Introduction to Airflow

INTRODUCTION TO AIRFLOW IN PYTHON

Mike Metzger
Data Engineer

What is data engineering?

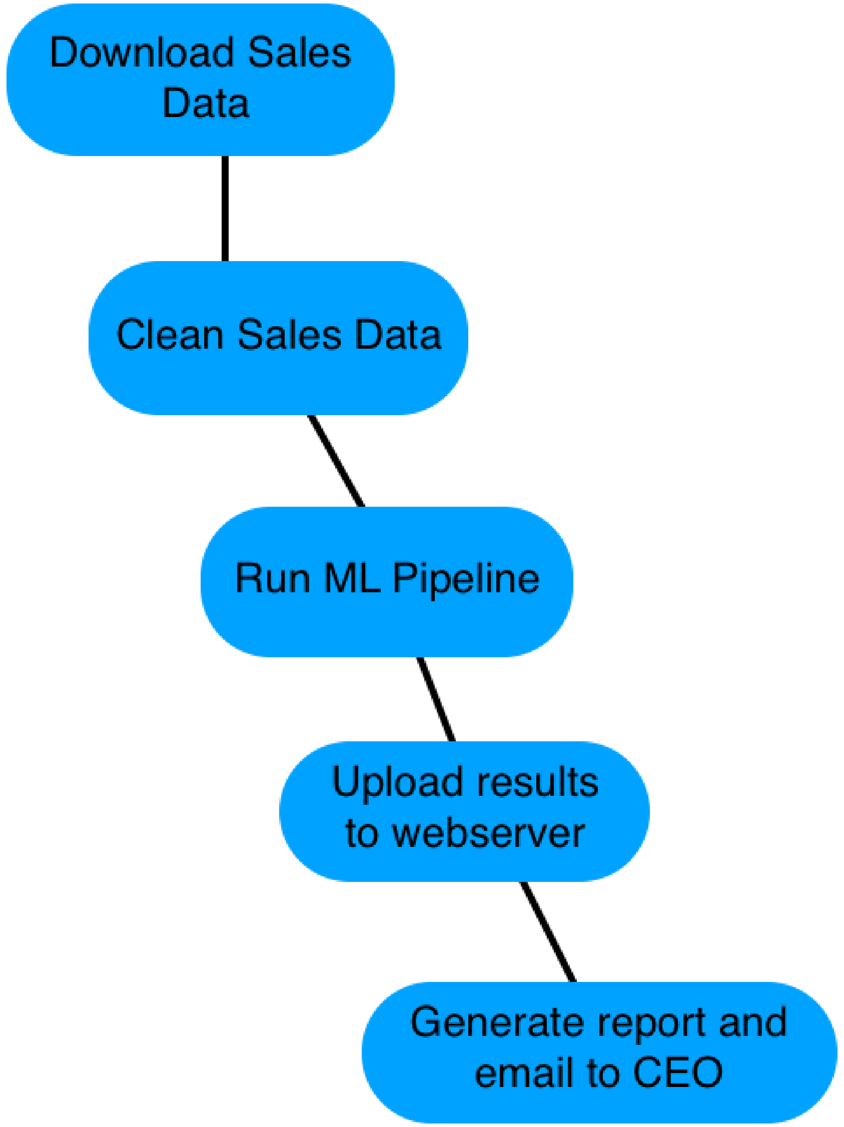
Data engineering is:

- Taking any action involving data and turning it into a reliable, repeatable, and maintainable process.

What is a workflow?

A *workflow* is:

- A set of steps to accomplish a given data engineering task
  - Such as: downloading files, copying data, filtering information, writing to a database, etc
- Of varying levels of complexity
- A term with various meaning depending on context



What is Airflow?

Airflow is a platform to program workflows, including:

- Creation
- Scheduling
- Monitoring

Airflow continued...

- Can implement programs from any language, but workflows are written in Python
- Implements workflows as DAGs: Directed Acyclic Graphs
- Accessed via code, command-line, or via web interface / REST API

[1] https://airflow.apache.org/docs/stable/

Other workflow tools

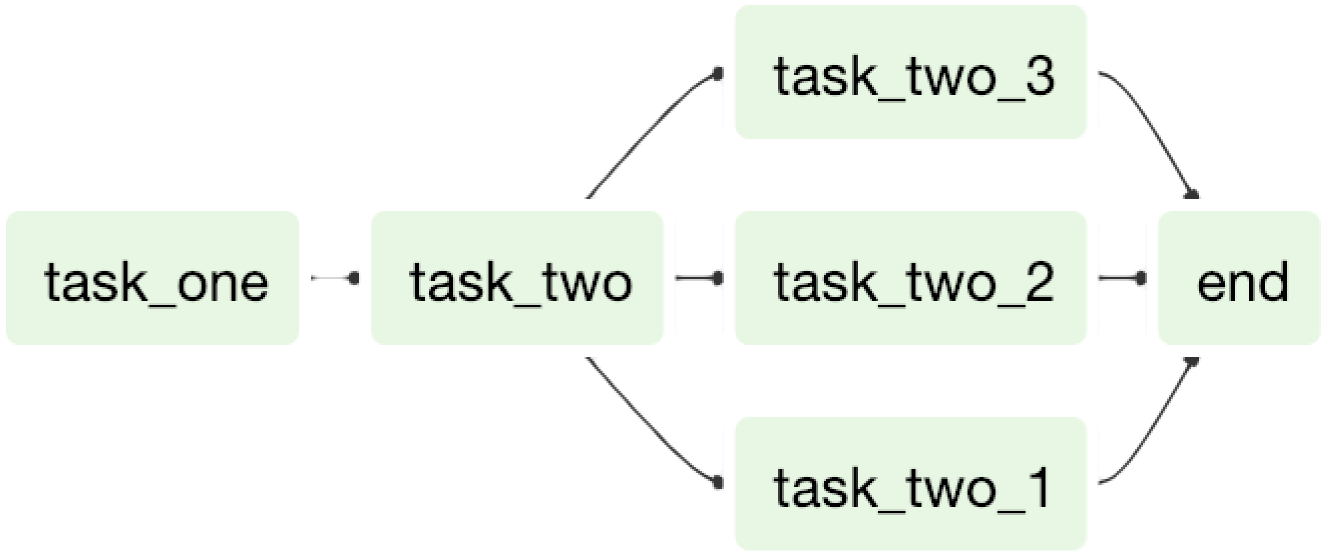
Other tools:

- Luigi
- SSIS
- Bash scripting

Quick introduction to DAGs

A *DAG* stands for *Directed Acyclic Graph*

- In Airflow, this represents the set of tasks that make up your workflow.
- Consists of the tasks and the dependencies between tasks.
- Created with various details about the DAG, including the name, start date, owner, etc.
- Further depth in the next lesson.



DAG code example

Simple DAG definition:

```
etl_dag = DAG(
    dag_id='etl_pipeline',
    default_args={"start_date": "2024-01-08"}
)
```

Running a workflow in Airflow

Running a simple Airflow task

```
airflow tasks test <dag_id> <task_id> [execution_date]
```

Using a DAG named *example-etl*, a task named *download-file* on 2024-01-10:

```
airflow tasks test example-etl download-file 2024-01-10
```

Let's practice!

