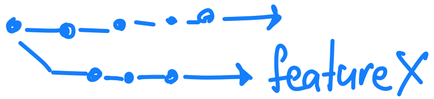
Why I Follow CI/CD Principles When Writing Code: Building Robust and Reproducible Applications

Artem Kislovskiy

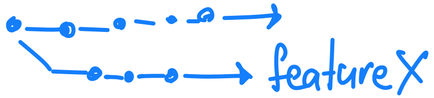
Continuous Integration

1. code locally on a feature branch



Continuous Integration

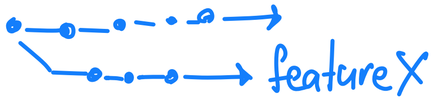
1. code locally on a feature branch



2. open a pull request

Continuous Integration

1. code locally on a feature branch

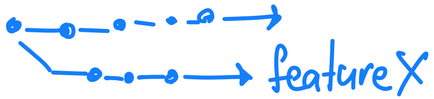


2. open a pull request

3. run automated checks & tests

Continuous Integration

1. code locally on a feature branch



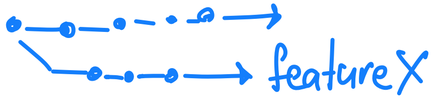
2. open a pull request

3. run automated checks & tests

4. if tests pass, merge to main

Continuous Integration

1. code locally on a feature branch



2. open a pull request

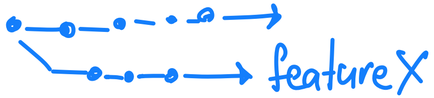
3. run automated checks & tests

4. if tests pass, merge to main

5. once merged run the tests again

Continuous Integration

1. code locally on a feature branch



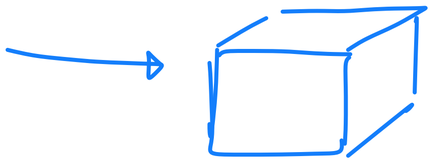
2. open a pull request

3. run automated checks & tests

4. if tests pass, merge to main

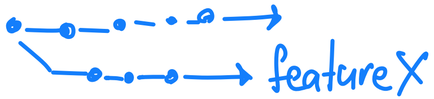
5. once merged run the tests again

6. if tests pass build an artifact



Continuous Integration

1. code locally on a feature branch



2. open a pull request

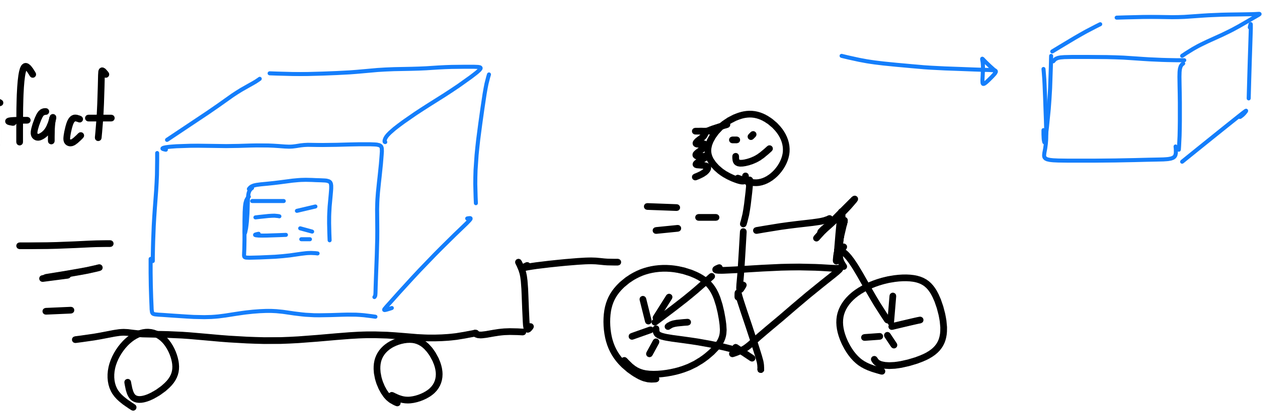
3. run automated checks & tests

4. if tests pass, merge to main

5. once merged run the tests again

6. if tests pass build an artifact

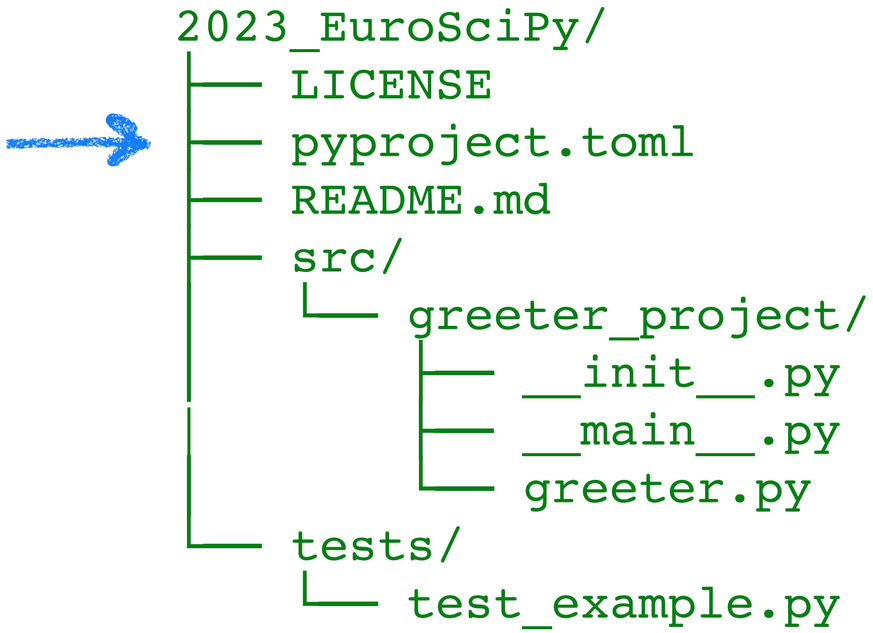
7. after it builds deliver the artifact



(Before CI) Python package

(Before CI) Python package

Packaging Python Projects



(Before CI) Python package

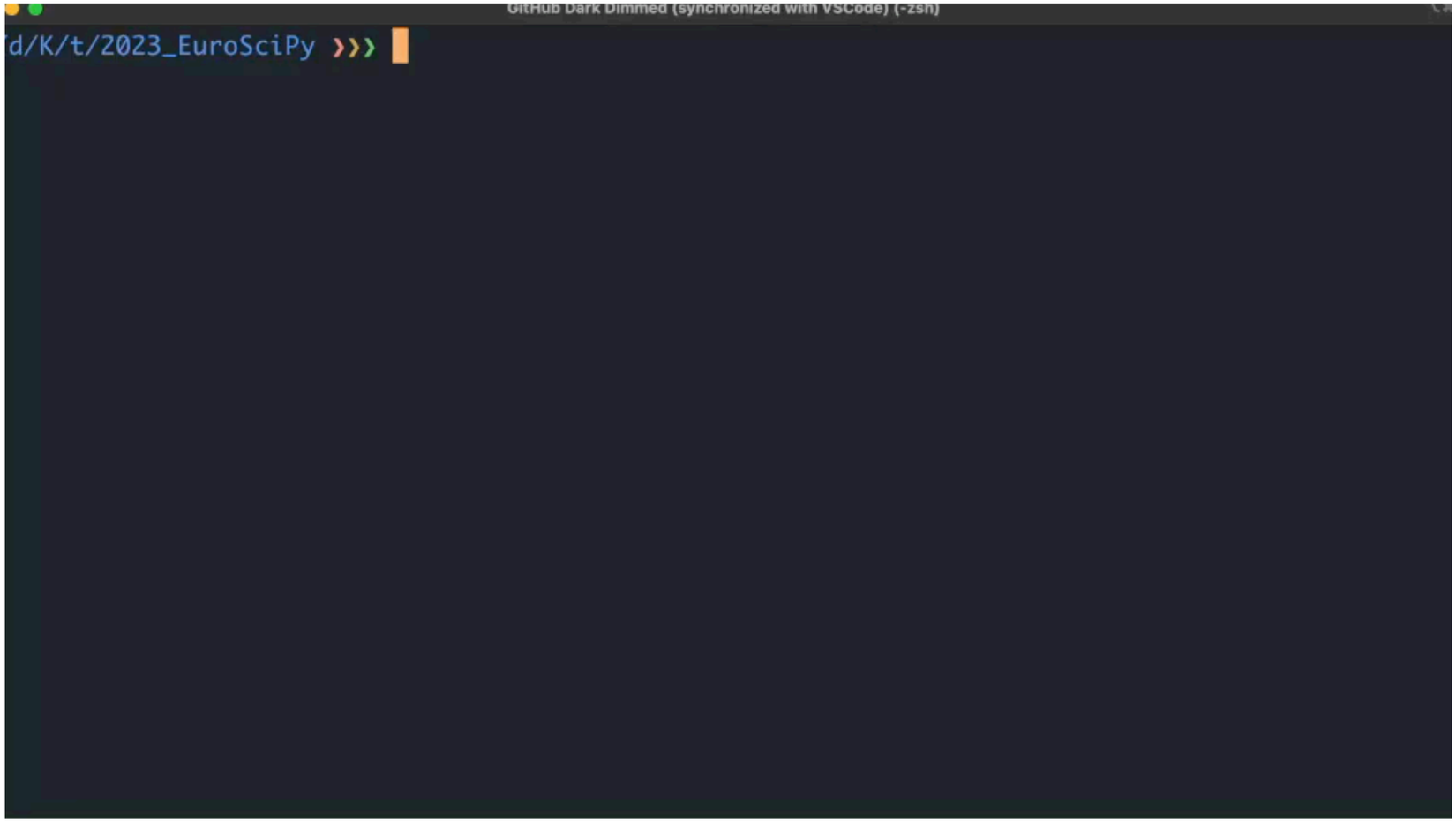
```
2023_EuroSciPy/
├── LICENSE
├── pyproject.toml
├── README.md
├── src/
│   └── greeter_project/
│       ├── __init__.py
│       ├── __main__.py
│       └── greeter.py
└── tests/
    └── test_example.py
```

```python
# __main__.py
import sys

from greeter_project import greeter

greeter.greet(" ".join(sys.argv[1:]))
```

```python
# greeter.py
import art


def greet(message: str) -> None:
