

What Everybody Ought to Know About Airflow

Marc Lamberti - Head of Customer Education @Astronomer



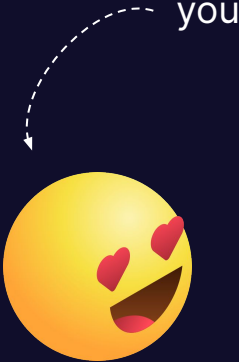
| INGEST | DATA LAKE | METADATA | COMPUTE ENGINES | PIPELINES | PRACTITIONERS APPS | | GOVERNANCE |
|---|---|--|--|---|---|--|------------|
| Ingest Tech Google Cloud Pub/Sub FOLIO Beam Spark PULSAR Redpanda BENEATH memphis StreamNative nifi upasolver CONFLUENT KASKADA Ftek decodable Ingest SaaS Airbyte Fivetran Segment Keboola Rivory Databdo rudderstack datacoral Stitch MATILLION SNOWFLOW | Object Storage Microsoft Azure Blob Storage Google Cloud Storage amazon S3 ORACLE CLOUD IBM Cloud Object Storage Alibaba Cloud hadoop MINIO zadara cloudfiles DigitalOcean SwiftStack wasabi CEPH PURESTORAGE CrowdStorage VAST filebase | Metastore databricks Unity Catalog Tabular CLOUDERA Data Version Control Infra lakeFS Hesse Open Table Formats ONEHOUSE Shudi ICEBERG GRC Delta Lake | Distributed Compute databricks Spark Cloudwick ASSEMBLY Cloud Foundry YANFIRE dask Azure HDInsight Amazon EMR akka trino Azure Databricks CLOUDERA bodai RAY Analytics Engines DATA FUSION Google BigQuery Synapse amazon EMR snowflake pinot databricks cloudflyte StarRocks FIREBOLT HASKET Imply ClickHouse amazon EMR dremio POLAR ROCKSET CASSIANDATA pintabao star-tree Dubble druid Starburst Yellowbrick HYDRA DuckDB DORIS | Orchestration Airflow PREFECT Luigi Flyte dagster ASTRONJMER Datorious Shipyard MAQE Kestra DAGWORKS Data Quality / Observability KENSU pandera great-expectations metaspine lightup SODA Datafold WHYLABS Bigeye GRIFFIN MONTE CARLO Databand UNRAVEL IQ-robot Anomalo HoloClean aws-labs/deequ elementary oceanbase timeseer.AI | MLOps End-to-End OctoML ABACUS.AI METAFLOW baseten Verta AIBLE DataRobot Wallaroo cnvrg.io Hugging Face mlflow SELDON DOWING Kedro W&B graft Kubeflow CLEAR ML StreamCloud ZenML dotData Iterative neptune.ai comet Amazon SageMaker BENTOML deci Modular DAGHub dataiku Qwak Valohai Data Centric AI/ML DYC activeloop Pachyderm GRAVITI ECTIO Stack AI Galileo VOXELS xethub DataLad snoriel dstack DataLoop Labelbox scale ML observability and monitoring deepchecks mono Superwise fiddler DataBuck arize Apres census EVIDENTLY AI Arthur WHYLABS Giskard badook hyperaigence.io ROBUST INTELLIGENCE TRUBRA opona GANTRY Feature Stores HOPSWORKS redis scribble Data FEAST TAYTON Notebooks count Noteable Deepnote Analytics Workflow dataform dbt Querybook databricks | Data Catalog/ Governance ckan Amundsen BigD DatsLab boomi raito MARQUEZ atlan Apache Atlas magda IMMUTA OKERA ETL Watson dataworld Alation SELECT STAR zeeneo Platform sterenna Colibra Acryl Data | |

Source: LakeFS

$$7 \times 4 \times 30 = 840$$









```
pip install apache-airflow==2.7
```



```
curl -Lf0 'https://airflow.apache.org/docs/apache-airflow/2.7.0/docker-compose.yaml'  
&& docker compose up -d
```



```
helm repo add apache-airflow https://airflow.apache.org  
helm upgrade --install airflow apache-airflow/airflow --namespace airflow --create-namespace
```





```
brew install astro  
astro dev init  
astro dev start
```

```
> .astro  
> dags  
> include  
> plugins  
> tests  
.dockerignore  
.env  
.gitignore  
airflow_settings.yaml  
Dockerfile  
packages.txt  
README.md  
requirements.txt
```





```
pip install airflowctl  
airflowctl init my_airflow_project --build-start
```

```
└─ my_airflow_project  
  └─ .airflowctl  
  └─ .env  
  └─ dags  
  └─ logs  
  └─ plugins  
  └─ .env  
  └─ .gitignore  
  └─ airflow-webserver.pid  
  └─ airflow.cfg  
  └─ airflow.db  
  └─ requirements.txt  
  └─ settings.yaml  
  └─ standalone_admin_password.txt  
  └─ webserver_config.py
```





```
AIRFLOW__SECRETS__USE_CACHE=True
```





Airflow DAGs Cluster Activity Datasets Security Browse Admin Docs Astronomer 10:15 UTC AU

