



FOSDEM '24

Data Workflows

translating dbt to Apache Airflow

Tatiana Al-Chueyr ◎ Staff Software Engineer

ASTRONOMER

Brussels ◎ 24 February 2024

What is dbt?



Untitled — Edited

View Zoom Add Page Insert Table Chart Text Shape Media Comment Collaborate Format Document

customer_id,

```
min(order_date) as first_order,  
max(order_date) as most_recent_order,  
count(order_id) as number_of_orders  
from orders
```

group by customer_id

),

customer_payments as (

```
select  
    orders.customer_id,  
    sum(amount) as total_amount
```

from payments

left join orders on
 payments.order_id = orders.order_id

group by orders.customer_id

Text

Body*

Style Layout More

Font

Helvetica Neue

Regular 14 pt

B *I* U ~~S~~

Character Styles

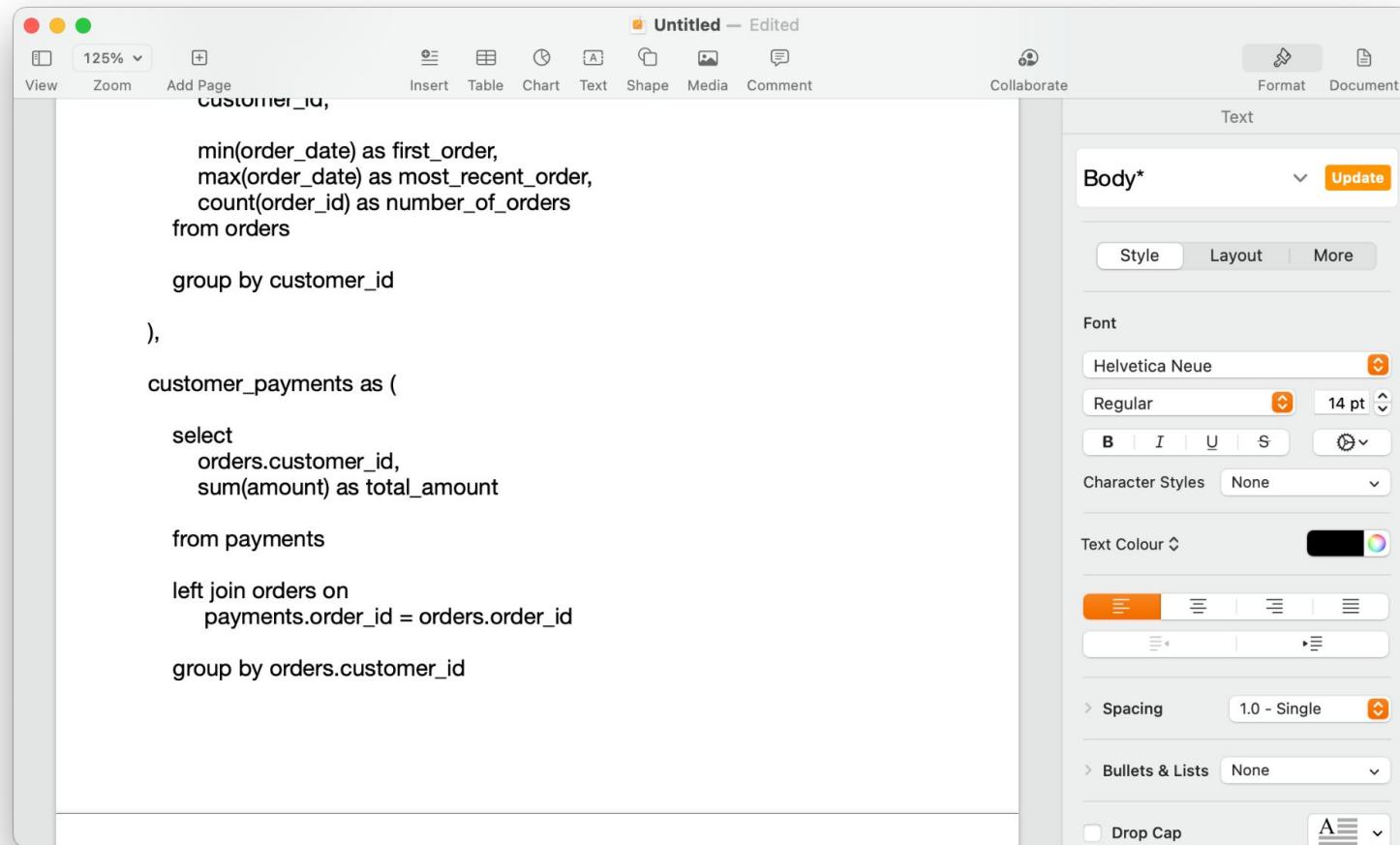
Text Colour

Text Alignment

Spacing

Bullets & Lists

Drop Cap

A screenshot of a Mac OS X-style document editor window titled "Untitled — Edited". The main text area contains a SQL query. The sidebar on the right is a "Text" panel with various styling options. At the top of the sidebar are buttons for "Style", "Layout", and "More". Below that is a "Font" section with a dropdown menu set to "Helvetica Neue", a font size dropdown set to "14 pt", and style buttons for bold (B), italic (I), underline (U), and strikethrough (S). There is also a "Character Styles" dropdown set to "None". Further down are sections for "Text Colour" (black), "Text Alignment" (with buttons for left, center, right, and justify, and sub-buttons for align, center, right, and justify), "Spacing" (set to "1.0 - Single"), "Bullets & Lists" (set to "None"), and "Drop Cap" (unchecked). A small "A" icon with a drop-down arrow is at the bottom right of the sidebar.