    print(art.text2art(message))
```

(Before CI) Python package

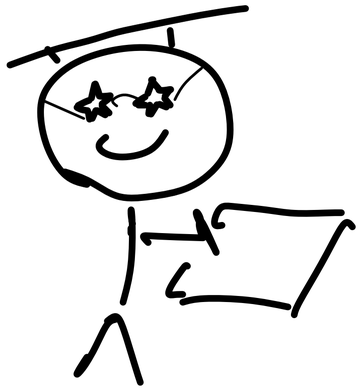
```
2023_EuroSciPy/
├── LICENSE
├── pyproject.toml
├── README.md
├── src/
│   └── greeter_project/
│       ├── __init__.py
│       ├── __main__.py
│       └── greeter.py
└── tests/
    └── test_example.py
```



(Before CI) **pyproject.toml**

Configuring setuptools using pyproject.toml files

PEP 621



https://setuptools.pypa.io/en/latest/userguide/pyproject_config.html 133%

SETUPTOOLS

Search

LINKS

Home

PyPI

User guide

Quickstart

Package Discovery and Namespace Packages

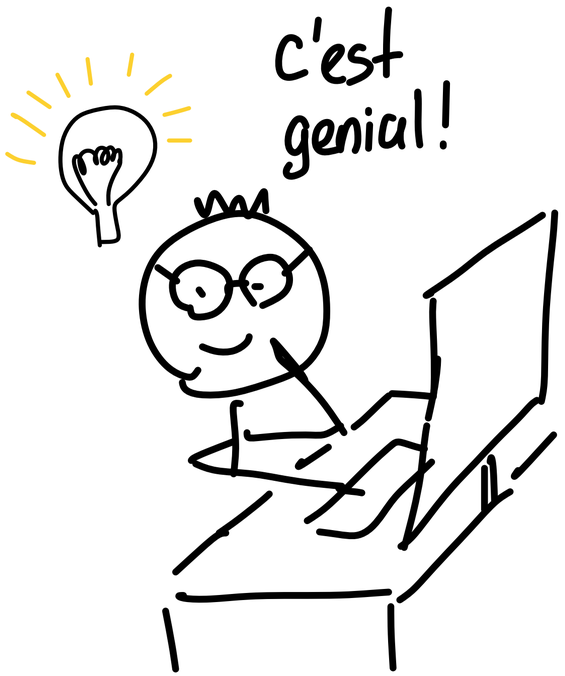
Dependencies Management in Setuptools

v: latest

Back to top

```toml
[build-system]
requires = ["setuptools -scm"]
build-backend = "setuptools.build_meta"

[project]
name = "my_package"
authors = [
    {name = "Josiah Carberry", email = "josiah_carberry@brown.edu"},
]
description = "My package description"
readme = "README.rst"
requires-python = ">=3.7"
keywords = ["one", "two"]
license = {text = "BSD-3-Clause"}
classifiers = [
    "Framework :: Django",
    "Programming Language :: Python :: 3",
]
dependencies = [
    "requests",
    'importlib-metadata; python_version<"3.8"',
]
dynamic = ["version"]

[project.optional-dependencies]
pdf = ["ReportLab>=1.2", "RXP"]
rest = ["docutils>=0.3", "pack ==1.1, ==1.3"]

[project.scripts]
my-script = "my_package.module:function"
```

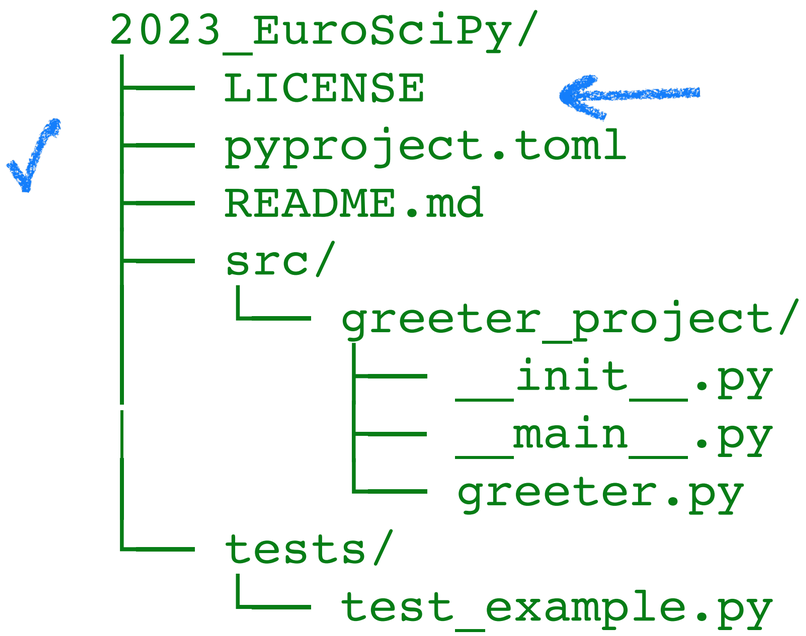
(Before CI) pyproject.toml




```toml
# pyproject.toml

[build-system]
requires = ["setuptools", "setuptools-scm"]
build-backend = "setuptools.build_meta"

[project]
name = "greeter_project"
version = "1.0.0"
dependencies = [
    "art==6.0",
]
```

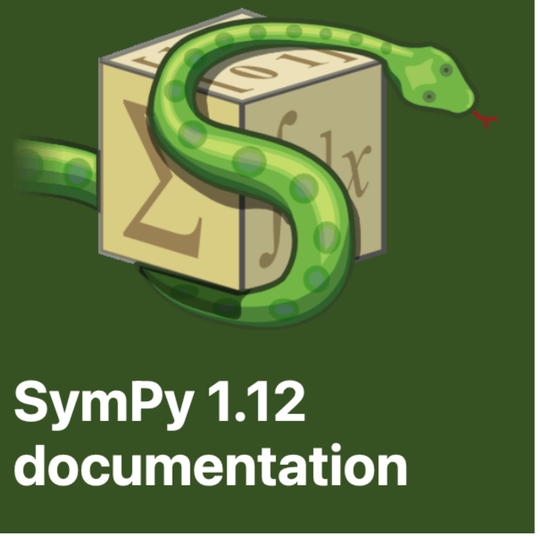
```
$ python -m venv .2023_EuroSciPy
$ source .2023_EuroSciPy/bin/activate
$ python -m pip install --editable .
$ python src/greeter_project/__main__.py “Hello EuroSciPY”
$ python src/greeter_project “Hello EuroSciPy”
$ python -m greeter_project “Hello EuroSciPy”
```

Packaging Python Projects



(Before CI) LICENCE

https://docs.sympy.org/latest/tutorials/intro-tutorial/intro.html 133%



Why SymPy?

