Guided Tour to DAG Authoring

Jed Cunningham

Staff Software Engineer @ astronomer.io

Committer / PMC

ASTRONOMER

(After) Party Under the Stars

Wednesday, September 20th
6:30pm-10:00pm

The Sheraton Centre
123 Queen St W
(7 min walk)

RSVP Now

Airflow Summit
September 19 - 21, 2023
Toronto, Canada
airflowsummit.org

Let's flow together

Workshop

Get Airflow Certified

Thursday, September 21st
12:00 pm in Trinity 4

Marc Lamberti
Head of Customer Education
at Astronomer

Agenda

- Deferrable Tasks — Airflow 2.2
- Dynamic Task Mapping — Airflow 2.3
- Dynamic DAGs
- Datasets — Airflow 2.4
- Setup / Teardown — Airflow 2.7
- Params

Deferrable Tasks

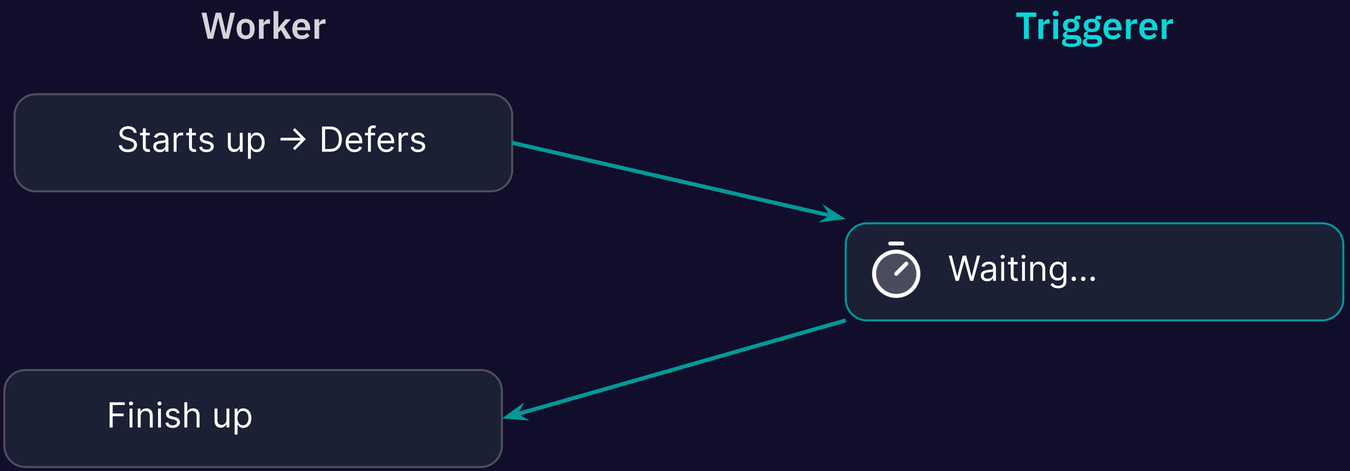
ASTRONOMER

Deferrable Tasks

→ Airflow 2.2

→ Operator/Sensor that can run async

Don’t take up a worker slot


Worker
Triggerer
Starts up → Defers
Waiting...
Finish up

Deferrable KPO

```
KubernetesPodOperator(
    task_id="kpo",
    ...,
    deferrable=True,
)
```

ASTRONOMER

Deferrable KPO

```
$ kubectl get pods -l airflow-worker -w

NAME              STATUS
def-kpo-8czrhyrp   Running
def-kpo-8czrhyrp   Completed
...
def-kpo-69gnklwz   Running
def-kpo-69gnklwz   Completed
```

Tons of support now!

→ Providers with deferrable support:

- AWS
- Google
- Azure
- DBT
- K8s
- More!

→ `[operators] default_deferrable`

Custom Operators / Sensors

```
def execute():
    self.defer(
        trigger=SomeTigger(),
        method_name="execute_complete",
    )

def execute_complete():
    return
```

Custom Operators / Sensors

```
class SomeTrigger(BaseTrigger):
    def serialize(self):
        return ("path.to.SomeTrigger", {})

    async def run(self):
        yield TriggerEvent()
```

Dynamic Task Mapping

Dynamic Task Mapping

→ Airflow 2.3

→ “For loop” for your tasks

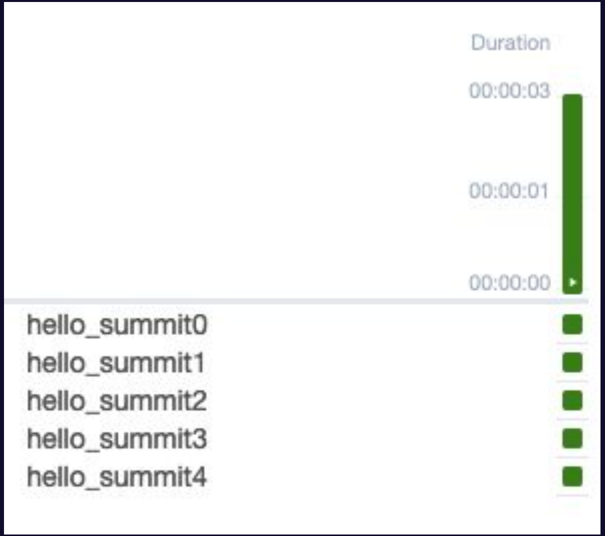
Based on output from a previous task, or static list

→ “Reduce” tasks

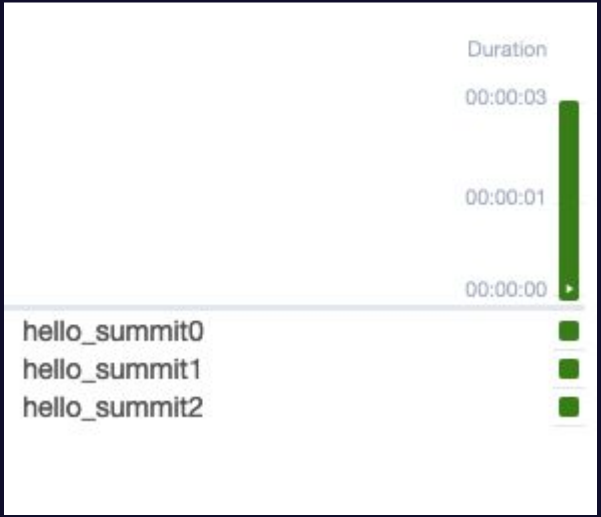
Task that operates on all results of a mapped task

