

Benchmarking the Performance of Dynamically Generated DAGs

Rahul Vats
Engineering Manager
@ Astronomer

Tati Al-Chueyr
Principal Software Engineer
@ Astronomer

Airflow Summit, Seattle, USA - 7 October 2025

ASTRONOMER

Agenda

1. Motivation
2. Benchmark Principles
3. Representative Workflows
4. Metrics
5. Implementation
6. Results
7. Next steps

1. Motivation

Dynamic DAGs in Airflow

Because everything in Airflow is code, we can generate DAGs dynamically

```
from datetime import datetime
from airflow.sdk import DAG
from airflow.providers.standard.operators.bash import BashOperator

dag_configs = [
    {"dag_id": "dynamic_dag_1", "command": "echo DAG 1"},
    {"dag_id": "dynamic_dag_2", "command": "echo DAG 2"},
]

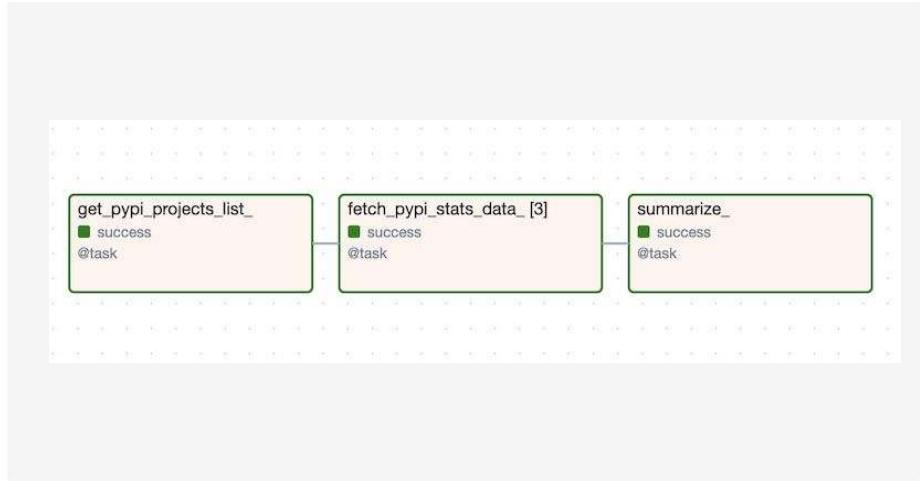
def create_dag(dag_id: str, command: str) -> DAG:
    with DAG(dag_id=dag_id, start_date=datetime(2025, 10, 7), schedule="@daily") as dag:
        BashOperator(task_id="run_cmd", bash_command=command)
    return dag

for cfg in dag_configs:
    globals()[cfg["dag_id"]] = create_dag(cfg["dag_id"], cfg["command"])
```

Dynamic DAGs in Airflow

The DAG Factory library, for example, builds Airflow DAGs out of YAML files

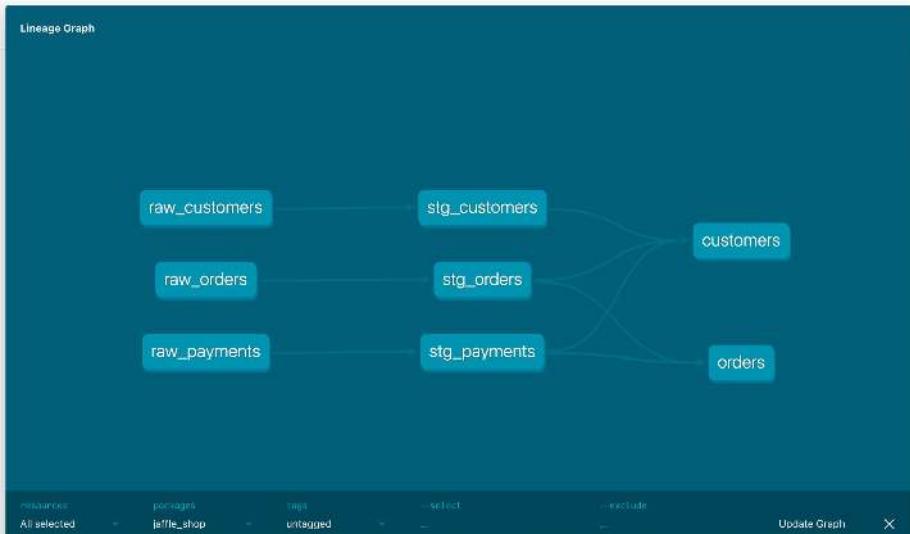
```
example_pypi_stats_dagfactory:  
  default_args:  
    start_date: 2025-10-07  
  tasks:  
    - task_id: "get_pypi_projects_list"  
      decorator: airflow.sdk.task  
      python_callable: pypi_stats.get_pypi_projects_list  
    - task_id: "fetch_pypi_stats_data"  
      decorator: airflow.sdk.task  
      python_callable: pypi_stats.fetch_pypi_stats_data  
      expand:  
        package_name: +get_pypi_projects_list  
    - task_id: "summarize"  
      decorator: airflow.sdk.task  
      python_callable: pypi_stats.summarize  
      values: +fetch_pypi_stats_data
```



```
$ pip install dag-factory
```

Dynamic DAGs in Airflow

The Cosmos package dynamically translates dbt pipelines into Airflow DAGs

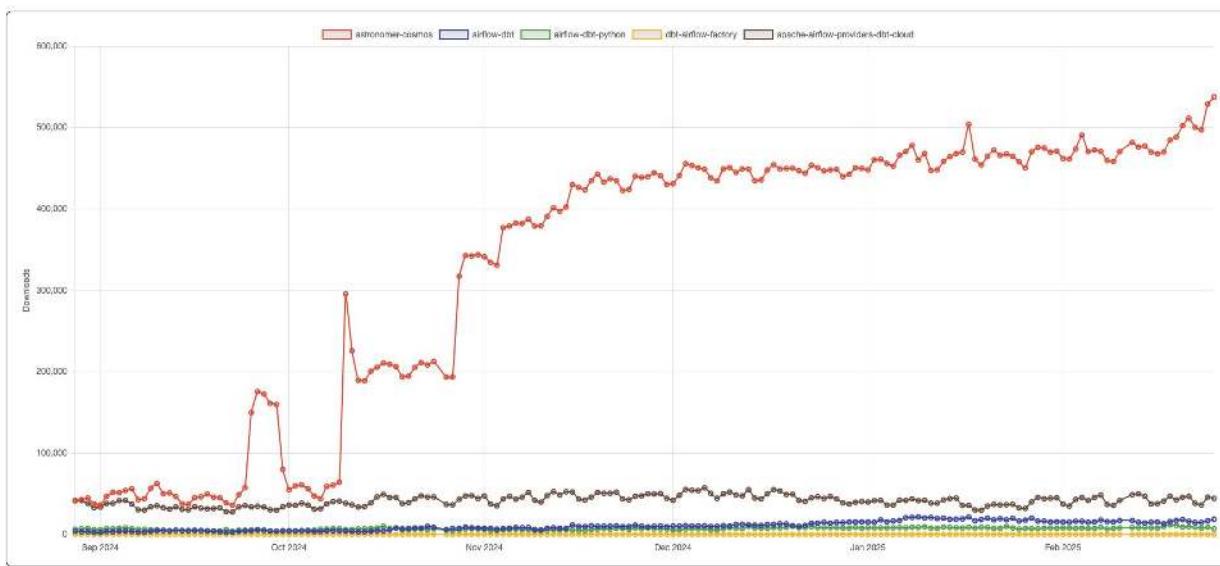


```
$ pip install astronomer-cosmos
```

