



The Secret to Airflow's Evergreen Build

CI/CD magic

Amogh Desai, Jarek Potiuk, Pavan Kumar

3.0



Amogh Desai

Apache Airflow PMC member & committer
Senior Software Engineer at Astronomer



Jarek Potiuk

Apache Airflow PMC member & committer
Member of Apache Software Foundation
Member of ASF Security Committee



Pavan Kumar Gopidesu

Apache Airflow committer & Member The Apache Software Foundation
Technical Lead at Tavant

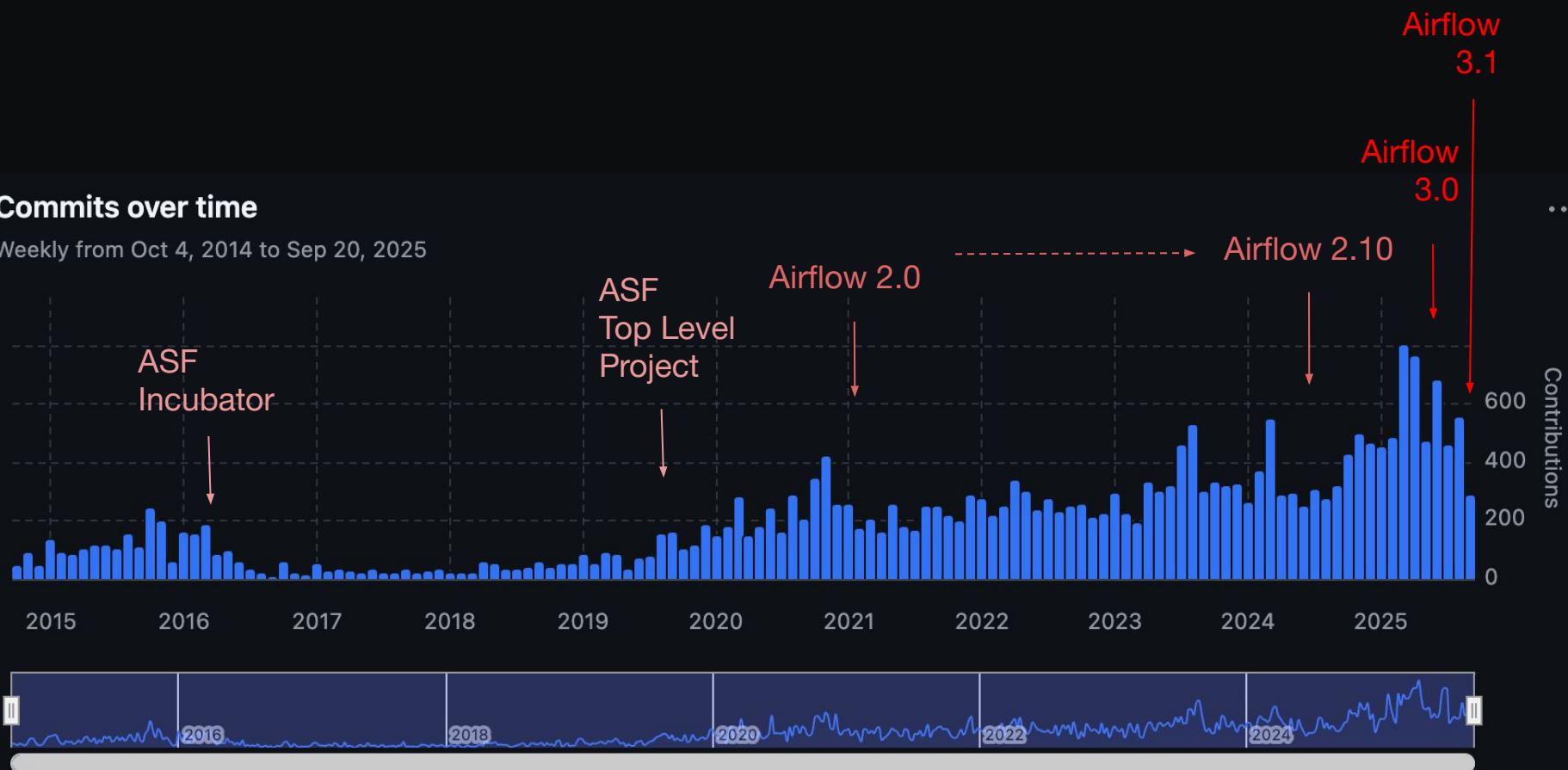
Agenda

- Why CI?
- Past
- Present
- Future

Why CI ?

Commits over time

Weekly from Oct 4, 2014 to Sep 20, 2025



September 3, 2025 – September 10, 2025

Period: 1 week ▾

Overview

219 Active pull requests

86 Active issues

160
Merged pull requests

59
Open pull requests

52
Closed issues

34
New issues

Summary

Excluding merges, **71 authors** have pushed **150 commits** to main and **212 commits** to all branches.