dbt (Data Build Tool) Core is an **open-source** tool for **data transformations** and analysis, using **SQL**

Growing in popularity as a standard
for sql analysts and data mart
builders

250

Contributors

6K

Total Commits

6.5K

Github Stars



dbt is the T in ELT. Organize, cleanse, denormalize, filter, rename, and pre-aggregate the raw data in your warehouse so that it's ready for analysis.

Why using dbt & Airflow?





Airflow is the de facto standard for job scheduling and workflow management

Strong Community of Data
Professionals Who Know Airflow

12M Monthly
Downloads

62% Downloads
are Airflow 2

Pace of Innovation Only
Accelerating, Quarterly Releases

2.2K Contributors

17K Total Commits

896 Commits in
Last 90 Days

Usage is Growing Exponentially,
Airflow 2 Changed the Trajectory

27K GitHub Stars

26K Slack Community

Airflow & dbt Core (OSS) high-level comparison

Airflow

- **Python** based and is meant for authoring, scheduling, and monitoring workflows
- **Flexible** and can be used for a wider range of tasks and use cases
- **Complex** interface and requires a deeper understanding of workflow management to write SQL transformations

dbt Core

- **SQL** based focused specifically on transforming and analyzing data
- **Specialized** and provides a more focused set of features and tools for working with data in a data warehouse
- **Simple** interface for working with data and SQL transformations.

Where dbt beats Airflow

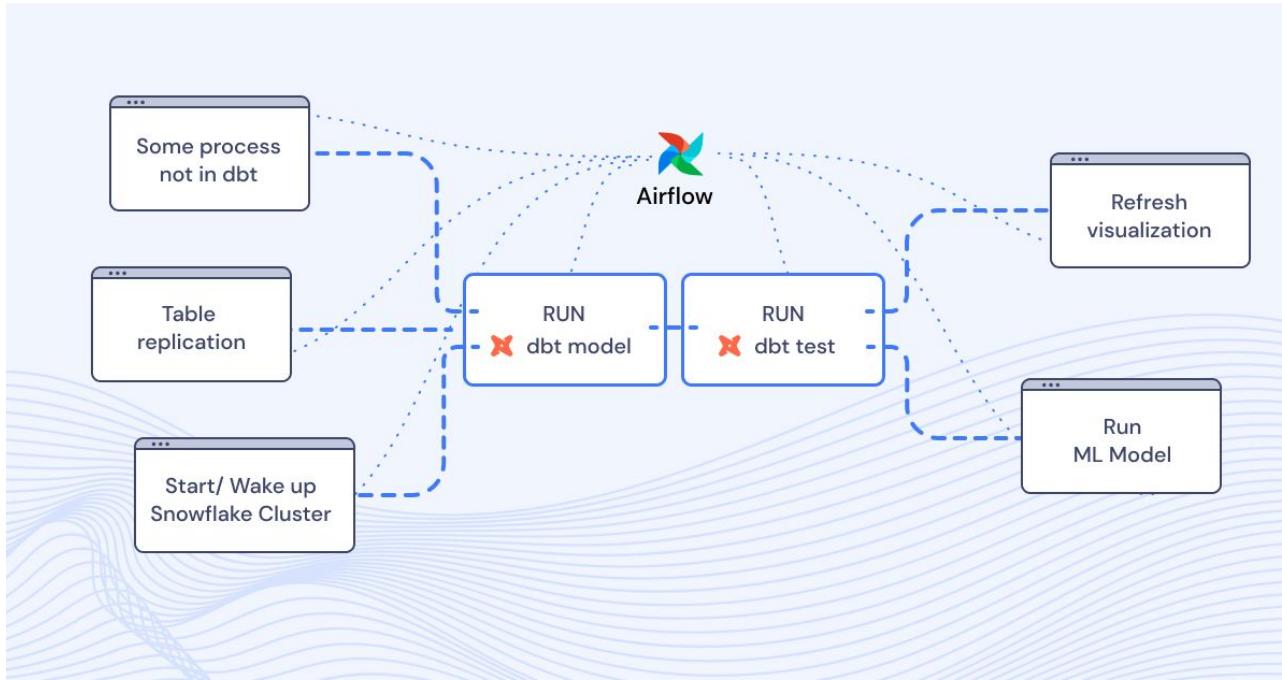
- Robust **suite of declarative tests** for each of your SQL models.
- **Dependency Management** dependencies between SQL models are automatically defined via Jinja templating.
- Easily assign **model schemas** through declarative yaml

Where Airflow beats dbt

- Flexibility in defining the “E” & “L” steps of ETL.
- As much as Airflow’s Complex Infrastructure can be a double-edged sword, it does allow for ML Operations to be run (against an Kubernetes infrastructure). There’s simply **more** that you can do with Airflow than dbt
- Using Airflow, you can achieve **everything** that dbt does, it’d just require more code maintenance (and technical know-how) to do it.