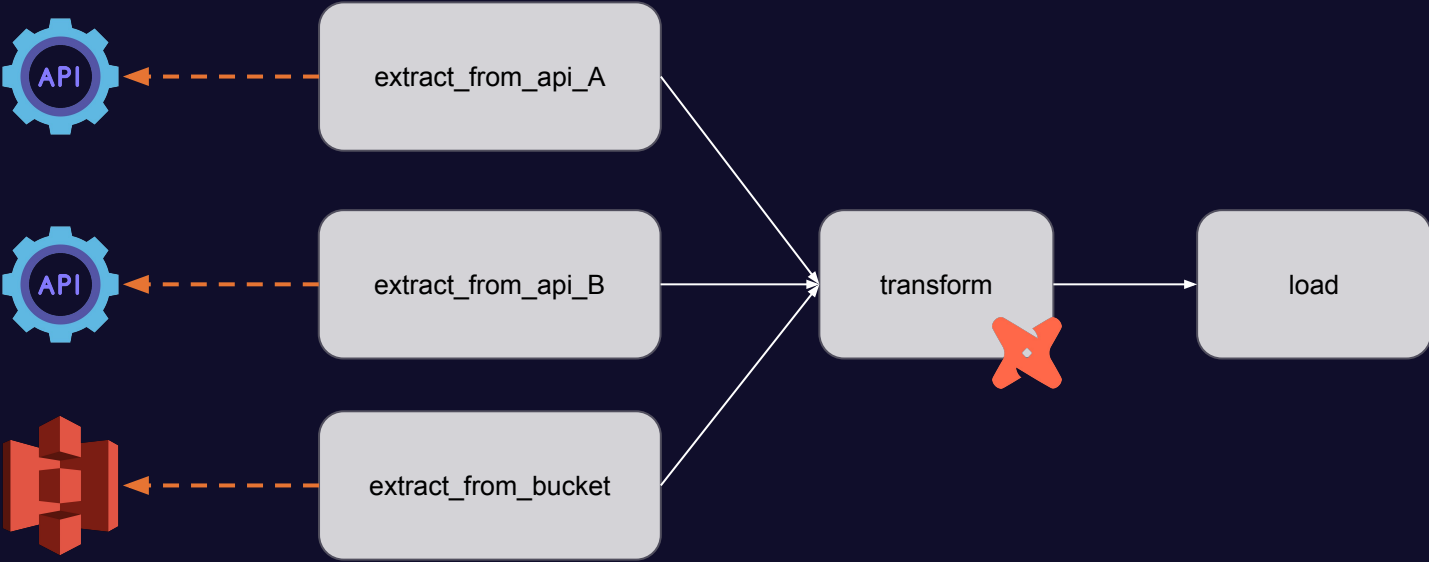
DAGs

All 8 Active 5 Paused 3 Running 0 Failed 2 Filter DAGs by tag Search DAGs Auto-refresh

| DAG | Owner | Runs | Schedule | Last Run | Next Run | Recent Tasks | Actions | Links |
|--|---------|-------|----------|----------------------|----------------------|--------------|---------|-------|
| <input checked="" type="checkbox"/> edubot_airflow_101_completion edubot skilljar | airflow | 2 | None | 2023-06-21, 11:43:18 | | 4 | | ... |
| <input checked="" type="checkbox"/> edubot_course_feedback sigma snowflake typeform | airflow | 498 8 | @daily | 2023-08-28, 00:00:00 | 2023-08-29, 00:00:00 | 2 1 2 | | ... |
| <input type="checkbox"/> edubot_create_groups skilljar | airflow | 8 4 | @daily | 2023-08-09, 10:30:32 | 2023-08-28, 00:00:00 | 3 1 | | ... |
| <input checked="" type="checkbox"/> edubot_education_report edubot skilljar | airflow | 1 | None | 2023-06-02, 12:52:03 | | 2 | | ... |
| <input type="checkbox"/> edubot_group_summary edubot skilljar | airflow | | None | | | | | ... |

Showing 1-8 of 8 DAGs





```
from airflow.models import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime

def _extract():
    return 42

def _transform(ti):
    val = ti.xcom_pull(task_id='extract')
    return val + 42

def _load(ti):
    val = ti.xcom_pull(task_id='transform')
    print(val)

with DAG('my_dag', start_date=datetime(2023, 1, 1), schedule='my dag does that', tags=['team_a']):

    extract = PythonOperator(
        task_id='extract',
        python_callable=_extract
    )

    transform = PythonOperator(
        task_id='transform',
        python_callable=_transform
    )

    load = PythonOperator(
        task_id='load',
        python_callable=_load
    )

    extract >> transform >> load
```





imports



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

DAG



```
@dag(start_date=datetime(2023, 1, 1),
      schedule='@daily',
      description='my dags does that',
      tags=['team_a']
)
def my_dag:
```






```
dag = DAG(...)  
task = Pyhton0perator(dag=dag)
```




```
with DAG(...):  
    task = Python0perator(...)
```

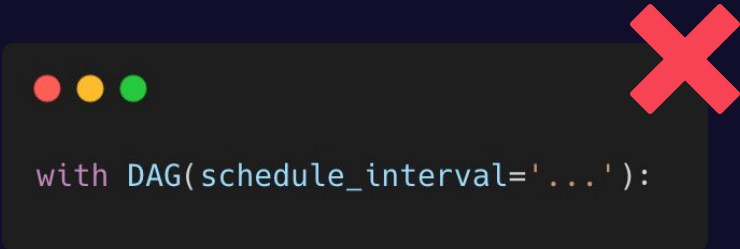


```
with DAG(...) as dag:  
    task = Python0perator(...)
```

