AIRFLOW SUMMIT

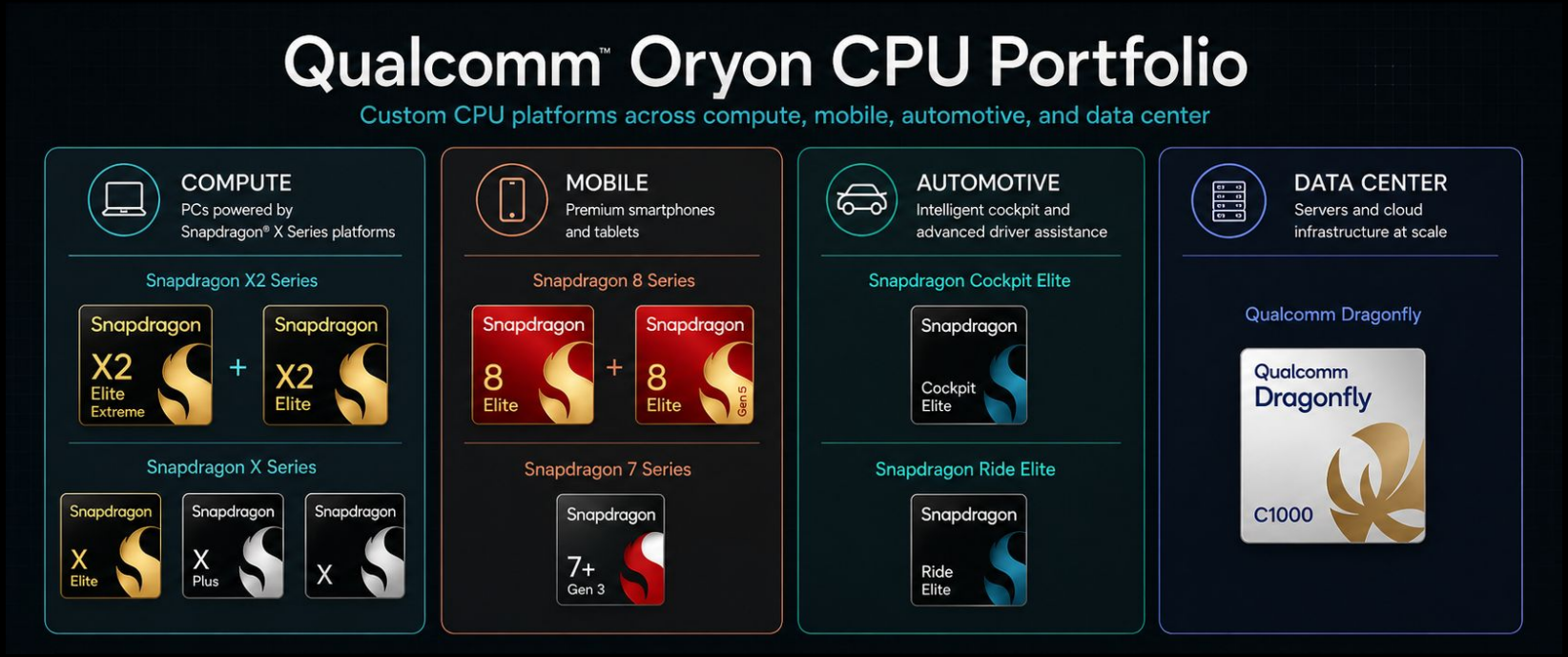
Airflow in a Box: Methodology or Madness?



Nicholas Redd

Senior CPU Engineer | DV Infrastructure & Methodology

Custom CPUs @ QCOM



Airflow @ QCOM

2,500+	4.5M+	Celery + Edge	7 days
DAGs in production	task instances	across sites + silicon	a single EDA task

Our tasks are not simple.

EDA workflows can demand 256 GB of RAM, run for days, and emit hundreds of gigabytes.

When a DAG is wrong, we don’t find out in minutes. We find out on Monday.

About Me



Georgia®

Caltech

Google AI Quantum

Qualcomm



Methodologies Makes Improvement Repeatable

A shared, repeatable approach for turning intent into reliable outcomes.

PRINCIPLES

What matters

→

PRACTICES

How people act

→

FEEDBACK

How the approach improves

Tools can encode a methodology. They are not the methodology.

Madness Is Improvement Without Feedback

You cannot improve what you cannot test

FAST FEEDBACK change → test locally → learn → next change

SLOW FEEDBACK change → deploy + wait → guess → avoid risk

For DAGs: no fast local environment means slow proof and fewer safe iterations.