Not like this:

```
for file in {s3 bucket}:
    BashOperator({file})
```



Duration
00:00:03
00:00:01
00:00:00
hello_summit0
hello_summit1
hello_summit2
hello_summit3
hello_summit4

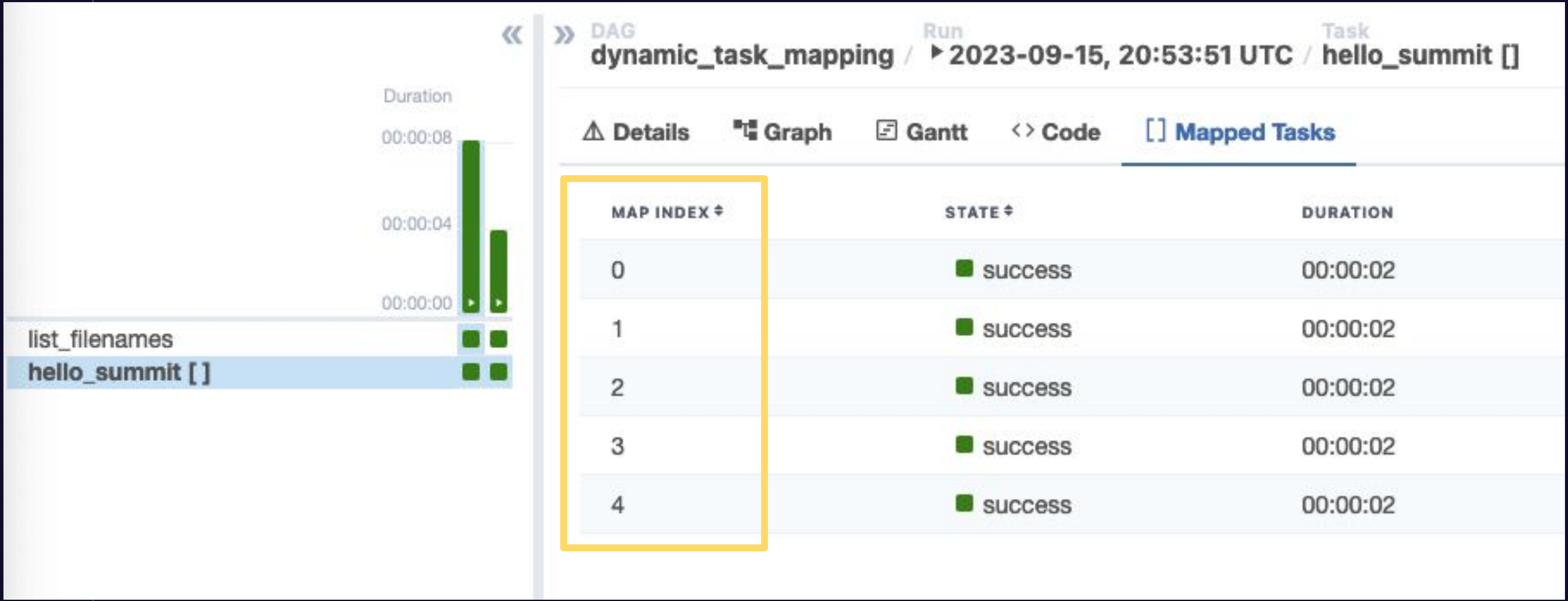


Duration
00:00:03
00:00:01
00:00:00
hello_summit0
hello_summit1
hello_summit2

Like this:

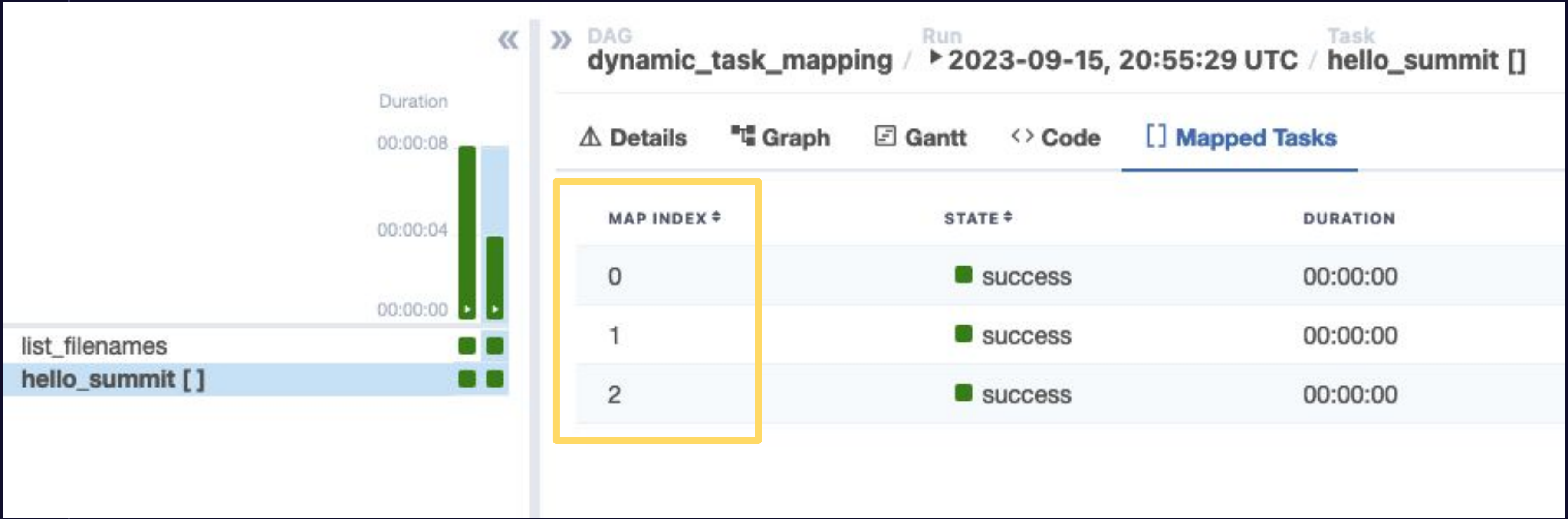
```
list_filenames = S3ListOperator(...)


SomeOperator
    .partial(task_id="hello_summit")
    .expand(thing=list_filenames.output)
```

Like this:



MAP INDEX	STATE	DURATION
0	🟩 success	00:00:02
1	🟩 success	00:00:02
2	🟩 success	00:00:02
3	🟩 success	00:00:02
4	🟩 success	00:00:02

Like this:



DAG dynamic_task_mapping / Run 2023-09-15, 20:55:29 UTC / Task hello_summit []