Unless you’re on Astro + Cosmos ;)

Why compare? Let's use both

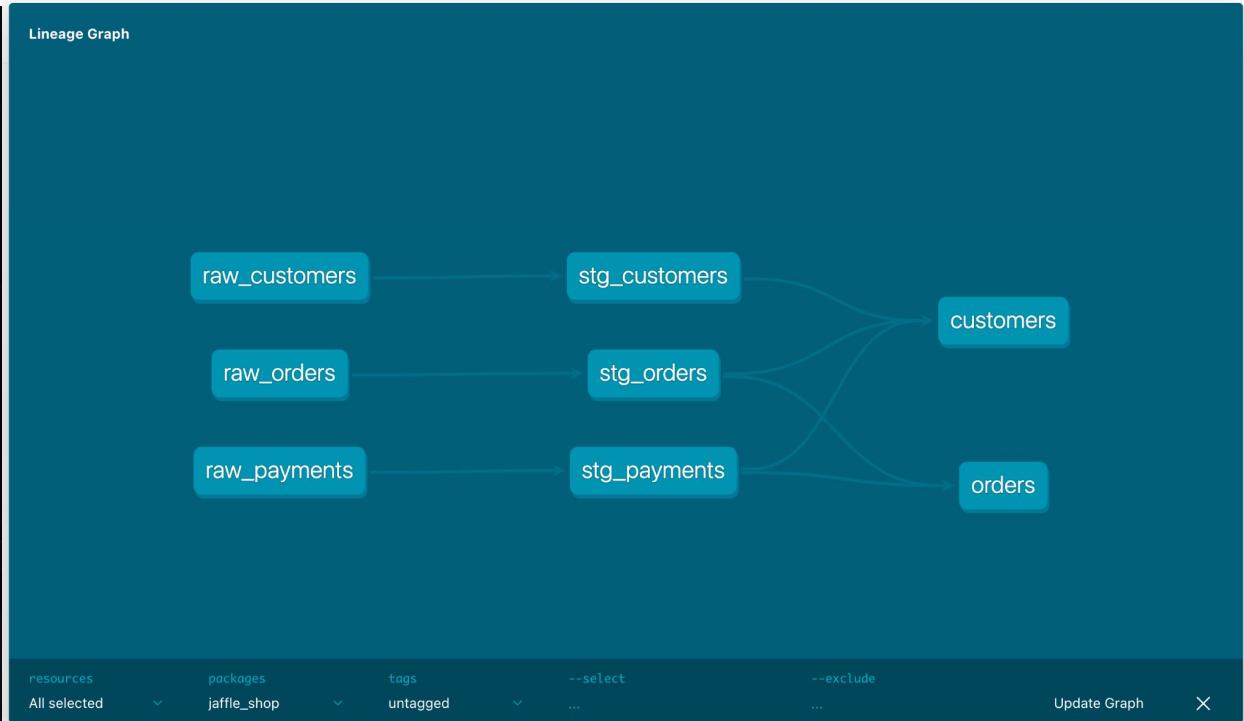


How to use dbt & Airflow?



Just like Airflow, dbt has DAGs

```
├── LICENSE  
├── README.md  
├── dbt_project.yml  
├── macros  
|   └── drop_table.sql  
├── models  
|   ├── customers.sql  
|   ├── docs.md  
|   ├── orders.sql  
|   ├── overview.md  
|   ├── schema.yml  
|   └── staging  
|       ├── schema.yml  
|       ├── stg_customers.sql  
|       ├── stg_orders.sql  
|       └── stg_payments.sql  
├── packages.yml  
├── profiles.yml  
└── seeds  
    ├── raw_customers.csv  
    ├── raw_orders.csv  
    └── raw_payments.csv
```



Just like Airflow, dbt has database connections

```
├── LICENSE
├── README.md
├── dbt_project.yml
└── macros
    └── drop_table.sql
└── models
    ├── customers.sql
    ├── docs.md
    ├── orders.sql
    ├── overview.md
    ├── schema.yml
    └── staging
        ├── schema.yml
        ├── stg_customers.sql
        ├── stg_orders.sql
        └── stg_payments.sql
└── packages.yml
└── profiles.yml
└── seeds
    ├── raw_customers.csv
    ├── raw_orders.csv
    └── raw_payments.csv
```

