Testing Airflow Dags with dagtest

Victor Chiapaikeo
Aldo Orozco

Etsy

Airflow Summit
Let’s flow together
September 19-21, 2023,
Toronto, Canada
Agenda

- Airflow @ Etsy
- User Archetypes
- The Problem
- Solutions & Inspiration
- dagtest
- Internals of dagtest
- Call to Action
Airflow @ Etsy
Airflow @ Etsy

- 4 Environments
- 2000+ Active DAGs in prod
- ~30k tasks running daily
- 100s of active users
- 7 VMs - LocalExecutor - 1.10.3 → 1 K8s cluster - KubernetesExecutor - 2.6.3
User Archetypes
User Archetypes

- Data Engineers, Product Engineers, ML Engineers, Data Analysts / Scientists, etc.
- 2 Categories:
  - DAG Owner
  - Platform Developer
DAG Owners

- Majority of users (80%)
- Owns 1+ DAG(s)
- Familiar with writing dags and interacting via the UI
- Oncaller
- Checks internal and OS docs
- **May not know much about docker, k8s or python installations**
Platform Builder

- Minority (20%)
- Enabler of DAG owners
- Core maintainer or power user
- Creates new custom sensors, operators, macros, etc.
- Might be more knowledgeable of docker / python
The Problem
# Initial Symptoms

<table>
<thead>
<tr>
<th>Commit Message</th>
<th>Author</th>
<th>Date</th>
<th>SHA</th>
<th>Verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fix base path (#12846)</td>
<td>gauravanand25 (ganand)</td>
<td>committed 3 weeks ago</td>
<td>528248d</td>
<td>✔️</td>
</tr>
<tr>
<td>Fix base_path arg (#12845)</td>
<td>gauravanand25 (ganand)</td>
<td>committed 3 weeks ago</td>
<td>83e169c</td>
<td>✔️</td>
</tr>
<tr>
<td>Fix args (#12844)</td>
<td>gauravanand25 (ganand)</td>
<td>committed 3 weeks ago</td>
<td>7c74dfa</td>
<td>✔️</td>
</tr>
<tr>
<td>Migrate to spark job (#12841)</td>
<td>gauravanand25 (ganand)</td>
<td>committed 3 weeks ago</td>
<td>40c0a50</td>
<td>✔️</td>
</tr>
<tr>
<td>Update settings.yml (#12842)</td>
<td>mgallatin2 (mgallatin)</td>
<td>committed 3 weeks ago</td>
<td>32e939b</td>
<td>✔️</td>
</tr>
</tbody>
</table>
### More Symptoms

<table>
<thead>
<tr>
<th>Status</th>
<th>Experiment Name</th>
<th>Frequency</th>
<th>Assignee</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>exp_coarse_grained_base</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>5</td>
</tr>
<tr>
<td>On</td>
<td>exp_coarse_grained_baseb</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>5</td>
</tr>
<tr>
<td>On</td>
<td>exp_coarse_grained_basec</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>1</td>
</tr>
<tr>
<td>On</td>
<td>exp_coarse_grained_base_webboe</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>4</td>
</tr>
<tr>
<td>Off</td>
<td>exp_coarse_grained_lfv22b</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>5</td>
</tr>
<tr>
<td>Off</td>
<td>exp_coarse_grained_lfv23</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>3</td>
</tr>
<tr>
<td>Off</td>
<td>exp_coarse_grained_lfv23_webboe</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>5</td>
</tr>
<tr>
<td>Off</td>
<td>exp_coarse_grained_lfv24</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>5</td>
</tr>
<tr>
<td>Off</td>
<td>exp_coarse_grained_lfv25</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>2</td>
</tr>
<tr>
<td>Off</td>
<td>exp_coarse_grained_lfv26</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>1</td>
</tr>
<tr>
<td>Off</td>
<td>exp_coarse_grained_lfv27</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>3</td>
</tr>
<tr>
<td>Off</td>
<td>exp_coarse_grained_lfv28</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>5</td>
</tr>
<tr>
<td>On</td>
<td>exp_coarse_grained_lfv30b</td>
<td>@daily</td>
<td>jeskreiswinkler</td>
<td>5</td>
</tr>
</tbody>
</table>
What About...

