



Clearing Airflow obstructions

@tati_alchueyr

Principal Data Engineer @ BBC Datalab

Airflow
Summit 2021

Online
15 July 2021

Heads up!

- During this **session** there will be some **quizzes**
- Be prepared to either:
 - **Scan** the **QR code**
 - **Access** using the **URL**



@tati_ alchueyr. doc

- **Brazilian** living in London since 2014
- Principal Data **Engineer** at the **BBC Datalab** team
- Graduated in **Computer Engineering** at Unicamp
- **Passionate** software developer for **18 years**
- Experience in the **private** and **public** sectors
- Developed software for **Medicine**, **Media** and **Education**
- Loves **Open Source**
- Loves **Brazilian Jiu Jitsu**
- Proud mother of **Amanda** (v4.0)



I ❤️ Airflow Community & Summit



Tomek
Urbaszek



Jarek
Potiuk



Kaxil Naik



Ash
Berlin-Taylor



Leah
Cole

BBC.Datalab.Hummingbirds

The work presented here is the result of lots of **teamwork** within **one squad** of a much **larger team** and **organisation**



Darren
Mundy



David
Hollands



Richard
Bownes



Tatiana
Al-Chueyr



active squad team members



Bettina
Hermant



Marc
Oppenheimer



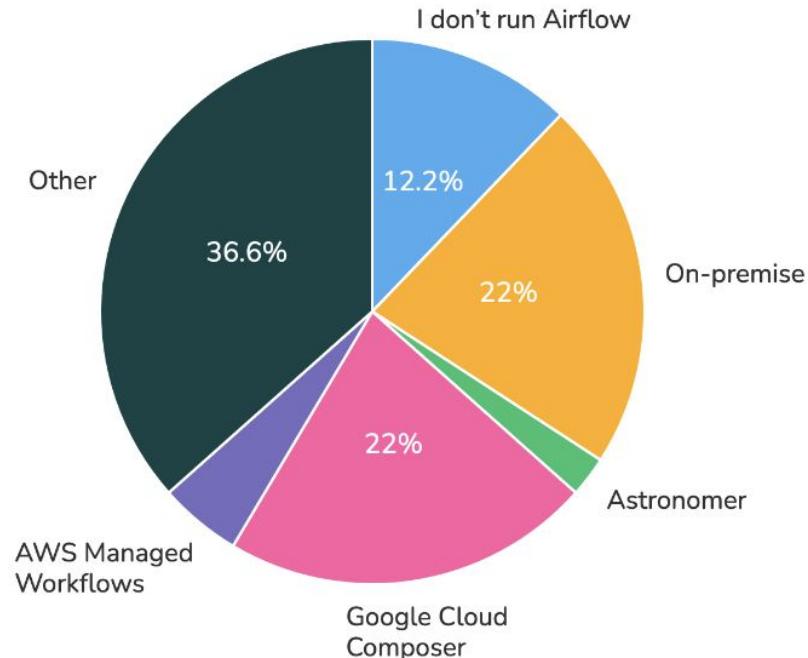
directly contributed in the past

Quiz time how do you run Airflow?

- A. I don't run Airflow
- B. On-premise
- C. Astronomer.io
- D. Google Cloud Composer
- E. AWS Managed Workflows
- F. Other



Quiz time how do you run Airflow?



* responses from Airflow Summit 2021 participants, during the presentation

How we use Airflow

when things went wrong

How we use Airflow when things go wrong

Getting Started

https://s9881134f9714f1edp-tp.appspot.com/admin/airflow/clear

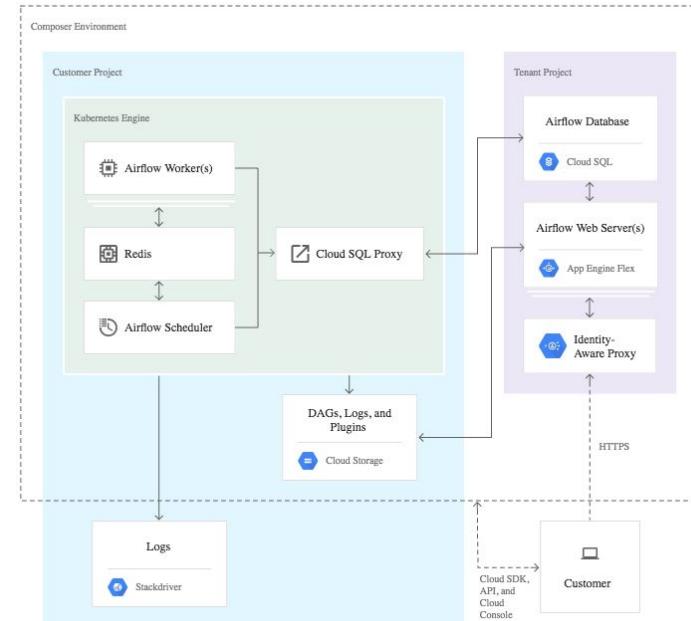
80%

Other Bookmarks

```
Node: a77e7fb0831f
Traceback (most recent call last):
  File "/opt/python3.6/lib/python3.6/site-packages/flask/app.py", line 2447, in wsgi_app
    response = self.wsgi_app(environ, start_response)
  File "/opt/python3.6/lib/python3.6/site-packages/flask/app.py", line 1952, in full_dispatch_request
    rv = self.handle_user_exception(e)
  File "/opt/python3.6/lib/python3.6/site-packages/flask/app.py", line 1821, in handle_user_exception
    reraise(exc_type, exc_value, tb)
  File "/opt/python3.6/lib/python3.6/site-packages/flask/_compat.py", line 39, in reraise
    raise value
  File "/opt/python3.6/lib/python3.6/site-packages/flask/app.py", line 1958, in full_dispatch_request
    rv = self.dispatch_request(*args, **kwargs)
  File "/opt/python3.6/lib/python3.6/site-packages/flask/app.py", line 1936, in dispatch_request
    return self.view_functions[rule.endpoint](**req.view_args)
  File "/opt/python3.6/lib/python3.6/site-packages/flask_admin/base.py", line 69, in inner
    return self._run_view(**kwargs)
  File "/opt/python3.6/lib/python3.6/site-packages/flask_admin/base.py", line 368, in _run_view
    return fn(self, *args, **kwargs)
  File "/opt/python3.6/lib/python3.6/site-packages/flask_login/utils.py", line 258, in decorated_view
    return func(*args, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/airflow/www/utils.py", line 292, in wrapper
    return f(*args, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/airflow/www/utils.py", line 339, in wrapper
    return f(*args, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/airflow/www/views.py", line 1363, in clear
    include_upstreamupstream)
  File "/usr/local/lib/python3.6/dist-packages/airflow/www/models/dag.py", line 1253, in sub_dag
    for t in regex_match:
  File "/usr/local/lib/python3.6/dist-packages/airflow/www/models/dag.py", line 1253, in <dictcomp>
    for t in regex_match + also_include
  File "/opt/python3.6/lib/python3.6/copy.py", line 161, in deepcopy
    y = copier(co)
  File "/usr/local/lib/python3.6/dist-packages/airflow/www/models/baseoperator.py", line 687, in __deepcopy__
    setattr(result, k, copy.deepcopy(v, memo))
  File "/opt/python3.6/lib/python3.6/copy.py", line 188, in deepcopy
    memo[id(obj)], memo, rrv)
  File "/opt/python3.6/lib/python3.6/copy.py", line 274, in _reconstruct
    y = func(*args)
  File "/home/airflow/gcs/dags/config/variant.py", line 12, in __new__
    cls.gcs_bucket = kwargs.pop("gcs_bucket")
KeyError: 'gcs_bucket'
```

How we use Airflow **infrastructure**

- **Managed** Airflow
 - Cloud Composer (GCP)
 - Terraform
- **Celery Executors** GCP constraint
 - Running within Kubernetes (GKE)
- **Outdated** version **1.10.12**
 - Upgrades have been time consuming
 - Last: 1.10.4 => 1.10.12 @ Dec '20
 - GCP supports newer releases
 - 1.10.5 since April '21 (March '21)
 - 2.0.1 since May '21 (Feb '21)





How we use Airflow **operators**

- Execute within **Airflow executors**
 - BaseOperator
 - BashOperator
 - DummyOperator
 - **PythonOperator**
 - ShortCircuitOperator
 - **TriggerDagRunOperator**
 - GCSDeleteObjectsOperator
- Delegate to **Kubernetes** in a dedicated GKE cluster
 - **GKEPodOperator**
- Delegate to **Apache Beam** (Dataflow)
 - **DataflowPythonOperator**
 - DataflowCreatePythonJobOperator

How we use Airflow local, dev, int, prod

Airflow Dags Data Profiling Browse Admin Docs About

database-sounds-dev-mauve 2021-07-14 23:47:36 UTC

DAGs

Search:

DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
airflow_monitoring	None	airflow	Hummingbirds	2021-07-14 23:41	1	0
backfill_user_activity	1 day, 00:00:00	Hummingbirds	Hummingbirds	2021-05-13 13:00	1	0
ingest_availability	hourly	Hummingbirds	Hummingbirds	2021-07-14 14:06	1	0
ingest_cold_start	0:00:00	Hummingbirds	Hummingbirds	2021-07-14 20:00	1	0
ingest_programme_metadata	0:00:00	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0
ingest_user_activity	0:00:00	Hummingbirds	Hummingbirds	2021-07-05 08:00	1	0
precompute_recommendations_xanitas_comedy_featured_1_1	None	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0
precompute_recommendations_xanitas_drama_featured_1_1	None	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0
precompute_recommendations_xanitas_gripping_stories_1_1	None	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0
precompute_recommendations_xanitas_music_mixes_1_1	None	Hummingbirds	Hummingbirds	2021-07-11 16:00	1	0
precompute_recommendations_xanitas_recommended_for_you_1_1	None	Hummingbirds	Hummingbirds	2021-07-11 16:00	1	0
precompute_recommendations_xanitas_thought_provoking_1_1	None	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0
train_model	None	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0

Observation: 4 As, 4 S, 4 E, 4 C, 4 R

Airflow Dags Data Profiling Browse Admin Docs About

database-sounds-prod-virt 2021-07-14 23:46:10 UTC

DAGs

Search:

DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
airflow_monitoring	None	airflow	Hummingbirds	2021-07-14 23:43	1	0
backfill_user_activity	1 day, 00:00:00	Hummingbirds	Hummingbirds	2021-07-08 08:00	1	0
ingest_availability	hourly	Hummingbirds	Hummingbirds	2021-07-14 14:00	1	0
ingest_cold_start	0:00:00	Hummingbirds	Hummingbirds	2021-07-14 20:00	1	0
ingest_programme_metadata	0:00:00	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0
ingest_user_activity	0:00:00	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0
precompute_recommendations_xanitas_comedy_featured_1_1	None	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0
precompute_recommendations_xanitas_drama_featured_1_1	None	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0
precompute_recommendations_xanitas_music_mixes_1_1	None	Hummingbirds	Hummingbirds	2021-07-12 00:00	1	0
precompute_recommendations_xanitas_recommended_for_you_1_1	None	Hummingbirds	Hummingbirds	2021-07-11 16:00	1	0
precompute_recommendations_xanitas_thought_provoking_1_1	None	Hummingbirds	Hummingbirds	2021-07-13 00:00	1	0
train_model	None	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0

Airflow Dags Data Profiling Browse Admin Docs About

database-sounds-dev-orange 2021-07-14 23:47:43 UTC

DAGs

Search:

DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
airflow_monitoring	None	airflow	Hummingbirds	2021-07-14 23:41	1	0
backfill_user_activity	1 day, 00:00:00	Hummingbirds	Hummingbirds	2021-05-10 15:00	1	0
ingest_availability	hourly	Hummingbirds	Hummingbirds	2021-04-13 09:00	1	0
ingest_cold_start	0:00:00	Hummingbirds	Hummingbirds	2021-07-14 14:38	1	0
ingest_programme_metadata	0:00:00	Hummingbirds	Hummingbirds	2021-07-14 20:00	1	0
ingest_user_activity	0:00:00	Hummingbirds	Hummingbirds	2021-07-12 08:00	1	0
precompute_recommendations_xanitas_comedy_featured_1_1	None	Hummingbirds	Hummingbirds	2021-07-12 00:00	1	0
precompute_recommendations_xanitas_music_mixes_1_1	None	Hummingbirds	Hummingbirds	2021-07-12 00:00	1	0
precompute_recommendations_xanitas_recommended_for_you_1_1	None	Hummingbirds	Hummingbirds	2021-07-07 16:00	1	0
precompute_recommendations_xanitas_recommended_for_you_1_3	None	Hummingbirds	Hummingbirds	2021-07-13 00:00	1	0
train_model	None	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0

Airflow Dags Data Profiling Browse Admin Docs About

database-sounds-int-bleu 2021-07-14 23:47:06 UTC

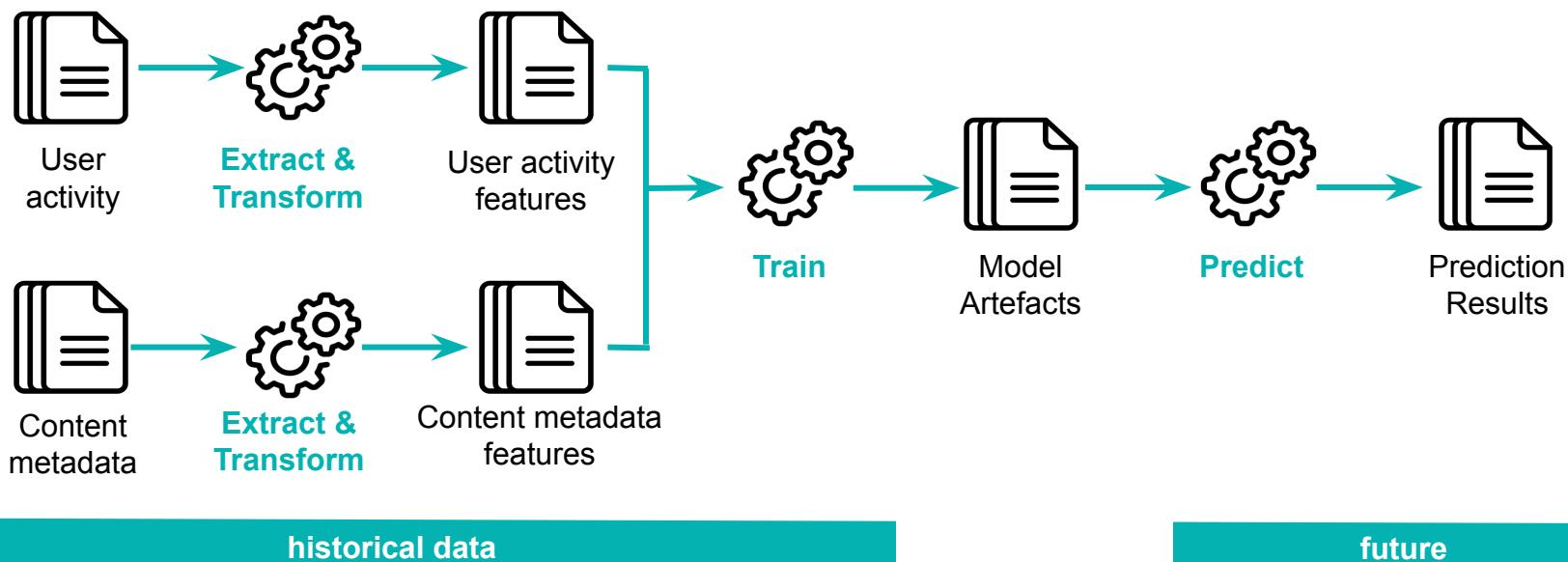
DAGs

Search:

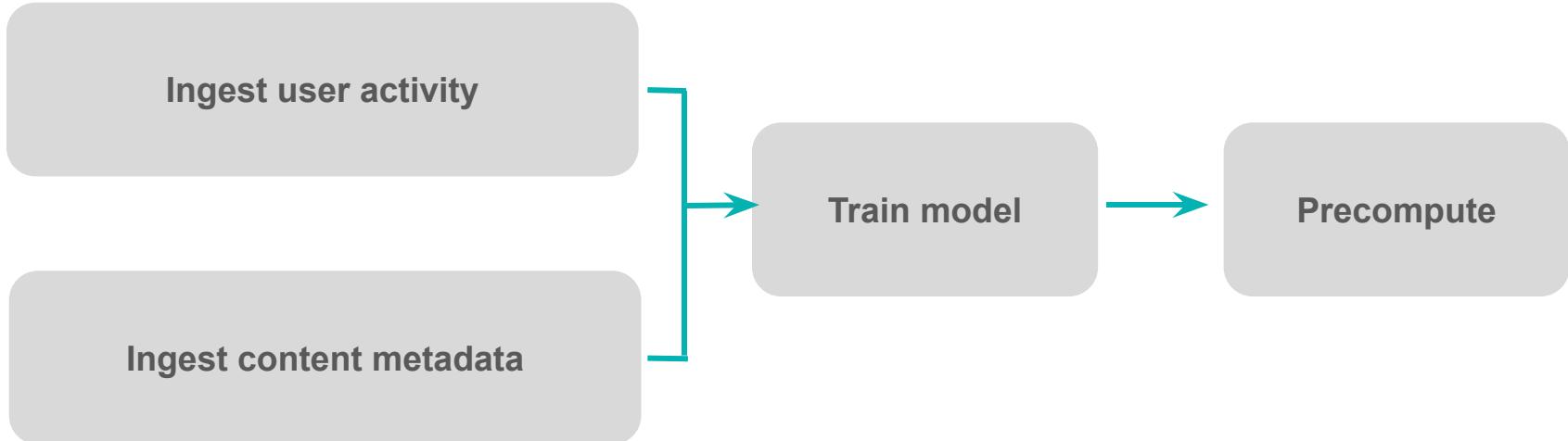
DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
airflow_monitoring	None	airflow	Hummingbirds	2021-07-14 23:40	1	0
backfill_user_activity	1 day, 00:00:00	Hummingbirds	Hummingbirds	2021-07-13 00:00	1	0
ingest_availability	hourly	Hummingbirds	Hummingbirds	2021-07-14 13:00	1	0
ingest_cold_start	0:00:00	Hummingbirds	Hummingbirds	2021-07-14 20:00	1	0
ingest_programme_metadata	0:00:00	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0
ingest_user_activity	0:00:00	Hummingbirds	Hummingbirds	2021-07-12 08:00	1	0
precompute_recommendations_xanitas_comedy_featured_1_1	None	Hummingbirds	Hummingbirds	2021-07-12 00:00	1	0
precompute_recommendations_xanitas_drama_featured_1_1	None	Hummingbirds	Hummingbirds	2021-07-12 00:00	1	0
precompute_recommendations_xanitas_music_mixes_1_1	None	Hummingbirds	Hummingbirds	2021-07-11 16:00	1	0
precompute_recommendations_xanitas_recommended_for_you_1_1	None	Hummingbirds	Hummingbirds	2021-07-13 00:00	1	0
precompute_recommendations_xanitas_thought_provoking_1_1	None	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0
train_model	None	Hummingbirds	Hummingbirds	2021-07-14 08:00	1	0

How we use Airflow **application**

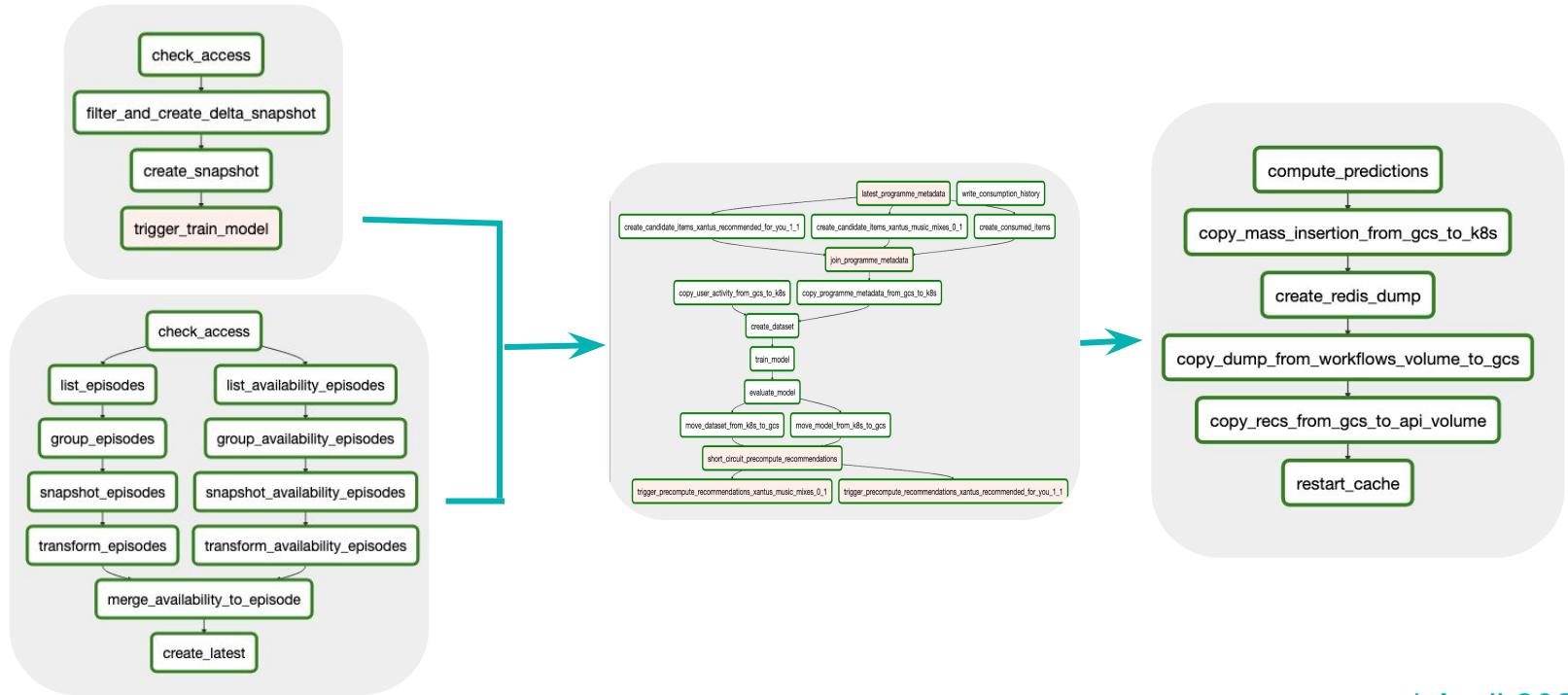
- Data integration: **ETL** (extract, transform, load)
- **Machine learning**: training and precomputation



How we use Airflow **application**



How we use Airflow application



* April 2021

Quiz time which is our most stable DAG?