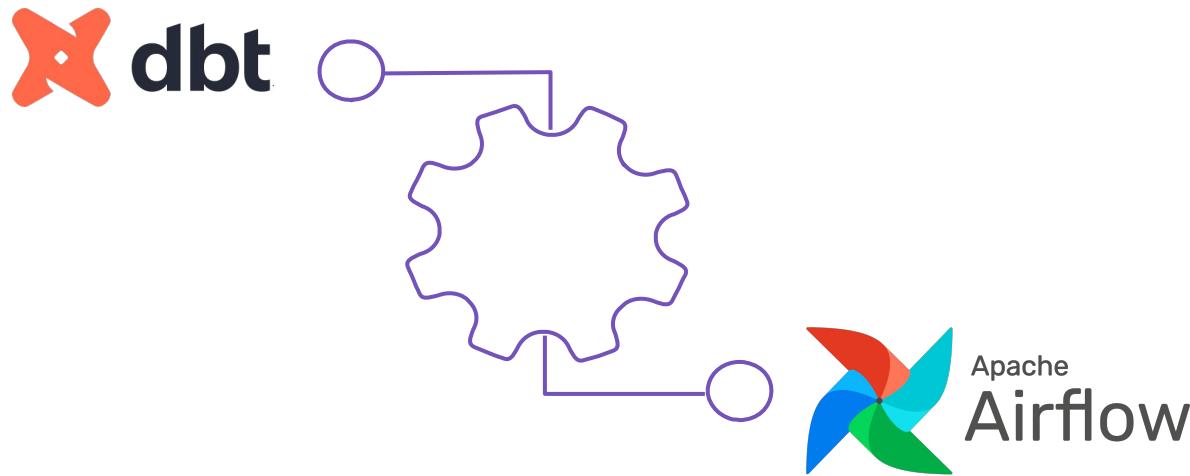
profiles.yml (~/Code/astron.../dbt/jaffle_shop_demo) - VIM

```
1 jaffle_shop:
2   target: dev
3   outputs:
4     dev:
5       type: postgres
6       host: "{{ env_var('POSTGRES_HOST') }}"
7       user: "{{ env_var('POSTGRES_USER') }}"
8       password: "{{ env_var('POSTGRES_PASSWORD') }}"
9       port: "{{ env_var('POSTGRES_PORT') | int }}"
10      dbname: postgres
11      schema: public
12      threads: 4
```

12,17

All

How can we bring them **together**?



DAG

≡ ~~Google~~ Translate



Text

Images

Documents

Websites

Detect language



Polish English



Portuguese

Lineage Graph



resources packages tags --select-- exclude
All selected jaffle_shop untagged Update Graph X

Translation

Send feedback

There are a few translation approaches

The screenshot shows a web browser window with the title "How Airflow + dbt Work Together" and the URL "getdbt.com/blog/dbt-airflow/". The main content is titled "Implementation" and includes sections for "Airflow + dbt Cloud" and "Airflow + dbt Core", each with bullet points. Below these are sections for "Other Perspectives" and "Audience Q&A". To the right of the main content is a sidebar with the dbt logo and three benefits: "Accelerate speed to insight", "Democratize data responsibly", and "Build trust in data across business". It also features a "Create a free account" button, a "Book a demo" link, and a call to action for Coalesce 2023.

For those who are ready to move on to configuration, below are guides to each approach:

Airflow + dbt Cloud

- Install the [dbt Cloud Provider](#), which enables you to orchestrate and monitor dbt jobs in Airflow without needing to configure an API
- Step-by-step [tutorial](#) with video
- [Code examples](#) for a quick start in your local environment

Airflow + dbt Core

- [Considerations](#) for using the dbt CLI + BashOperator, or using the KubernetesPodOperator for each dbt job

Other Perspectives

- [Shopify Engineering](#) recently shared [lessons learned from running Apache Airflow at scale](#), to much [discussion](#) from others in the data-engineering community.
- [Gitlab](#) open-sources their data engineering infrastructure, including comprehensive [docs](#) and [examples](#) of how they use dbt Core with Airflow.

Audience Q&A

<https://www.getdbt.com/blog/dbt-airflow/>

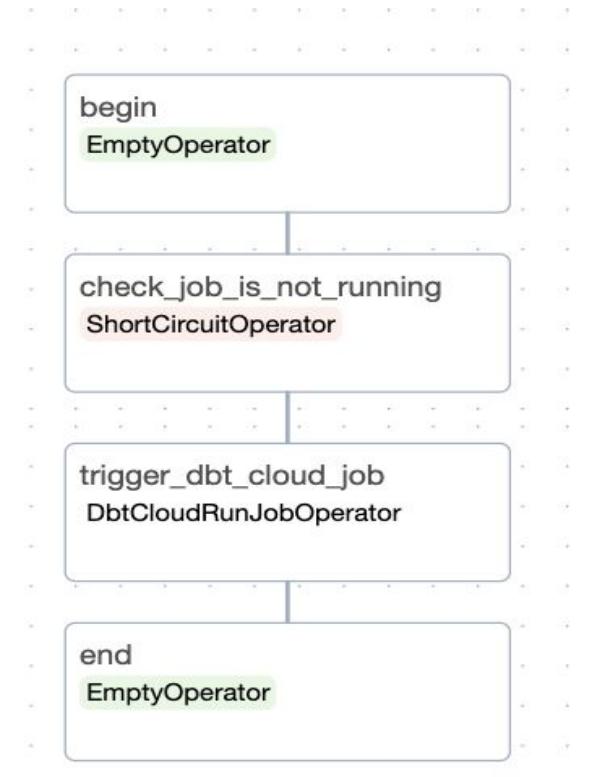
apache-airflow-providers-dbt-cloud

```
@dag(
    start_date=datetime(2022, 2, 10),
    schedule_interval="@daily",
    catchup=False,
    default_view="graph",
    doc_md=__doc__,
)
def check_before_running_dbt_cloud_job():
    begin, end = [EmptyOperator(task_id=id) for id in ["begin", "end"]]

    check_job = ShortCircuitOperator(
        task_id="check_job_is_not_running",
        python_callable=_check_job_not_running,
        op_kwargs={"job_id": JOB_ID},
    )

    trigger_job = DbtCloudRunJobOperator(
        task_id="trigger_dbt_cloud_job",
        dbt_cloud_conn_id=DBT_CLOUD_CONN_ID,
        job_id=JOB_ID,
        check_interval=600,
        timeout=3600,
    )

    begin >> check_job >> trigger_job >> end
```



