Airflow Summit 2024

Integrating dbt-core with Airflow

Overcoming Performance Hurdles

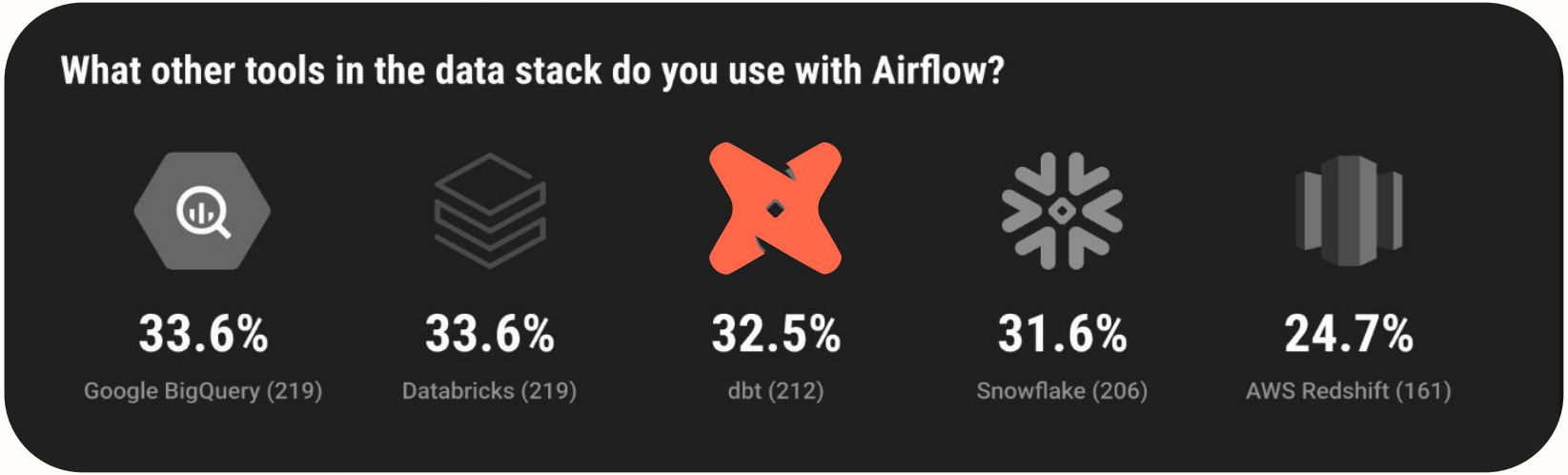
Pankaj Koti Senior Software Engineer
Tati Al-Chueyr Staff Software Engineer

San Francisco, 11 September 2024

ASTRONOMER

dbt in Airflow 2023 Airflow Survey

32.5% of the **2023 Apache Airflow Survey** respondents use dbt



dbt in Airflow Airflow Summit 2023

dbt-core and Airflow was one of the **most popular** topics in **2023**

- “Airflow at Monzo: Evolving our data platform as the bank scales” by Jonathan Rainer & Ed Sparkes
- “Using Dynamic Task Mapping to Orchestrate dbt” by Pádraic Slattery
- “Building an Airflow Pipeline with dbt and Snowflake” by Rishi Kar & George Yates
- “A Single Pane of Glass on Airflow using Astro Python SDK, Snowflake, dbt, and Cosmos” by Luan Moreno Medeiros Maciel
- “Manifest destiny: Orchestrating dbt using Airflow” by Jonathan Talmi

https://airflowsummit.org/sessions/2023/

dbt in Airflow Airflow Summit 2024

dbt-core and Airflow remains a **popular** topic in this year's summit

- "Building on Cosmos: Making dbt on Airflow Easy" by Lewis Macdonald & Ethan Stone (11:00 on Tuesday 10/09)
- "dbt-Core & Airflow 101: Building Data Pipelines Demystified" by Luan Moreno Medeiros Maciel (14:00 on Tuesday 10/09)
- "Integrating dbt with Airflow: Overcoming Performance Hurdles" by Tatiana Al-Chueyr & Pankaj Koti

https://airflowsummit.org/sessions/2024/

dbt in Airflow Airflow Summit 2024

dbt-core and Airflow remains a **popular** topic in this year's summit

- "Building on Cosmos: Making dbt on Airflow Easy" by Lewis Macdonald & Ethan Stone (11:00 on Tuesday 10/09)
- "dbt-Core & Airflow 101: Building Data Pipelines Demystified" by Luan Moreno Medeiros Maciel (14:00 on Tuesday 10/09)
- "Integrating dbt with Airflow: Overcoming Performance Hurdles" by Tatiana Al-Chueyr & Pankaj Koti

https://airflowsummit.org/sessions/2024/

dbt in Airflow Airflow Summit 2024

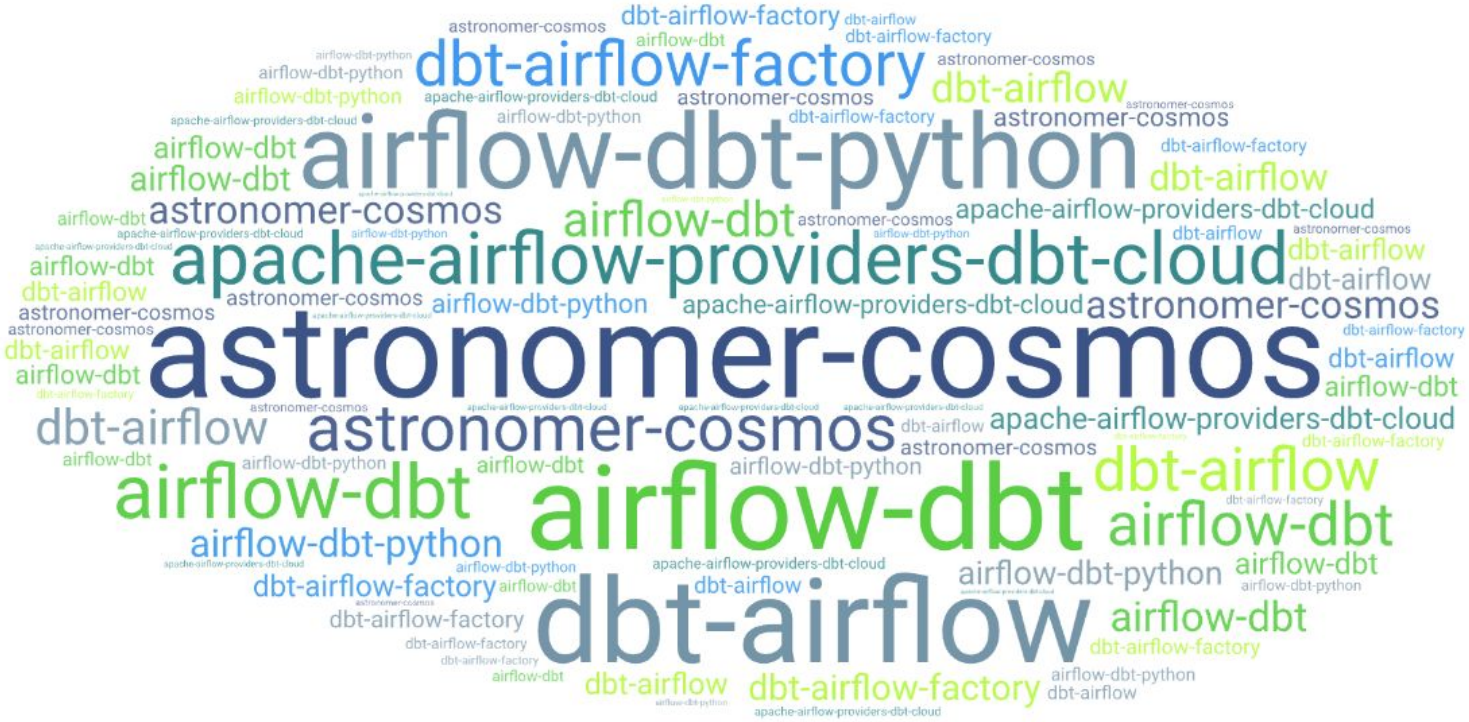
dbt-core and Airflow remains a **popular** topic in this year's summit