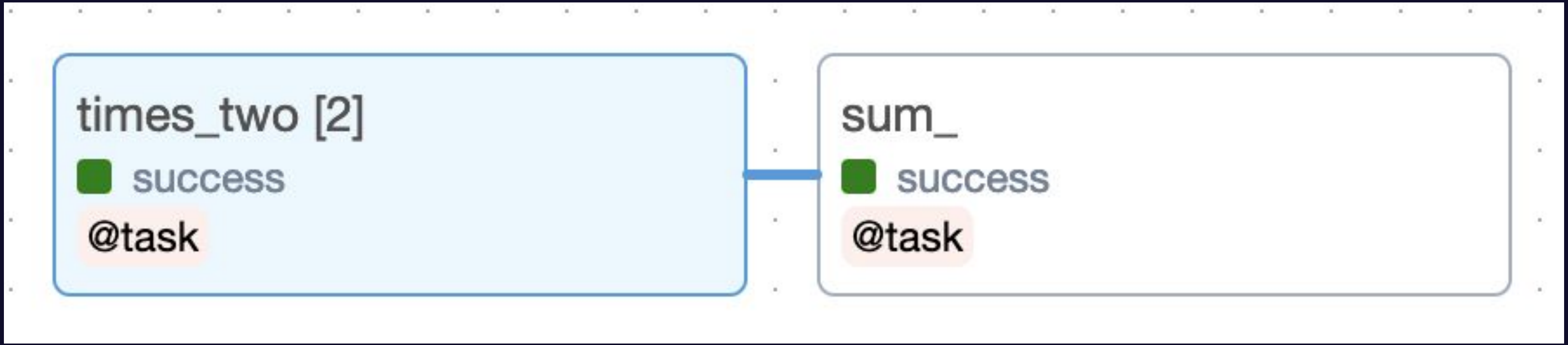
MAP INDEX	STATE	DURATION
0	🟩 success	00:00:00
1	🟩 success	00:00:00
2	🟩 success	00:00:00

Reduce:

```
doubled = times_two.expand(x=[1, 2])
sum_(doubled)
```

ASTRONOMER

Reduce:



Dynamic DAGs

Auto Registration

```
for thing in list_of_things:
    with DAG(f“generated_dag_{thing}”, ...) ~~as dag~~:
        …
        ~~globals()[dag_id]~~
```

Dynamic DAGs

```
for thing in list_of_things:
    with DAG(f"generated_dag_{thing}", ...):
        ...
```

Magic Loop in 2.4?

```
desired_id = get_parsing_context().dag_id


for thing in list_of_things:
    dag_id = f"generated_dag_{thing}"
    if desired_id and desired_id != dag_id:
        continue
    …
```

Dynamic DAGs

- Positives:
  - Easy code reuse
- Negatives:
  - Debugging complexity
  - Scaling

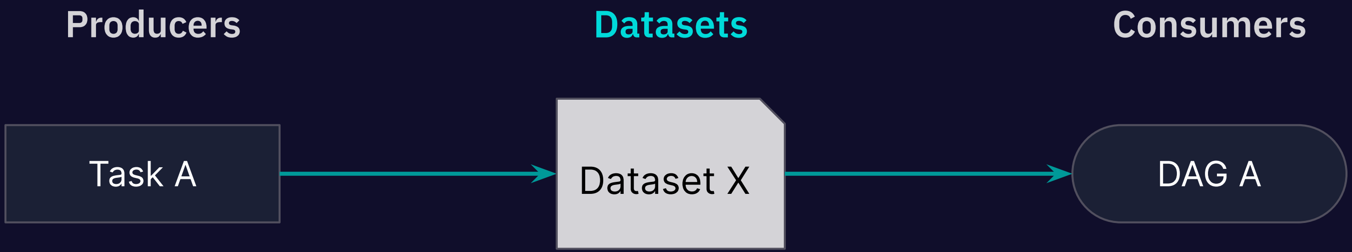
Datasets

Datasets

→ Airflow 2.4

→ Data aware scheduling

Schedule DAG runs based on tasking updating data

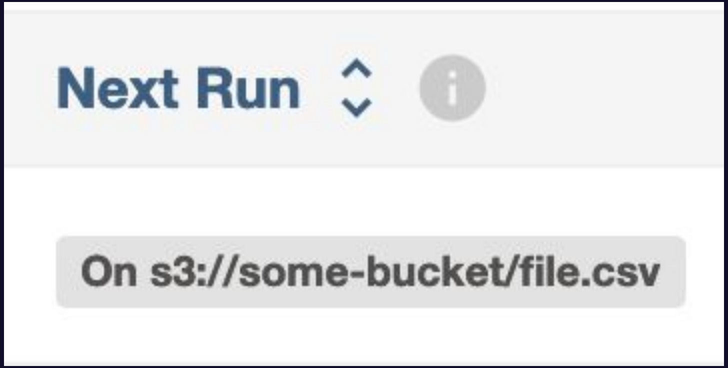

Producers
Datasets
Consumers
Task A
Dataset X
DAG A

ASTRONOMER

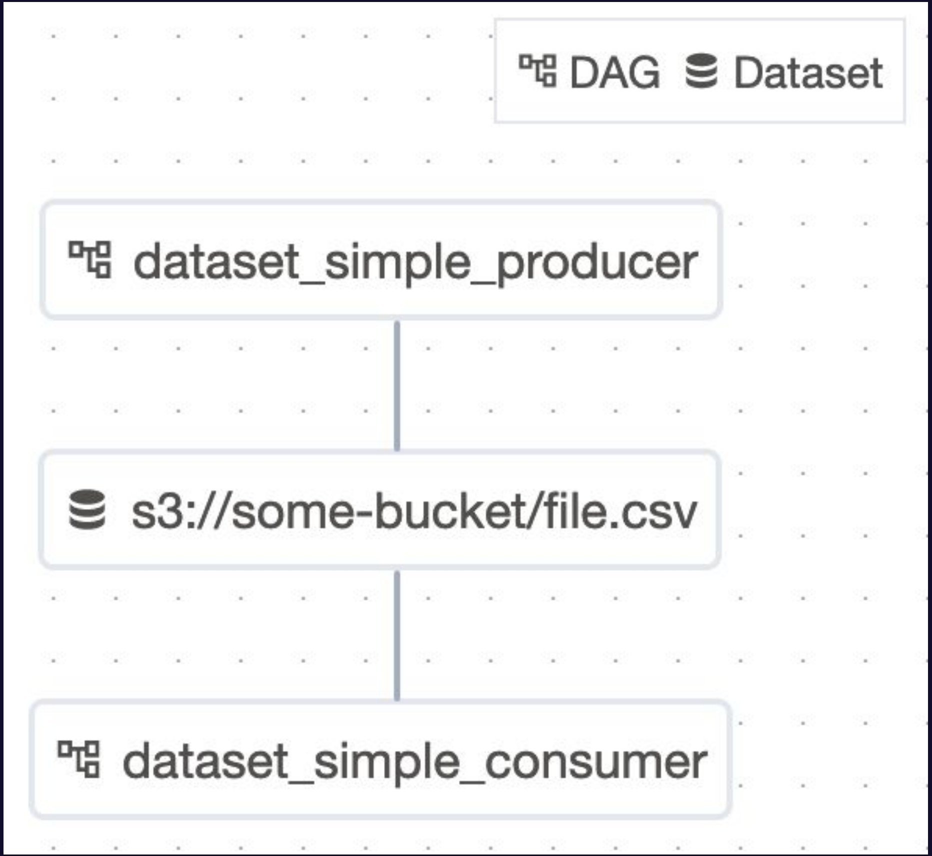
```
MyOperator(
    outlets=[
        Dataset("s3://some-bucket/file.csv")
    ],
    ...,
)
```