On main, **941 files** have changed and there have been **17,283 additions** and **6,226 deletions**

Top Committers



Past

More than 5 years of evolution

- Travis CI -> GitHub Actions
- Basic assumptions:
 - Use CI image (700+ dependencies)
 - Breeze commands as a driver
 - CI jobs reproducible locally
- Initially Bash
- Rewritten to Python via Outreachy Internship
- Switched to uv last year

Why so complex?

- Airflow unit tests (DB, non, DB, lowest deps)
- Provider unit tests (DB, non-DB, back-compatibility, lowest deps)
- Integration tests
- Docker image build tests
- Docker compose tests
- Helm unit tests
- Helm Chart + K8S tests + Docker compose tests
- Constraints generation
- ...

← Tests AMD

Tests AMD #16407

Re-run all jobs

Summary

Jobs

Build info

Summarize warnings

Run pin-versions hook

Basic tests

Breeze unit tests

Shared secrets_masker tests

Shared logging tests

Shared timezones tests

React UI tests

Check translation completeness

Static checks: basic checks only

Test git clone on Windows

Upgrade checks

Test Airflow release commands

Test Airflow standalone command

Build CI images

Build CI linux/amd64 image 3.10

Triggered via schedule 2 days ago

 potiuk -o_bff6c95 main

Status

Success

Total duration

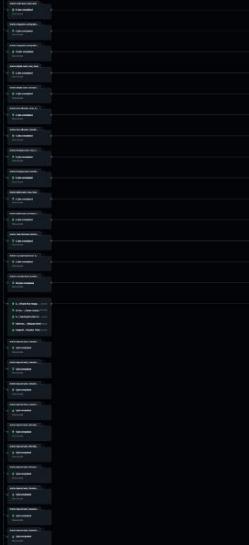
1h 17m 5s

Artifacts

108

ci-amd.yml

on: schedule



It's a kind of magic (it's all about people)

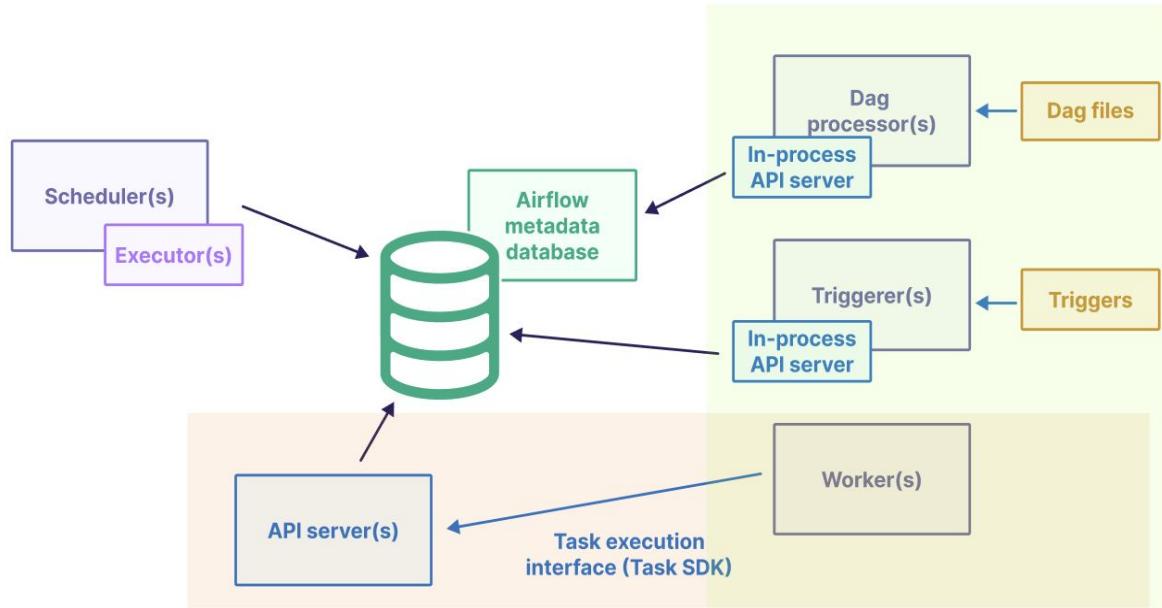
- Many, Many, Many moving parts and workflows
- Jarek as original author and (SPOF)
- Python rewrite with Outreachy interns (Bowna and Edith)
- More people involved over last year [#internal-airflow-ci-cd](#)
 - Amogh, Pavan, Jens, Bugra, Aritra, Elad, Ash, Kaxil, Wei Lee
 - (many more)

Present

We didn't just improve CI/CD for fun, but we
did it to prepare for **Airflow 3!**

Story Time: The Big Picture

Airflow 3 Architecture



Providers – Now to consume from **task SDK** instead of core

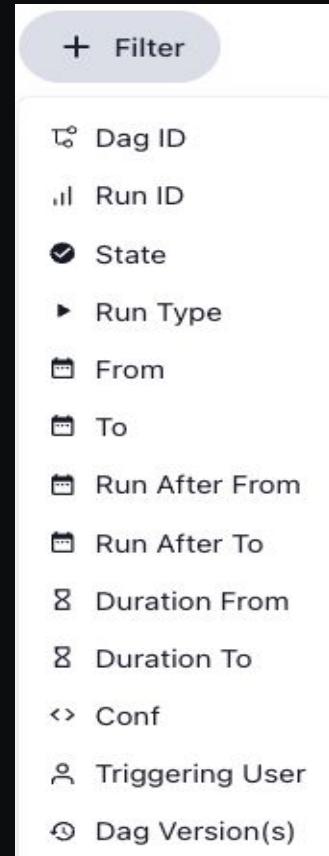
Airflowctl – **kubectl** but for airflow

