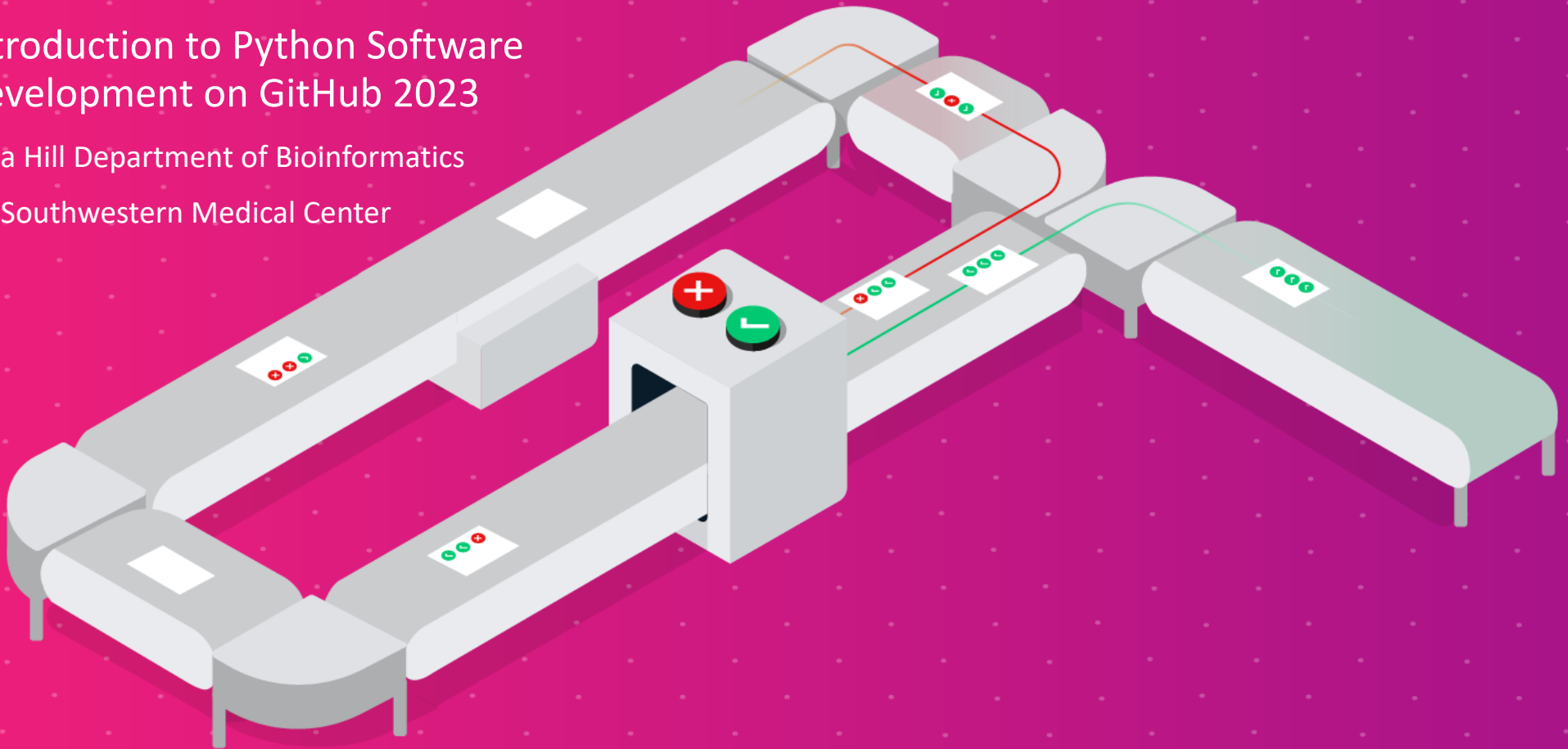


CI/CD Fundamentals

Introduction to Python Software
Development on GitHub 2023

Lyda Hill Department of Bioinformatics

