Behaviour Driven Development In Airflow

Ole Christian Langfjæran

In Norway

Father of 3 boys

unacast.

unacast.

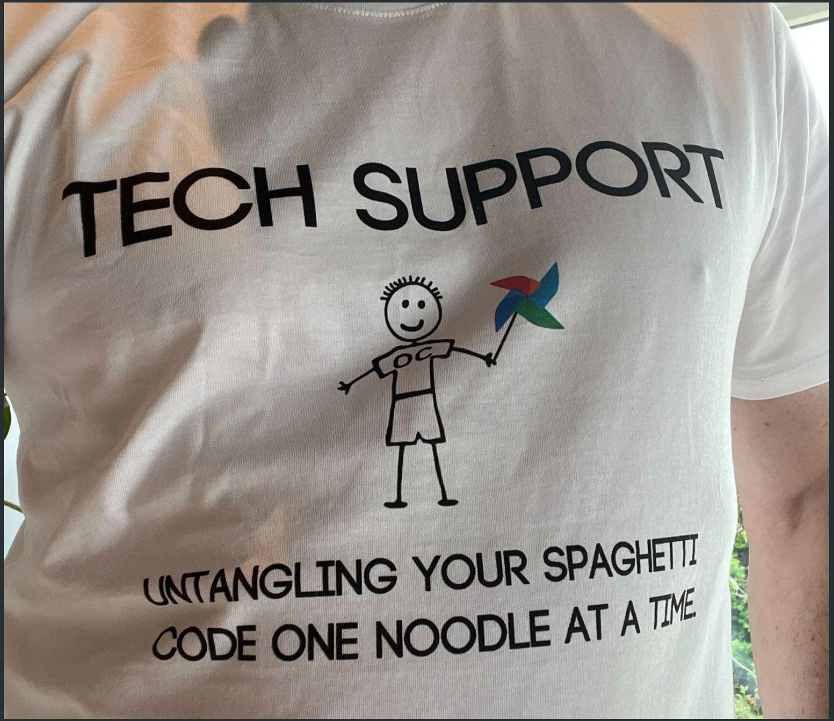
Location insights

Devops/platform engineer

Scientists > engineers

Location insights

Devops/platform engineer



unacast.

Airflow

since 2018

Currently we’re doing some major code refactoring

AIP-31 instead of DAG factories

Automated tests

What to test?

validation (import errors)
rendering
execute
task dependencies

Some are difficult

```python
def test_render_template(self, session, clean_dags_and_dagruns):
    operator = AwsToAwsBaseOperator(
        task_id="dynamodb_to_s3_test_render",
        dag=self.dag,
        source_aws_conn_id="{{ ds }}",
        dest_aws_conn_id="{{ ds }}",
    )
    ti = TaskInstance(operator, run_id="something")
    ti.dag_run = DagRun(
        dag_id=self.dag.dag_id,
        run_id="something",
        execution_date=timezone.datetime(2020, 1, 1),
        run_type=DagRunType.MANUAL,
    )
    session.add(ti)
    session.commit()
    ti.render_templates()
    assert "2020-01-01" == getattr(operator, "source_aws_conn_id")
    assert "2020-01-01" == getattr(operator, "dest_aws_conn_id")
```

airflow/tests/providers/amazon/aws/transfers/test_base.py

```
def test_execute(self, dag_maker):
    with conf_vars({("email", "email_backend"): "tests.operators.test_email.send_email_test"}):
        with dag_maker(
            "test_dag",
            default_args={"owner": "airflow", "start_date": DEFAULT_DATE},
            schedule=INTERVAL,
            serialized=True,
        ):
            task = EmailOperator(
                to="airflow@example.com",
                subject="Test Run",
                html_content="The quick brown fox jumps over the lazy dog",
                task_id="task",
                files=["/tmp/Report-A-{{ ds }}.csv"],
                custom_headers={"Reply-To": "reply_to@example.com"},
            )
        dag_maker.create_dagrun()
        task.run(start_date=DEFAULT_DATE, end_date=DEFAULT_DATE)
    assert send_email_test.call_count == 1
    call_args = send_email_test.call_args.kwargs
    assert call_args["files"] == ["/tmp/Report-A-2016-01-01.csv"]
    assert call_args["custom_headers"] == {"Reply-To": "reply_to@example.com"}
```

```python
def test_retries_present():
 dag_bag = DagBag(dag_folder='dags/', include_examples=False)
 for dag in dag_bag.dags:
     retries = dag_bag.dags[dag].default_args.get('retries', [])
     error_msg = 'Retries not set to 2 for DAG {id}'.format(id=dag)
     assert retries == 2, error_msg
```

github.com/astronomer/airflow-testing-guide/test_dag_validation.py

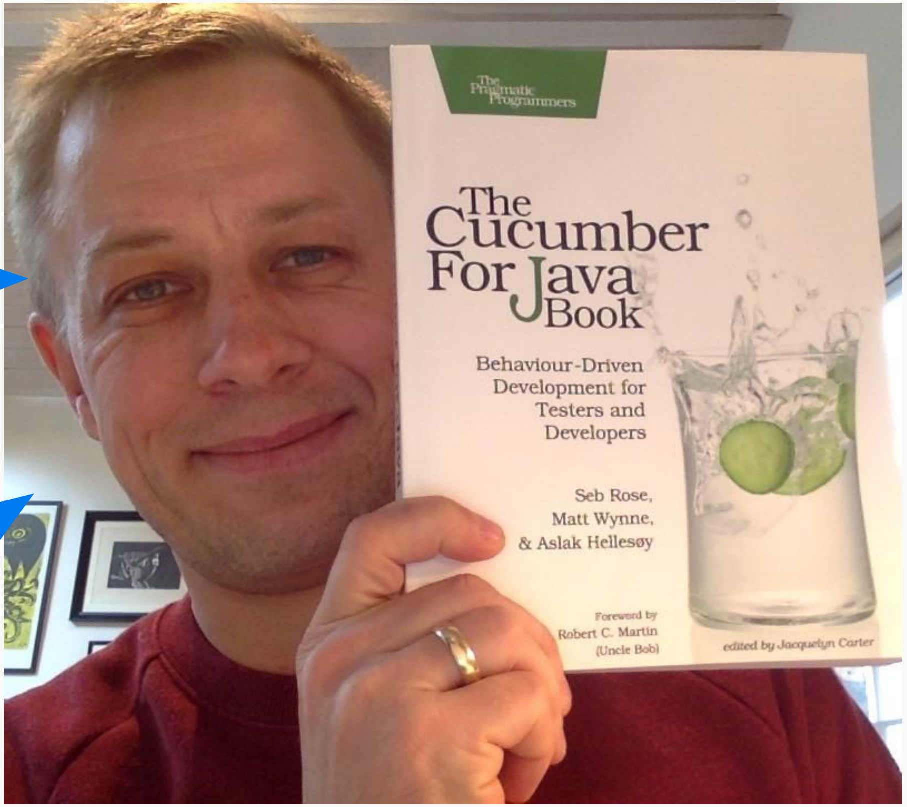
Scientists > engineers

can it be more readable? DRY?