INTRODUCTION TO AIRFLOW IN PYTHON

Airflow DAGs

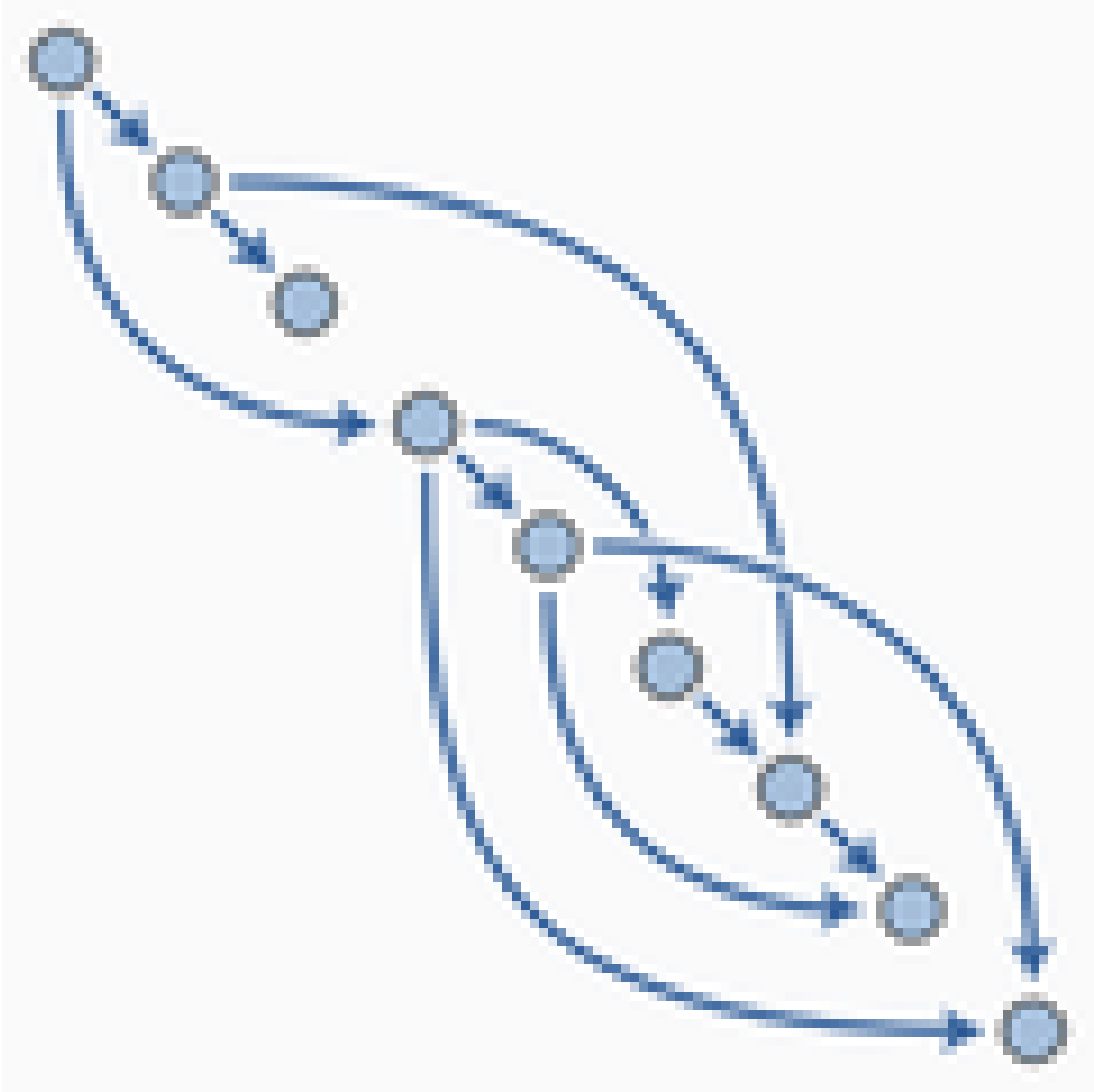
INTRODUCTION TO AIRFLOW IN PYTHON

Mike Metzger
Data Engineer

What is a DAG?

DAG, or *Directed Acyclic Graph*:

- *Directed*, there is an inherent flow representing dependencies between components.
- *Acyclic*, does not loop / cycle / repeat.
- *Graph*, the actual set of components.
- Seen in Airflow, Apache Spark, dbt



[1] https://en.m.wikipedia.org/wiki/Directed_acyclic_graph

DAG in Airflow

Within Airflow, DAGs:

- Are written in Python (but can use components written in other languages).
- Are made up of components (typically *tasks*) to be executed, such as operators, sensors, etc.
- Contain dependencies defined explicitly or implicitly.
  - ie, Copy the file to the server before trying to import it to the database service.

Define a DAG

Example DAG:

```
from airflow import DAG

from datetime import datetime
default_arguments = {
  'owner': 'jdoe',
  'email': 'jdoe@datacamp.com',
  'start_date': datetime(2020, 1, 20)
}

with DAG('etl_workflow', default_args=default_arguments ) as etl_dag:
```

Define a DAG (before Airflow 2.x)

Example DAG:

```python
from airflow import DAG

from datetime import datetime
default_arguments = {
  'owner': 'jdoe',
  'email': 'jdoe@datacamp.com',
  'start_date': datetime(2020, 1, 20)
}

etl_dag = DAG('etl_workflow', default_args=default_arguments )
```

DAGs on the command line

Using `airflow`:

- The `airflow` command line program contains many subcommands.
- `airflow -h` for descriptions.
- Many are related to DAGs.
- `airflow dags list` to show all recognized DAGs.

Command line vs Python

Use the command line tool to:

- Start Airflow processes
- Manually run DAGs / Tasks
- Get logging information from Airflow

Use Python to:

- Create a DAG
- Edit the individual properties of a DAG

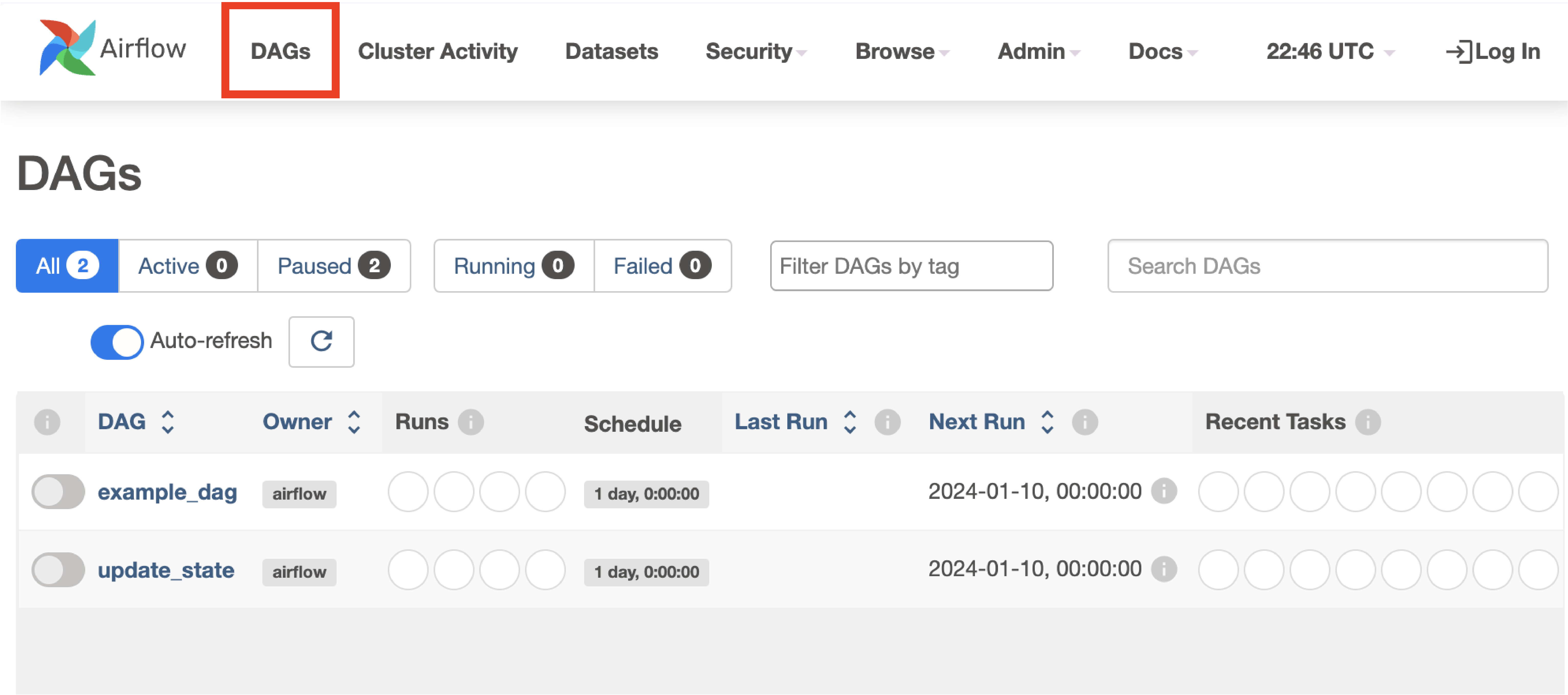
Let's practice!