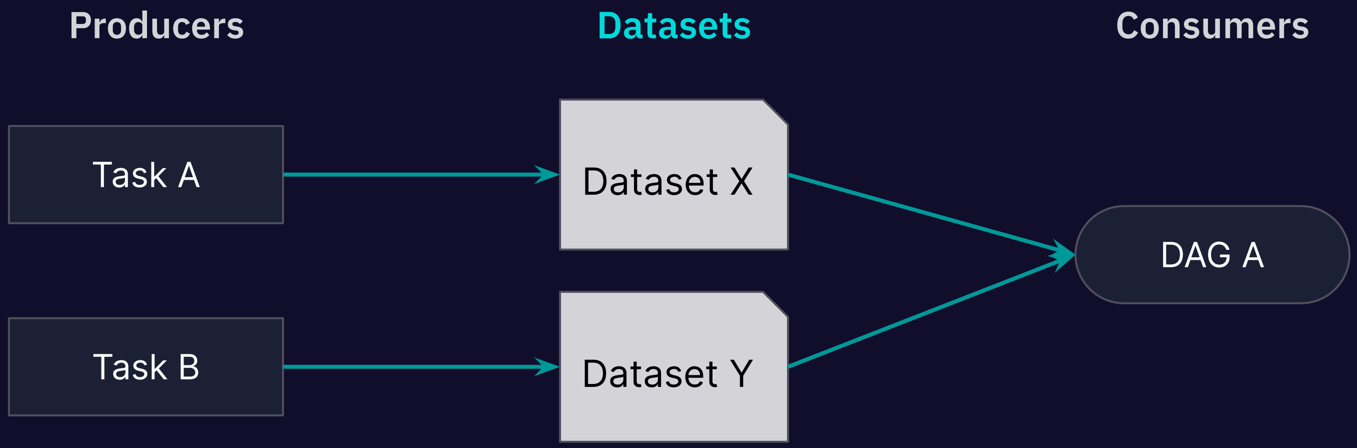
ASTRONOMER

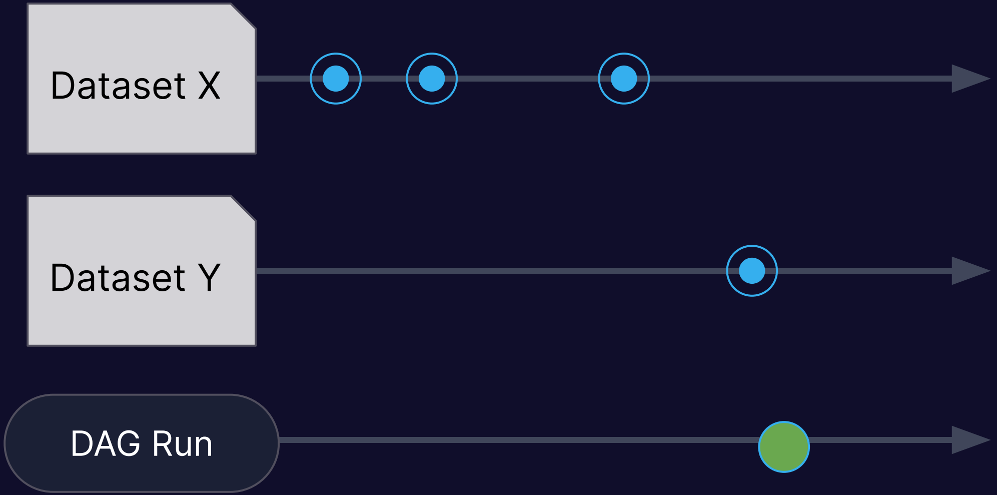
```
with DAG(
    schedule=[
        Dataset("s3://some-bucket/file.csv")
    ],
    ...,
):
    ...
```

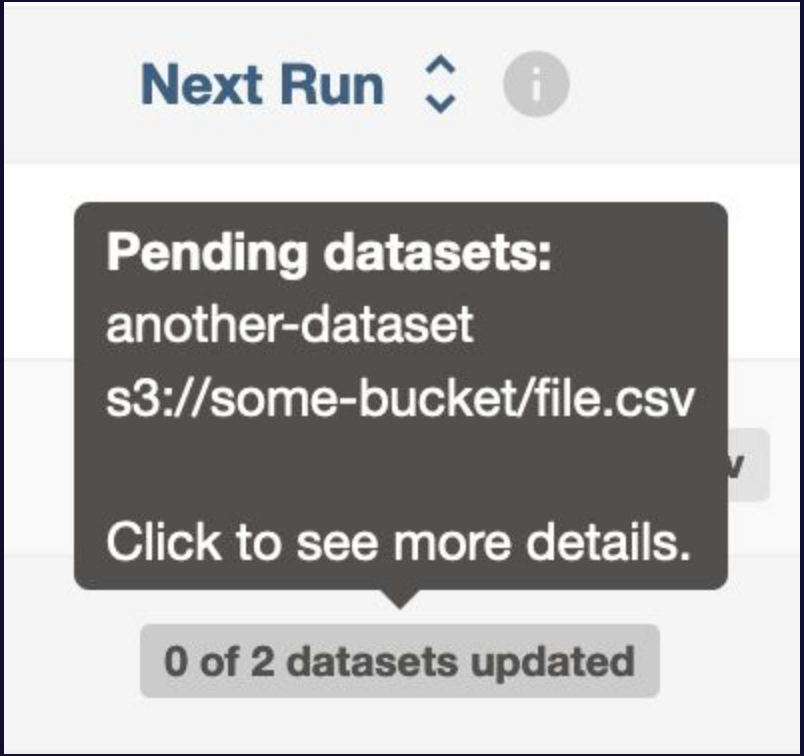


On s3://some-bucket/file.csv


DAG
Dataset
dataset_simple_producer
s3://some-bucket/file.csv
dataset_simple_consumer