↑ Back to top

There are many computer algebra systems out there. This Wikipedia article lists many of them. What makes SymPy a better choice than the alternatives?

First off, SymPy is completely free. It is open source, and licensed under the liberal BSD license, so you can modify the source code and even sell it if you want to. This contrasts with popular commercial systems like Maple or Mathematica that cost hundreds of dollars in licenses.

Second, SymPy uses Python. Most computer algebra systems invent their own language. Not

(Before CI) **LICENCE**

NumPy

The fundamental package for scientific computing with Python

LATEST RELEASE: NUMPY 1.25. VIEW ALL RELEASES

numpy.org is now available in Japanese and Portuguese 2023-08-02

POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

OPEN SOURCE

Distributed under a liberal BSD license, NumPy is developed and maintained publicly on GitHub by a vibrant, responsive, and diverse community.

(Before CI) LICENCE

Choose an open source license

An open source license protects contributors and users. Businesses and savvy developers won't touch a project without this protection.

Which of the following best describes your situation?

I need to work in a community.

Use the **license preferred by the community** you're contributing to or depending on. Your project will fit right in.

If you have a dependency that doesn't have a license, ask its maintainers to **add a license**.

I want it simple and permissive.

The **MIT License** is short and to the point. It lets people do almost anything they want with your project, like making and distributing closed source versions.

Babel, **.NET**, and **Rails** use the MIT License.

I care about sharing improvements.

The **GNU GPLv3** also lets people do almost anything they want with your project, *except* distributing closed source versions.

Ansible, **Bash**, and **GIMP** use the GNU GPLv3.

What if none of these work for me?

My project isn't software.

There are licenses for that.

I want more choices.

More licenses are available.

I don't want to choose a license.

Here's what happens if you don't.

About Terms of Service Help improve this page

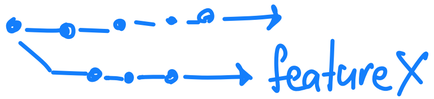
Curated with ❤️ by GitHub, Inc. and You!