https://registry.astronomer.io/dags/dbt_cloud_operational_check/versions/3.0.0

BashOperator

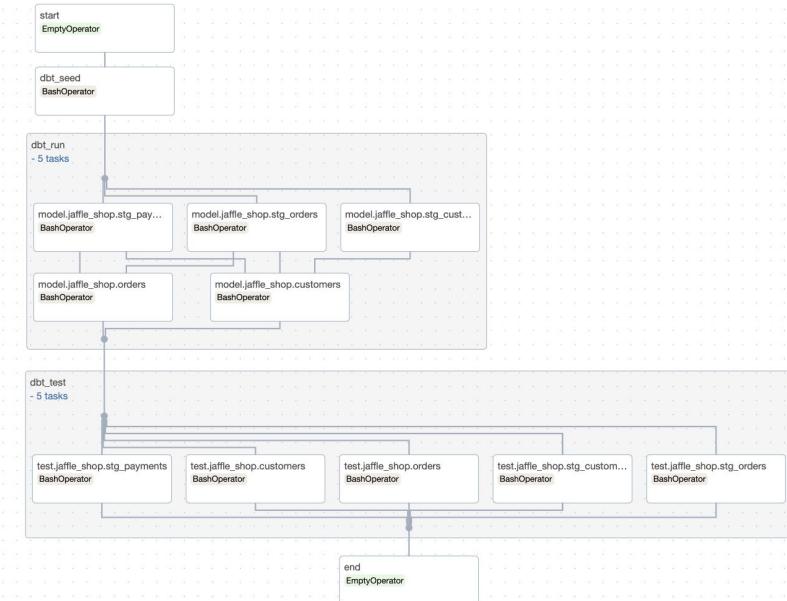
```
with DAG(
    "dbt_basic_dag",
    start_date=datetime(2020, 12, 23),
    description="A sample Airflow DAG to invoke dbt runs using a
BashOperator",
    schedule_interval=None,
    catchup=False,
) as dag:
    dbt_seed = BashOperator(
        task_id="dbt_seed",
        bash_command=f"dbt seed --profiles-dir {DBT_PROJECT_DIR}
--project-dir {DBT_PROJECT_DIR}",
    )
    dbt_run = BashOperator(
        task_id="dbt_run",
        bash_command=f"dbt run --profiles-dir {DBT_PROJECT_DIR}
--project-dir {DBT_PROJECT_DIR}",
    )
    dbt_test = BashOperator(
        task_id="dbt_test",
        bash_command=f"dbt test --profiles-dir {DBT_PROJECT_DIR}
--project-dir {DBT_PROJECT_DIR}",
    )
    dbt_seed >> dbt_run >> dbt_test
```



https://registry.astronomer.io/dags/dbt_basic/versions/1.0.3

dbt manifest parsing + BashOperator

```
with DAG(
    "dbt_advanced_dag_utility",
    start_date=datetime(2020, 12, 23),
    description="A dbt wrapper for Airflow using a utility class",
    schedule_interval=None,
    catchup=False,
    doc_md=__doc__
) as dag:
    start_dummy = DummyOperator(task_id="start")
    dbt_seed = BashOperator(
        task_id="dbt_seed",
        bash_command=(
            f"dbt {DBT_GLOBAL_CLI_FLAGS} seed "
            f"--profiles-dir {DBT_PROJECT_DIR} --project-dir {DBT_PROJECT_DIR}"
        ),
    )
    end_dummy = DummyOperator(task_id="end")
    dag_parser = DbtDagParser(
        dbt_global_cli_flags=DBT_GLOBAL_CLI_FLAGS,
        dbt_project_dir=DBT_PROJECT_DIR,
        dbt_profiles_dir=DBT_PROJECT_DIR,
        dbt_target=DBT_TARGET,
    )
    dbt_run_group = dag_parser.get_dbt_run_group()
    dbt_test_group = dag_parser.get_dbt_test_group()
    start_dummy >> dbt_seed >> dbt_run_group >> dbt_test_group >> end_dummy
```



https://registry.astronomer.io/dags/dbt_advanced_utility/versions/1.0.3

Approaches comparison

dbt (Cloud) provider

- Minimalistic DAG
- Hard to identify failing dbt node
- Inefficient retry (re-run all dbt nodes)
- + Trivial DAG parsing
- + Few worker slots
- + Asynchronous or Synchronous
- Vendor lock-in