- "Building on Cosmos: Making dbt on Airflow Easy" by Lewis Macdonald & Ethan Stone (11:00 on Tuesday 10/09)
- "dbt-Core & Airflow 101: Building Data Pipelines Demystified" by Luan Moreno Medeiros Maciel (14:00 on Tuesday 10/09)
- "Integrating dbt with Airflow: Overcoming Performance Hurdles" by Tatiana Al-Chueyr & Pankaj Koti

intro | why | what | metrics | solutions

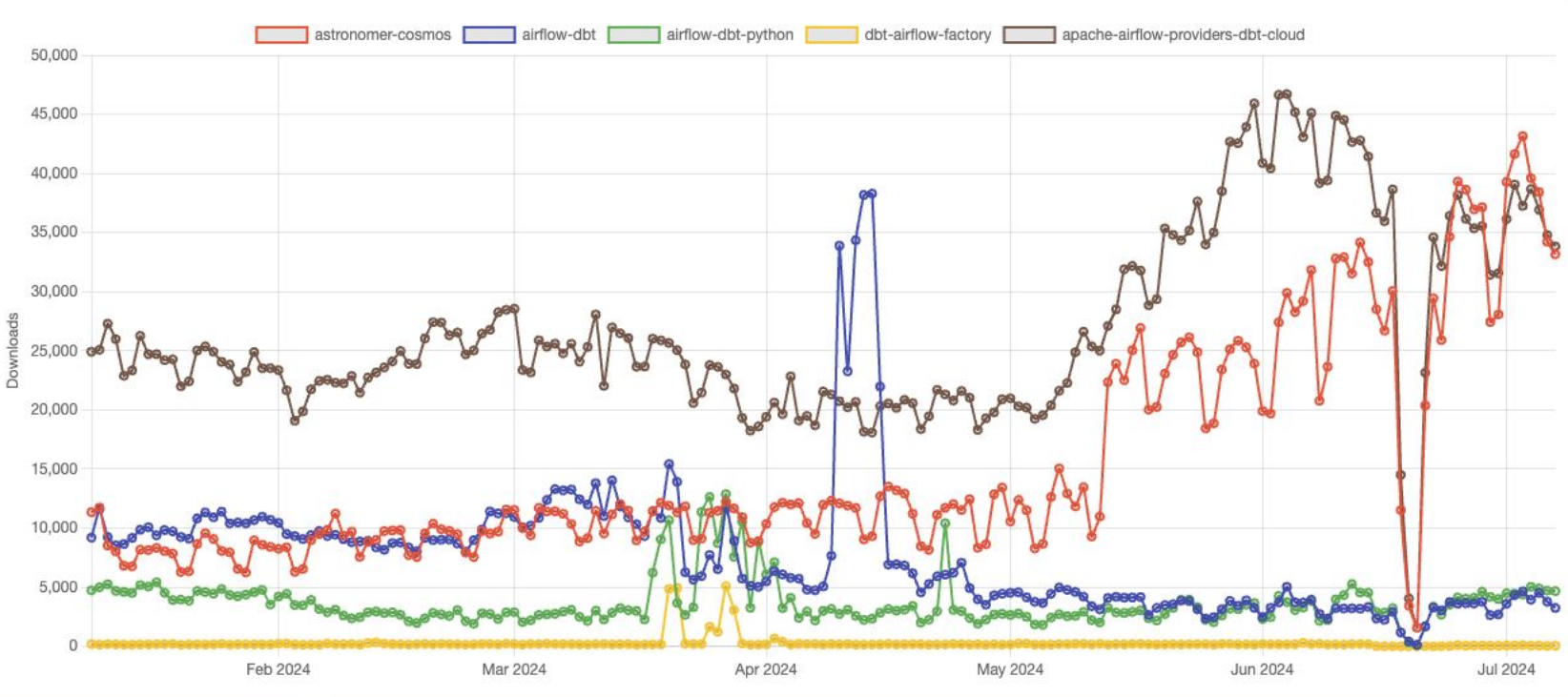
https://airflowsummit.org/sessions/2024/

dbt in Airflow OSS Tools



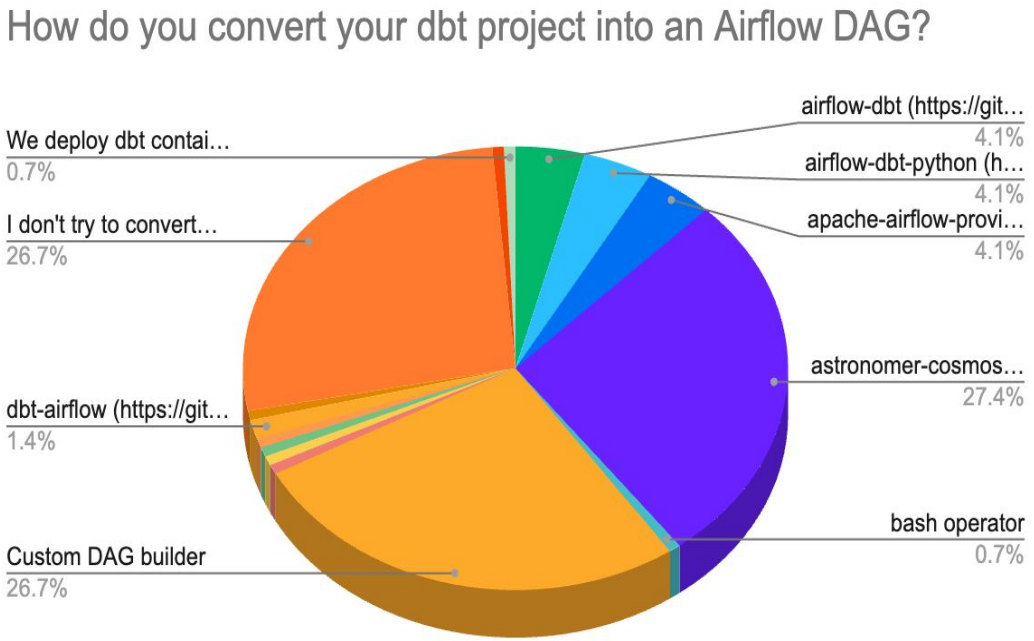
dbt in Airflow OSS Tools Adoption

PyPI downloads for OSS popular tools used to run dbt in Airflow



dbt in Airflow OSS Tools Non-Adopters

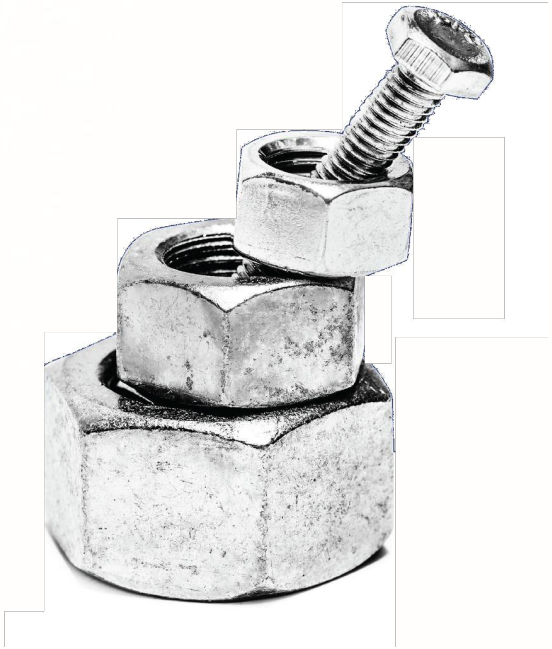
53.4% of the **dbt in Airflow survey** respondents don't use any OSS tools



dbt in Airflow Performance is a challenge

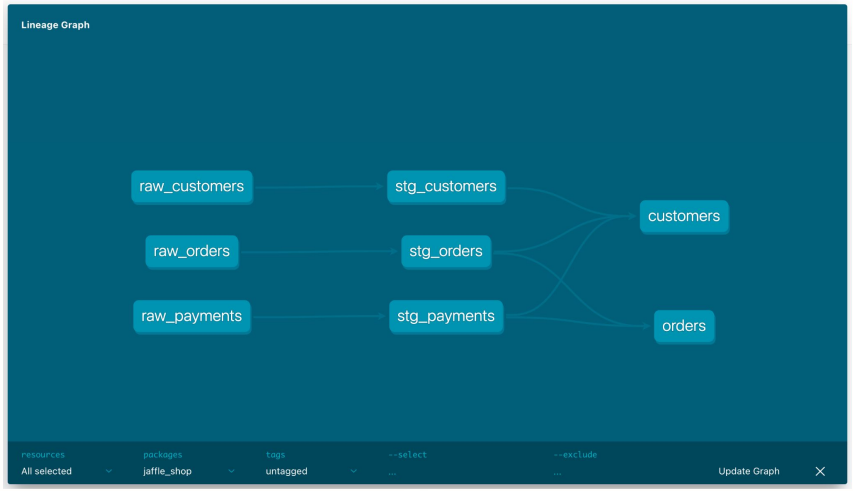
Performance was the second most popular challenge raised by 33.3% of the **dbt in Airflow survey** respondents. The most popular challenge was integrating dbt and Airflow from separate repositories (35.9%)