https://choosealicense.com/

or follow -> Anwesha Das 🙃

Continuous Integration

1. code locally on a feature branch



2. open a pull request

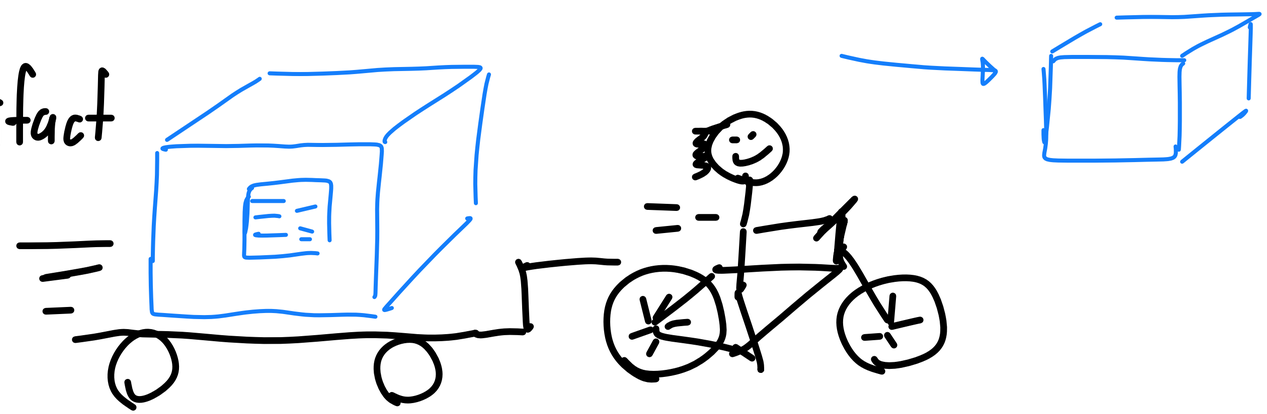
3. run automated checks & tests

4. if tests pass, merge to main

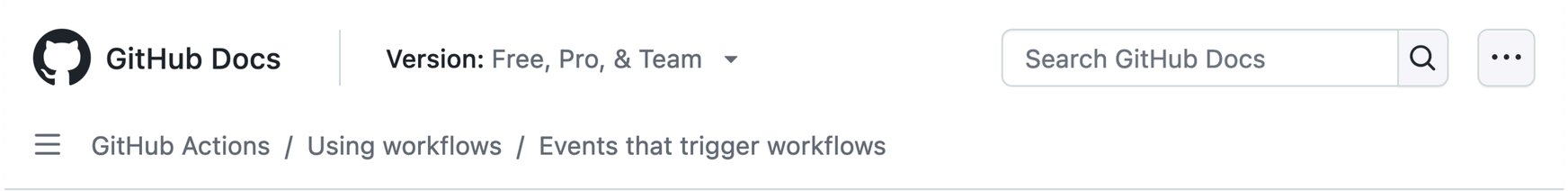
5. once merged run the tests again

6. if tests pass build an artifact

7. after it builds deliver the artifact



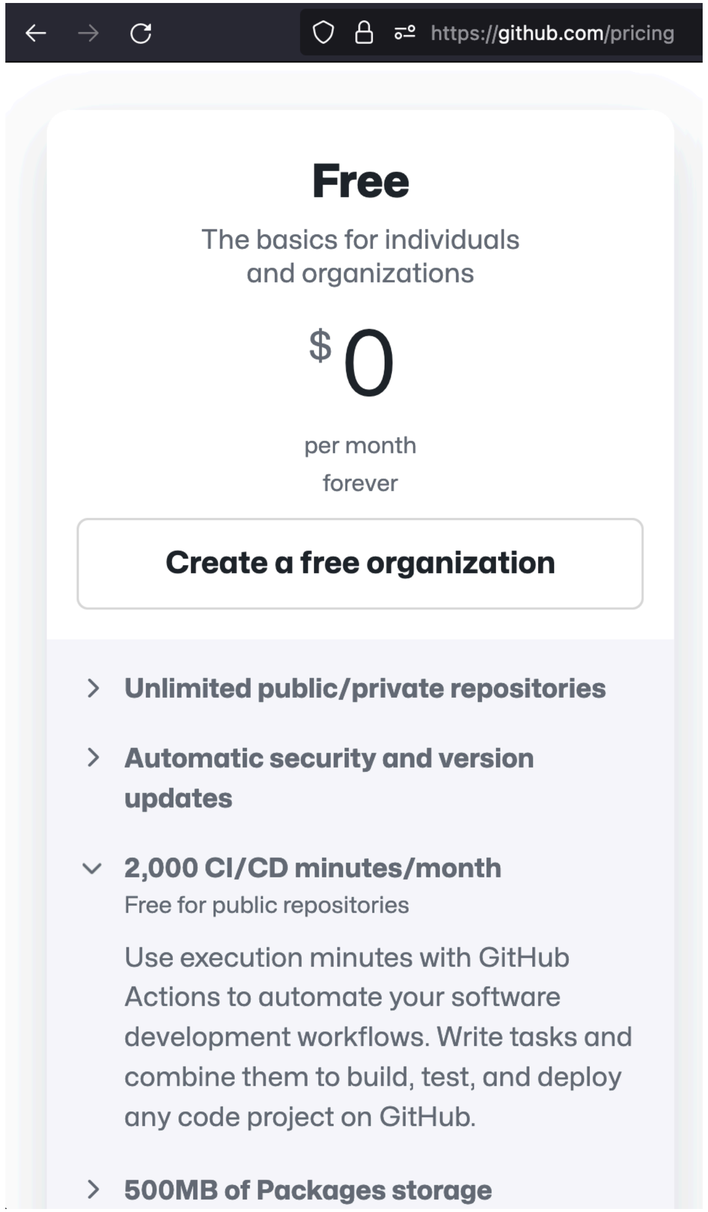
GitHub Actions

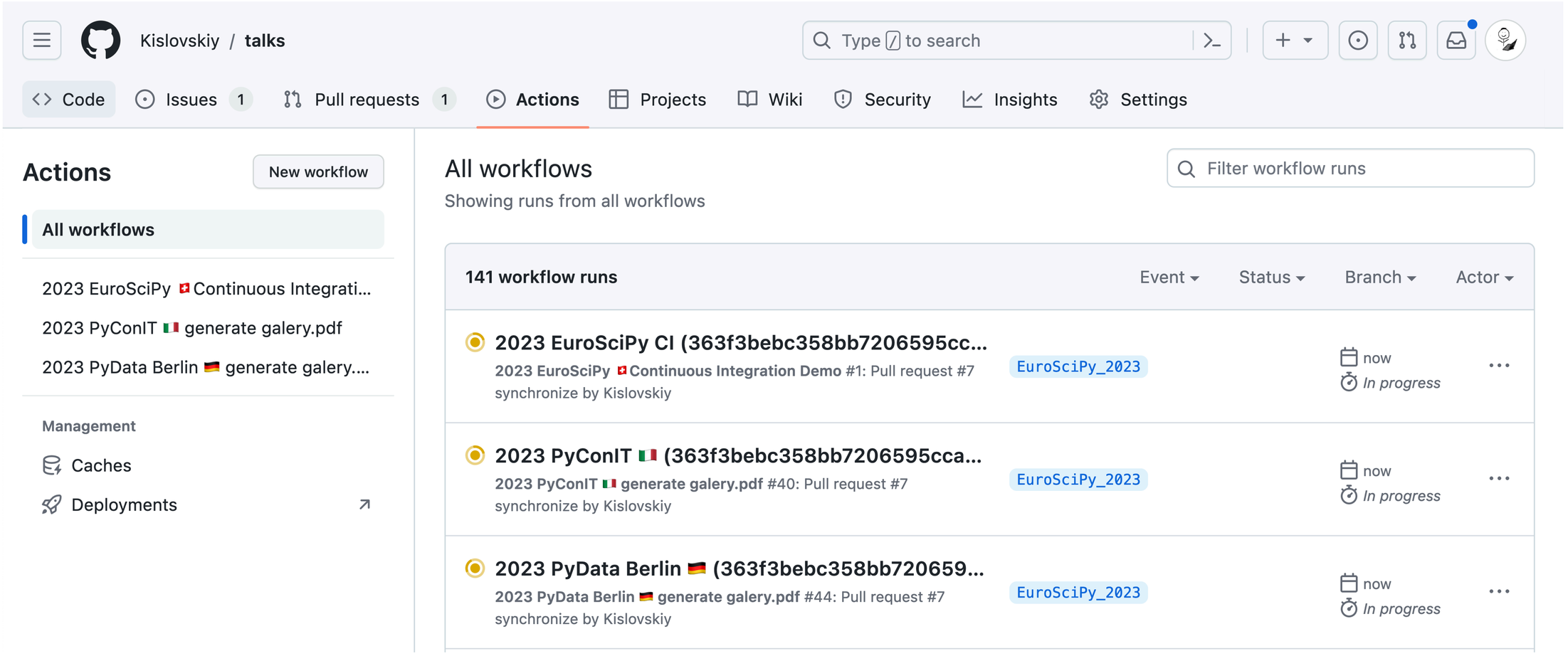


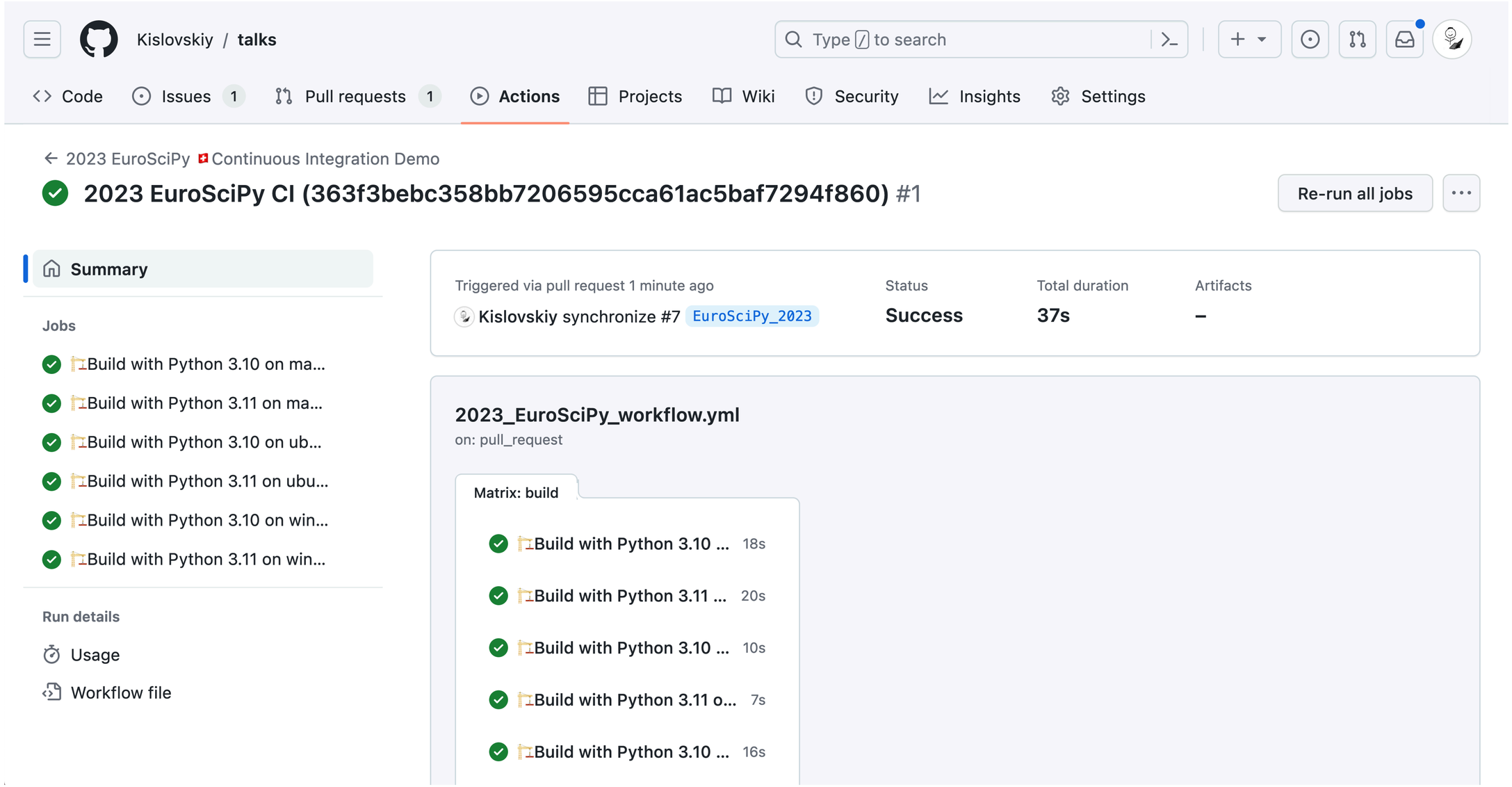
Events that trigger workflows

You can configure your workflows to run when specific activity on GitHub happens, at a scheduled time, or when an event outside of GitHub occurs.

merge_group
milestone
page_build
project
project_card
project_column
public
pull_request
pull_request_comment (use issue_comment)
pull_request_review
pull_request_review_comment
pull_request_target
push
registry_package
release
repository_dispatch
schedule
status
watch
workflow_call
workflow_dispatch


https://github.com/pricing
Free
The basics for individuals and organizations
$0
per month
forever
Create a free organization
Unlimited public/private repositories
Automatic security and version updates
2,000 CI/CD minutes/month
Free for public repositories
Use execution minutes with GitHub Actions to automate your software development workflows. Write tasks and combine them to build, test, and deploy any code project on GitHub.
500MB of Packages storage