Last Year

Semiconductor (Chip) Design Workflow Orchestration with Airflow



Pytest Plugin for Airflow

Unit testing an Airflow codebase:

- 'airflow dags list-import errors' is not enough!
- No official Pytest support ☹
- Custom Pytest plugin -> apache-airflow-devel-common -> pytest_plugin.py source code

```
@pytest.mark.db_test
@pytest.mark.smoke
@pytest.mark.execution_timeout(3000)
def test_integrity(dags_path):
    """
    Replaces `airflow dags list-import-errors`
    """
    dag_bag = DagBag(
        dag_folder=dags_path, include_examples=False, read_dags_from_db=False
    )
    assert not dag_bag.import_errors, "DAG import errors are present :("
    slowpokes = [
        s for s in dag_bag.dagbag_stats if s.duration > duration(seconds=10)
    ]
    assert not slowpokes, "Slowpokes are present :("
```

Last Year

```python
@pytest.mark.db_test
@pytest.mark.smoke
@pytest.mark.execution_timeout(3000)
def test_integrity(dags_path):
    """
    Replaces `airflow dags list-import-errors`
    """
    dag_bag = DagBag(
        dag_folder=dags_path, include_examples=False, read_dags_from_db=False
    )
    assert not dag_bag.import_errors, "DAG import errors are present :("
    slowpokes = [
        s for s in dag_bag.dagbag_stats if s.duration > duration(seconds=10)
    ]
    assert not slowpokes, "Slowpokes are present :("
```

MY ACTUAL ADVICE, LAST YEAR

airflow dags list-import-errors is not enough

No official pytest support ☹

Custom Pytest plugin →
apache-airflow-devel-common →
pytest_plugin.py source code

Today, that is the ONLY thing I am talking about.

pytest-airflow-in-a-box

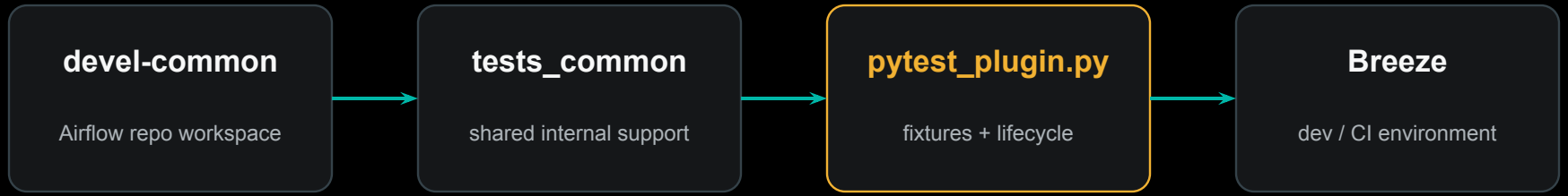




Already on PyPI

Scan to install

Origin Story



The useful behavior

bootstrap · metadata DB · Dag fixtures · cleanup

rebuilt as a supported, pip-installable pytest plugin

A Real DAG Test Proves State + XComs

```python
from pytest_airflow_in_a_box.matchers import succeeded

def test_my_dag(dag_bag, run_dag):
    result = run_dag(dag_bag.dags["my_dag_id"])

    assert result == {
        "extract": succeeded(),
        "load": succeeded(),
    }
    assert result.xcoms["load"] == 42
```

- real DagRun
- task states matched
- return XCom checked
- lazy until requested

One test proves both task states and returned data.

Test Airflow Behavior, Not Just Python Callables

AIRFLOW SUMMIT

A BRANCH FUNCTION PROVES

```python
assert choose() == "chosen"
```

Verifies `choose()` returned a string

A DAG RUN PROVES

```python
from pytest_airflow_in_a_box.matchers import skipped
result = dag_maker.run()
assert result.order == ["choose", "chosen"]
assert result["rejected"] == skipped()
```

Verify `rejected` was skipped

```python
from pytest_airflow_in_a_box.matchers import failed, succeeded, upstream_failed
assert result == {"produce": succeeded(21), "boom": failed(ValueError),
                  "consume": upstream_failed()}
```

One whole-run contract. One readable diff.

The Fidelity Ladder: Climb Only as Needed

Choose the lowest rung you need. Fixture or call first; state exercised second.