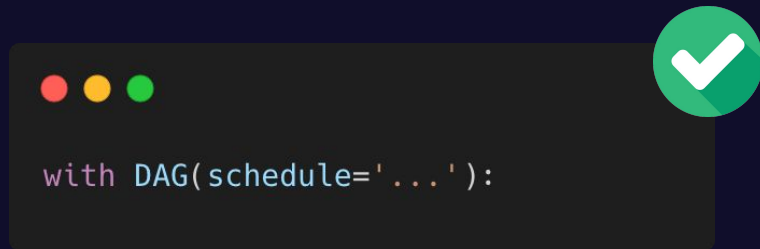


```
@dag(...)  
def my_dag():  
    task = Python0perator(...)  
dag()
```





```
with DAG(schedule_interval='...'):
```



```
with DAG(schedule='...'):
```





```
with DAG(schedule='@daily'):
```



stateless

```
with DAG(schedule=timedelta(days=3)):
```



relative

```
from airflow.timetables.events import EventsTimetable

with DAG(
    schedule=EventsTimetable(
        event_dates=[
            pendulum.datetime(2022, 4, 5, 8, 27),
            pendulum.datetime(2022, 4, 17, 8, 27),
            pendulum.datetime(2022, 4, 22, 20, 50),
        ],
        description="My Team's Baseball Games",
        restrict_to_events=False,
    ),
    ...,
):
```



Freedom!





imports



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

DAG



```
@dag(start_date=datetime(2023, 1, 1),
      schedule='@daily',
      description='my dags does that',
      tags=['team_a'])
def my_dag:
```

Tasks



```
@task
def extract():
    return 42

@task
def transform(val):
    return val + 42

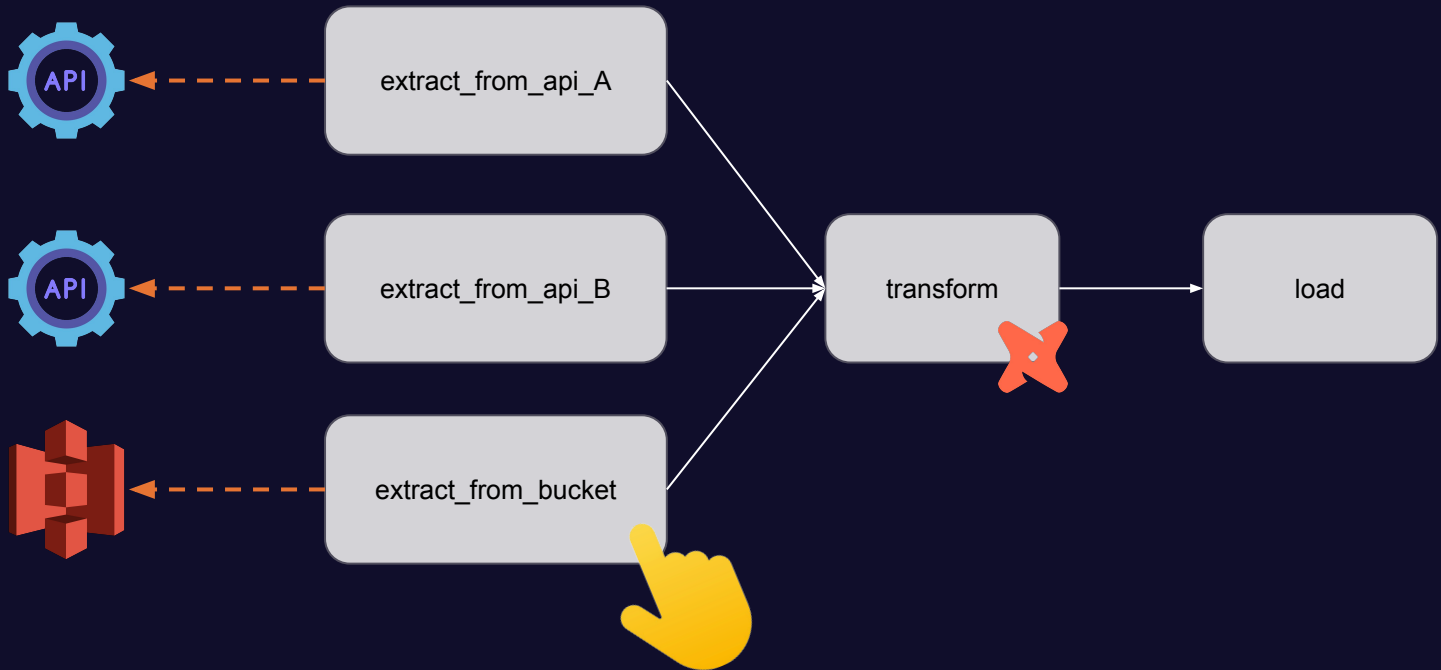
@task
def load(val):
    print(val)
```

dependencies



```
val = extract()
new_val = transform(val)
load(new_val)
```







Add Connection

Connection Id *

Connection Type *
Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.

