

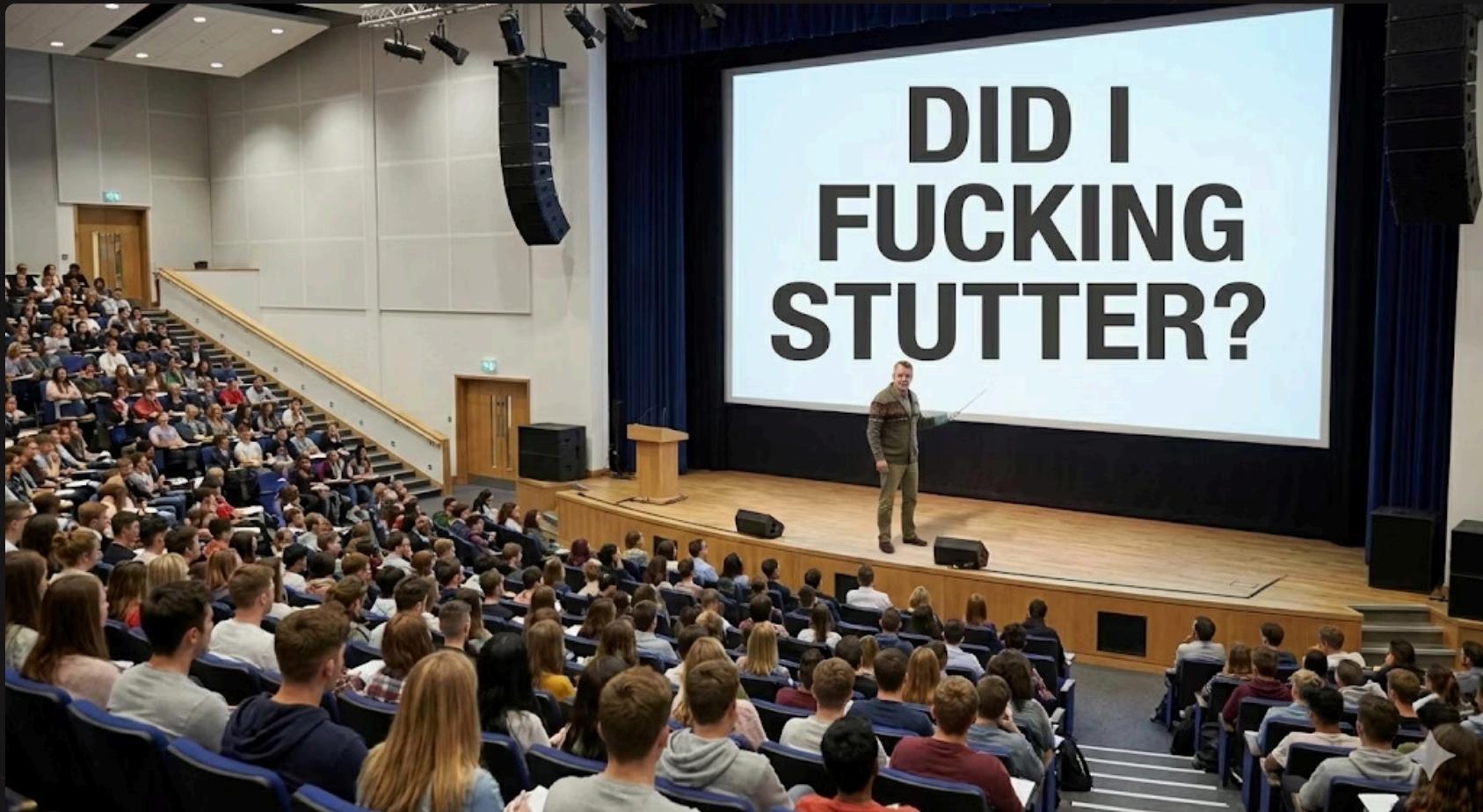
Day6 - Continuous Integration

02476 Machine Learning Operations

Nicki Skafte Detlefsen, Associate Professor, DTU Compute

January 2026

50 people and 5 groups are marked in my little black book...



