Clearing Airflow obstructions

@tati_alchueyr
Principal Data Engineer @ BBC Datalab
Heads up!

- During this **session** there will be some **quizzes**

- Be prepared to either:
  - **Scan** the **QR code**
  - **Access** using the **URL**
- **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
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
- 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
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- **Outdated** version 1.10.12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
How we use Airflow **application**

- **Data integration:** ETL (extract, transform, load)
- **Machine learning:** training and precomputation
How we use Airflow **application**

- Ingest user activity
- Ingest content metadata
- Train model
- Precompute
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
A. Ingest & transform Content Metadata
   ○ ~ 225k records
   ○ Transforms ~12 GB => 57 MB

B. Ingest & transform User Activity
   ○ ~ 3.5 million records
   ○ Output: ~ 2 GB

C. Train model
   ○ Output: ~ 8 GB model & artefacts

D. Precompute recommendations
   ○ ~ 3.5 million records
   ○ Output: ~ 2.5 GB

Quiz time which is our most stable DAG? tips
Quiz time which is our most stable DAG?

* 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
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

Quiz time which is our most stable DAG? [details]
Removal of workflows obstructions when things went wrong
Obstruction 1: The programme metadata chronic issue

- Ingest user activity
- Ingest content metadata
- Train model
- Precompute
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

- Ingest user activity
- Ingest content metadata
- Train model
- Precompute
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**!
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 historical data.
Obstruction 3 When precompute failed due to training
Obstruction 3

When precompute failed due to training historical data.

```
"message": "No such object: datalab-sounds-prod-6c75-data/recommenders/xantus/model/2021-07-08T00:00:00+00:00/xantus.pkl"
```
Obstruction 3 When precompute failed due to training historical data future
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> on host airflow-worker-867b96c854-jzqw7


[2021-07-08 12:15:57,286] {pod_launcher.py:173} INFO - Event: move-model-to-gcs-3deac821b57047619c1c9505ddc5db18 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-3deac821b57047619c1c9505ddc5db18

[2021-07-08 12:17:57,499] {taskinstance.py:1152} ERROR - 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+00:00/xantus.pkl || true
'

[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

```bash
$ 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

- Ingest user activity
- Ingest content metadata
- Train model
  - Precompute
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

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":"Bad Request","code":400}"

("/opt/python3.6/lib/python3.6/site-packages/kubernetes/client/rest.py", line 231, in request
raise ApiException(http_resp=r
kubernetes.client.rest.ApiException: (400
Reason: Bad Request
Stuck with an obstructed DAG after an Airflow upgrade? You might want to try an intermittent DAG after an Airflow upgrade.

**Solution**

### airflow.contriboperators.kubernetes_pod_operator

*Executes task in a Kubernetes POD*

**Module Contents**

```python
class airflow.contriboperators.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_selector=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, dns_policy=None, schedulername=None, full_pod_spec=None, init_containers=None, log_events_on_failure=True, do_xcom_push=False, pod_template_file=None, priority_class_name=None, *args, **kwargs)[source]
```

[Source](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

Ingest user activity

Ingest content metadata

Train model

Precompute
Obstruction 5 When the Dataflow job said no

DAG: precompute_recommendations

Graph View

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

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
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

<table>
<thead>
<tr>
<th>successful</th>
<th>unsuccessful</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute-predictions-21628eba</td>
<td>compute-predictions-191557b2</td>
</tr>
<tr>
<td>2021-02-24_20_06_12-17726452853028084617</td>
<td>2021-02-28_20_01_56-16233421279575525541</td>
</tr>
<tr>
<td>25 February 2021</td>
<td>1st March 2021</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>successful</th>
<th>unsuccessful</th>
</tr>
</thead>
<tbody>
<tr>
<td>consumption-history/</td>
<td>1.87 GiB</td>
<td>1.88 GiB</td>
</tr>
<tr>
<td>consumed-items/</td>
<td>55.63 MiB</td>
<td>55.68 MiB</td>
</tr>
<tr>
<td>candidate-items/rfy/</td>
<td>25.06 MiB</td>
<td>25.09 MiB</td>
</tr>
<tr>
<td>xantus.pkl</td>
<td>4.67 GiB</td>
<td>4.69 GiB</td>
</tr>
<tr>
<td>item_features.npz</td>
<td>1.73 MiB</td>
<td>1.73 MiB</td>
</tr>
<tr>
<td>mapping.json</td>
<td>473.84 MiB</td>
<td>476.01 MiB</td>
</tr>
</tbody>
</table>

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
Obstruction 5 When the Dataflow job said no

Dataflow job started using the shuffle service unexpectedly

Tatiana Al-Chueyr
tatiana.alchueyr@bbc.co.uk
March 1, 2021 at 11:37:41 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: 2021-02-20_20_01_56-162324122796755255641).

It seems the Dataflow job ran out of disk, since the job logs contain the error:

```
RuntimeError: OSError: [Errno 26] No space left on device [while running 'Predict/Predict']
```

When comparing this unsuccessful (ID: 2021-02-20_20_01_56-162324122796755255641) run with the previous successful run (ID: 2021-02-20_20_01_56-177256528538288866617), 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 BKD or environment. We are unsure that, by default, the shuffle service enables the...
Obstruction 6 The never ending Dataflow job

- Ingest user activity
- Ingest content metadata
- Train model
- Precompute
Obstruction 6 The never ending Dataflow job

# datalab_devops_int – Jun 14th

google cloud monitoring APP 6:46 PM
Incident #0.m3oacbpboewe 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 ...

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
The never ending Dataflow job

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:
- compute-predictions-xantu-05130255-opno-harness-mtj1
  Root cause: The worker lost contact with the service.
- compute-predictions-xantu-05130255-opno-harness-2x4v
  Root cause: The worker lost contact with the service.
- compute-predictions-xantu-05130255-opno-harness-5gkd
  Root cause: The worker lost contact with the service.
- compute-predictions-xantu-05130255-opno-harness-2t1n
  Root cause: The worker lost contact with the service.
Obstruction 6  The never ending Dataflow job

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>End time</th>
<th>Elapsed time</th>
<th>Start time</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute-predictions-xantus-music-mixes-1-1-87d98f40</td>
<td>Batch</td>
<td>May 13, 2021, 10:44:25 AM</td>
<td>3 days 5 hr</td>
<td>May 10, 2021, 5:21:29 AM</td>
</tr>
<tr>
<td>compute-predictions-xantus-music-mixes-1-1-</td>
<td>Batch</td>
<td>May 13, 2021,</td>
<td>5 hr 12 min</td>
<td>May 13, 2021,</td>
</tr>
</tbody>
</table>
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
  - Unsynchronised 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

```python
"precompute_recommendations_dag": {
  "start_date": "2020-12-07T00:00:00.000Z",
  "redis_image": "marketplace.gcr.io/googleapis/redis",
  "dataflow": {
    "num_workers": "48",
    "notused_comments": "38 vCPU with 15 GB RAM per vCPU",
    "machine_type": "custom-38-460800-ext"
  },
  "path": {
    "candidate_items_gcs_prefix": "higcs_base_path/s/o/metadata/candidate-items/(( ts ))/",
    "workflows_redis_mass_path": "higcs_base_path/s/precomputed-recommendations/redis-mass-insertion/(( ts ))/",
    "workflows_redis_dump_path": "higcs_base_path/s/precomputed-recommendations/redis-dump/(( ts ))/",
    "api_redis_dump_path": "higcs_base_path/s/precomputed_recommendations/",
    "precompute_recommendations_dag_id": "precompute_recommendations_h(identifier)s"
  }
},
"ingest_user_activity_dag": {
  "start_date": "2020-12-07T00:00:00.000Z",
  "ingest_interval_hours": "6",
  "snapshot_interval_days": "129",
  "max_active_dag_runs": "3",
  "ingest_catchup": "1",
  "user_activity_bucket": "datalab-sounds-data-dev-ff72_ua",
  "user_activity_prefix": "dev-test_key",
  "path": {
    "delta_prefix": "user-activity/ua/deltas/(( ts ))/(formatted_instant_intervals)/",
    "list_original_path": "workflows/user-activity-to-process/original/(( ts ))/(formatted_instant_intervals)/",
    "user_activity_snapshots_path": "user-activity/ua/snapshots/(( ts ))/(formatted_snapshot_intervals)/",
    "list_deltas_path": "workflows/user-activity-to-process/deltas/(( ts ))/(formatted_snapshot_intervals)/"
  }
},
```

https://www.astronomer.io/guides/dynamically-generating-dags
Smell 2 Configuration (mis)patterns
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**
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