AIRFLOW SUMMIT

Airflow at OpenAI 3.0

Ping Zhang & Hongyi Wang

Agenda

- Airflow Journey at OpenAI
- Challenges
- Reliability
- Scaling
- Tooling
- Future work

From Expectation to Reality

AIRFLOW SUMMIT

Early Days at OpenAI

- 20 engineers on the data platform team
- No dedicated Airflow team, limited engineering bandwidth
- Each team used its own orchestration: Dagster, notebooks, scripts
- This flexibility helped early on, but made production harder

The Shift to Airflow

- We made a deliberate shift: unify on Airflow as the backbone
- Moved DAGs into the monorepo, tied task logic to orchestration
- Adopted software best practices: review, test, deploy together
- This gave us a consistent, flexible foundation to scale
- Set the stage for solving reliability, scaling, and usability

The Shift to Airflow

- We made a deliberate shift: unify on Airflow as the backbone
- Moved DAGs into the monorepo, tied task logic to orchestration
- Adopted software best practices: review, test, deploy together
- This gave us a consistent, flexible foundation to scale
- Set the stage for solving reliability, scaling, and usability

Challenges at Scale

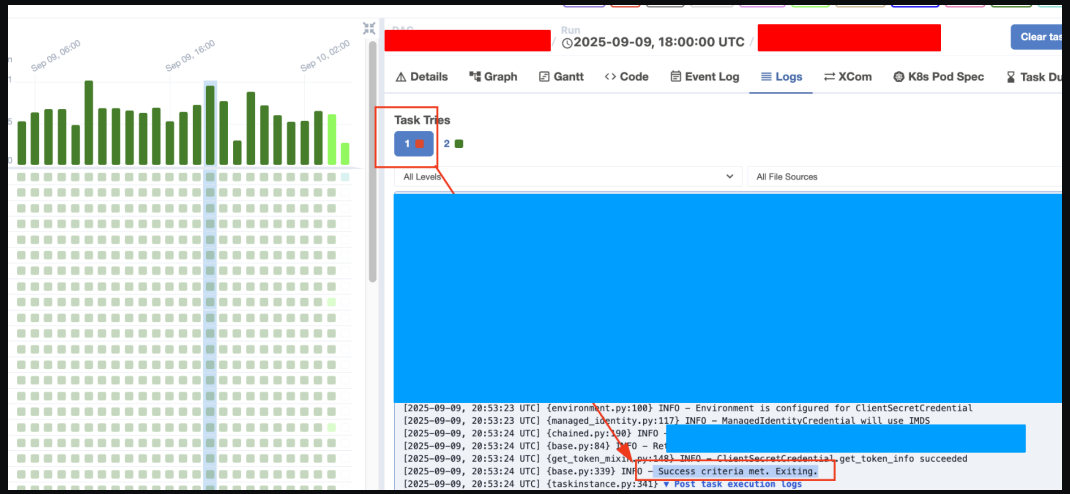
- **Reliability**: transient infra failures disrupted daily pipelines
- **Performance**: scheduler, metadata DB, and file I/O under pressure
- **Simplicity**: users needed fast iteration, but tooling was too slow
- Airflow became mission-critical faster than our team could grow