- A. Ingest & transform Content Metadata
- B. Ingest & transform User Activity
- C. Train model
- D. Precompute recommendations



Quiz time which is our most stable DAG? tips

A. Ingest & transform Content Metadata **Python Operator**

- ~ 225k records
- Transforms ~12 GB => 57 MB

B. Ingest & transform User Activity **Dataflow Operator**

- ~ 3.5 million records
- Output: ~ 2 GB

C. Train model **Kubernetes Operator**

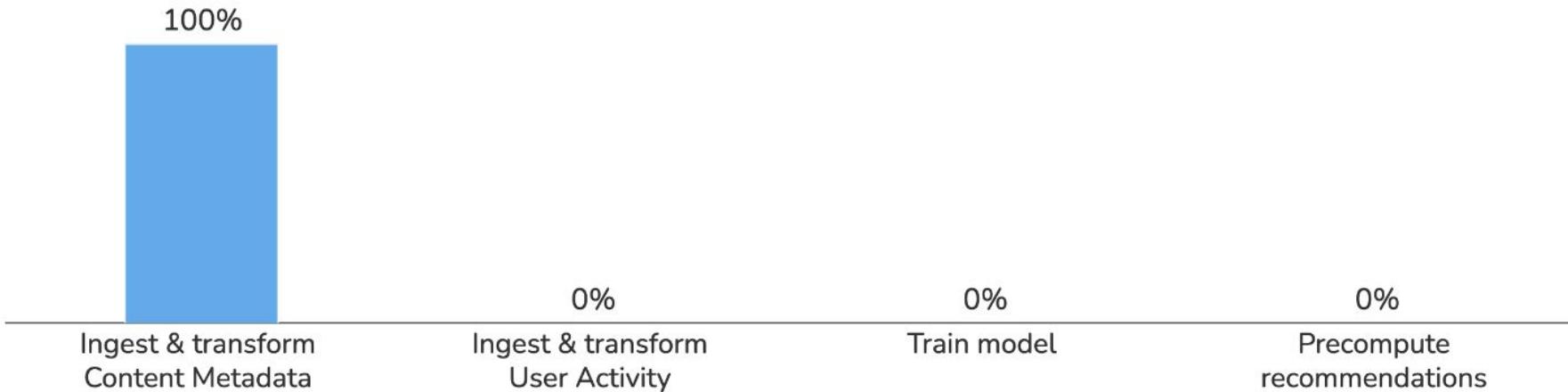
- Output: ~ 8 GB model & artefacts

D. Precompute recommendations **Dataflow Operator**

- ~ 3.5 million records
- Output: ~ 2.5 GB



Quiz time which is our most stable DAG? attendees

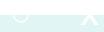


* responses from Airflow Summit 2021 participants, during the presentation

Quiz time which is our most stable DAG? **answer**

A. Ingest & transform Content Metadata

- 2 live incidents



B. Ingest & transform User Activity

- 1 live incident



C. Train model

- 2 live incidents

D. Precompute recommendations

- 2 live incidents

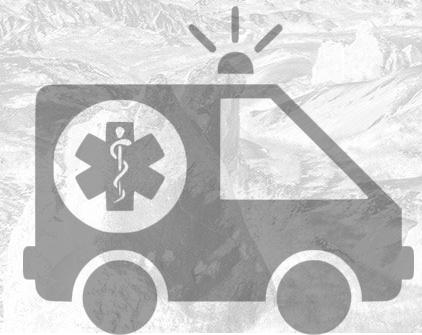
* from October 2020 until April 2021

Quiz time which is our most stable DAG? **details**

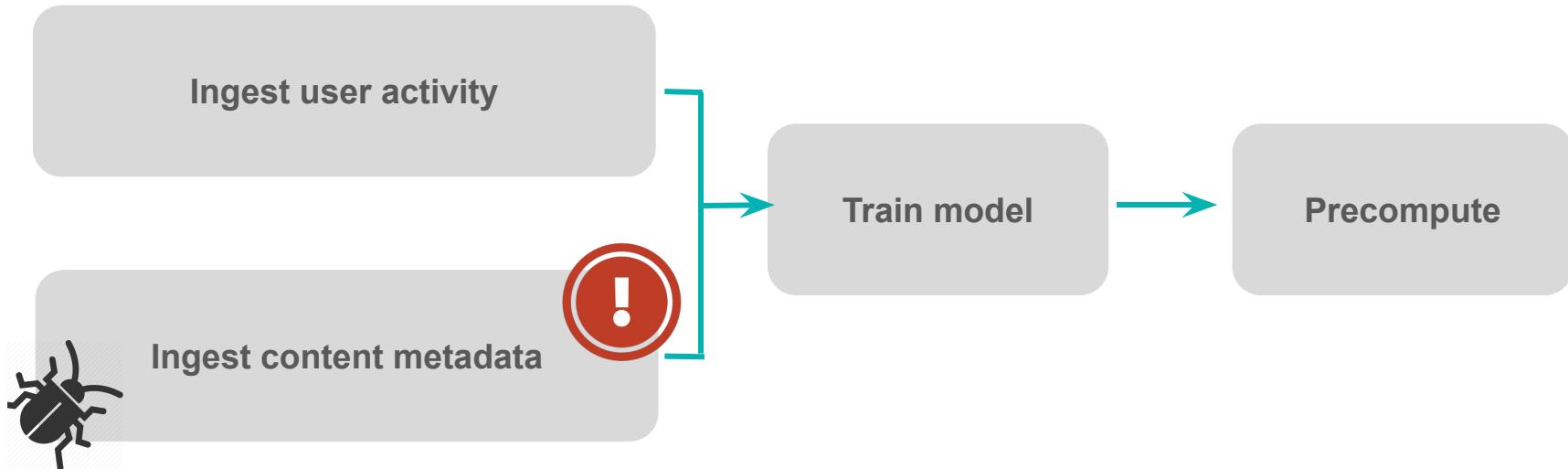
- A. Ingest & transform Content Metadata
 - Insufficient CPU
 - Spikes -> Timeouts (during higher volumes) / Continuing from where it stopped
- B. Ingest & transform User Activity
 - Idempotency issue
- C. Train model
 - K8s Pod reattach
 - Scheduling leading to two tasks running concurrently
- D. Precompute recommendations
 - Change to default job settings in Dataflow
 - GCS access limit
 - Non-stop Dataflow job

Removal of workflows obstructions

when things went wrong



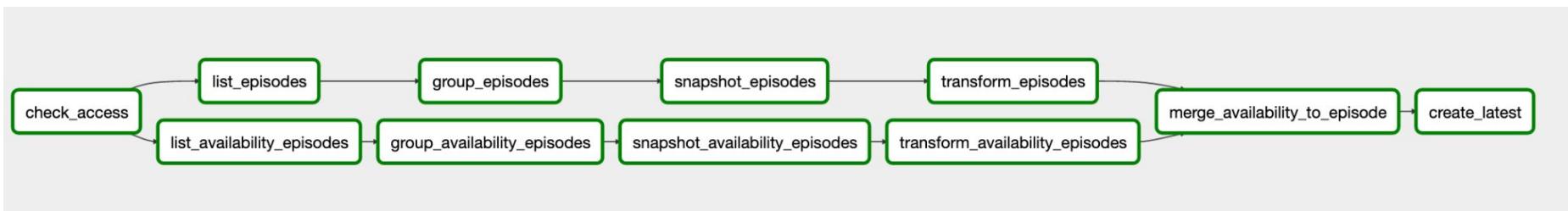
Obstruction 1 The programme metadata chronic issue



Obstruction 1 The programme metadata chronic issue

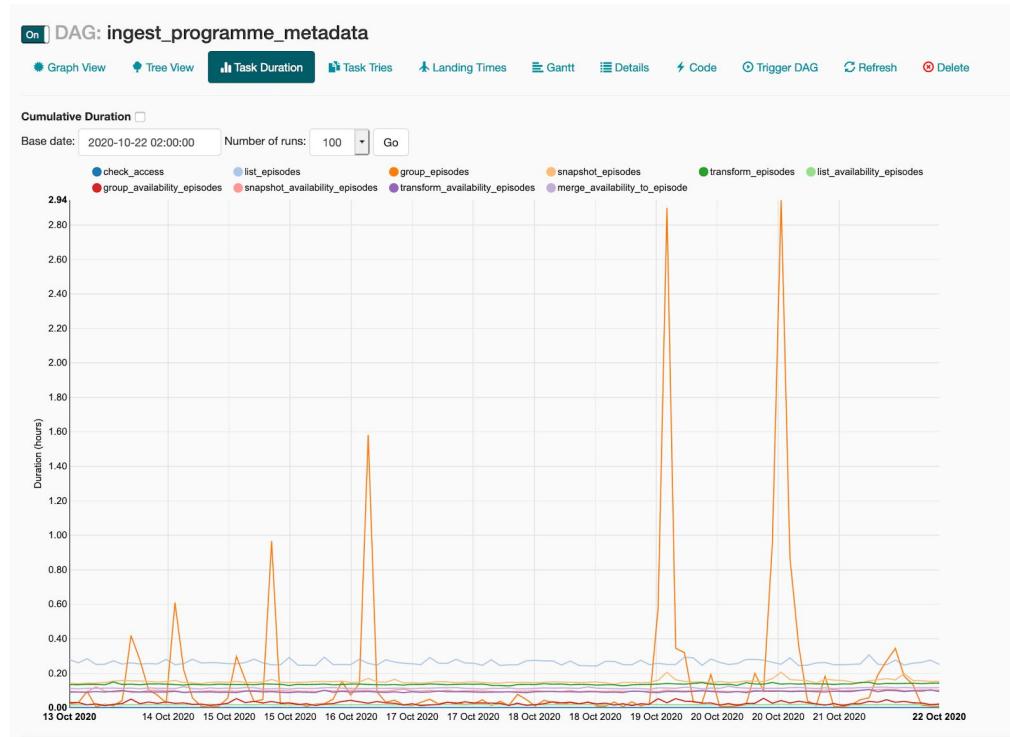
DAG's goals

- Import objects from AWS S3 (protected by STS) into Google Cloud Storage
- Requirements: between dozens and thousands KB-sized objects
- Filter and enrich the metadata
- Merge multiple streams of data and create an up-to-date snapshot

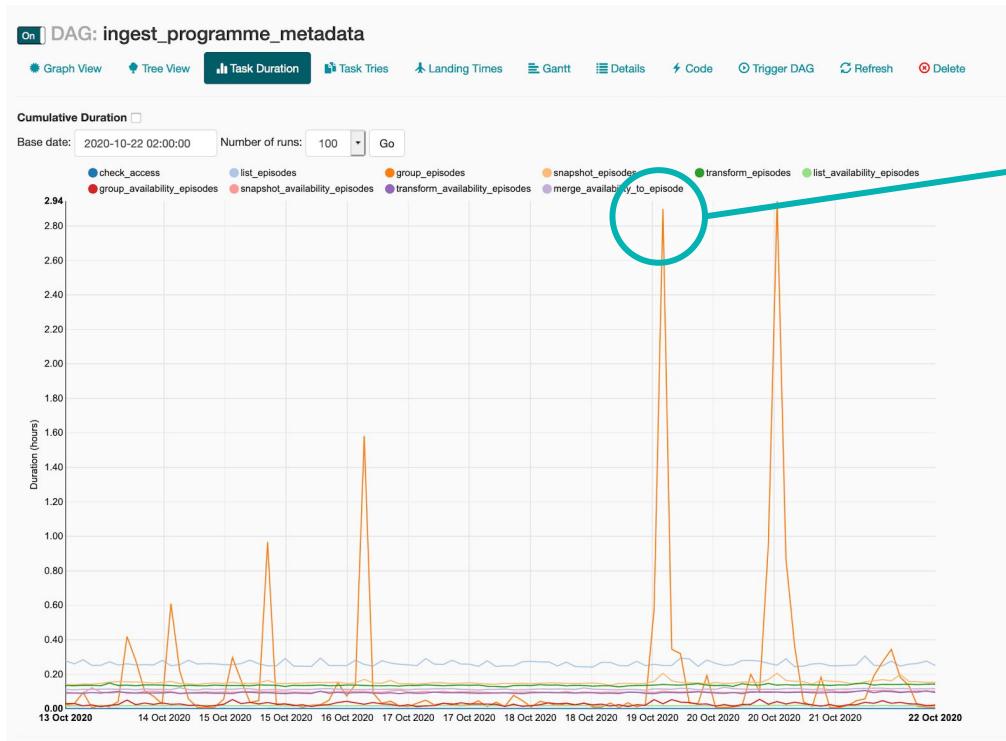


Mostly implemented using subclasses of the **Python Operator** class

Obstruction 1 The programme metadata chronic issue



Obstruction 1 The programme metadata chronic issue



Issue

Depending on the volumes of data, a single **PythonOperator** task which usually takes **10 min** could take almost **3h!**