Why you should care about today

3 years ago, the day before this lecture, the internet went down for a couple of hours because someone f..ked up their continues integration at [Fastly](#).

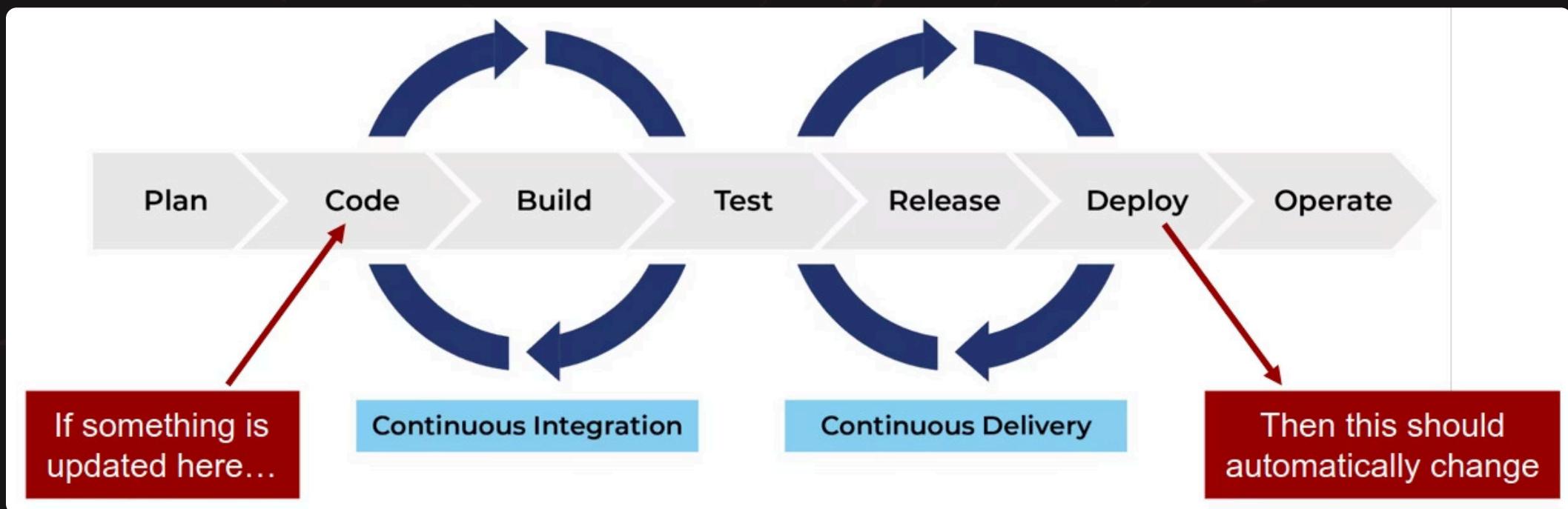
Dev at Fastly : I'll just push this small change to production

Dev at Fastly 2 seconds later:



Continues X

🔥 Term refers to a set of software practices for automating tedious tasks and make sure changes in a pipeline are continuously propagated through the pipeline



The different types

CI	CD	CML
Continues Integration	Continues Deployment	Continues Machine Learning
 How to automatically secure that code does not break during development?	 How to get your code/application to the user automatically? + monitor life cycle	 How to automatically retrain machine learning models when data and code changes?
 App independent concept	 App dependent concept	 Specific to ML applications

MLOps levels

The **Maturity model** overall describes the DevOps practices to run a successful MLOps environment.

Intended to identify gaps in an existing organization's attempt to implement such an environment.

- 💡 Estimate the scope of the work for new engagements.
- 💡 Establish realistic success criteria.
- 💡 Identify deliverables you'll hand over at the conclusion of the engagement.