Producers
Datasets
Consumers
Task A
Task B
Dataset X
Dataset Y
DAG A

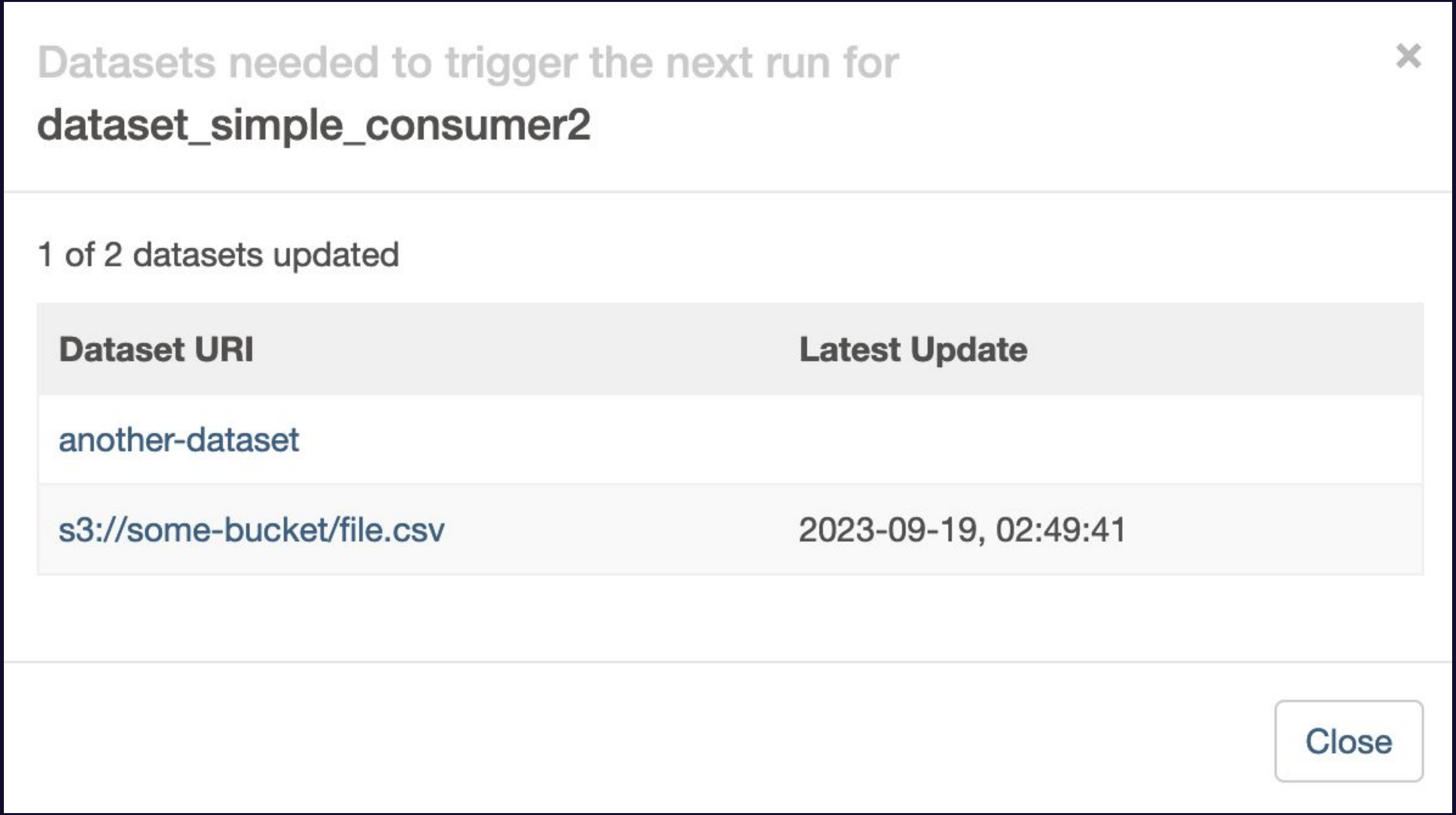

Dataset X
Dataset Y
DAG Run



Pending datasets:
another-dataset
s3://some-bucket/file.csv

Click to see more details.

0 of 2 datasets updated



Datasets needed to trigger the next run for
dataset_simple_consumer2

1 of 2 datasets updated

Dataset URI	Latest Update
another-dataset	
s3://some-bucket/file.csv	2023-09-19, 02:49:41

→ What can a Dataset be?

Setup / Teardown

Setup / Teardown

→ Airflow 2.7

→ “Bookend” tasks (support tasks)

Cleared automatically

Teardown runs if setup ran

Teardown not considered for DAG run state

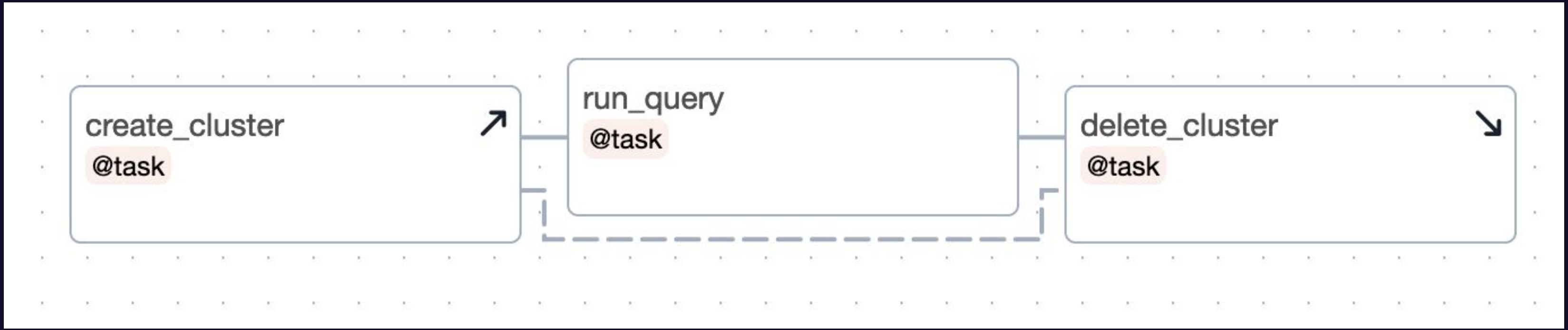
Setup / Teardown

```
create_cluster >> run_query >> delete_cluster
```