Edge cases define reliability

Edge case - Pod can be killed anytime

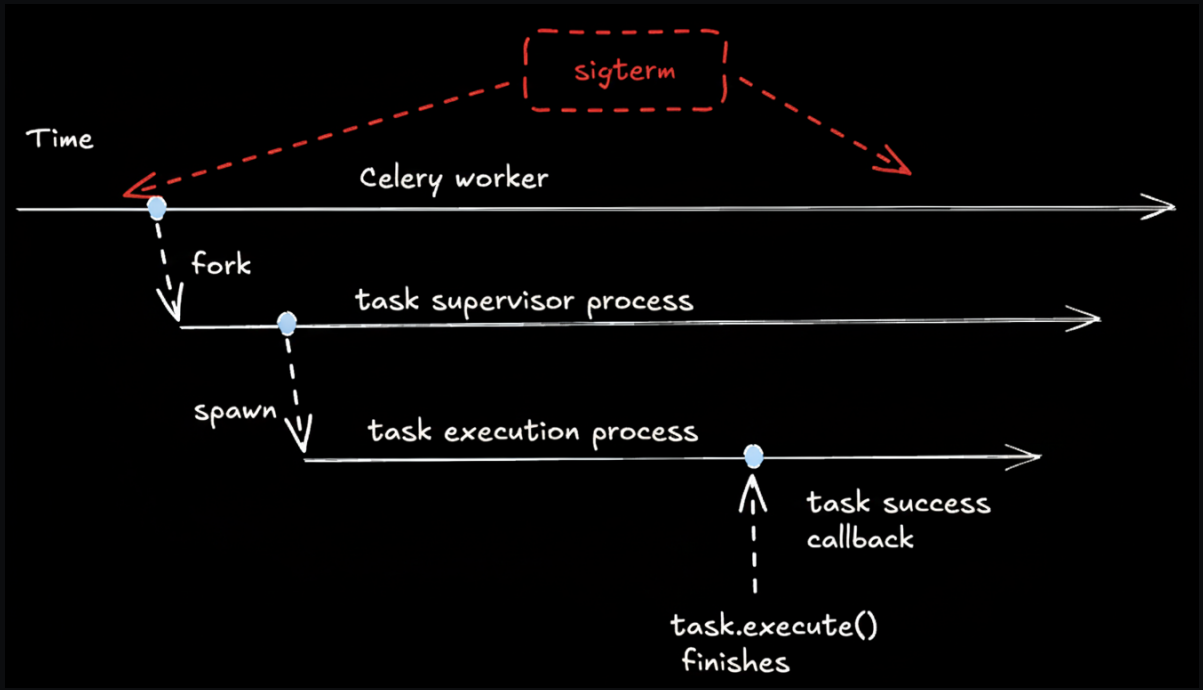
Problems:

- Sensor criteria met but task marked as failed
- State mismatch executor events, leading to retry or failed



Edge case - Pod can be killed anytime

Cause: pod received sigterm at different time



Edge case - Pod killed forcibly

Problems:

- Missing task execution logs led to confusion to users

Edge case - Pod killed forcibly

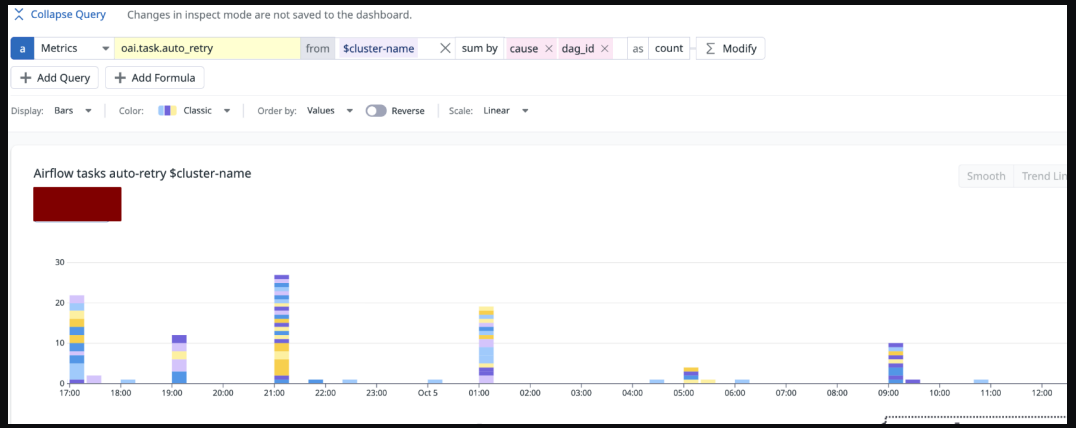
Cause:

- Celery worker will finish all currently executing tasks before it actually terminates
- Pod only waits terminationGracePeriodSeconds before being forcibly killed
  - Led to zombie tasks, airflow tasks killed via `-9`
  - Airflow log handler does not have chance to upload logs to remote storage

Reliability - Auto retry & proper preStop

Solution:

- Proper preStop to sigterm airflow supervisor processes on pod deletion
- Auto-infra-retry policy → detects infra exceptions and reschedules automatically
  - Sigterm to the pod
  - Executor events
  - Zombie tasks