Aslak Hellesøy

inspiration

evangelist


The Pragmatic Programmers
The Cucumber For Java Book
Behaviour-Driven Development for Testers and Developers
Seb Rose, Matt Wynne, & Aslak Hellesøy
Foreword by Robert C. Martin (Uncle Bob)
edited by Jacquelyn Carter

BDD

Behaviour Driven Development

A language to describe code behaviour

Readable documentation, that is also **executable tests**

```
Feature: Is it Friday yet?
 As a human
 I want to know if it is Friday
 So I know that I can soon relax

 Scenario: Monday isn't Friday
   Given today is Monday
   When I ask whether it's Friday yet
   Then I should be told "Nope"
```

BDD

Behaviour Driven Development

Narrative

Acceptance criteria

```
Feature: Is it Friday yet?
 As a human
 I want to know if it is Friday
 So I know that I can soon relax

 Scenario: Monday isn't Friday
   Given today is Monday
   When I ask whether it's Friday yet
   Then I should be told "Nope"
```

STEVE MARTIN
CHEVY CHASE
MARTIN SHORT
¡THREE AMIGOS!
An L.A. FILMS/LORNE MICHAELS Production A LANDIS/FOLSEY Film ¡THREE AMIGOS!
ALFONSO ARAU TONY PLANA PATRICE MARTINEZ Songs by RANDY NEWMAN Score by ELMER BERNSTEIN
Executive Producer STEVE MARTIN Written by STEVE MARTIN · LORNE MICHAELS · RANDY NEWMAN
Produced by LORNE MICHAELS and GEORGE FOLSEY, JR. Directed by JOHN LANDIS
ORION
In Association With HOME BOX OFFICE

Business (product owner)

Development

Testing

BDD

Behaviour Driven Development

Given: the initial context at the beginning of the scenario, in one or more clauses;

When: the event that triggers the scenario;

Then: the expected outcome, in one or more clauses.

```
Feature: Is it Friday yet?
 As a human
 I want to know if it is Friday
 So I know that I can soon relax


 Scenario: Monday isn't Friday
   Given today is Monday
   When I ask whether it's Friday yet
   Then I should be told "Nope"
```

BDD

Behaviour Driven Development

Plain text files (.feature)

Syntax is named Gherkin

Best known interpreters are **cucumber.io** and **behave** (Python)

```
Feature: Is it Friday yet?
 As a human
 I want to know if it is Friday
 So I know that I can soon relax

 Scenario: Monday isn't Friday
   Given today is Monday
   When I ask whether it's Friday yet
   Then I should be told "Nope"
```

Oh behave

A concrete example

```
Feature: Should be able to run simple BigQuery behave steps


 Scenario: Count rows
   Given the bigquery sql
     """
     SELECT "Clark Kent" as name
     """
   When we run the query
   Then the result should have 1 row
```

Github/VS Code/IntelliJ provides automatic syntax highlighting of .feature files

Step implementations

For reading the gherkin files

context object is supplied by behave

```python
@given("the bigquery sql")
def _given_query(context):
    context.query = context.text
```

features/steps/bigquery_steps.py

```python
@when("we run the query")
def _run_query(context):
    query_job = run_query(context.query)

    # Wait for result and add result to context
    context.query_result = query_job.result()
```

features/steps/airflow_steps.py

```
@then("the result should have {num_rows} row")
def _result_should_have_num_rows(context, num_rows):
    assert str(context.query_result.total_rows) == str(
        num_rows
    ), f'Expected "{num_rows}" but was
"{context.query_result.total_rows}"'
```

```gherkin
Scenario: All BigQueryInsertJobOperator should always use labels
   Given airflow is running
   And tasks type is
"airflow.providers.google.cloud.operators.bigquery.BigQueryInsertJobOperator"
   Given attribute "configuration.labels"
   Then value should contain "turbine_environment"
```

test driven

```
Scenario: Hotspot threshold reached
   Given max hotspots "3"
   And the following bumps
       | bump_id   | location_event_timestamp       | location_event_latitude | location_event_longitude |
       | bump_id_1 | 2023-02-15 21:59:29.000000 UTC | 35.18276678             | -92.63838978             |
       | bump_id_2 | 2023-02-15 21:59:29.000000 UTC | 35.18276678             | -92.63838978             |
       | bump_id_3 | 2023-02-15 21:59:29.000000 UTC | 35.18276678             | -92.63838978             |
       | bump_id_4 | 2023-02-15 21:59:29.000000 UTC | 35.18276678             | -92.63838978             |
   When we run pre-clustering (create_bump_collection)
   Then the result should have 0 row
```