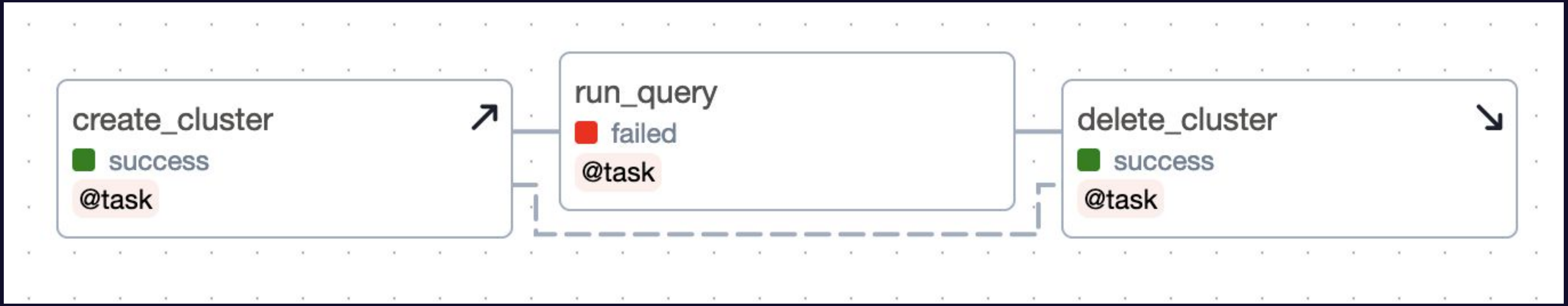
↓

```
create_cluster >> run_query
run_query >> delete_cluster.as_teardown(setups=create_cluster)
```