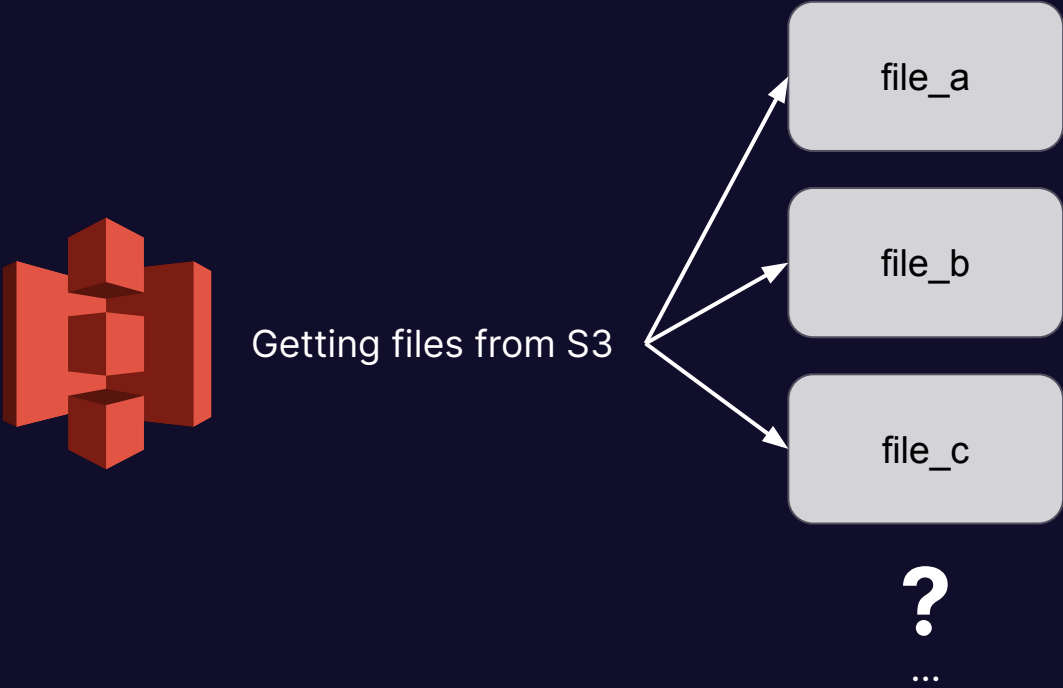
Description

AWS Access Key ID

AWS Secret Access Key

Extra



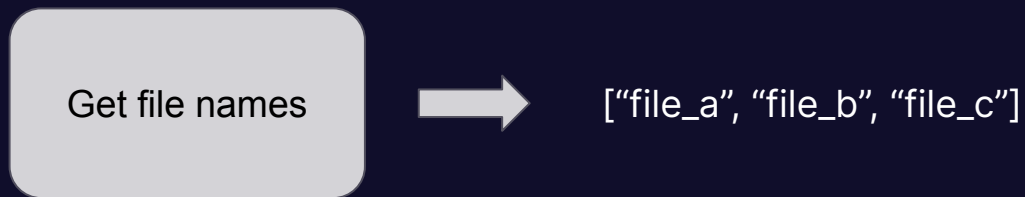




```
def dag():

    copy_to_snowflake = S3ToSnowflakeOperator.partial(
        task_id="load_files_to_snowflake",
        stage="MY_STAGE",
        table="COMBINED_HOMES",
        schema="MYSHEMA",
        file_format="(type = 'CSV',field_delimiter = ',', skip_header=1)",
        snowflake_conn_id="snowflake",
    ).expand(s3_keys=get_s3_files(current_prefix="{{ ds_nodash }}" ))
```



