

Continues Integration (CI)

Machine Learning Operations

Nicki Skafte Detlefsen,

Postdoc

DTU Compute

Why you should care about today's exercises



Dev at Fastly : I'll just push this small change to production

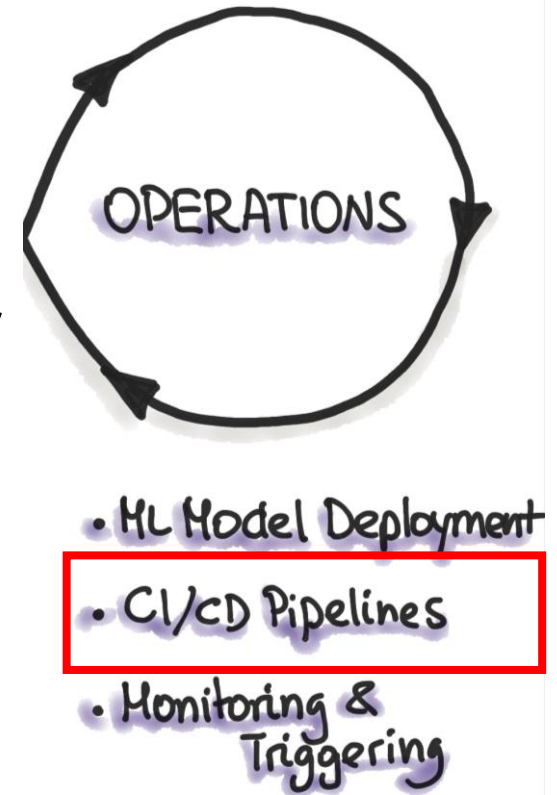
Dev at Fastly 2 seconds later:



What is continues integration?



- Software practice
 - Frequently commit code to shared repository
 - By committing sooner than later, errors are captured early
 - Make merging easier
 - Automate build + test
- App independent



What is continues deployment?



- How to get your code to the user
- Covered in later lecture
- App dependent



- ML Model Deployment
- CI/CD Pipelines
- Monitoring & Triggering

What should you know about CI?



- CI is one of those topics that are best though as learning-by-doing
- If you understand how to use git, the rest is basically googling stuff

Let's look at a practical example

A small case study



All the metric in the world – now in Pytorch

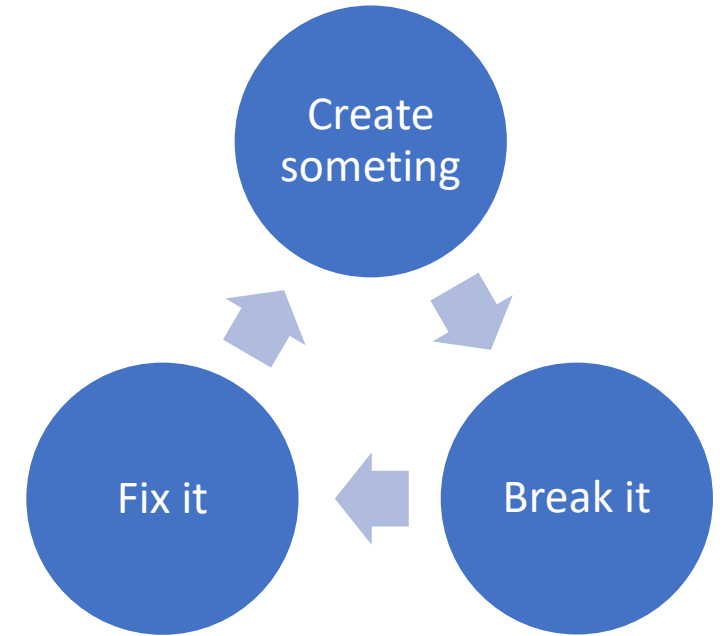
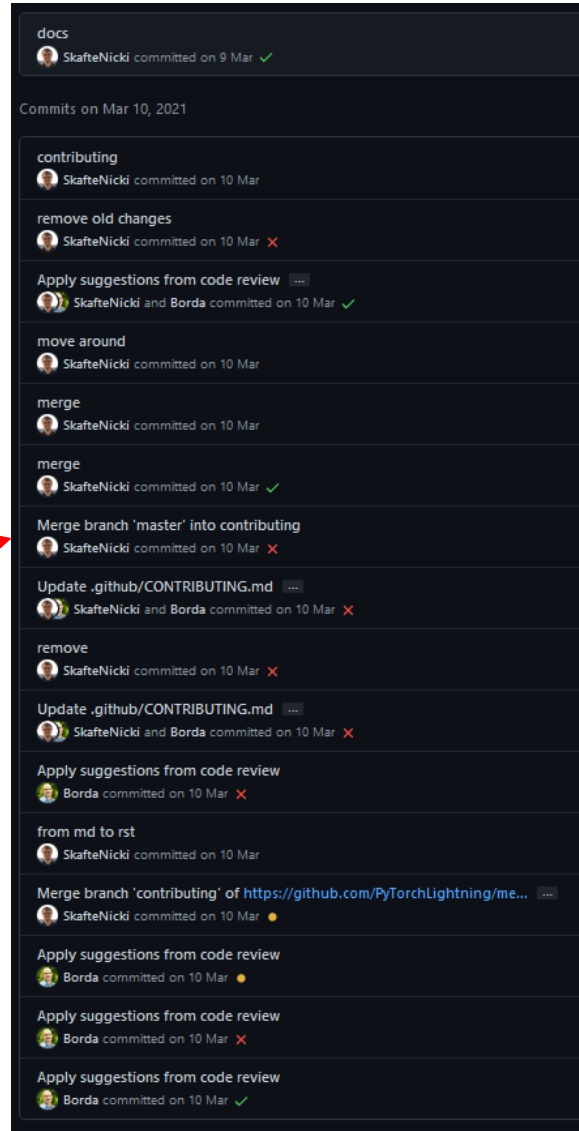
The screenshot displays the GitHub repository for TorchMetrics. The main content area shows a list of recent pull requests, including changes to CHANGELOG.md, LICENSE, MANIFEST.in, Makefile, README.md, azure-pipelines.yml, pyproject.toml, requirements.txt, setup.cfg, and setup.py. The README section features the TorchMetrics logo and the tagline 'Machine learning metrics for distributed, scalable PyTorch applications.' Below this, there are links for 'What is Torchmetrics', 'Implementing a metric', 'Built-in metrics', 'Docs', 'Community', and 'License'. At the bottom, there are badges for Python versions (3.6, 3.7, 3.8, 3.9), PyPI package (0.3.2), downloads (372k), conda (v0.3.2), downloads (25k), License (Apache 2.0), CI testing (base passing), Azure Pipelines (failed), codecov (97%), slack chat, and docs (passing). The sidebar on the right shows 71 contributors and a language distribution chart indicating Python at 99.9% and Makefile at 0.1%.