BashOperator (one task per cmd)

- Minimalistic DAG
- Hard to identify failing dbt node
- Inefficient retry (re-run all dbt nodes)
- + Trivial DAG parsing
- + Few worker slots
- Downstream use cases dependent on every dbt node succeeding

BashOperator (multiple tasks)

- + Detailed DAG similar to dbt
- + Failing dbt node is easy to identify
- + Efficient retries
- DAG parsing can become slow
- Worker slots grows with dbt nodes
- + Independent downstream use cases can succeed

Alternative approaches

- Translate DAG without dbt manifest
- Translate DAG using dynamic task mapping to group dbt nodes
- Pre-generate static DAG translating all tasks of interest
- Execute dbt commands using different operators
 - KubernetesPodOperator
 - DockerOperator
- Execute dbt compiled SQL using Airflow database-specific hooks
- ...



Astronomer **Cosmos**

“Open-source library that allows you to run
dbt Core projects as
Airflow DAGs and **Task Groups**
with a **few lines of code**.”

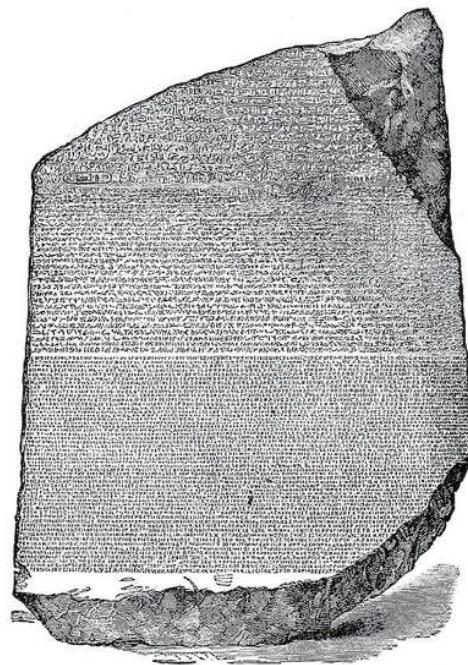
```
$ pip install astronomer-cosmos
```

<https://github.com/astronomer/astronomer-cosmos>

https://astronomer.github.io/astronomer-cosmos/getting_started/astro.html



Astronomer **Cosmos**



<https://www.britishmuseum.org/blog/everything-you-ever-wanted-know-about-rosetta-stone>

Cosmos

≡ Google Translate



Text

Images

Documents

Websites

Detect language



Polish English

Lineage Graph



resources

packages

tags

--select

--exclude

...

...

Update Graph X



customers
- SUCCESS
+ 2 tasks

orders
- SUCCESS
+ 2 tasks

Send feedback

Translating from dbt to Airflow with Cosmos

```
import os
from datetime import datetime
from pathlib import Path
from cosmos import DbtDag, ProjectConfig, ProfileConfig
from cosmos.profiles import PostgresUserPasswordProfileMapping

DEFAULT_DBT_ROOT_PATH = Path(__file__).parent / "dbt"
DBT_ROOT_PATH = Path(os.getenv("DBT_ROOT_PATH", DEFAULT_DBT_ROOT_PATH))

profile_config = ProfileConfig(
    profile_name="jaffle_shop",
    target_name="dev",
    profile_mapping=PostgresUserPasswordProfileMapping(
        conn_id="airflow_db",
        profile_args={"schema": "public"},
    ),
)
basic_cosmos_dag = DbtDag(
    project_config=ProjectConfig(
        DBT_ROOT_PATH / "jaffle_shop",
    ),
    profile_config=profile_config,
    schedule_interval="@daily",
    start_date=datetime(2023, 1, 1),
    catchup=False,
    dag_id="basic_cosmos_dag",
)
```

