lib/python3.12/site-packages/airflow/models/taskinstance.py", line 432, in _execute_callable
return execute_callable(context=context, **execute_callable_kwargs)
File "/home/airflow/.local/lib/python3.12/site-packages/airflow/models/baseoperator.py", line 401, in wrapper
return func(self, *args, **kwargs)
File "/home/airflow/.local/lib/python3.12/site-packages/airflow/operators/python.py", line 235, in execute
return_value = self.execute_callable()
File "/home/airflow/.local/lib/python3.12/site-packages/airflow/operators/python.py", line 252, in execute_callable
return self.python_callable(*self.op_args, **self.op_kwargs)
File "/opt/airflow/code/train.py", line 11, in run
from sklearn.linear_model import LogisticRegression
ModuleNotFoundError: No module named 'sklearn'
[2025-09-29, 01:52:34 UTC] {taskinstance.py:1206} INFO - Marking task as FAILED. dag_id=ml_training, task_id=train, run_id=manual__2025-09-29T01:52:22.740243+00:00,
[2025-09-29, 01:52:34 UTC] {standard_task_runner.py:110} ERROR - Failed to execute job 18 for task train (No module named 'sklearn'; 2726)

Airflow Recent release: 3.1.0



https://airflow.apache.org/blog/airflow-3.1.0/

State of Airflow in 2025

Thousands of DAGs are running daily.

Airflow remains the scheduler and dependency manager, not the compute engine.

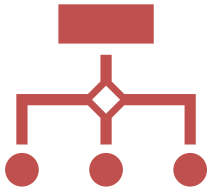
Orchestrate jobs across heterogenous systems.

DAGs often have data quality and validation steps.

External data availability often start DAGs in integration use cases.

Some teams use DAGs beyond data and ML, including sending emails, moving files, etc.

Limitation of Airflow



DAGs are defined when the DAG file is parsed, but complex workflows are data-dependent!

Example: One task discover 20 tables in an S3 path, then need to create 20 different tasks with different schema definition. Everything is doable with code, but it gets messy!

Airflow is fundamentally a batch scheduler; one task needs to finish for another to start!

Example: Executing a long running (streaming) task like Kafka consumer that continuously consume data and generate results cause the DAG to stuck!

Modern Orchestrators

Tool	Strengths	Challenges
Airflow	Very mature, strong for batch ETL / Python orchestration	Static DAGs, no streaming support
Prefect	Supports dynamic workflow	Limited non-Python support, can support data flows but not optimized
dbt	Excellent for SQL and data lineage	SQL-only, and no streaming support
Kubeflow	Specialize in ML pipelines on K8; native TensorFlow/PyTorch integration	Requires K8 expertise, limited outside ML, heavy for small projects – no general support for streaming