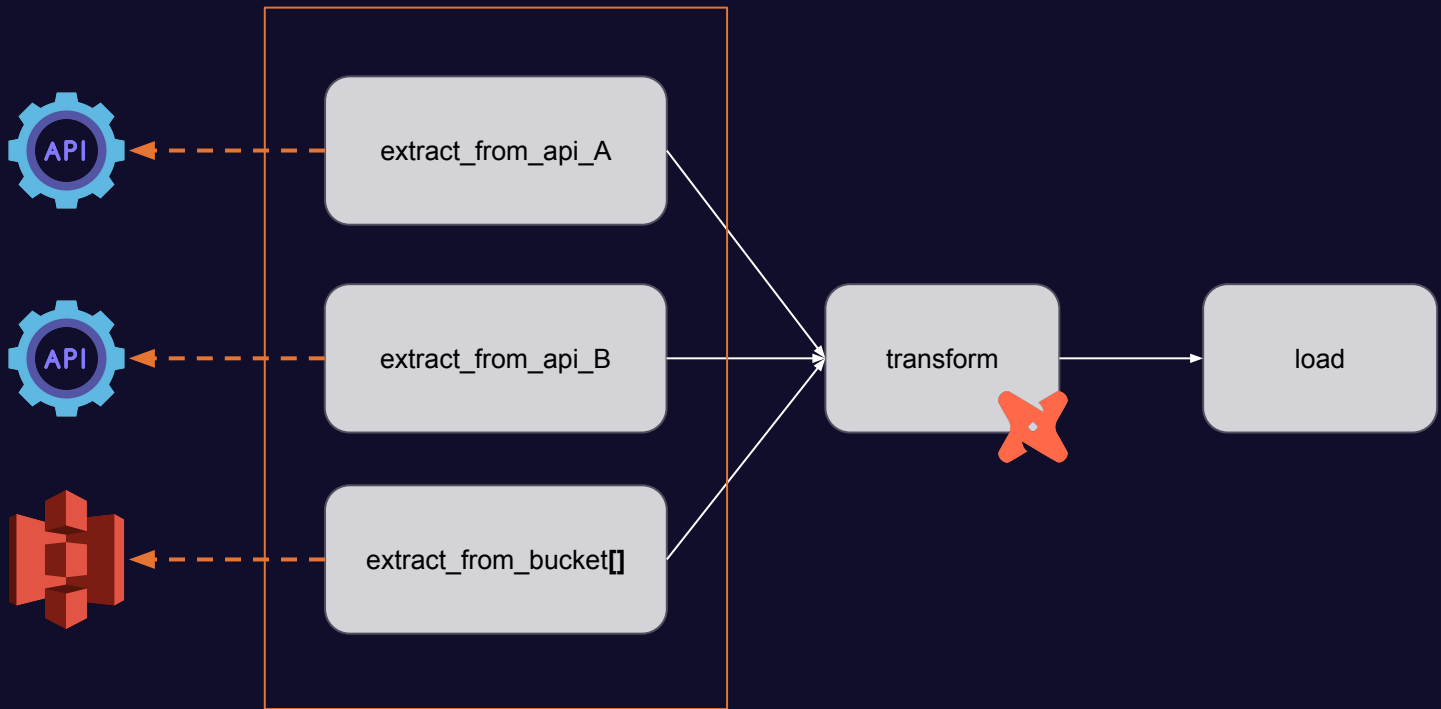


XComArg(["file_a", "file_b", "file_c"]).map(my_func)



```
def add_prefix(filename):  
    return "bucket_" + filename
```







```
import datetime

from airflow import DAG
from airflow.example_dags.subdags.subdag import subdag
from airflow.operators.subdag import SubDagOperator

DAG_NAME = "example_subdag_operator"

with DAG(dag_id=DAG_NAME, start_date=datetime.datetime(2022, 1, 1), schedule="@once") as dag:

    sub = SubDagOperator(
        task_id="section-1",
        subdag=subdag(DAG_NAME, "section-1", dag.default_args),
    )
```

```
from airflow import DAG

def subdag(parent_dag_name, child_dag_name, args) -> DAG:

    dag_subdag = DAG(dag_id=f"{parent_dag_name}.{child_dag_name}", start_date=datetime(2022, 1, 1), schedule="@daily")

    return dag_subdag
```





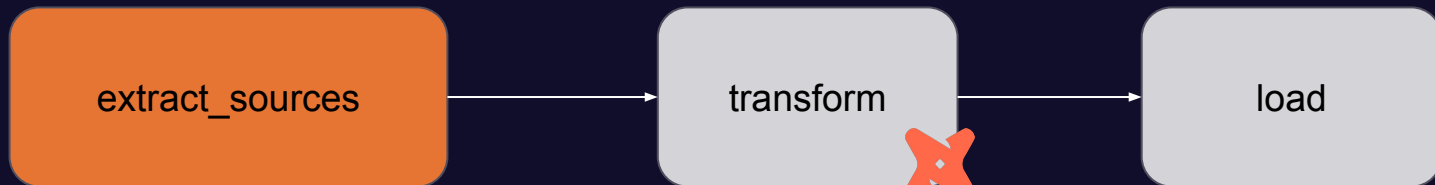
```
from airflow.decorators import task_group

with DAG(...):

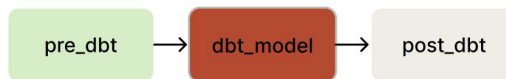
    # Start task group definition
    @task_group(group_id='my_task_group')
    def tg1():
        t1 = EmptyOperator(task_id='task_1')
        t2 = EmptyOperator(task_id='task_2')

        t1 >> t2
    # End task group definition
```