Level	Description	Highlights	Technology
0	No MLOps	<ul style="list-style-type: none">Difficult to manage full machine learning model lifecycleThe teams are disparate and releases are painfulMost systems exist as "black boxes," little feedback during/post deployment	<ul style="list-style-type: none">Manual builds and deploymentsManual testing of model and applicationNo centralized tracking of model performanceTraining of model is manual
1	DevOps but no MLOps	<ul style="list-style-type: none">Releases are less painful than No MLOps, but rely on Data Team for every new modelStill limited feedback on how well a model performs in productionDifficult to trace/reproduce results	<ul style="list-style-type: none">Automated buildsAutomated tests for application code
2	Automated Training	<ul style="list-style-type: none">Training environment is fully managed and traceableEasy to reproduce modelReleases are manual, but low friction	<ul style="list-style-type: none">Automated model trainingCentralized tracking of model training performanceModel management
3	Automated Model Deployment	<ul style="list-style-type: none">Releases are low friction and automaticFull traceability from deployment back to original dataEntire environment managed: train > test > production	<ul style="list-style-type: none">Integrated A/B testing of model performance for deploymentAutomated tests for all codeCentralized tracking of model training performance
4	Full MLOps Automated Operations	<ul style="list-style-type: none">Full system automated and easily monitoredProduction systems are providing information on how to improve and, in some cases, automatically improve with new modelsApproaching a zero-downtime system	<ul style="list-style-type: none">Automated model training and testingVerbose, centralized metrics from deployed model

This lecture: continues integration

Core task:

🔥 How to automatically secure that code does not break during development? 🔥

3 steps to do this:

- 💡 Use version control: Frequently committing code to a shared repository
- 💡 Write (unit)test for your code: Should capture unwanted bugs in your code
- 💡 Automate build + testing: Automatically run test so code cannot be merged without working

A small case study for continuous integration

The screenshot shows the GitHub repository page for TorchMetrics. At the top, there's a table of recent commits:

File	Commit Message	Time Ago
MANIFEST.in	update CI	6 months ago
Makefile	rename tests/ (#1091)	5 months ago
README.md	Code cleaning after classification refactor 2/n (#1252)	7 days ago
pyproject.toml	CI: re-use checks (#1261)	2 months ago
requirements.txt	Set minimum pytorch version to 1.8 + cleanup (#1263)	2 months ago
setup.cfg	CI: re-use checks (#1261)	2 months ago
setup.py	CI: Enable testing with Python 3.10 (#1132)	5 months ago

Below the commits is the README.md file content:

```
README.md
```

TorchMetrics
Machine learning metrics for distributed, scalable PyTorch applications.

What is Torchmetrics • Implementing a metric • Built-in metrics • Docs • Community • License

python 3.7 | 3.8 | 3.9 | 3.10 | pypi package 0.10.3 | downloads 28M | conda v0.10.3 | downloads 585k | License Apache 2.0

CI testing - complete | Azure Pipelines succeeded | codecov 39%

slack chat | docs passing | DOI 10.5281/zenodo.5844769 | JOSS 10.21105/joss.04101 | pre-commit.ci passed

The right side of the screenshot shows the GitHub repository summary card with sections for Contributors (169) and Languages (Python 99.9%, Other 0.1%).

CI step 1: version control

User version control:

- 💡 Code changes are tracked
- 💡 Branches for parallel work

Commit frequently:

- 💡 Catch errors sooner than later
- 💡 Revert back easily to when things were working
- 💡 Merging can be done automatically

Create it → Break it → Fix it →...



The screenshot shows a GitHub repository's commit history for the 'docs' branch. It highlights frequent commits made by users SkafteNicki and Borda on March 10, 2021. The commits are organized into several logical steps or tasks:

- Initial Setup:** 'docs' (SkafteNicki), 'Commits on Mar 10, 2021'
- Contributing Updates:** 'contributing' (SkafteNicki), 'remove old changes' (SkafteNicki), 'Apply suggestions from code review' (SkafteNicki and Borda), 'move around' (SkafteNicki), 'merge' (SkafteNicki), 'merge' (SkafteNicki), 'Merge branch 'master' into contributing' (SkafteNicki), 'Update .github/CONTRIBUTING.md' (SkafteNicki and Borda)
- Automated Fixes:** 'remove' (SkafteNicki), 'Update .github/CONTRIBUTING.md' (SkafteNicki and Borda), 'Apply suggestions from code review' (Borda), 'from md to rst' (SkafteNicki), 'Merge branch 'contributing' of https://github.com/PyTorchLightning/me...' (SkafteNicki), 'Apply suggestions from code review' (Borda), 'Apply suggestions from code review' (Borda), 'Apply suggestions from code review' (Borda)

