Boosting **dbt Core** workflows performance

Airflow Deferrable Capabilities

Pankaj Koti
Senior Software Engineer
@ Astronomer

Pankaj Singh
Senior Software Engineer
@ Astronomer

Tati Al-Chueyr
Principal Software Engineer
@ Astronomer

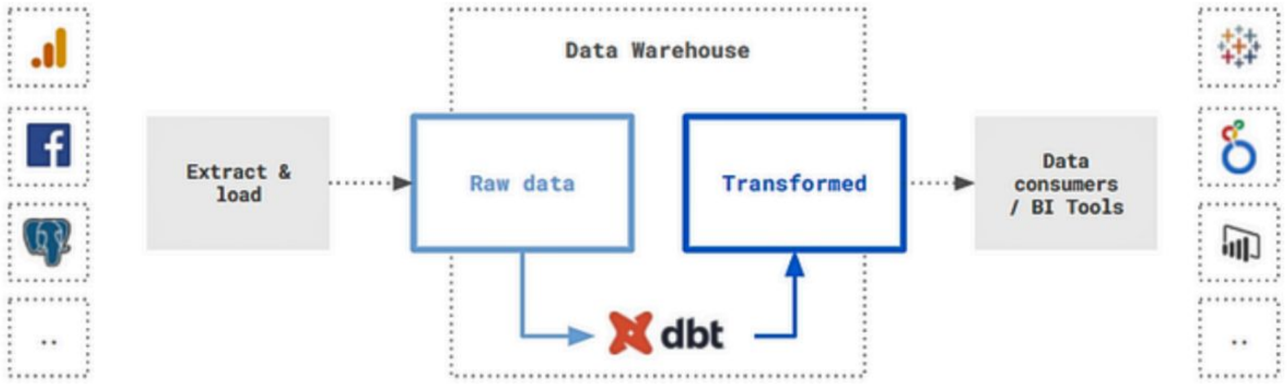
Airflow Summit, Seattle, USA - 8 October 2025

Agenda

1. What is dbt

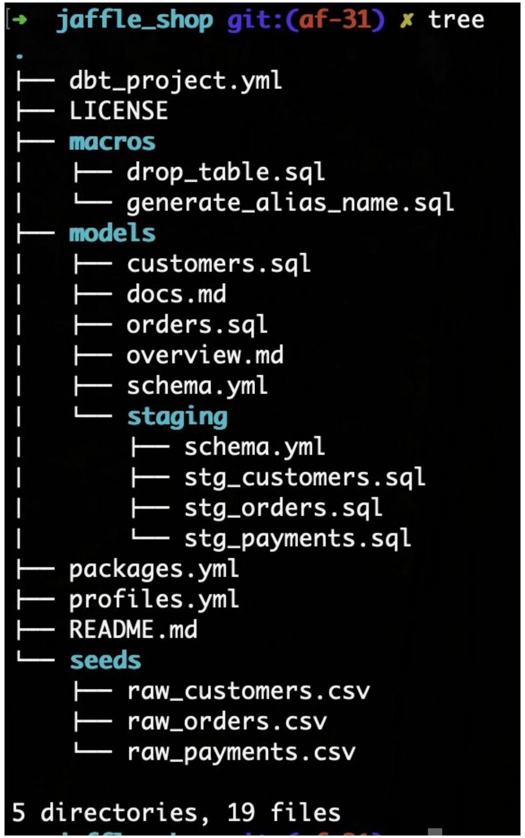
dbt Core what is it?

dbt Core is an open-source tool that empowers **data practitioners** to **transform**