- `airflow dag test` cli
- Running locally (breeze)
  - Colima / Docker Desktop
  - Resource restrictions
  - Run a single DAG on the CLI
  - Permissions
- GCP Composer
  - Too many steps

Updating and testing a deployed DAG

To test updates to your DAGs in your test environment:

1. Copy the deployed DAG that you want to update to `data/test`.
2. Update the DAG.
3. Test the DAG.
   a. Check for syntax errors.
   b. Check for task-specific errors.
4. Make sure the DAG runs successfully.
5. Turn off the DAG in your test environment.
   a. Go to the Airflow UI > DAGs page.
   b. If the DAG you’re modifying runs constantly, turn off the DAG.
   c. To expedite outstanding tasks, click the task and **Mark Success**.
6. Deploy the DAG to your production environment.
   a. Turn off the DAG in your production environment.
   b. Upload the updated DAG to the `dags/` folder in your production environment.
The problem

- If testing is not SUPER simple, users will work around it - testing in prod or no tests at all.
- Testing is risky in production - overwrite data
Solutions & Inspiration
Solutions

● We shouldn’t...
  • Require users to manually gsutil cp some dag file to some bucket
  • Require users to pull some large image down
  • Require users to pip install some large package with transitive dependencies and then setup confusing plugin steps
● Test environment should be isolated from production
● Needs to be AS SIMPLE AS POSSIBLE
Solutions

- Some inspiration:
  - Databricks
  - Spark Shell
  - Apache Beam - run pipelines locally with LocalRunner
  - Client / server architecture
Back to the 2 archetypes

Airflow Dag Owners

Airflow Platform Developers

Clima

Airflow Breeze

OpenAPI Initiative

Docker
dagtest
dagtest

- Attempts to empower users with the minimum requirements necessary to run a dag (or parts of a dag)
- Client part of client/server architecture
- Packaged as an internal PyPi package
- Tiny (13KB vs the base Airflow package’s tar.gz at 11MB)
- Users send dags to an adhoc/test environment ONLY running with SA’s that do not have access to Production buckets
Testing w/ dagtest

```
pip install etsy-dagtest

dagtest path/to/my/dag.py [<execution-date>] [--dry-run] [--ignore-dependencies] [--task] [--end-date]
```

```
vchiapaikeo@7676:airflow (testing-is-fun)$ dagtest
Checking boundary-layer not installed: CHECKED
Checking boundary-layer2 is properly installed: CHECKED - boundary-layer2 - 2.2.2
Checking boundary-layer-etsy-plugin not installed: CHECKED
Checking boundary-layer-etsy-plugin2 is properly installed: CHECKED - boundary-layer-etsy-plugin 2 - 2.0.88


dagtest: error: the following arguments are required: environments/prod/dags/dag.py
```