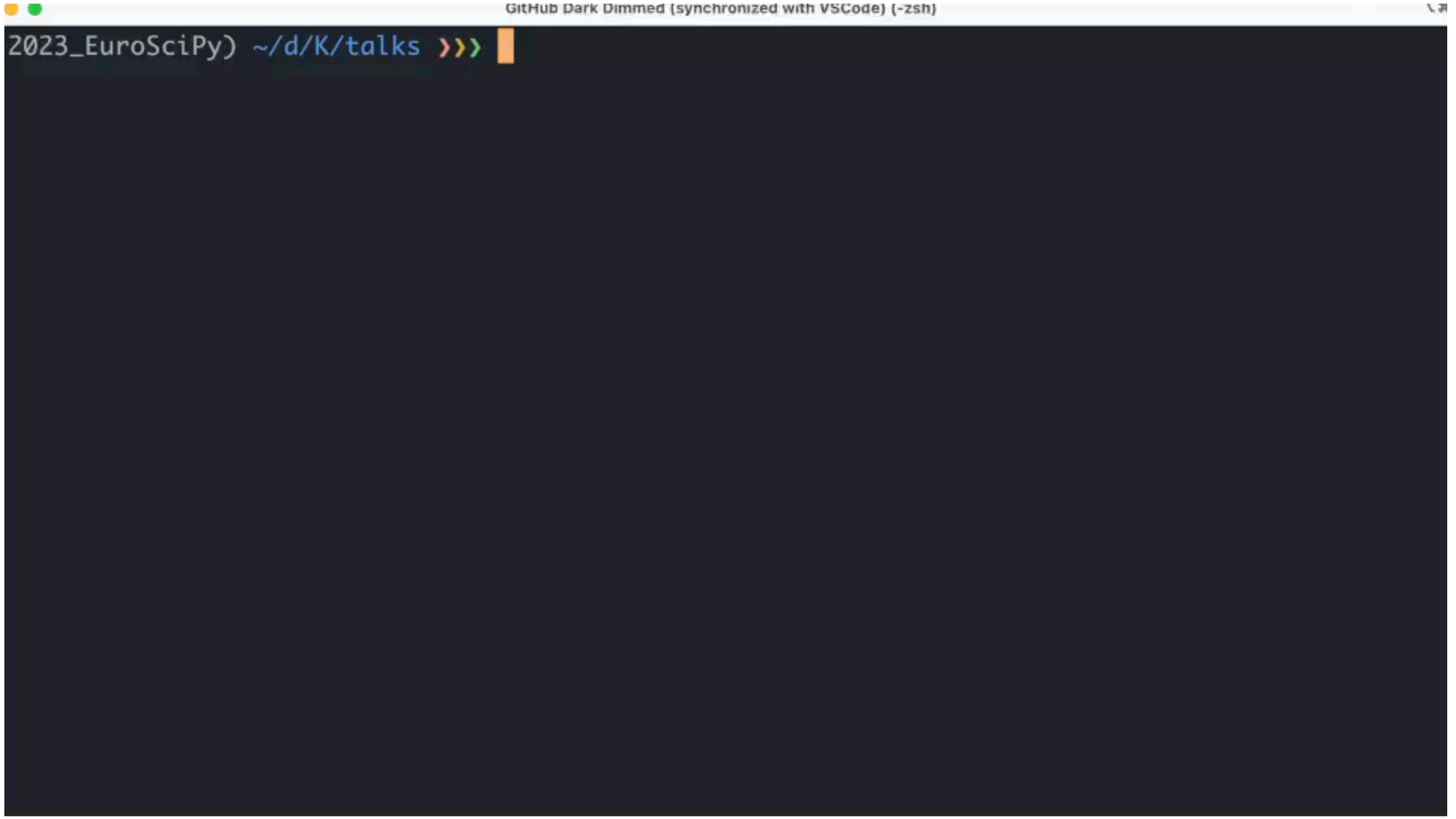

Kislovskiy / talks
Type / to search
Code
Issues 1
Pull requests 1
Actions
Projects
Wiki
Security
Insights
Settings
Actions
New workflow
All workflows
2023 EuroSciPy Continuous Integrati...
2023 PyConIT generate galery.pdf
2023 PyData Berlin generate galery....
Management
Caches
Deployments
All workflows
Showing runs from all workflows
Filter workflow runs
141 workflow runs
Event
Status
Branch
Actor
2023 EuroSciPy CI (363f3bebc358bb7206595cc...
2023 EuroSciPy Continuous Integration Demo #1: Pull request #7
synchronize by Kislovskiy
EuroSciPy_2023
now
In progress
2023 PyConIT (363f3bebc358bb7206595cca...
2023 PyConIT generate galery.pdf #40: Pull request #7
synchronize by Kislovskiy
EuroSciPy_2023
now
In progress
2023 PyData Berlin (363f3bebc358bb720659...
2023 PyData Berlin generate galery.pdf #44: Pull request #7
synchronize by Kislovskiy
EuroSciPy_2023
now
In progress


Kislovskiy / talks
Type / to search
Code
Issues 1
Pull requests 1
Actions
Projects
Wiki
Security
Insights
Settings
2023 EuroSciPy Continuous Integration Demo
2023 EuroSciPy CI (363f3bebc358bb7206595cca61ac5baf7294f860) #1
Re-run all jobs
Summary
Jobs
Build with Python 3.10 on ma...
Build with Python 3.11 on ma...
Build with Python 3.10 on ub...
Build with Python 3.11 on ubu...
Build with Python 3.10 on win...
Build with Python 3.11 on win...
Run details
Usage
Workflow file
Triggered via pull request 1 minute ago
Kislovskiy synchronize #7 EuroSciPy_2023
Status
Success
Total duration
37s
Artifacts
–
2023_EuroSciPy_workflow.yml
on: pull_request
Matrix: build
Build with Python 3.10 ... 18s
Build with Python 3.11 ... 20s
Build with Python 3.10 ... 10s
Build with Python 3.11 o... 7s
Build with Python 3.10 ... 16s

```yaml
# .github/workflows/ci.yml

name: "Continuous Integration"
run-name: "CI for (${{ github.sha }})"

on:
  pull_request:
    branches:
      - main
  push:
    branches:
      - main
  workflow_dispatch:

defaults:
  run:
    shell: bash -e {0}
```

