Implementing Airflow Governance With Cluster Policies

3.0

Karen Braganza
Senior Airflow Reliability Engineer
@ Astronomer

Introduction to cluster policies

- Defined as Python functions
- Check or mutate Airflow objects.
- Enforce standards
- Useful for governance.

Types

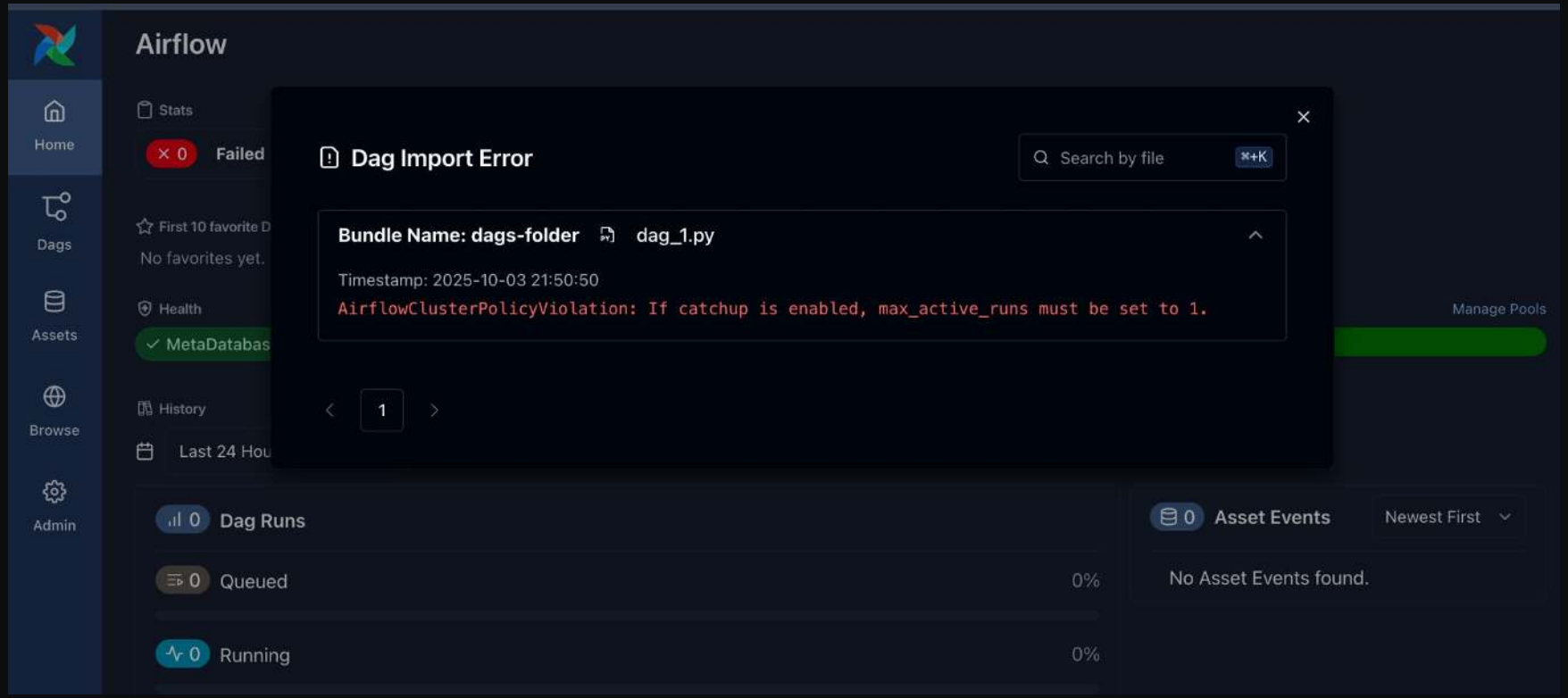
- dag policy
- task policy
- task instance mutation hook
- pod mutation hook