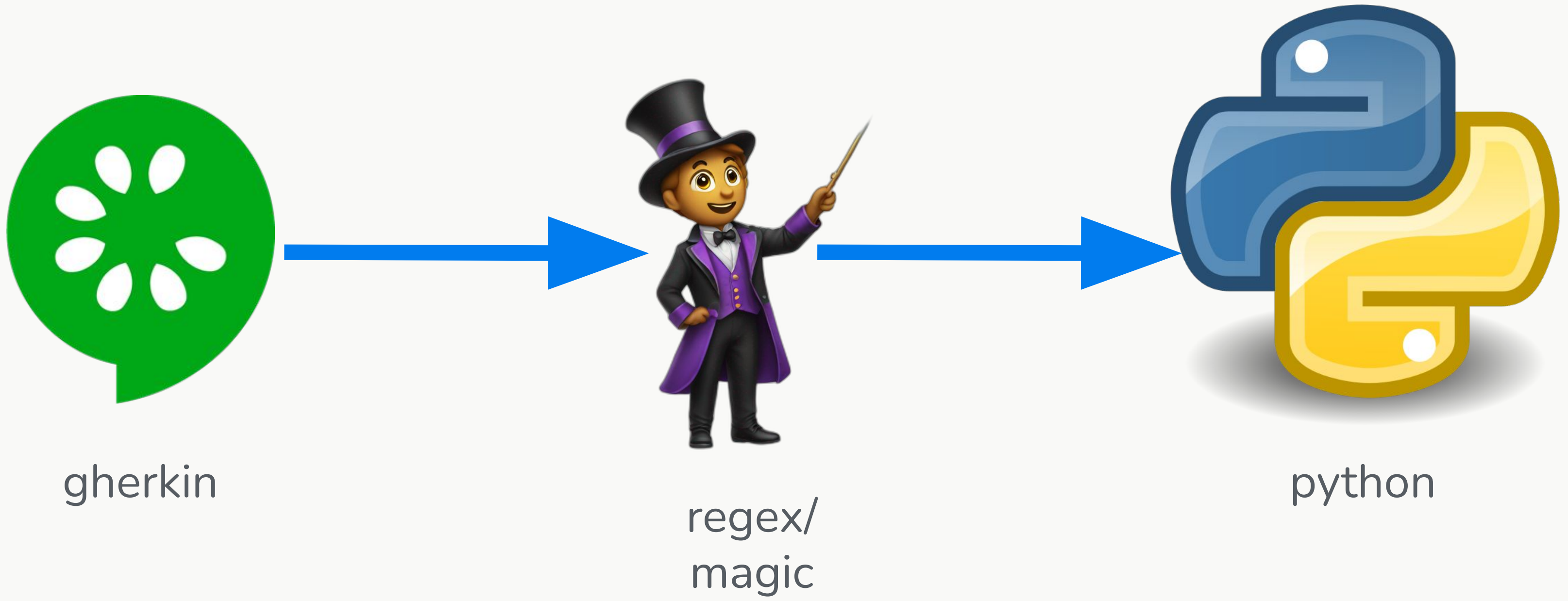
this example runs bigquery and asserts the output

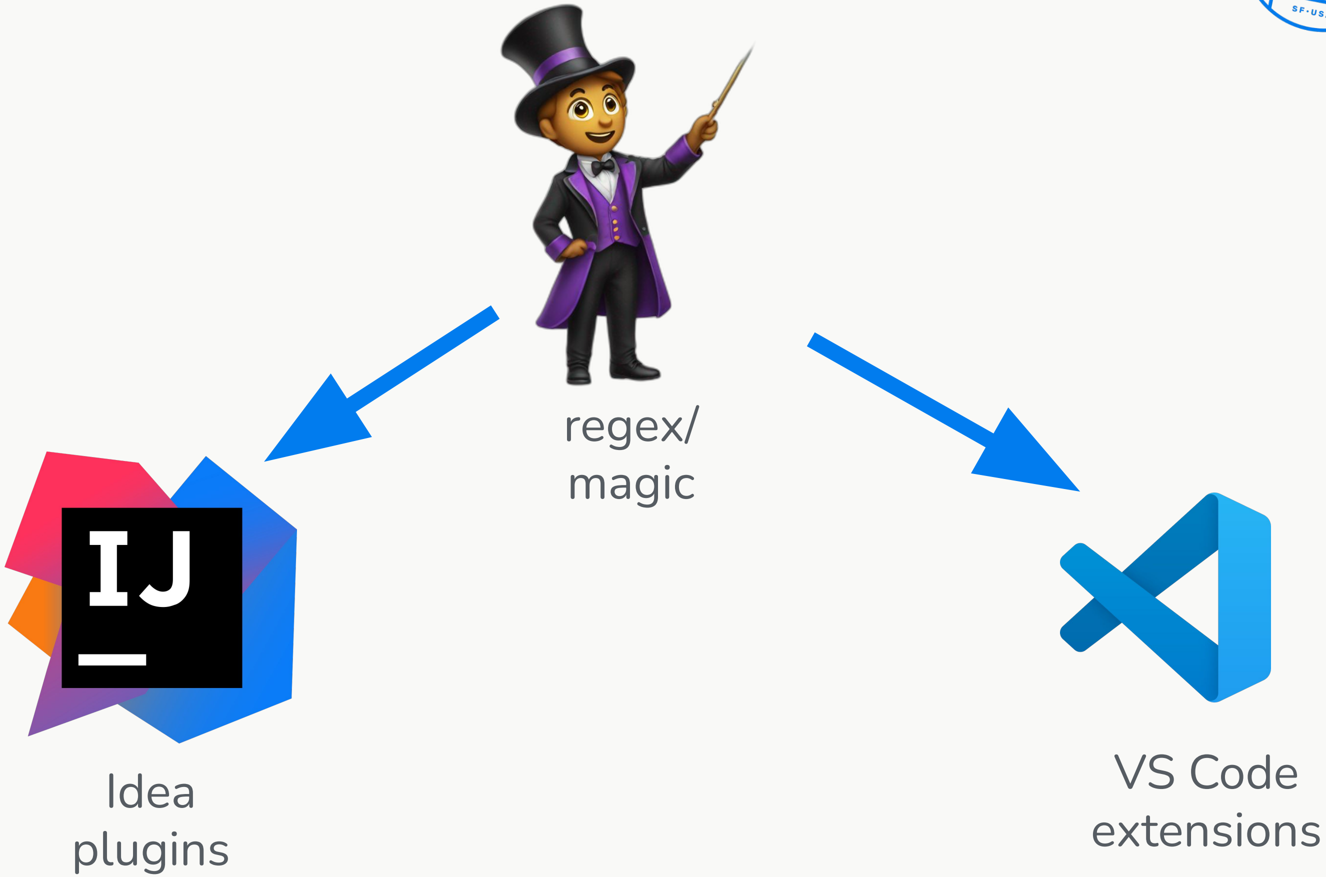
this looks nice?

maybe?

but it’s not a breeze

(pun intended [potiuk])