dbt in Airflow No solution fits all

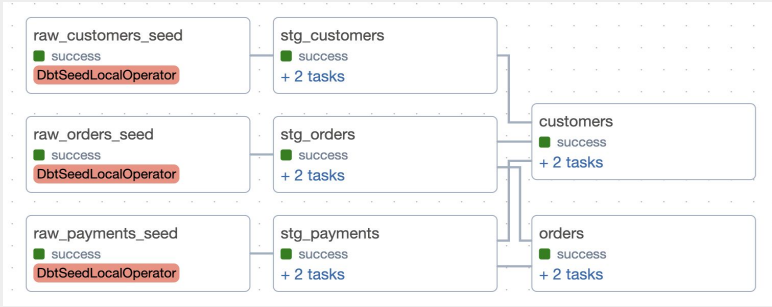


dbt in Airflow Cosmos approach

dbt



Apache Airflow




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

dbt in Airflow Cosmos approach

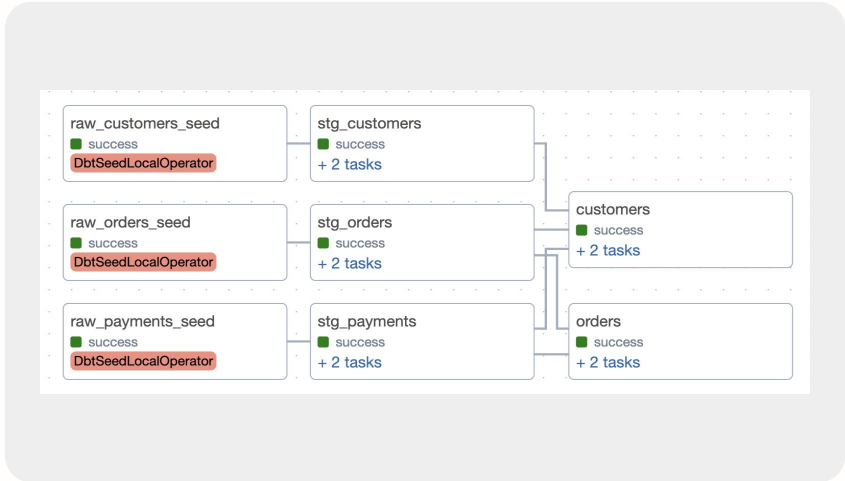
```python
import os
from datetime import datetime
from pathlib import Path
from cosmos import DbtDag, ProjectConfig, ProfileConfig
from cosmos.profiles import PostgresUserPasswordProfileMapping

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

Apache Airflow



Cosmos Key Features

Translate a dbt-core workflow into an Airflow workflow

- Easily render a dbt-core project as an Airflow DAG or Task Group
- Automatically map Airflow connections into dbt profile files
- Dynamically create Airflow datasets for data-aware scheduling
- Only retry necessary dbt transformations
- Generate dbt docs and host them through the Airflow UI
- **Growing active open-source community**

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

Cosmos Flexibility

The user is in control

Where to declare DB credentials

- user-defined profiles.yml
- dynamically create profile from Airflow connection

How to parse the dbt project

- dbt ls command
- dbt manifest
- dbt ls output
- custom

How the DAG is rendered

- dbt selectors
- test behaviour
- customise the conversion
- args

Run dbt your way

Airflow worker

- Local
- Virtualenv
- Docker

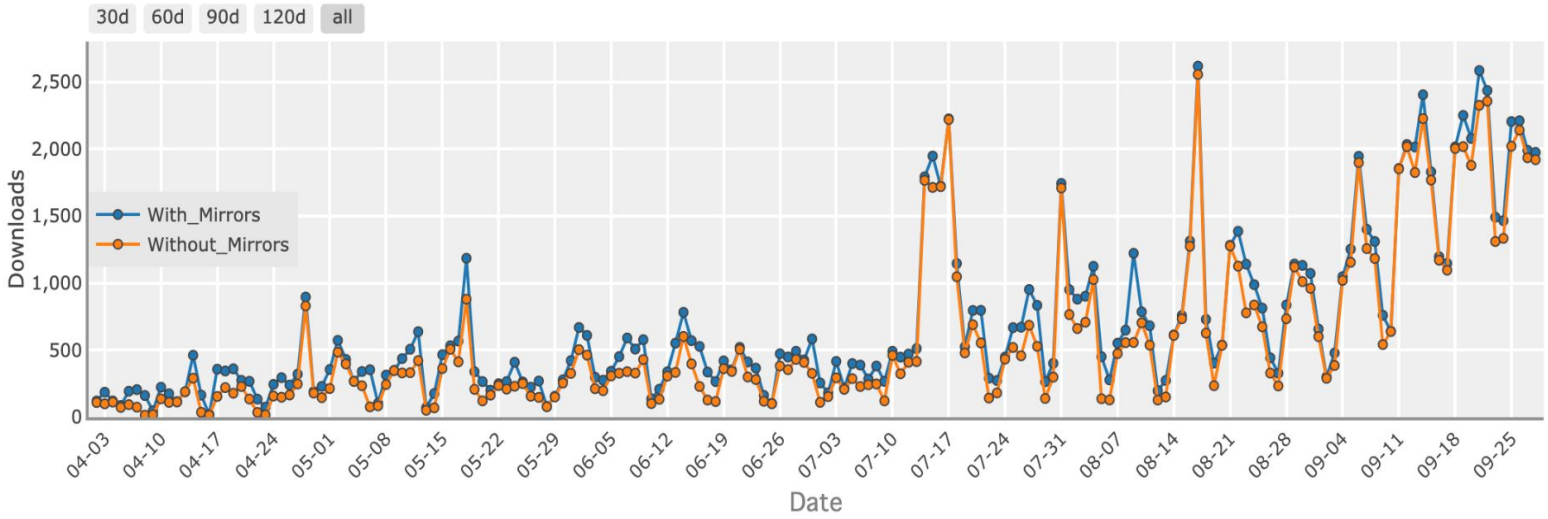
Remotely

- Kubernetes
- AWS EKS
- Azure ACI

April-Sept 2023

Cosmos Adoption

- 44k downloads in a month (September 2023)
- 244 stars in Github (September 2023)

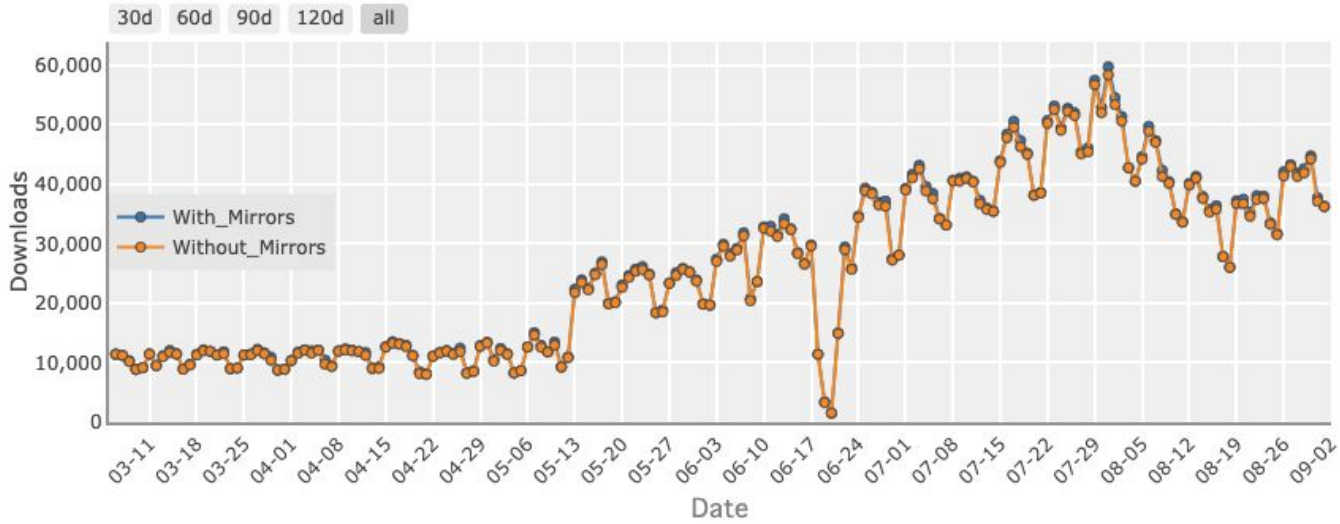


https://pypistats.org/packages/astronomer-cosmos

March-Sept 2024

Cosmos Adoption

- > 1.1 million downloads in a month (Aug-Sept 2024)
- 576 stars in Github (September 2024)
- > 60 Astronomer customers



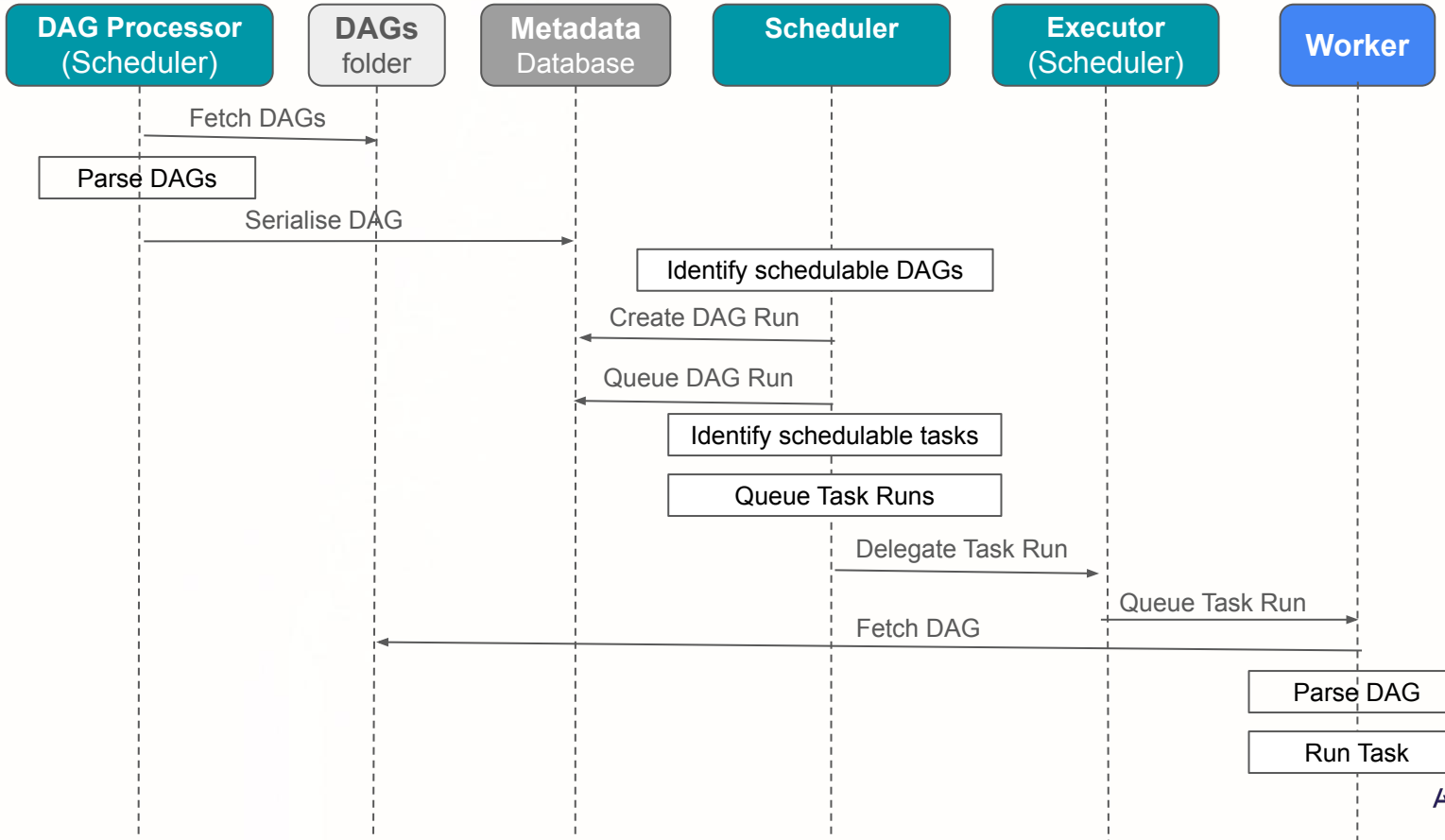
https://pypistats.org/packages/astronomer-cosmos

Cosmos Trade-off

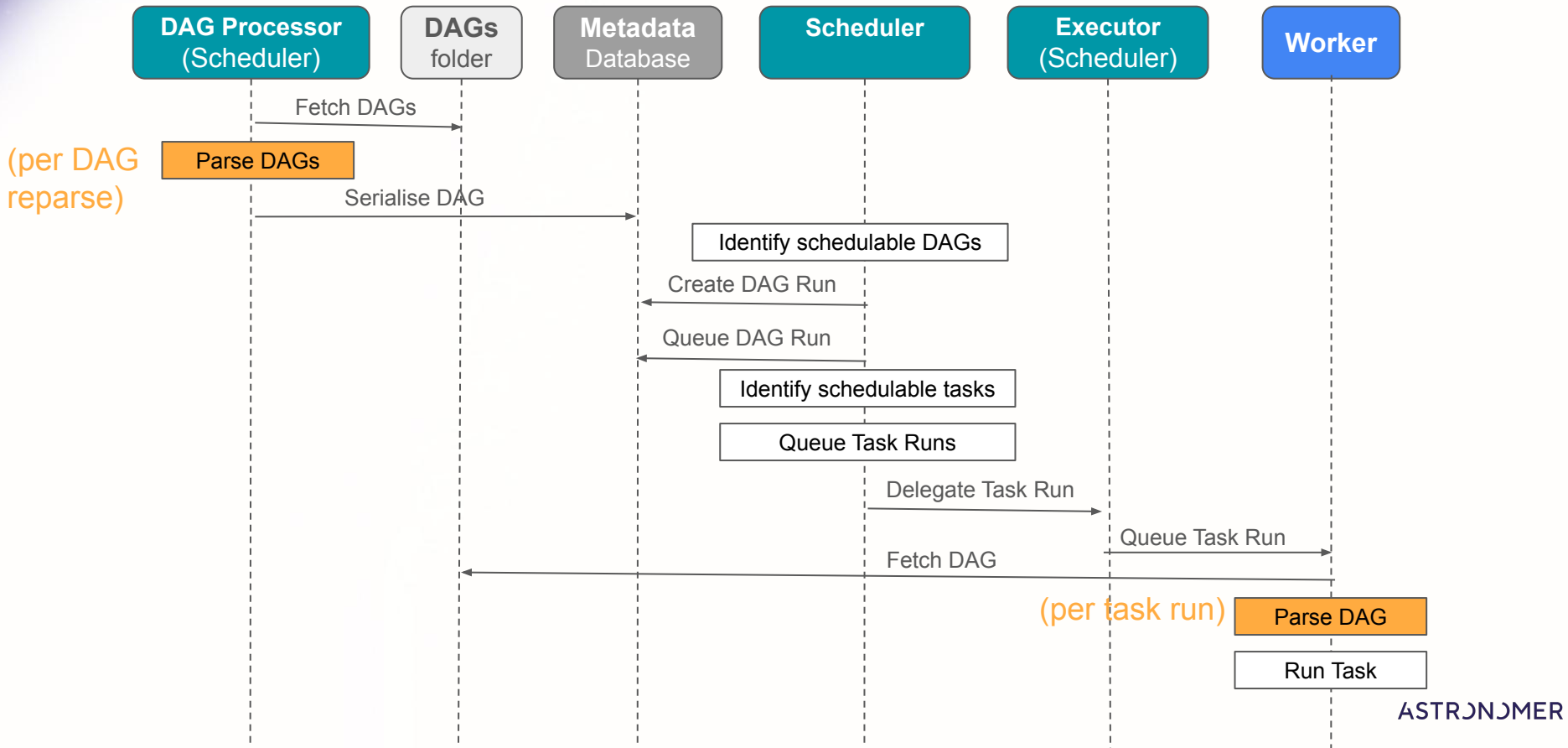
Each tool has their pros and cons

- **Dynamic** DAG rendering **increases the DAG Parsing time**
  - Larger CPU and/or memory consumption
  - Higher DAG processor time
  - Longer task queueing time
- To run one dbt model per task is **slower** than to run multiple dbt models per task
- To run a small dbt-core pipeline in my terminal is **faster** than to run on a distributed orchestration platform

Apache Airflow When DAGs are parsed



Apache Airflow When DAGs are parsed



Cosmos DAG Parsing Steps

1. (optional)