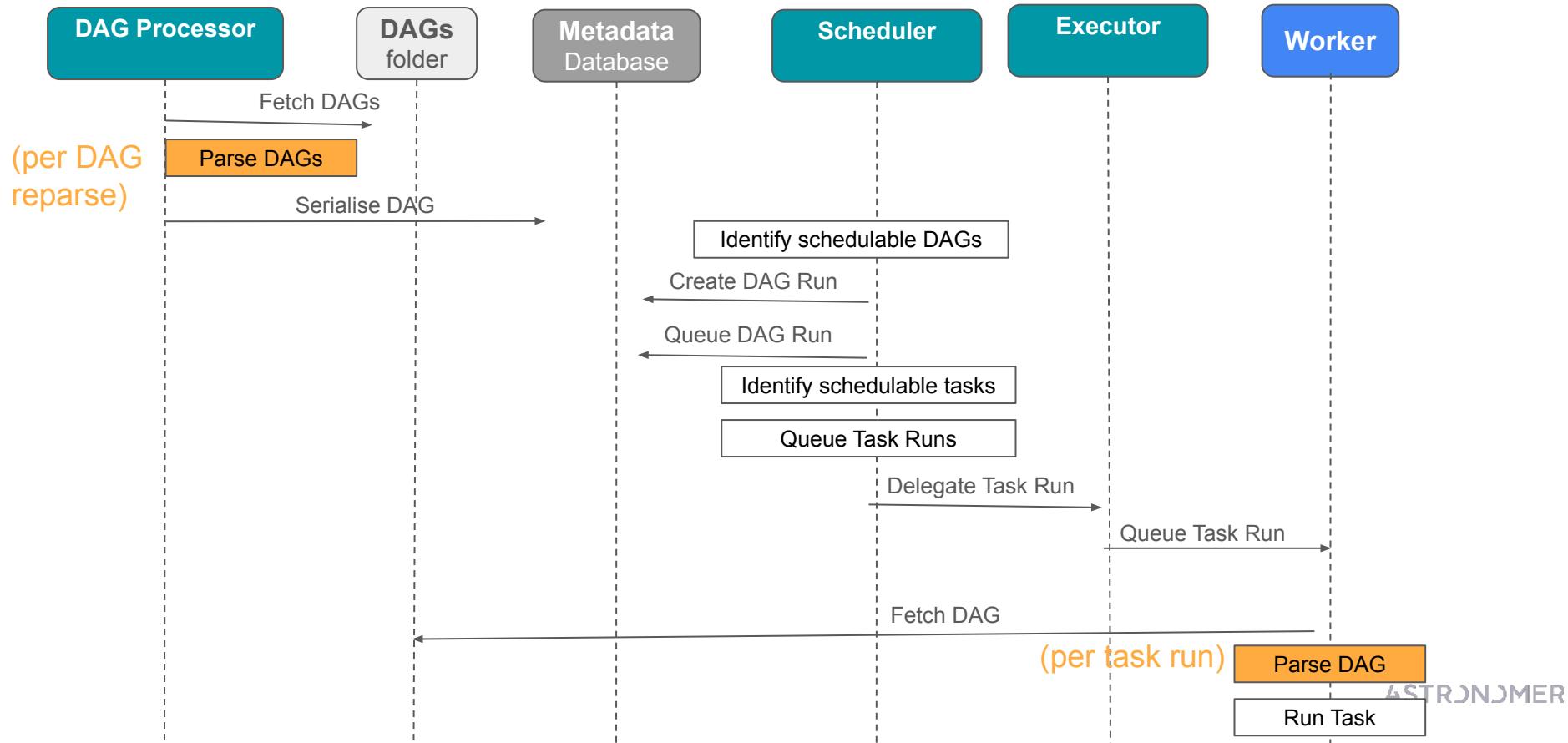
ASTRONOMER

High adoption of Dynamic DAGs tools

- Over 20M+ monthly downloads in PyPI (just Cosmos & DAG Factory)
- Millions of dynamically build DAGs run every month in Astro



How often Airflow reparse(Dynamic) DAGs



Dynamic DAGs needs

- DAGs are parsed both by the DAG Processor and **every** Worker node
- Compared to non-dynamic DAGs, dynamic DAGs will likely:
 - Consume more CPU
 - Consume more memory
 - Take longer to be parsed
- Users can be surprised by:
 - DAG Timeout issues
 - Long task queue times
 - Resource consumption not only on the DAG Processor, but also on worker nodes

Dynamic DAGs issues

Dags taking a long time to appear in the UI and staying in a queued state...



Via web form



Annie Friedman Internal • Aug 06 10:43

They said Dags were taking sometimes upwards of 30 minutes to reflect changes and tasks are being queued for upwards of 10min. They are a bit high on dag processor and worker cpu but not horribly so. From my cursory glance, I suspect this is exacerbated from their high rate of dag only deploys. Is there anything on the backend we can do to alleviate some of the pressure from the deploys? They aren't interested in using ephemeral deployments and their ci/cd runs a dag only deploy with every developer's commits. They also aren't interested in changing their ci/cd process at this time. This is effecting both dev and prod and is new since they have moved to hosted.

<https://astronomer.zendesk.com/agent/tickets/80415>

Dynamic DAGs issues

cosmos task taking too long

Via API



Jun 26 01:43

To: [REDACTED] [Show more](#)

Hi, I have an issue with a dag "ingestion_dbt_starfish_retail_orders", where the dag is scheduled to run every 10min but each run is taking more than 10min after I moved to cosmos. This dbt job is running several dbt models based on a dbt tag. After moving to cosmos, the length of each run has increased because the dbt compile takes place for each individual model instead of just once when running with the tag. Do you have any suggestions on reducing the time taken when using cosmos and running dbt models using dbt tags

Workspace: Data Team

Deployment: [REDACTED]-data-prod-astro-deployment

Stakeholders who cares about performance?

- **End-users:** can lose money due to misbehaving workflows
- **Airflow Developers:** want to improve - and not degrade - performance over versions
- **Sales:** so they communicate metrics to prospective customers with confidence
- **Product Marketing:** wants to compare against competitors

Lack of benchmark standardization

- **No clear standard** for running **performance benchmarks** on Apache Airflow
- Users and companies very often rely on **ad-hoc benchmarking**
- There is **lack of consistency** and **manual overhead**
- **Lack of history**, results are usually presented in one-off spreadsheets, docs and slides

2. Benchmark Principles

Clear objectives

- Define **what** you want to **measure** (throughput, latency, resource usage, etc)
- Tie benchmarks to **real-world use patterns** (peak loads, typical queries, business workflows)
- Motivation:
 - Find bottlenecks
 - Comparing against a baseline

Workload Design

- **Representativeness:** Use realistic workloads, not just synthetic stress tests.
- **Variability:** Include different query types, request patterns, and concurrency levels.
- **Scaling:** Test both typical and extreme workloads (steady state + stress testing).

Environment consistency

- Ensure test environments are **isolated and reproducible** (same hardware, cloud instance type, config).
- Minimize **external noise**: background jobs, network contention, autoscaling effects.
- Use **version control** for test configs, datasets, and scripts.