Shared Library – Common code (timezone, secrets masker, logging)

Community Driven Beginning

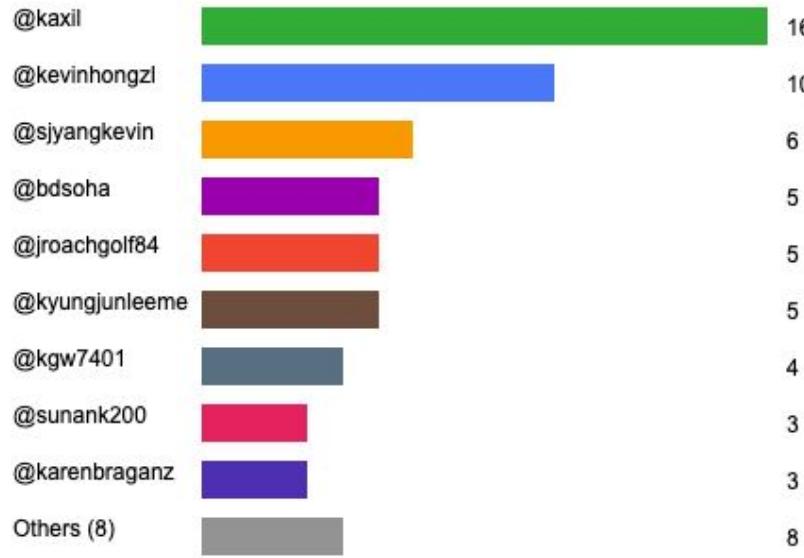
Big Goals need a Strong Community!

- MyPy Upgrade (1.9.0 => 1.16.1)
 - Cleaning up `mypy ignores`
- Update providers to use `BaseHook` & `BaseOperator` from task SDK
- Improving UI / API `filtering` options

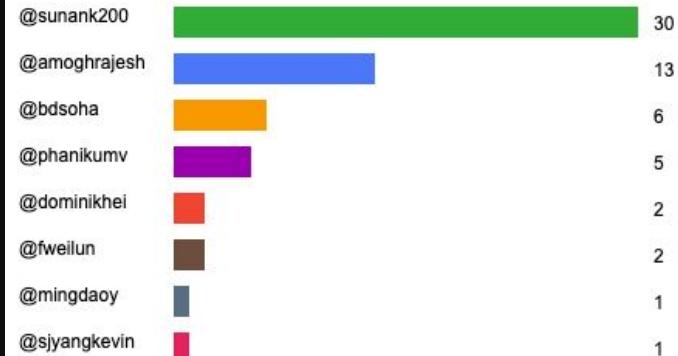


Community Stats

BaseOperator Migration - 17 contributors, 100+ providers



BaseHook Refactoring - 60+ PRs from 8 contributors



MyPy Upgrade Initiative - 30+ contributors



The Challenge

Tons of PRs, lot of active contributors, massive architectural change...

How do we know what's **failing**?

Scheduled Runs

- CI-AMD: 28 1,7,13,19 * * *
- CI-ARM: 28 3,9,15,21 * * *

Tests AMD
ci-amd.yml

286 workflow run results

This workflow has a `workflow_dispatch` event trigger.

Run workflow ▾

| | | | |
|-----------------------------|------|--------------------------|-----|
| ✓ Tests AMD | main | Sep 25, 1:01 PM GMT+5:30 | ... |
| Tests AMD #16834: Scheduled | | ⌚ 1h 23m 18s | |
| ✓ Tests AMD | main | Sep 25, 1:00 AM GMT+5:30 | ... |
| Tests AMD #16792: Scheduled | | ⌚ 45m 9s | |
| ✓ Tests AMD | main | Sep 24, 7:06 PM GMT+5:30 | ... |
| Tests AMD #16730: Scheduled | | ⌚ 1h 27m 46s | |
| ✓ Tests AMD | main | Sep 21, 1:00 AM GMT+5:30 | ... |
| Tests AMD #16407: Scheduled | | ⌚ 1h 17m 5s | |
| ✓ Tests AMD | main | Sep 20, 7:02 PM GMT+5:30 | ... |
| Tests AMD #16394: Scheduled | | ⌚ 1h 18m 10s | |

Slack Bot for Failures

- Runs 2 times a day: 6 AM and 5 PM UTC
- Notifies on Slack:
#internal-airflow-ci-cd
- Started with: Main branch only
- Realized: Need release branch too



Example Improvements

GitHub Rate Limit Crisis

A dark-themed GitHub blog post card featuring a purple cat silhouette. The card has a black header bar with the GitHub logo and a white main area. At the top left, there's a small 'IMPROVEMENT' badge, followed by the date 'MAY 8, 2025 • 1 MINUTE READ'. The main title 'Updated rate limits for unauthenticated requests' is displayed prominently in white. Below the title is a large, semi-transparent purple cat silhouette.