Have I seen that
guy before?

CI step 1: Committing code



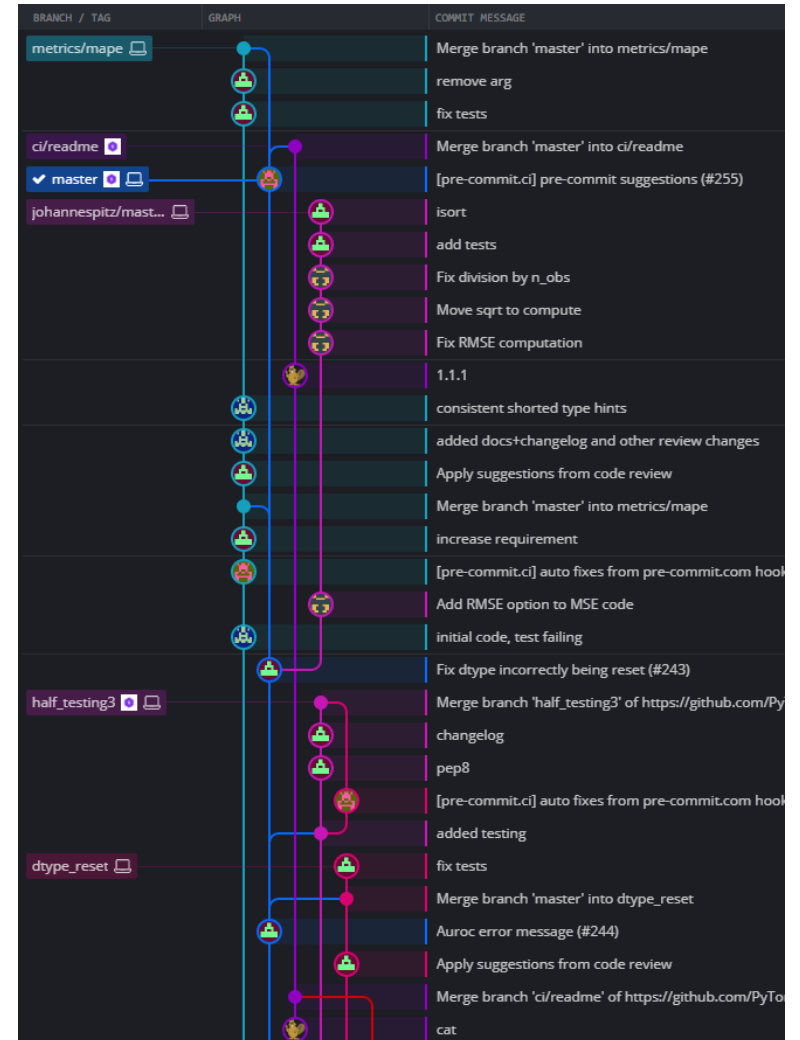
- Commit frequently
 - Catch errors sooner than later
 - Merging can be done automatically



CI step 1: Committing code



- Use branches
 - Enables parallel workflow
- Experimental features/changes are kept away from "stable" master



CI step 1: Committing code



- Use PRs, other can review your code

A screenshot of a GitHub Pull Request (PR) interface for the repository "PyTorchLightning / metrics". The PR is titled "testing readme examples #140" and is in the "Open" state. The interface shows the PR details, a list of files changed (4 files), and a code diff. Annotations in red text and boxes highlight key steps in the PR process: 1. Find PR (pointing to the "Pull requests" tab), 2. Check changed files (pointing to the "Files changed" tab), 3. Make one or more comments (pointing to a comment by "SkafteNicki" suggesting a rename and showing a suggested change), and 4. Send review (pointing to the "Finish your review" button). The code diff shows changes to ".github/set-minimal-versions.py".