22 Typed Fixtures, Zero Conftest

RUN AND INSPECT	
dag_bag	repository DAGs
dag_corpus	whole-DAG corpus
dag_maker	author / persist / run
run_dag	execute a real DagRun
run_task	one task, DB-free
render_task	render template fields
task_context	seed task context
create_task_instance	ORM task instance
create_dummy_dag	one-call DAG factory
DATABASE AND ENVIRONMENT	
session	metadata session
airflow_variables	seed Variables
airflow_connections	seed Connections

DATABASE AND ENVIRONMENT	
airflow_parse_secrets	parse-time secrets
airflow_configure	runtime config
airflow_components	reversible registration
airflow_home	isolated home path
airflow_dags_folder	parsed DAG path
testing_dag_bundle	testing bundle row
API AND LOGGING	
api_server_url	live API URL
api_client	authenticated client
api_base_url	publish lazy URL
cap_structlog	capture Airflow events

Full Certified Airflow Matrix

AIRFLOW 3 • PYTHON 3.10–3.14

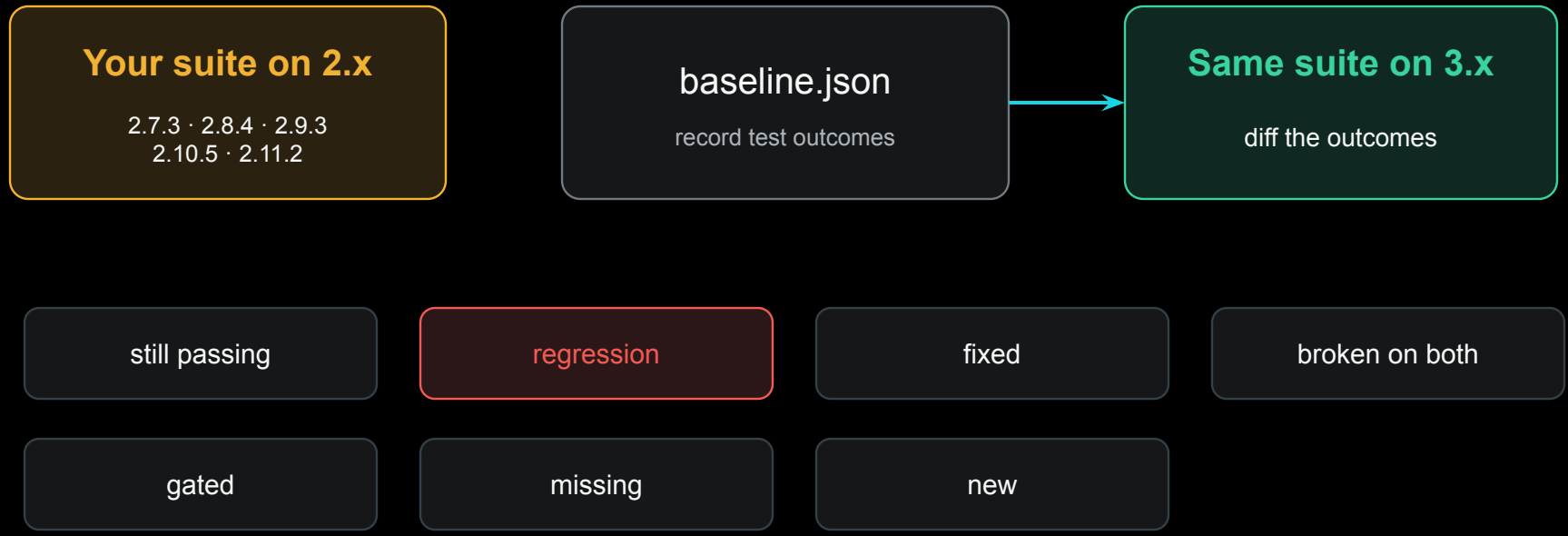
3.1.x	3.1.0 • 3.1.1 • 3.1.2 • 3.1.3 3.1.5 • 3.1.6 • 3.1.7 • 3.1.8
3.2.x	3.2.0 • 3.2.1 • 3.2.2
3.3.x	3.3.0 • 3.3.1

AIRFLOW 2 • MIGRATION TIER

2.7.3	≤ Python 3.11
2.8.4	≤ Python 3.11
2.9.3	≤ Python 3.12
2.10.5	≤ Python 3.12
2.11.2	≤ Python 3.12