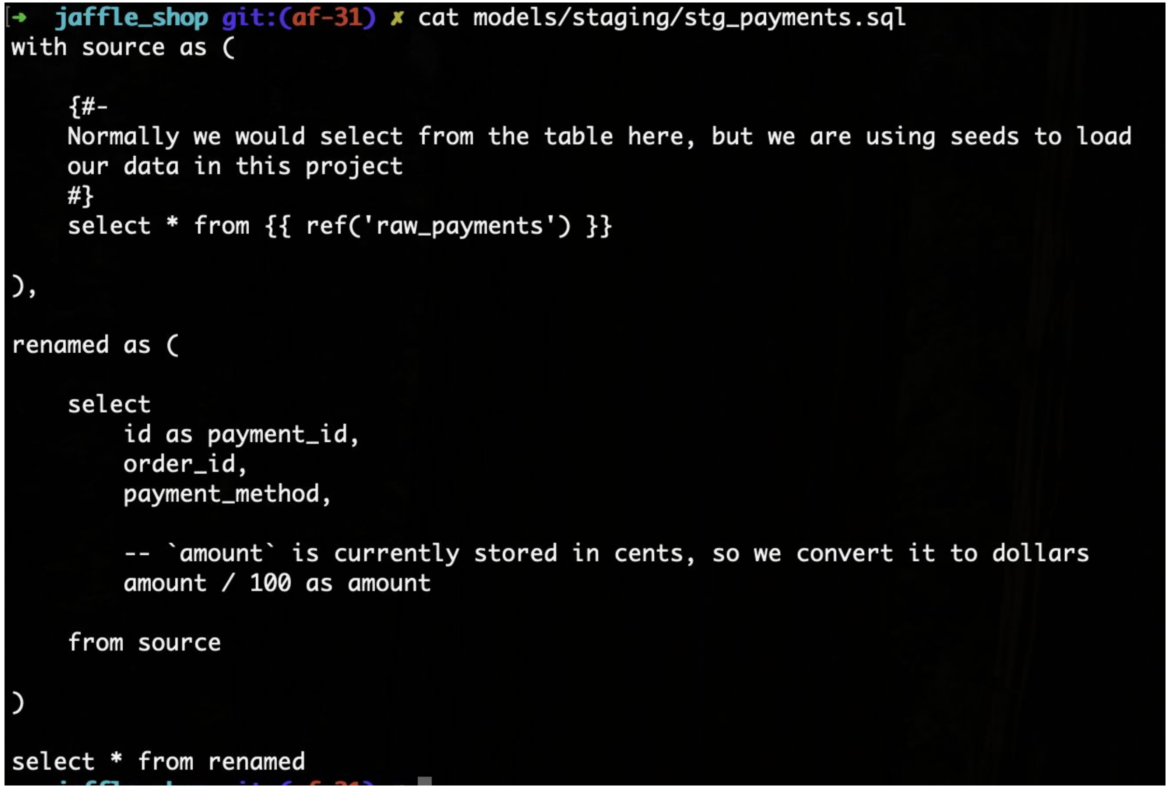


```
$ pip install dbt-core
```

dbt Core project structure & syntax

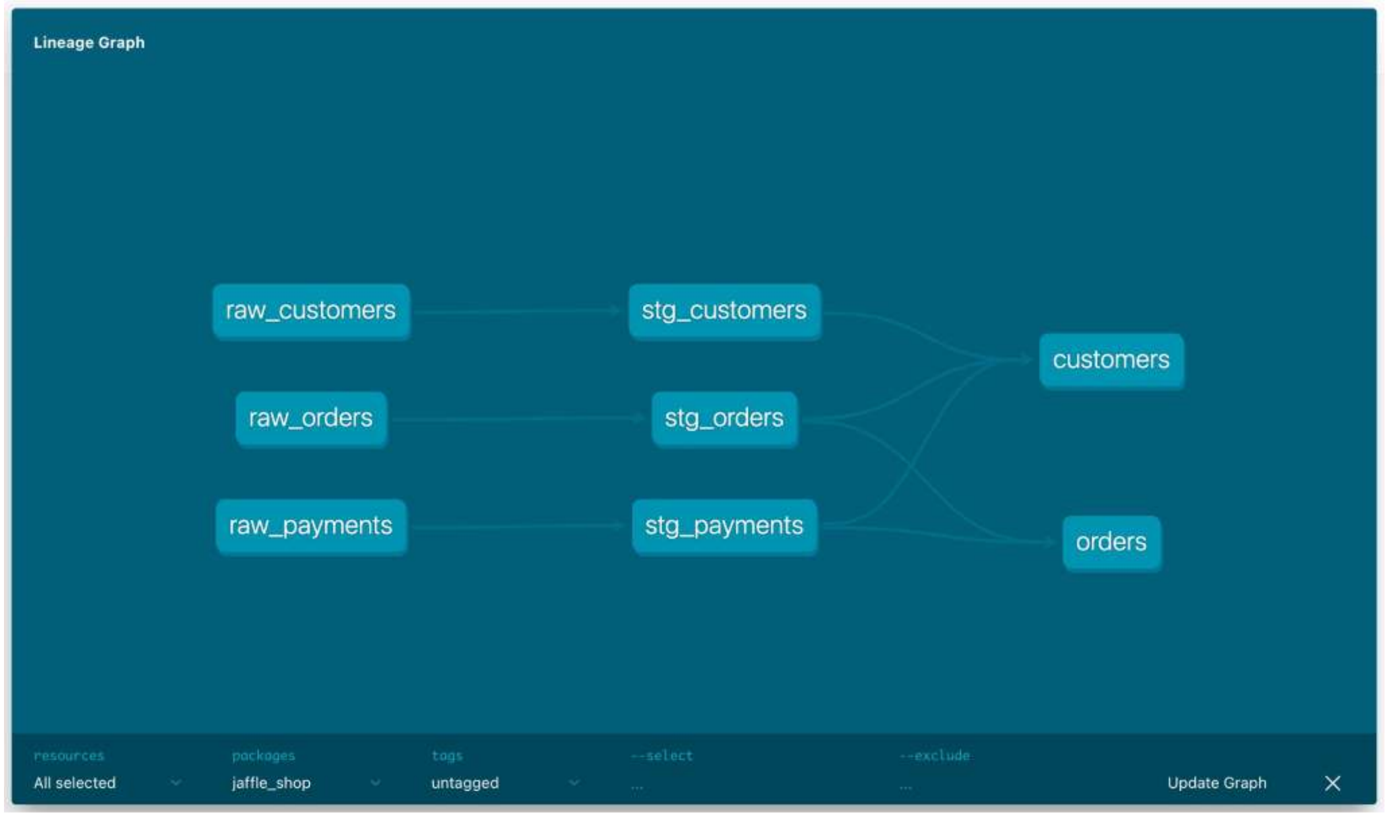


```
➜  jaffle_shop git:(af-31) ✗ tree
.
├── dbt_project.yml
├── LICENSE
├── macros
|   ├── drop_table.sql
|   └── generate_alias_name.sql
├── models
|   ├── customers.sql
|   ├── docs.md
|   ├── orders.sql
|   ├── overview.md
|   ├── schema.yml
|   └── staging
|       ├── schema.yml
|       ├── stg_customers.sql
|       ├── stg_orders.sql
|       └── stg_payments.sql
├── packages.yml
├── profiles.yml
├── README.md
└── seeds
    ├── raw_customers.csv
    ├── raw_orders.csv
    └── raw_payments.csv

5 directories, 19 files
```



```
➜  jaffle_shop git:(af-31) ✗ cat models/staging/stg_payments.sql
with source as (

    {#-
    Normally we would select from the table here, but we are using seeds to load
    our data in this project
    #}
    select * from {{ ref('raw_payments') }}

),

renamed as (

    select
        id as payment_id,
        order_id,
        payment_method,

        -- `amount` is currently stored in cents, so we convert it to dollars
        amount / 100 as amount

    from source

)

select * from renamed
```

dbt Core pipeline visualisation



resources: All selected
packages: jaffle_shop
tags: untagged
--select: ...
--exclude: ...

ASTRONOMER

dbt Core limitations

dbt Core does not...

- Schedule jobs in production
- Run scalable extract and load tasks
- Have centralised logging or alerts handling
- Make the project's documentation available to other team members

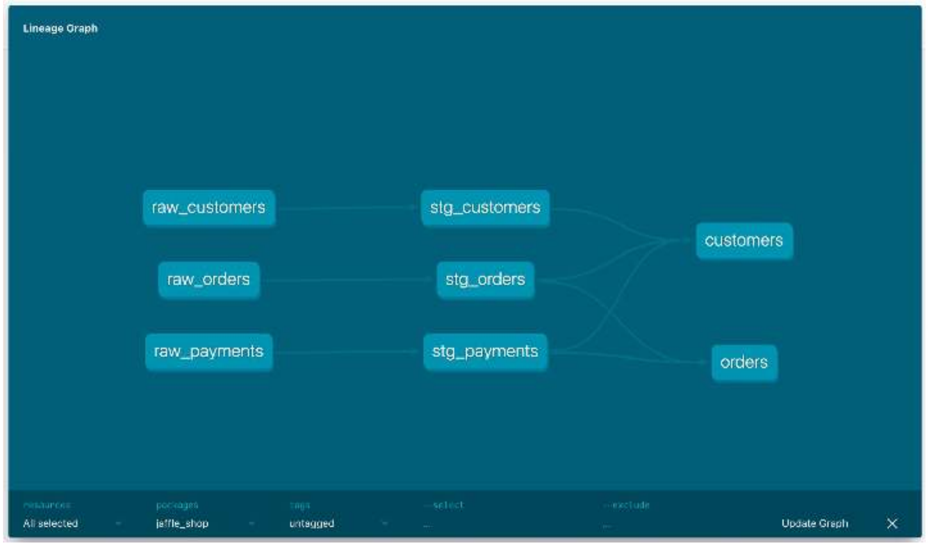
dbt Labs, creator and maintainer of dbt Core, solves these issues via their proprietary and commercial platform, dbt Cloud.