PyTorchLightning / metrics

generated from PyTorchLightning/lightning-sandbox

1. Find PR

testing readme examples #140

2. Check changed files

4. Send review

3. Make one or more comments

should we rename this to be more descriptive:

Suggested change

```
4 - LUT_PYTHON_TORCH = {  
4 + MIN_PYTHON_TORCH_COMPATIBLE = {
```

```
5 + '3.8': '1.4',  
6 + '3.9': '1.7.1',  
7 + }  
8 +  
9 +  
10 + def set_min_torch_by_python(fpath: str = 'requirements.txt') -> None:  
11 + py_ver = f'{sys.version_info.major}.{sys.version_info.minor}'  
12 + if py_ver not in LUT_PYTHON_TORCH:  
13 +     return  
14 + req = re.sub(r'torch=[\d.]*', f'torch={LUT_PYTHON_TORCH[py_ver]}', open(fpath).read())  
15 + open(fpath, 'w').write(req)  
16 +
```

CI step 2: Automating stuff



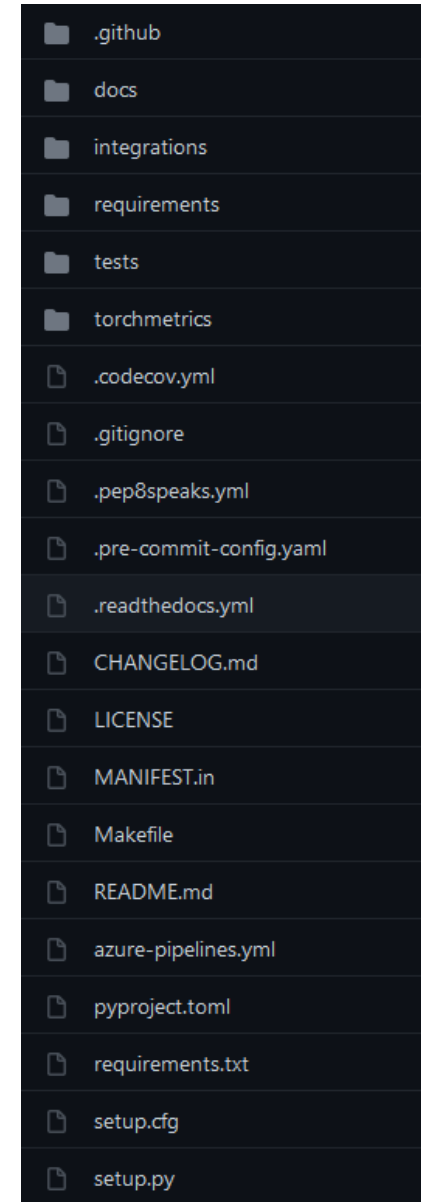
- What can be automated: EVERYTHING
 - Functional tests
 - Documentation creation
 - Linters (which check style formatting)
 - Security checks
 - Code coverage
 - Custom checks...

A small case study





Ranked by importance (biased)

1. Source code
2. Tests
3. Setup
4. CI workflows
5. Documentation
6. ★



Source code



master	metrics / torchmetrics /	Go to file	Add file	...
	SkaftNicki Fix dtype incorrectly being reset (#243) ...	✓ cda5dbd	5 days ago	 History
..				
classification	Allow logit input in classification metrics (#200)			6 days ago
functional	Fix dtype incorrectly being reset (#243)			5 days ago
regression	Add differentiability testing to more metrics [3/n] (#225)			13 days ago
retrieval	Information Retrieval (5/5) (#160)			last month
utilities	Allow logit input in classification metrics (#200)			6 days ago
wrappers	Feature pre commit yaml (#145)			2 months ago
__about__.py	Show must go on (#198)			25 days ago
__init__.py	Add Specificity metric (#210)			15 days ago
average.py	allow MetricCollection with args (#176)			29 days ago
collections.py	Added add_metrics method to MetricCollection (#221)			14 days ago
metric.py	Fix dtype incorrectly being reset (#243)			5 days ago
py.typed	Make torchmetrics PEP 561 compatible (#215)			19 days ago
setup_tools.py	Remove numpy dependency from the core library (#212)			18 days ago

Test code



In total we have 6365 tests

master	metrics / tests /	Go to file	Add file	...
	SkaftNicki Fix dtype incorrectly being reset (#243) ...	✓ cda5dbd	5 days ago	History
..				
bases	Fix dtype incorrectly being reset (#243)		5 days ago	
classification	Auroc error message (#244)		5 days ago	
functional	Refactor Information Retrieval tests (#156)		last month	
helpers	Ports are reused breaking parallel tests (#226)		14 days ago	
regression	Add differentiability testing to more metrics [3/n] (#225)		13 days ago	
retrieval	Remove numpy dependency from the core library (#212)		18 days ago	
wrappers	Adds indexing operation to Metric class (#142)		2 months ago	
__init__.py	minor refactor tests (#21)		3 months ago	
test_utilities.py	more tests		2 months ago	