Benchmark experiment life cycle



3. Representative Workflows

Some workflows are too small



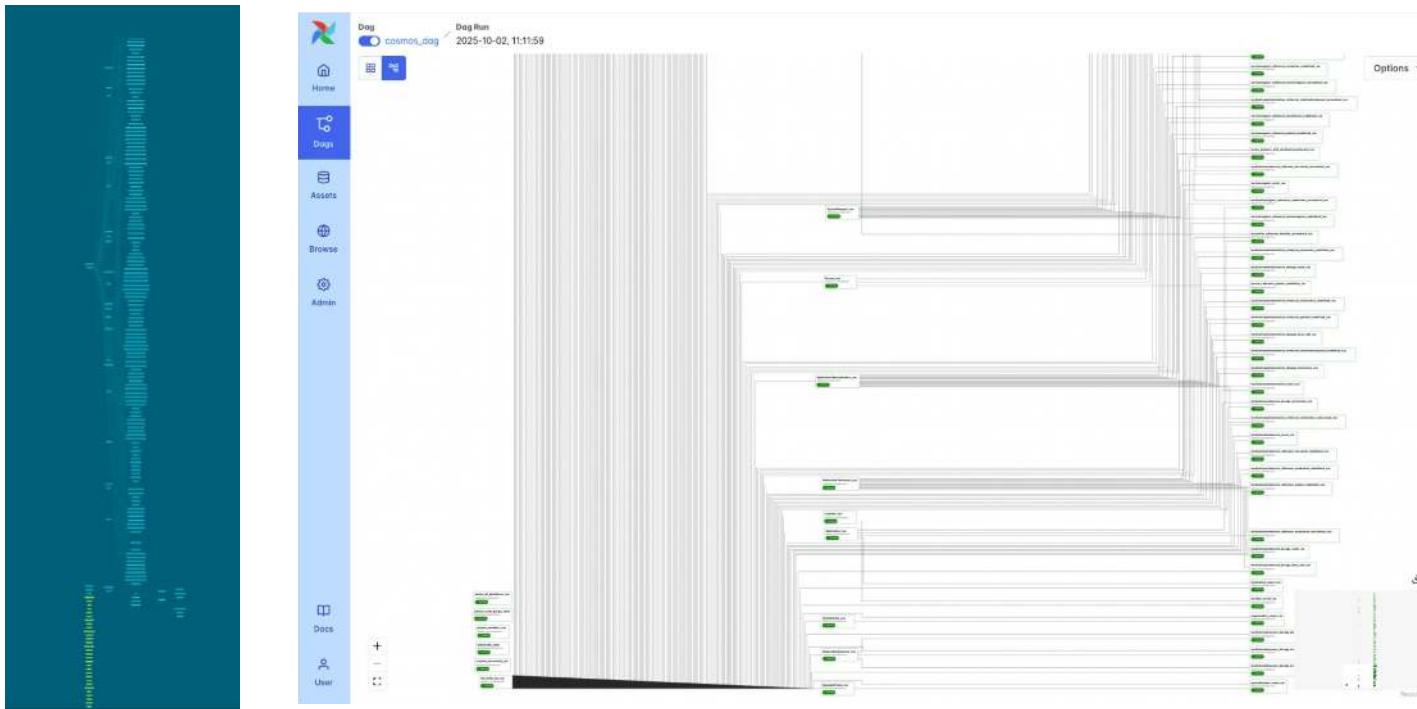
<https://github.com/dbt-labs/jaffle-shop-classic>

Synthetic workflows can not be representative



<https://github.com/astronomer/astronomer-cosmos/pull/827>

Real (open source) dbt project



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

4. Measurement & Metrics

Measurement & Metrics

- Core metrics:
 - DAG Run
 - Task Throughput
 - Error rate
 - Resource utilization
 - Memory
- Secondary metrics:
 - Startup time
 - Queue time
- Monitor system health
 - Logs, GC, caching, throttling

Statistical Significance

- Run **multiple iterations**, don't rely on single runs.
- **Be aware of variance** (especially in cloud environments)
- Use **statistical techniques** (confidence intervals, standard deviation) to confirm results are stable:
 - Standard deviation
 - Percentiles (p50, p95, p99)

5. Implementation

Experiment goal

Understand Cosmos 1.10 performance compared to dbt Core and dbt Cloud, when splitting the execution of a dbt pipeline in one or multiple commands, using a representative dbt project.