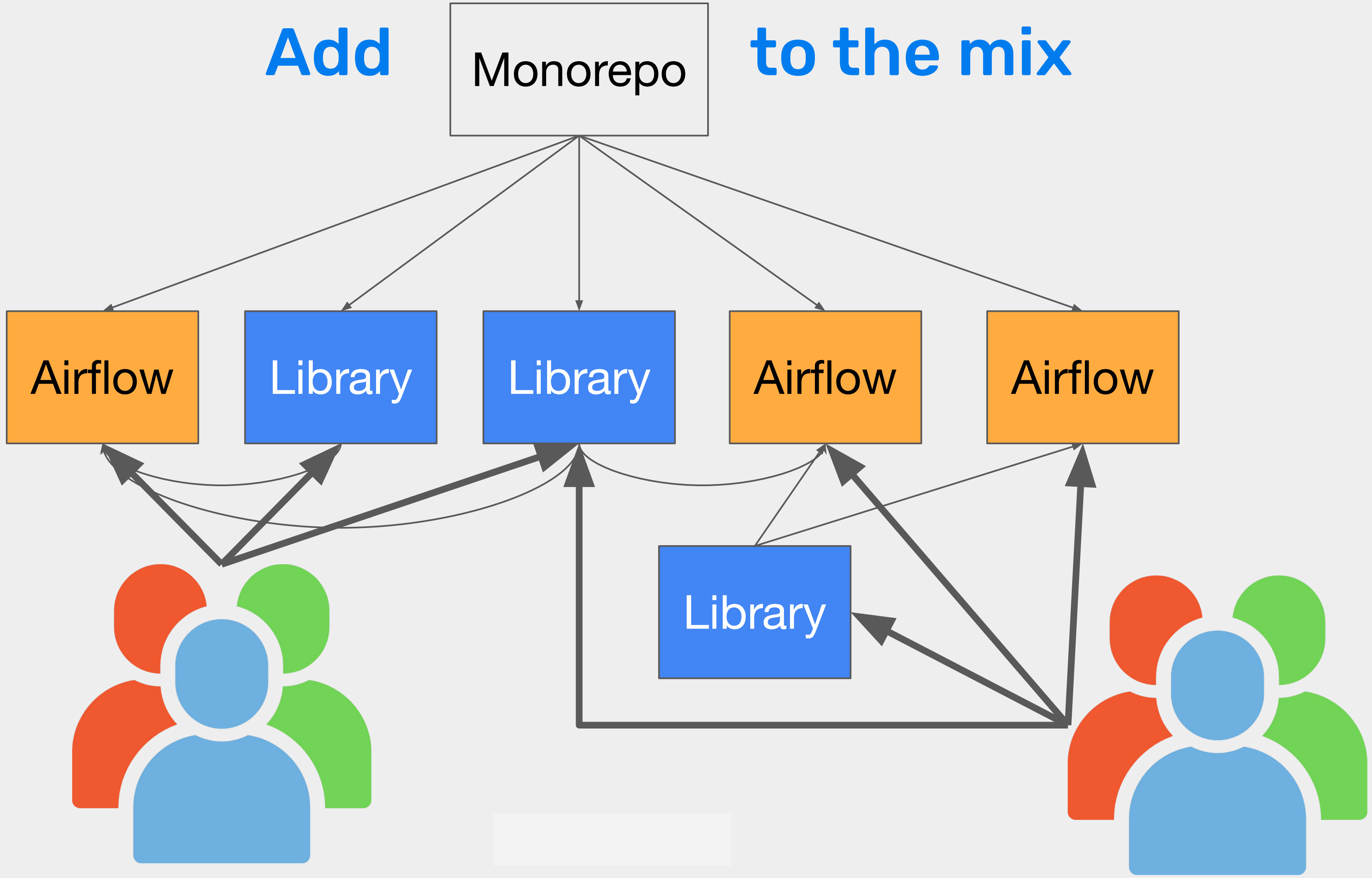

gherkin
regex/
magic
python


regex/
magic
Idea
plugins
VS Code
extensions

Devs be like

Where is the python step?
Why doesn’t intellisense work?
Where is the stack trace?
How can I run this scenario in VS Code?
IntelliJ? PyCharm?
How do I debug?

Add Monorepo to the mix



unacast.

let’s try pytest

no extra tooling, write steps as code

let’s have a look

```python
@task()
def create_random_number():
    return random.randint(1, 100)


@task
def print_a_random_number(the_number: str):
    return f"The random number is: '{the_number}'"


with DAG(dag_id="example"):
    random_number_task = create_random_number()
    print_a_random_number(random_number_task)
```

```python
def test_print_the_number(bdd: TurbineBDD):
    bdd.given_dag("example")
    bdd.given_xcom(task_id="create_random_number", value=22)
    bdd.given_task("print_a_random_number")
    bdd.when_I_render_the_task_template_fields()
    bdd.and_I_execute_the_task()
    bdd.then_it_should_match(equal_to("The random number is: '22'"))
```