UT Southwestern Medical Center



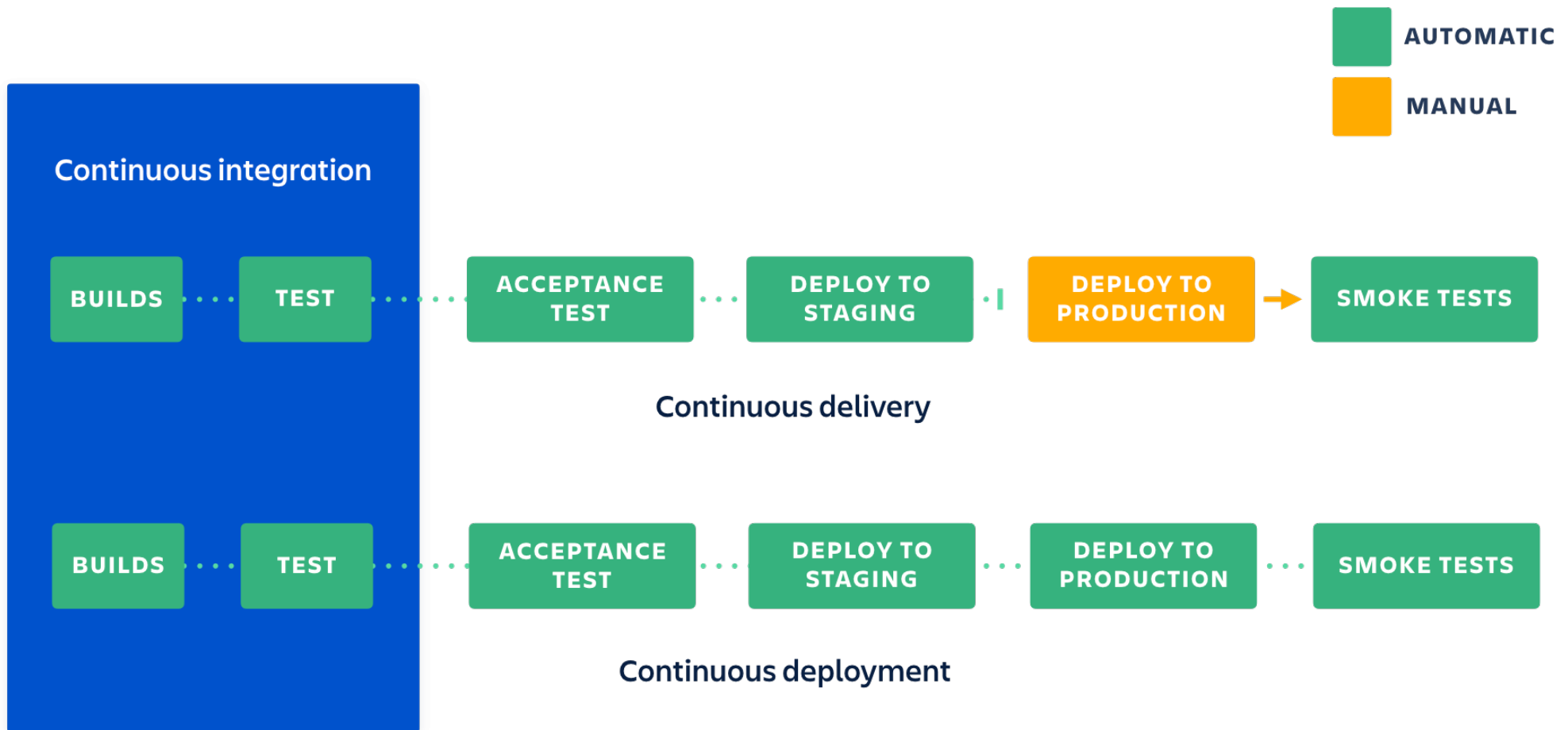
<https://about.codecov.io/blog/common-steps-in-a-complete-continuous-integration-workflow/>