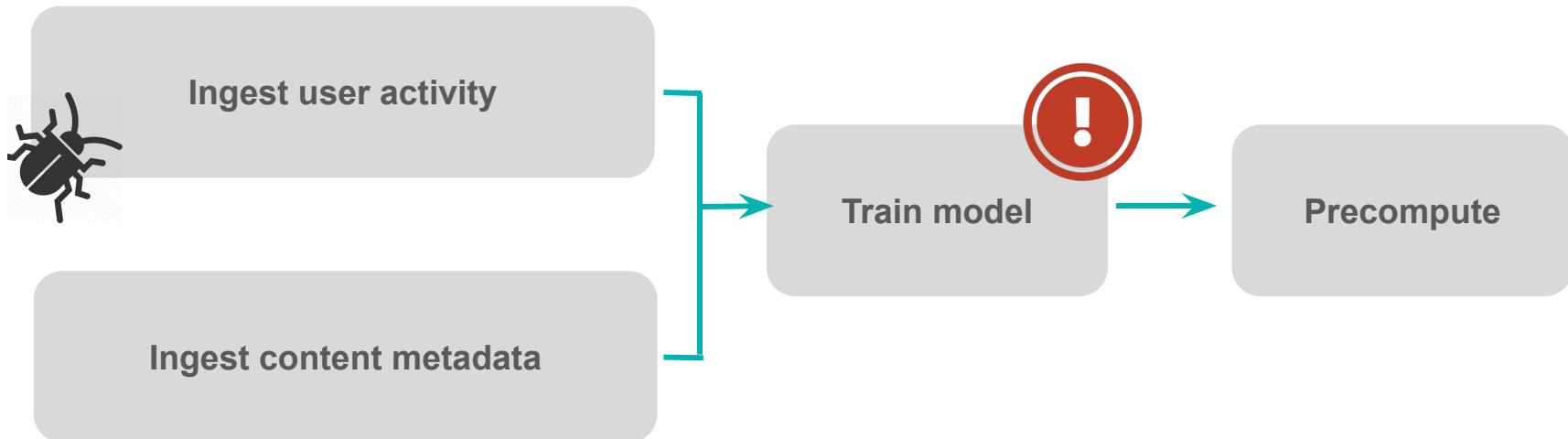
Consequences

Delay
Blocked Airflow executor

Solutions

Increase timeouts
Improve machine type
Delegate processing to another service

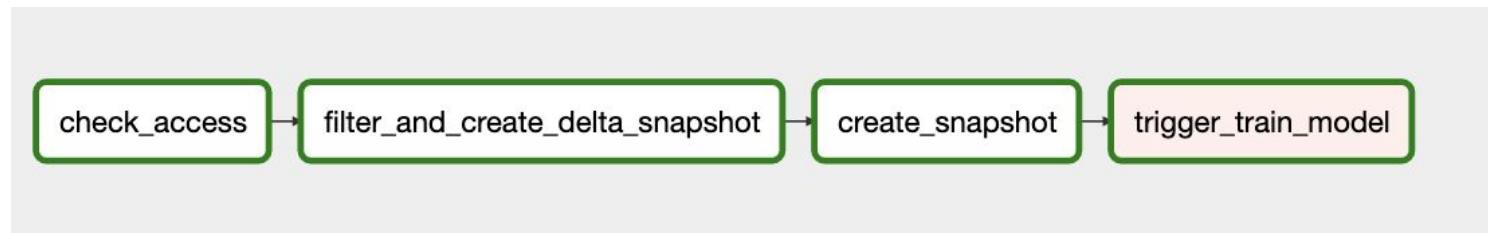
Obstruction 2 When the user activity workflow failed



Obstruction 2 When the user activity workflow failed

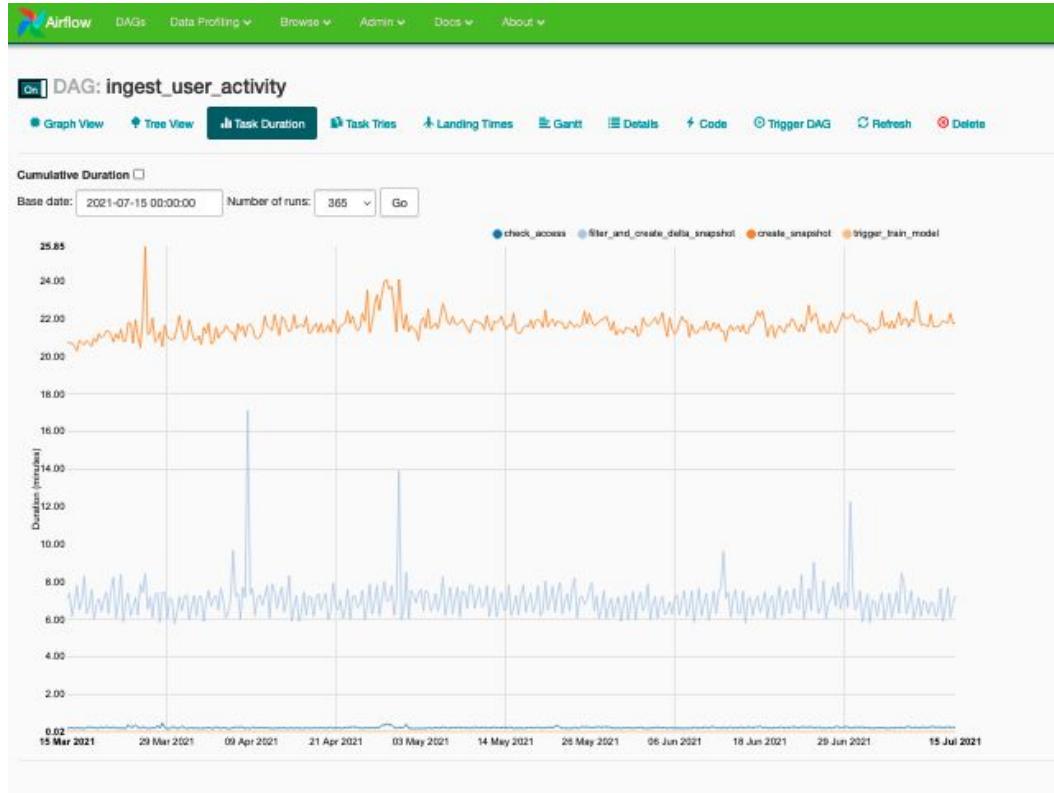
DAG's goals

- Read from user activity Parquet files in Google Cloud Storage
- Filter relevant activity and metadata
- Export a snapshot for the relevant interval of time
- Requirements: millions of records in MB-sized files

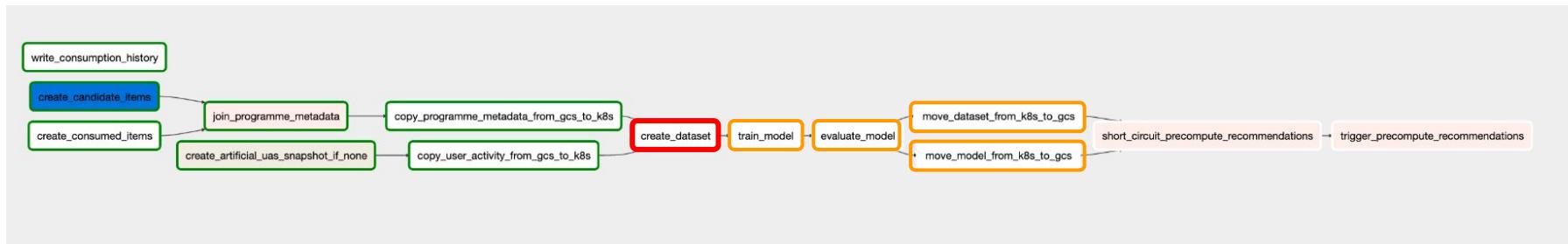
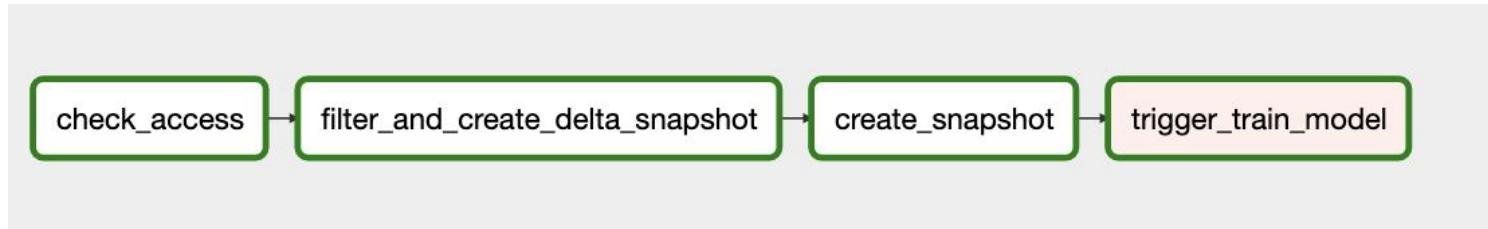


Mostly implemented using subclasses of the [Dataflow Operator](#) class

Obstruction 2 When the user activity workflow failed



Obstruction 2 When the user activity workflow failed



Obstruction 2 When the user activity workflow failed

Troubleshooting

- The volume of user activity meant to train the model had **doubled!**



Obstruction 2 When the user activity workflow failed

What happened

- Dataflow took longer than expected to run a job triggered by Airflow
- Airflow retried
- Both jobs completed successfully - and output the data in the same directory!
- The setup to train the model didn't expect to handle such spike in the volume of data and failed

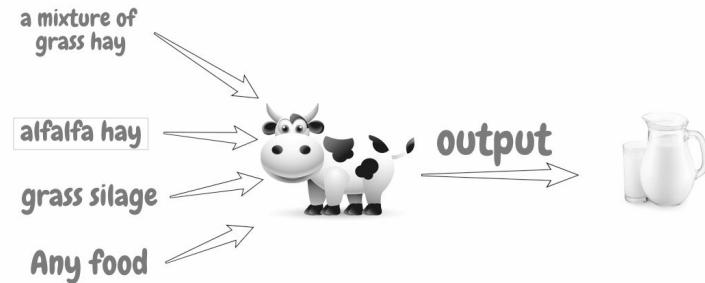
Obstruction 2 When the user activity workflow failed