Create profiles.yml

2. (optional)

Run dbt deps

3.

Parse the dbt project

1. dbt ls --output json
2. manifest.json
3. dbt_ls_output.json
4. Cosmos parser

4. (optional)

Select dbt nodes

a. Pre-computed

b. Cosmos selector

c. no selection

5.

Build the Airflow TaskGroup or DAG

Cosmos Task Run Steps (after DAG re-parsing)

1. (optional)

Create profiles.yml

2. (optional)

Create Python virtualenv

Specific to ExecutionMode. VIRTUALENV

3. (optional)

Run dbt deps

4.

Run dbt command

(dependent upon execution mode & invocation method)

1. dbtRunner (python)
2. Subprocess (dbt cmd)
3. K8s.. etc

5. (optional)

Custom user callback

Cosmos Performance Disclaimer

Your choices on how to use Cosmos directly affect how Cosmos-powered DAGs will perform in your Airflow deployment.

Cosmos Performance Improvements

Throughout the past months, several people have actively worked on to improve the performance using various strategies. This talk will discuss some of these.

Cosmos Timeline

0.1 - 12.2022
0.2 - 01.2023
0.3 - 01.2023
0.4 - 02.2023
0.5 - 03.2023
0.6 - 04.2023
0.7 - 05.2023

1.0 - 07.2023

1.1 - 09.2023
1.2 - 10.2023

1.3 - 01.2024

1.4 - 05.2024
1.5 - 06.2024

1.6 - 08.2024

Dec 2022
Astronomer Hack Week

Sept 2023
Airflow Summit '23

Sept 2024
Airflow Summit '24

Cosmos Timeline

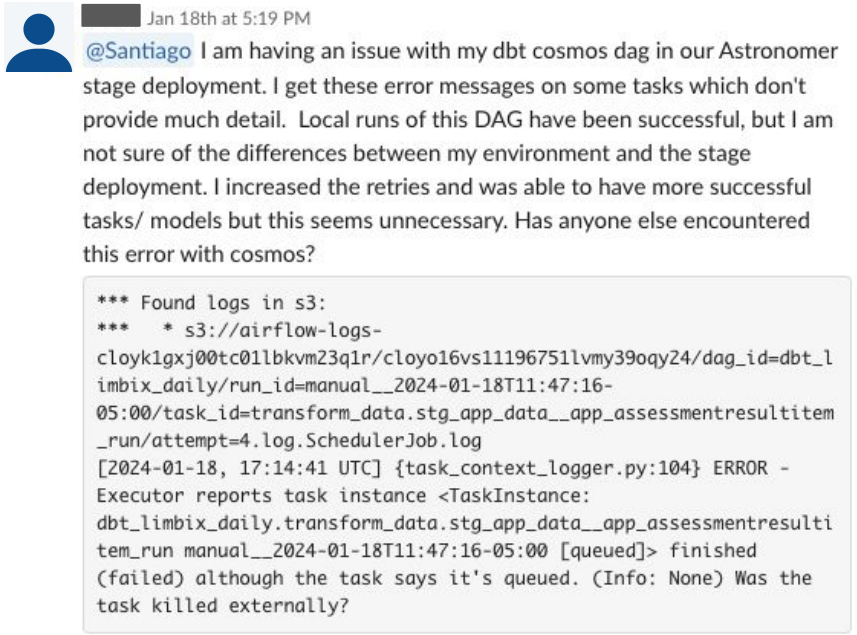
0.1 - 12.2022	**Some (non-performance) features**
0.2 - 01.2023	
0.3 - 01.2023	
0.4 - 02.2023	
0.5 - 03.2023	
0.6 - 04.2023	
0.7 - 05.2023	
1.0 - 07.2023	dbt global flags, **render DAG with LoadMode.DBT_LS without connection**
1.1 - 09.2023	**model versioning**, Athena, custom node rendering, detach render/execution
1.2 - 10.2023	
1.3 - 01.2024	**YAML selector support**, Vertica, Snowflake encrypted key, DbtDocsGCSOperator
1.4 - 05.2024	**dbt Docs in Airflow UI**, Azure Container Instance, DbtBuild operators
1.5 - 06.2024	AWS EKS, Clickhouse
1.6 - 08.2024	**LoadMode.DBT_MANIFEST from remote store,** render **Source Nodes**, Teradata

ASTRONOMER

Performance Hurdles

Jan 24

Cosmos 1.1 DAG Timeout issues



[redacted] Jan 18th at 5:19 PM

@Santiago I am having an issue with my dbt cosmos dag in our Astronomer stage deployment. I get these error messages on some tasks which don't provide much detail. Local runs of this DAG have been successful, but I am not sure of the differences between my environment and the stage deployment. I increased the retries and was able to have more successful tasks/ models but this seems unnecessary. Has anyone else encountered this error with cosmos?