It is not uncommon to use multiple orchestrators in the same enterprise

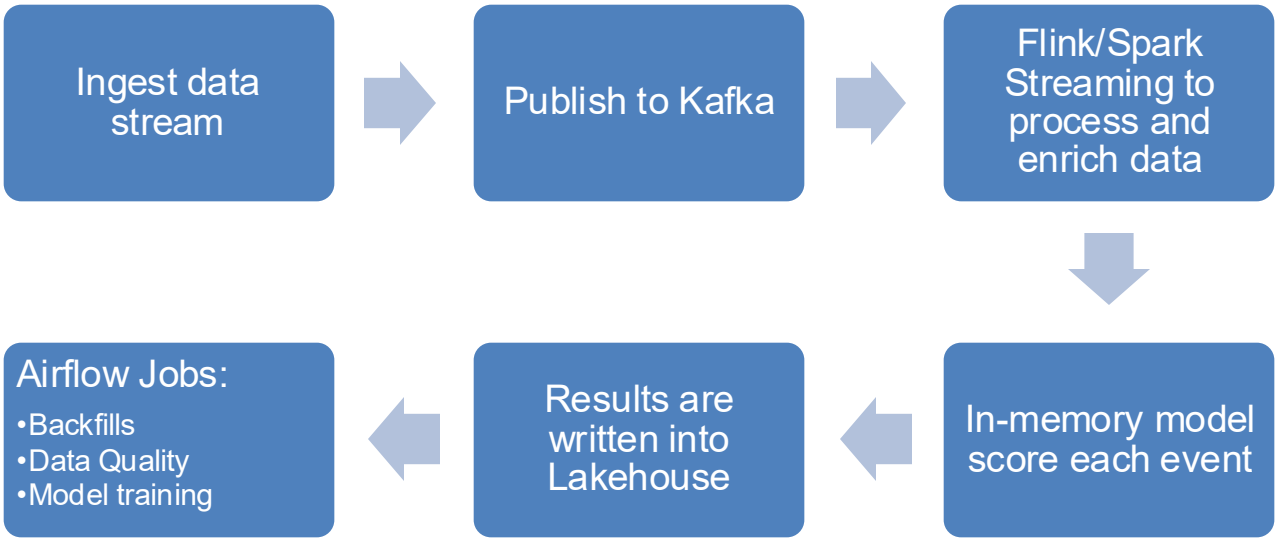
Challenge with streaming workflows

A task can start processing the output before its predecessor finishes!