Airflow web interface

INTRODUCTION TO AIRFLOW IN PYTHON

Mike Metzger
Data Engineer

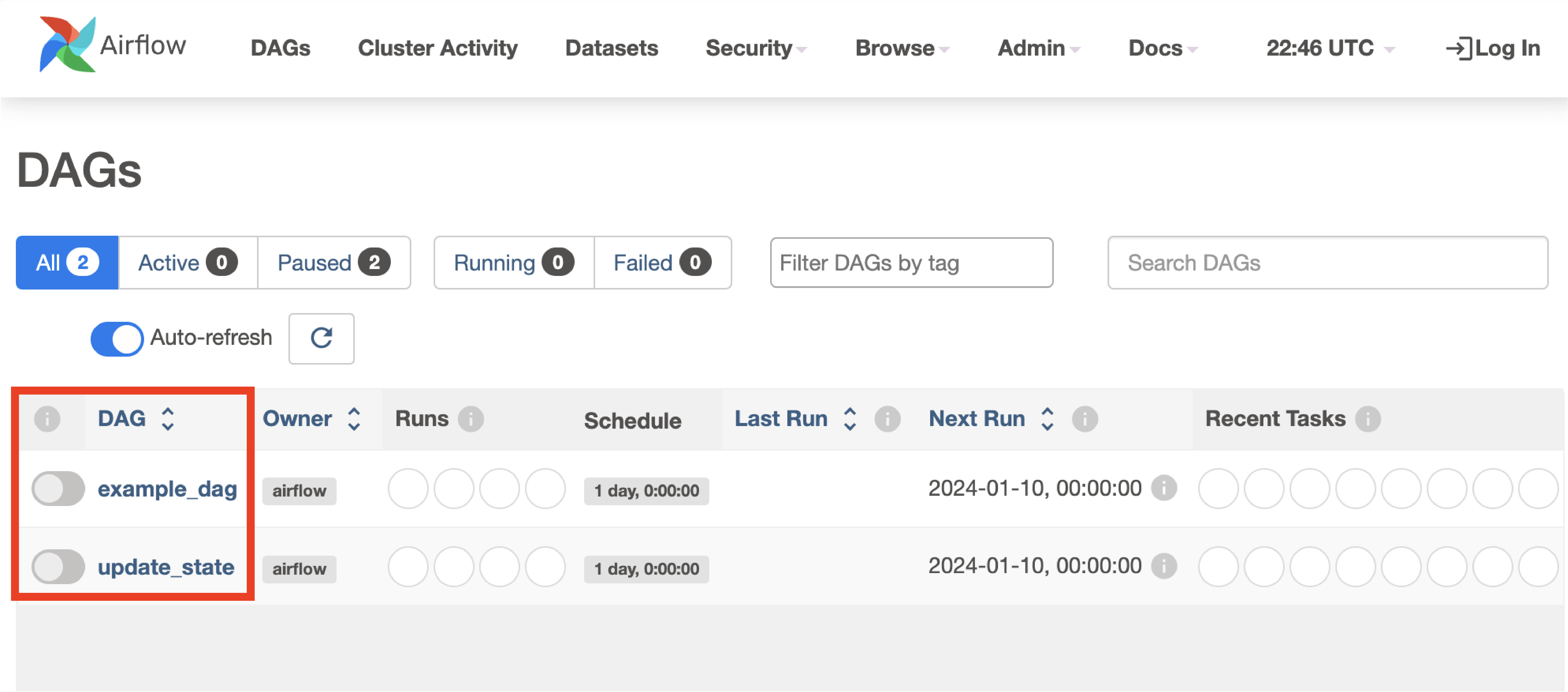
DAGs view



DAGs

	DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
	example_dag	airflow	○ ○ ○ ○	1 day, 0:00:00		2024-01-10, 00:00:00	○ ○ ○ ○ ○ ○ ○ ○
	update_state	airflow	○ ○ ○ ○	1 day, 0:00:00		2024-01-10, 00:00:00	○ ○ ○ ○ ○ ○ ○ ○

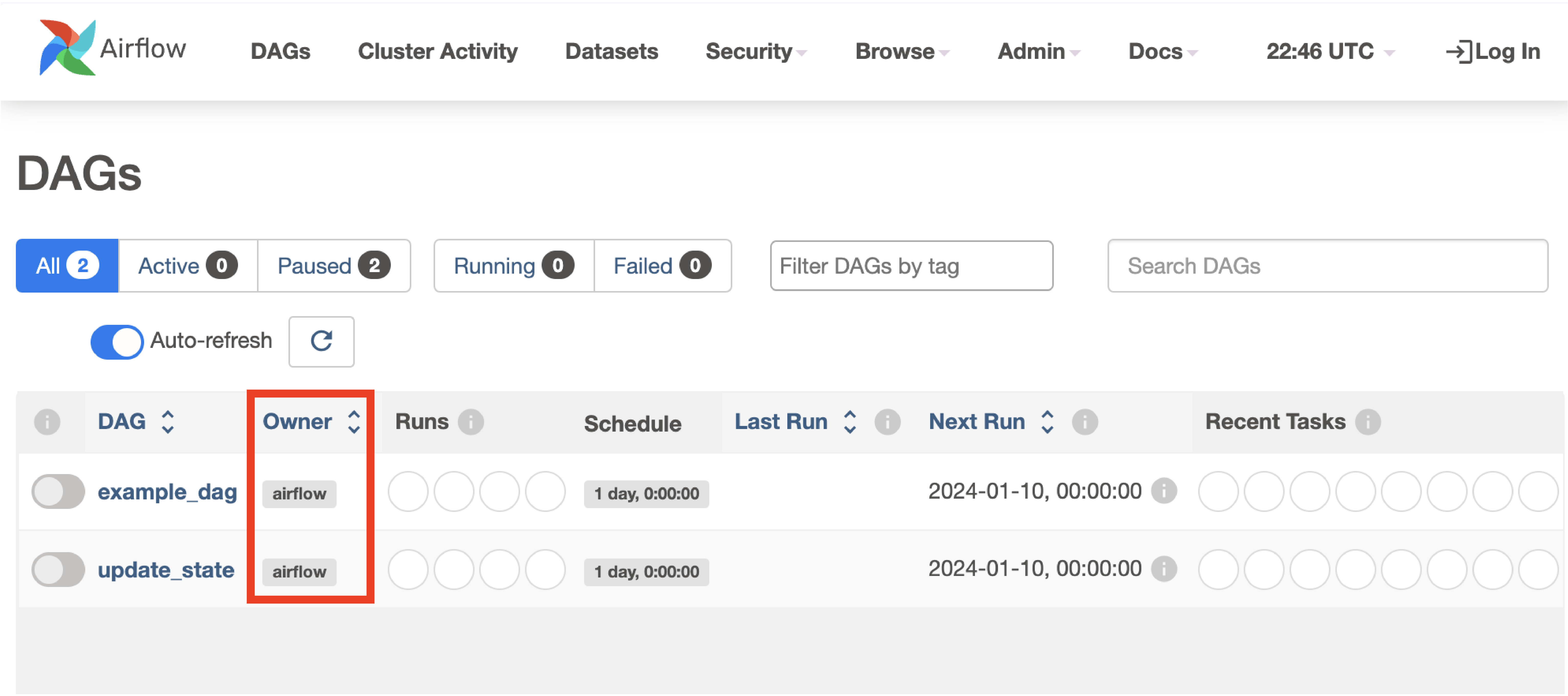
DAGs view DAGs



DAGs

	DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
	example_dag	airflow		1 day, 0:00:00		2024-01-10, 00:00:00	
	update_state	airflow		1 day, 0:00:00		2024-01-10, 00:00:00	