Experiment goal

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
```

Experiment goal

cmd: 1



cmd: 3



#cmd: 13



Metrics considered

- Pipeline execution time
- Memory consumption (average and standard deviation)
- CPU (average and standard deviation)

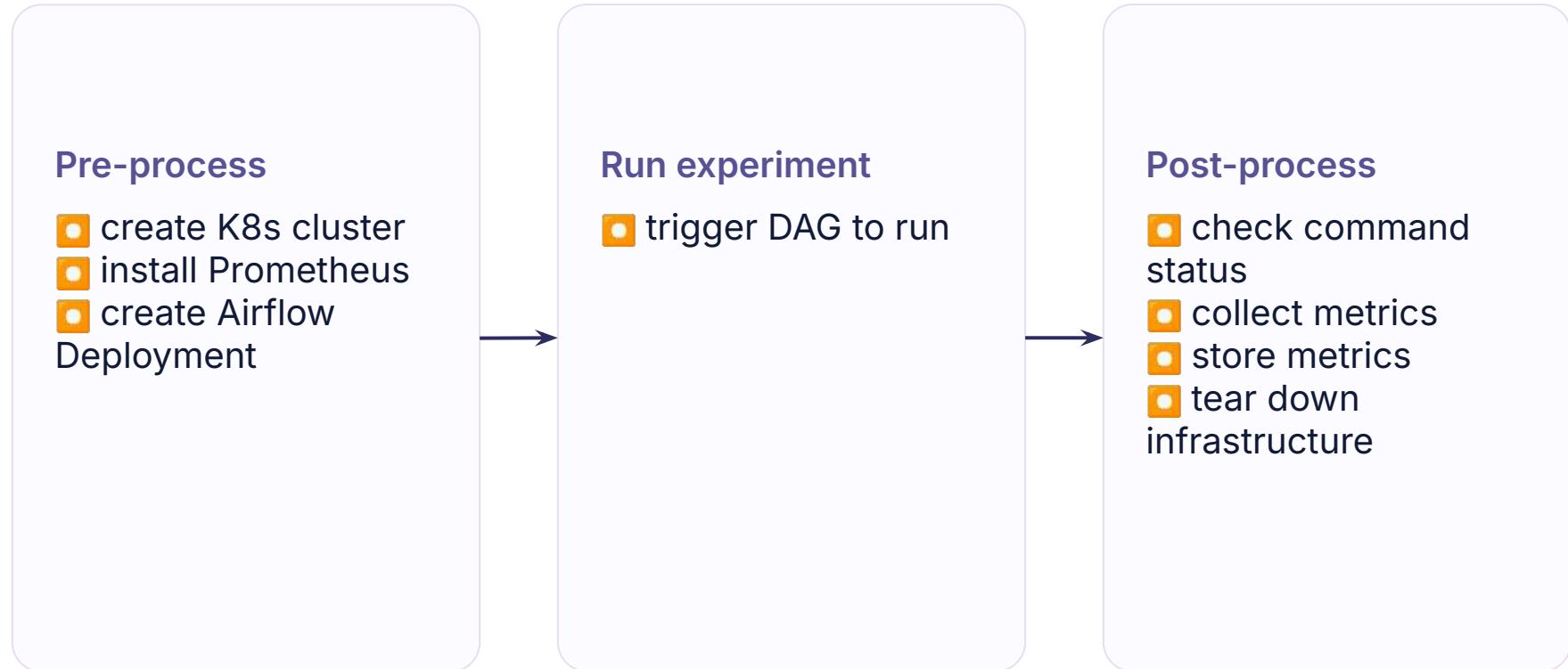
Benchmark experiment life cycle dbt Cloud



Benchmark experiment life cycle dbt Core



Benchmark experiment life cycle **Airflow**



Repository

We strongly believe that benchmarks should be public and reproducible by anyone in the community, and for this reason we've open-sourced this repository:

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

6. Results

Results

Platform	Airflow Command	Airflow DAG	dbt Command	Granularity	Duration	Max CPU Utilization	Stddev CPU Utilization	Max Memory Usage
dbt Cloud	N/A	N/A	dbt build	single command	0:05:10	N/A	N/A	N/A

Results

Platform	Airflow Command	Airflow DAG	dbt Command	Granularity	Duration	Max CPU Utilization	Stddev CPU Utilization	Max Memory Usage
dbt Cloud	N/A	N/A	dbt build	single command	0:05:10	N/A	N/A	N/A
dbt Core	N/A	N/A	dbt run	single command	0:05:05	0.39	0.06	306 MiB
dbt Core	N/A	N/A	dbt run	multi command (one per model)	0:31:50	0.39	0.06	306 MiB

<https://github.com/astro-databricks/cosmos-benchmark/pull/4>

<https://github.com/astro-databricks/cosmos-benchmark/pull/5>

Results

Platform	Airflow Command	Airflow DAG	dbt Command	Granularity	Duration	Max CPU Utilization	Stddev CPU Utilization	Max Memory Usage
dbt Cloud	N/A	N/A	dbt build	single command	0:05:10	N/A	N/A	N/A
dbt Core	N/A	N/A	dbt run	single command	0:05:05	0.39	0.06	306 MiB
dbt Core	N/A	N/A	dbt run	multi command (one per model)	0:31:50	0.39	0.06	306 MiB
Airflow OSS	airflow dags test	DbtBuildLocalOperator	dbt build	single command	0:05:59	0.18	0.03	537 MiB
Airflow OSS	airflow dags test	DbtDag	dbt run	multi command (one per model)	0:27:26	0.19	0.25	1 GiB

<https://github.com/astro-nomer/cosmos-benchmark/pull/6>

Results

Platform	Airflow Command	Airflow DAG	dbt Command	Granularity	Duration	Max CPU Utilization	Stddev CPU Utilization	Max Memory Usage
dbt Cloud	N/A	N/A	dbt build	single command	0:05:10	N/A	N/A	N/A
dbt Core	N/A	N/A	dbt run	single command	0:05:05	0.39	0.06	306 MiB
dbt Core	N/A	N/A	dbt run	multi command (one per model)	0:31:50	0.39	0.06	306 MiB
Airflow OSS	airflow dags test	DbtBuildLocalOperator	dbt build	single command	0:05:59	0.18	0.03	537 MiB
Airflow OSS	airflow dags test	DbtDag	dbt run	multi command (one per model)	0:27:26	0.19	0.25	1 GiB
Airflow OSS	airflow dags trigger	DbtBuildLocalOperator	dbt build	single command	0:05:50	1.3	0.09	1.6 GB
Airflow OSS	airflow dags trigger	DbtDag	dbt run	multi command (one per model)	0:15:13	3	0.15	2.5 GB

<https://github.com/astronomer/cosmos-benchmark/pull/7>



Boosting dbt Core Workflows Performance

With Airflow's Deferrable Capabilities

12:00 PT • Wednesday, October 8, 2025



Pankaj Koti
Software Engineer
@ Astronomer
Apache Airflow Committer



Tatiana Al-Chueyr
Staff Software Engineer
@ Astronomer
Cosmos Tech Lead



Pankaj Singh
Senior Software Engineer
@ Astronomer
Apache Airflow Committer

7. Next steps

Next steps

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

- Have a **configuration-driven** approach to run the tests - and track those over time
- Leverage **Airflow 3 APIs** to trigger and monitor the status of Airflow jobs
- Store **results consistently** in a way we can track experiments over time - publically
- Automate tests via the **CI** based on changes
- Collect **more metrics**
- Extend benchmark to run in **Astro**

8. Take away

Performance/benchmark testing isn't just about
running stress tools—it's about designing **fair**,
reproducible, and **meaningful** experiments that
guide decision-making.

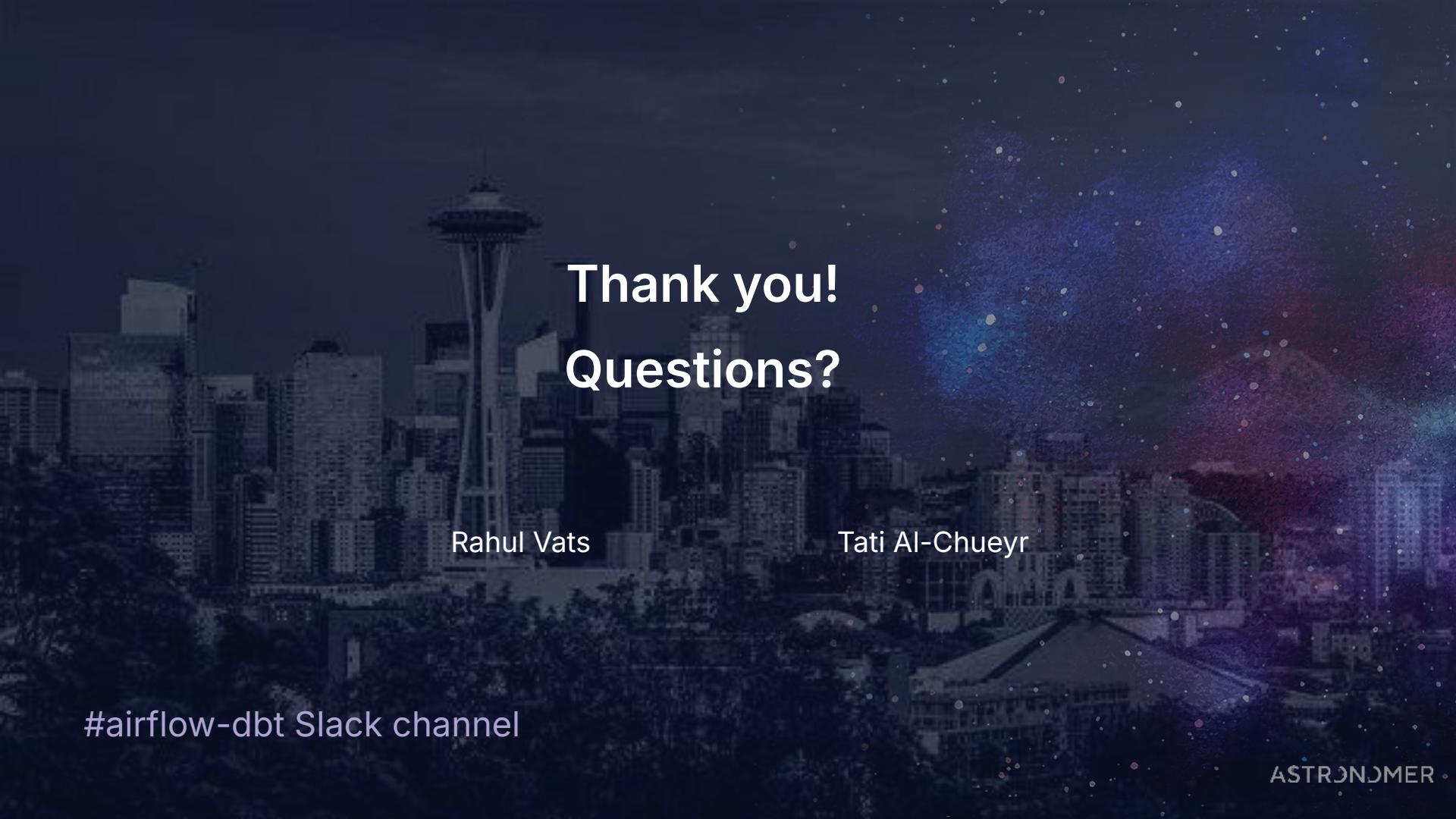
We need to have a **Open Source standard** to run
benchmarks on Airflow to allow the project to
continue being a **leader** among **orchestration tools**



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

The image shows the front cover of a book titled "Orchestrating dbt with Apache Airflow® using Cosmos". The cover is dark blue with a red spine. It features a stylized orange and green graphic of two overlapping shapes resembling wings or arrows. The title is written in white and yellow text. To the left of the book, there is a vertical column of Python code. The code defines a DAG (Directed Acyclic Graph) named "example_injection_dag". It uses the dbt library to define a project and a dbt task. The dbt task is annotated with the "operator_type" parameter set to "Apache Airflow". The code also includes logic to skip tasks if they have already run. The code is color-coded with syntax highlighting for readability.