https://github.com/judoole/airflow-aip-31-bonanza/.../tests/test_example.py

```python
def test_number_of_warnings(bdd: TurbineBDD):
    """Useful for preparing for Airflow upgrades"""
    bdd.when_I_get_all_dagbag_warnings()
    bdd.then_it_should_match(has_length(less_than(315)))


def test_import_errors(bdd: TurbineBDD):
    """Ensure that have no errors in our DAGs"""
    bdd.when_I_get_the_DagBag()
    bdd.then_it_should_match(has_property("import_errors", has_length(0)))
    bdd.then_it_should_match(has_property("dags", has_length(greater_than(0))))
```

```python
@pytest.mark.parametrize("dag", [
    "cdl_signals_test_pipeline_dk",
    "cdl_signals_test_pipeline_no-uat",
    ...
])
def test_cdl_pipeline_contains_correct_tasks(bdd: TurbineBDD, dag):
    bdd.given_dag(dag)
    bdd.when_I_get_all_the_tasks_ids()
    bdd.then_it_should_match(has_item("create_outputs.create_output_dataset"))
    bdd.then_it_should_match(has_item("create_outputs.create_processed_bucket"))
    bdd.then_it_should_match(has_item("create_outputs.create_untar_bucket"))
    ...
```

```
def test_task_load_to_bigquery_is_rendered_correctly(bdd: TurbineBDD):
   bdd.given_dag("cdl_signals_test_pipeline_dk")
   bdd.given_execution_date("2024-08-13")
   bdd.given_task("load_files.load_to_bigquery")
   bdd.when_I_render_the_task_template_fields()
   bdd.then_it_should_match(
       has_property("downstream_task_ids",
                    has_item("load_files.check_incoming_signals")))
```

```python
def test_h3_catalogue_renders_valid_sql(bdd: TurbineBDD):
    bdd.given_a_dag()
    bdd.given_execution_date("2030-12-31")
    bdd.given_task(
        H3Catalogue(
            source_signals="temp1",
            destination="temp2",
            h3_level=10,
            carto="carto-eu-{{ds}}",
        )
    )
    bdd.when_I_render_the_task_template_fields()
    # Use sqlfluff to parse the rendered SQL
    bdd.then_it_should_match(has_query(a_valid_sql()))
```

```python
def test_use_populate_from_datacontract(bdd: TurbineBDD):
    @bql.query(sql="SELECT count(*) as total FROM `{{project_id}}.{{dataset_id}}.{{table_id}}`")
    def simple_query(project_id: str, dataset_id: str, table_id: str) -> None:
        pass

    bdd.given_I(populate_table_with_data(
        contract=EXAMPLE_CONTRACT_SINGLE_FILE,
        table_id="test_table"))
    bdd.given_a_dag()
    bdd.given_task(simple_query(
        project_id=bdd.context['table_test_table'].project,
        dataset_id=bdd.context['table_test_table'].dataset_id,
        table_id=bdd.context['table_test_table'].table_id))
    bdd.when_I_render_the_task_template_fields()
    bdd.when_I_execute_the_task()
    bdd.when_I(get_query_result_rows())
    bdd.then_it_should_match(has_item(
        has_entries({
            'total': equal_to(3)
        })
    ))
```

Actually runs the query on BigQuery

implementation

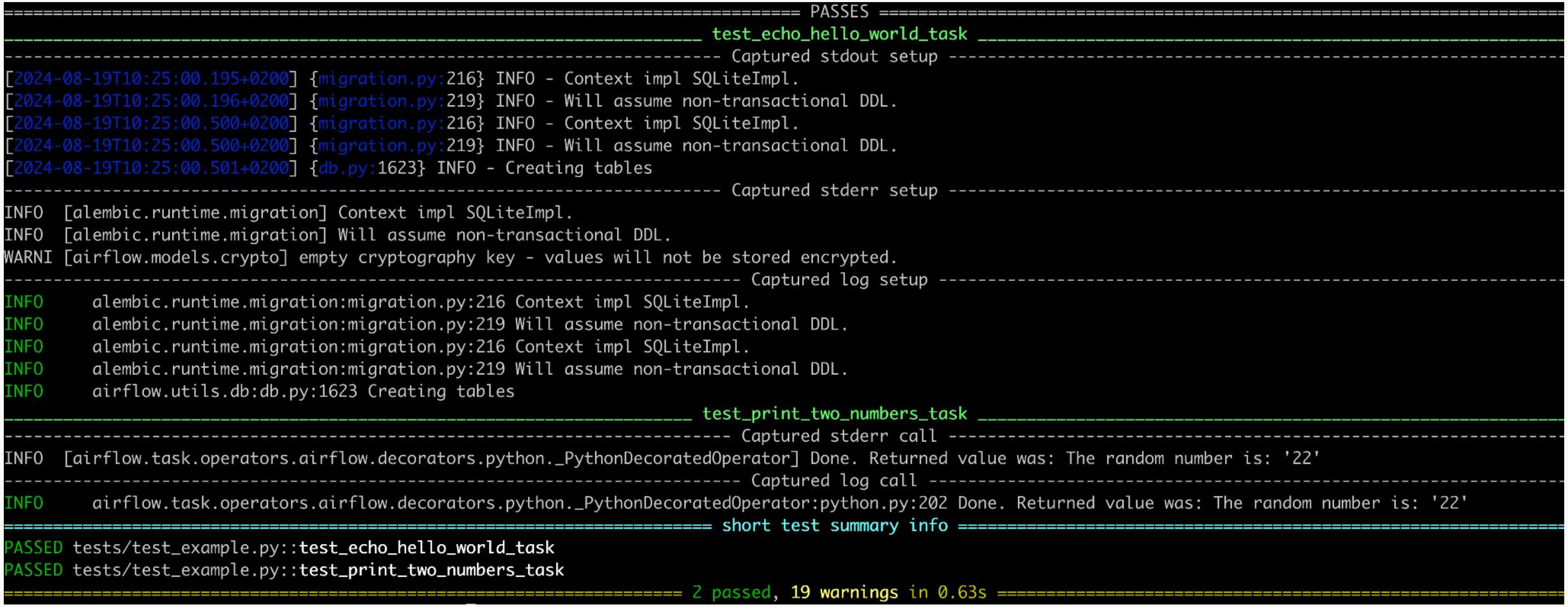
```python
@dataclass
class TurbineBDD:
    """A class for managing BDD context in relation to Airflow.
    This class contains methods for given, when, then, and other BDD
keywords.
    The given statements save the state of the system,
    the when statements act upon the state and save the result,
    and the then statements verify the state.
    """
```

```python
dag_bag: DagBag
dag: DAG = None
task: Operator = None
# It just saves the latest thing we retrieve
# Typically in a when statement
it: Any = None
# DAG context
execution_date: pendulum.DateTime = pendulum.today('UTC').add(-1)
dag_run_conf: Dict = None
xcoms: List = field(default_factory=list)
task_instance: TaskInstance = None
```

```python
def given_dag(self, dag_id_or_dag: Any):
    """Given a DAG with the given dag_id exists, and save it as the current DAG."""
    if isinstance(dag_id_or_dag, DAG):
        self.dag = dag_id_or_dag
    else:
        self.dag = self.dag_bag.get_dag(dag_id_or_dag)
    self.it = self.dag
```