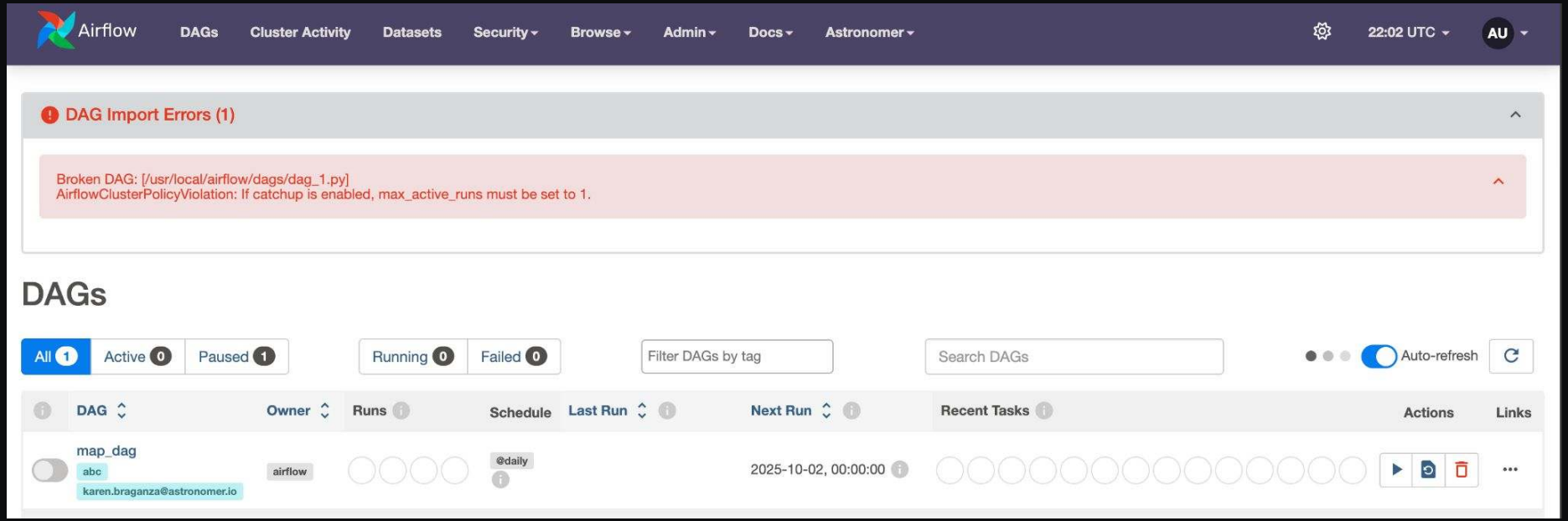
Dag Policies

- Applied to a dag object
- When it is loaded in the dagbag.
- Check and mutate parameters in the dag definition.

Governance Use Cases

- dag_id naming convention
- dag description, tags, owner_links
- dagrun_timeout


Airflow
Home
Dags
Assets
Browse
Admin
Stats
0
Failed
First 10 favorite D
No favorites yet.
Health
MetaDatabas
History
Last 24 Hou
Manage Pools
Dag Import Error
Search by file
⌘+K
Bundle Name: dags-folder
dag_1.py
Timestamp: 2025-10-03 21:50:50
AirflowClusterPolicyViolation: If catchup is enabled, max_active_runs must be set to 1.
1
0
Dag Runs
0
Queued
0%
0
Running
0%
0
Asset Events
Newest First
No Asset Events found.


Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
Astronomer
22:02 UTC
AU
DAG Import Errors (1)
Broken DAG: [/usr/local/airflow/dags/dag_1.py]
AirflowClusterPolicyViolation: If catchup is enabled, max_active_runs must be set to 1.
DAGs
All 1
Active 0
Paused 1
Running 0
Failed 0
Filter DAGs by tag
Search DAGs
Auto-refresh
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
Actions
Links
map_dag
abc
karen.braganza@astronomer.io
airflow
@daily
2025-10-02, 00:00:00

Disallow catchup with max_active_runs>1

```python
from airflow.exceptions import AirflowClusterPolicyViolation

def dag_policy(dag):
    allowed_dags = ["example_dag"]
    if dag.dag_id not in allowed_dags and dag.catchup and dag.max_active_runs!=1:
        raise AirflowClusterPolicyViolation("If catchup is enabled, max_active_runs must be set to 1.")
```

Require owner email in tags

```python
from airflow.exceptions import AirflowClusterPolicyViolation

def dag_policy(dag):
    email_found = False
    if not dag.tags:
        raise AirflowClusterPolicyViolation("Dag tags are missing.")
    for tag in dag.tags:
        if "@astronomer.io" in tag:
            email_found = True
    if not email_found:
        raise AirflowClusterPolicyViolation("At least one owner email is required.")
```

Task Policies

- Applied to a task object (BaseOperator)
- When it is loaded in the dagbag.
- Check and mutate task parameters.
- Applies to every task instance of the task object.