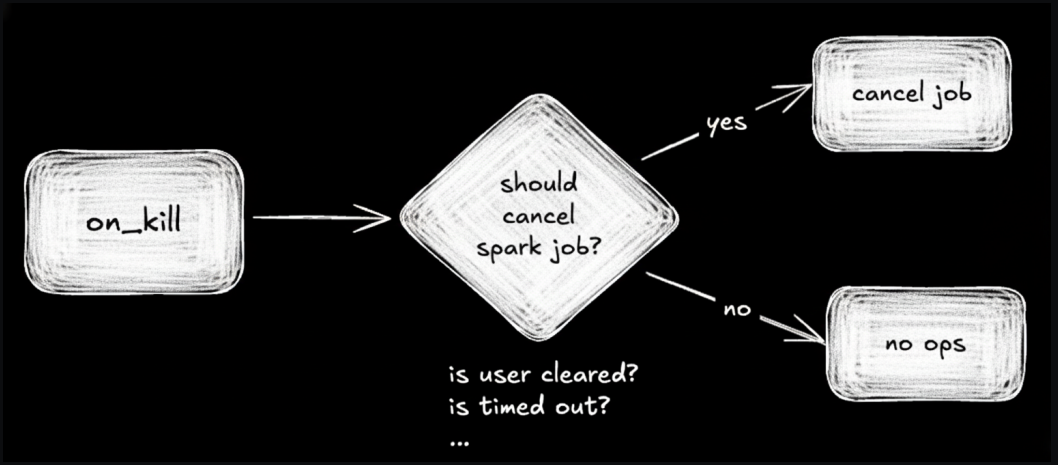
Reliability - Task idempotency key

Goal: Avoid unnecessary spark job rerun due to airflow task retries

Solution:

- Add airflow task idempotency key
- Differentiate user-cleared

  retry v.s. infra auto-retry

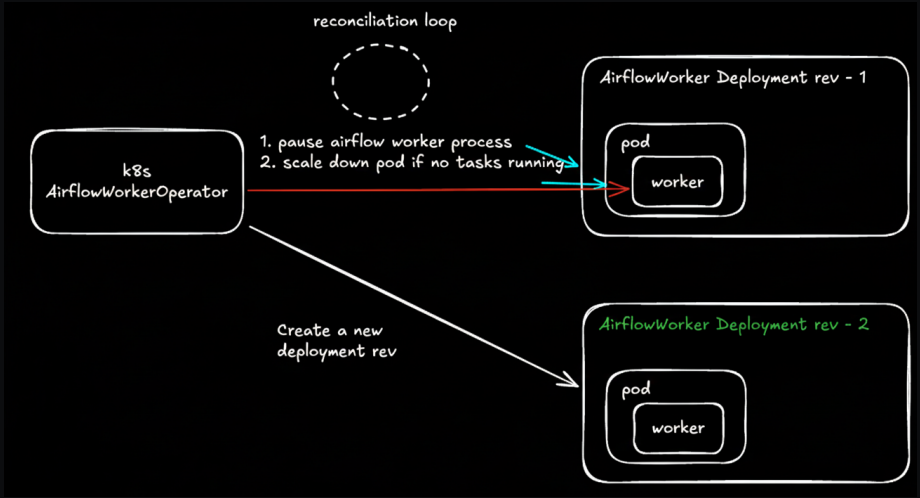


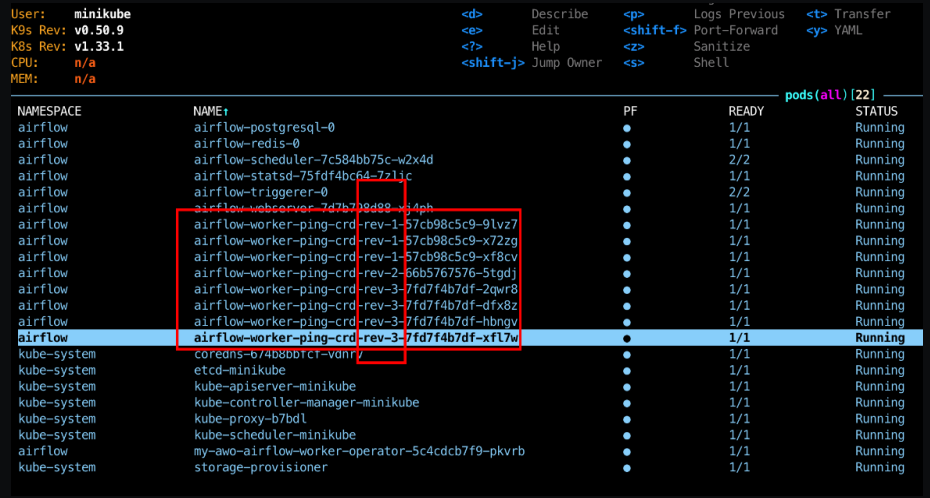
AirflowWorkerPool CRD

Problem: Worker pods get restarted during deployment, killing tasks

Solution:

- Custom AirflowWorkerPool CRD manages the lifecycle of each Airflow worker deployment generation





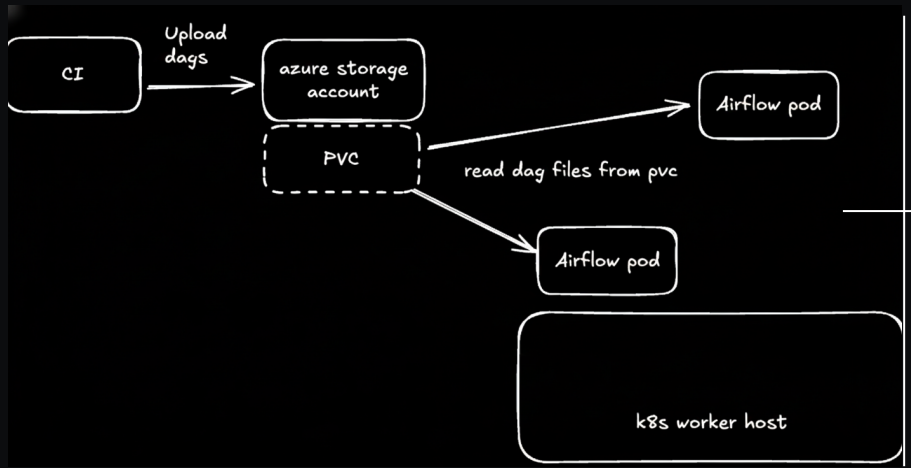
Bottlenecks define scalability

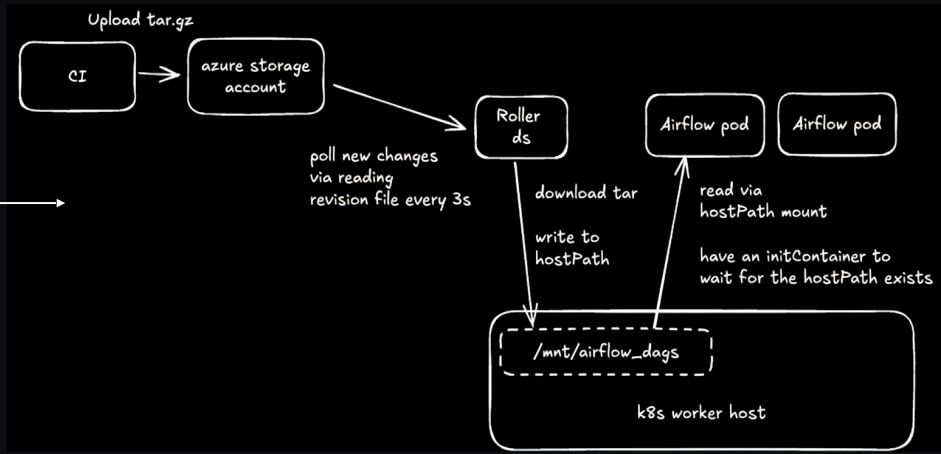
DagRoller DaemonSet

Problem: Slow DAG synchronization over remote storage

Solution:

- A dag-roller DaemonSet syncs dag files from storage account to hostPath
- Airflow pods access DAGs locally through host mounted SSD volumes





Multi-Cluster sharding

Problem: Scaling limit per cluster

Solution:

- Split workloads by use case
- Each cluster has dedicated metadata DB, redis, and k8s cluster
- Focus on building layers to manage many clusters

Toolings define efficiency

Self-Serve Tooling

Local task execution

- continue improving reliability and scheduler responsiveness to make local test results match production behavior

Custom backfill at scale

- add better queue control, visibility, and safety features as usage and workloads grow

Slack bot assistant

- smarter and more accurate by learning from real support threads and connecting more deeply to metadata

YAML-first DAGs

- expand coverage, simplify common cases, reduce boilerplate and generator improvements

Future Work

Reliable scheduler behavior

- continue reducing control plane latency and making infra issues invisible to users

Horizontal scalability

- expand multi-cluster support and worker pool management to serve more teams cleanly

Lower the learning curve

- Backfills are simple. Local tests are fast. Signals are clear. Self-serve

Feedback-driven platform

- strengthen the loop from support channel to Slack bot to infra and docs, so the system improves as usage grows

We Are Hiring

- https://openai.com/careers/engineering-manager-data-infrastructure/
- https://openai.com/careers/data-infrastructure-engineer/