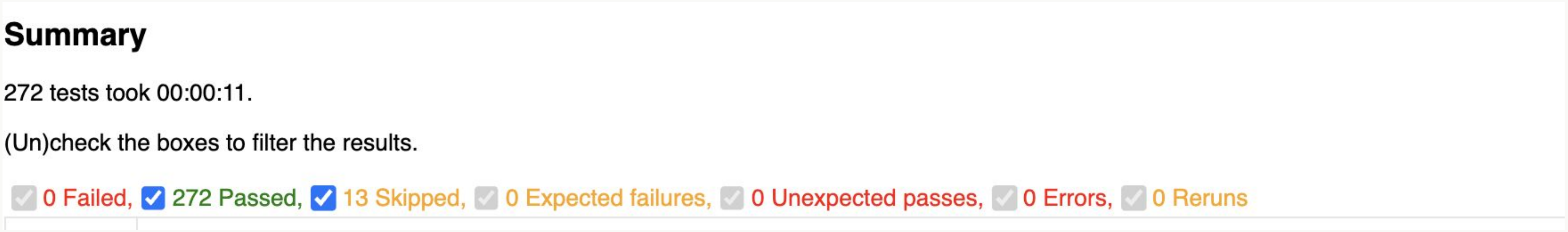
```python
def then_it_should_match(self, matcher: str):
    """Verify that "it" matches something.
    This is a simple wrapper around hamcrest's assert_that.

    See https://pyhamcrest.readthedocs.io/en/release-1.8/library/
    for more information on how to use hamcrest matchers.
    """
    assert_that(self.it, matcher)
```

run it



```
=============================================================================== PASSES ===============================================================================
______________________________________________________________________ test_echo_hello_world_task ______________________________________________________________________
------------------------------------------------------------------------ Captured stdout setup ------------------------------------------------------------------------
[2024-08-19T10:25:00.195+0200] {migration.py:216} INFO - Context impl SQLiteImpl.
[2024-08-19T10:25:00.196+0200] {migration.py:219} INFO - Will assume non-transactional DDL.
[2024-08-19T10:25:00.500+0200] {migration.py:216} INFO - Context impl SQLiteImpl.
[2024-08-19T10:25:00.500+0200] {migration.py:219} INFO - Will assume non-transactional DDL.
[2024-08-19T10:25:00.501+0200] {db.py:1623} INFO - Creating tables
------------------------------------------------------------------------ Captured stderr setup ------------------------------------------------------------------------
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
WARNI [airflow.models.crypto] empty cryptography key - values will not be stored encrypted.
-------------------------------------------------------------------------- Captured log setup --------------------------------------------------------------------------
INFO     alembic.runtime.migration:migration.py:216 Context impl SQLiteImpl.
INFO     alembic.runtime.migration:migration.py:219 Will assume non-transactional DDL.
INFO     alembic.runtime.migration:migration.py:216 Context impl SQLiteImpl.
INFO     alembic.runtime.migration:migration.py:219 Will assume non-transactional DDL.
INFO     airflow.utils.db:db.py:1623 Creating tables
_____________________________________________________________________ test_print_two_numbers_task _____________________________________________________________________
------------------------------------------------------------------------- Captured stderr call -------------------------------------------------------------------------
INFO  [airflow.task.operators.airflow.decorators.python._PythonDecoratedOperator] Done. Returned value was: The random number is: '22'
-------------------------------------------------------------------------- Captured log call --------------------------------------------------------------------------
INFO     airflow.task.operators.airflow.decorators.python._PythonDecoratedOperator:python.py:202 Done. Returned value was: The random number is: '22'
======================================================================= short test summary info =======================================================================
PASSED tests/test_example.py::test_echo_hello_world_task
PASSED tests/test_example.py::test_print_two_numbers_task
===================================================================== 2 passed, 19 warnings in 0.63s =====================================================================
```

standard pytest invocation in terminal



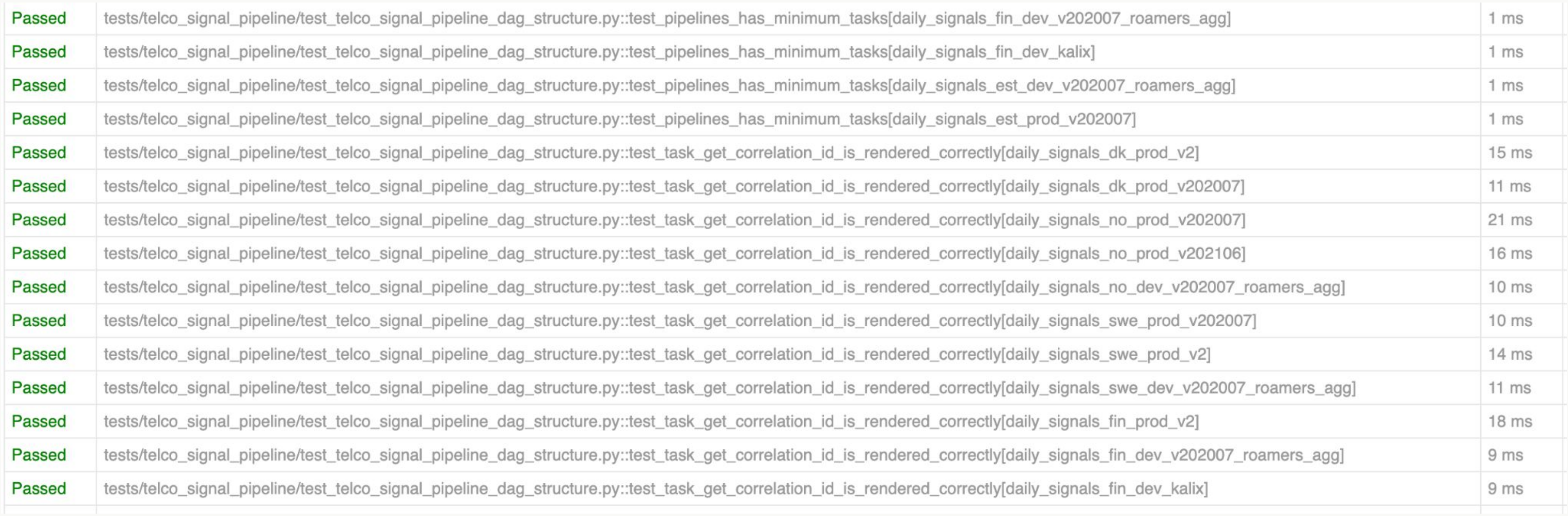
Summary

272 tests took 00:00:11.

(Un)check the boxes to filter the results.