```
1  dbt_project = dbt.DbtProject()
2
3  dbt_project.project_name = "cosmos"
4  dbt_project.target_name = "dev"
5  dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
6      conn_id=POSTGRES_CONN_ID,
7      profile_args={"schema": SCHEMA_NAME},
8  ),
9
10
11  # Only needed if you want to use dbt as an operator
12  _execution_config = dbt_execution_config()
13  dbt_executable = dbt_executable(_execution_config)
14
15
16  @dag(
17      params={
18          "my_department": "DEFINITION"
19      },
20      type="dbt"
21  )
22  def example_injection_dag():
23      dbt_task = dbt_task(dbt_project=dbt_project,
24                          dbt_executable=dbt_executable,
25                          operator_type="Apache Airflow")
26
27      dbt_task.skip_if_null(task_id="pre_dbt")
28
29      _pre_dbt = dbt_task()
30
31      return _pre_dbt
32
33
34  _pre_dbt = _pre_dbt()
35
36  dbt_project = dbt.DbtProject()
37  dbt_project.project_name = "cosmos"
38  dbt_project.target_name = "dev"
39  dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
40      conn_id=POSTGRES_CONN_ID,
41      profile_args={"schema": SCHEMA_NAME},
42  ),
43
44  # Only needed if you want to use dbt as an operator
45  _execution_config = dbt_execution_config()
46  dbt_executable = dbt_executable(_execution_config)
47
48
49  @dag(
50      params={
51          "my_department": "DEFINITION"
52      },
53      type="dbt"
54  )
55  def example_injection_dag():
56      dbt_task = dbt_task(dbt_project=dbt_project,
57                          dbt_executable=dbt_executable,
58                          operator_type="Apache Airflow")
59
60      dbt_task.skip_if_null(task_id="pre_dbt")
61
62      _pre_dbt = dbt_task()
63
64      return _pre_dbt
65
66
67  _pre_dbt = _pre_dbt()
68
69  dbt_project = dbt.DbtProject()
70  dbt_project.project_name = "cosmos"
71  dbt_project.target_name = "dev"
72  dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
73      conn_id=POSTGRES_CONN_ID,
74      profile_args={"schema": SCHEMA_NAME},
75  ),
76
77  # Only needed if you want to use dbt as an operator
78  _execution_config = dbt_execution_config()
79  dbt_executable = dbt_executable(_execution_config)
80
81
82  @dag(
83      params={
84          "my_department": "DEFINITION"
85      },
86      type="dbt"
87  )
88  def example_injection_dag():
89      dbt_task = dbt_task(dbt_project=dbt_project,
90                          dbt_executable=dbt_executable,
91                          operator_type="Apache Airflow")
92
93      dbt_task.skip_if_null(task_id="pre_dbt")
94
95      _pre_dbt = dbt_task()
96
97      return _pre_dbt
98
99
100 _pre_dbt = _pre_dbt()
101
102 dbt_project = dbt.DbtProject()
103 dbt_project.project_name = "cosmos"
104 dbt_project.target_name = "dev"
105 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
106     conn_id=POSTGRES_CONN_ID,
107     profile_args={"schema": SCHEMA_NAME},
108 ),
109
110 # Only needed if you want to use dbt as an operator
111 _execution_config = dbt_execution_config()
112 dbt_executable = dbt_executable(_execution_config)
113
114
115 @dag(
116     params={
117         "my_department": "DEFINITION"
118     },
119     type="dbt"
120 )
121 def example_injection_dag():
122     dbt_task = dbt_task(dbt_project=dbt_project,
123                         dbt_executable=dbt_executable,
124                         operator_type="Apache Airflow")
125
126     dbt_task.skip_if_null(task_id="pre_dbt")
127
128     _pre_dbt = dbt_task()
129
130     return _pre_dbt
131
132
133 _pre_dbt = _pre_dbt()
134
135 dbt_project = dbt.DbtProject()
136 dbt_project.project_name = "cosmos"
137 dbt_project.target_name = "dev"
138 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
139     conn_id=POSTGRES_CONN_ID,
140     profile_args={"schema": SCHEMA_NAME},
141 ),
142
143 # Only needed if you want to use dbt as an operator
144 _execution_config = dbt_execution_config()
145 dbt_executable = dbt_executable(_execution_config)
146
147
148 @dag(
149     params={
150         "my_department": "DEFINITION"
151     },
152     type="dbt"
153 )
154 def example_injection_dag():
155     dbt_task = dbt_task(dbt_project=dbt_project,
156                         dbt_executable=dbt_executable,
157                         operator_type="Apache Airflow")
158
159     dbt_task.skip_if_null(task_id="pre_dbt")
160
161     _pre_dbt = dbt_task()
162
163     return _pre_dbt
164
165
166 _pre_dbt = _pre_dbt()
167
168 dbt_project = dbt.DbtProject()
169 dbt_project.project_name = "cosmos"
170 dbt_project.target_name = "dev"
171 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
172     conn_id=POSTGRES_CONN_ID,
173     profile_args={"schema": SCHEMA_NAME},
174 ),
175
176 # Only needed if you want to use dbt as an operator
177 _execution_config = dbt_execution_config()
178 dbt_executable = dbt_executable(_execution_config)
179
180
181 @dag(
182     params={
183         "my_department": "DEFINITION"
184     },
185     type="dbt"
186 )
187 def example_injection_dag():
188     dbt_task = dbt_task(dbt_project=dbt_project,
189                         dbt_executable=dbt_executable,
190                         operator_type="Apache Airflow")
191
192     dbt_task.skip_if_null(task_id="pre_dbt")
193
194     _pre_dbt = dbt_task()
195
196     return _pre_dbt
197
198
199 _pre_dbt = _pre_dbt()
200
201 dbt_project = dbt.DbtProject()
202 dbt_project.project_name = "cosmos"
203 dbt_project.target_name = "dev"
204 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
205     conn_id=POSTGRES_CONN_ID,
206     profile_args={"schema": SCHEMA_NAME},
207 ),
208
209 # Only needed if you want to use dbt as an operator
210 _execution_config = dbt_execution_config()
211 dbt_executable = dbt_executable(_execution_config)
212
213
214 @dag(
215     params={
216         "my_department": "DEFINITION"
217     },
218     type="dbt"
219 )
220 def example_injection_dag():
221     dbt_task = dbt_task(dbt_project=dbt_project,
222                         dbt_executable=dbt_executable,
223                         operator_type="Apache Airflow")
224
225     dbt_task.skip_if_null(task_id="pre_dbt")
226
227     _pre_dbt = dbt_task()
228
229     return _pre_dbt
230
231
232 _pre_dbt = _pre_dbt()
233
234 dbt_project = dbt.DbtProject()
235 dbt_project.project_name = "cosmos"
236 dbt_project.target_name = "dev"
237 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
238     conn_id=POSTGRES_CONN_ID,
239     profile_args={"schema": SCHEMA_NAME},
240 ),
241
242 # Only needed if you want to use dbt as an operator
243 _execution_config = dbt_execution_config()
244 dbt_executable = dbt_executable(_execution_config)
245
246
247 @dag(
248     params={
249         "my_department": "DEFINITION"
250     },
251     type="dbt"
252 )
253 def example_injection_dag():
254     dbt_task = dbt_task(dbt_project=dbt_project,
255                         dbt_executable=dbt_executable,
