RUCIO
SCIENTIFIC DATA MANAGEMENT

ATLAS
EXPERIMENT

Testing Software in Rucio

- **Concepts**
- **Tools**
- **Continuity**

Concepts

Testing is part of software development

Gains

- Less errors in production
- The code is much better comprehensible
  - On-boarding of new developers is easier
- The code is more robust
  - Refactoring is not as scary when there are tests
- Developers think about modularity
  - For better testability of modules

Tools

- Continuous integration
  - GitHub Actions #3669 (switched from Travis CI)
- Test automation tools
  - Pytest #3906 (switched from nosetests)
- Static code analysis
  - mypy for Python (using PEP 484 – Type Hints)
    - Planned until the next Rucio release 1.26

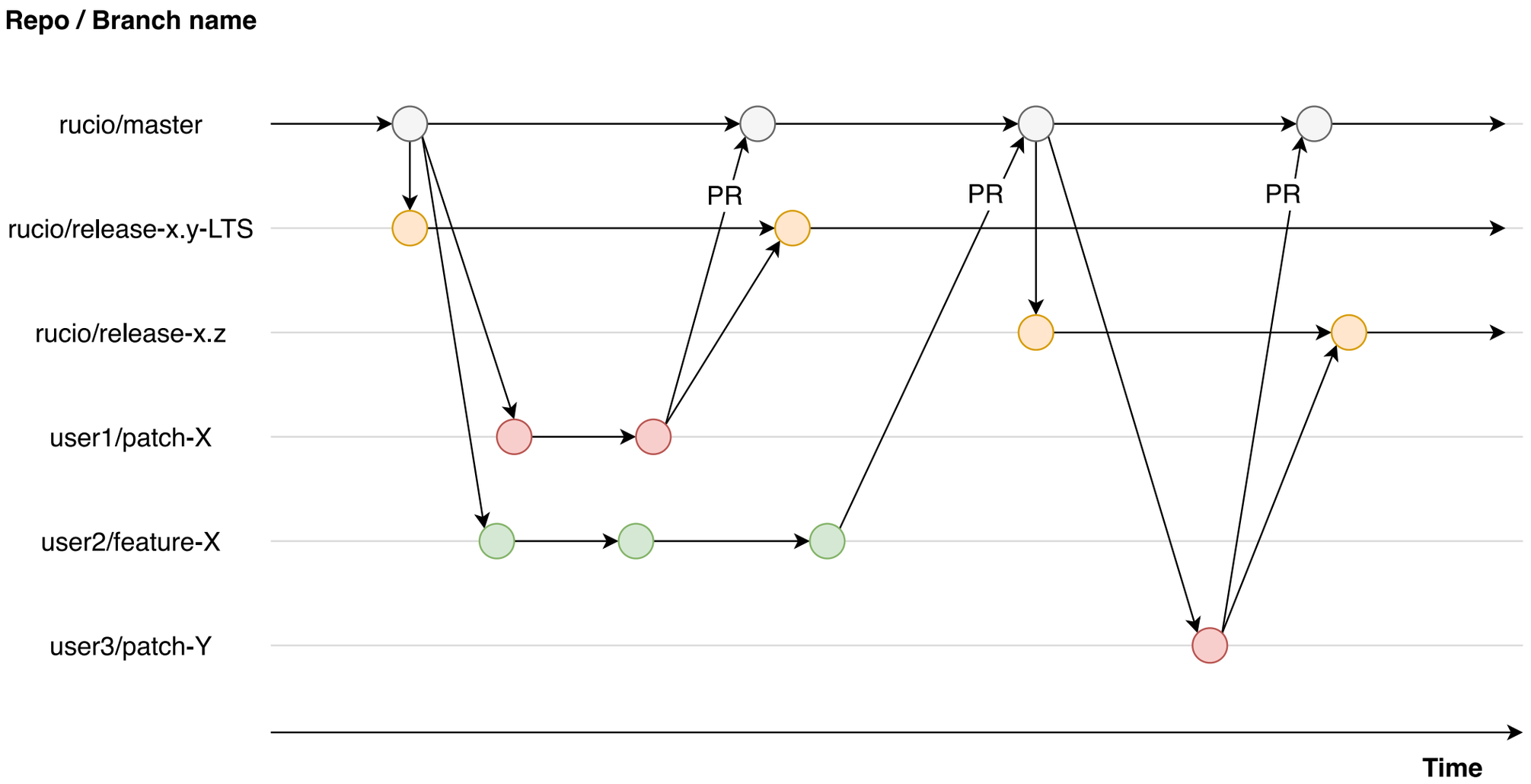
GitHub Actions

- Flexibility for GitHub projects
  - events like labeling an issue or a timer can trigger a workflow
  - cache autotest container images and boost our test time (saved time varies based on image cache usage, but roughly 3 to 4 minutes)
- On Travis a run took ~45 min and on GitHub Actions only ~15 min
  - 20 parallel jobs on GitHub Actions vs. 5 parallel jobs on Travis
  - we have seen over 100 concurrently running jobs on GitHub Actions at times

GitHub Actions (cont'd)

- Workflows as YAML files under .github/workflows
- Matrix testing in the CI and locally on a dev machine
  - Running the matrix tests locally when we were using Travis was not possible
  - Based on container technology (docker or podman)
  - Supporting test runs for multiple Python/database combinations
  - Currently 53 test jobs with testing against the latest release branch (27 otherwise)

Git Branching Strategy for Rucio



pytest

- Nosetests no longer in active development
  - pytest was suggested by nose as one alternative framework
- Plugin infrastructure
  - parallel testing with pytest-xdist
    - reduces the overall time of `` `tools/run_tests_docker.sh -a` `` roughly from 6 to 4 minutes on my local machine with 8 CPU threads