What happened

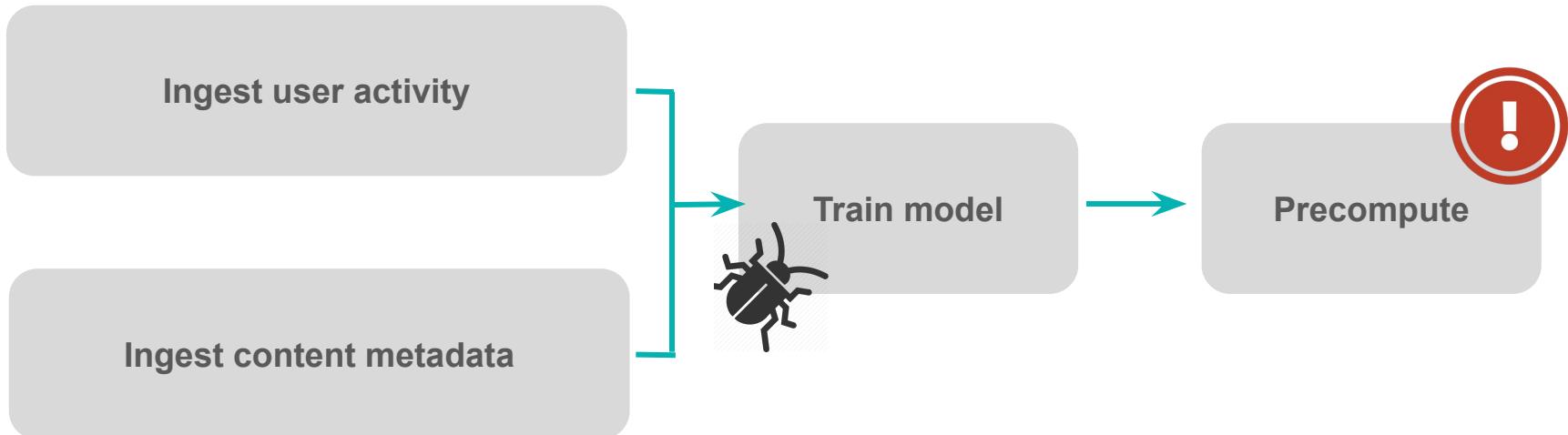
- Dataflow took longer than expected to run a job triggered by Airflow
- Airflow retried
- Both jobs completed successfully - and output the data in the same directory!
- The setup to train the model didn't expect to handle such spike in the volume of data and failed

Solution

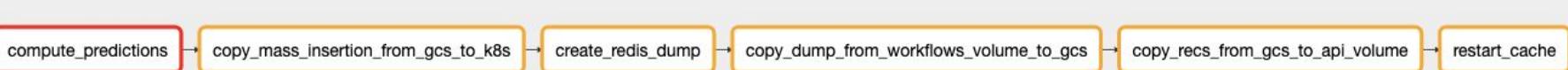
- Have idempotent tasks
- Clear the target path before processing a task



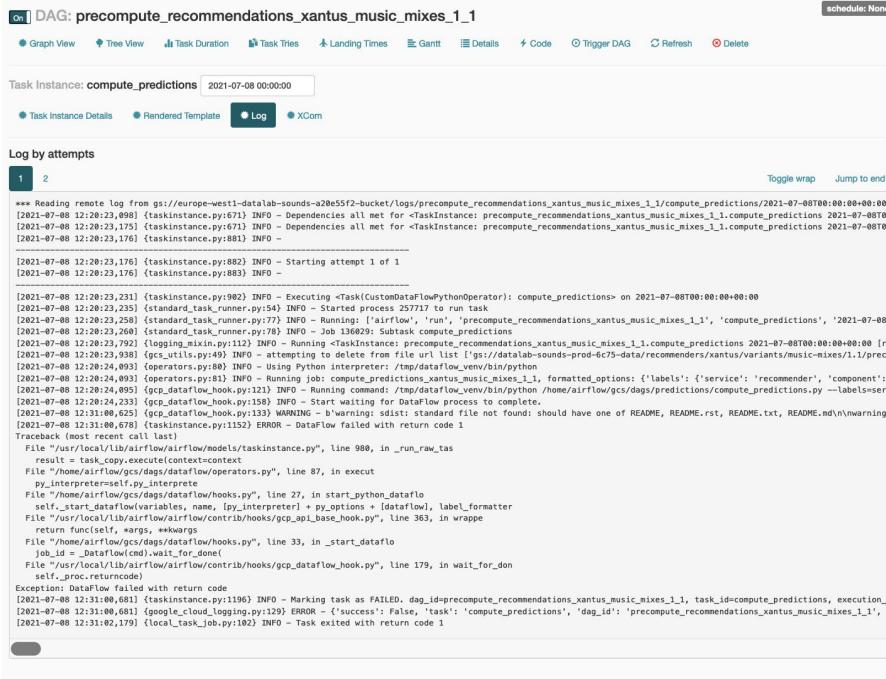
Obstruction 3 When precompute failed due to training



Obstruction 3 When precompute failed due to training

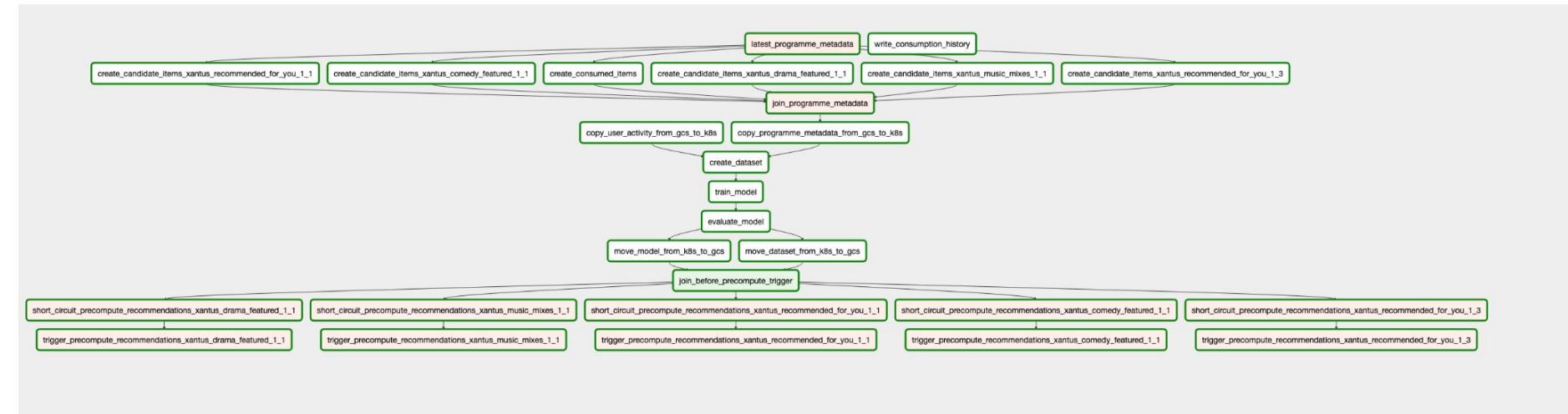


Obstruction 3 When precompute failed due to training

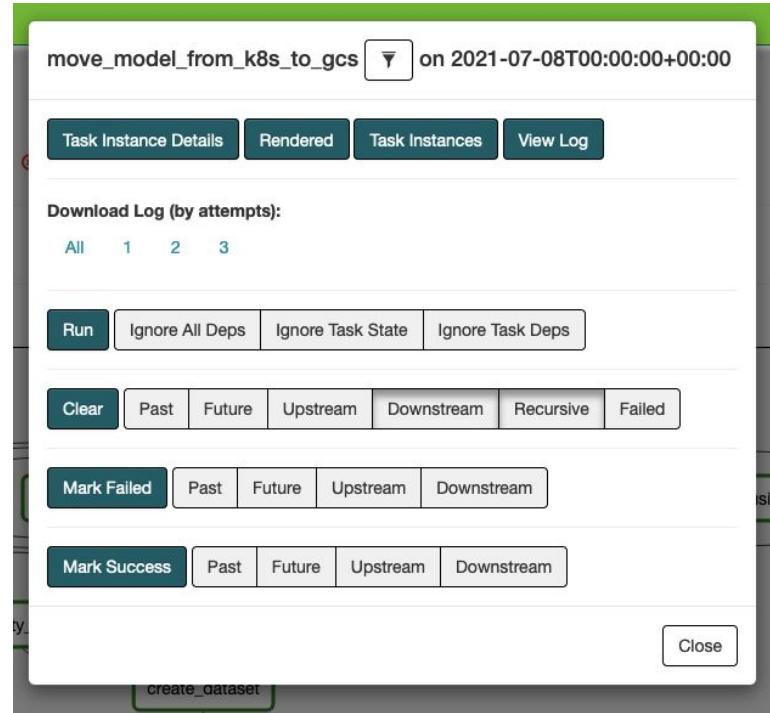


```
"message": "No such object:  
datalab-sounds-prod-6c75-data/recommenders/xan-  
tus/model1/2021-07-08T00:00:00+00:00/xantus.p  
kl"
```

Obstruction 3 When precompute failed due to training



Obstruction 3 When precompute failed due to training



Obstruction 3 When precompute failed due to training

```
[2021-07-08 12:15:55,278] {logging_mixin.py:112}
```

```
INFO - Running <TaskInstance:
```

```
train_model.move_model_from_k8s_to_gcs
```

```
2021-07-08T00:00:00+00:00 [running] on host  
airflow-worker-867b96c854-jzqw7
```

```
[2021-07-08 12:15:55,422]
```

```
{gcp_container_operator.py:299} INFO - Using gcloud  
with application default credentials.
```

```
[2021-07-08 12:15:57,286] {pod_launcher.py:173}
```

```
INFO - Event:
```

```
move-model-to-gcs-3deac821b57047619c1c9505ddc  
5db18 had an event of type Pending
```

```
[2021-07-08 12:15:57,286] {pod_launcher.py:139}
```

```
WARNING - Pod not yet started:
```

```
move-model-to-gcs-3deac821b57047619c1c9505ddc  
5db18
```

```
[2021-07-08 12:17:57,499] {taskinstance.py:1152}
```

```
ERROR - Pod Launching failed: Pod Launching failed:  
Pod took too long to start
```

```
[2021-07-08 12:18:59,584] {pod_launcher.py:156} INFO
```

```
- b'gsutil -m rm
```

```
gs://datalab-sounds-prod-6c75-data/recommenders/xantus/model/2021-07-08T00:00:00+xantus.pkl ||  
true\n'
```

```
[2021-07-08 12:18:59,911] {pod_launcher.py:156} INFO
```

```
- b'gsutil -m mv
```

```
/data/recommenders/xantus/model/2021-07-08T00:00:00+00:00/xantus.pkl (...)
```

```
[2021-07-08 12:19:35,295] {pod_launcher.py:156} INFO
```

```
- b'Operation completed over 1 objects/4.7 GiB.
```

```
\n'
```

```
[2021-07-08 12:19:35,536] {pod_launcher.py:156} INFO
```

```
- b'rm -rf
```

```
/data/recommenders/xantus/model/2021-07-08T00:00:00+00:00/xantus.pkl\n'
```

```
[2021-07-08 12:19:36,687] {taskinstance.py:1071} INFO
```

```
- Marking task as SUCCESS.dag_id=train_model,
```

Obstruction 3 When precompute failed due to training

```
$ kubectl logs move-model-to-gcs-a0b5193a42e040aaa37b3ad82953ee29 -n xantus-training

gsutil -m rm gs://datalab-sounds-prod-6c75-data/recommenders/xantus/model/2021-07-08T00:00:00+00:00/xantus.pkl || true
CommandException: 1 files/objects could not be removed.

gsutil -m mv /data/recommenders/xantus/model/2021-07-08T00:00:00+00:00/xantus.pkl
gs://datalab-sounds-prod-6c75-data/recommenders/xantus/model/2021-07-08T00:00:00+00:00/xantus.pkl

Copying file:///data/recommenders/xantus/model/2021-07-08T00:00:00+00:00/xantus.pkl [Content-Type=application/octet-stream]...

Removing file:///data/recommenders/xantus/model/2021-07-08T00:00:00+00:00/xantus.pkl...
| [1/1 files][ 4.7 GiB/ 4.7 GiB] 100% Done 111.3 MiB/s ETA 00:00:00
Operation completed over 1 objects/4.7 GiB.

rm -rf /data/recommenders/xantus/model/2021-07-08T00:00:00+00:00/xantus.pkl
```