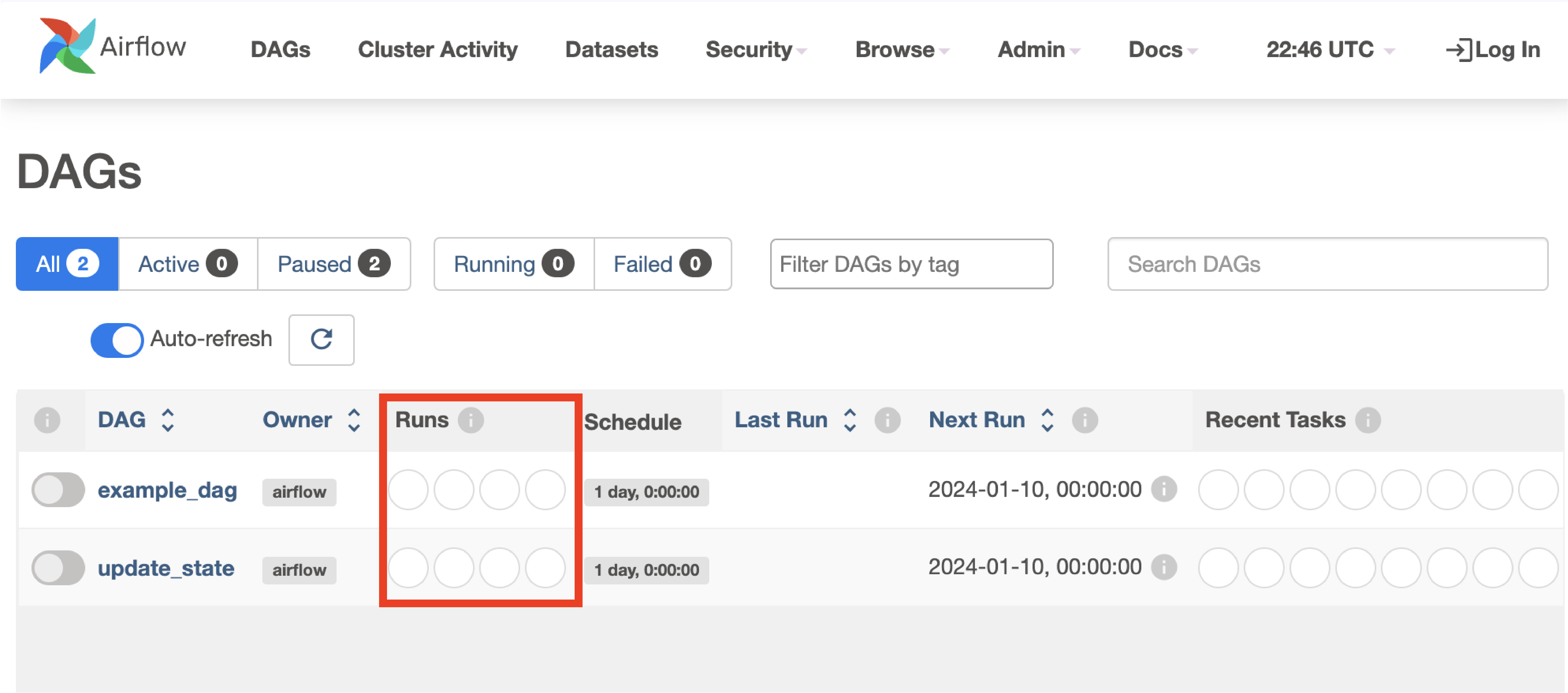
DAGs view owner



DAGs

	DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
	example_dag	airflow		1 day, 0:00:00		2024-01-10, 00:00:00	
	update_state	airflow		1 day, 0:00:00		2024-01-10, 00:00:00	

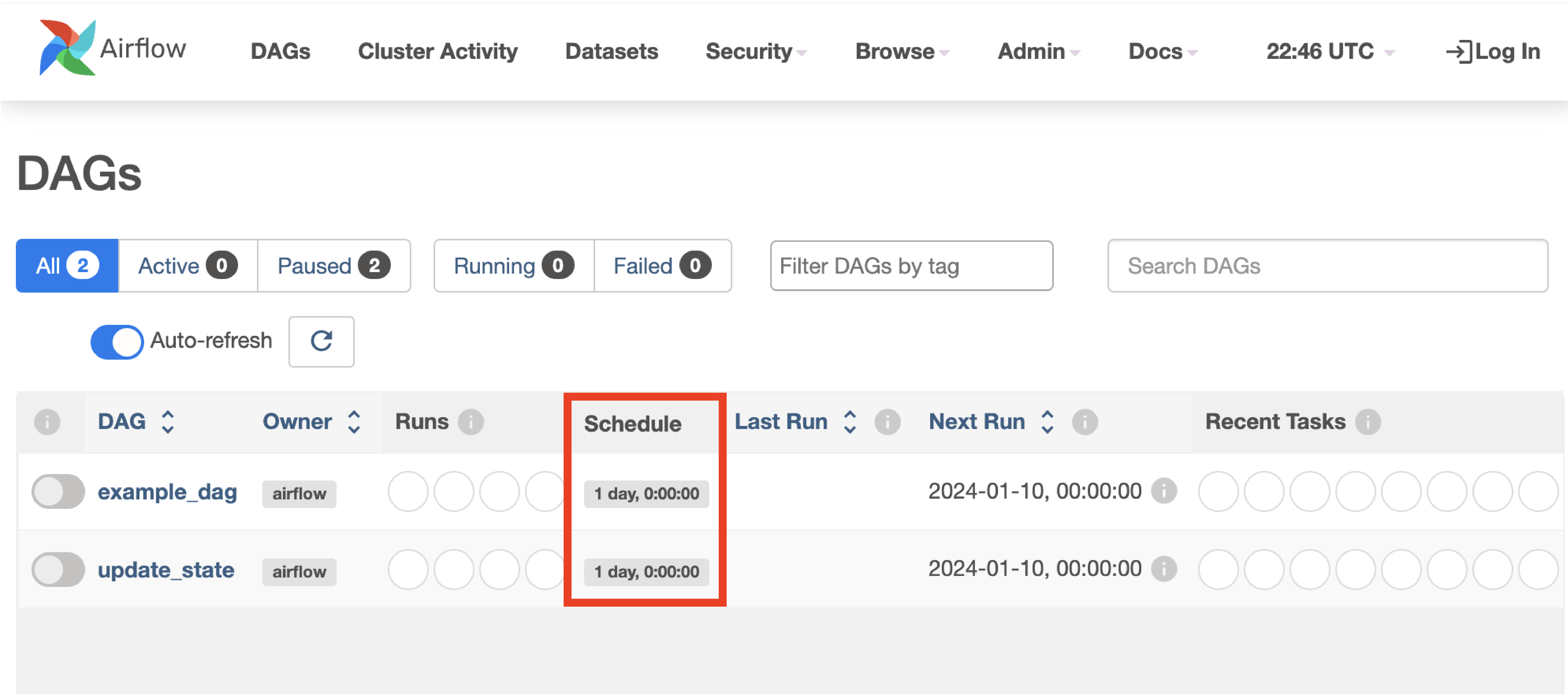
DAGs view runs



DAGs

	DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
	example_dag	airflow	○○○○	1 day, 0:00:00		2024-01-10, 00:00:00	○○○○○○○○
	update_state	airflow	○○○○	1 day, 0:00:00		2024-01-10, 00:00:00	○○○○○○○○