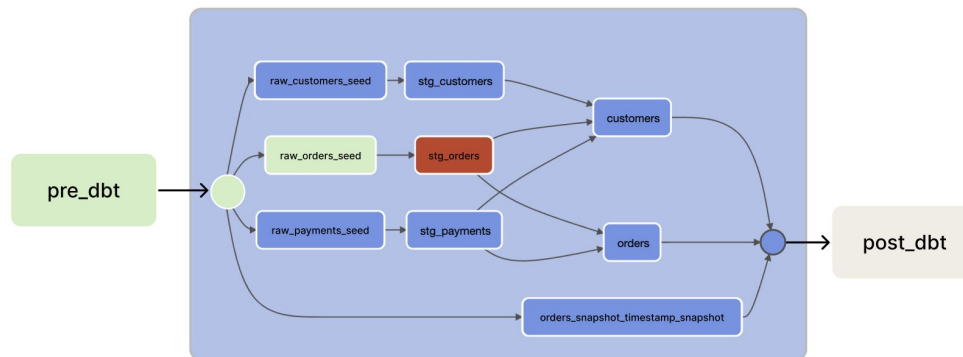




Before Cosmos



With Cosmos





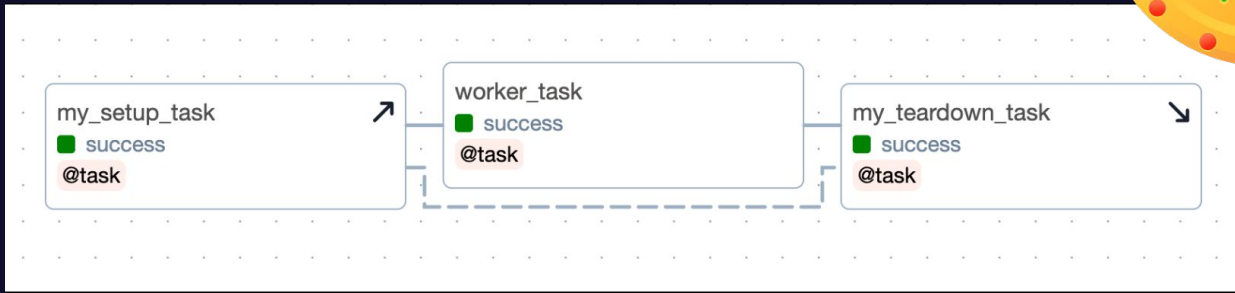
```
from airflow.decorators import task, setup, teardown

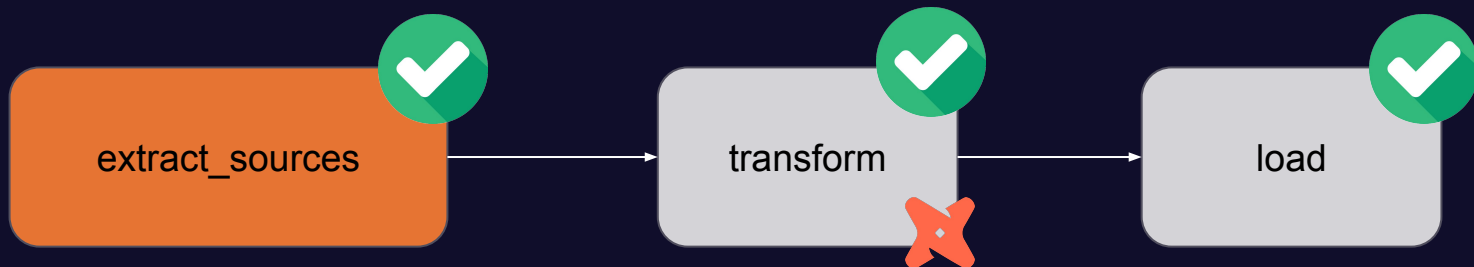
@setup
def my_setup_task():
    print("Setting up resources!")
    my_cluster_id = "cluster-2319"
    return my_cluster_id

@task
def worker_task():
    return "Doing some work!"

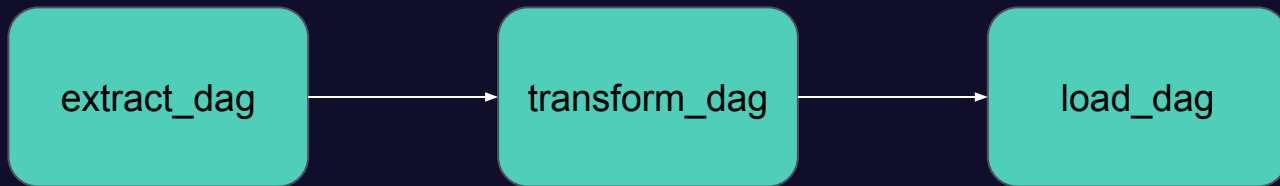
@teardown
def my_teardown_task(my_cluster_id):
    return f"Tearing down {my_cluster_id}!"

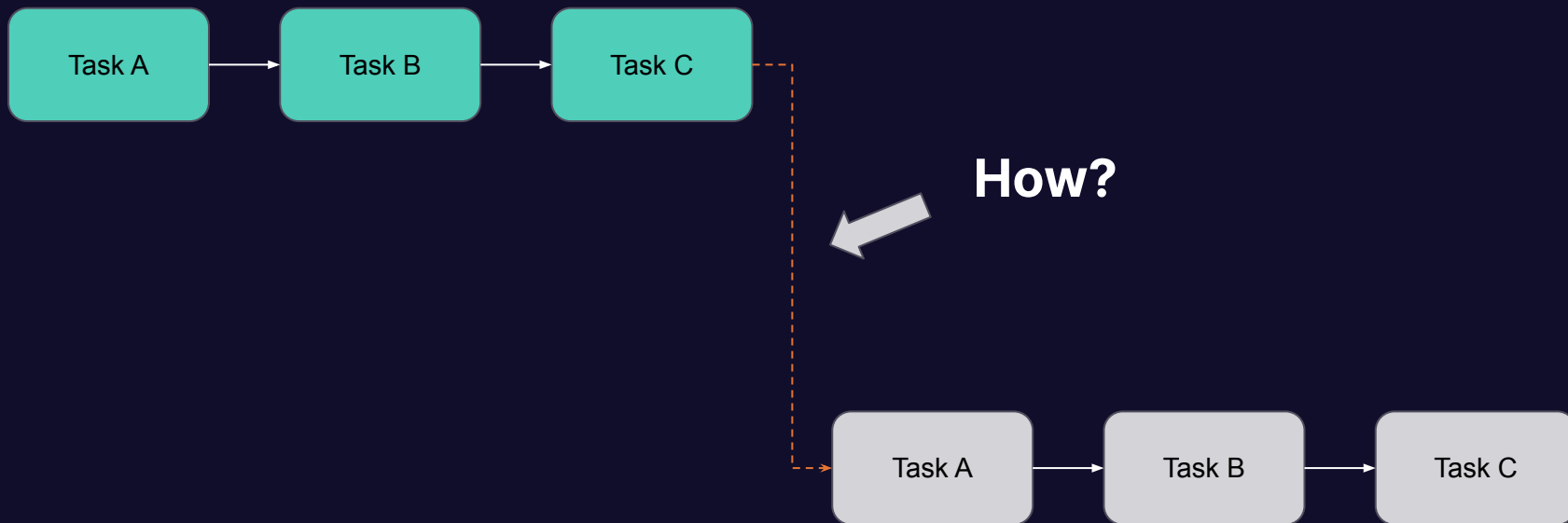
my_setup_task_obj = my_setup_task()
my_setup_task_obj >> worker_task() >> my_teardown_task(my_setup_task_obj)
```