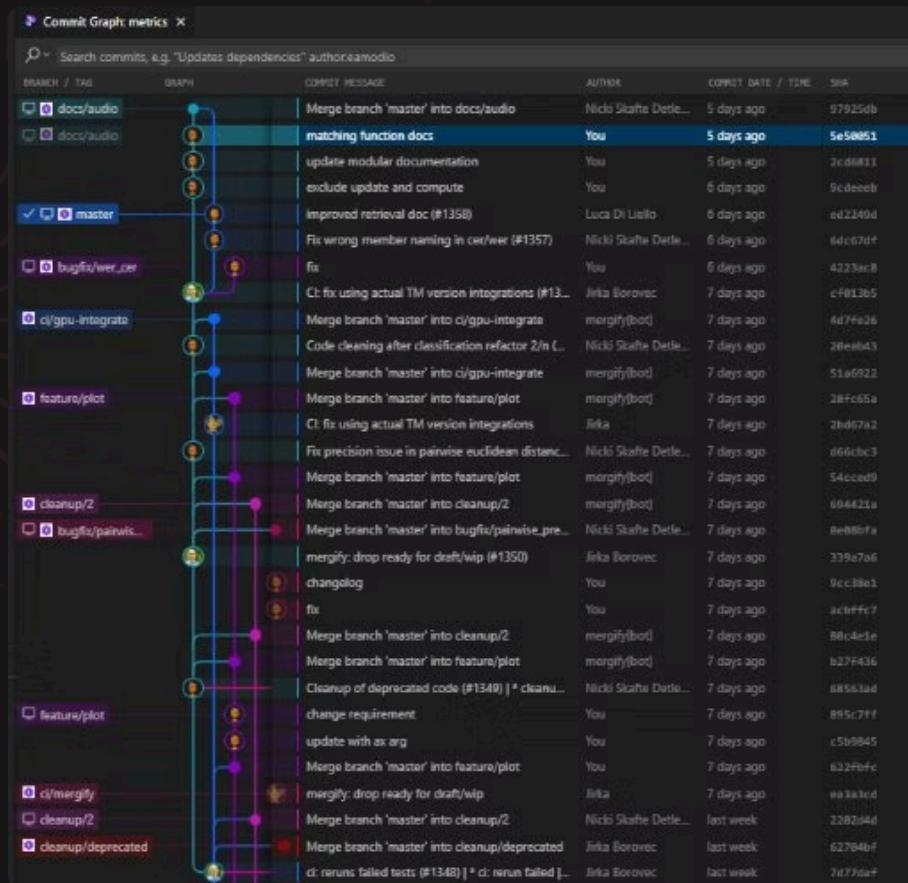
CI step 1: Use branches

Parallel workflow

Experimental features changes are kept away from master/main

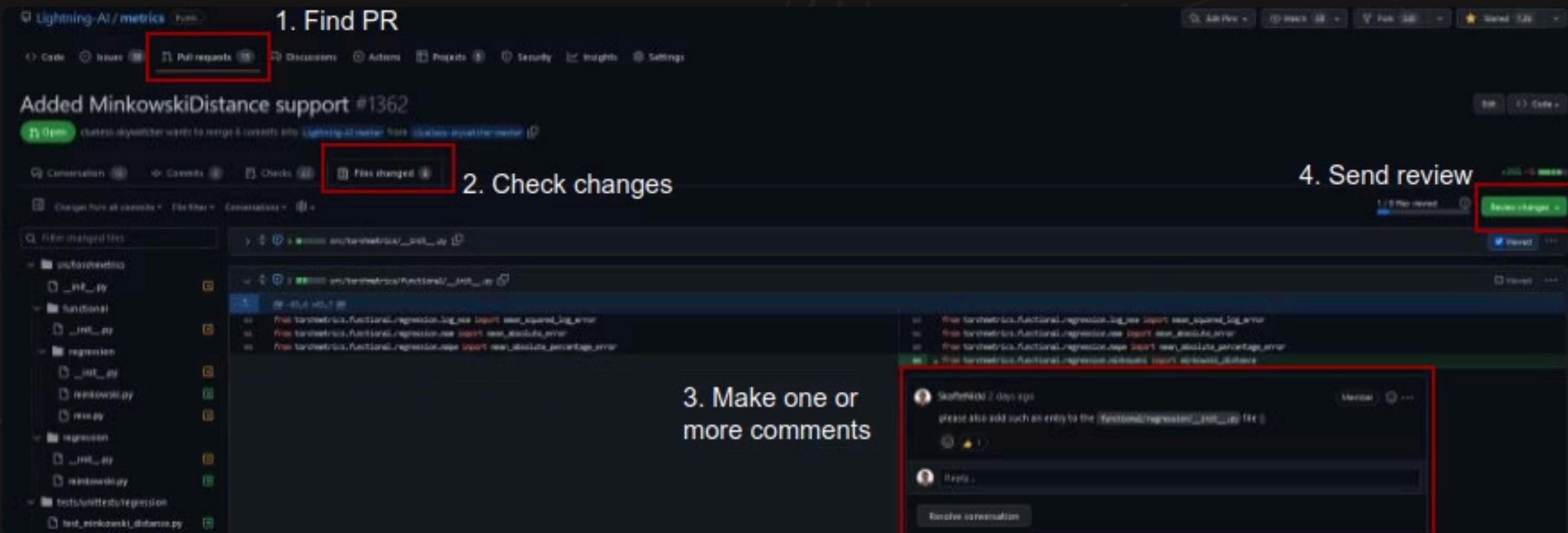
Recommend extensions for VS code:

- GitLens
- Git Graph
- GitHub Pull Requests



CI step 1: Use pull requests

⚠️ No commit can be pushed to master without being in a pull request



CI step 1: pre-commit

✓ Check that everything is up to standard before commits are created

```
! .pre-commit-config.yaml X
! .pre-commit-config.yaml
1 default_language_version:
2   python: python3
3
4 repos:
5   - repo: https://github.com/pre-commit/pre-commit-hooks
6     rev: v4.4.0
7     hooks:
8       - id: end-of-file-fixer
9       - id: trailing whitespace
10      # - id: check-json
11      # - id: check-yaml
12      - id: check-toml
13      - id: check-docstring-first
14      - id: check-executables-have-shebangs
15      - id: check-case-conflict
16      - id: detect-private-key
17
18   - repo: https://github.com/astral-sh/ruff-pre-commit
19     rev: v0.1.3
20     hooks:
21       - id: ruff
22         args: [--fix, --exit-non-zero-on-fix]
23
24   - repo: https://github.com/astral-sh/ruff-pre-commit
25     rev: v0.1.3
26     hooks:
27       - id: ruff-format
28
29   - repo: https://github.com/codespell-project/codespell
30     rev: v2.2.5
31     hooks:
32       - id: codespell
33         additional_dependencies: [tomli]
```

```
dtu_mllops on 🐫 main [!?!] via 🐫 v3.11.5 🐫 mllops
> git commit -m "implementation of client"
fix end of files..... Failed
- hook id: end-of-file-fixer
- exit code: 1
- files were modified by this hook

Fixing s8_monitoring/exercise_files/client.py

trim trailing whitespace..... Passed
check toml..... (no files to check) Skipped
check docstring is first..... Passed
check that executables have shebangs..... Passed
check for case conflicts..... Passed
detect private key..... Passed
ruff..... Failed
- hook id: ruff
- exit code: 1
- files were modified by this hook

s8_monitoring\exercise_files\client.py:17:12: S113 Probable use of requests call without timeout
Found 2 errors (1 fixed, 1 remaining).

ruff-format..... Passed
codespell..... Passed
markdownlint-docker..... (no files to check) Skipped
```

CI step 2: write tests

Tests are the cornerstones of continuous integration

💡 *unit tests* are arguably the most important.