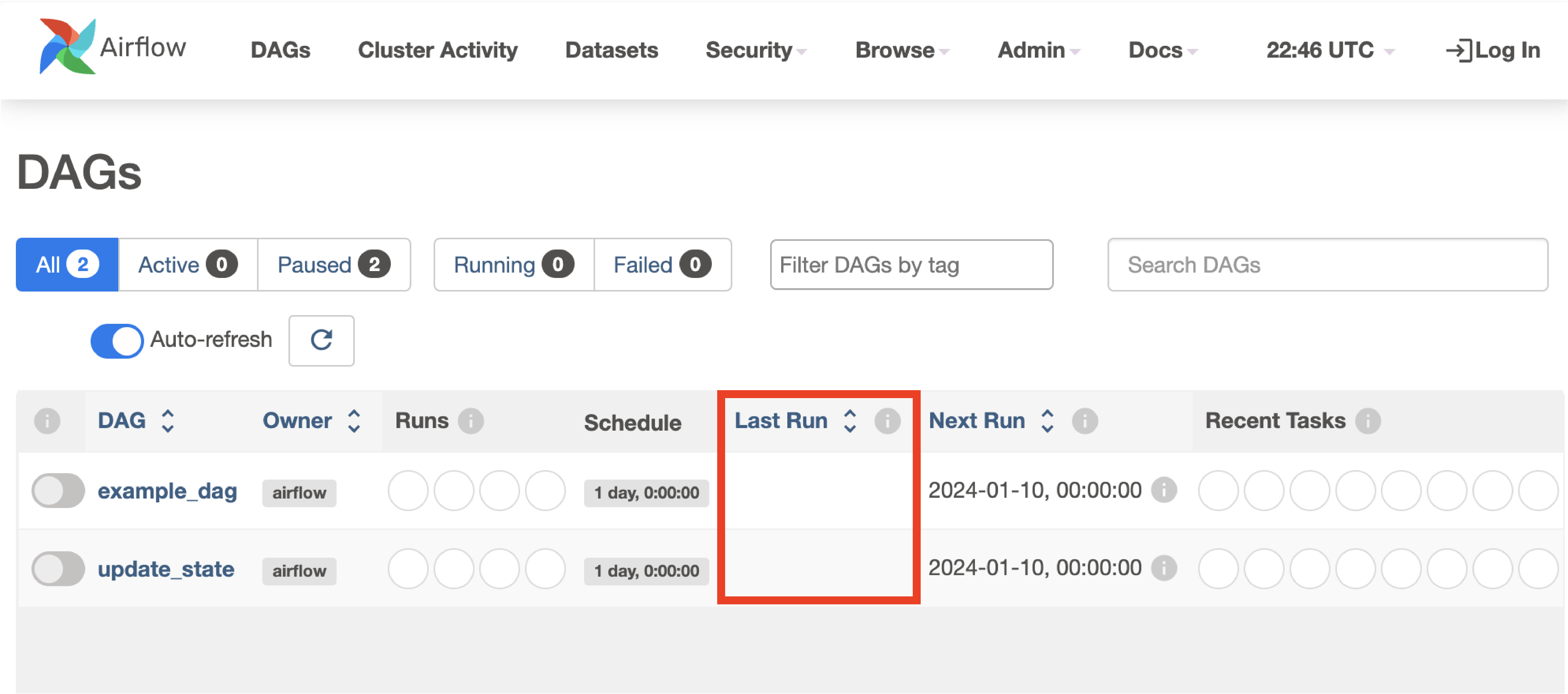
DAGs view schedule



DAGs

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
example_dag	airflow		1 day, 0:00:00		2024-01-10, 00:00:00	
update_state	airflow		1 day, 0:00:00		2024-01-10, 00:00:00	

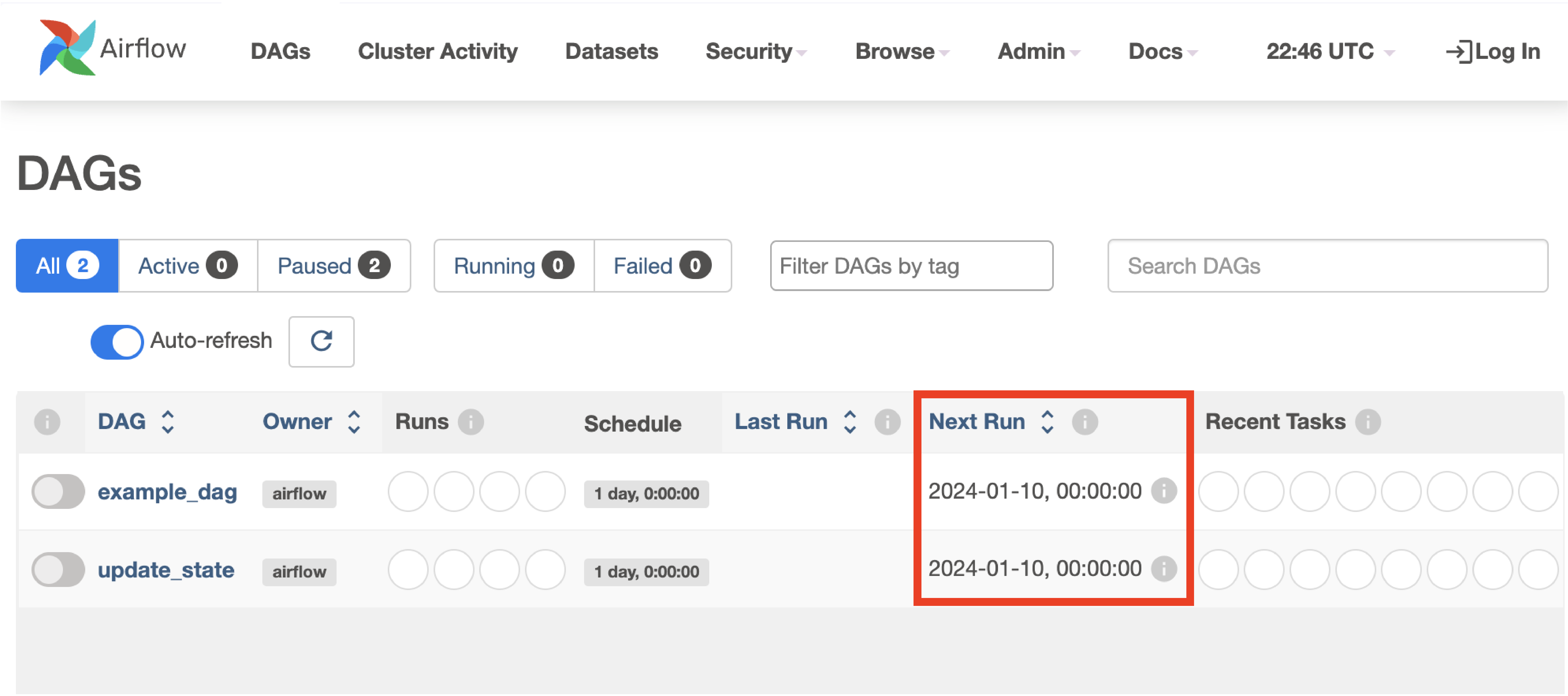
DAGs view last run



DAGs

	DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
	example_dag	airflow		1 day, 0:00:00		2024-01-10, 00:00:00	
	update_state	airflow		1 day, 0:00:00		2024-01-10, 00:00:00	