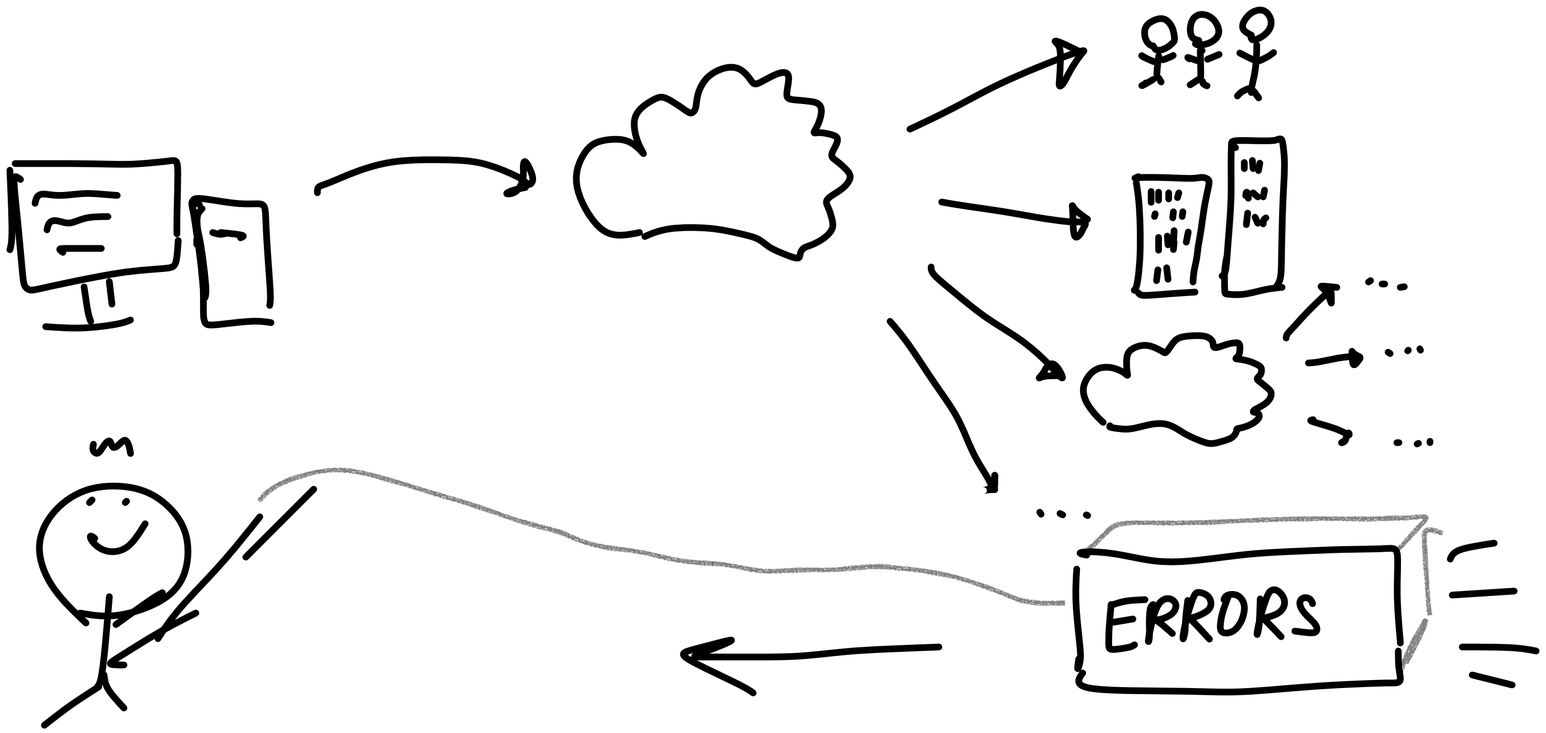
```yaml
# .github/workflows/ci.yml

<…>

jobs:
  build:
    name: Build with Python ${{ matrix.python-version }} on ${{ matrix.os }}
    runs-on: ${{ matrix.os }}

    strategy:
      fail-fast: false
      matrix:
        os: [ macos-latest, ubuntu-latest, windows-latest ]
        python-version: [ "3.9", "3.10", "3.11" ]

    steps:
      - uses: actions/checkout@v3
        with:
          fetch-depth: 0
      - name: Set up Python ${{ matrix.python-version }}
        uses: actions/setup-python@v4
        with:
          python-version: ${{ matrix.python-version }}
```