Newer 3.x: live probes + warning • Unknown 2.x: fail closed

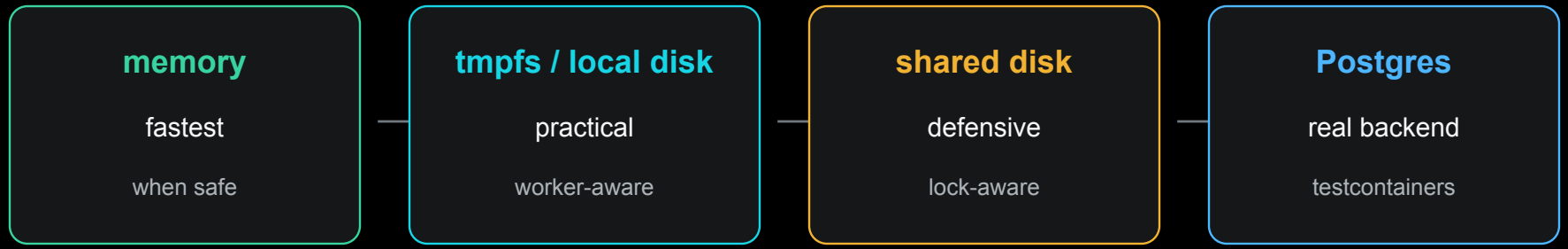
Airflow 2 Is a Migration Bridge



One CI provisioner can run both environments and produce the diff.

Tuning the SQLite Database

Every database-backed test gets real Airflow metadata.



SQLite tuned for tests WAL · foreign keys · busy timeout · safe pooling

If the environment cannot safely honor the requested mode, it fails instead of silently falling back.

Testing in Parallel with pytest-xdist

```
pytest -n auto --dist loadgroup
```

Use loadgroup when tests share Airflow-global names.

Per-worker resources

sessions, servers, reports

Bundle-name isolation

generated names avoid contention

Collision handling

seeded names fail visibly

Corpus locking

one producer, portable JSON

Serial-only reset

whole-DB reset stays out of workers

How _compat Finds the Backend You Have

1 CLASSIFY
package metadata

Airflow 2 vs 3
certified vs probed

→

2 PROBE
capabilities.py

symbols • signatures
model fields • locations

→

3 VERIFY + EXPOSE
immutable contract

certified row matches
fixtures consume one
object

DagBag location • task runner • API surface • executor contract

One stable capability object. No version branches scattered through fixtures.

Built-In Smoke Tests

```
pytest --airflow-smoke --dag-folder=dags/
```

12

generated checks in the bundled catalog

- import integrity
- serialization round-trip
- schedule sanity
- no top-level Variable access
- no top-level I/O

The same checks can operate across the whole DAG corpus.

A Real Airflow Test Environment



AIRFLOW DOCTOR

```
pytest --airflow-doctor --dag-folder=dags --cov=src
```

It can catch a proud 100% coverage report that never measured the DAG files.

Test Your Own Airflow Components

10 COMPONENT KINDS

timetable	listener
executor	XCom backend
weight strategy	notifier
secrets backend	policy
plugin	provider

```python
from pytest_airflow_in_a_box.components import check_component

check_component(MyTimetable).raise_for_problems()
```

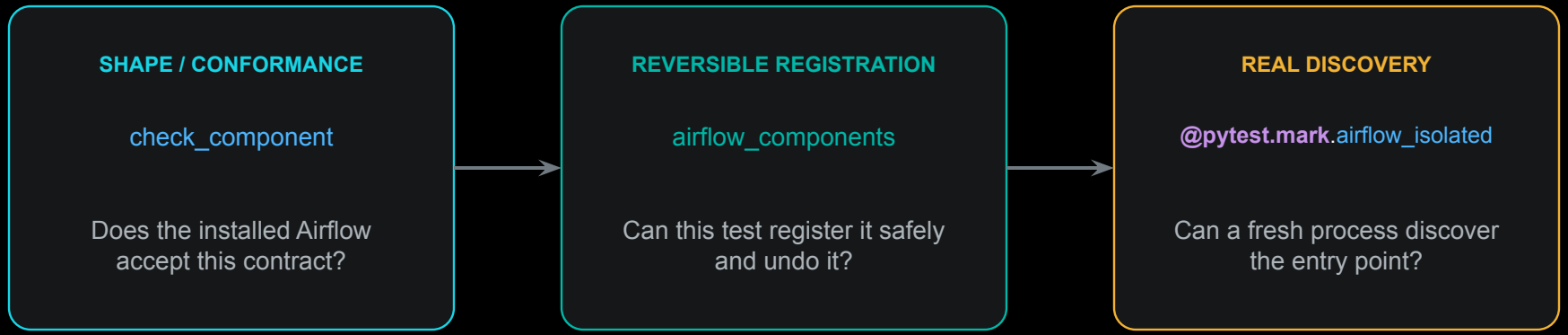
class · instance · provider callable

25

machine-readable problem codes

Never constructs the class.
Findings accumulate.
raise_for_problems() opts in.