256                         operator_type="Apache Airflow")
257
258     dbt_task.skip_if_null(task_id="pre_dbt")
259
260     _pre_dbt = dbt_task()
261
262     return _pre_dbt
263
264
265 _pre_dbt = _pre_dbt()
266
267 dbt_project = dbt.DbtProject()
268 dbt_project.project_name = "cosmos"
269 dbt_project.target_name = "dev"
270 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
271     conn_id=POSTGRES_CONN_ID,
272     profile_args={"schema": SCHEMA_NAME},
273 ),
274
275 # Only needed if you want to use dbt as an operator
276 _execution_config = dbt_execution_config()
277 dbt_executable = dbt_executable(_execution_config)
278
279
280 @dag(
281     params={
282         "my_department": "DEFINITION"
283     },
284     type="dbt"
285 )
286 def example_injection_dag():
287     dbt_task = dbt_task(dbt_project=dbt_project,
288                         dbt_executable=dbt_executable,
289                         operator_type="Apache Airflow")
290
291     dbt_task.skip_if_null(task_id="pre_dbt")
292
293     _pre_dbt = dbt_task()
294
295     return _pre_dbt
296
297
298 _pre_dbt = _pre_dbt()
299
300 dbt_project = dbt.DbtProject()
301 dbt_project.project_name = "cosmos"
302 dbt_project.target_name = "dev"
303 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
304     conn_id=POSTGRES_CONN_ID,
305     profile_args={"schema": SCHEMA_NAME},
306 ),
307
308 # Only needed if you want to use dbt as an operator
309 _execution_config = dbt_execution_config()
310 dbt_executable = dbt_executable(_execution_config)
311
312
313 @dag(
314     params={
315         "my_department": "DEFINITION"
316     },
317     type="dbt"
318 )
319 def example_injection_dag():
320     dbt_task = dbt_task(dbt_project=dbt_project,
321                         dbt_executable=dbt_executable,
322                         operator_type="Apache Airflow")
323
324     dbt_task.skip_if_null(task_id="pre_dbt")
325
326     _pre_dbt = dbt_task()
327
328     return _pre_dbt
329
330
331 _pre_dbt = _pre_dbt()
332
333 dbt_project = dbt.DbtProject()
334 dbt_project.project_name = "cosmos"
335 dbt_project.target_name = "dev"
336 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
337     conn_id=POSTGRES_CONN_ID,
338     profile_args={"schema": SCHEMA_NAME},
339 ),
340
341 # Only needed if you want to use dbt as an operator
342 _execution_config = dbt_execution_config()
343 dbt_executable = dbt_executable(_execution_config)
344
345
346 @dag(
347     params={
348         "my_department": "DEFINITION"
349     },
350     type="dbt"
351 )
352 def example_injection_dag():
353     dbt_task = dbt_task(dbt_project=dbt_project,
354                         dbt_executable=dbt_executable,
355                         operator_type="Apache Airflow")
356
357     dbt_task.skip_if_null(task_id="pre_dbt")
358
359     _pre_dbt = dbt_task()
360
361     return _pre_dbt
362
363
364 _pre_dbt = _pre_dbt()
365
366 dbt_project = dbt.DbtProject()
367 dbt_project.project_name = "cosmos"
368 dbt_project.target_name = "dev"
369 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
370     conn_id=POSTGRES_CONN_ID,
371     profile_args={"schema": SCHEMA_NAME},
372 ),
373
374 # Only needed if you want to use dbt as an operator
375 _execution_config = dbt_execution_config()
376 dbt_executable = dbt_executable(_execution_config)
377
378
379 @dag(
380     params={
381         "my_department": "DEFINITION"
382     },
383     type="dbt"
384 )
385 def example_injection_dag():
386     dbt_task = dbt_task(dbt_project=dbt_project,
387                         dbt_executable=dbt_executable,
388                         operator_type="Apache Airflow")
389
390     dbt_task.skip_if_null(task_id="pre_dbt")
391
392     _pre_dbt = dbt_task()
393
394     return _pre_dbt
395
396
397 _pre_dbt = _pre_dbt()
398
399 dbt_project = dbt.DbtProject()
400 dbt_project.project_name = "cosmos"
401 dbt_project.target_name = "dev"
402 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
403     conn_id=POSTGRES_CONN_ID,
404     profile_args={"schema": SCHEMA_NAME},
405 ),
406
407 # Only needed if you want to use dbt as an operator
408 _execution_config = dbt_execution_config()
409 dbt_executable = dbt_executable(_execution_config)
410
411
412 @dag(
413     params={
414         "my_department": "DEFINITION"
415     },
416     type="dbt"
417 )
418 def example_injection_dag():
419     dbt_task = dbt_task(dbt_project=dbt_project,
420                         dbt_executable=dbt_executable,
421                         operator_type="Apache Airflow")
422
423     dbt_task.skip_if_null(task_id="pre_dbt")
424
425     _pre_dbt = dbt_task()
426
427     return _pre_dbt
428
429
430 _pre_dbt = _pre_dbt()
431
432 dbt_project = dbt.DbtProject()
433 dbt_project.project_name = "cosmos"
434 dbt_project.target_name = "dev"
435 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
436     conn_id=POSTGRES_CONN_ID,
437     profile_args={"schema": SCHEMA_NAME},
438 ),
439
440 # Only needed if you want to use dbt as an operator
441 _execution_config = dbt_execution_config()
442 dbt_executable = dbt_executable(_execution_config)
443
444
445 @dag(
446     params={
447         "my_department": "DEFINITION"
448     },
449     type="dbt"
450 )
451 def example_injection_dag():
452     dbt_task = dbt_task(dbt_project=dbt_project,
453                         dbt_executable=dbt_executable,
454                         operator_type="Apache Airflow")
455
456     dbt_task.skip_if_null(task_id="pre_dbt")
457
458     _pre_dbt = dbt_task()
459
460     return _pre_dbt
461
462
463 _pre_dbt = _pre_dbt()
464
465 dbt_project = dbt.DbtProject()
466 dbt_project.project_name = "cosmos"
467 dbt_project.target_name = "dev"
468 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
469     conn_id=POSTGRES_CONN_ID,
470     profile_args={"schema": SCHEMA_NAME},
471 ),
472
473 # Only needed if you want to use dbt as an operator
474 _execution_config = dbt_execution_config()
475 dbt_executable = dbt_executable(_execution_config)
476
477
478 @dag(
479     params={
480         "my_department": "DEFINITION"
481     },
482     type="dbt"
483 )
484 def example_injection_dag():
485     dbt_task = dbt_task(dbt_project=dbt_project,
486                         dbt_executable=dbt_executable,
487                         operator_type="Apache Airflow")
488
489     dbt_task.skip_if_null(task_id="pre_dbt")
490
491     _pre_dbt = dbt_task()
492
493     return _pre_dbt
494
495
496 _pre_dbt = _pre_dbt()
497
498 dbt_project = dbt.DbtProject()
499 dbt_project.project_name = "cosmos"
500 dbt_project.target_name = "dev"
501 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
502     conn_id=POSTGRES_CONN_ID,
503     profile_args={"schema": SCHEMA_NAME},
504 ),
505
506 # Only needed if you want to use dbt as an operator
507 _execution_config = dbt_execution_config()
508 dbt_executable = dbt_executable(_execution_config)