Setup / Teardown

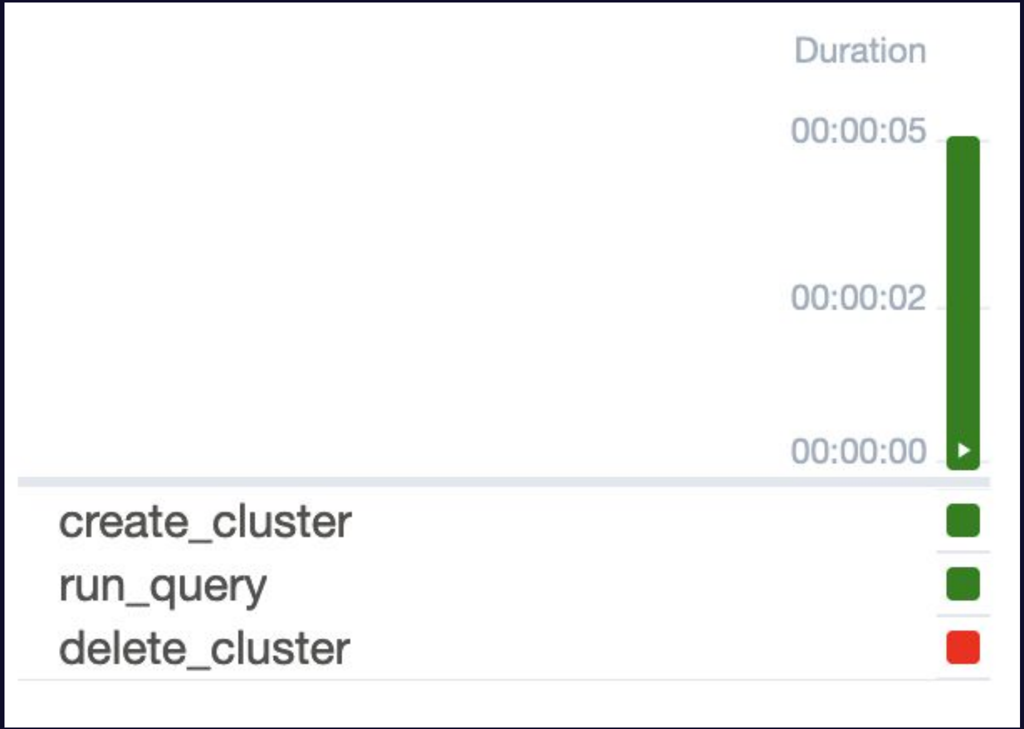


Teardown always runs



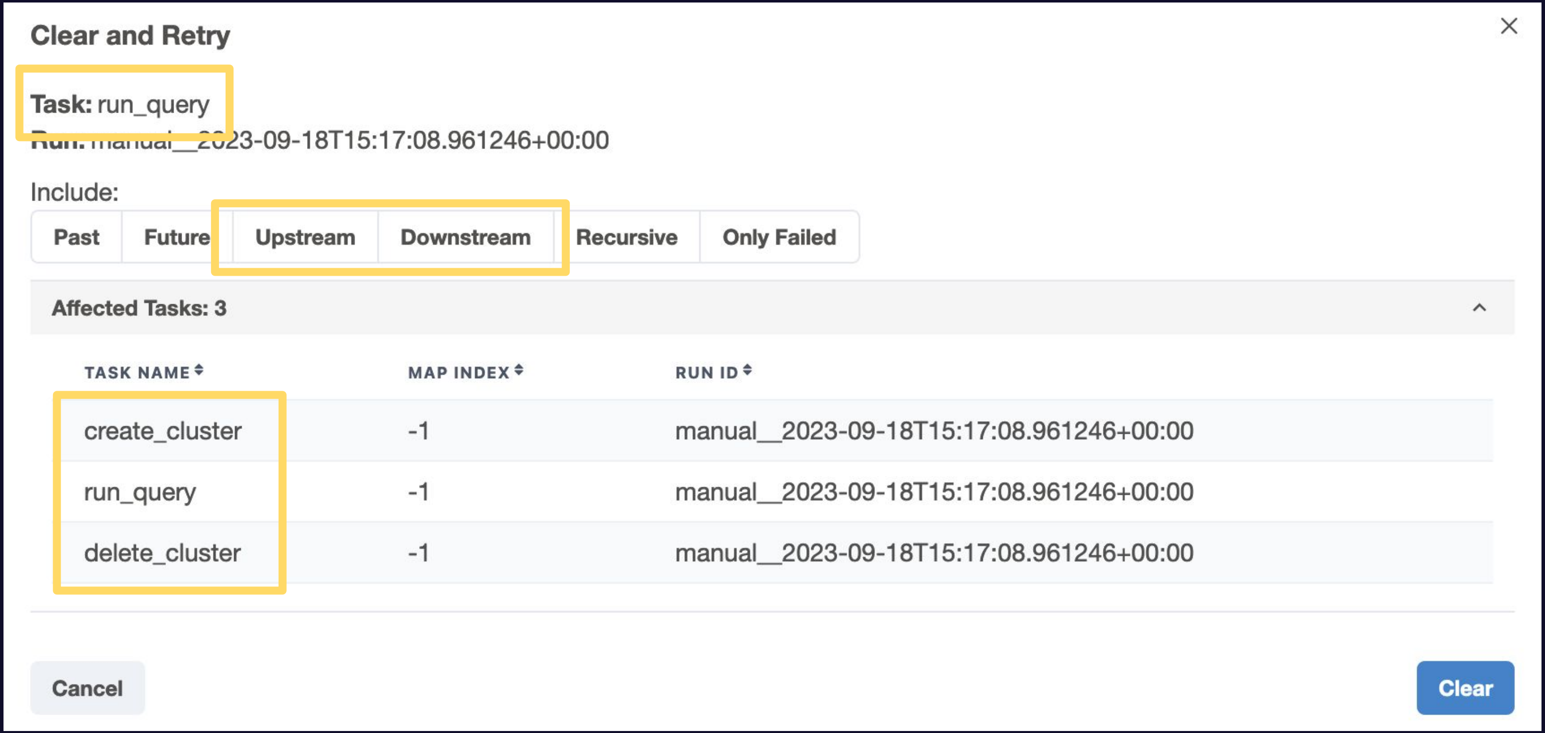
ASTRONOMER

DAG Run State



create_cluster	🟩
run_query	🟩
delete_cluster	🟥

Clearing



Clear and Retry

Task: run_query
Run: manual__2023-09-18T15:17:08.961246+00:00

Include:

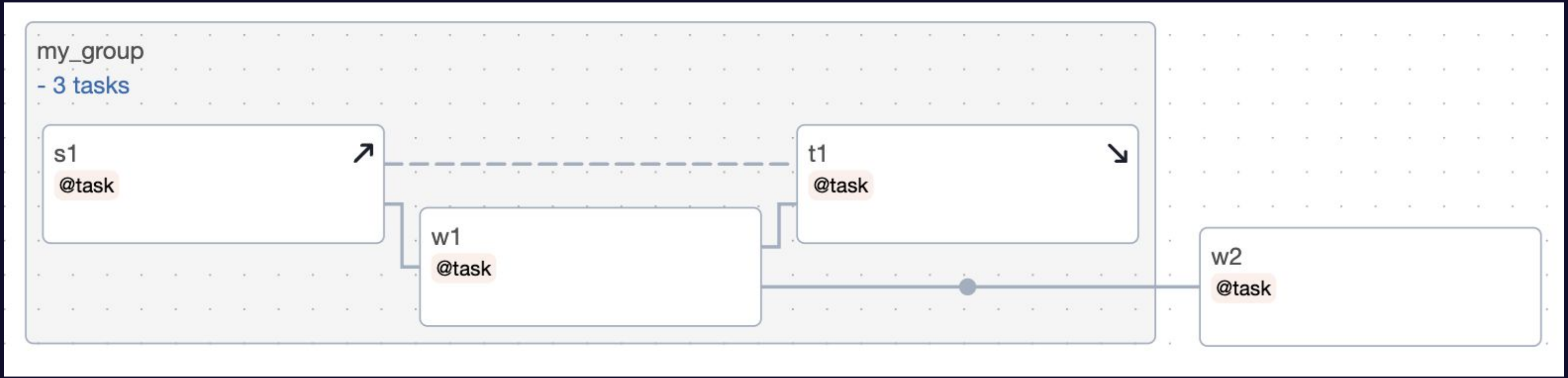
Affected Tasks: 3

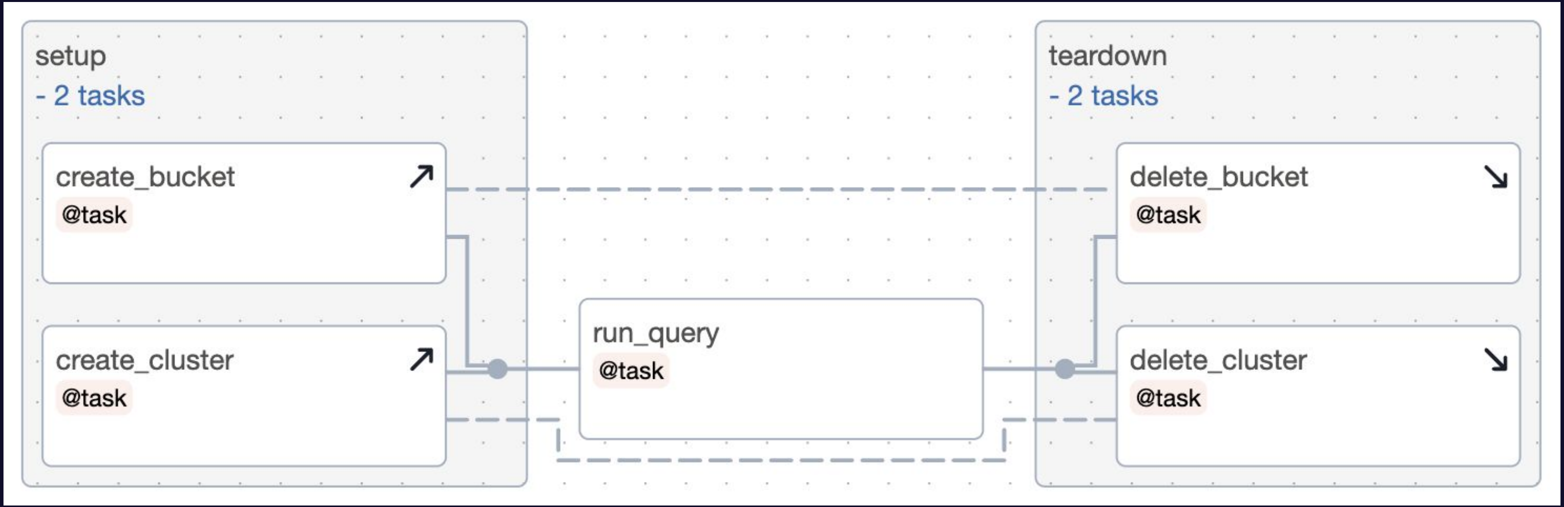
TASK NAME	MAP INDEX	RUN ID
create_cluster	-1	manual__2023-09-18T15:17:08.961246+00:00
run_query	-1	manual__2023-09-18T15:17:08.961246+00:00
delete_cluster	-1	manual__2023-09-18T15:17:08.961246+00:00

```
@setup
def create_cluster():
       ...

@teardown
def delete_cluster():
       ...

with create_cluster() >> delete_cluster():
       run_query()
```

```
with TaskGroup("my_group") as tg:
    s1 = s1()
    s1 >> w1() >> t1().as_teardown(setups=s1)
```

```
tg >> w2()
```