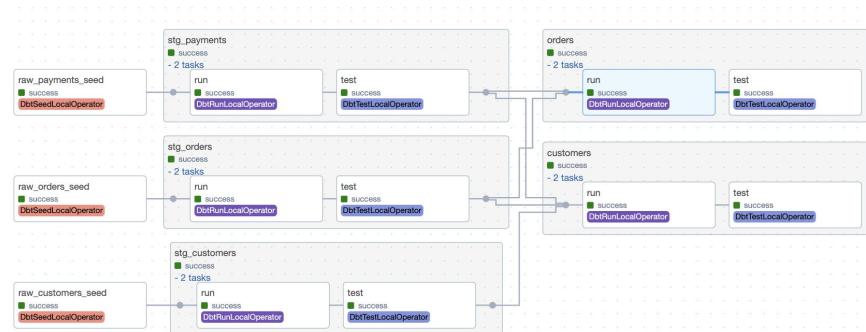


Translating from dbt to Airflow with Cosmos

```
import os
from datetime import datetime
from pathlib import Path
from cosmos import DbtDag, ProjectConfig, ProfileConfig
from cosmos.profiles import PostgresUserPasswordProfileMapping

DEFAULT_DBT_ROOT_PATH = Path(__file__).parent / "dbt"
DBT_ROOT_PATH = Path(os.getenv("DBT_ROOT_PATH", DEFAULT_DBT_ROOT_PATH))

profile_config = ProfileConfig(
    profile_name="jaffle_shop",
    target_name="dev",
    profile_mapping=PostgresUserPasswordProfileMapping(
        conn_id="airflow_db",
        profile_args={"schema": "public"},
    ),
)
basic_cosmos_dag = DbtDag(
    project_config=ProjectConfig(
        DBT_ROOT_PATH / "jaffle_shop",
    ),
    profile_config=profile_config,
    schedule_interval="@daily",
    start_date=datetime(2023, 1, 1),
    catchup=False,
    dag_id="basic_cosmos_dag",
)
```



Demo



Cosmos key features



- Easily bring your dbt core projects within **Astro/Airflow** projects
- Author **SQL models** as in plain dbt projects
- Render your project as an Airflow DAG or Task Group
- Flexibility on the translation method and dbt execution
- Schedule with Airflow's robust features: cron, datasets, timetables
- Visualize the SQL associated to an Airflow Task
- Skip the paid dbt cloud subscription
- Growing active open-source community

Cosmos dbt execution

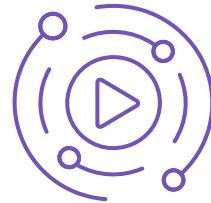


- Run SQL and Python dbt models (also deps, seeds, docs, ...)
- Customize the **arguments** used to run dbt
- Run dbt commands...
 - in the worker node, using `PythonOperator` subclasses
 - in the worker node, using `VirtualenvOperator` subclasses
 - in the worker node, using `DockerOperator` subclasses
 - remotely, using `KubernetesPodOperator` subclasses

https://astronomer.github.io/astronomer-cosmos/getting_started/execution-modes.html#local

<https://astronomer.github.io/astronomer-cosmos/configuration/operator-args.html>

https://astronomer.github.io/astronomer-cosmos/getting_started/execution-modes.html



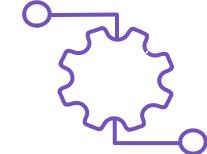
Cosmos dbt execution

- Customize the path to the **dbt binary** to avoid dependency conflicts

Airflow / DBT	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7
2.2				x	x	x	x	x
2.3	x	x		x	x	x	x	x
2.4	x	x	x					
2.5	x	x	x					
2.6	x	x	x	x	x			
2.7	x	x	x	x	x			
2.8	x	x	x	x	x		x	x

https://astronomer.github.io/astronomer-cosmos/getting_started/execution-modes-local-conflicts.html

Cosmos DAG translation



- Several **dbt DAG parsing strategies** are available
 - `dbt manifest.json`
 - `dbt ls`
 - `dbt ls file`
 - **custom parser**
 - **automatic**
- Configurable **Airflow DAG rendering**
 - several built-in operators
 - multiple strategies for **rendering test nodes**
 - **select** and **exclude** nodes using dbt selectors syntax
 - customize the translation by dbt resource type

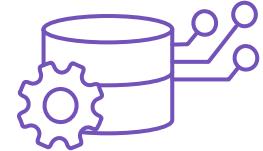
<https://astronomer.github.io/astronomer-cosmos/configuration/parsing-methods.html>

<https://astronomer.github.io/astronomer-cosmos/configuration/render-config.html>

<https://astronomer.github.io/astronomer-cosmos/configuration/testing-behavior.html>

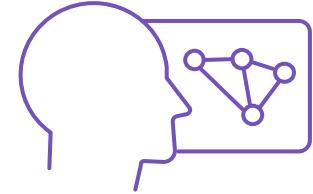
<https://astronomer.github.io/astronomer-cosmos/configuration/selecting-excluding.html>

Cosmos profile conversion



- Convert your Airflow connections into dbt profiles.yml using existing ProfileMapping classes
- Create custom ProfileMapping classes
- Bring your own dbt profiles.yml

<https://astronomer.github.io/astronomer-cosmos/profiles/index.html>



Cosmos user-centered

- **Visualize dbt resources and their lineage similar to dbt, but in Airflow**
- **Retry individual dbt nodes**
- **Visualize dbt compiled SQL in Airflow task instances**
- **Generate and export dbt docs to GCS, S3, Azure or customize**

DAG: basic_cosmos_dag

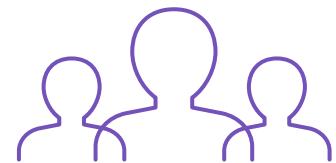
Task Instance: customers.run at 2023-09-13, 14:34:59