509
510
511 @dag(
512     params={
513         "my_department": "DEFINITION"
514     },
515     type="dbt"
516 )
517 def example_injection_dag():
518     dbt_task = dbt_task(dbt_project=dbt_project,
519                         dbt_executable=dbt_executable,
520                         operator_type="Apache Airflow")
521
522     dbt_task.skip_if_null(task_id="pre_dbt")
523
524     _pre_dbt = dbt_task()
525
526     return _pre_dbt
527
528
529 _pre_dbt = _pre_dbt()
530
531 dbt_project = dbt.DbtProject()
532 dbt_project.project_name = "cosmos"
533 dbt_project.target_name = "dev"
534 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
535     conn_id=POSTGRES_CONN_ID,
536     profile_args={"schema": SCHEMA_NAME},
537 ),
538
539 # Only needed if you want to use dbt as an operator
540 _execution_config = dbt_execution_config()
541 dbt_executable = dbt_executable(_execution_config)
542
543
544 @dag(
545     params={
546         "my_department": "DEFINITION"
547     },
548     type="dbt"
549 )
550 def example_injection_dag():
551     dbt_task = dbt_task(dbt_project=dbt_project,
552                         dbt_executable=dbt_executable,
553                         operator_type="Apache Airflow")
554
555     dbt_task.skip_if_null(task_id="pre_dbt")
556
557     _pre_dbt = dbt_task()
558
559     return _pre_dbt
560
561
562 _pre_dbt = _pre_dbt()
563
564 dbt_project = dbt.DbtProject()
565 dbt_project.project_name = "cosmos"
566 dbt_project.target_name = "dev"
567 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
568     conn_id=POSTGRES_CONN_ID,
569     profile_args={"schema": SCHEMA_NAME},
570 ),
571
572 # Only needed if you want to use dbt as an operator
573 _execution_config = dbt_execution_config()
574 dbt_executable = dbt_executable(_execution_config)
575
576
577 @dag(
578     params={
579         "my_department": "DEFINITION"
580     },
581     type="dbt"
582 )
583 def example_injection_dag():
584     dbt_task = dbt_task(dbt_project=dbt_project,
585                         dbt_executable=dbt_executable,
586                         operator_type="Apache Airflow")
587
588     dbt_task.skip_if_null(task_id="pre_dbt")
589
590     _pre_dbt = dbt_task()
591
592     return _pre_dbt
593
594
595 _pre_dbt = _pre_dbt()
596
597 dbt_project = dbt.DbtProject()
598 dbt_project.project_name = "cosmos"
599 dbt_project.target_name = "dev"
600 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
601     conn_id=POSTGRES_CONN_ID,
602     profile_args={"schema": SCHEMA_NAME},
603 ),
604
605 # Only needed if you want to use dbt as an operator
606 _execution_config = dbt_execution_config()
607 dbt_executable = dbt_executable(_execution_config)
608
609
610 @dag(
611     params={
612         "my_department": "DEFINITION"
613     },
614     type="dbt"
615 )
616 def example_injection_dag():
617     dbt_task = dbt_task(dbt_project=dbt_project,
618                         dbt_executable=dbt_executable,
619                         operator_type="Apache Airflow")
620
621     dbt_task.skip_if_null(task_id="pre_dbt")
622
623     _pre_dbt = dbt_task()
624
625     return _pre_dbt
626
627
628 _pre_dbt = _pre_dbt()
629
630 dbt_project = dbt.DbtProject()
631 dbt_project.project_name = "cosmos"
632 dbt_project.target_name = "dev"
633 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
634     conn_id=POSTGRES_CONN_ID,
635     profile_args={"schema": SCHEMA_NAME},
636 ),
637
638 # Only needed if you want to use dbt as an operator
639 _execution_config = dbt_execution_config()
640 dbt_executable = dbt_executable(_execution_config)
641
642
643 @dag(
644     params={
645         "my_department": "DEFINITION"
646     },
647     type="dbt"
648 )
649 def example_injection_dag():
650     dbt_task = dbt_task(dbt_project=dbt_project,
651                         dbt_executable=dbt_executable,
652                         operator_type="Apache Airflow")
653
654     dbt_task.skip_if_null(task_id="pre_dbt")
655
656     _pre_dbt = dbt_task()
657
658     return _pre_dbt
659
660
661 _pre_dbt = _pre_dbt()
662
663 dbt_project = dbt.DbtProject()
664 dbt_project.project_name = "cosmos"
665 dbt_project.target_name = "dev"
666 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
667     conn_id=POSTGRES_CONN_ID,
668     profile_args={"schema": SCHEMA_NAME},
669 ),
670
671 # Only needed if you want to use dbt as an operator
672 _execution_config = dbt_execution_config()
673 dbt_executable = dbt_executable(_execution_config)
674
675
676 @dag(
677     params={
678         "my_department": "DEFINITION"
679     },
680     type="dbt"
681 )
682 def example_injection_dag():
683     dbt_task = dbt_task(dbt_project=dbt_project,
684                         dbt_executable=dbt_executable,
685                         operator_type="Apache Airflow")
686
687     dbt_task.skip_if_null(task_id="pre_dbt")
688
689     _pre_dbt = dbt_task()
690
691     return _pre_dbt
692
693
694 _pre_dbt = _pre_dbt()
695
696 dbt_project = dbt.DbtProject()
697 dbt_project.project_name = "cosmos"
698 dbt_project.target_name = "dev"
699 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
700     conn_id=POSTGRES_CONN_ID,
701     profile_args={"schema": SCHEMA_NAME},
702 ),
703
704 # Only needed if you want to use dbt as an operator
705 _execution_config = dbt_execution_config()
706 dbt_executable = dbt_executable(_execution_config)
707
708
709 @dag(
710     params={
711         "my_department": "DEFINITION"
712     },
713     type="dbt"
714 )
715 def example_injection_dag():
716     dbt_task = dbt_task(dbt_project=dbt_project,
717                         dbt_executable=dbt_executable,
718                         operator_type="Apache Airflow")
719
720     dbt_task.skip_if_null(task_id="pre_dbt")
721
722     _pre_dbt = dbt_task()
723
724     return _pre_dbt
725
726
727 _pre_dbt = _pre_dbt()
728
729 dbt_project = dbt.DbtProject()
730 dbt_project.project_name = "cosmos"
731 dbt_project.target_name = "dev"
732 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
733     conn_id=POSTGRES_CONN_ID,
734     profile_args={"schema": SCHEMA_NAME},
735 ),
736
737 # Only needed if you want to use dbt as an operator
738 _execution_config = dbt_execution_config()
739 dbt_executable = dbt_executable(_execution_config)
740
741
742 @dag(
743     params={
744         "my_department": "DEFINITION"
745     },
746     type="dbt"
747 )
748 def example_injection_dag():
749     dbt_task = dbt_task(dbt_project=dbt_project,
750                         dbt_executable=dbt_executable,
751                         operator_type="Apache Airflow")
752
753     dbt_task.skip_if_null(task_id="pre_dbt")
754
755     _pre_dbt = dbt_task()
756
757     return _pre_dbt
758
759
760 _pre_dbt = _pre_dbt()
761
762 dbt_project = dbt.DbtProject()
763 dbt_project.project_name = "cosmos"
764 dbt_project.target_name = "dev"
765 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