IMPROVEMENT

MAY 8, 2025 • 1 MINUTE READ

Updated rate limits for unauthenticated requests

May 2025:

- 60 requests / hr for unauthenticated requests
- Many workflows use [Github](#) for various things
- Random Failures

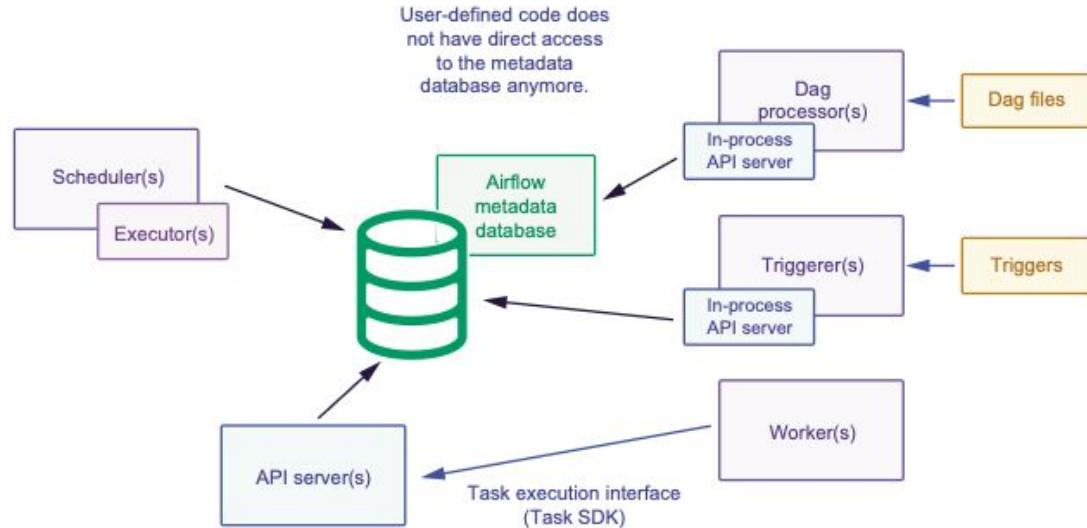
GitHub Rate Limit Crisis

Workflow scripts:

- `get-devel_deps`, `entrypoint_ci` ...
- 20+ scripts using raw.githubusercontent.com
- Switch to [**Github API**](https://api.github.com)
- Systematic Fix across 20+ files / as needed

Recap of Airflow 3 Architecture

Airflow 3 architecture



- Task SDK has an API client to talk to API server
- API Server hosts execution API

Hidden Problem

```
def test_xcom_api_server():
    """Test XCom storage and retrieval through the API server."""
    # ... setup code for test instance, client, session ...

    # Serialize the value to simulate the client SDK
    value = serialize(orig_value)

    # Test that the value is serialized correctly
    assert value == ser_value

    # Store XCom value via API
    response = client.post(
        f"/execution/xcoms/{ti.dag_id}/{ti.run_id}/{ti.task_id}/xcom_1",
        json=value,
    )

    assert response.status_code == 201

    # Retrieve stored value from database
    stored_value = session.execute(
        XComModel.get_many(
            key="xcom_1",
            dag_ids=ti.dag_id,
            task_ids=ti.task_id,
            run_id=ti.run_id,
        ).with_only_columns(XComModel.value)
    ).first()

    # Deserialize and verify the stored value matches expected
    deserialized_value = XComModel.deserialize_value(stored_value)
    assert deserialized_value == deser_value
```

```
def test_xcom_pull_worker():
    """Test XCom pull functionality in worker context."""
    # ... setup code for spy_agency, mock_supervisor_comms ...

    spy_agency.spy_on(deserialize)

    class CustomOperator(BaseOperator):
        def execute(self, context):
            value = context["ti"].xcom_pull(key="key")
            print(f"Pulled XCom Value: {value}")

    test_task_id = "pull_task"
    task = CustomOperator(task_id=test_task_id)

    # Create runtime task instance with map_index if needed
    extra_for_ti = {"map_index": map_indexes} if map_indexes in (1, None) else {}
    runtime_ti = create_runtime_ti(task=task, **extra_for_ti)

    # Serialize expected XCom value
    ser_value = BaseXCom.serialize_value(xcom_values)

    def mock_send_side_effect(*args, **kwargs):
        msg = kwargs.get("msg") or args[0]
        if isinstance(msg, GetXComSequenceSlice):
            return XComSequenceSliceResult(root=[ser_value])
        return XComResult(key="key", value=ser_value)

    mock_supervisor_comms.send.side_effect = mock_send_side_effect

    # Execute task and verify XCom deserialization was called
    run(runtime_ti, context=runtime_ti.get_template_context(), log=mock.MagicMock())
    spy_agency.assert_spy_called_with(deserialize, ser_value)
```