2. Why Cosmos

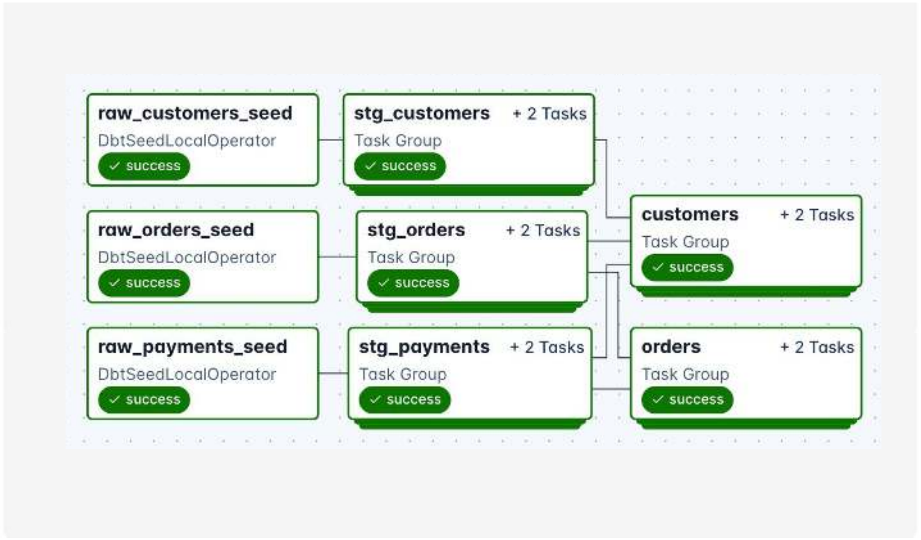
Why Cosmos?

Cosmos “magically” translates dbt pipelines in Airflow DAGs

dbt



Apache Airflow

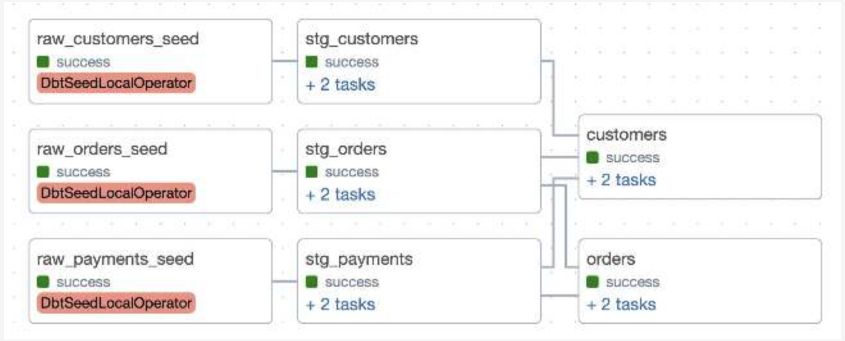


```
$ pip install astronomer-cosmos
```

Why Cosmos?

```python
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="@daily",
   start_date=datetime(2023, 1, 1),
   catchup=False,
   dag_id="basic_cosmos_dag",
)
```