ExternalTaskSensor

TriggerDagRunOperator

Datasets! 





```
from airflow.datasets import Dataset

with DAG(...):
    MyOperator(
        outlets=[Dataset("s3://dataset-bucket/example.csv")],
        ...,
    )

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

| | | | | Schedule: Triggered by datasets | | | |
|-------------------------------------|---------------------------------|---------|------|---------------------------------|---|-------------------------|--|
| <input checked="" type="checkbox"/> | consumes_dataset_1 | airflow | ○○○○ | Dataset | i | 0 of 1 datasets updated | |
| | consumes dataset-scheduled | | | | | | |
| <input checked="" type="checkbox"/> | consumes_dataset_1_and_2 | airflow | ○○○○ | Dataset | i | 1 of 2 datasets updated | |
| | consumes dataset-scheduled | | | | | | |



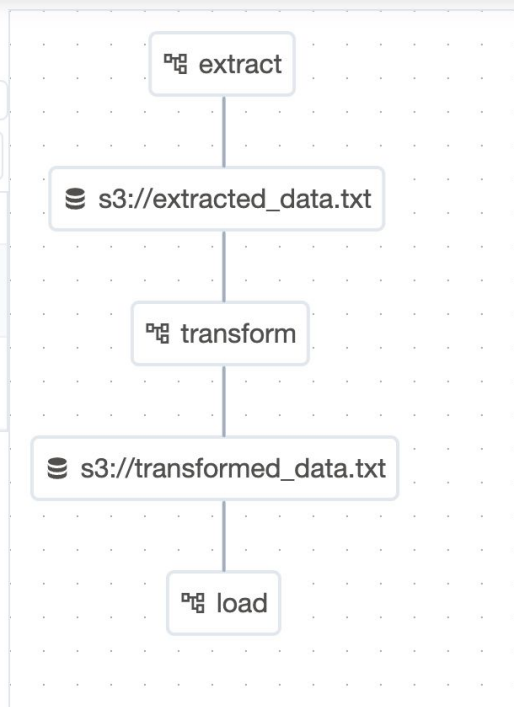


Datasets

Filter datasets with updates in the past: **All Time** 30 days 7 days 24 hours 1 hour

Search by URI...

| URI | LAST UPDATE |
|---------------------------|------------------|
| s3://extracted_data.txt | Total Updates: 0 |
| s3://transformed_data.txt | Total Updates: 0 |







```
# in include/common.py
from airflow.decorators import task

@task
def request_api(endpoint):
    response = requests.get(f'http://my_api/{endpoint}')

# in dags/dag_a.py
from include.common import request_api

with DAG(...):
    request_api.override(task_id='request_api_a')('endpoint_a')

# in dags/dag_b.py
from include.common import request_api

with DAG(...):
    request_api.override(task_id='request_api_b')('endpoint_b')
```





```
from airflow.utils.edgemoifier import Label

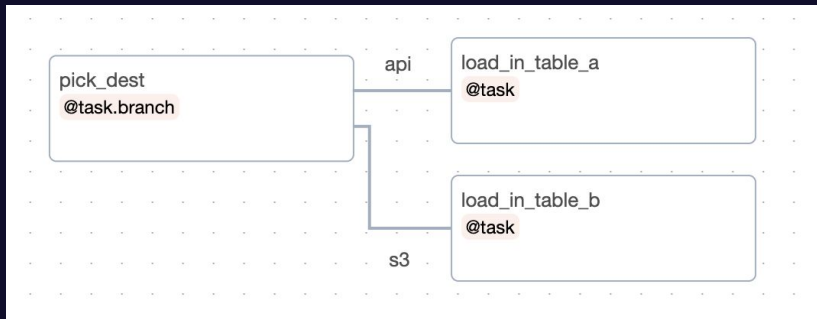
with DAG(...):

    @task.branch(task_id='pick_dest')
    def random_choice():
        return random.choice(['load_in_a', 'load_in_b'])

    @task
    def load_in_table_a():
        pass

    @task
    def load_in_table_b():
        pass

    t = random_choice()
    t >> Label('api') >> load_in_table_a()
    t >> Label('s3') >> load_in_table_b()
```