Hidden Problem

```
def test_xcom_api_server():
    """Test XCom storage and retrieval through the API server."""
    # ... setup code for test instance, client, session ...

    # Serialize the value to simulate the client SDK
    value = serialize(orig_value)

    # Test that the value is serialized correctly
    assert value == ser_value

    # Store XCom value via API
    response = client.post(
        f"/execution/xcoms/{ti.dag_id}/{ti.run_id}/{ti.task_id}/xcom_1",
        json=value,
    )

    assert response.status_code == 201

    # Retrieve stored value from database
    stored_value = session.execute(
        XComModel.get_many(
            key="xcom_1",
            dag_ids=ti.dag_id,
            task_ids=ti.task_id,
            run_id=ti.run_id,
        ).with_only_columns(XComModel.value)
    ).first()

    # Deserialize and verify the stored value matches expected
    deserialized_value = XComModel.deserialize_value(stored_value)
    assert deserialized_value == deser_value
```

```
def test_xcom_pull_worker():
    """Test XCom pull functionality in worker context."""
    # ... setup code for spy_agency, mock_supervisor_comms ...

    spy_agency.spy_on(deserialize)

    class CustomOperator(BaseOperator):
        def execute(self, context):
            value = context["ti"].xcom_pull(key="key")
            print(f"Pulled XCom Value: {value}")

    test_task_id = "pull_task"
    task = CustomOperator(task_id=test_task_id)

    # Create runtime task instance with map_index if needed
    extra_for_ti = {"map_index": map_indexes} if map_indexes in (1, None) else {}
    runtime_ti = create_runtime_ti(task=task, **extra_for_ti)

    # Serialize expected XCom value
    ser_value = BaseXCom.serialize_value(xcom_values)

    def mock_send_side_effect(*args, **kwargs):
        msg = kwargs.get("msg") or args[0]
        if isinstance(msg, GetXComSequenceSlice):
            return XComSequenceSliceResult(root=[ser_value])
        return XComResult(key="key", value=ser_value)

    mock_supervisor_comms.send.side_effect = mock_send_side_effect

    # Execute task and verify XCom deserialization was called
    run(runtime_ti, context=runtime_ti.get_template_context(), log=mock.MagicMock())
    spy_agency.assert_spy_called_with(deserialize, ser_value)
```

Hidden Problem

```
def test_xcom_api_server():
    """Test XCom storage and retrieval through the API server."""
    # ... setup code for test instance, client, session ...

    # Serialize the value to simulate the client SDK
    value = serialize(orig_value)

    # Test that the value is serialized correctly
    assert value == ser_value

    # Store XCom value via API
    response = client.post(
        f"/execution/xcoms/{ti.dag_id}/{ti.run_id}/{ti.task_id}/xcom_1",
        json=value,
    )

    assert response.status_code == 201

    # Retrieve stored value from database
    stored_value = session.execute(
        XComModel.get_many(
            key="xcom_1",
            dag_ids=ti.dag_id,
            task_ids=ti.task_id,
            run_id=ti.run_id,
        ).with_only_columns(XComModel.value)
    ).first()

    # Deserialize and verify the stored value matches expected
    deserialized_value = XComModel.deserialize_value(stored_value)
    assert deserialized_value == deser_value
```

```
def test_xcom_pull_worker():
    """Test XCom pull functionality in worker context."""
    # ... setup code for spy_agency, mock_supervisor_comms ...

    spy_agency.spy_on(deserialize)

    class CustomOperator(BaseOperator):
        def execute(self, context):
            value = context["ti"].xcom_pull(key="key")
            print(f"Pulled XCom Value: {value}")

    test_task_id = "pull_task"
    task = CustomOperator(task_id=test_task_id)

    # Create runtime task instance with map_index if needed
    extra_for_ti = {"map_index": map_indexes} if map_indexes in (1, None) else {}
    runtime_ti = create_runtime_ti(task=task, **extra_for_ti)

    # Serialize expected XCom value
    ser_value = BaseXCom.serialize_value(xcom_values)

    def mock_send_side_effect(*args, **kwargs):
        msg = kwargs.get("msg") or args[0]
        if isinstance(msg, GetXComSequenceSlice):
            return XComSequenceSliceResult(root=[ser_value])
        return XComResult(key="key", value=ser_value)

    mock_supervisor_comms.send.side_effect = mock_send_side_effect

    # Execute task and verify XCom deserialization was called
    run(runtime_ti, context=runtime_ti.get_template_context(), log=mock.MagicMock())
    spy_agency.assert_spy_called_with(deserialize, ser_value)
```

Hidden Problem

Deserialize response of `get_all` when we call `XCom.get_all` #53020

Merged amoghrajesh merged 5 commits into apache:main from astronomer:quick-fix-for-xcom on Jul 10

Conversation 11 Commits 5 Checks 75 Files changed 3 +65 -6

amoghrajesh commented on Jul 8

closes: [#53018](#)

The data returned from the api comes in without deserialisation as the contract between task sdk and the api server is simple: json objects.

So the returned data must be deserialised.