Obstruction 3 When precompute failed due to training

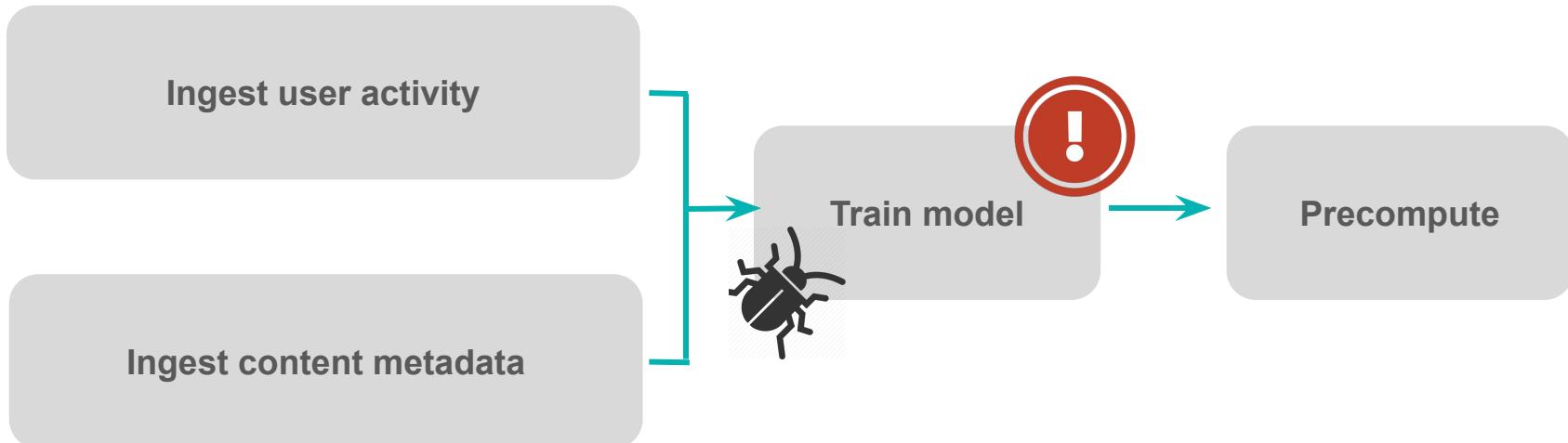
What happened

- The GKE node pool, where jobs were executed, was struggling
- Airflow Kubernetes Operator timed out after a long time waiting (Kubernetes didn't know)
- A new task retry was triggered
- Both pods run concurrently and due to how we implemented idempotency, the data was deleted - but the last task retry was successful

Solution

- Use newer version of the KubernetesPodOperator
- Confirm by the end of the task that the desired artefact exists

Obstruction 4 Intermittent DAG after an Airflow upgrade

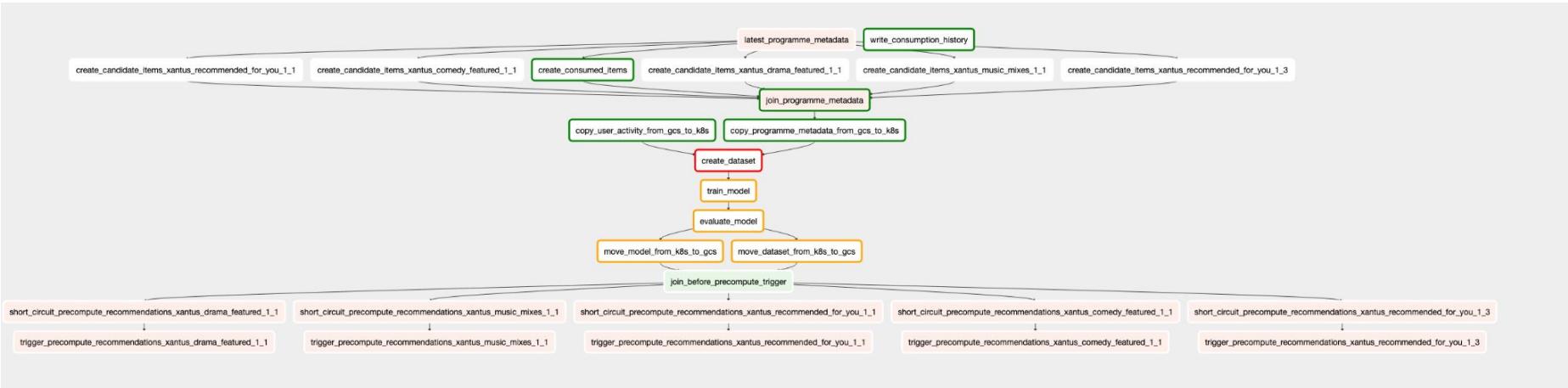


Obstruction 4 Intermittent DAG after an Airflow upgrade

What happened

- After upgrading from Airflow 1.10.4 to 1.10.12 some KubernetesPodOperator tasks became intermittent
- Legit running pods failed
- The logs seemed to show that new jobs were trying to reattach to previously existing Pods and failed

Obstruction 4 Intermittent DAG after an Airflow upgrade



Obstruction 4 Intermittent DAG after an Airflow upgrade

The screenshot shows the Airflow UI for a task instance named 'create_dataset'. The task was run on 2021-01-30T00:00:00+00:00. The page includes tabs for 'Task Instance Details', 'Rendered', 'Task Instances', and 'View Log'. Below these are buttons for 'Run', 'Ignore All Deps', 'Ignore Task State', and 'Ignore Task Deps'. Further down are buttons for 'Clear', 'Past', 'Future', 'Upstream', 'Downstream', 'Recursive', and 'Failed'. At the bottom are buttons for 'Mark Failed', 'Past', 'Future', 'Upstream', and 'Downstream', followed by 'Mark Success', 'Past', 'Future', 'Upstream', and 'Downstream'. A 'Close' button is at the bottom right. The URL in the browser's address bar is 'http://127.0.0.1:8080/admin/airflow/instances?task_id=create_dataset&run_id=1&_flt0=1'.

HTTP response headers: HTTPHeaderDict({'Audit-Id': '133bc1f2-388b-490c-bdc6-34053685d5ee', 'Content-Type': 'application/json', 'Date': 'Sat, 30 Jan 2021 09:11:33 GMT', 'Content-Length': '231'})

HTTP response body:

b'{"kind":"Status","apiVersion":"v1","metadata":{},"status":"Failure","message":"container \\\"base\\\" in pod \\\"create-dataset-17a3b6f132e44544a836550be367c670\\\" is waiting to start:
ContainerCreating","reason":"BadRequest","code":400}\n

(...)

"/opt/python3.6/lib/python3.6/site-packages/kubernetes/client/rest.py", line 231, in reques

```
    raise ApiException(http_resp=r
kubernetes.client.rest.ApiException: (400
Reason: Bad Request
```

Obstruction 4 Intermittent DAG after an Airflow upgrade

Solution

`airflow.contrib.operators.kubernetes_pod_operator`

Executes task in a Kubernetes POD

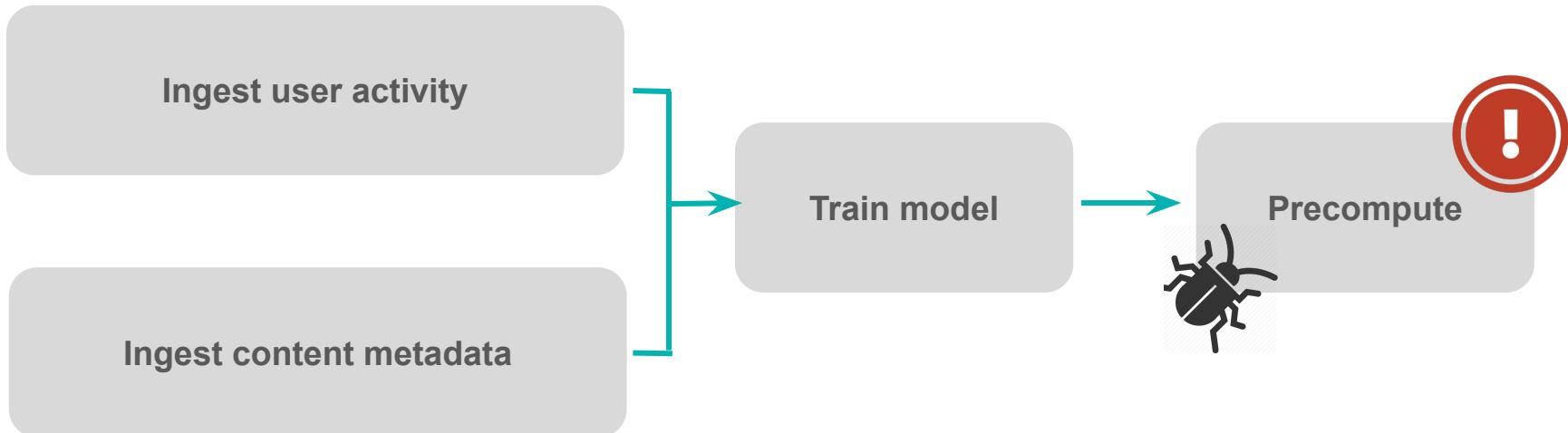
Module Contents

```
class airflow.contrib.operators.kubernetes_pod_operator.KubernetesPodOperator(namespace=None, image=None, name=None,
cmds=None, arguments=None, ports=None, volume_mounts=None, volumes=None, env_vars=None, secrets=None, in_cluster=None,
cluster_context=None, labels=None, reattach_on_restart=True, startup_timeout_seconds=120, get_logs=True,
image_pull_policy='IfNotPresent', annotations=None, resources=None, affinity=None, config_file=None, node_selectors=None,
image_pull_secrets=None, service_account_name='default', is_delete_operator_pod=False, hostnetwork=False,
tolerations=None, configmaps=None, security_context=None, pod_runtime_info_envs=None, dnspolicy=None, schedulername=None,
full_pod_spec=None, init_containers=None, log_events_on_failure=False, do_xcom_push=False, pod_template_file=None,
priority_class_name=None, *args, **kwargs) [source]
```

Bases: `airflow.models.BaseOperator`

https://airflow.apache.org/docs/apache-airflow/1.10.12/_api/airflow/contrib/operators/kubernetes_pod_operator/index.html