Shape Is Not Registration



CI Without Dependency Surprises

AIRFLOW SUMMIT

GITHUB ACTION

```
uses: nredd/pytest-airflow-in-a-box/action@v0
```

- 8 inputs · 3 outputs
- constraints-pinned environment
- validates before install
- optional report artifacts

FOUR RUNTIME DEPENDENCIES

pytest	pytest-timeout
packaging	sqlalchemy

Airflow is deliberately not one of them.

BYOA (Bring Your Own Airflow)

Built with a lot of help from uv, ruff, and ty.

Certifying Version Support

24	4	2	weekly	100%
compatibility matrix legs	platforms	database backends	upstream canary	branch-coverage union

WHAT THE MATRIX COVERS

- Airflow 3.1–3.3 across Python 3.10–3.14
- glibc Linux · macOS · ARM Linux · Alpine/musl
- parallel loadgroup + one serial reference + real-Docker Postgres

A new Airflow release should tell the maintainer before it tells your users.

Experimenting with Upstream

THE STAND-IN STRATEGY

Remove upstream fixture registration.

Replace it with a marker-only shim.

Every missing fixture or behavior becomes countable.

2,131 Airflow tests
models · serialization · timetables

45%

Breeze tests pass without modification

+12 pts

after fixing concentrated gaps

326 → 173 errors

92 → 2 on Airflow 2.9.3

The compatibility gap moved down the stack. It did not vanish.

What it found

plugin bugs · operational feature · one Airflow regression

Try It Out!

```
uv add --dev "pytest-airflow-in-a-box[airflow3]"
pytest --airflow-smoke --dag-folder=dags
```



Docs nredd.github.io/pytest-airflow-in-a-box/

PyPI pypi.org/project/pytest-airflow-in-a-box/

GitHub Action github.com/nredd/pytest-airflow-in-a-box/action@v0

Install it. Point it at your DAG folder. Make the hard behavior easy to test.

Questions?



Personal + project links

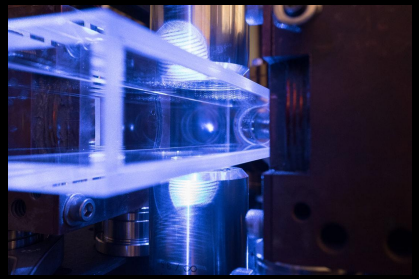
Nicholas Redd

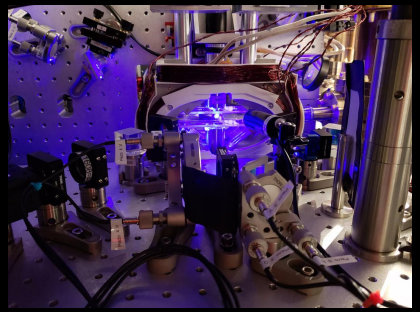
github.com/nredd
nicholaseredd@gmail.com
redd@qti.qualcomm.com

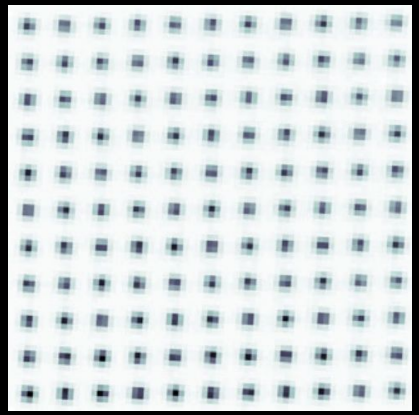


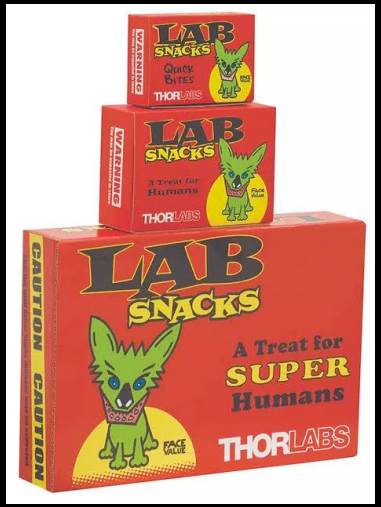
Qualcomm jobs

Appendix — Endres Lab at Caltech: Lasers









Appendix — Quantum Hardware at Google: Qubits