Reviewers: kaxil, vatsrahul1001, ashb

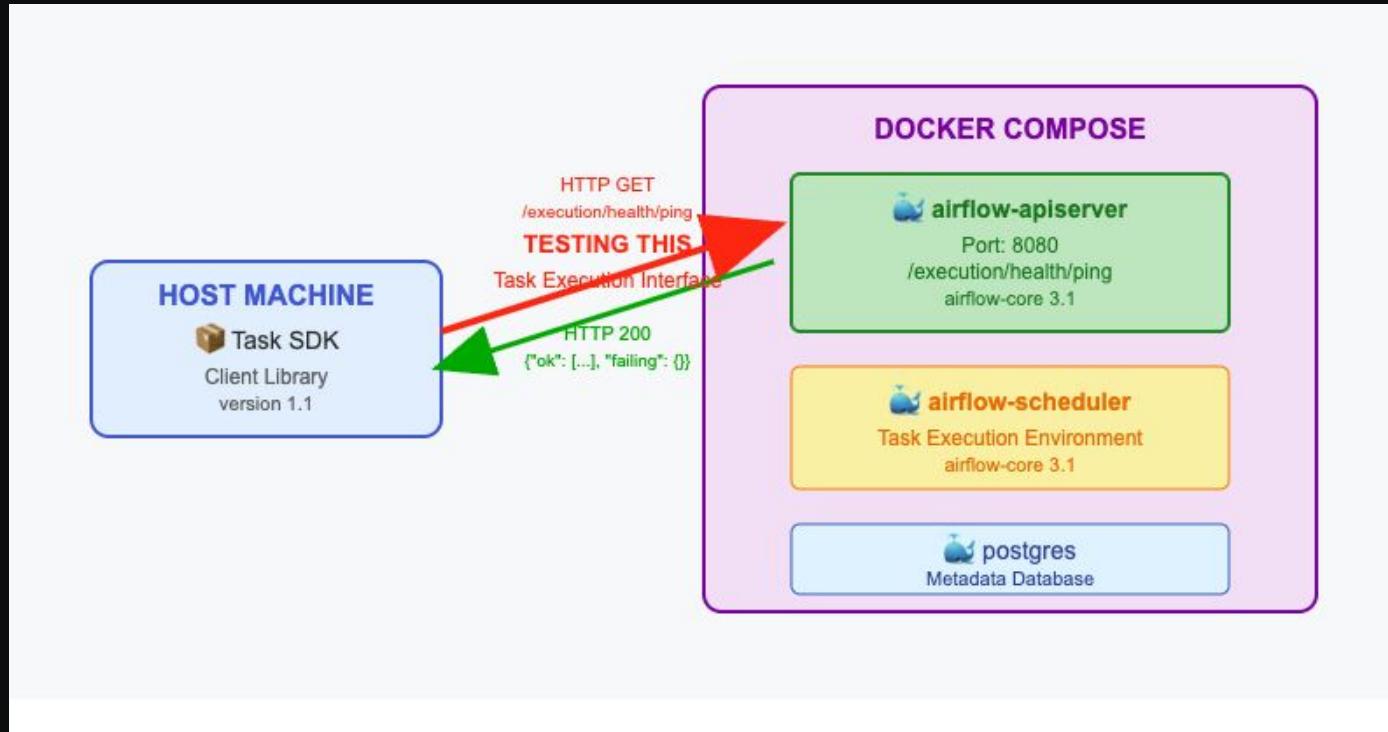
Assignees:

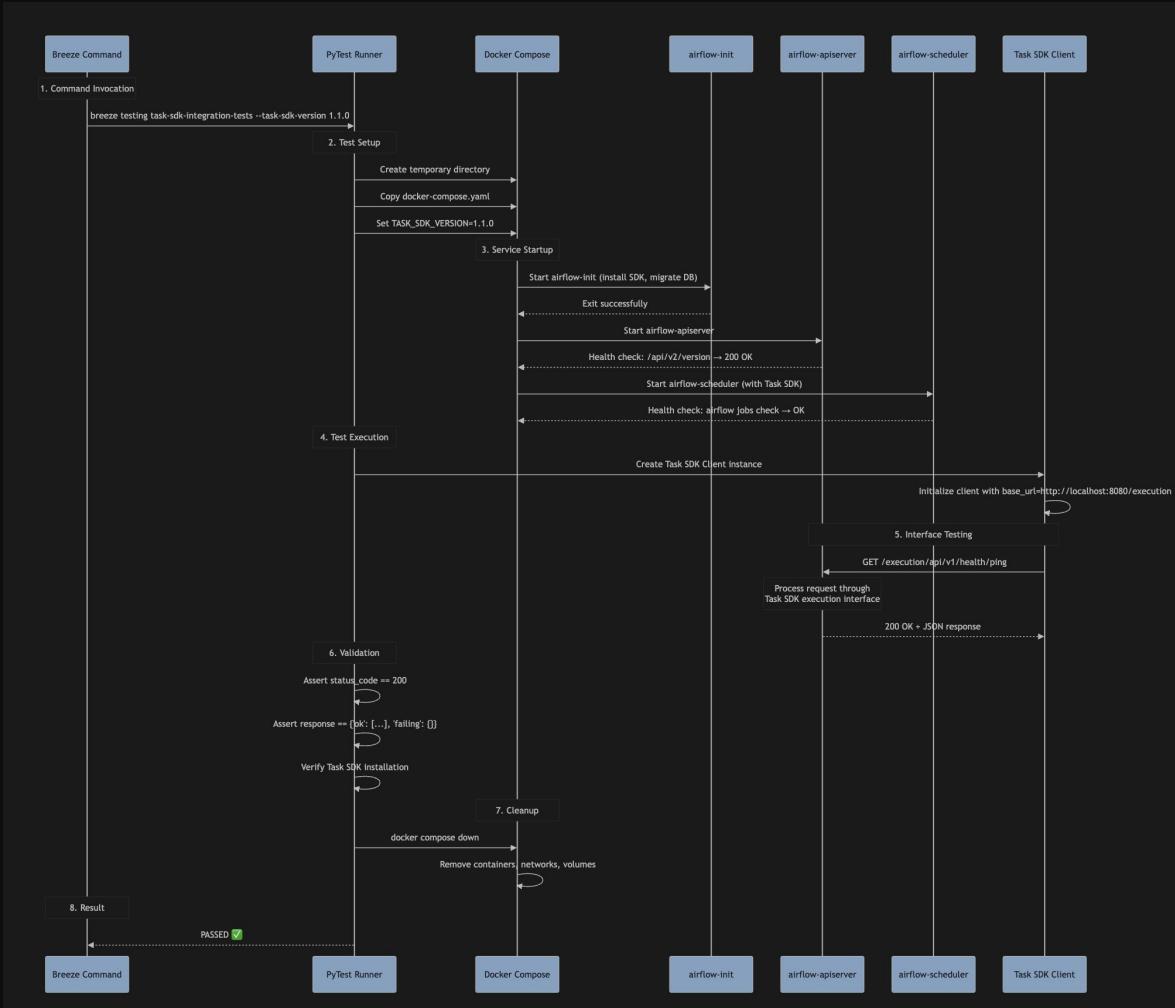
Bug came in during 3.0.3rc4

Unit test vs. Integration test



Task SDK Integration Testing



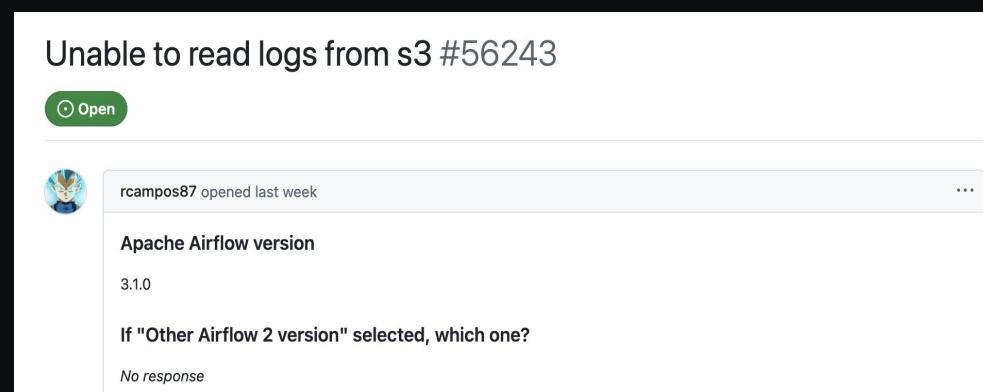


Making Processes Robust

Making Processes Robust

Beyond Unit Tests: E2E DAG Testing In CI

- Does all integrations between airflow components working?
- Is example dags working?
- Remote logging broken?



Test execution In CI

Additional PROD image tests / Test e2e integration tests with PROD image / Test e2e integration tests with PROD image

succeeded 5 hours ago in 10m 56s

Search log

Test e2e integration tests

```
260 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_branch_labels] PASSED
261 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_branch_operator] PASSED
262 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_branch_python_operator_decorator] PASSED
263 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_complex] PASSED
264 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_custom_weight] PASSED
265 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_dag_decorator] PASSED
266 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_dynamic_task_mapping] PASSED
267 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_dynamic_task_mapping_with_no_taskflow_operators] PASSED
268 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_external_task_marker_parent] PASSED
269 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_nested_branch_dag] PASSED
270 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_sensor_decorator] PASSED
271 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_setup_teardown] PASSED
272 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_setup_teardown_taskflow] PASSED
273 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_short_circuit_decorator] PASSED
274 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_short_circuit_operator] PASSED
275 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_simplest_dag] PASSED
276 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_skip_dag] PASSED
277 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_task_group] PASSED
278 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_task_group_decorator] PASSED
279 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_task_mapping_second_order] PASSED
280 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_time_delta_sensor_asyncl] PASSED
281 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_trigger_controller_dag] PASSED
282 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_trigger_target_dag] PASSED
283 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_weekday_branch_operator] PASSED
284 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_workday_timetable] PASSED
285 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_xcom] PASSED
286 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_xcom_args] PASSED
287 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[example_xcom_args_with_operators] PASSED
288 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[latest_only] PASSED
289 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[latest_only_with_trigger] PASSED
290 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[tutorial] PASSED
291 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[tutorial_dag] PASSED
292 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[tutorial_taskflow_api] PASSED
293 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[tutorial_taskflow_api_virtualenv] PASSED
294 tests/airflow_e2e_tests/basic_tests/test_example_dags.py::TestExampleDags::test_example_dags[tutorial_taskflow_templates] PASSED
295 =====
296 TEST EXECUTION SUMMARY
297 =====
298 Total Tests: 44
299 Passed: 44
300 Failed: 0
301 Execution Time: 387.99s
302 Reports generated: test_report.json
```