Governance Use Cases

- callbacks
- worker queues
- ban operators
- executor_config

Restrict usage of a specific connection

```python
from airflow.exceptions import AirflowClusterPolicyViolation

def task_policy(task: "BaseOperator") -> None:
    allowed_dags=["ex1_pg_authorized"]
    if task.dag.dag_id not in allowed_dags:
        for attribute in dir(task):
            if attribute.endswith("conn_id"):
                val = getattr(task, attribute)
                if val=="restricted_postgres":
                    raise AirflowClusterPolicyViolation("Use of restricted_postgres connection is not authorized.")
```

Run KPO & deferrable tasks on a lightweight queue

```python
def task_policy(task: "BaseOperator") -> None:
    cls_path = task.__class__.__module__ + "." + task.__class__.__name__
    kpo_path = "airflow.providers.cncf.kubernetes.operators.pod.KubernetesPodOperator"
    is_deferrable = getattr(task, "deferrable", None)
    if cls_path==kpo_path or is_deferrable:
        task.queue="lightweight-worker-queue"
```

Task Instance Mutation Hook

- Applied to a task instance object
- When the TI is initialized on the scheduler
- Again after the TI has landed on a worker.
- Selectively applied to some task instances of a task.

Governance Use Cases

- Mutate task instances between:
  - tries
  - map indices
  - runs

Remove success callbacks for manual & backfill runs

```
def task_instance_mutation_hook(task_instance):
    if isinstance(task_instance.run_id, str):
        if "manual" in task_instance.run_id or "backfill" in task_instance.run_id:
            task_instance.task.on_success_callback=None
```

Assign map indices>5 to a separate queue

```
def task_instance_mutation_hook(task_instance):
    if task_instance.map_index>5:
        task_instance.queue="spare-high-resource-queue"
```

This does not work!

Run manual and backfill dags on a specific queue

```
def task_instance_mutation_hook(task_instance):
    if "manual" in task_instance.run_id or "backfill" in task_instance.run_id:
        task_instance.queue="manual-and-backfill-runs-queue"
```

TypeError: argument of type 'NoneType' is not iterable

Add retry callback only before the final retry

```python
from airflow.providers.smtp.notifications.smtp import SmtpNotifier

def task_instance_mutation_hook(task_instance):
    if isinstance(task_instance.try_number, int) and isinstance(task_instance.max_tries, int):
        if task_instance.try_number==task_instance.max_tries:
            task_instance.task.on_retry_callback=SmtpNotifier(to="karen.braganza@astronomer.io", subject="The TI is about to run its final retry!")
```

Pod Mutation Hook

- Applied to a Kubernetes pod created using the KubernetesExecutor or KubernetesPodOperator.
- Run when building the pod.
- Examine or mutate a kubernetes.client.models.V1Pod object.

Governance Use Cases

- sidecar containers
- NodeSelector
- annotations
- default pod resources.
- termination grace period

Add tolerations to KE worker pod

```python
def pod_mutation_hook(pod: V1Pod) -> None:
    from kubernetes.client import V1Toleration

    pod.spec.tolerations = [
        V1Toleration(
            key="node-group",
            operator="Equal",
            value="airflow-worker",
            effect="NoSchedule",
        )
    ]
```

Overall Considerations

Cluster policy mutations lack user visibility.

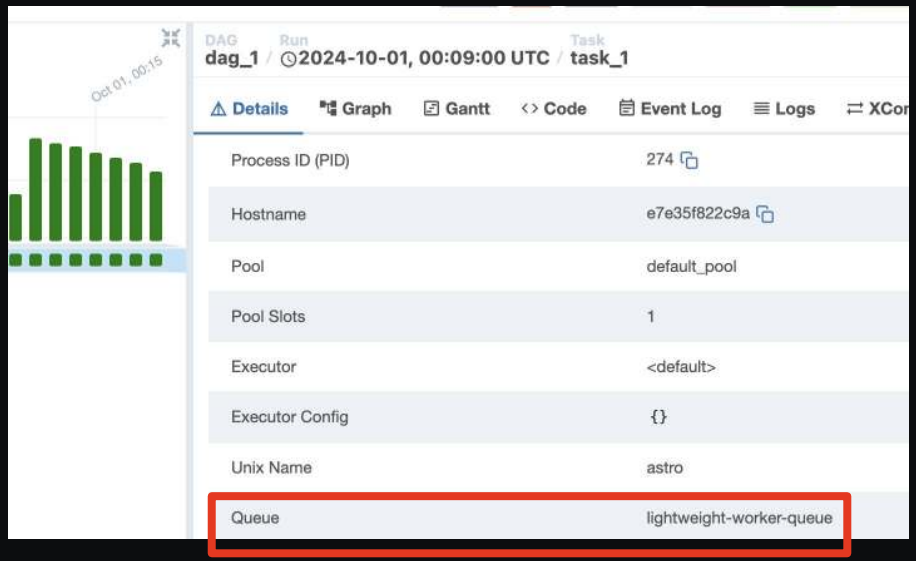
```python
from datetime import datetime
from airflow.decorators import dag, task

@dag(start_date=datetime(2024, 10, 1), schedule="* * * * *")
def dag_1():

    @task(queue="default-queue")
    def task_1():
        print("This is task 1!")

    task_1()

dag_1()
```



Overall Considerations

Cluster policies can impact critical Airflow components like the scheduler!