```
*** Found logs in s3:
***   * s3://airflow-logs-
cloyk1gxj00tc01lbkvm23q1r/cloyo16vs11196751lvmy39oqy24/dag_id=dbt_l
imbix_daily/run_id=manual__2024-01-18T11:47:16-
05:00/task_id=transform_data.stg_app_data__app_assessmentresultitem
_run/attempt=4.log.SchedulerJob.log
[2024-01-18, 17:14:41 UTC] {task_context_logger.py:104} ERROR -
Executor reports task instance <TaskInstance:
dbt_limbix_daily.transform_data.stg_app_data__app_assessmentresulti
tem_run manual__2024-01-18T11:47:16-05:00 [queued]> finished
(failed) although the task says it's queued. (Info: None) Was the
task killed externally?
```

https://github.com/astronomer/astronomer-cosmos/issues/520

Jan 24

Cosmos 1.1 DAG Timeout support

Hi [redacted] I got some ideas from the cosmos team that could help, along what you heard from support already:

> There are a few things you could try to do to improve the overall performance of Cosmos:
>
> i) Consider using a more performant parsing method (https://astronomer.github.io/astronomer-cosmos/configuration/parsing-methods.html). For large dbt projects, it may be handy to use DBT_MANIFEST

```
from cosmos import DbtTaskGroup, RenderConfig


DbtTaskGroup(
    (...),
    render_config=RenderConfig(
        load_method=LoadMode.DBT_MANIFEST,
    )
)
```

> ii) Consider selecting a subrange of nodes of interest (https://astronomer.github.io/astronomer-cosmos/configuration/selecting-excluding.html) by using select and exclude statements:

```
from cosmos import DbtTaskGroup, RenderConfig

jaffle_shop = DbtTaskGroup(
    render_config=RenderConfig(
        select=["tag:my_tag"],
    )
)
```

> iii) Another action that could help in dealing with this type of challenge is to increase the DAG parsing timeout in the Airflow configuration, which can be done, for instance, by setting the environment variable: `AIRFLOW__CORE__DAGBAG_IMPORT_TIMEOUT`

The team also let me know that we are actively working and looking for alternatives to further improve the performance of Cosmos. Could you give us more details about your dbt project available in `limbix_dbt`? How many dbt models/seeds/snapshots does it have?

Also, would you be interested in having a call with one of the members of the cosmos team to review in more detail?

https://github.com/astronomer/astronomer-cosmos/issues/520

Feb 24

Cosmos 1.2 Slowness report

zeedevio commented on Feb 7

I'm using Astro Airflow locally.

So I'm trying to execute DBT transformation models using airflow and find it to be extremely slow.
I'm looking for help to improve the performance of my airflow dag.

I'm trying to run 56 transformation models, I have done the following test:

- **DBT cloud:** it completes my model transformation in **4 minutes**
- **Testing Profile Locally**: using `dbt run --profiles-dir /usr/local/airflow/include/dbt/` it completes my transformations in **6 minutes**

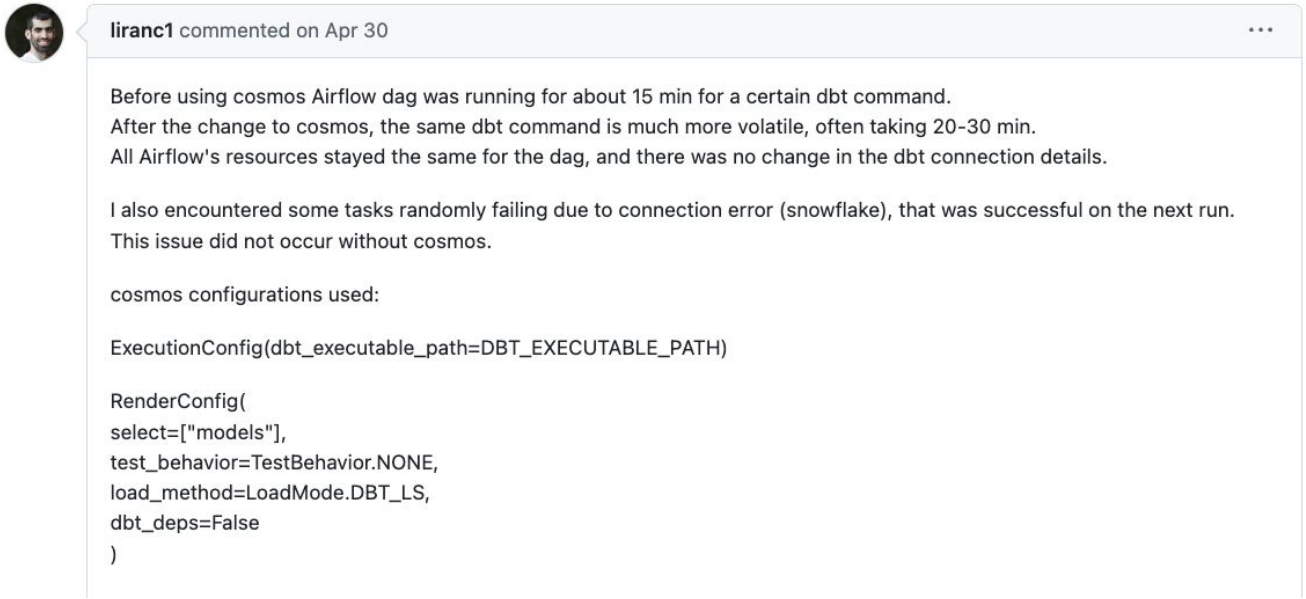
```
08:02:58
08:02:58  Finished running 35 view models, 8 table models, 13 incremental models in 0 hours 6 minutes and 30.74 seconds (390.74s).
08:02:58
08:02:58  Completed successfully
08:02:58
08:02:58  Done. PASS=56 WARN=0 ERROR=0 SKIP=0 TOTAL=56
```

- **AIRFLOW:**: it takes **44 minutes** and i get a fail

https://github.com/astronomer/astronomer-cosmos/issues/840

Apr 24

Cosmos 1.3 Performance degradation



liranc1 commented on Apr 30

Before using cosmos Airflow dag was running for about 15 min for a certain dbt command.
After the change to cosmos, the same dbt command is much more volatile, often taking 20-30 min.
All Airflow's resources stayed the same for the dag, and there was no change in the dbt connection details.

I also encountered some tasks randomly failing due to connection error (snowflake), that was successful on the next run.
This issue did not occur without cosmos.

cosmos configurations used:

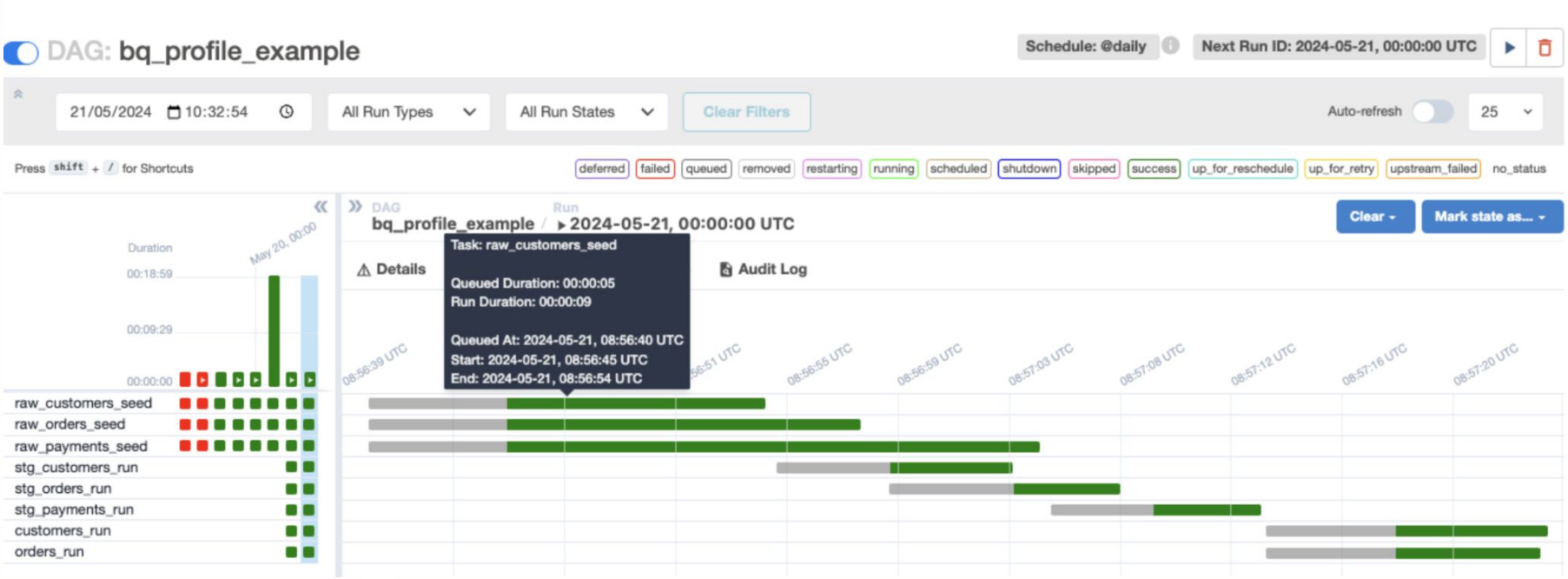
ExecutionConfig(dbt_executable_path=DBT_EXECUTABLE_PATH)

RenderConfig(
select=["models"],
test_behavior=TestBehavior.NONE,
load_method=LoadMode.DBT_LS,
dbt_deps=False
)

https://github.com/astronomer/astronomer-cosmos/issues/932

May 24

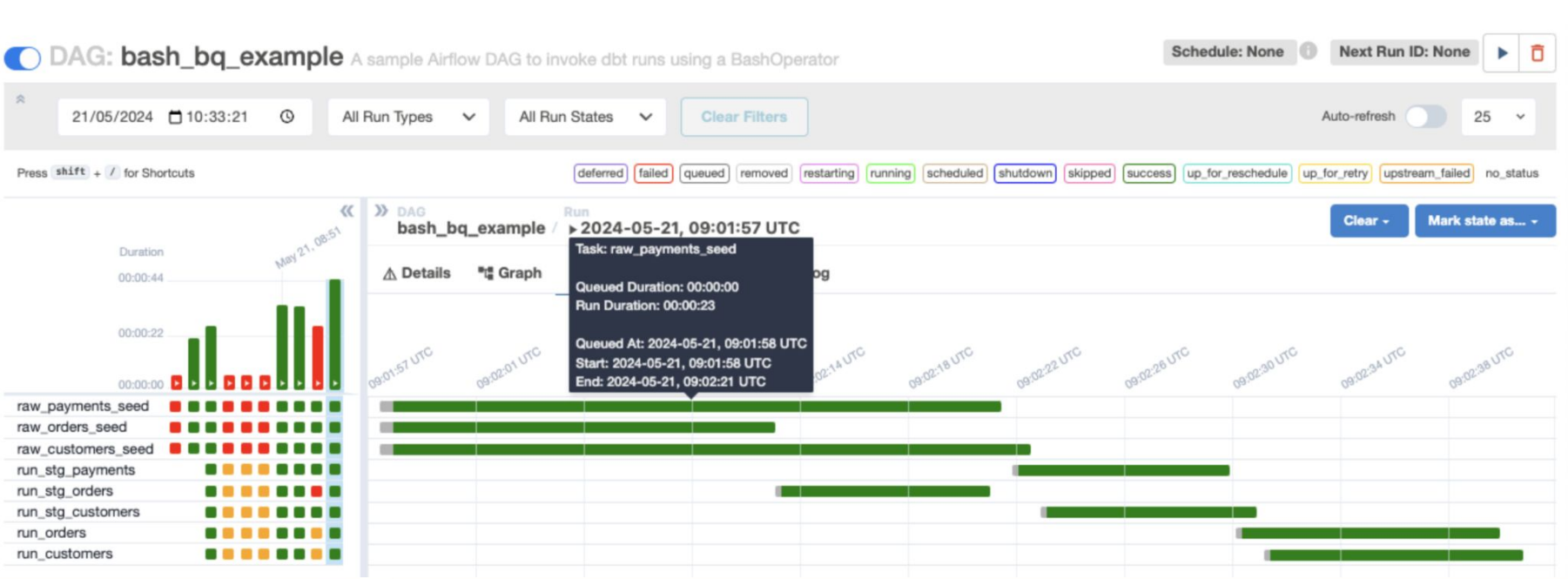
Cosmos 1.4 Large task queueing time



https://github.com/astronomer/astronomer-cosmos/issues/990

May 24

Cosmos 1.4 Large task queueing time



https://github.com/astronomer/astronomer-cosmos/issues/990

Measuring Performance

Performance Metrics

DAG Parsing time

- (optional) Create profile
- (optional) Run dbt deps
- **Parse the given dbt project**
- (optional) Identify the selected dbt nodes
- **Build the Airflow DAG or TaskGroup**

Task Run time

- (optional) Create profile
- (optional) Run dbt deps
- (optional) Create virtualenv
- **Setup/Run dbt command**
- (optional) Callback

Performance Metrics

DAG Parsing time

- (optional) Create profile
- (optional) Run dbt deps
- **Parse the given dbt project**
- (optional) Identify the selected dbt nodes
- **Build the Airflow DAG or TaskGroup**

Task Run time

- (optional) Create profile
- (optional) Run dbt deps
- (optional) Create virtualenv
- **Setup/Run dbt command**
- (optional) Callback

Task Queue time

- DAG Parsing time

DAG Run time

- A combination of the previous metrics

Airflow Deployment

Astro Runtime 11.10.0 (Airflow 2.9.3)

Execution

- Executor Celery
- Worker type A5 (1 vCPU and 2GiB RAM)
- Concurrency 1
- Storage 10 GiB
- Min # Workers 4
- Max # Workers 4

Scheduler

- High Availability on
- Medium
  - Scheduler 1 vCPU and 2 GiB
  - DAG Processor 1 vCPU and 2 GiB

Benchmark Project

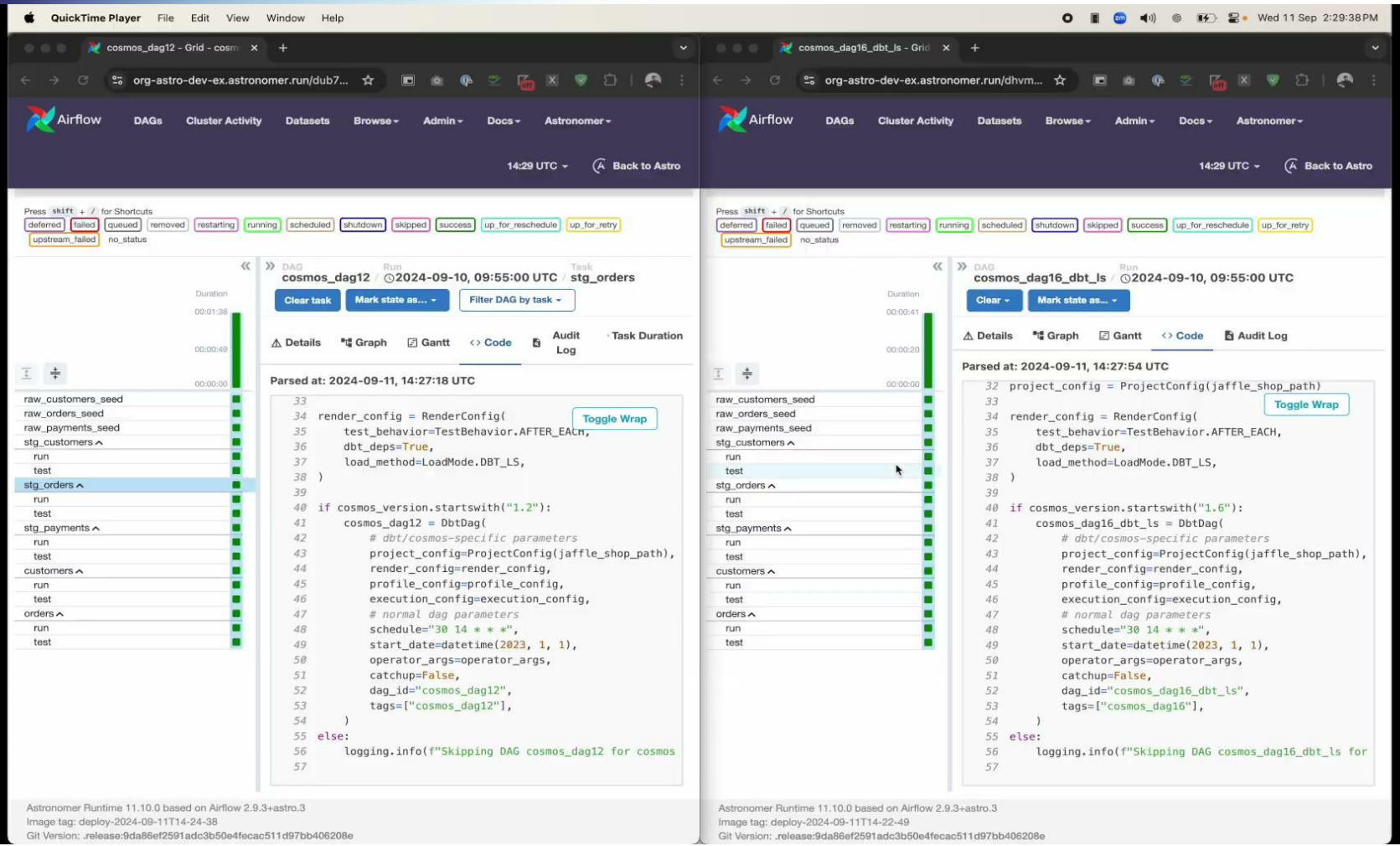
https://github.com/astronomer/airflow-summit-2024-cosmos/

dbt project: (old) Jaffle Shop

Baseline

	Cosmos 1.2.5 (`LoadMode.DBT_LS` & `ProfileMapping`)
DAG Parsing time	00:00:08
Task Run time	00:00:09
Task Queue time	00:00:09
DAG Run time	00:01:29

Overcoming Performance Hurdles



https://drive.google.com/file/d/1R-v3flgj5mnJWoqLe-OE0OirybdqRPAY/view?usp=sharing

ASTRONOMER

Performance Improvements

- 1.2:
  - Baseline
- 1.3:
  - Introduction of `LoadMode.DBT_LS_FILE`
- 1.4:
  - Script to evaluate performance
  - Introduce `InvocationMode.DBT_RUNNER`