Automated Cherry-Picking: Seamless PR Propagation Across Branches

1. Propagate changes to branches seamlessly
2. Over 300 + developers working across different components
3. Enhance team productivity

Automated Cherry-Picking: Seamless PR Propagation Across Branches

0 Open ✓ 814 Closed

Author ▾ Label ▾ Projects ▾ Milest...

0 Bump axios from 1.11.0 to 1.12.0 in /airflow-core/src/airflow/ui ✓ area:UI backport-to-v3-0-test dependencies javascript #55551 by dependabot (bot) was merged 2 weeks ago • Approved ➔ Airflow 3.1.0

0 Remove config loader from react tests ✓ area:UI backport-to-v3-0-test #55541 by bbovenzi was merged 2 weeks ago • Approved ➔ Airflow 3.1.0

github-actions (bot) commented 2 weeks ago

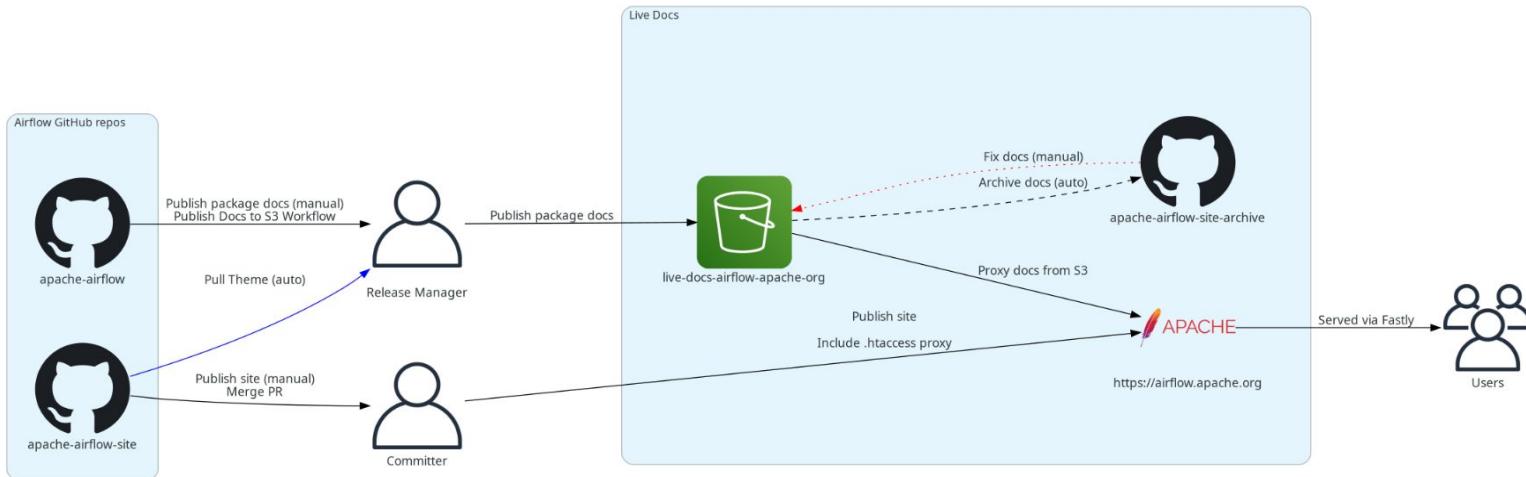
Backport successfully created: v3-0-test

| Status | Branch | Result |
|--------|-----------|----------|
| ✓ | v3-0-test | PR 55544 |

From Manual to Automated: Documentation Publishing Journey

1. Apache Airflow site 20+ GB
2. A lot of manual efforts for release managers to publish docs
3. Contributions to Airflow site is minimal
4. Hard to patch adhoc documentation changes

Current docs publish process



Some more things

Significant refactors / changes done over the year

- Securing GitHub Actions -> Get rid of “pull request target”
- Switching builds to uv
- Implementing uv workspace support
- Switching to prek as pre-commit runner
- Dependency versions check

Future

What's next?

- **UV lock** support for constraints generation
- Considering splitting **Providers** - and maybe more - repos
- Apache Trusted Releases Platform
- Trusted Publishing to **PyPI**
- e2e tests for UI



Summary

- CI is needed!
- People are most important
- CI needs you!

Questions?