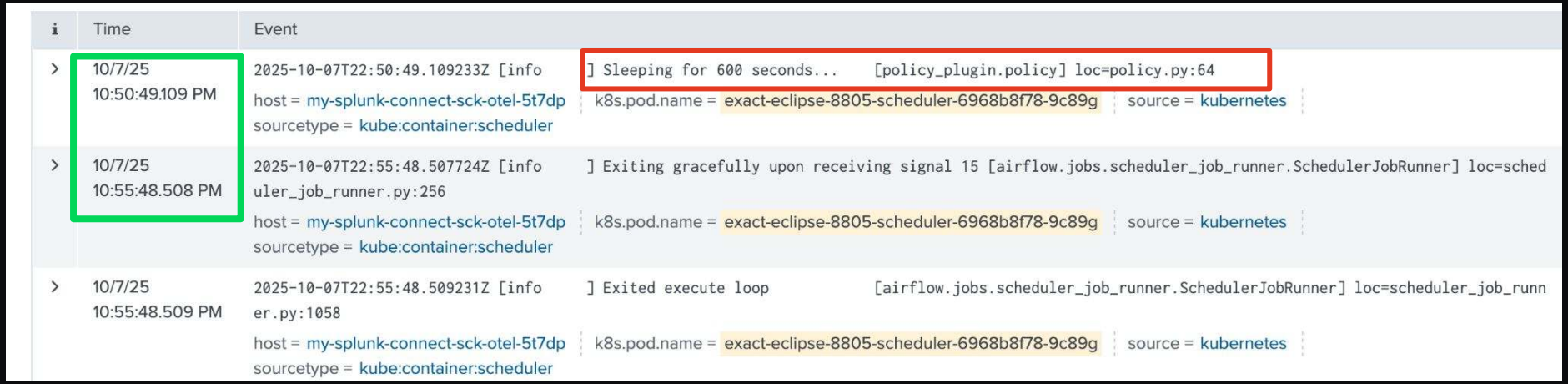
```python
import time
import logging

logger = logging.getLogger(__name__)


def task_instance_mutation_hook(task_instance):
    logger.info("Sleeping for 600 seconds...")
    time.sleep(600)
```

Overall Considerations

Cluster policies can impact critical Airflow components like the scheduler!



i	Time	Event
>	10/7/25 10:50:49.109 PM	2025-10-07T22:50:49.109233Z [info] Sleeping for 600 seconds... [policy_plugin.policy] loc=policy.py:64 host = my-splunk-connect-sck-otel-5t7dp k8s.pod.name = exact-eclipse-8805-scheduler-6968b8f78-9c89g source = kubernetes sourcetype = kube:container:scheduler
>	10/7/25 10:55:48.508 PM	2025-10-07T22:55:48.507724Z [info] Exiting gracefully upon receiving signal 15 [airflow.jobs.scheduler_job_runner.SchedulerJobRunner] loc=scheduler_job_runner.py:256 host = my-splunk-connect-sck-otel-5t7dp k8s.pod.name = exact-eclipse-8805-scheduler-6968b8f78-9c89g source = kubernetes sourcetype = kube:container:scheduler
>	10/7/25 10:55:48.509 PM	2025-10-07T22:55:48.509231Z [info] Exited execute loop [airflow.jobs.scheduler_job_runner.SchedulerJobRunner] loc=scheduler_job_runner.py:1058 host = my-splunk-connect-sck-otel-5t7dp k8s.pod.name = exact-eclipse-8805-scheduler-6968b8f78-9c89g source = kubernetes sourcetype = kube:container:scheduler

Implementation

- airflow_local_settings.py
- custom module with pluggy (using a setuptools entrypoint)

Questions?

Karen Braganza
Senior Airflow Reliability Engineer
at Astronomer

AIRFLOW SUMMIT

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

References

- OSS Airflow docs
- Astronomer docs
- OSS Airflow GitHub code