Test example 1



Can be simple

```
def test_warning_on_nan(tmpdir):  
    preds = torch.randint(3, size=(20, ))  
    target = torch.randint(3, size=(20, ))  
  
    with pytest.warns(  
        UserWarning,  
        match='.* nan values found in confusion matrix have been replaced with zeros.',  
    ):  
        confusion_matrix(preds, target, num_classes=5, normalize='true')
```

Test example 2



Can be very complex

```
@pytest.mark.parametrize("normalize", ['true', 'pred', 'all', None])
@pytest.mark.parametrize(
    "preds, target, sk_metric, num_classes, multilabel",
    [
        (_input_binary_prob.preds, _input_binary_prob.target, _sk_cm_binary_prob, 2, False),
        (_input_binary_logits.preds, _input_binary_logits.target, _sk_cm_binary_prob, 2, False),
        (_input_binary.preds, _input_binary.target, _sk_cm_binary, 2, False),
        (_input_mlb_prob.preds, _input_mlb_prob.target, _sk_cm_multilabel_prob, NUM_CLASSES, True),
        (_input_mlb_logits.preds, _input_mlb_logits.target, _sk_cm_multilabel_prob, NUM_CLASSES, True),
        (_input_mlb.preds, _input_mlb.target, _sk_cm_multilabel, NUM_CLASSES, True),
        (_input_mcls_prob.preds, _input_mcls_prob.target, _sk_cm_multiclass_prob, NUM_CLASSES, False),
        (_input_mcls_logits.preds, _input_mcls_logits.target, _sk_cm_multiclass_prob, NUM_CLASSES, False),
        (_input_mcls.preds, _input_mcls.target, _sk_cm_multiclass, NUM_CLASSES, False),
        (_input_mdmc_prob.preds, _input_mdmc_prob.target, _sk_cm_multidim_multiclass_prob, NUM_CLASSES, False),
        (_input_mdmc.preds, _input_mdmc.target, _sk_cm_multidim_multiclass, NUM_CLASSES, False)]
)
class TestConfusionMatrix(MetricTester):

    @pytest.mark.parametrize("ddp", [True, False])
    @pytest.mark.parametrize("dist_sync_on_step", [True, False])
    def test_confusion_matrix(
        self, normalize, preds, target, sk_metric, num_classes, multilabel, ddp, dist_sync_on_step
    ):
        self.run_class_metric_test(
            ddp=ddp,
            preds=preds,
            target=target,
            metric_class=ConfusionMatrix,
            sk_metric=partial(sk_metric, normalize=normalize),
            dist_sync_on_step=dist_sync_on_step,
            metric_args={
                "num_classes": num_classes,
                "threshold": THRESHOLD,
                "normalize": normalize,
                "multilabel": multilabel
            }
        )
```

Setup files



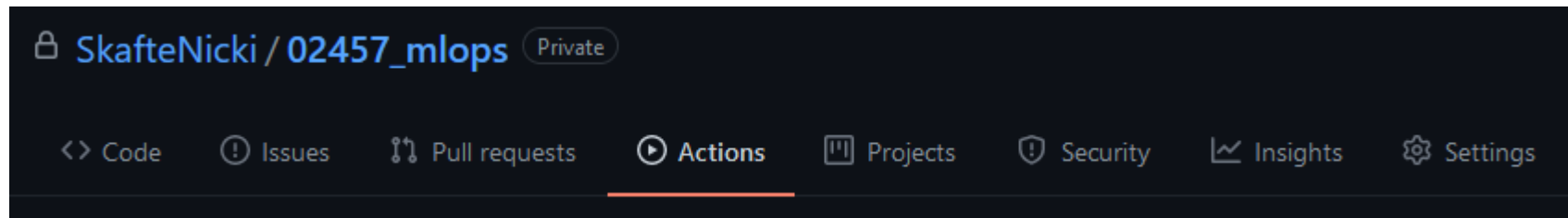
- Contains all information regarding the project
- Allow people to do:
python setup.py install
or if uploaded to pip
pip install my_package

```
setup(
    name='torchmetrics',
    version=about.__version__,
    description=about.__docs__,
    author=about.__author__,
    author_email=about.__author_email__,
    url=about.__homepage__,
    download_url=os.path.join(about.__homepage__, 'archive', 'master.zip'),
    license=about.__license__,
    packages=find_packages(exclude=['tests', 'docs']),
    long_description=long_description,
    long_description_content_type='text/markdown',
    include_package_data=True,
    zip_safe=False,
    keywords=['deep learning', 'machine learning', 'pytorch', 'metrics', 'AI'],
    python_requires='>=3.6',
    setup_requires=[],
    install_requires=setup_tools._load_requirements(_PATH_ROOT),
    project_urls={
        "Bug Tracker": os.path.join(about.__homepage__, 'issues'),
        "Documentation": "https://torchmetrics.rtfd.io/en/latest/",
        "Source Code": about.__homepage__,
    },
    classifiers=[
        'Environment :: Console',
        'Natural Language :: English',
        # How mature is this project? Common values are
        # 3 - Alpha, 4 - Beta, 5 - Production/Stable
        'Development Status :: 3 - Alpha',
        # Indicate who your project is intended for
        'Intended Audience :: Developers',
        'Topic :: Scientific/Engineering :: Artificial Intelligence',
        'Topic :: Scientific/Engineering :: Image Recognition',
        'Topic :: Scientific/Engineering :: Information Analysis',
        # Pick your license as you wish
        # 'License :: OSI Approved :: BSD License',
        'Operating System :: OS Independent',
        # Specify the Python versions you support here. In particular, ensure
        # that you indicate whether you support Python 2, Python 3 or both.
        'Programming Language :: Python :: 3',
        'Programming Language :: Python :: 3.6',
        'Programming Language :: Python :: 3.7',
        'Programming Language :: Python :: 3.8',
        'Programming Language :: Python :: 3.9',
    ],
)
```