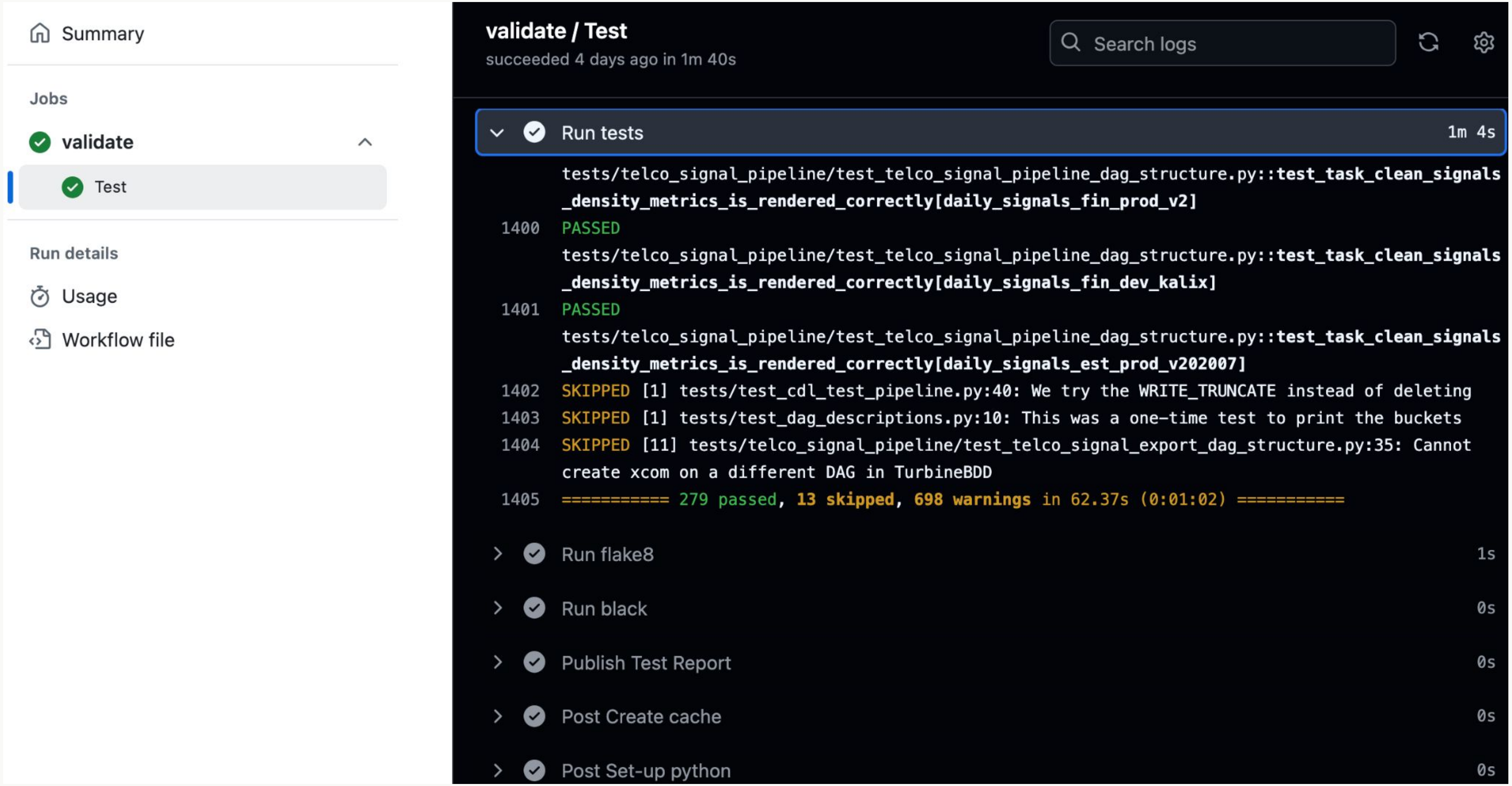
✓ 0 Failed, ✓ 272 Passed, ✓ 13 Skipped, ✓ 0 Expected failures, ✓ 0 Unexpected passes, ✓ 0 Errors, ✓ 0 Reruns

example from a project in the monorepo



Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_pipelines_has_minimum_tasks[daily_signals_fin_dev_v202007_roamers_agg]	1 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_pipelines_has_minimum_tasks[daily_signals_fin_dev_kalix]	1 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_pipelines_has_minimum_tasks[daily_signals_est_dev_v202007_roamers_agg]	1 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_pipelines_has_minimum_tasks[daily_signals_est_prod_v202007]	1 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_get_correlation_id_is_rendered_correctly[daily_signals_dk_prod_v2]	15 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_get_correlation_id_is_rendered_correctly[daily_signals_dk_prod_v202007]	11 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_get_correlation_id_is_rendered_correctly[daily_signals_no_prod_v202007]	21 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_get_correlation_id_is_rendered_correctly[daily_signals_no_prod_v202106]	16 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_get_correlation_id_is_rendered_correctly[daily_signals_no_dev_v202007_roamers_agg]	10 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_get_correlation_id_is_rendered_correctly[daily_signals_swe_prod_v202007]	10 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_get_correlation_id_is_rendered_correctly[daily_signals_swe_prod_v2]	14 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_get_correlation_id_is_rendered_correctly[daily_signals_swe_dev_v202007_roamers_agg]	11 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_get_correlation_id_is_rendered_correctly[daily_signals_fin_prod_v2]	18 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_get_correlation_id_is_rendered_correctly[daily_signals_fin_dev_v202007_roamers_agg]	9 ms
Passed	tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_get_correlation_id_is_rendered_correctly[daily_signals_fin_dev_kalix]	9 ms

example from a project in the monorepo



Summary

Jobs

validate

Test

Run details

Usage

Workflow file

validate / Test

succeeded 4 days ago in 1m 40s

Search logs

Run tests 1m 4s

```
     tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_clean_signals
     _density_metrics_is_rendered_correctly[daily_signals_fin_prod_v2]
1400 PASSED
     tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_clean_signals
     _density_metrics_is_rendered_correctly[daily_signals_fin_dev_kalix]
1401 PASSED
     tests/telco_signal_pipeline/test_telco_signal_pipeline_dag_structure.py::test_task_clean_signals
     _density_metrics_is_rendered_correctly[daily_signals_est_prod_v202007]
1402 SKIPPED [1] tests/test_cdl_test_pipeline.py:40: We try the WRITE_TRUNCATE instead of deleting
1403 SKIPPED [1] tests/test_dag_descriptions.py:10: This was a one-time test to print the buckets
1404 SKIPPED [11] tests/telco_signal_pipeline/test_telco_signal_export_dag_structure.py:35: Cannot
     create xcom on a different DAG in TurbineBDD
1405 =========== 279 passed, 13 skipped, 698 warnings in 62.37s (0:01:02) ===========
```