Obstruction 5 When the Dataflow job said no



Obstruction 5 When the Dataflow job said no

On DAG: precompute_recommendations

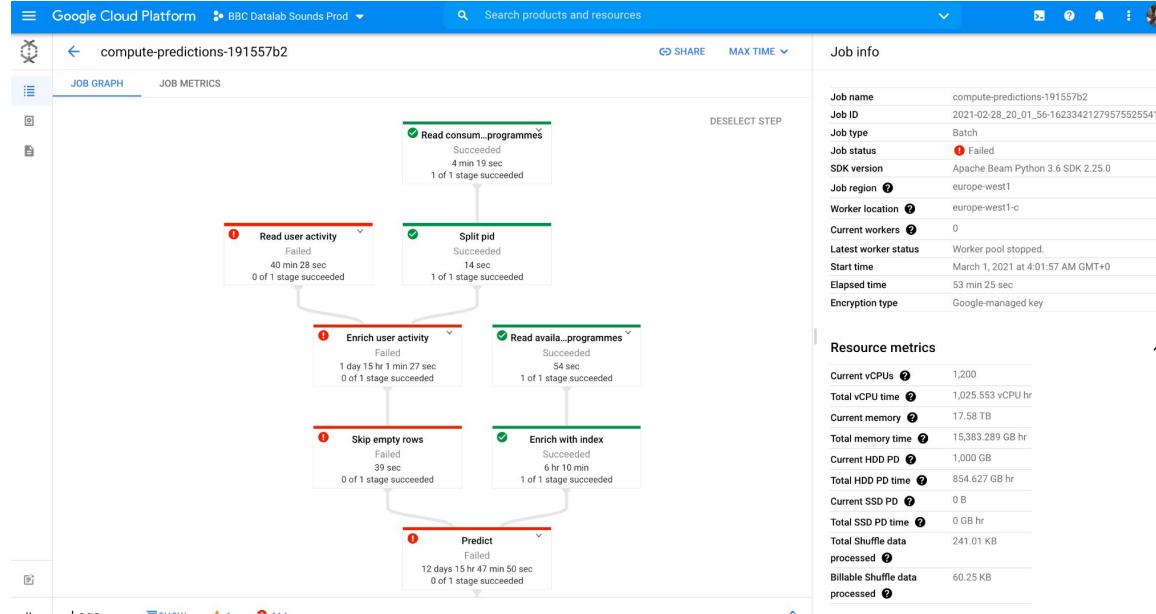
Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Code Trigger DAG Refresh Delete

failed Base date: 2021-02-28 16:00:01 Number of runs: 25 Run: trig_2021-02-28T16:00:00+00:00 Layout: Left->Right Go

CustomDataFlowPythonOperator GKEPodOperator KubectlOperator

```
graph LR; A[compute_predictions] --> B[copy_mass_insertion_from_gcs_to_k8s]; B --> C[create_redis_dump]; C --> D[copy_dump_from_workflows_volume_to_gcs]; D --> E[copy_recs_from_gcs_to_api_volume]; E --> F[restart_cache]
```

Obstruction 5 When the Dataflow job said no



OSError: [Errno 28] No space left on device During handling

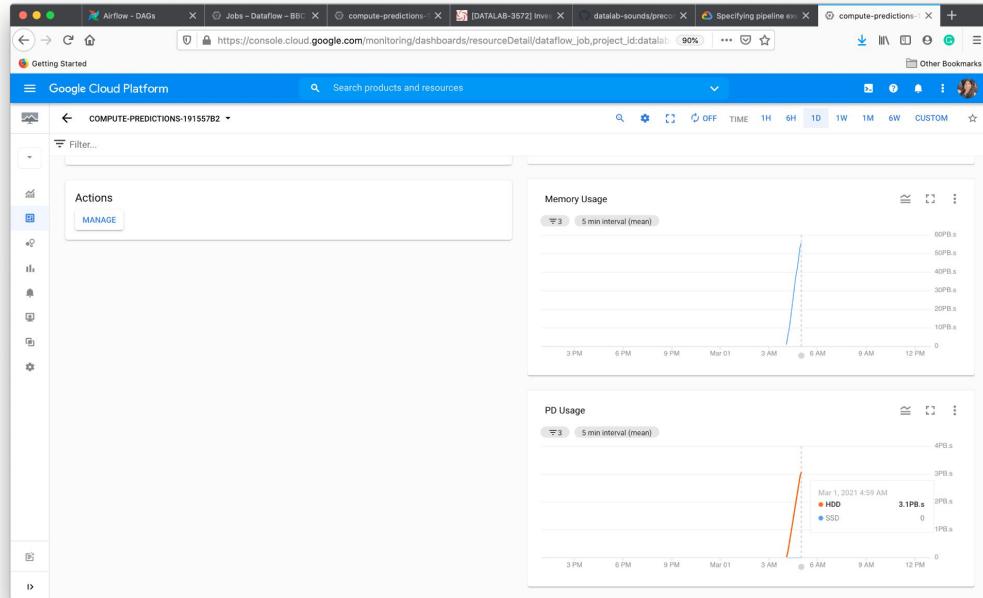
Obstruction 5 When the Dataflow job said no

	successful compute-predictions-21628eba 2021-02-24_20_06_12-17726452853028084617	unsuccessful compute-predictions-191557b2 2021-02-28_20_01_56-16233421279575525541
	25 February 2021	1st March 2021
consumption-history/	1.87 GiB	1.88 GiB
consumed-items/	55.63 MiB	55.68 MiB
candidate-items/rfy/	25.06 MiB	25.09 MiB
xantus.pkl	4.67 GiB	4.69 GiB
item_features.npz	1.73 MiB	1.73 MiB
mapping.json	473.84 MiB	476.01 MiB

If a batch job uses Dataflow Shuffle, then the default is 25 GB; otherwise, the default is 250 GB.

<https://cloud.google.com/dataflow/docs/guides/specifying-exec-params#python>

Obstruction 5 When the Dataflow job said no



Job info	
Job name	compute-predictions-21628eba
Job ID	2021-02-24_20_06_12-17726452853028084617
Job type	Batch
Job status	✓ Succeeded
SDK version	Apache Beam Python 3.6 SDK 2.25.0
Job region	europe-west1
Worker location	europe-west1-b
Current workers	0
Latest worker status	Worker pool stopped.
Start time	February 25, 2021 at 4:06:14 AM GMT+0
Elapsed time	2 hr 13 min
Encryption type	Google-managed key
Resource metrics	
Current vCPUs	1,200
Total vCPU time	2,633,094 vCPU hr
Current memory	17.58 TB
Total memory time	39,496,416 GB hr
Current HDD PD	9.77 TB
Total HDD PD time	21,942,453 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr
Labels	
airflow-version	v1-10-12-commonspark

Obstruction 5 When the Dataflow job said no

Google Cloud Platform BBC DataLab Sounds Prod Search products and resources

Case 27052933 REOPEN

Dataflow job started using the shuffle service unexpectedly

Tatiana Al-Chueyr
tatiana.alchueyr@bbc.co.uk
March 1, 2021 at 11:37:43 AM GMT+0

At the moment, our end-users are being offered stale recommendations in production. The reason for this is because a Dataflow job failed (ID: 2021-02-28_20_01_56-16233421279575525541).

It seems the Dataflow job run out of disk, since the the job logs contain the error:
...
RuntimeError: OSError: [Errno 28] No space left on device [while running 'Predict/Predict']
...

When comparing this unsuccessful (ID: 2021-02-28_20_01_56-16233421279575525541) run with the previous successful run (ID: 2021-02-24_20_06_12-17726452853028084617), we noticed the data shuffle service was *automatically* enabled in the failed job. We realised this by looking at the Job resource metrics and seeing the following unexpected metrics: "Total Shuffle data processed" and "Billable Shuffle data processed".

We were surprised to see the shuffling charges because we did not make any changes to the code itself - nor did we update the CSV or environment. We are aware that by default the shuffle service reduces the

Attributes

Project
datalab-sounds-prod-6c75

Priority
P2

Status
Closed

Category
Big Data

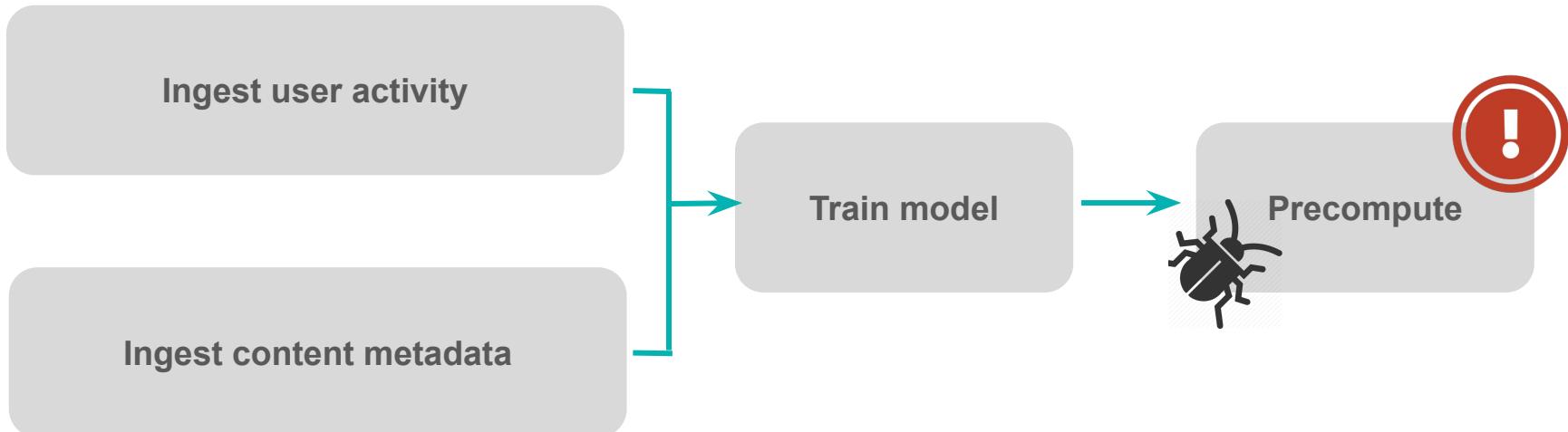
Component
Cloud Dataflow

CC

Solution

```
--experiments=shuffle_mode=appliance
```

Obstruction 6 The never ending Dataflow job



Obstruction 6 The never ending Dataflow job

datalab_devops_int - Jun 14th



google cloud monitoring APP 6:46 PM

Incident #0.m3oacbpbewe is ongoing

[Composer - datalab-sounds-int-bleu] Workflow failure

Workflow Runs for datalab-monitoring-int-f09d Cloud Composer Workflow labels

{project_id=datalab-monitoring-int-f09d, workflow_name=datalab-sounds-int-bleu.precompute_recommendations_xantus_recommended_for_you_1_1} is above the threshold of 0.000 ...



Obstruction 6 The never ending Dataflow job

```
[2021-05-13 11:52:34,884] {gcp_dataflow_hook.py:1 (...) Traceback (most recent call last):\n  File\n    "/home/airflow/gcs/dags/predictions/compute_predictions.py", line 337, in run\n        return pipeline\n  File\n    "/tmp/dataflow_venv/lib/python3.6/site-packages/apache_beam/pipeline.py", line 569, in __exit__\n        self.result.wait_until_finish()\n  File\n    "/tmp/dataflow_venv/lib/python3.6/site-packages/apache_beam/runners/dataflow/dataflow_runner.py",\n    line 1650, in wait_until_finish\n        self)\napache_beam.runners.dataflow.dataflow_runner.DataflowRuntimeException: Dataflow pipeline\nfailed. State: FAILED,\n(.. )
```

The job failed because a work item has failed 4 times. Look in previous log entries for the cause of each one of the 4 failures. For more information, see

<https://cloud.google.com/dataflow/docs/guides/common-errors>. The work item was attempted on these workers:

```
\n  compute-predictions-xantu-05130255-opno-harness-mtj1\n  Root cause: The worker lost\n  contact with the service.,\n  compute-predictions-xantu-05130255-opno-harness-2x4v\n  Root cause:\n  The worker lost contact with the service.,\n  compute-predictions-xantu-05130255-opno-harness-5gkd\n  Root cause: The worker lost contact with the service.,\n  compute-predictions-xantu-05130255-opno-harness-2t1n\n  Root cause: The worker lost contact with\n  the service.'
```

Obstruction 6 The never ending Dataflow job

Name	Type	End time	Elapsed time	Start time
compute-predictions-xantus-music-mixes-1-1-87d98f40	Batch	May 13, 2021, 10:44:25 AM	3 days 5 hr	May 10, 2021, 5:21:29 AM
compute-predictions-xantus-music-mixes-1-1-	Batch	May 13, 2021,	5 hr 12 min	May 13, 2021,