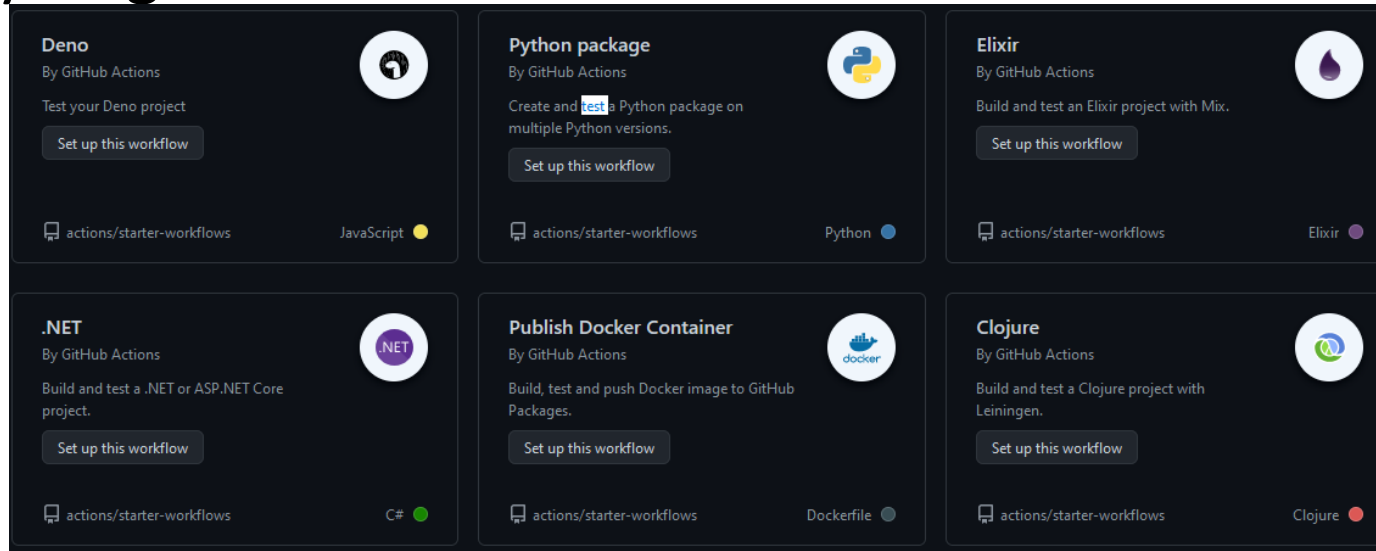

Github actions



- Build-in CI for github
- Free 2,000 automation minutes/month (public repository)