766     conn_id=POSTGRES_CONN_ID,
767     profile_args={"schema": SCHEMA_NAME},
768 ),
769
770 # Only needed if you want to use dbt as an operator
771 _execution_config = dbt_execution_config()
772 dbt_executable = dbt_executable(_execution_config)
773
774
775 @dag(
776     params={
777         "my_department": "DEFINITION"
778     },
779     type="dbt"
780 )
781 def example_injection_dag():
782     dbt_task = dbt_task(dbt_project=dbt_project,
783                         dbt_executable=dbt_executable,
784                         operator_type="Apache Airflow")
785
786     dbt_task.skip_if_null(task_id="pre_dbt")
787
788     _pre_dbt = dbt_task()
789
790     return _pre_dbt
791
792
793 _pre_dbt = _pre_dbt()
794
795 dbt_project = dbt.DbtProject()
796 dbt_project.project_name = "cosmos"
797 dbt_project.target_name = "dev"
798 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
799     conn_id=POSTGRES_CONN_ID,
800     profile_args={"schema": SCHEMA_NAME},
801 ),
802
803 # Only needed if you want to use dbt as an operator
804 _execution_config = dbt_execution_config()
805 dbt_executable = dbt_executable(_execution_config)
806
807
808 @dag(
809     params={
810         "my_department": "DEFINITION"
811     },
812     type="dbt"
813 )
814 def example_injection_dag():
815     dbt_task = dbt_task(dbt_project=dbt_project,
816                         dbt_executable=dbt_executable,
817                         operator_type="Apache Airflow")
818
819     dbt_task.skip_if_null(task_id="pre_dbt")
820
821     _pre_dbt = dbt_task()
822
823     return _pre_dbt
824
825
826 _pre_dbt = _pre_dbt()
827
828 dbt_project = dbt.DbtProject()
829 dbt_project.project_name = "cosmos"
830 dbt_project.target_name = "dev"
831 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
832     conn_id=POSTGRES_CONN_ID,
833     profile_args={"schema": SCHEMA_NAME},
834 ),
835
836 # Only needed if you want to use dbt as an operator
837 _execution_config = dbt_execution_config()
838 dbt_executable = dbt_executable(_execution_config)
839
840
841 @dag(
842     params={
843         "my_department": "DEFINITION"
844     },
845     type="dbt"
846 )
847 def example_injection_dag():
848     dbt_task = dbt_task(dbt_project=dbt_project,
849                         dbt_executable=dbt_executable,
850                         operator_type="Apache Airflow")
851
852     dbt_task.skip_if_null(task_id="pre_dbt")
853
854     _pre_dbt = dbt_task()
855
856     return _pre_dbt
857
858
859 _pre_dbt = _pre_dbt()
860
861 dbt_project = dbt.DbtProject()
862 dbt_project.project_name = "cosmos"
863 dbt_project.target_name = "dev"
864 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
865     conn_id=POSTGRES_CONN_ID,
866     profile_args={"schema": SCHEMA_NAME},
867 ),
868
869 # Only needed if you want to use dbt as an operator
870 _execution_config = dbt_execution_config()
871 dbt_executable = dbt_executable(_execution_config)
872
873
874 @dag(
875     params={
876         "my_department": "DEFINITION"
877     },
878     type="dbt"
879 )
880 def example_injection_dag():
881     dbt_task = dbt_task(dbt_project=dbt_project,
882                         dbt_executable=dbt_executable,
883                         operator_type="Apache Airflow")
884
885     dbt_task.skip_if_null(task_id="pre_dbt")
886
887     _pre_dbt = dbt_task()
888
889     return _pre_dbt
890
891
892 _pre_dbt = _pre_dbt()
893
894 dbt_project = dbt.DbtProject()
895 dbt_project.project_name = "cosmos"
896 dbt_project.target_name = "dev"
897 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
898     conn_id=POSTGRES_CONN_ID,
899     profile_args={"schema": SCHEMA_NAME},
900 ),
901
902 # Only needed if you want to use dbt as an operator
903 _execution_config = dbt_execution_config()
904 dbt_executable = dbt_executable(_execution_config)
905
906
907 @dag(
908     params={
909         "my_department": "DEFINITION"
910     },
911     type="dbt"
912 )
913 def example_injection_dag():
914     dbt_task = dbt_task(dbt_project=dbt_project,
915                         dbt_executable=dbt_executable,
916                         operator_type="Apache Airflow")
917
918     dbt_task.skip_if_null(task_id="pre_dbt")
919
920     _pre_dbt = dbt_task()
921
922     return _pre_dbt
923
924
925 _pre_dbt = _pre_dbt()
926
927 dbt_project = dbt.DbtProject()
928 dbt_project.project_name = "cosmos"
929 dbt_project.target_name = "dev"
930 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
931     conn_id=POSTGRES_CONN_ID,
932     profile_args={"schema": SCHEMA_NAME},
933 ),
934
935 # Only needed if you want to use dbt as an operator
936 _execution_config = dbt_execution_config()
937 dbt_executable = dbt_executable(_execution_config)
938
939
940 @dag(
941     params={
942         "my_department": "DEFINITION"
943     },
944     type="dbt"
945 )
946 def example_injection_dag():
947     dbt_task = dbt_task(dbt_project=dbt_project,
948                         dbt_executable=dbt_executable,
949                         operator_type="Apache Airflow")
950
951     dbt_task.skip_if_null(task_id="pre_dbt")
952
953     _pre_dbt = dbt_task()
954
955     return _pre_dbt
956
957
958 _pre_dbt = _pre_dbt()
959
960 dbt_project = dbt.DbtProject()
961 dbt_project.project_name = "cosmos"
962 dbt_project.target_name = "dev"
963 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
964     conn_id=POSTGRES_CONN_ID,
965     profile_args={"schema": SCHEMA_NAME},
966 ),
967
968 # Only needed if you want to use dbt as an operator
969 _execution_config = dbt_execution_config()
970 dbt_executable = dbt_executable(_execution_config)
971
972
973 @dag(
974     params={
975         "my_department": "DEFINITION"
976     },
977     type="dbt"
978 )
979 def example_injection_dag():
980     dbt_task = dbt_task(dbt_project=dbt_project,
981                         dbt_executable=dbt_executable,
982                         operator_type="Apache Airflow")
983
984     dbt_task.skip_if_null(task_id="pre_dbt")
985
986     _pre_dbt = dbt_task()
987
988     return _pre_dbt
989
990
991 _pre_dbt = _pre_dbt()
992
993 dbt_project = dbt.DbtProject()
994 dbt_project.project_name = "cosmos"
995 dbt_project.target_name = "dev"
996 dbt_project.profile_mapping = PostgresUserPasswordProfileMapping(
997     conn_id=POSTGRES_CONN_ID,
998     profile_args={"schema": SCHEMA_NAME},
999 ),
1000
1001 # Only needed if you want to use dbt as an operator
1002 _execution_config = dbt_execution_config()
1003 dbt_executable = dbt_executable(_execution_config)
```

A dark, atmospheric photograph of the Seattle skyline at night. The Space Needle is prominent in the center-left. The city lights are reflected in the water in the foreground, and a bridge arches across the scene. The background is a deep blue/purple.

Thank you! Questions?

Rahul Vats

Tati Al-Chueyr

#airflow-dbt Slack channel

ASTRONOMER