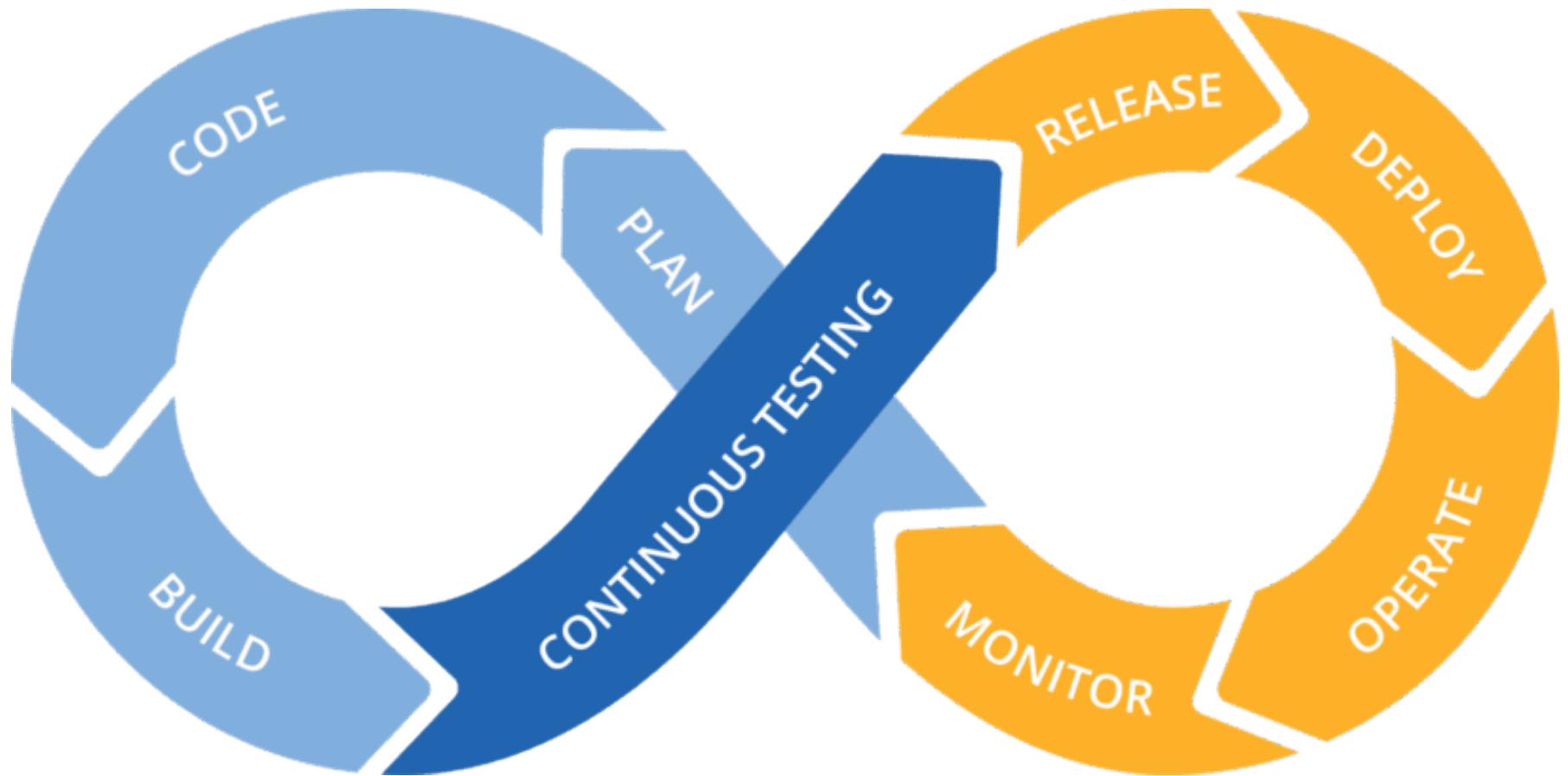
UT Southwestern
Medical Center

UTSW
Department of
Bioinformatics

What does CI/CD stand for?

- Continuous Integration
- Continuous Delivery
- Continuous Deployment





<https://deploybot.com/blog/continuous-development>

UT Southwestern
Medical Center



Vocabulary

- **Build**

Vocabulary

- **Build** – Convert source code files into a standalone software that anyone can run on their machine.