- Many ready to go workflows



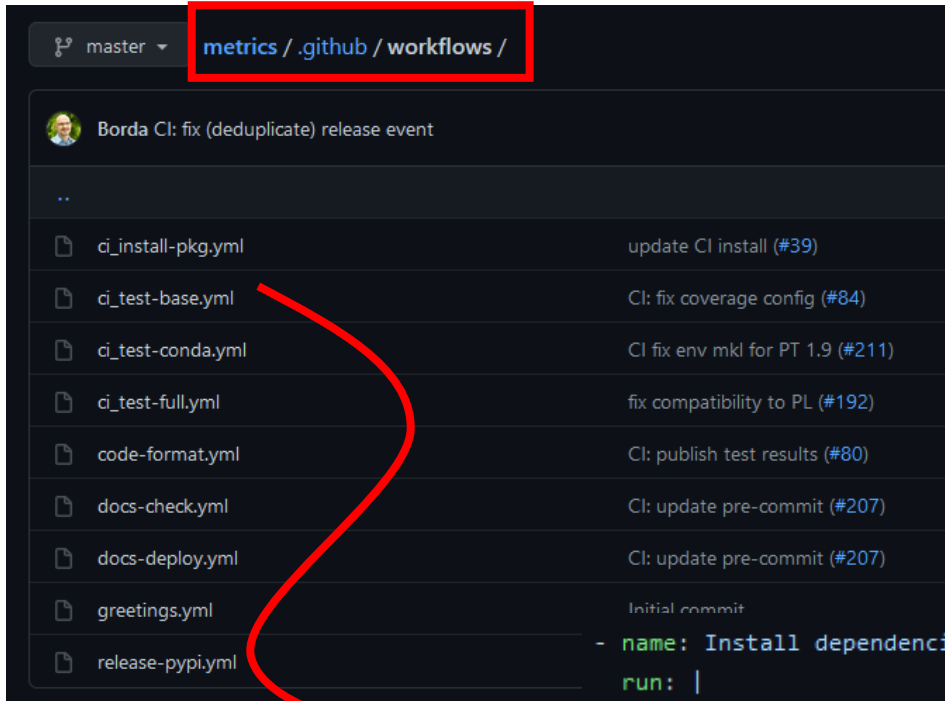
The anatomy of a workflow file



- When tests should be triggered
- Define Operating system + python version
- Setup python
- Install dependencies
- Check linting (stop if errors)
- Run tests

```
02457 mlops / .github / workflows / python-package.yml in main
<> Edit new file Preview
1 name: Python package
2 on:
3   push:
4     branches: [ main ]
5   pull_request:
6     branches: [ main ]
7 jobs:
8   build:
9     runs-on: ubuntu-latest
10    strategy:
11      fail-fast: false
12      matrix:
13        python-version: [3.7, 3.8, 3.9]
14    steps:
15      - uses: actions/checkout@v2
16      - name: Set up Python ${ matrix.python-version }
17        uses: actions/setup-python@v2
18        with:
19          python-version: ${ matrix.python-version }
20      - name: Install dependencies
21        run: |
22          python -m pip install --upgrade pip
23          python -m pip install flake8 pytest
24          if [ -f requirements.txt ]; then pip install -r requirements.txt; fi
25      - name: Lint with flake8
26        run: |
27          # stop the build if there are Python syntax errors or undefined names
28          flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics
29          # exit-zero treats all errors as warnings. The GitHub editor is 127 chars wide
30          flake8 . --count --exit-zero --max-complexity=10 --max-line-length=127 --statistics
31      - name: Test with pytest
32        run: |
33          pytest
```

A small case study



```
Initial commit
- name: Install dependencies
  run: |
    python -m pip install --upgrade --user pip
    pip install --requirement ./requirements.txt --find-links https://download.pytorch.org/whl/cpu/torch_stable.html
    pip install "pytest>6.0" "pytest-cov>2.10" --upgrade-strategy only-if-needed
    python --version
    pip --version
    pip list
  shell: bash

- name: Test Package [only]
  run: |
    # NOTE: run coverage on tests does not propagate faler status for Win, https://github.com/nedbat/coveragepy/issues/1003
    python -m pytest torchmetrics -v --cov=torchmetrics --junitxml=junit/test-results-${{ runner.os }}-${{ matrix.python-version }}.xml
```

All of our workflows – 36 in total



✓ CI testing - complete
on: pull_request

✓ pytest (ubuntu-20.04, 3.6, minimal)

✓ pytest (ubuntu-20.04, 3.8, minimal)

✓ pytest (ubuntu-20.04, 3.8, latest)

✓ pytest (ubuntu-20.04, 3.9, latest)

✓ pytest (macOS-10.15, 3.6, minimal)

✓ pytest (macOS-10.15, 3.8, minimal)

✓ pytest (macOS-10.15, 3.8, latest)

✓ pytest (macOS-10.15, 3.9, latest)

✓ pytest (windows-2019, 3.6, mini...)

✓ pytest (windows-2019, 3.8, mini...)

✓ pytest (windows-2019, 3.8, latest)

✓ pytest (windows-2019, 3.9, latest)

✓ WIP

✓ WIP

✓ Check Code formatting
on: pull_request

✓ flake8

✓ imports-check-isort

✓ typing-check-mypy

✓ Docs check
on: pull_request

✓ test-docs

✓ make-docs

✓ CI testing - base
on: pull_request

✓ doctest (ubuntu-20.04, 3.7)

✓ doctest (windows-2019, 3.7)

✓ doctest (macOS-10.15, 3.7)

✓ Azure Pipelines

✗ PyTorchLightning.metrics Re-run

✓ Mergify

✓ Summary

✓ Codecov

✓ codecov/patch

✓ codecov/project

✓ PyTorch & Conda
on: pull_request

✓ conda (3.7, 1.3)

✓ conda (3.7, 1.4)

✓ conda (3.7, 1.5)

✓ conda (3.7, 1.6)

✓ conda (3.7, 1.7)

⚠ conda (3.7, 1.8)

✓ conda (3.7, 1.9)

Not github actions

✓ Install pkg
on: pull_request

✓ pkg-check

✓ pkg-install (ubuntu-20.04, 3.7)

✓ pkg-install (macOS-10.15, 3.7)