Debug GitHub Actions with act

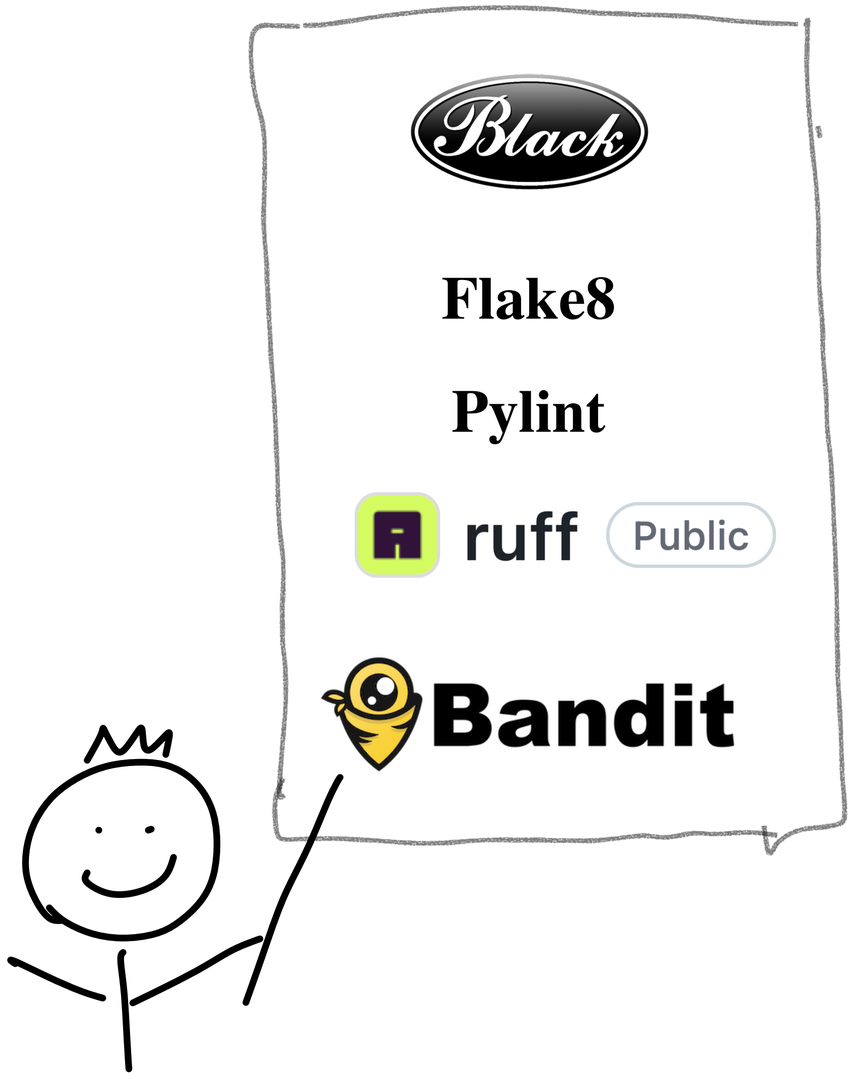


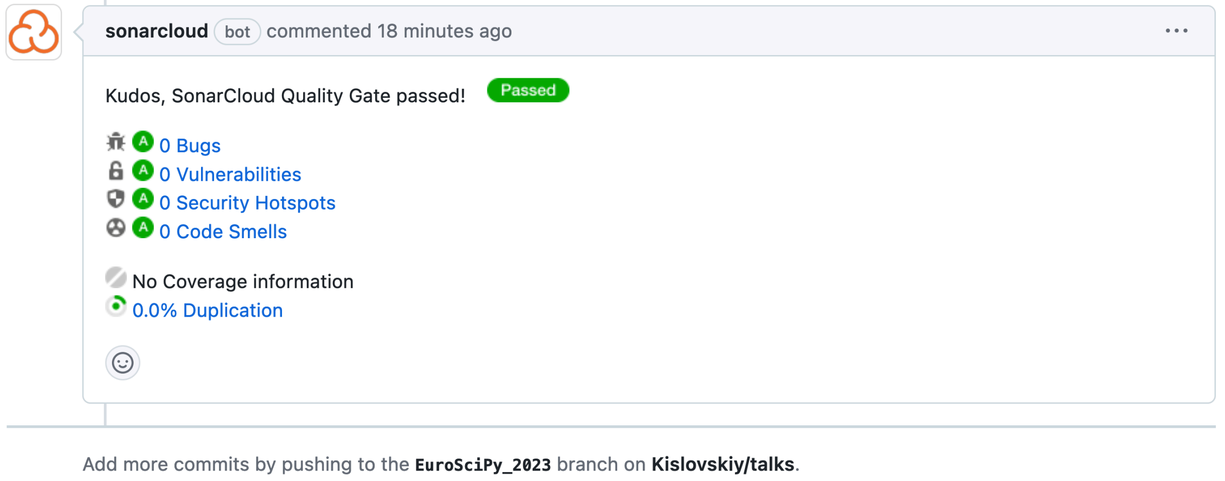
https://github.com/nektos/act

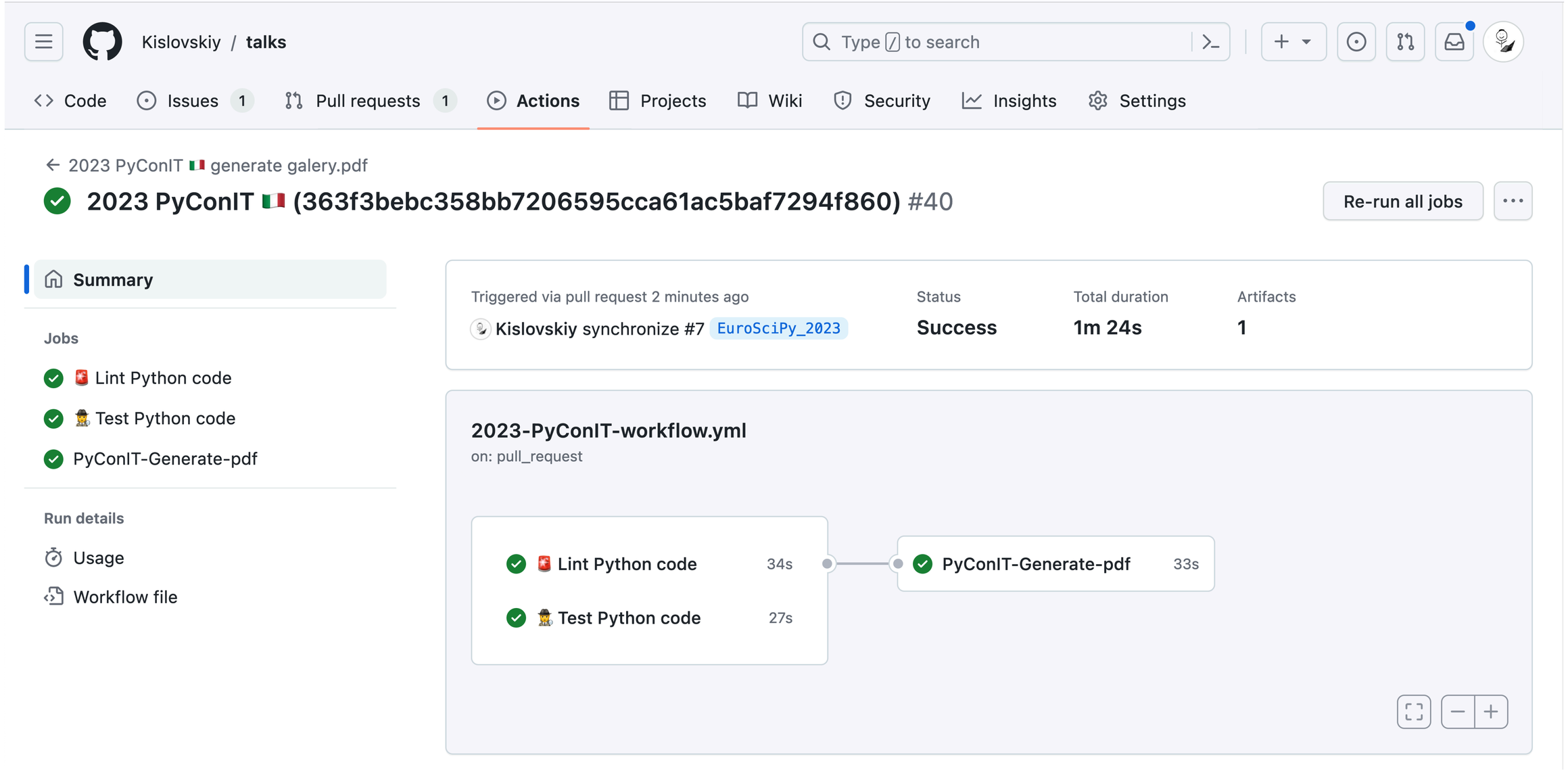
Why Continuous Integration?



Static Analysers



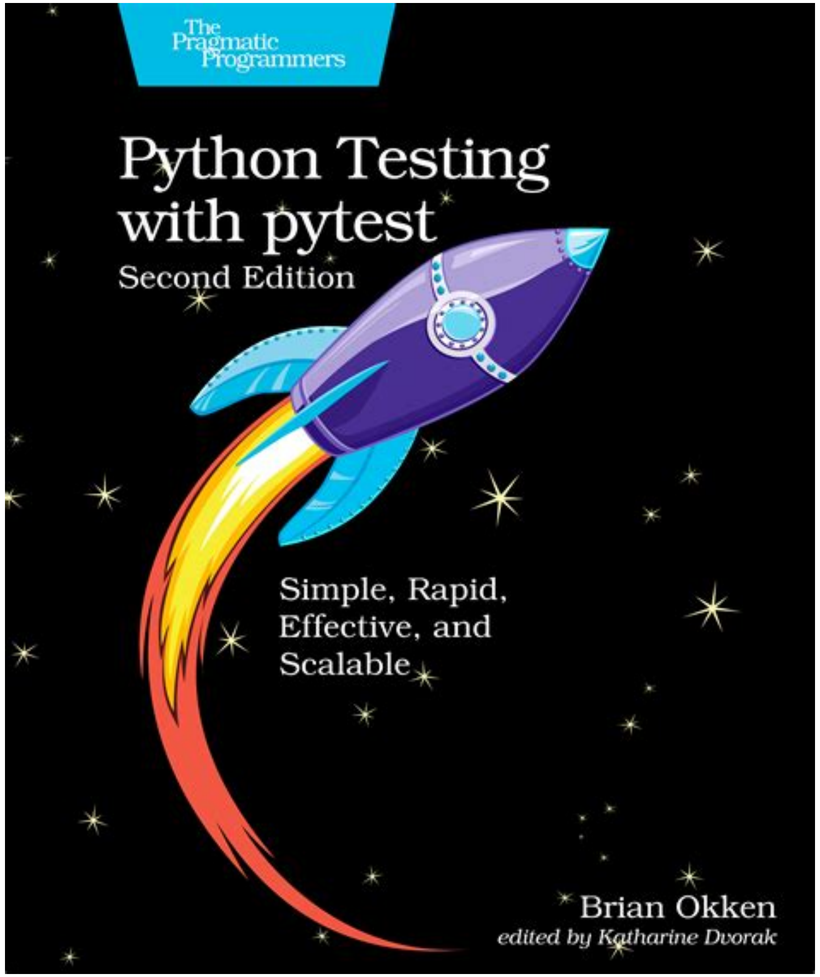



Kislovskiy / talks
Type / to search
Code
Issues 1
Pull requests 1
Actions
Projects
Wiki
Security
Insights
Settings
2023 PyConIT 🇮🇹 generate galery.pdf
2023 PyConIT 🇮🇹 (363f3bebc358bb7206595cca61ac5baf7294f860) #40
Re-run all jobs
Summary
Jobs
Lint Python code
Test Python code
PyConIT-Generate-pdf
Run details
Usage
Workflow file
Triggered via pull request 2 minutes ago
Kislovskiy synchronize #7 EuroSciPy_2023
Status
Success
Total duration
1m 24s
Artifacts
1
2023-PyConIT-workflow.yml
on: pull_request
Lint Python code 34s
Test Python code 27s
PyConIT-Generate-pdf 33s

```yaml
# .github/workflows/ci.yml

<…>
jobs:
  lint:
    name: 🚨 Lint Python code
    runs-on: ubuntu-latest
    steps:
      - name: Setup Python environment
        uses: actions/checkout@v3
        with:
          fetch-depth: 0
      - uses: actions/setup-python@v4
        with:
          python-version: ${{ env.PYTHON_VERSION }}
      - name: Install dependencies
        run:
          pip install -r 2023_PyData_Berlin/requirements.txt
      - name: Run Black
        run:
          black 2023_PyData_Berlin --check
      - name: Run Flake8
        run:
          flake8 --config 2023_PyData_Berlin/.flake8 2023_PyData_Berlin
```