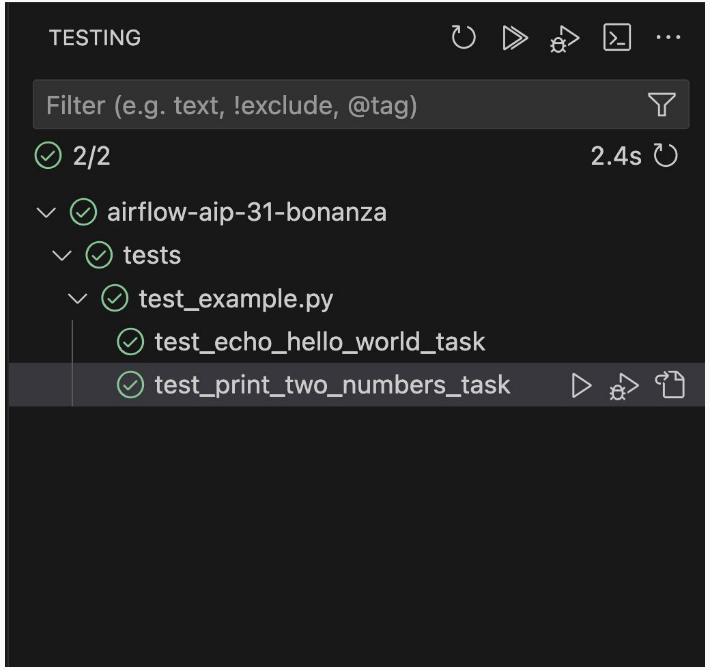
Run flake8 1s

Run black 0s

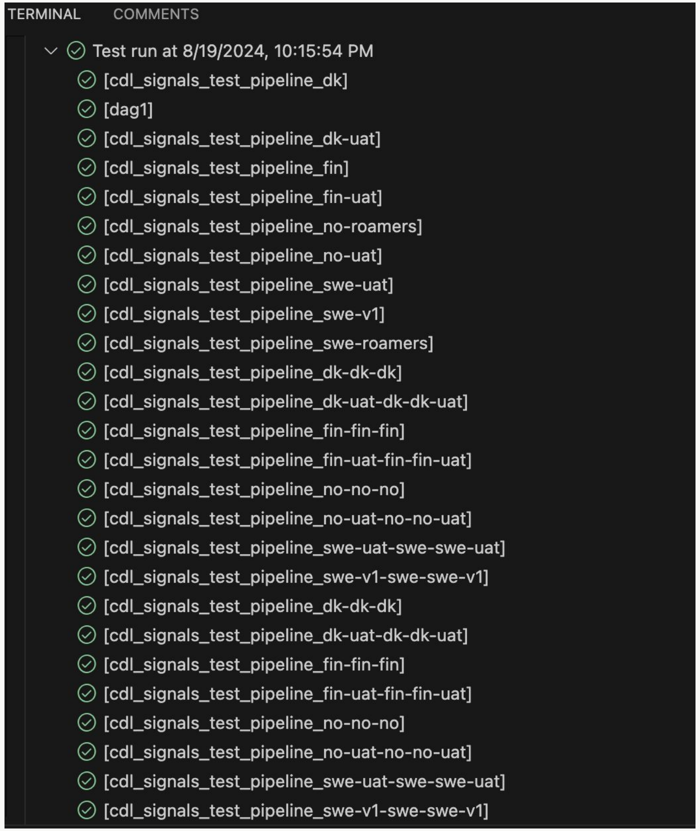
Publish Test Report 0s

Post Create cache 0s

Post Set-up python 0s



```
TESTING
Filter (e.g. text, !exclude, @tag)
2/2                                   2.4s
airflow-aip-31-bonanza
  tests
    test_example.py
      test_echo_hello_world_task
      test_print_two_numbers_task
```



```
TERMINAL    COMMENTS

Test run at 8/19/2024, 10:15:54 PM
  [cdl_signals_test_pipeline_dk]
  [dag1]
  [cdl_signals_test_pipeline_dk-uat]
  [cdl_signals_test_pipeline_fin]
  [cdl_signals_test_pipeline_fin-uat]
  [cdl_signals_test_pipeline_no-roamers]
  [cdl_signals_test_pipeline_no-uat]
  [cdl_signals_test_pipeline_swe-uat]
  [cdl_signals_test_pipeline_swe-v1]
  [cdl_signals_test_pipeline_swe-roamers]
  [cdl_signals_test_pipeline_dk-dk-dk]
  [cdl_signals_test_pipeline_dk-uat-dk-dk-uat]
  [cdl_signals_test_pipeline_fin-fin-fin]
  [cdl_signals_test_pipeline_fin-uat-fin-fin-uat]
  [cdl_signals_test_pipeline_no-no-no]
  [cdl_signals_test_pipeline_no-uat-no-no-uat]
  [cdl_signals_test_pipeline_swe-uat-swe-swe-uat]
  [cdl_signals_test_pipeline_swe-v1-swe-swe-v1]
  [cdl_signals_test_pipeline_dk-dk-dk]
  [cdl_signals_test_pipeline_dk-uat-dk-dk-uat]
  [cdl_signals_test_pipeline_fin-fin-fin]
  [cdl_signals_test_pipeline_fin-uat-fin-fin-uat]
  [cdl_signals_test_pipeline_no-no-no]
  [cdl_signals_test_pipeline_no-uat-no-no-uat]
  [cdl_signals_test_pipeline_swe-uat-swe-swe-uat]
  [cdl_signals_test_pipeline_swe-v1-swe-swe-v1]
```

automatically picks up tests in VS Code
debug inside VS Code, with breakpoints in your tests

demo it?

20 min

AMA

🥋 @judole on everything
code @ github.com/judoole/airflow-bdd/