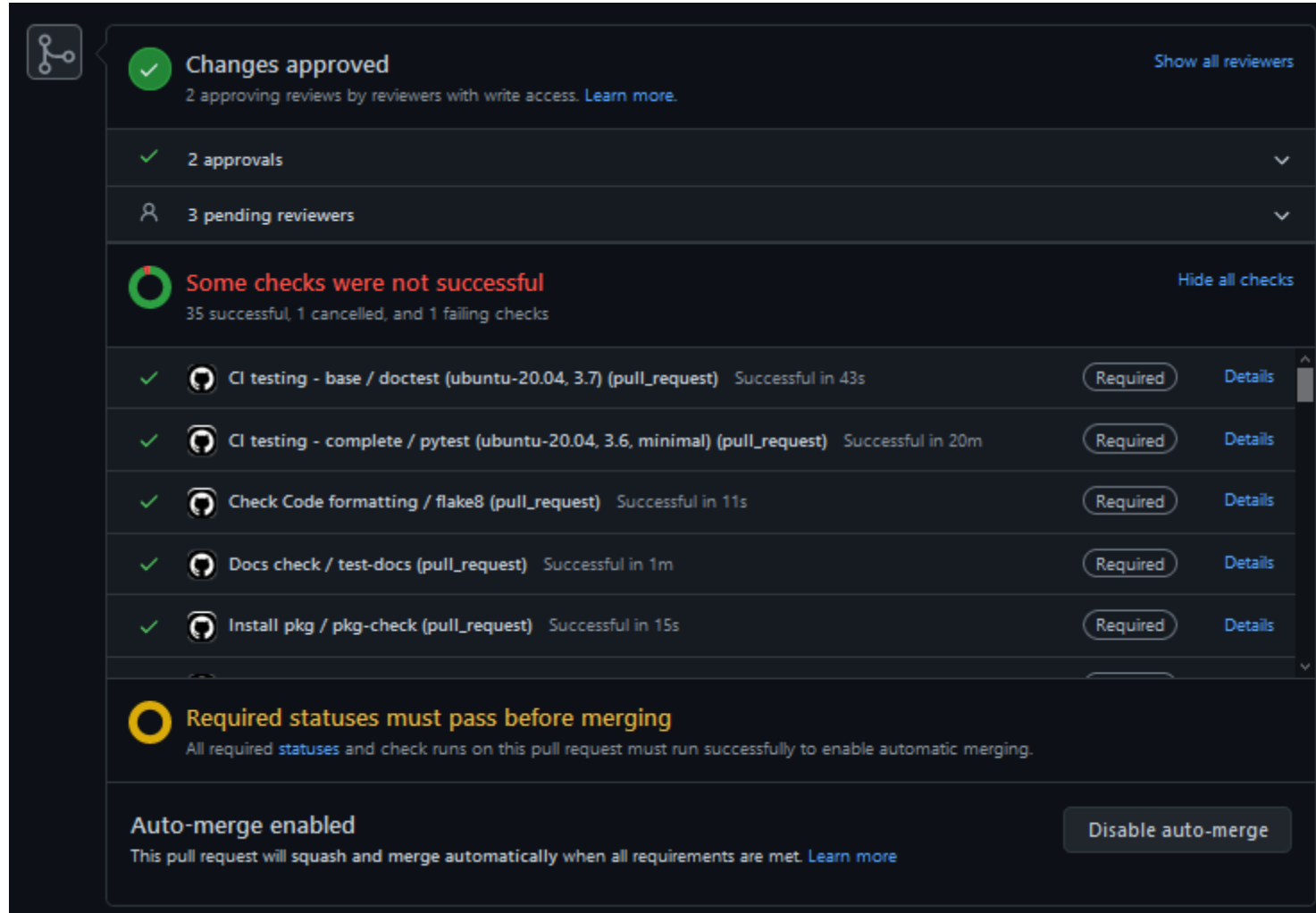
✓ pkg-install (windows-2019, 3.7)

Test combination of

- Hardware setup
- Operating system
- Python version
- Dependencies

Runs tests, docs, coverage, lintint, package install etc

Each PR triggers all tests



The screenshot shows a GitHub pull request interface. At the top, a green checkmark icon indicates 'Changes approved' with the text '2 approving reviews by reviewers with write access. [Learn more.](#)' and a 'Show all reviewers' link. Below this, a summary shows '2 approvals' and '3 pending reviews'. A green circle with a red dot indicates 'Some checks were not successful' with the text '35 successful, 1 cancelled, and 1 failing checks' and a 'Hide all checks' link. A list of checks follows, all marked as 'Successful':

- CI testing - base / doctest (ubuntu-20.04, 3.7) (pull_request) Successful in 43s (Required) [Details](#)
- CI testing - complete / pytest (ubuntu-20.04, 3.6, minimal) (pull_request) Successful in 20m (Required) [Details](#)
- Check Code formatting / flake8 (pull_request) Successful in 11s (Required) [Details](#)
- Docs check / test-docs (pull_request) Successful in 1m (Required) [Details](#)
- Install pkg / pkg-check (pull_request) Successful in 15s (Required) [Details](#)