Testing

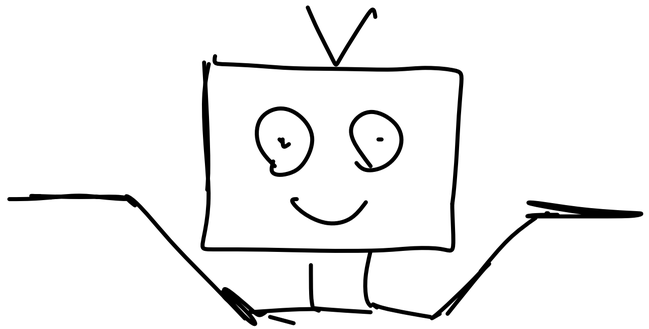

Pytest

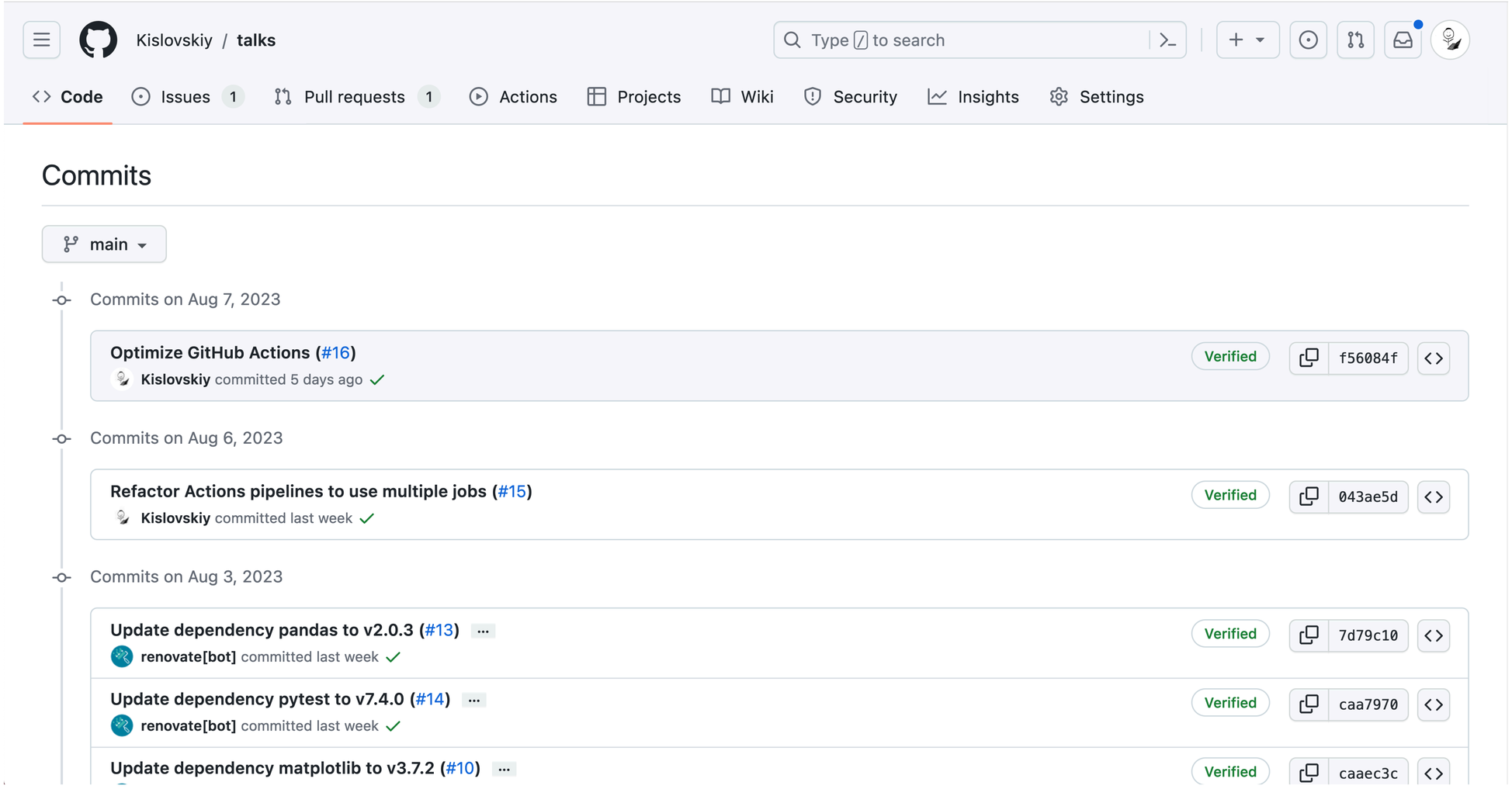

The Pragmatic Programmers
Python Testing with pytest
Second Edition
Simple, Rapid, Effective, and Scalable
Brian Okken
edited by Katharine Dvorak

Automated dependency updates

Renovate

Dependabot




Kislovskiy / talks
Type / to search
Code
Issues 1
Pull requests 1
Actions
Projects
Wiki
Security
Insights
Settings
Commits
main
Commits on Aug 7, 2023
Optimize GitHub Actions (#16)
Kislovskiy committed 5 days ago
Verified
f56084f
Commits on Aug 6, 2023
Refactor Actions pipelines to use multiple jobs (#15)
Kislovskiy committed last week
Verified
043ae5d
Commits on Aug 3, 2023
Update dependency pandas to v2.0.3 (#13)
renovate[bot] committed last week
Verified
7d79c10
Update dependency pytest to v7.4.0 (#14)
renovate[bot] committed last week
Verified
caa7970
Update dependency matplotlib to v3.7.2 (#10)
Verified
caaec3c

Update dependency pandas to v2.0.3 #13

Edit Code

Merged **Kislovskiy** merged 1 commit into `main` from `renovate/pandas-2.x` last week

Conversation 1 Commits 1 Checks 5 Files changed 1 +1 −1

renovate bot commented last week Contributor

This PR contains the following updates:

Package	Change	Age	Adoption	Passing	Confidence
pandas	`==2.0.1` -> `==2.0.3`	45d	9%	99%	high

Release Notes

▾ pandas-dev/pandas (pandas)

`v2.0.3`: Pandas 2.0.3

Compare Source

This is a patch release in the 2.0.x series and includes some regression and bug fixes. We recommend that all users upgrade to this version.

See the full whatsnew for a list of all the changes.

The release will be available on the defaults and conda-forge channels:

conda install pandas

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.

None yet

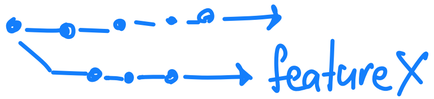
Notifications Customize

Unsubscribe

You're receiving notifications because you're

Continuous Integration

1. code locally on a feature branch



2. open a pull request

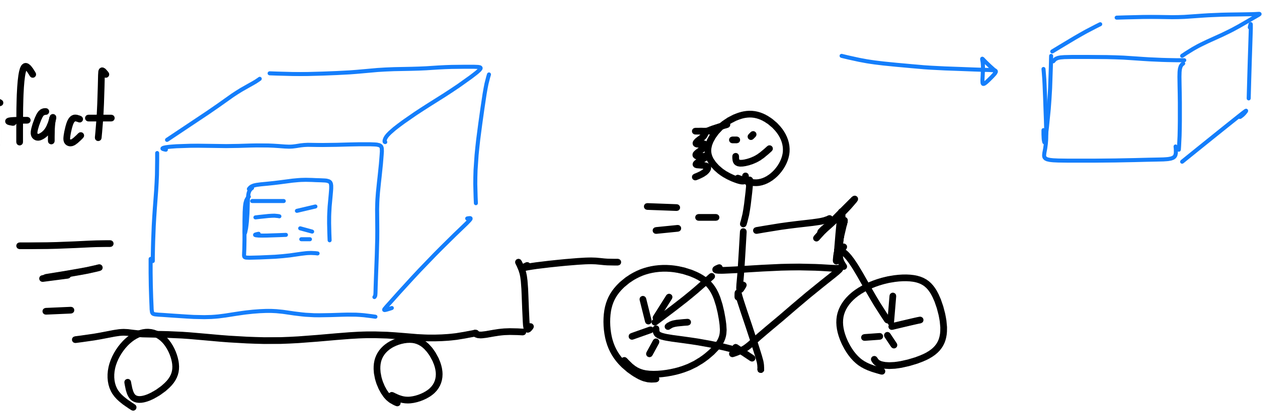
3. run automated checks & tests

4. if tests pass, merge to main

5. once merged run the tests again

6. if tests pass build an artifact

7. after it builds deliver the artifact



https://github.com/Kislovskiy/talks