vchiapaikeo@7676:airflow (testing-is-fun)$
```
Checking boundary-layer not installed: CHECKED
Checking boundary-layer2 is properly installed: CHECKED - boundary-layer2 - 2.2.2
Checking boundary-layer-etsy-plugin not installed: CHECKED
Checking boundary-layer-etsy-plugin2 is properly installed: CHECKED - boundary-layer-etsy-plugin2 - 2.0.88

2023-09-13 14:12:33,804 - boundary-layer v. 2.2.2 - INFO - Loaded plugins default, etsy
Checking tool's version: PASSED
Python DAG detected

Execution Details:
===============================================================================
Run Type | Test Run
Username | vchiapaikeo
Skip PR Tests | False
DAG File Name | my_first_dag.py
DAG ID | None
Execution Date Start | 2023-09-12T00:00:00
Execution Date End | None
Task ID | None
Task ID is Regex | False
Ignore Dependencies | False
Ignore First Depends on Past | False
Local Include Directory Path | None
Conf | None
Request Id | 4b25e7b6
===============================================================================

Proceed with the test? [yes|y/no|n]: y
Uploading DAG... DONE
Running Unit Tests... SUCCESS
Initiating Test Run... STARTED
Waiting for first task to start... DONE

DAG my_first_dag_1694628754 logs available at:
  https://web.airflow-adhoc.etsy-syseng-gke-prod.etsyclick.com/dags/my_first_dag_1694628754/grid?dag_run_id=backfill_2023-09-12T00%3A00%3A00%2B00%3A00

Please Note: These jobs will not respect the scheduler. Toggling these DAGs on will have no effect.
Testing w/ dagtest
Internals
# This version should match with whatever version is set in
# airflow/overrides.yaml#L1
FROM airflow:2.5.3-python3.10-etsy-0.0.5

USER root

ENV AIRFLOW_HOME=/opt/airflow

RUN export ACCEPT_EULA=Y && \
    apt-get update && apt-get upgrade -y && \
    apt-get install -y procps htop && \
    rm -r /var/lib/apt/lists /var/cache/apt/archives

COPY modules/etsy-test-api $AIRFLOW_HOME/etsy-test-api

RUN chown -R airflow:root $AIRFLOW_HOME/etsy-test-api

COPY tests $AIRFLOW_HOME/tests

RUN chown -R airflow:root $AIRFLOW_HOME/tests

COPY requirements-test.txt $AIRFLOW_HOME/requirements-test.txt

RUN chown -R airflow:root $AIRFLOW_HOME/requirements-test.txt

USER airflow

RUN pip install -e $AIRFLOW_HOME/etsy-test-api

RUN pip install --r $AIRFLOW_HOME/requirements-test.txt

CMD ["bash", "-c", "gunicorn --bind 0.0.0.0:9107 -k gevent --workers 4 --t 120 --access-logfile --log-level debug etsy_test_api.server:app"]
What happens when users call dagtest?

1. pip install etsy-test-client
2. dagtest path/to/my/dag.py
3. Makes POST request to upload file to files-sync API server
4. Files sync has R/W mount for dags and writes dags to a dedicated adhoc directory
5. Second request made to test API server to perform validation and initiate test
6. Validation (place dag in dagbag) happens on test-api server (fast)
7. Pod created to run airflow dags backfill command
8. airflow dags backfill /mnt/path/to/my/dag.py
9. User provided a link to adhoc dag and can control from there

Filestore (NFS)
Environments

**Test**
- Airflow running KubernetesExecutor
- Workload
- Identity Pool
- Test SA
- GCS Buckets (test)
- Bigquery Datasets (dev)
- Quotas (test)
- Filestore (NFS Mount)

**Production**
- Airflow running KubernetesExecutor
- Workload
- Identity Pool
- Prod SA
- GCS Buckets (prod)
- Bigquery Datasets (prod)
- Quotas (prod)
- Filestore (NFS Mount)
The Outcome

- An empowered dag owner who can safely test dags outside of Production and faster iteration cycles
- PRs for dags can be merged with more confidence
- Lower likelihood that Production data is corrupted during testing
- Users uncover permissions issues in adhoc instead of Production
- If modifying a large Production dag, a single task or set of new tasks can be isolated and tested
Call to Action
Call to Action

- Gap in OSS offering
- `airflow dags test` does not work well for even slightly customized environments
- `airflow dags backfill` does not have proper REST API support
- Trigger dag functionality in REST API requires that a dag exists on some instance
- Trigger dag also must run the dag from E2E (can’t test single or subset of tasks)
- Users shy away from Airflow because of the difficulty in testing / developing
- How can we build a better solution for this?
Questions?