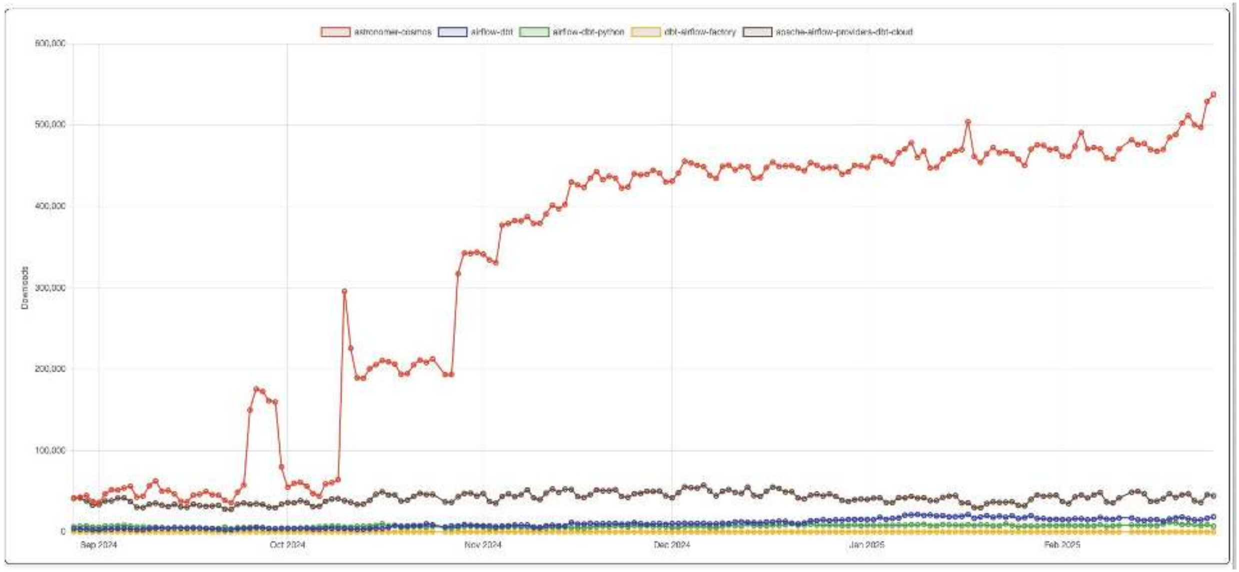
Apache Airflow



ASTRONOMER

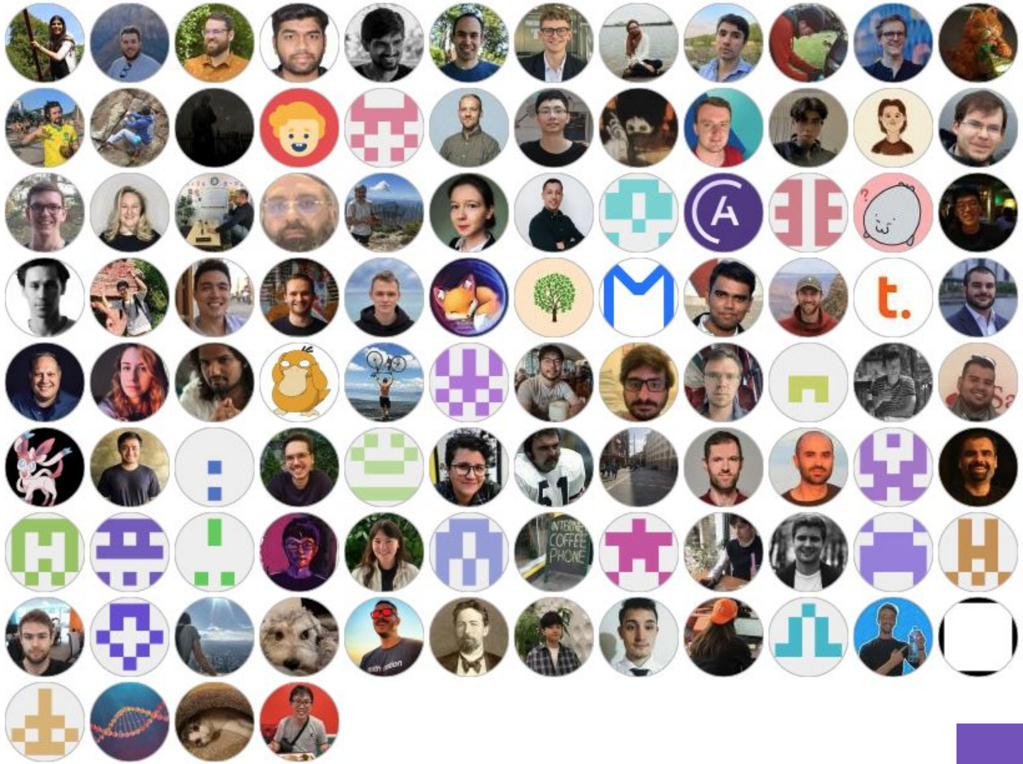
Cosmos adoption

- 20M+ downloads in PyPI per month (August-September 2025)
- 1k stars in Github
- Millions of Cosmos tasks are run every month (>20% Astro customers)



The driving force behind Cosmos

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



144 contributors

3. Performance challenge

Performance has been a priority

	1.2.5	1.3 (DBT LS FILE)	1.4	1.5	1.6
DAG Parsing time	00:00:08	00:00:02	00:00:07	00:00:02	00:00:02
Task Run time	00:00:09	00:00:08	00:00:06	00:00:05	00:00:04
Task Queue time	00:00:09	00:00:04	00:00:05	00:00:01	00:00:01
DAG Run time	00:01:29	00:00:55	00:01:18	00:00:43	00:00:42

Slides https://airflowsummit.org/slides/2024/40-Overcoming-Performance-Hurdles-in-Integrating-dbt-with-Airflow.pdf
Recording: https://www.youtube.com/watch?v=gnJPFGvqLzU

ASTRONOMER

The cost of running dbt commands

The following approaches accomplish the same outcome.
Which one is faster?

cmd: 1

```
$ dbt build
```

cmd: 3

```
$ dbt seed
$ dbt run
$ dbt test
```

#cmd: 13

```
$ dbt seed --select raw_customers
$ dbt seed --select raw_orders
$ dbt seed --select raw_payments

$ dbt run --select stg_customers
$ dbt run --select stg_orders
$ dbt run --select stg_payments
$ dbt run --select customers
$ dbt run --select orders

$ dbt test --select stg_customers
$ dbt test --select stg_orders
$ dbt test --select stg_payments
$ dbt test --select customers
$ dbt test --select orders
```

The cost of running dbt

These are the **total times** to run **the same example pipeline** (Jaffle Shop) with **dbt Core** 1.10.1 using **Snowflake** in **the three ways described** previously (using M1 Pro)

cmd: 1

```
$ dbt build
```

17s

cmd: 3

```
$ dbt seed
$ dbt run
$ dbt test
```

34s

#cmd: 13

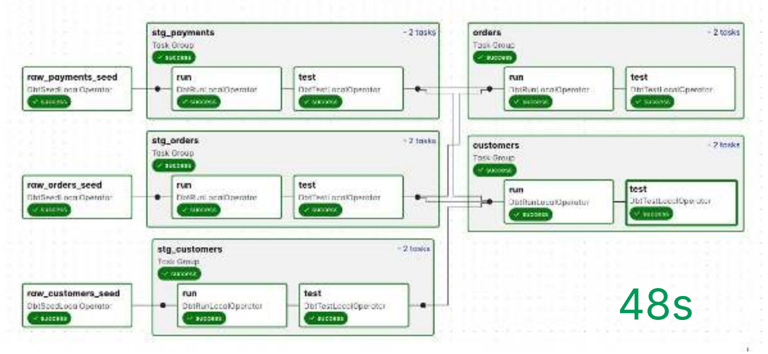
```
$ dbt seed --select raw_customers
$ dbt seed --select raw_orders
$ dbt seed --select raw_payments

$ dbt run --select stg_customers
$ dbt run --select stg_orders
$ dbt run --select stg_payments
$ dbt run --select customers
$ dbt run --select orders

$ dbt test --select stg_customers
$ dbt test --select stg_orders
$ dbt test --select stg_payments
$ dbt test --select customers
$ dbt test --select orders
```

61s

Reducing task execution time with Cosmos 1.10

```python
from datetime import datetime
from cosmos import DbtDag, ProjectConfig, ProfileConfig
from include.constants import jaffle_shop_path

project_config = ProjectConfig(
    dbt_project_path=jaffle_shop_path,
)

profile_config = ProfileConfig(
    profile_name="jaffle_shop",
    target_name="dev",
    profiles_yml_filepath=jaffle_shop_path / "profiles.yml",
)

snowflake_dag = DbtDag(
    project_config=project_config,
    profile_config=profile_config,
    start_date=datetime(2023, 1, 1),
    dag_id="snowflake_dag",
    tags=["profiles"],
)
```

Standard behaviour of running **one task per dbt node** in Cosmos

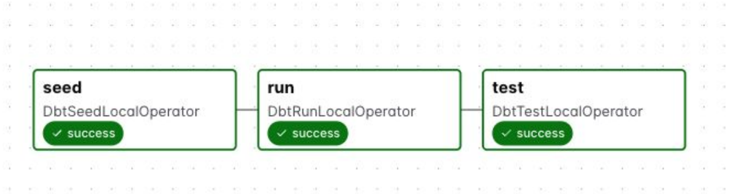


48s

Reducing task execution time with Cosmos 1.10

```python
from airflow.sdk import DAG
from cosmos import DbtRunLocalOperator, DbtSeedLocalOperator, DbtTestLocalOperator,
ProfileConfig

from include.constants import jaffle_shop_path


with DAG("snowflake_dag_per_resource") as dag:
    seed = DbtSeedLocalOperator(
        task_id="seed",
        profile_config=profile_config,
        project_dir=jaffle_shop_path,
    )

    run = DbtRunLocalOperator(
        task_id="run",
        profile_config=profile_config,
        project_dir=jaffle_shop_path,
    )

    test = DbtTestLocalOperator(
        task_id="test",
        profile_config=profile_config,
        project_dir=jaffle_shop_path,
    )

    seed >> run >> test
```