my_group
- 3 tasks
s1
@task
t1
@task
w1
@task
w2
@task

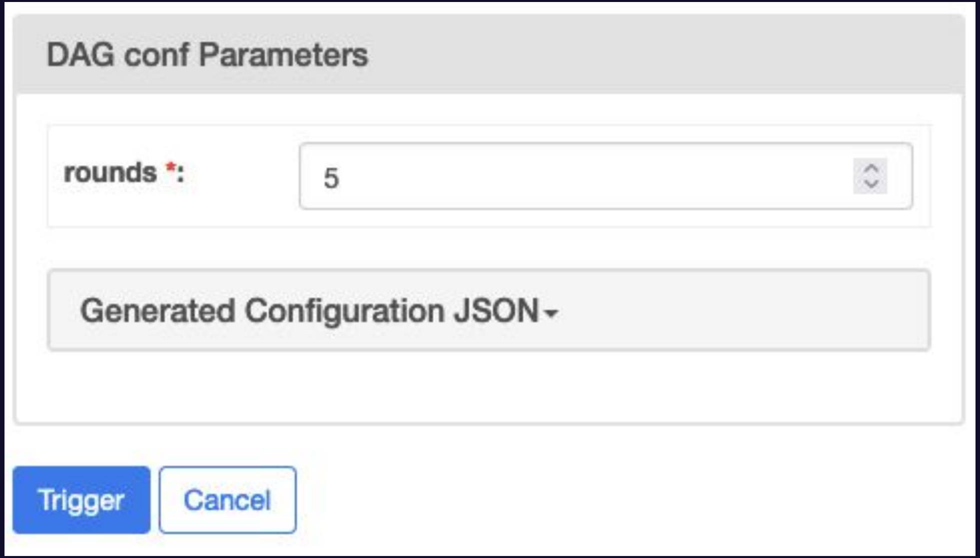

setup
- 2 tasks
create_bucket
@task
create_cluster
@task
run_query
@task
teardown
- 2 tasks
delete_bucket
@task
delete_cluster
@task

Params

Params

- Provide input to runs
- JSON Schema

```
with DAG(
    "some_dag",
    params={
        "rounds": Param(5, type="integer", minimum=3),
    },
 ):
```



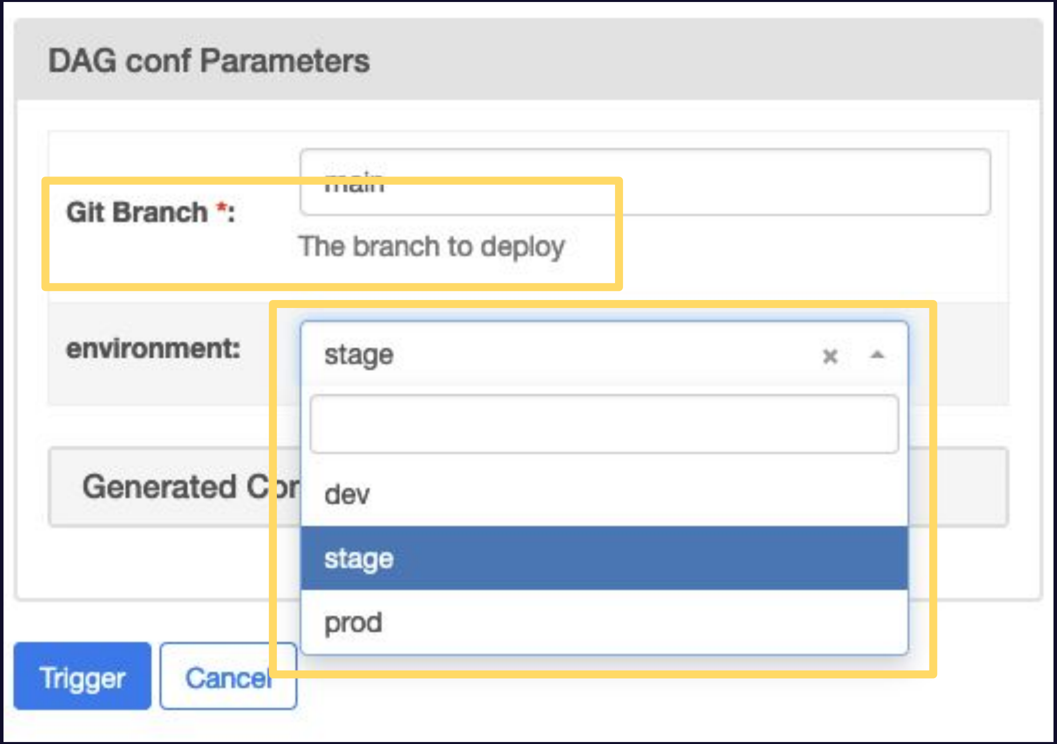
DAG conf Parameters

rounds *: 5

```
def summit(params):
    print(f"Doing {params['rounds']} rounds!")

PythonOperator(task_id="summit", python_callable=summit)
```

```
BashOperator(
    task_id="hello",
    bash_command='echo "Doing $ROUNDS rounds!"',
    env={"ROUNDS": "{{ params.rounds }}"},
)
```

```
"branch": Param(
    default="main",
    type="string",
    title=”Git Branch”,
    description="The branch to deploy",
),
"environment": Param(
    enum=["dev", "stage", "prod"],
    default="stage"
),
```



DAG conf Parameters

Git Branch *: main

The branch to deploy

environment: stage

Get Involved!

- Over 2600 contributors
- All contributions are valuable
- Join #development-first-pr-support on Slack

Q/A