```
"""  
### Load DAG  
#### Purpose  
Some info for your dag with how to operate it or what it interacts  
with  
#### Notes  
- Supports markdown features  
- [Supports linking to external content](https://example.com/)  
"""  
  
with DAG(..., doc_md=__doc__):
```

DAG: load Schedule: Dataset On s3://transformed_data.txt

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code Audit Log

DAG Docs

Load DAG

Purpose
Some info for your dag with how to operate it or what it interacts with

Notes

- Supports markdown features
- Supports linking to external content





DAGs

Cluster Activity

Datasets

Security

Browse

Admin

Docs

Astronomer

DAGs

All **3** Active **0** Paused **3**

Running **0** Failed **0**

Filter DAGs by tag

Search DAGs

| <i>i</i> | DAG <i>⌵</i> | Owner <i>⌵</i> | Runs <i>i</i> | Schedule | Last Run <i>⌵</i> <i>i</i> | Next Run <i>⌵</i> <i>i</i> |
|--------------------------|-----------------------------|----------------|---------------|------------------|----------------------------|-------------------------------|
| <input type="checkbox"/> | extract <i>example</i> | airflow | ○○○○ | @daily <i>i</i> | | 2023-09-10, 00:00:00 <i>i</i> |
| <input type="checkbox"/> | load <i>example</i> | airflow | ○○○○ | Dataset <i>i</i> | | On s3://transformed_data.txt |
| <input type="checkbox"/> | transform <i>example</i> | airflow | ○○○○ | Dataset <i>i</i> | | On s3://extracted_data.txt |



```
» DAG
extract

Details Graph Gantt <> Code

Parsed at: 2023-09-11, 13:29:19 UTC

1 from airflow.decorators import dag, task
2 from datetime import datetime
3 from datasets import extracted_data
4
5 @dag(
6     start_date=datetime(2023, 1, 1),
7     schedule='@daily',
8     catchup=False,
9     tags=['example'],
10 )
11 def extract():
12
13     @task(outlets=[extracted_data])
14     def extract_data():
15         return {'key': 'value'}
16
17     extract_data()
18
19 extract()
```



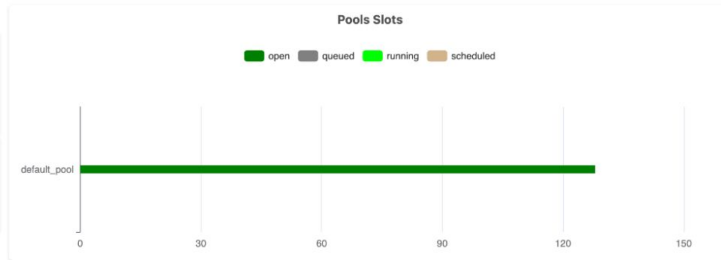


Cluster Activity

Live Metrics

Unpaused DAGs
2
out of 13 total DAGs

Top 5 longest Dag Runs to finish
No dag running
out of 0 total running Dag Runs



Health

MetaDatabase
status: HEALTHY

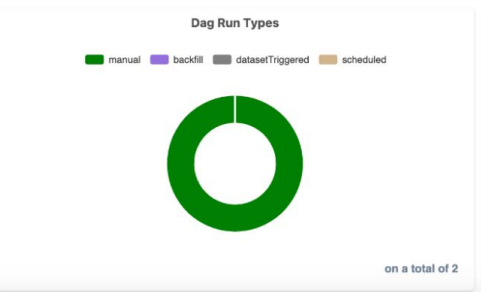
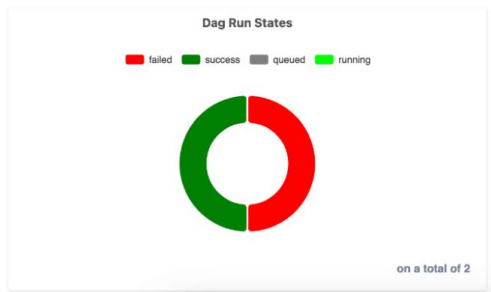
Scheduler
status: HEALTHY
last heartbeat: 2023-08-09, 18:49:19 UTC

Triggerer
status: HEALTHY
last heartbeat: 2023-08-09, 18:49:19 UTC

Dag Processor
status: UNKNOWN

Historical metrics

Start Date: 08/08/2023, 06:47:37 PM | End Date: 08/09/2023, 06:47:37 PM | over the last 1d00:00:00 | [Clear Filters](#)





11/09/2023, 13:39:15 25 All Run Types All Run States [Clear Filters](#) Auto-refresh

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 Run Task
load 2023-09-11, 13:39:08 UTC pick_dest Clear task Mark state as... Filter Tasks

[Details](#) [Graph](#) [Gantt](#) [Code](#) [Logs](#)

[More Details](#) [Rendered Template](#) [XCom](#) [List Instances, all runs](#)

Task Instance Notes:

- on 2023-09-11T13:39:08: got a failure because of ...

Save Note Cancel

Task Instance Details

| | |
|--------------|---|
| Status | ■ failed |
| Task ID | pick_dest 🔗 |
| Run ID | dataset_triggered__2023-09-11T13:39:08.179701+00:00 🔗 |
| Operator | @task.branch |
| Trigger Rule | all_success |
| Duration | 00:00:00 |



```
from airflow.decorators import dag
from airflow.providers.slack.notifications.slack import send_slack_notification
from datetime import datetime

@dag(
    start_date=datetime(2023, 1, 1),
    on_failure_callback=[
        send_slack_notification(
            text="The DAG {{ dag.dag_id }} failed",
            channel="#monitoring",
            username="Airflow",
        )
    ],
):
    def load():
        ...
```