Vocabulary

- **Build** – Convert source code files into a standalone software that anyone can run on their machine.
- **Test**

Vocabulary

- **Build** – Convert source code files into a standalone software that anyone can run on their machine.
- **Test** – Evaluate and verify software can do what it is supposed to do.

How do we know this function works?

```
1  ✓ def square(x):  
2      """ Return the square of a number. """  
3      return x*x  
4
```

We check it with a unit test

```
1  ✓ def square(x):  
2      """ Return the square of a number. """  
3      return x*x  
4  
5  ✓ def test_square():  
6      """ Ensure our square() function works. """  
7      assert square(2) == 4  
8
```

Test subgroups

- **Acceptance test** – A group of unit tests that ensure the software meets specifications e.g. of a contract.
- **Smoke test** – A group of unit tests that act as a sanity check for severe failures. If you run the software, does smoke come out of the computer?

Vocabulary

- **Build** – Convert source code files into a standalone software that anyone can run on their machine.
- **Test** – Evaluate and verify software can do what it is supposed to do.
- **Release**

Vocabulary

- **Build** – Convert source code files into a standalone software that anyone can run on their machine.
- **Test** – Evaluate and verify software can do what it is supposed to do.
- **Release** – A build that is a new or upgraded version of the software.

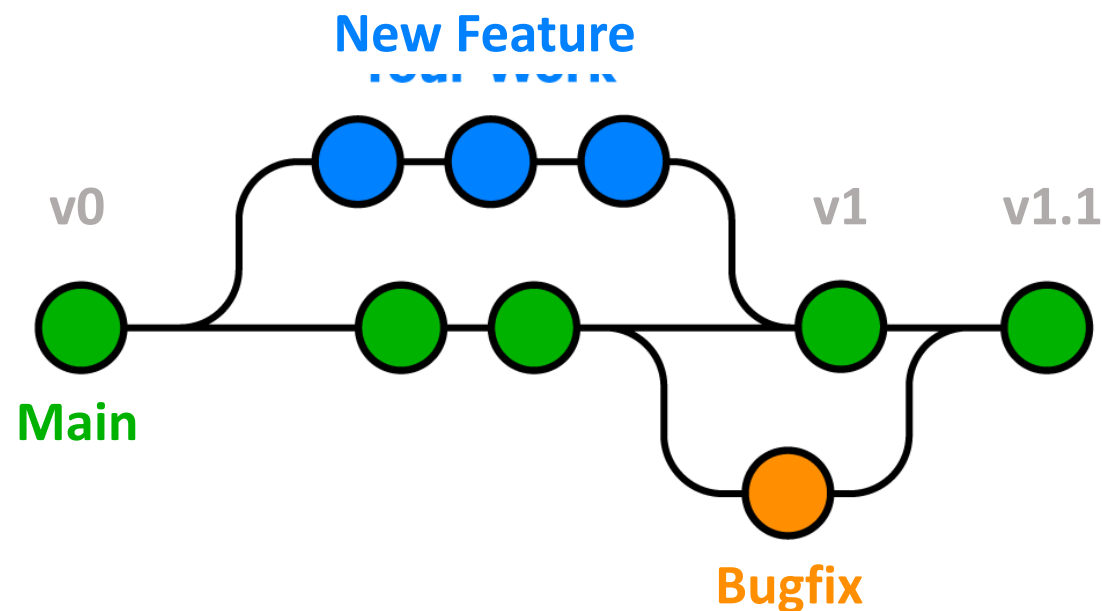
Vocabulary

- **Build** – Convert source code files into a standalone software that anyone can run on their machine.
- **Test** – Evaluate and verify software can do what it is supposed to do.
- **Release** – A build that is a new or upgraded version of the software.
- **Deploy**

Vocabulary

- **Build** – Convert source code files into a standalone software that anyone can run on their machine.
- **Test** – Evaluate and verify software can do what it is supposed to do.
- **Release** – A build that is a new or upgraded version of the software.
- **Deploy** – Make the software available for use.

The goal of CI/CD is to ensure developments do not stray far from the main branch



Why is CI/CD useful?