    - multivo suite on Actions is roughly down from 23 to 18 minutes
  - Our own plugin to enable noparallel tests and randomization of parallel test order

Concepts of pytest

- Simplicity

```
def inc(x):
    return x + 1


def test_answer():
    assert inc(3) == 5
```

- Detailed introspection

```
_____________________ test_answer _____________________

    def test_answer():
>       assert inc(3) == 5
E       assert 4 == 5
E        +  where 4 = inc(3)

test_sample.py:6: AssertionError
============= short test summary info =============
FAILED test_sample.py::test_answer - assert 4 == 5
```

Type Hints – PEP 484

- In Python 3

```
def get_transfertools(rse_id: str) -> "Set[str]":
```

https://mypy.readthedocs.io/en/stable/cheat_sheet_py3.html

- In Python 2 (for Rucio clients/common code)

```
def get_transfertools(rse_id):
    # type: (str) -> Set[str]
```

https://mypy.readthedocs.io/en/stable/cheat_sheet.html

Mypy static type checker

- Works with type hints
- IDEs run similar type checking
  - PyCharm and mypy use official type hints
    https://github.com/python/typeshed
- Is a project under the official python organisation on GitHub:
  https://github.com/python/mypy
- Should be integrated in the CI and can be run locally
  - Would just need a pip install or your distro's mypy package
  - Create a `` `tools/check_mypy.sh` ``

Summary

- Continuous Integration is a key component of testing software
  - The CI needs availability, extensibility and flexibility
- The ease of use and automation of tools like the CI or pytest help developers
- Allowing the local run of the same tests enables quicker switching of a CI framework

RUCIO
SCIENTIFIC DATA MANAGEMENT

ATLAS EXPERIMENT

Thanks for listening

Questions?

Links

- GitHub Actions Documentation
  https://help.github.com/en/actions
- Pytest Documentation
  https://docs.pytest.org/en/stable/index.html
- Mypy static type checker website
  http://mypy-lang.org/

all links retrieved 2021-03-22