  - Use & cache dbt partial parsing
  - Only run `dbt deps` when there is `packages.yml`
- 1.5:
  - Cache `LoadMode.DBT_LS` using Airflow variables
  - Cache `ProfileMapping`

Performance Improvements

- 1.6:
  - Cache `package-lock.yml`
  - Persist `LoadMode.VIRTUALENV` directory
  - Cache `LoadMode.DBT_LS` using remote store

1.3

Cosmos 1.3 Performance Improvements

Improvement in **DAG Parsing**

1.	2.	3.	4.	5.
Create profiles.yml	Run dbt deps	**Parse the dbt project**	**Select nodes**	Build the Airflow DAG

- Introduction of `LoadMode.DBT_LS_FILE` by @woogakoki
  - #733
  - Similar to `LoadMode.DBT_MANIFEST`
  - Users have to pre-compile the project (`dbt ls --output json`)
  - Cosmos understands this output file

"This should increase performance compared to using dbt_ls."

Cosmos 1.3 Performance Improvements

	1.2.5	1.3 (DBT LS FILE)
DAG Parsing time	00:00:08	00:00:02 ⬇
Task Run time	00:00:09	00:00:08
Task Queue time	00:00:09	00:00:04 ⬇
DAG Run time	00:01:29	00:00:55 ⬇

1.4

Cosmos 1.4 Performance Improvements

Improvement in **DAG Parsing**

1.	2.	3.	4.	5.
Create profiles.yml	**Run dbt deps**	Parse the dbt project	Select nodes	Build the Airflow DAG


a) Only run dbt deps when there is `packages.yml` by @AlgirdasDubickas @tatiana

- #1030

Cosmos 1.4 Performance Improvements

Improvement in **DAG Parsing**

1.	2.	3.	4.	5.
Create profiles.yml	Run dbt deps	**Parse the dbt project**	Select nodes	Build the Airflow DAG


b) Use & cache dbt partial parsing by @dwreeves @tatiana

- #800, #904
- Improvement in how dbt commands run (`LoadMode.DBT_LS`)
- Leverages dbt `partial_parse.msgpack`

"Improve the performance to run the benchmark DAG with 100 tasks by 34% and the benchmark DAG with 10 tasks by 22%"

Cosmos 1.4 Performance Improvements

Improvement in **Task Run**

1.	2.	3.	4.	5.
Create profiles.yml	Create py virtualenv	**Run dbt deps**	Select nodes	User def callback

a) Only run dbt deps when there is `packages.yml` by @AlgirdasDubickas @tatiana

- #1030

Cosmos 1.4 Performance Improvements

Improvement in **Task Run**

1.	2.	3.	4.	5.
Create profiles.yml	Create py virtualenv	Run dbt deps	**Run dbt command**	User def callback

b) Use & cache dbt partial parsing by @dwreeves @tatiana


- #800, #904
- Improvement in how dbt commands run (`LoadMode.DBT_LS`)
- Leverages dbt `partial_parse.msgpack`

"Improve the performance to run the benchmark DAG with 100 tasks by 34% and the benchmark DAG with 10 tasks by 22%"

Cosmos 1.4 Performance Improvements

Improvement in **Task Run**

1.	2.	3.	4.	5.
Create profiles.yml	Create py virtualenv	**Run dbt deps**	**Run dbt command**	User def callback

c) Introduction of `InvocationMode.DBT_RUNNER` by @jbandoro

- #850
- Avoid create subprocesses to run dbt commands in task execution
- Relies on dbt and Airflow being in the same Python virtualenv

"Using InvocationMode.DBT_RUNNER is almost 3x faster, and can speed up dag runs if there are a lot of models that execute relatively quickly since there seems to be a 1-2s speed up per task."

Cosmos 1.4 Performance Improvements

	1.2.5	1.4	
DAG Parsing time	00:00:08	00:00:07	⬇
Task Run time	00:00:09	00:00:06	⬇
Task Queue time	00:00:09	00:00:05	⬇
DAG Run time	00:01:29	00:01:18	⬇

1.5

Cosmos 1.5 Performance Improvements

Improvement in **DAG Parsing**

1.	2.	3.	4.	5.
Create profiles.yml	Run dbt deps	Parse the dbt project	Select nodes	Build the Airflow DAG

a) Cache `ProfileMapping` by @pankajastro

- #1046
- Similar to `partial_parse.msgpack` caching

"Enabling profile caching for 100 models DAG benchmark reduced the DAG run by 11%"

Cosmos 1.5 Performance Improvements

Improvement in **DAG Parsing**

1.	2.	3.	4.	5.
Create profiles.yml	Run dbt deps	**Parse the dbt project**	**Select nodes**	Build the Airflow DAG

b) Cache `LoadMode.DBT_LS` using Airflow variables by @tatiana

- #992 #1014
- Mechanism to cache output of dbt ls into Airflow variable
- Automatic purge if dbt project changes

"The example DAGs tested reduced the task queueing time significantly (from ~30s to ~0.5s) and the total DAG run time for Jaffle Shop from 1 min 25s to 40s (by more than 50%)."

Cosmos 1.5 Performance Improvements

kzajaczkowski commented on Jun 14 • edited

> **@kzajaczkowski**, thanks a lot! The feature is currently available in the 1.5.0a7 release. Would you be interested in testing it out and giving early feedback?

@tatiana, we did some preliminary testing using 1.5.0a4 release and observed one of a simple dags go from 00:13:45 to 00:02:11. Queuing times went almost to zero. We experienced also `pytest` dag tests go from a few minutes to seconds.

Cosmos 1.5 Performance Improvements

Improvement in **Task Run**

1.	2.	3.	4.	5.