Running **one task per dbt node type** with Cosmos

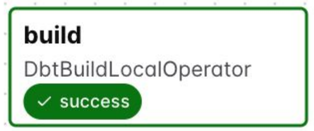


27s

Reducing task execution time with Cosmos 1.10

```python
from airflow.sdk import DAG

from cosmos import DbtBuildLocalOperator, ProfileConfig

from include.constants import jaffle_shop_path


profile_config = ProfileConfig(
    profile_name="jaffle_shop",
    target_name="dev",
    profiles_yml_filepath=jaffle_shop_path / "profiles.yml",
)

with DAG("snowflake_dag_single_task") as dag:
    build = DbtBuildLocalOperator(
        task_id="build",
        profile_config=profile_config,
        project_dir=jaffle_shop_path,
    )
    build
```

Running **one task** for the whole dbt workflow with Cosmos



20s

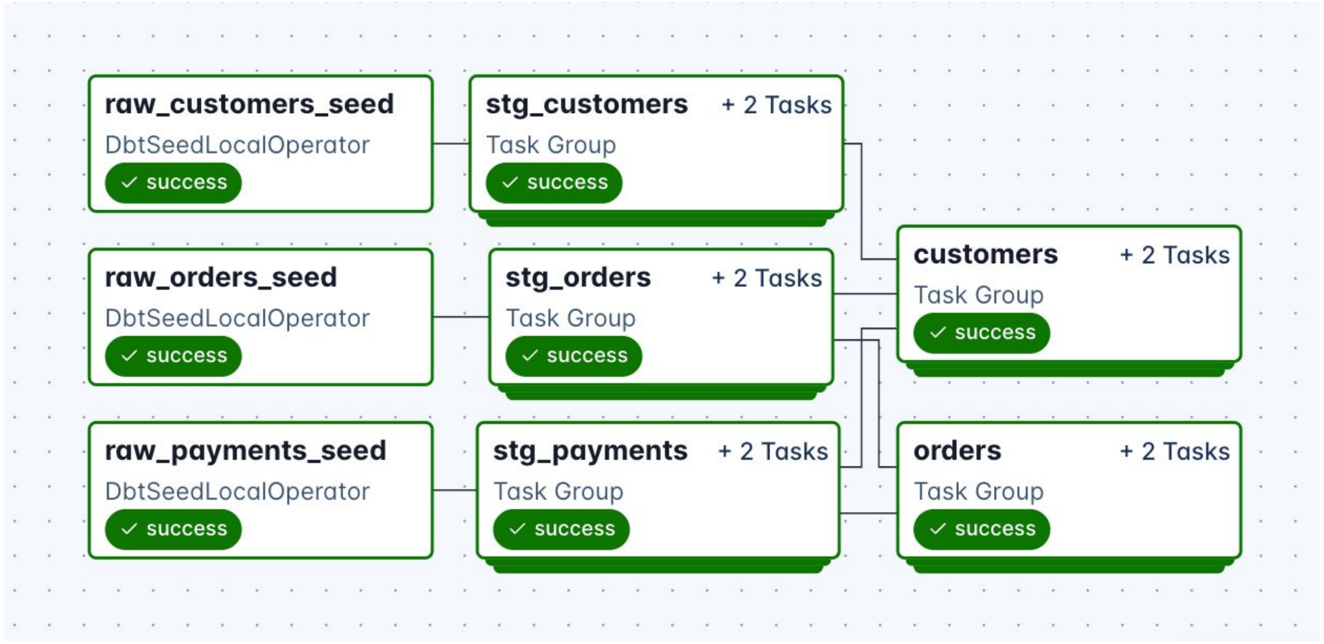
Cosmos & **dbt** performance

	1 command	**3 commands**	**13 commands**
dbt Core	17s	24s	61s
dbt Fusion	15s	26s	74s
Cosmos with dbt Core (*)	20s	27s	48s

(*) Using the `$ airflow dags test` command with Cosmos 1.11.0a1, Airflow 3.0.2, dbt Core 1.10 (installed in the same Python virtualenv as Airflow and Cosmos), against Snowflake. In practice, when using a production Airflow deployment such as Astro, the latency will be higher, due to their distributed nature.

Challenge

Is it possible to give users a **fine-grained visualisation** and **retry per model** capabilities without running a **dbt** command **for every seed or model**?