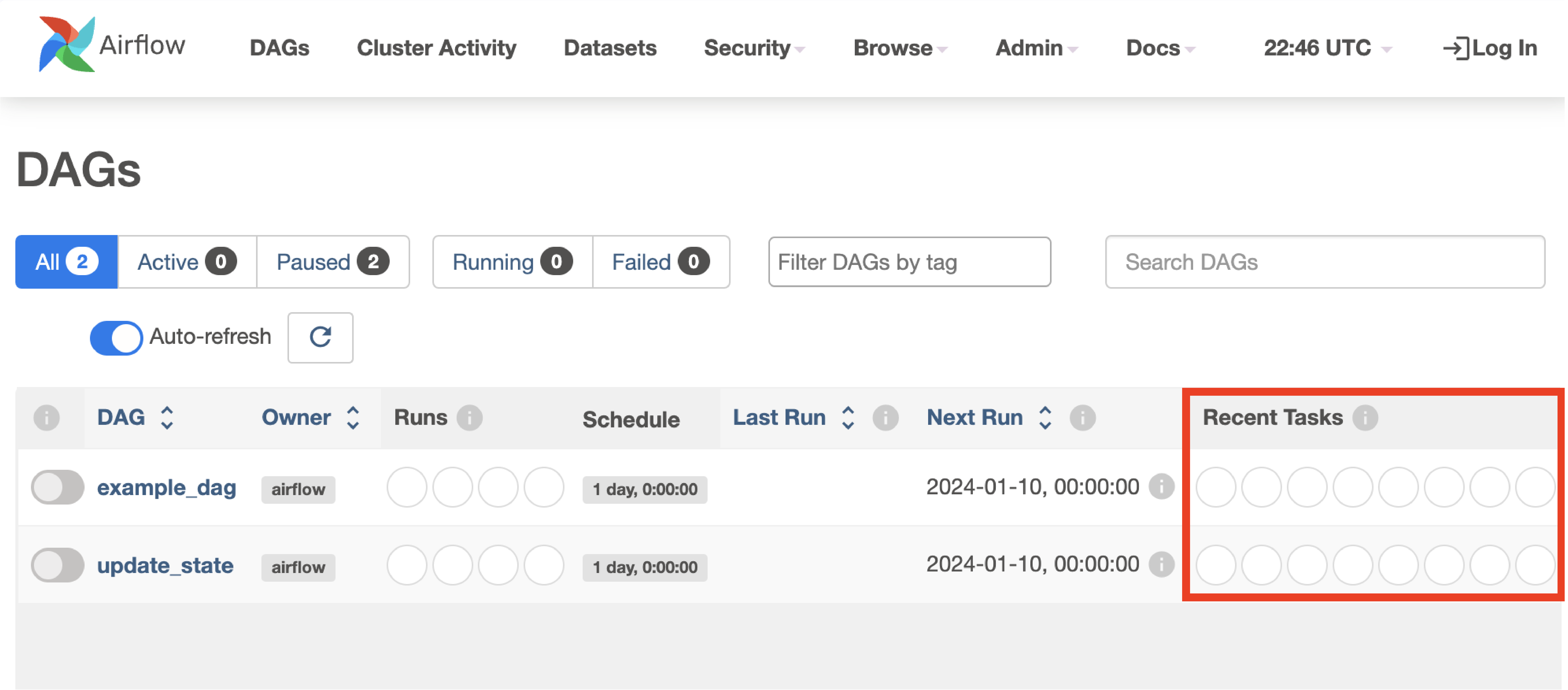
DAGs view next run



DAGs

	DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
	example_dag	airflow		1 day, 0:00:00		2024-01-10, 00:00:00	
	update_state	airflow		1 day, 0:00:00		2024-01-10, 00:00:00	

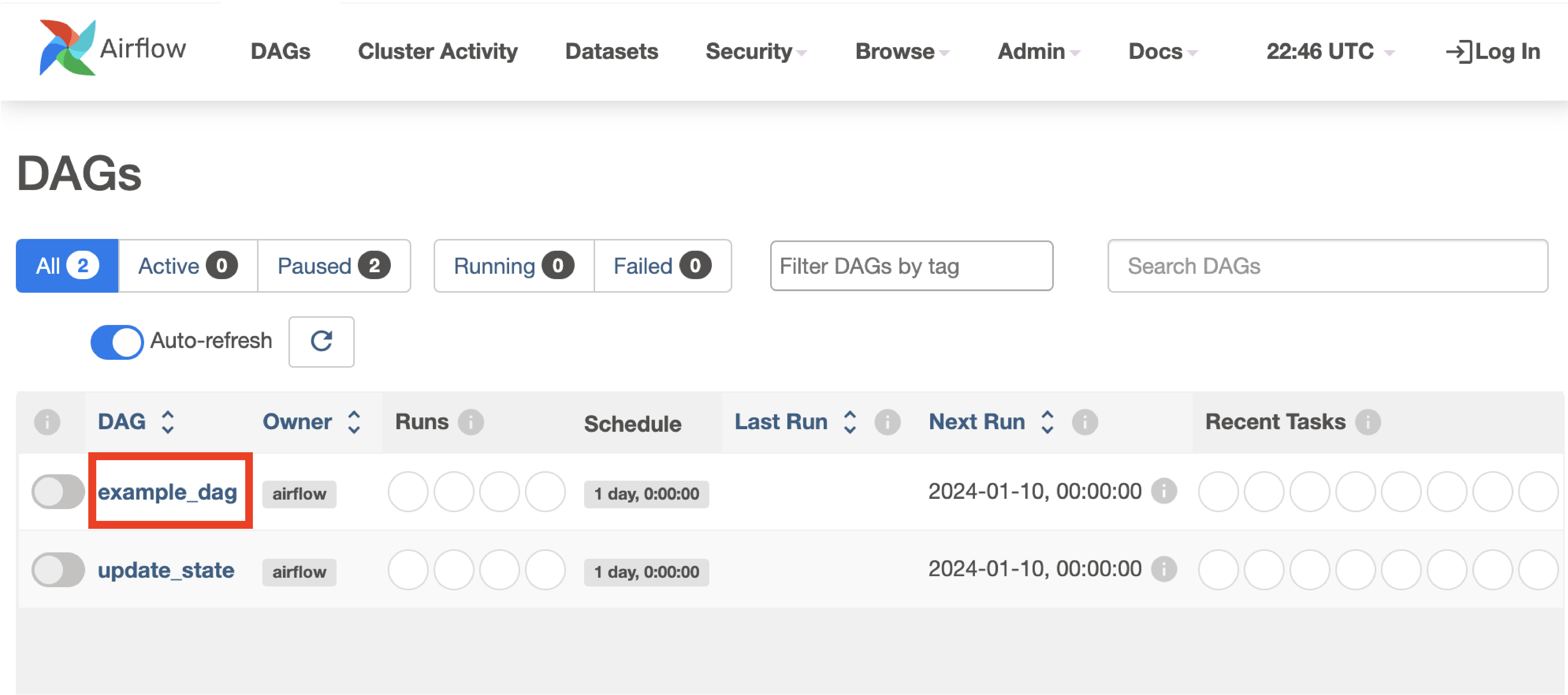
DAGs view recent tasks



DAGs

	DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
	example_dag	airflow	○○○○	1 day, 0:00:00		2024-01-10, 00:00:00	○○○○○○○○
	update_state	airflow	○○○○	1 day, 0:00:00		2024-01-10, 00:00:00	○○○○○○○○