Obstruction 6 The never ending Dataflow job

Solution

- Backport the latest Google Cloud operators in Apache Airflow
- Particularly:
 - DataflowCreatePythonJobOperator
 - DataflowJobStatusSensor

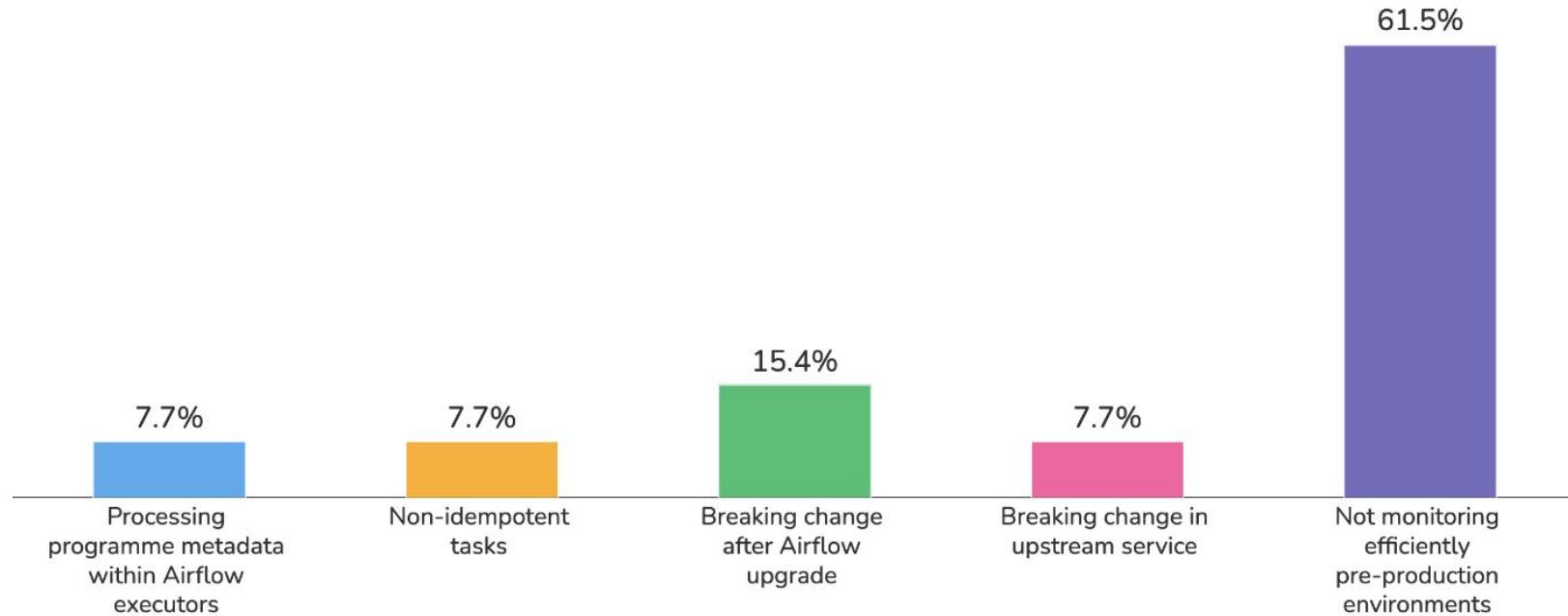
<https://medium.com/google-cloud/backporting-google-cloud-operators-in-apache-airflow-34b6c9efffc8>

Quiz time which do you reckon was the most costly issue?

- A. Processing programme metadata within Airflow executors
- B. Non-idempotent tasks
- C. Breaking change after Airflow upgrade
- D. Breaking change in upstream service
- E. Not monitoring efficiently pre-production environments



Quiz time which do you reckon was the most costly issue?



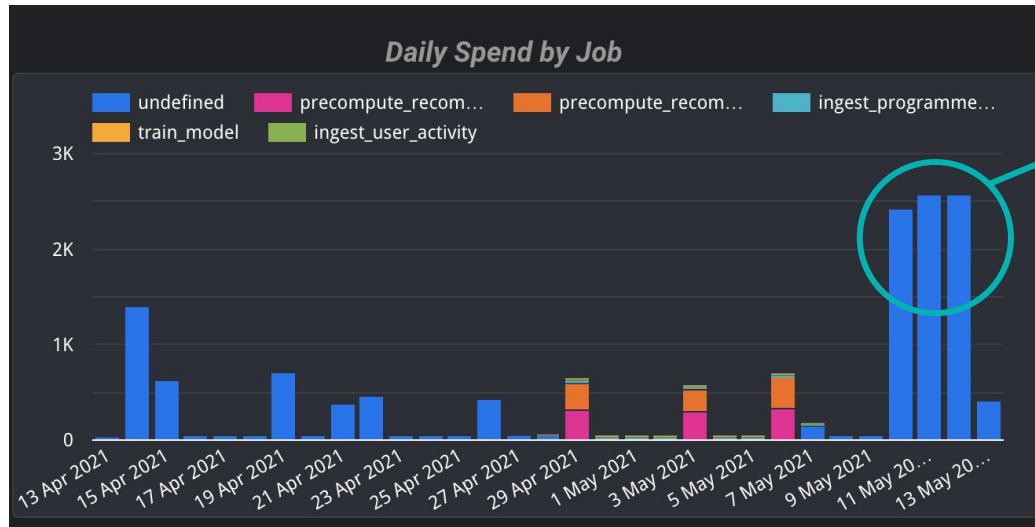
* responses from Airflow Summit 2021 participants, during the presentation

Quiz time which do you reckon was the most costly issue?

- A. Processing programme metadata within Airflow executors
- B. Non-idempotent tasks
- C. Breaking change after Airflow upgrade
- D. Breaking change in upstream service
- E. Not monitoring efficiently pre-production environments



Quiz time which do you reckon was the most costly issue?



The never ending Dataflow Job, triggered with **DataflowOperator** in our int environment, run for over 3 days and costed over £12k

Hygienic workflows throughout development

Hygienic pipelines

Smell 1 To plugin or not to plugin

- **Requirement:**
 - Have common packages across multiple DAGs without using PyPI or similar
- **Attempt:**
 - Use plugins to expose those
 - Deploy using
 - `gcloud composer environments storage dags import .`
 - `gcloud composer environments storage dags import .`
- **Problems**
 - Lots of broken deployments
 - Unynchronised upload of **plugins** and **DAGs** to **Composer** web server & workers
 - Issues in enabling [**DAG serialisation**](#) with **plugins**

Smell 1 To plugin or not to plugin

- **Solution:**
 - Stop using plugins
 - Use standard Python packages
 - Upload them using Google Cloud to the Bucket / path
 - `gsutil rm (...)`
 - `gsutil cp (...)`

Smell 2 Configuration (mis)patterns

- **Requirement:**
 - Have a strategy for handling environment-specific and common configuration
- **Attempt:**
 - To use Environment variables to declare each env-specific variable
 - To deploy using Terraform
 - To declare common configuration in the DAGs
- **Problems**
 - Each variant updated represented a Cloud Composer deployment
 - Redundant configuration definition across DAGs and multiple utilities modules
 - Hard to identify the data sources / targets paths

Smell 2 Configuration (mis)patterns

- **Solution:** have configuration files loaded into Airflow variables with paths and variables

```
70 "precompute_recommendations_dag": {
71     "start_date": "2020-12-07T06:00:00.00",
72     "redis_image": "marketplace.gcr.io/google/redis5",
73     "dataflow": {
74         "num_workers": "40",
75         "notused_comments": "30 vCPU with 15 GB RAM per vCPU",
76         "machine_type": "custom-30-460800-ext"
77     },
78     "paths": {
79         "candidate_items_gcs_prefix": "%(gcs_base_path)s/programme-metadata/candidate-items/{{ ts }}/",
80         "workflows_redis_mass_path": "%(gcs_base_path)s/precomputed-recommendations/redis-mass-insertion/{{ ts }}/",
81         "workflows_redis_dump_path": "%(gcs_base_path)s/precomputed-recommendations/redis-dump/{{ ts }}/",
82         "api_redis_dump_path": "%(volume_base_path)s/precomputed_recommendations/",
83         "precompute_recommendations_dag_id": "precompute_recommendations_%(identifier)s"
84     }
85 },
86 "ingest_user_activity_dag": {
87     "start_date": "2020-12-07T06:00:00.00",
88     "ingest_interval_hours": "8",
89     "snapshot_interval_days": "120",
90     "max_active_dag_runs": "1",
91     "ingest_catchup": "1",
92     "user_activity_bucket": "datalab-sounds-data-dev-ff72_uas",
93     "user_activity_prefix": "dev_test_key",
94     "paths": {
95         "deltas_prefix": "user-activity/uas/deltas/",
96         "delta_snapshot_path": "user-activity/uas/deltas/{{ ts }}/{{formatted_ingest_interval}}s/",
97         "list_original_paths": "workflows/user-activity-to-process/original/{{ ts }}/{{formatted_ingest_interval}}s/",
98         "user_activity_snapshots_path": "user-activity/uas/snapshots/{{ ts }}/{{formatted_snapshot_interval}}s/",
99         "list_deltas_path": "workflows/user-activity-to-process/deltas/{{ ts }}/{{formatted_snapshot_interval}}s/"
100    }
101},
```

<https://www.astronomer.io/guides/dynamically-generating-dags>

Smell 2 Configuration (mis)patterns

Screenshot of the Apache Airflow web interface showing a DAG named "precompute_recommendations_xantus_music_mixes_1_1".

The interface includes a navigation bar with links like "DAGs", "Data Profiling", "Browse", "Admin", "Docs", and "About". The URL is https://p83f47623eb9929c1p-tp.appspot.com/admin/airflow/tree?dag_id=precompute_recommendations_xantus_music_mixes_1_1. The timestamp is 2021-06-03 20:15:16 UTC.

The DAG tree view shows the following structure:

- [DAG] **precompute_recommendations_xantus_music_mixes_1_1**
 - compute_predictions
 - copy_mass_insertion_from_gcs_to_k8s
 - create_redis_dump
 - copy_dump_from_workflows_volume_to_gcs
 - copy_recs_from_gcs_to_api_volume
 - restart_cache

Below the tree view is a Gantt chart showing task execution status from May to June. The tasks are color-coded according to their status:

 - scheduled (grey)
 - skipped (light blue)
 - upstream_failed (orange)
 - up_for_reschedule (cyan)
 - up_for_retry (yellow)
 - failed (red)
 - success (green)
 - running (dark green)
 - queued (light grey)
 - no_status (white)

The Gantt chart shows several tasks in various states across the timeline.

Takeaways **Avoiding live incidents**



- Keep processing **out of Airflow executors**
- Idempotency** matters - and it can be hard!
- Backporting** is better than sticking to the past
- Reviewing **release notes** can help avoid live incidents
- Monitoring pre-production environments** can **save money**

Takeaways **Keeping the house clean**



- Avoid plugins
- A **delete-deploy** approach can avoid problems
- Early configuration-driven approach saves time



Much more than **obstructions**

With the help of **Apache Airflow**, Datalab:

- Was able to end a contract of the BBC, with an **external recommendation service**, by increasing in **59% the audience engagement**
- Serves **daily millions** of **personalised recommendations** to the **BBC audiences**
- Built a **configurable** Machine Learning pipeline **agnostic of the model**
 - Constantly adds **new variants** and **extends workflows**



Thank you!

Tatiana Al-Chueyr

@tati_alchueyr

Airflow
Summit 2021