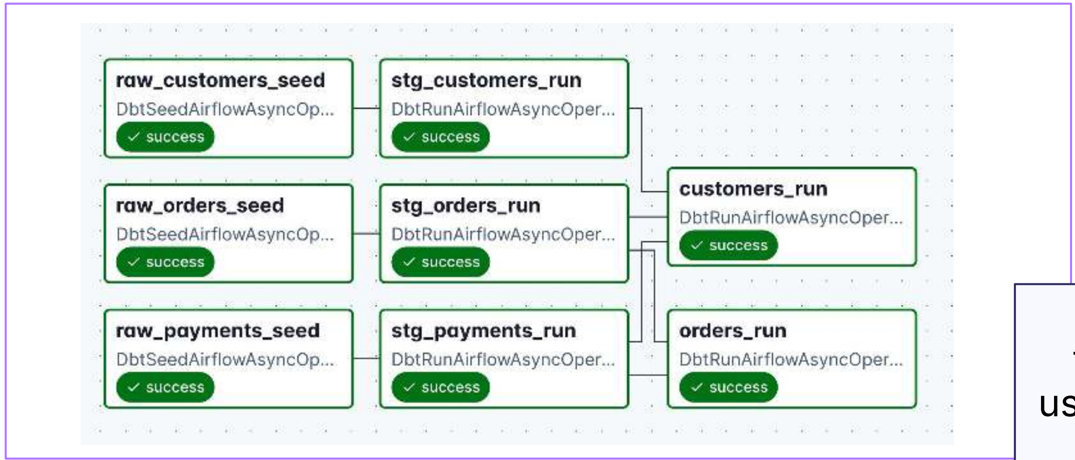


4. Strategy 1: Airflow Deferrable

Strategy 1 `ExecutionMode.AIRFLOW_ASYNC`

Setup task

pre-compile
SQL
with dbt

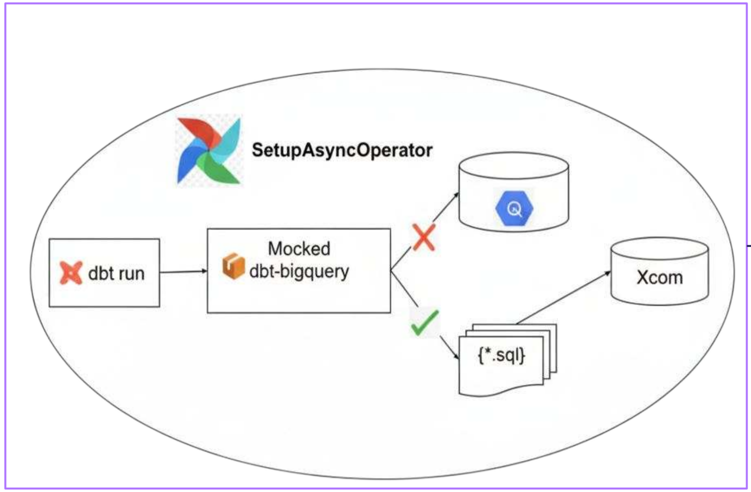


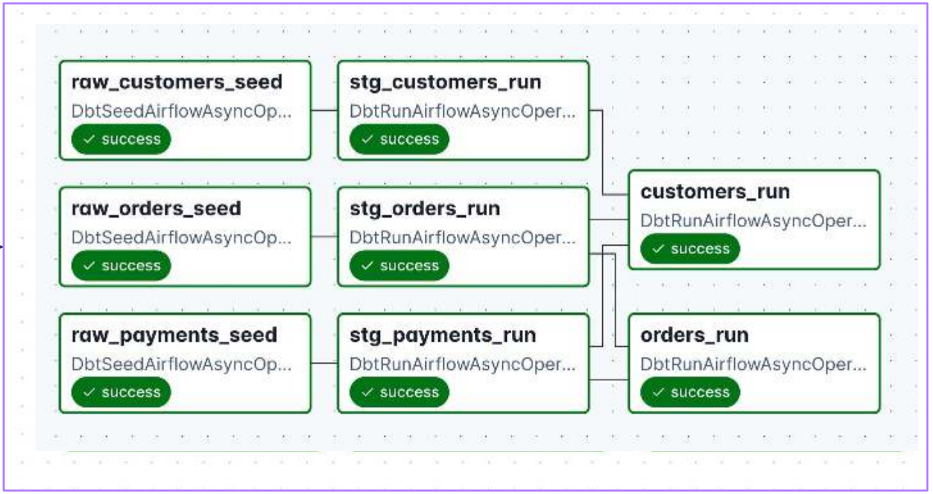
Run SQL
transformations
using Airflow native
deferrable
operators

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

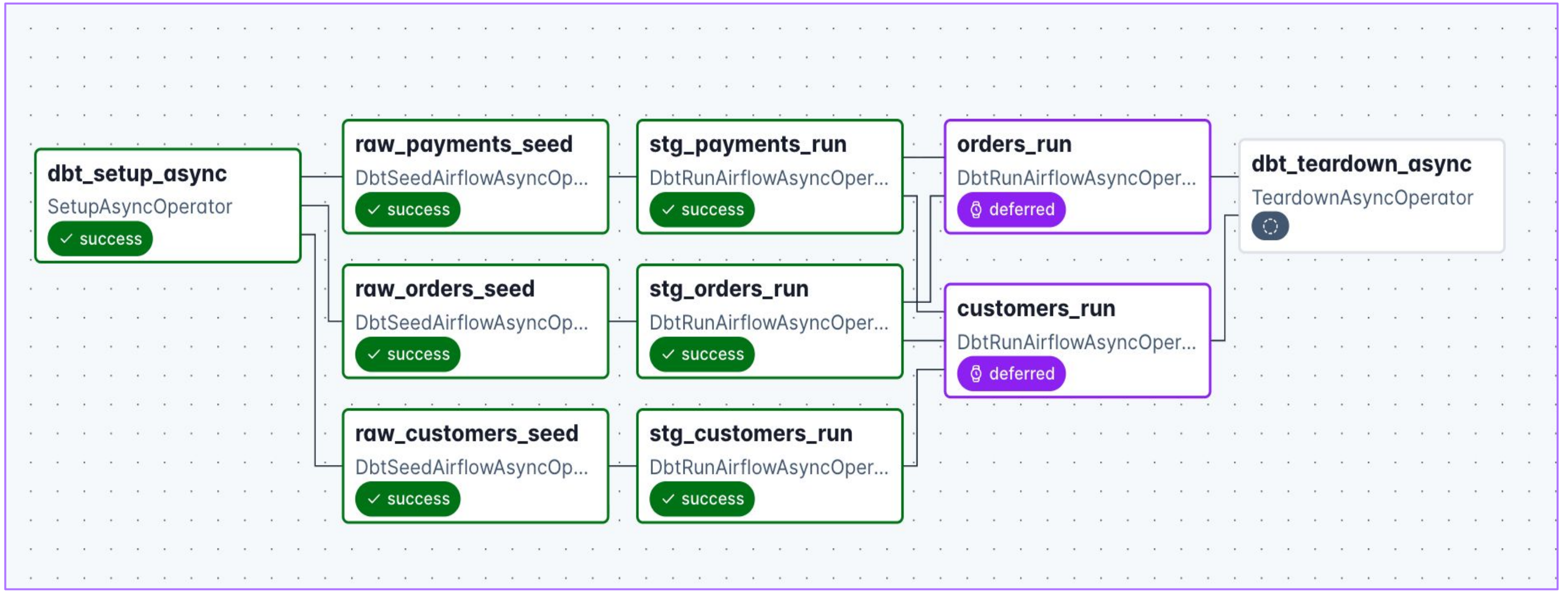
ASTRONOMER

Strategy 1 `ExecutionMode.AIRFLOW_ASYNC`





Strategy 1 `ExecutionMode.AIRFLOW_ASYNC`



Strategy 1 `ExecutionMode.AIRFLOW_ASYNC`
Performance improvements

Execution mode	Time to run project (minutes)
`dbt run`	**13**
Cosmos 1.9 (*) with **ExecutionMode.LOCAL** (Airflow with a local astro-cli setup)	**11**
Cosmos 1.9 (*) with **ExecutionMode.AIRFLOW_ASYNC** (Airflow with a local astro-cli setup)	**11**
Cosmos 1.11a6 (**) with **ExecutionMode.AIRFLOW_ASYNC** (Airflow with a local astro-cli setup)	**7**

(*) Using Cosmos 1.9 with Airflow 3.0.2, dbt Core 1.10 (installed in the same Python virtualenv as Airflow and Cosmos), against Postgres. We are using a dbt project that has 129 models.
(**) Cosmos 1.11a6 has two main improvements regarding the AIRFLOW_ASYNC, originally released in Cosmos 1.9:

- Use of XCom instead of Remote object store to exchange compiled SQL files #1934
- Reuse Python virtualenv across tasks running in the same worker node #1939

Strategy 1 `ExecutionMode.AIRFLOW_ASYNC`

Pros

- Reduced benchmark Airflow DAG run time by 36%
- Single dbt command invocation (less CPU/memory allocation)
- Non-blocking transformations in the data warehouse with Airflow deferrable operators

Cons

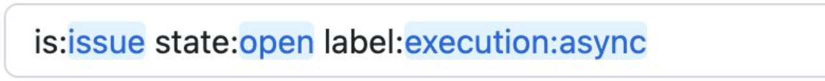
- Currently only supports dbt models
- Currently only supports BigQuery
- Implementation specific per data warehouse
- The dbt project cannot have models with a metadata-dependency on other models
- Some users reported issues related to Airflow deferrable operators, which can be challenging to reproduce
- Only works with dbt Core (not dbt Fusion)

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

Next steps

`ExecutionMode.AIRFLOW_ASYNC`

1. Available since Cosmos 1.9 for BigQuery
2. Significant performance improvements in Cosmos 1.11 pre-releases
3. We need feedback!
4. Maybe add non-async version
5. We've introduced telemetry to evaluate the adoption
6. We'd love contributions



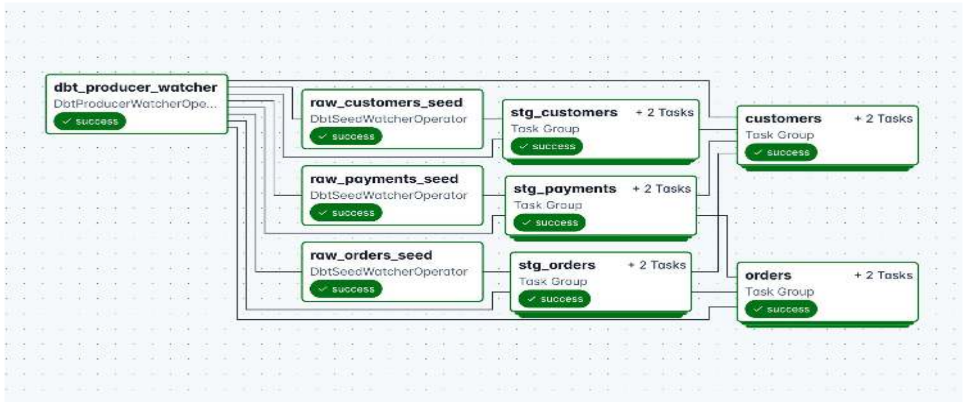
is:issue state:open label:execution:async

5. Strategy 2: Watcher

Strategy 2 `ExecutionMode.WATCHER`

Setup task

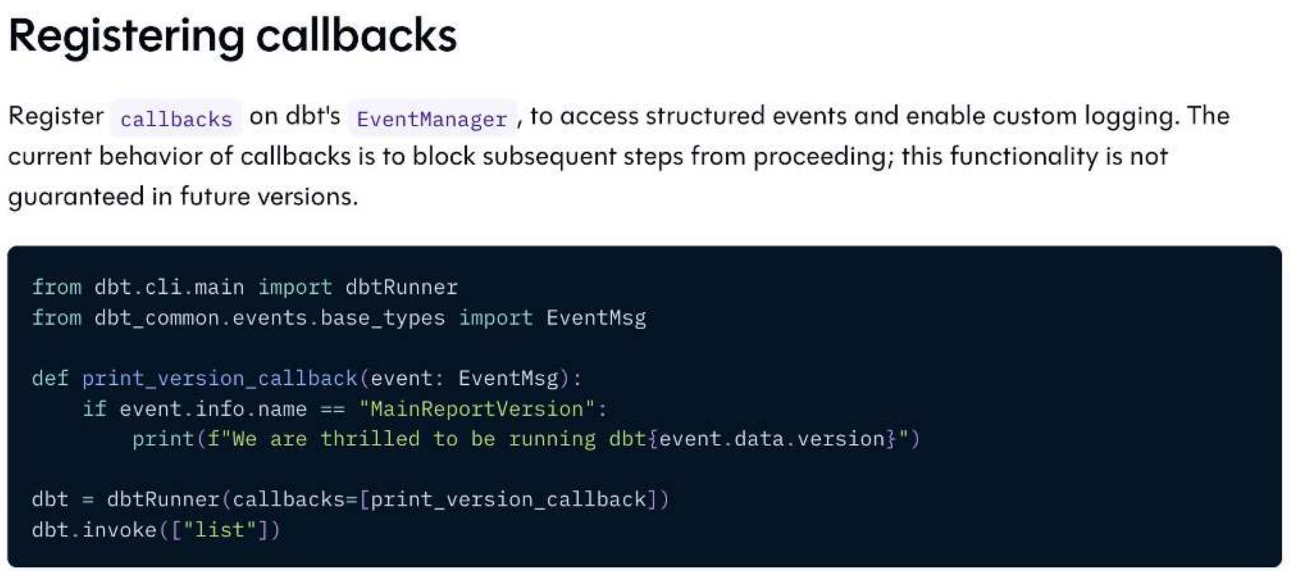
run dbt build in a single Airflow task



all the other tasks are sensors

https://github.com/astronomer/astronomer-cosmos/issues/1950

Strategy 2 `ExecutionMode.WATCHER`



Registering callbacks

Register `callbacks` on dbt's `EventManager`, to access structured events and enable custom logging. The current behavior of callbacks is to block subsequent steps from proceeding; this functionality is not guaranteed in future versions.

```python
from dbt.cli.main import dbtRunner
from dbt_common.events.base_types import EventMsg

def print_version_callback(event: EventMsg):
    if event.info.name == "MainReportVersion":
        print(f"We are thrilled to be running dbt{event.data.version}")

dbt = dbtRunner(callbacks=[print_version_callback])
dbt.invoke(["list"])
```

https://docs.getdbt.com/reference/programmatic-invocations#registering-callbacks

Strategy 2 `ExecutionMode.WATCHER`
Performance improvements

Execution mode	Number of threads	Time to run project (minutes)
`dbt build` in the CLI	4	**6 - 7**
`dbt run` for each model individually		**30**
Cosmos default **ExecutionMode.LOCAL** in Astro CLI locally		**10 - 15**
Cosmos proposed **ExecutionMode.WATCHER** in Astro CLI locally	1	**26**
	2	**14**
	4	**7**
	8	**4**
	16	**2**
The **ExecutionMode.WATCHER** in Airflow with an Astro deployment **(A10)**	8	**5**

https://github.com/google/fhir-dbt-analytics

ASTRONOMER

Strategy 2 `ExecutionMode.WATCHER`

Pros

- Reduced DAG run time to 1/5th of the original time
- Single dbt run
  - generates unified `run_results.json`
  - support dbt `pre-hook` & `post-hook`
- Data warehouse-agnostic implementation