Rendered Template

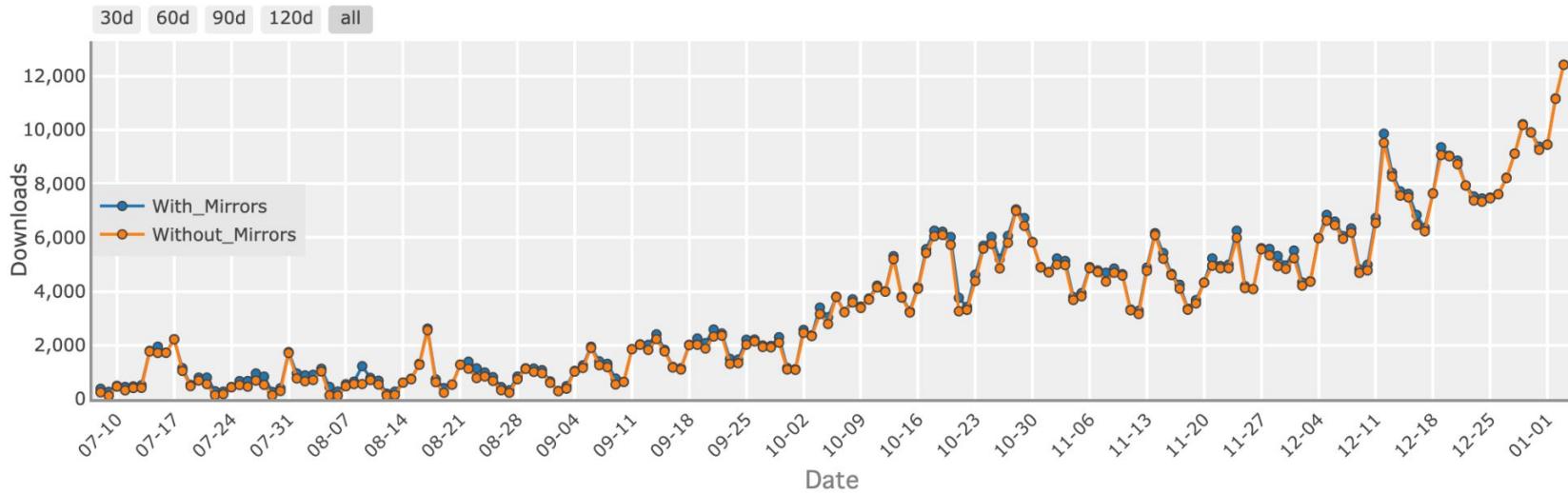
```
env
None
vars
None
compiled_sql
1 -- target/compiled/jaffle_shop/models/customers.sql
2 with customers as (
3
4     select * from "***"."public"."stg_customers"
5
6 ),
7
```

<https://astronomer.github.io/astronomer-cosmos/configuration/compiled-sql.html>
<https://astronomer.github.io/astronomer-cosmos/configuration/generating-docs.html>

Cosmos adoption

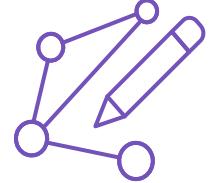


- **239k downloads** in a month (December 2023)
- **351 stars** in Github



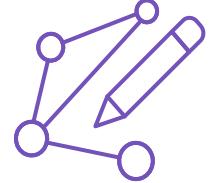
<https://pypistats.org/packages/astronomer-cosmos>

Cosmos next steps



- Showing [dbt](#) docs in the Airflow UI ([#737](#)) will be part of Cosmos 1.4)
- Strategies to improve [DAG](#) parsing performance
- Improve [openlineage](#) and dataset support
- Better support when [dbt](#) and [Airflow](#) are in separate repos
- Support running tasks in [dbt Cloud](#)

Cosmos next steps



A new version of Astronomer Runtime is available.
Version 7.6.0 was released on 2023-06-13, 00:00:00.

Ignore this update

dbt

Search for models...

Overview

Project Database Group

Tables and Views

- airflow
- public
- customers
- orders**
- raw_customers
- raw_orders
- raw_payments
- stg_customers
- stg_orders
- stg_payments

orders table

Details Description Columns Referenced By Depends On Code

Details

TAGS	PACKAGE	LANGUAGE	ACCESS	VERSION
untagged	jaffle_shop	sql	protected	

Description

This table has basic information about orders, as well as some derived facts based on payments

Columns

COLUMN	TYPE	DESCRIPTION	TESTS	MORE?
order_id		This is a unique identifier for an order	UN	>
customer_id		Foreign key to the customers table	NF	>
order_date		Date (UTC) that the order was placed		>
status		Orders can be one of the following sta...	A	>

Cosmos community

20 authors had 40 commits merged into main during November 2023.
Only 3 of these authors were Astronomer employees.

Since December 2022, 66 people contributed to the Cosmos repo.

There are 388 members in the #airflow-dbt Airflow Slack,
and daily interactions.



Cosmos references

- Intro to Cosmos website
- Github Repo astronomer/astronomer-cosmos
- Docs <https://astronomer.github.io/astronomer-cosmos/>
- Join the Apache Airflow slack #airflow-dbt channel
- More examples on how to use Cosmos
- Webinar “Introducing Cosmos” by Julian LaNeve

Note: Some of the slides were inspired by Julian LaNeve & Pádraic Slattery slides!



FOSDEM '24



thank you!

@tati_alchueyr

tatiana.alchueyr@astronomer.io

#airflow_dbt

ASTRONOMER

Brussels ◎ 24 February 2024