- Ensures disparate parts of the code base work together throughout development, preventing integration challenges.
- Protects against release of broken software.
- Allows for fast feedback from users and fast fixes from developers.

How do we implement CI/CD in practice?

- **Version control** (`git`)
- **Automatic testing** (`pytest`)
- **Automatic building** (`setuptools`, `pyproject.toml`)
- **Automatic deployment** (`twine`, `PyPI`)

Additional CI/CD tools in the workflow

- **Code quality** (linter such as `ruff`, code formatter such as `black`)
- **Test coverage check** (`codecov`)
- **Documentation** (`sphinx`, `numpydoc`)
- **Security checks** (`CodeQL`)

Local CI/CD workflows

- Can run some tools, such as the linter and code formatter, before pushing code to the repository
- Can automatically run some actions using `pre-commit`

Running the CI/CD workflow

- Need a continuous integration tool
 - Bitbucket (<https://bitbucket.org/product/features/pipelines>)
 - Jenkins (<https://jenkins.io>)
 - AWS CodePipeline (<https://aws.amazon.com/codepipeline>)
 - CircleCI (<https://circleci.com>)
 - Azure (<https://azure.microsoft.com/>)
 - Gitlab (<https://about.gitlab.com/>)
 - GitHub (<https://github.com/>)
 - Etc.
- These tools use a YAML file (or similar) to describe a series of actions that make up a workflow.

GitHub Actions Dashboard

The screenshot shows the GitHub Actions dashboard for the repository 'TheDeanLab / ASLM'. The top navigation bar includes links to Code, Issues (36), Pull requests (2), Discussions, Actions (selected), Projects (1), Security (49), Insights, and Settings. A search bar is located on the right. The left sidebar lists various workflow categories: All workflows, Build Docs, Build Wheel, Lint, pages-build-deployment, Tests (selected), Management, Caches, Deployments, and Runners (Beta). The main content area is titled 'Tests' and shows the workflow 'push_checks.yaml'. It indicates that the workflow has a 'workflow_dispatch' event trigger and provides a 'Run workflow' button. Below this, a table lists recent workflow runs:

	Event	Status	Branch	Actor
1,232 workflow runs				
This workflow has a workflow_dispatch event trigger. Run workflow				
✓ Make the XML a bit nicer Tests #1792: Pull request #643 opened by zacsimile	zacsimile:xml-advances	18 hours ago 31m 46s		...
✓ Expose BDV data source as array Tests #1791: Pull request #642 synchronize by zacsimile	zacsimile:ds-load	yesterday 43m 43s		...
✓ Expose BDV data source as array Tests #1790: Pull request #642 synchronize by zacsimile	zacsimile:ds-load	yesterday 41m 51s		...

GitHub Actions Workflow Example

The screenshot shows a GitHub Actions workflow run for the repository "Make the XML a bit nicer #1792". The workflow is named "test (3.9, windows-latest)" and has succeeded 17 hours ago in 31m 31s. The interface includes a sidebar with navigation links: Summary, Jobs, Run details, Usage, and Workflow file. The "Jobs" section is active, showing a list of jobs. The "test (3.9, windows-latest)" job is selected, displaying a list of steps with their status and duration.

← Tests

✓ Make the XML a bit nicer #1792

Re-run all jobs

Summary

Jobs

✓ test (3.9, windows-latest)