Create profiles.yml	Create py virtualenv	Run dbt deps	Run dbt command	User def callback

a) Cache `ProfileMapping` by @pankajastro

- #1046
- Similar to `partial_parse.msgpack` caching

"Enabling profile caching for 100 models DAG benchmark reduced the DAG run by 11%"

Cosmos 1.5 Performance Improvements

	1.2.5	1.4	1.5	
DAG Parsing time	00:00:08	00:00:07	00:00:02	⬇
Task Run time	00:00:09	00:00:06	00:00:05	⬇
Task Queue time	00:00:09	00:00:05	00:00:01	⬇
DAG Run time	00:01:29	00:01:18	00:00:43	⬇

1.6

Cosmos 1.6 Performance Improvements

Improvement in **DAG Parsing**

1.	2.	3.	4.	5.
Create profiles.yml	**Run dbt deps**	Parse the dbt project	Select nodes	Build the Airflow DAG

a) Cache `package-lock.yml` by @pankajastro

- #1086
- Similar to `profiles.yml` and `partial_parse.msgpack`

Cosmos 1.6 Performance Improvements

Improvement in **Task Run**

1.	2.	3.	4.	5.
Create profiles.yml	Create py virtualenv	**Run dbt deps**	Run dbt command	User def callback

a) Cache `package-lock.yml` by @pankajastro

- #1086
- Similar to `profiles.yml` and `partial_parse.msgpack`

Cosmos 1.6 Performance Improvements

	1.2.5	1.4	1.5	1.6	
DAG Parsing time	00:00:08	00:00:07	00:00:02	00:00:02	-75%
Task Run time	00:00:09	00:00:06	00:00:05	00:00:04	-56%
Task Queue time	00:00:09	00:00:05	00:00:01	00:00:01	-89%
DAG Run time	00:01:29	00:01:18	00:00:43	00:00:42	-52%

Cosmos 1.6 Performance Improvements

Improvement in **DAG Parsing**

1.	2.	3.	4.	5.
Create profiles.yml	Run dbt deps	**Parse the dbt project**	**Select nodes**	Build the Airflow DAG

b) Cache `LoadMode.DBT_LS` using remote store by @pankajkoti

- #1147
- Alternative to Airflow variable caching (Cosmos 1.5)

"Users would observe a slight delay for the tasks being in queued state (approx 1-2 seconds queued duration vs the 0-1 seconds previously in the Variable approach) due to remote storage calls."

Cosmos 1.6 Performance Improvements

Improvement in **Task Run**

1.	2.	3.	4.	5.
Create profiles.yml	**Create py virtualenv**	Run dbt deps	Run dbt command	User def callback


c) Persist `LoadMode.VIRTUALENV` directory by @LennartKloppenburg and @tatiana

- #611 #1079
- Avoid creating a new virtualenv for each task run
- Persist the virtualenv per Airflow worker node

"The example_virtualenv DAG saw the DAG's runtime go down from 2m31s to just 32s. I'd this improvement to be even more noticeable with more complex graphs and more python requirements."

Overview

Overview Performance Improvements

	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

Overview Performance Improvements

Released in the last 8 months

Summary

7 DAG Parsing improvements

2 Speed up improvements on running dbt

4 Improvements on Task run performance

DAG Run

Reduced to

25%

of the original time

Community

8

developers contributed to making Cosmos faster

Future

Performance Improvements Future

We could use help to bring this ideas to life!

- Introduce Airflow native (deferrable) Operators execution mode #1134
  - dbt-core is used to pre-compile SQL
  - Python native operators (e.g. `DatabricksSubmitRunOperator`) execute actual transformations
- Support representing dbt models as single Airflow task #881
- Support using `dbtRunner` when parsing dbt project with `LoadMode.DBT_LS` #865
- Support caching on remote store #1177, #1178, #1179
- Leverage Airflow magic loop #918

Takeaways

Takeaways

Cosmos 1.6 is faster than previous versions

- Some of Astronomer customers moved from using dbt Cloud to use Cosmos with confidence, while leveraging dynamic DAG building with `LoadMode.DBT_LS`

Understanding how Airflow works is critical

- DAG parsing happens for every task run

Having a systematic approach to measuring the progress

- Data-driven decisions

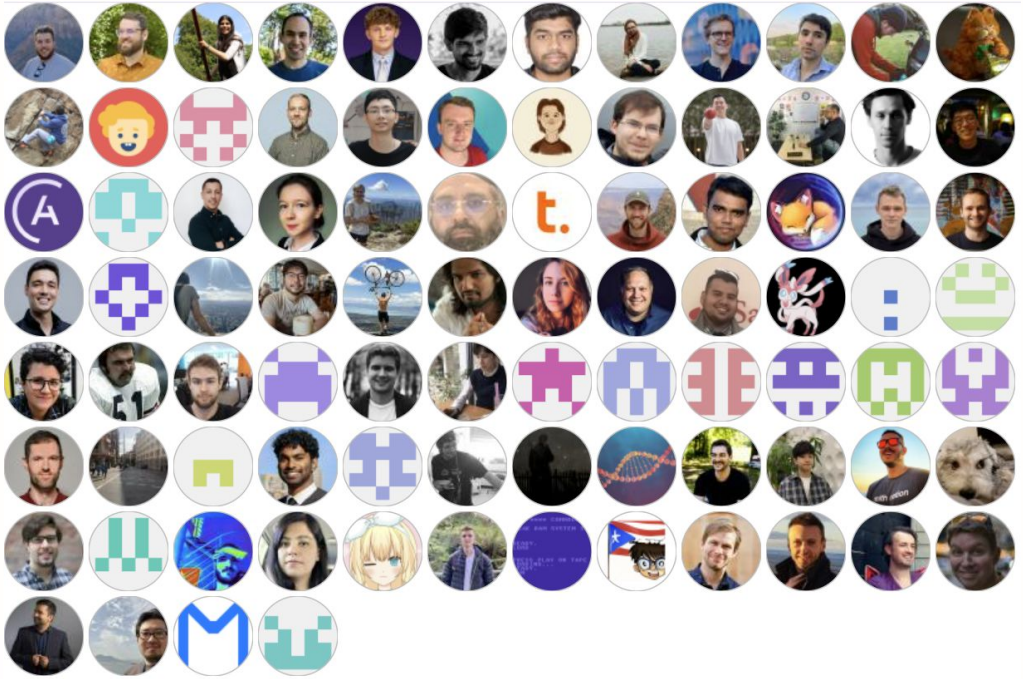
This work is an ongoing journey of collaboration

- Multiple people contributed to this work
- More work is planned

Credits

Cosmos Open Source Community

99 - and growing - contributors in Github (6 September 2024)



Cosmos Active maintainers

Julian LaNeve
CTO
@ Astronomer

Tati Al-Chueyr
Lead/ Staff Software Engineer
@ Astronomer

Justin Bandoro
Data Engineer
@ Kevala Analytics

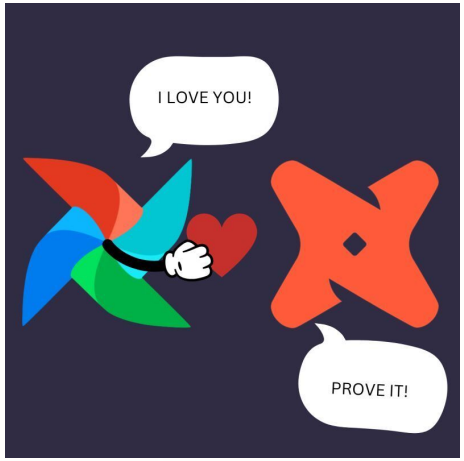
Daniel Reeves
Data Architect
@ Battery Ventures

Pankaj Singh
Senior Software Engineer
@ Astronomer

Pankaj Koti
Senior Software Engineer
@ Astronomer

Last but not least

dbt in Airflow survey



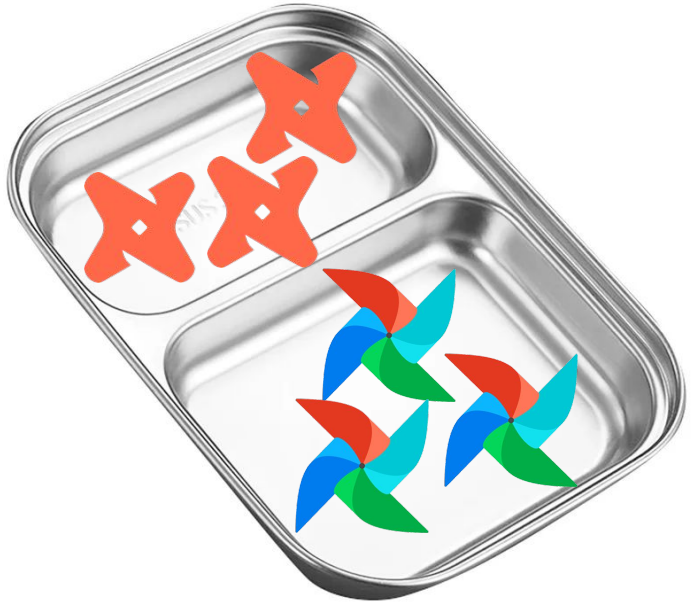
https://bit.ly/dbt-airflow-survey-2024

dbt in Airflow lunch table

Come and join us for lunch

13:30 - 14:30

We'll have a themed table for those interested in discussing how they are running dbt in Airflow



Thank you!

Any questions?

#airflow-dbt 

ASTRONOMER