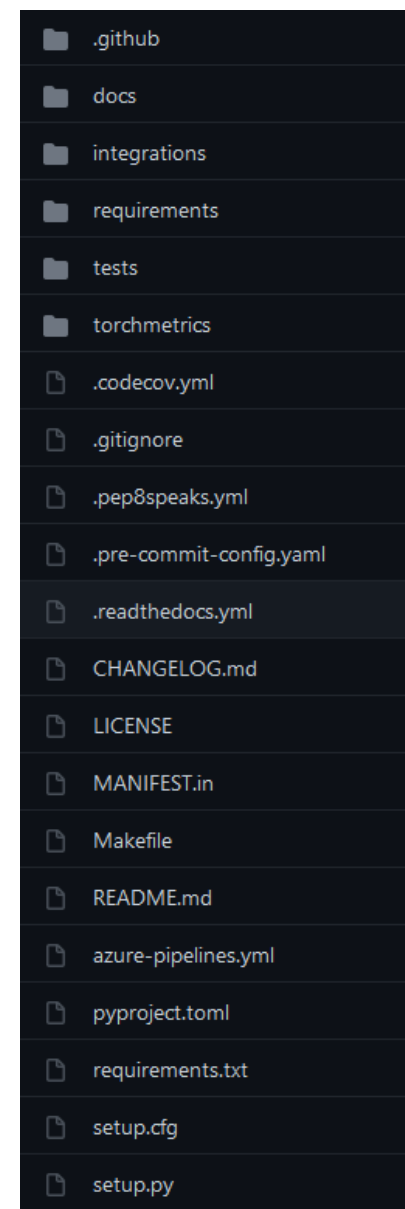
Below the checks, a yellow circle icon indicates 'Required statuses must pass before merging' with the text 'All required [statuses](#) and check runs on this pull request must run successfully to enable automatic merging.'

At the bottom, it states 'Auto-merge enabled' with the text 'This pull request will squash and merge automatically when all requirements are met. [Learn more](#)' and a 'Disable auto-merge' button.

Other files to consider



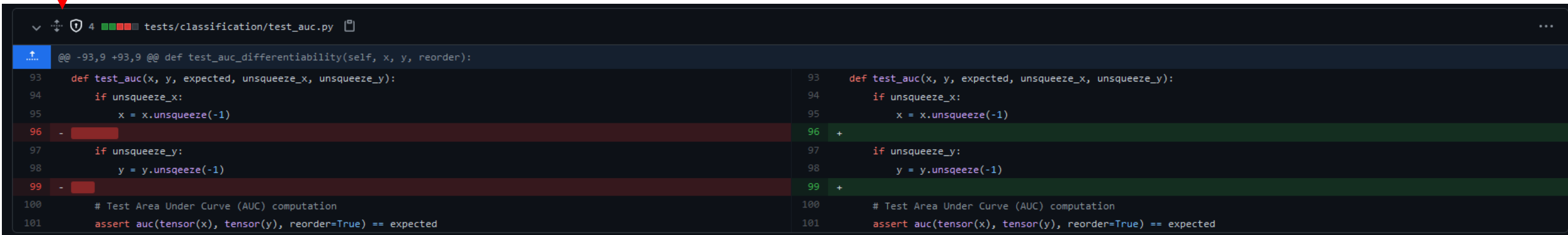
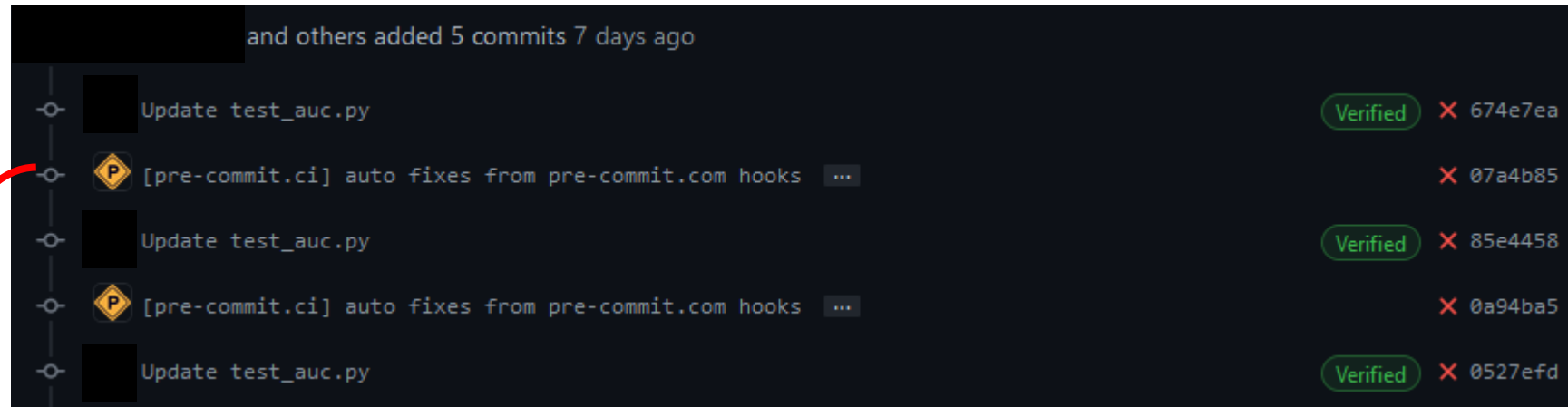
- requirements.txt
- LICENSE
- README.md
- .gitignore



Advanced: Use bots



Bots can take care of tedious task for you (like linting)



The future is here



Gabriele Petronella
@gabro27

So this just happened:

- a bot found a vulnerability in a dependency
- a bot sent a PR to fix it
- the CI verified the PR
- a bot merged it
- a bot celebrated the merge with a GIF