Batch

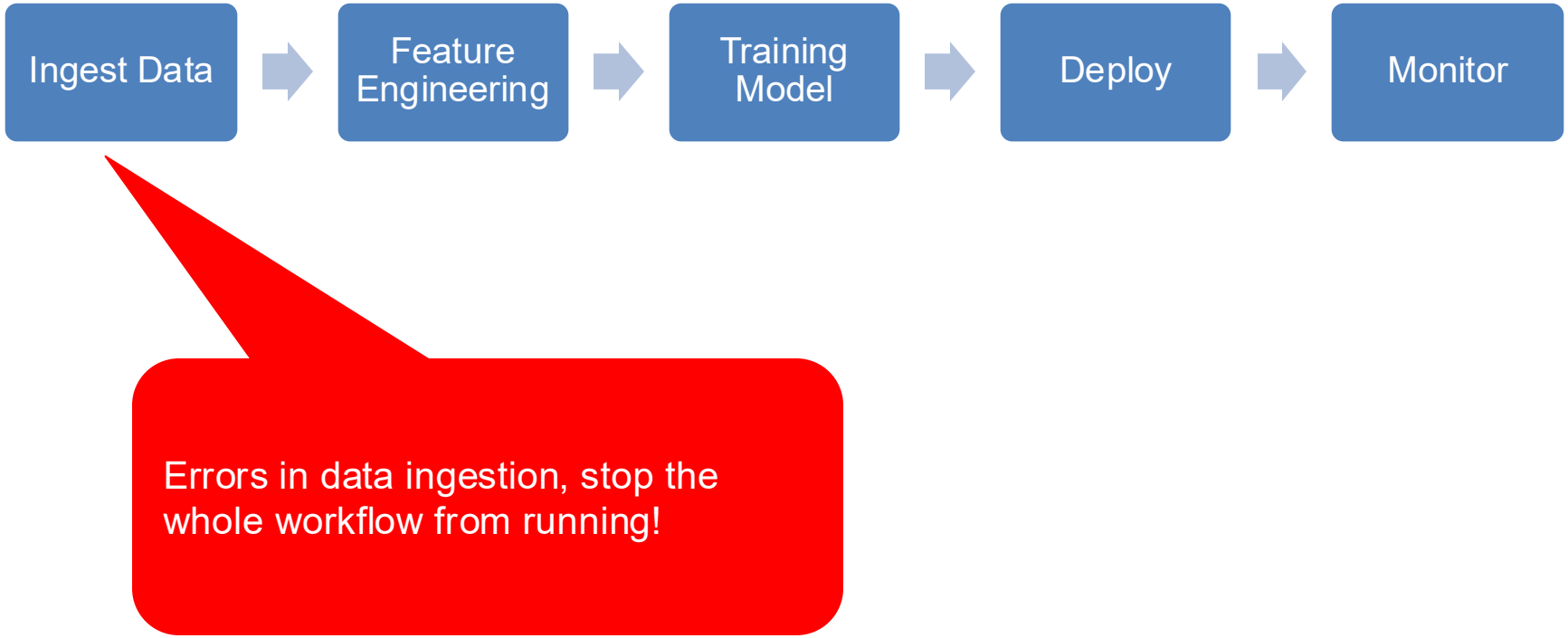
Pipeline / Stream

Streaming Orchestration



Ingestion: Kafka
Stream computing: Flink | Spark Streaming
Model serving: Flink operator
Output: Kafka | S3 | Lakehouse
Batch orchestration: Airflow | Prefect

ML workflow


Ingest Data
Feature Engineering
Training Model
Deploy
Monitor
Errors in data ingestion, stop the whole workflow from running!

ML workflow

When data become available,
“Backfill” is needed for delayed data!

e.g. weekly aggregation should be recomputed.

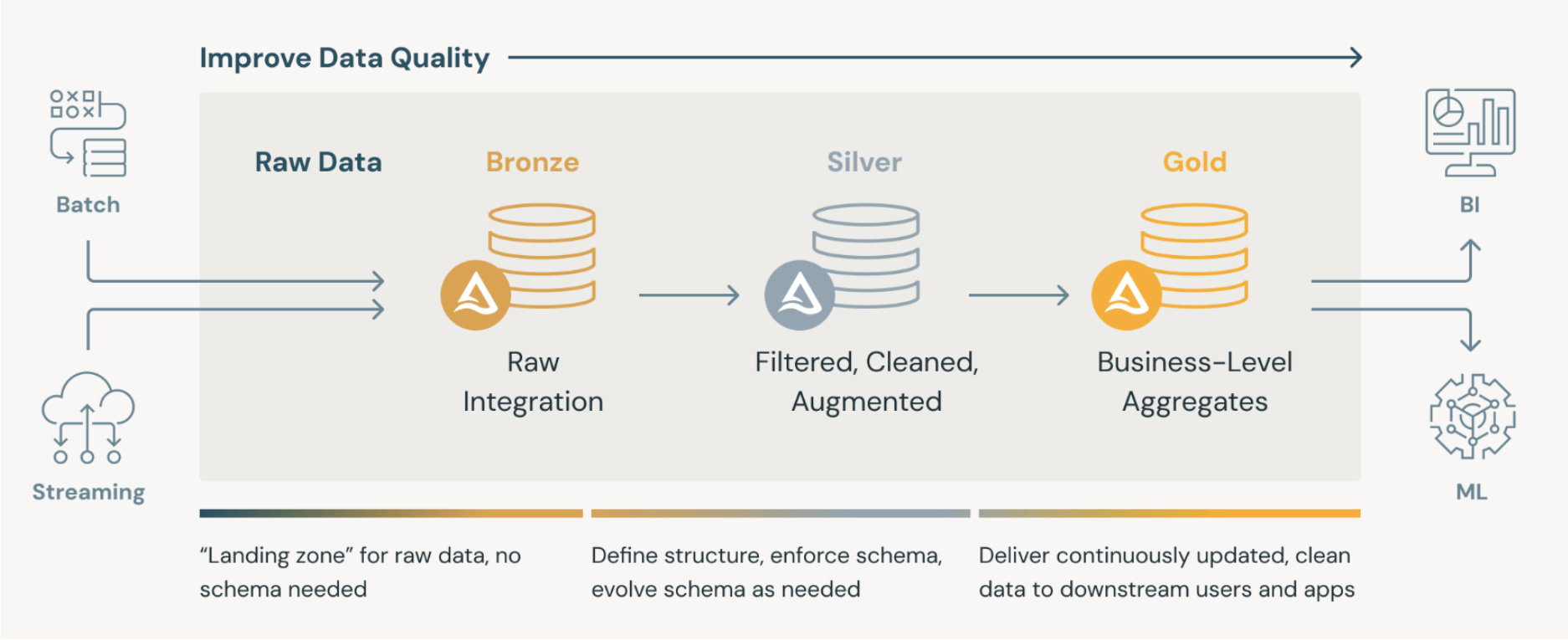
ML workflow

Failures in model training block new deployments.