Cons/Current Limitations

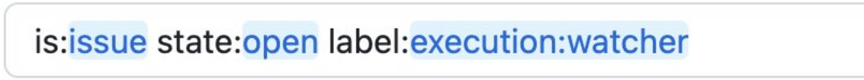
- Airflow worker is blocked by transformations happening in the data warehouse
- Retries have the same performance as `ExecutionMode.LOCAL`
- Currently relies on dbt and Airflow being installed in the same Python venv (some users report conflicts between dependencies)
- Unclear how these features should work: Cosmos callback, Airflow datasets/assets and OpenLineage events

https://github.com/astronomer/astronomer-cosmos/issues/1950

Next steps

`ExecutionMode.WATCHER`

1. We successfully ran a PoC during August 2025
2. First release estimate: end of October 2025 (available in **1.11.0a6**)
3. Work on making sensors deferrable
4. We need feedback!
5. We'd love contributions



```
is:issue state:open label:execution:watcher
```

6. Takeaways

Takeaways

1. To run the **same dbt pipeline** with **multiple dbt command** is **slow**
2. To use **Airflow deferrable operators** allows to not **wait for the transformation** in the **data warehouse**, which can save 36% DAG runtime
3. It is not always possible to **pre-compile** a **dbt project**
4. The **watcher approach** reduces the DAG runtime up to 80% and it is agnostic to **data-warehouse**
5. We need feedback and help!

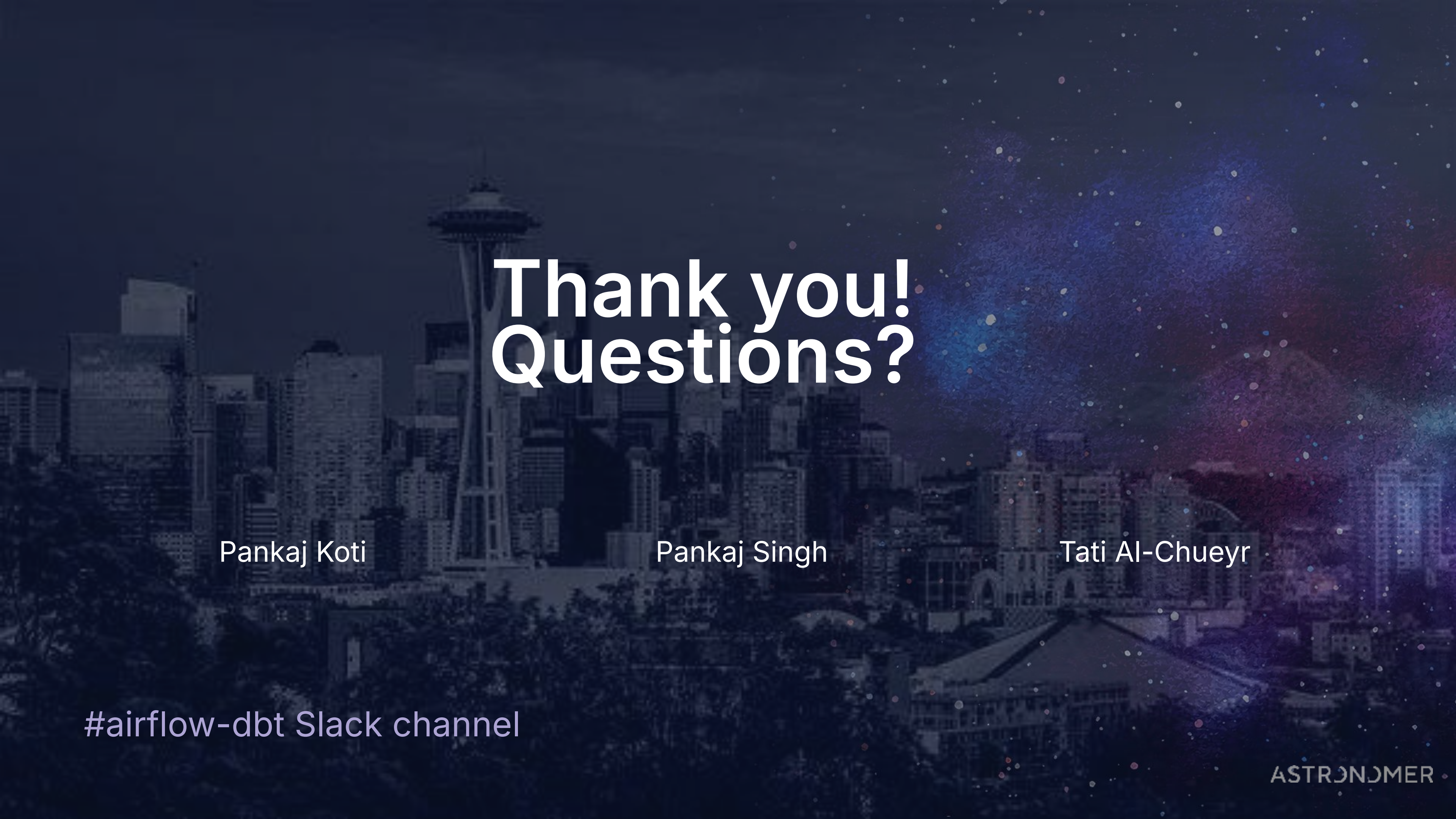
Learn more about how to run dbt with Apache Airflow and Cosmos

```
    target_name="dev",
    profile_mapping=PostgresUserPasswordProfileMapping(
        conn_id=POSTGRES_CONN_ID,
        profile_args={"schema": SCHEMA_NAME},
    ),
)

# Only needed if                           ments.txt file
_execution_confi
    dbt_executab
)


@dag(
    params={
        "my_depa
            "DEF
            type
        ),
    },
    tags=["out-o
)
def example_inje

    @task
    def pre_dbt(
        return c

    _pre_dbt = p

    dbt_project
        group_id="dbt_project",
        project_config=_project_config,
        profile_config=_profile_config,
        execution_config=_execution_config,
        operator_args={
            "vars": '{"my_department": "{{ ti.xcom_pull(task_ids="pre_dbt") }}"}',
```

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

ASTRONOMER



Thank you!
Questions?

Pankaj Koti

Pankaj Singh

Tati Al-Chueyr

#airflow-dbt Slack channel