Run details

Usage

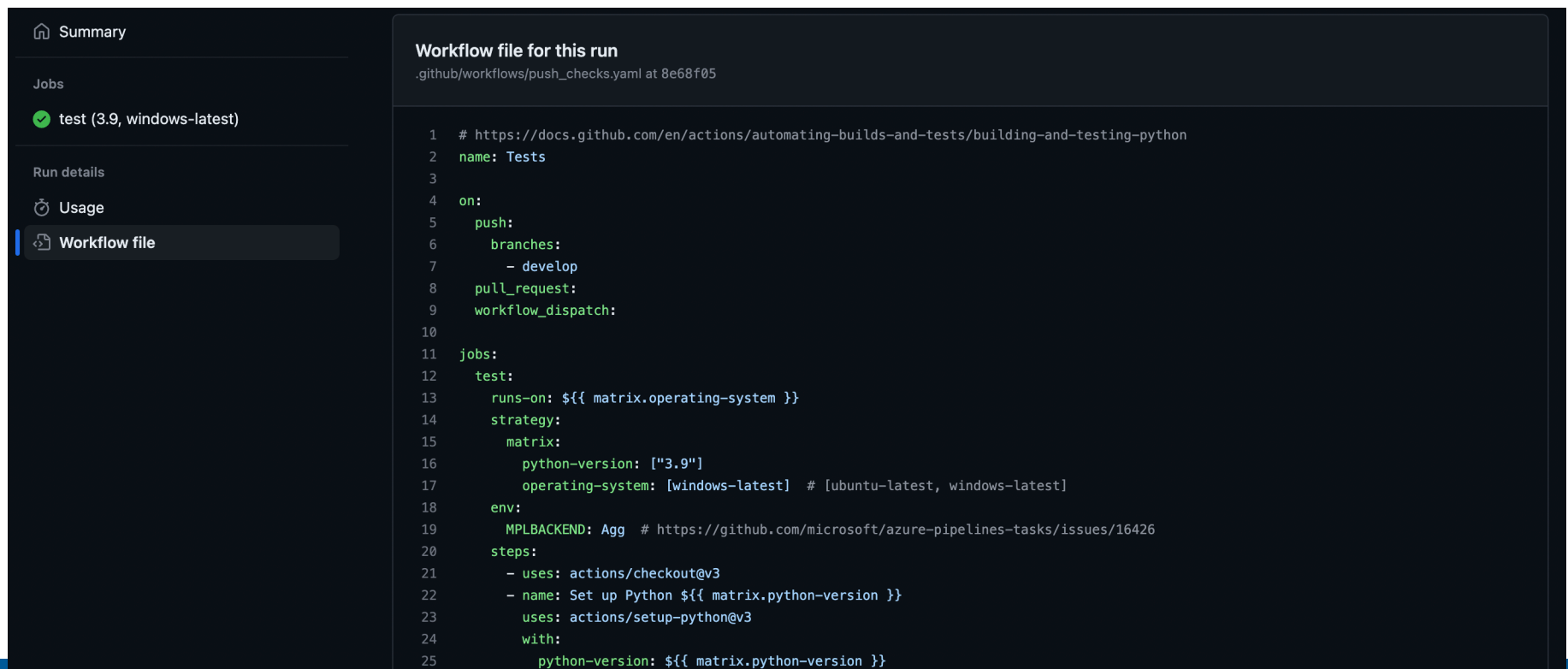
Workflow file

test (3.9, windows-latest)
succeeded 17 hours ago in 31m 31s

Search logs

> ✓ Set up job	2s
> ✓ Run actions/checkout@v3	8s
> ✓ Set up Python 3.9	0s
> ✓ Install dependencies	3m 38s
> ✓ Test with pytest	27m 36s
> ✓ Codecov	3s
> ✓ Post Set up Python 3.9	0s
> ✓ Post Run actions/checkout@v3	1s
> ✓ Complete job	0s

GitHub Actions Workflow Example



The screenshot shows the GitHub Actions interface for a workflow. On the left sidebar, the 'Workflow file' tab is selected. The main area displays the workflow file content, which is a YAML configuration for a workflow named 'Tests'.

Workflow file for this run
github/workflows/push_checks.yaml at 8e68f05

```
1 # https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
2 name: Tests
3
4 on:
5   push:
6     branches:
7       - develop
8   pull_request:
9   workflow_dispatch:
10
11 jobs:
12   test:
13     runs-on: ${ matrix.operating-system }
14     strategy:
15       matrix:
16         python-version: ["3.9"]
17         operating-system: [windows-latest] # [ubuntu-latest, windows-latest]
18     env:
19       MPLBACKEND: Agg # https://github.com/microsoft/azure-pipelines-tasks/issues/16426
20     steps:
21       - uses: actions/checkout@v3
22       - name: Set up Python ${ matrix.python-version }
23         uses: actions/setup-python@v3
24         with:
25           python-version: ${ matrix.python-version }
```