In practice, expensive training could be avoided if not enough new data is available?

Data Science Setup in Enterprises:

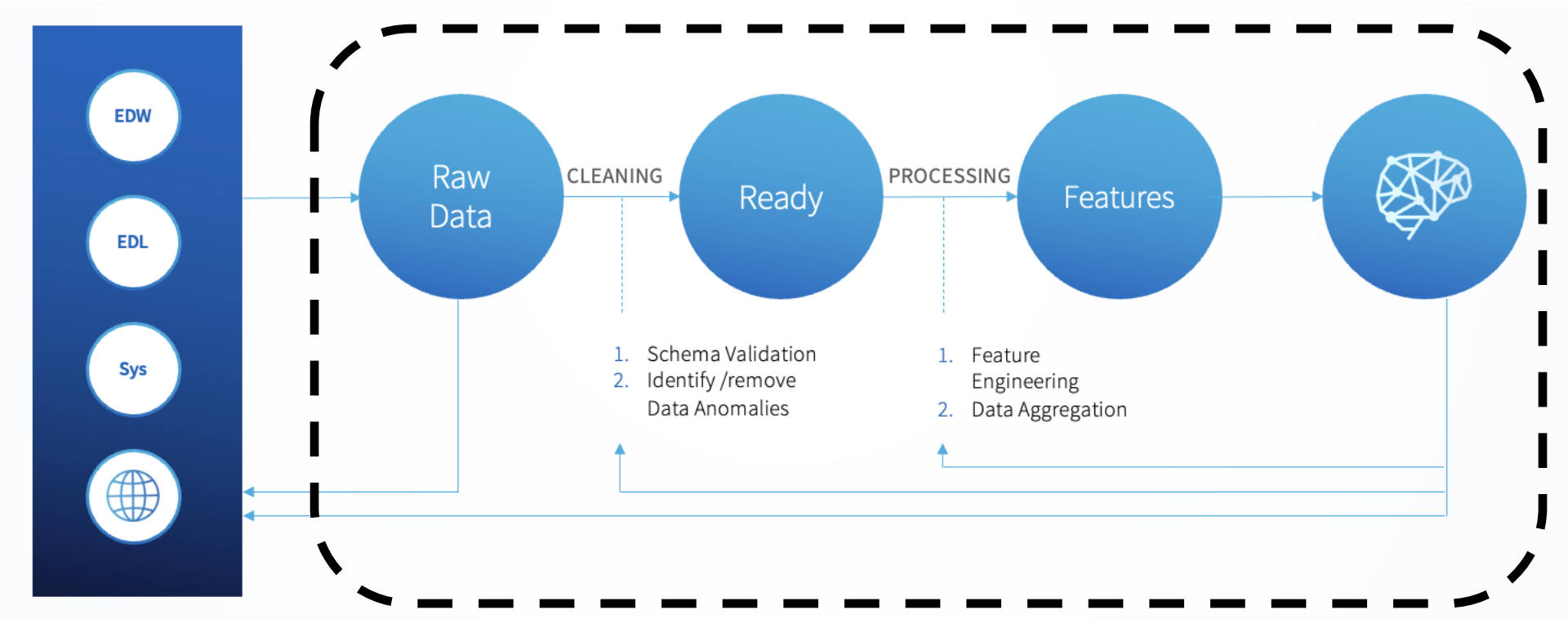
Medallion Architecture



Source: https://www.databricks.com/glossary/medallion-architecture

Data Science Setup in Enterprises:

Innovation Area



When not to use Orchestrator?

Is orchestrator enough for data pipeline?