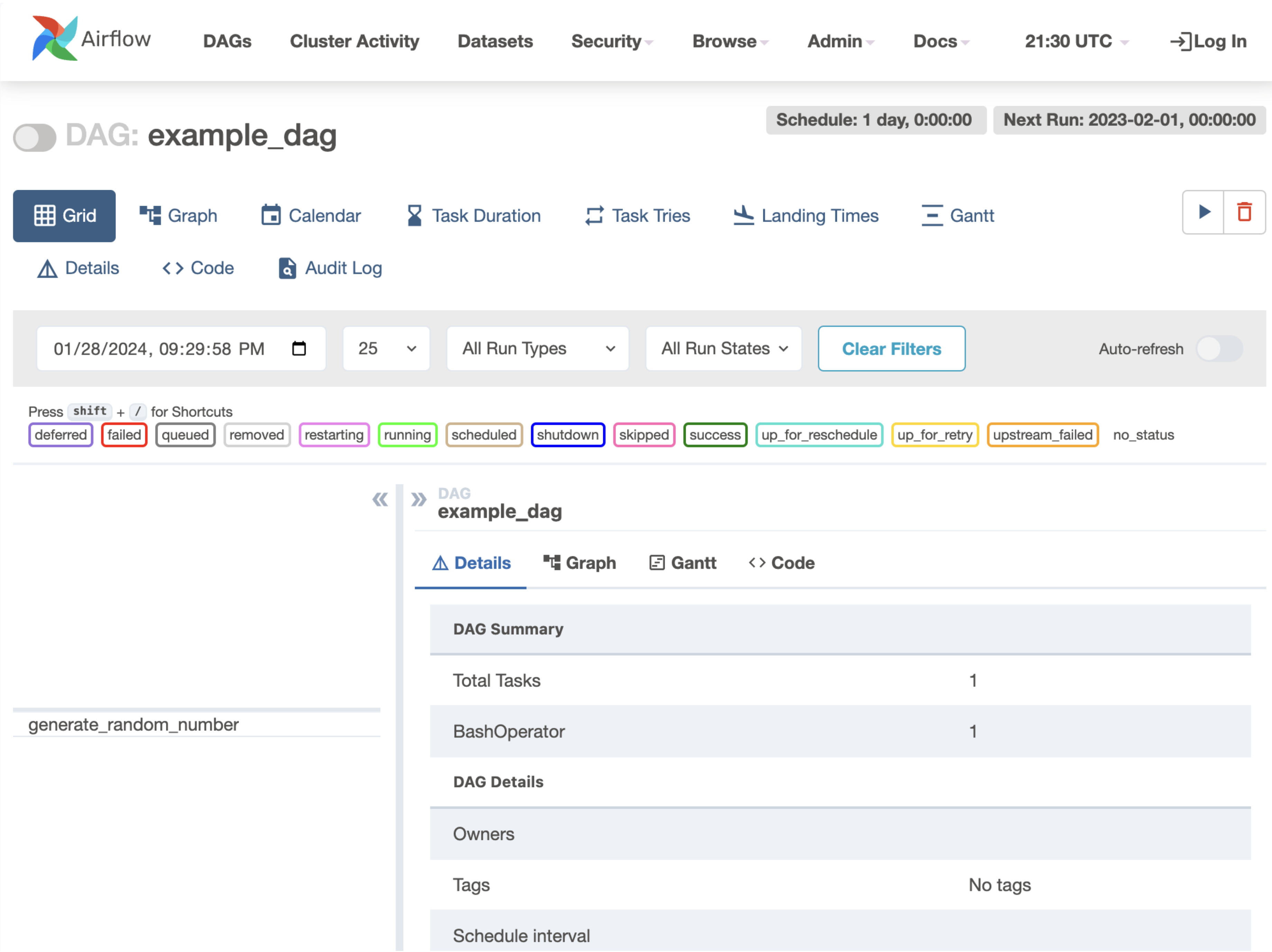
DAGs view example_dag



DAGs

	DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
	example_dag	airflow	○ ○ ○ ○	1 day, 0:00:00		2024-01-10, 00:00:00	○ ○ ○ ○ ○ ○ ○ ○
	update_state	airflow	○ ○ ○ ○	1 day, 0:00:00		2024-01-10, 00:00:00	○ ○ ○ ○ ○ ○ ○ ○

DAG detail view



DAG: example_dag

Schedule: 1 day, 0:00:00

Next Run: 2023-02-01, 00:00:00

01/28/2024, 09:29:58 PM | 25 | All Run Types | All Run States

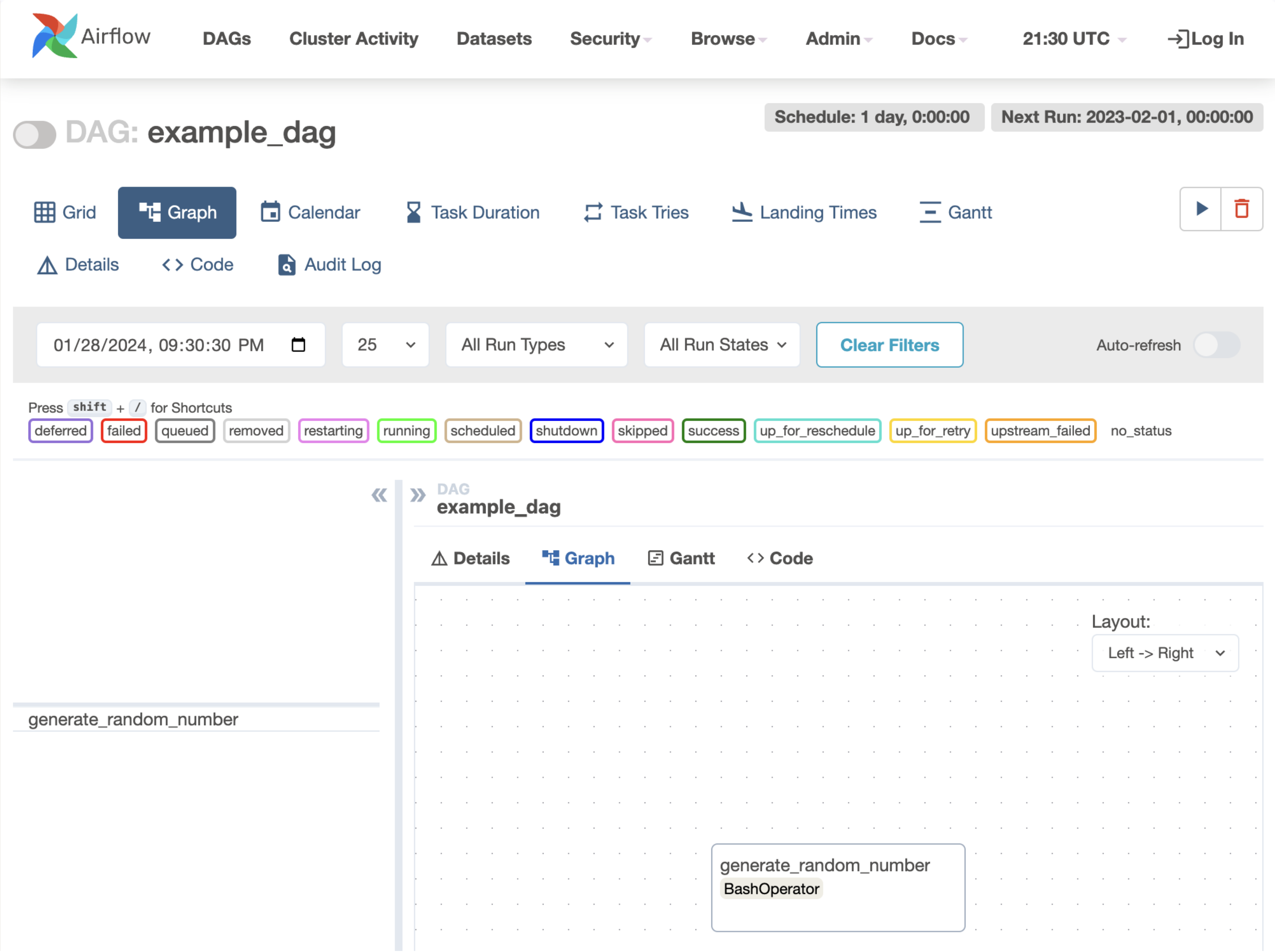
generate_random_number

DAG

example_dag

DAG Summary	
Total Tasks	1
BashOperator	1
DAG Details	
Owners	
Tags	No tags
Schedule interval	

DAG graph view



DAG: example_dag

Schedule: 1 day, 0:00:00

Next Run: 2023-02-01, 00:00:00

01/28/2024, 09:30:30 PM

25

All Run Types

All Run States

Press shift + / for Shortcuts

deferred failed queued removed restarting running scheduled shutdown skipped success up_for_reschedule up_for_retry upstream_failed no_status

DAG

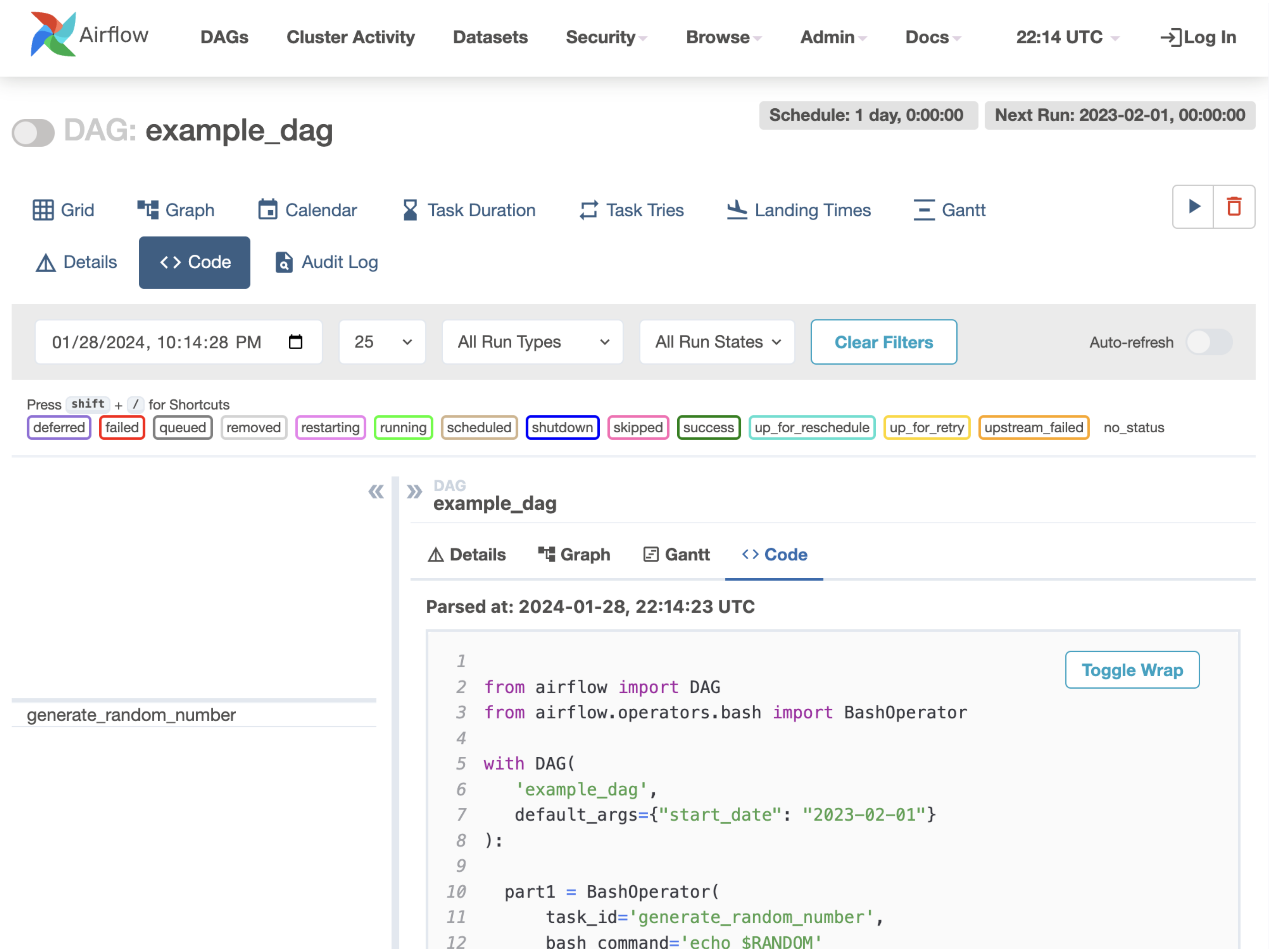
example_dag

Layout: Left -> Right

generate_random_number

generate_random_number
BashOperator

DAG code view



DAG: example_dag

Schedule: 1 day, 0:00:00

Next Run: 2023-02-01, 00:00:00

01/28/2024, 10:14:28 PM

25

DAG
example_dag

Parsed at: 2024-01-28, 22:14:23 UTC

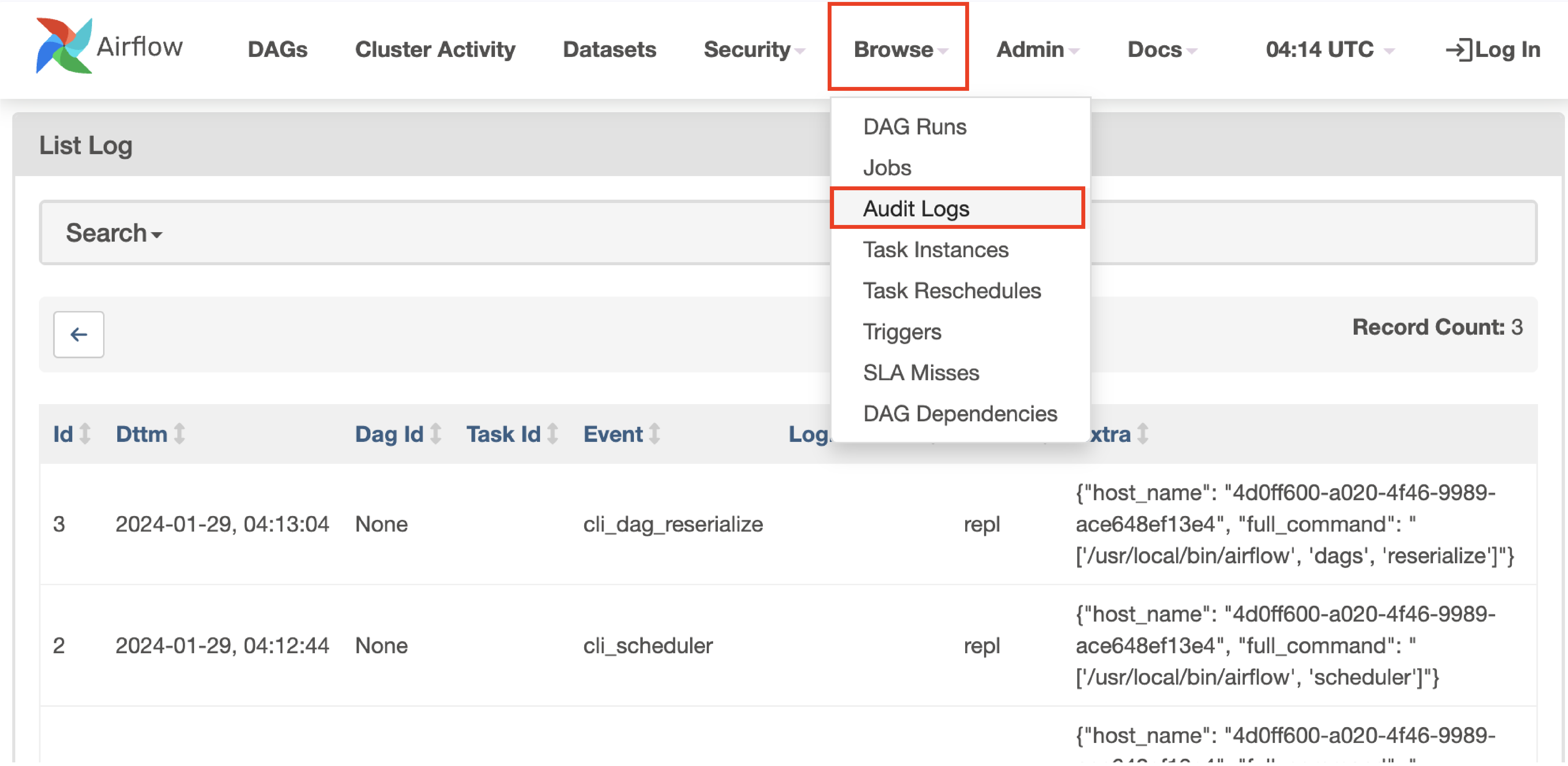
```

from airflow import DAG
from airflow.operators.bash import BashOperator

with DAG(
   'example_dag',
   default_args={"start_date": "2023-02-01"}
):

  part1 = BashOperator(
      task_id='generate_random_number',
      bash_command='echo $RANDOM'
```

Audit logs



Id	Dttm	Dag Id	Task Id	Event	Log		xtra
3	2024-01-29, 04:13:04	None		cli_dag_reserialize		repl	{"host_name": "4d0ff600-a020-4f46-9989-ace648ef13e4", "full_command": "['/usr/local/bin/airflow', 'dags', 'reserialize']"}
2	2024-01-29, 04:12:44	None		cli_scheduler		repl	{"host_name": "4d0ff600-a020-4f46-9989-ace648ef13e4", "full_command": "['/usr/local/bin/airflow', 'scheduler']"}
							{"host_name": "4d0ff600-a020-4f46-9989-

Web UI vs command line

In most cases:

- Equally powerful depending on needs
- Web UI is easier
- Command line tool may be easier to access depending on settings

Let's practice!

INTRODUCTION TO AIRFLOW IN PYTHON