💡 A single unittest, tests a small part of your code

💡 By testing code in small pieces, bugs are easier to find

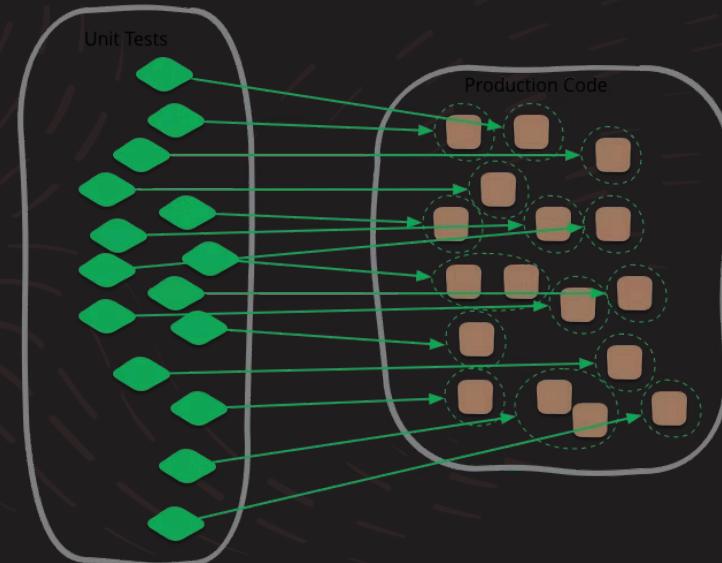
Other test types worth considering:

🔥 Integration tests

🔥 Regression tests

🔥 Performance tests

🔥 Security tests



CI step 2: write tests

💡 By Python convention your source code should either be

src/<project_name>

(src-layout)

or

<project_name>

(flat-layout)

metrics / src / torchmetrics /	
SkafteNicki	Fix wrong member naming in cer/ver (#1357)
	6 days ago
..	
audio	CI re-use checks (#1261)
	2 months ago
classification	Code cleaning after classification refactor 2/m (#1252)
	7 days ago
detection	Revert mAP matching vectorization (#1327)
	19 days ago
functional	Code cleaning after classification refactor 2/m (#1252)
	7 days ago
image	Add normalize argument to certain image metrics (#1246)
	21 days ago
multimodal	Add CLIP score (#1314)
	11 days ago
nominal	Add Theil's U Statistic (Uncertainty) Metric (#1337)
	10 days ago
regression	Add LogCosh Error (#1316)
	21 days ago
retrieval	CI re-use checks (#1261)
	2 months ago
text	Fix wrong member naming in cer/ver (#1357)
	6 days ago
utilities	Code cleaning after classification refactor 2/m (#1252)
	7 days ago
wrappers	Code cleaning after classification refactor 2/m (#1252)
	7 days ago
__about__.py	docs: Update the number of metrics + covered domains (#1344)
	11 days ago
__init__.py	Exact match multiclass (#1343)
	8 days ago
aggregation.py	move: torchmetrics >> src/
	6 months ago
collections.py	Code cleaning after classification refactor 2/m (#1252)
	7 days ago
metric.py	Cleanup of deprecated code (#1349)
	7 days ago
py.typed	move: torchmetrics >> src/
	6 months ago

CI step 2: write tests

For tests, the convention is to either place the tests in a separate tests folder, or put the tests in the same folder as the function/class/submodule they are testing.

```
|── README.md  
|── src/  
|   |── __init__.py  
|   |── important_functions.py  
|── tests/  
|   |── __init__.py  
|   |── test_important_functions.py
```

```
|── README.md  
|── src/  
|   |── __init__.py  
|   |── submodule/  
|       |── __init__.py  
|       |── important_functions.py  
|       |── test_important_functions.py
```

master - metrics / tests / unittests /	
...	
audio	Set minimum pytorch version to 1.8 + cleanup (#1263) 2 months ago
bases	Code cleaning after classification refactor 2/n (#1252) 7 days ago
classification	Cleanup of deprecated code (#1349) 7 days ago
detection	Revert mAP matching vectorization (#1327) 19 days ago
helpers	Add CLIP score (#1314) 11 days ago
image	Add normalize argument to certain image metrics (#1246) 21 days ago
multimodal	Add CLIP score (#1314) 11 days ago
nominal	Add Thiel's U Statistic (Uncertainty) Metric (#1337) 10 days ago
pairwise	Fix precision issue in pairwise euclidean distance (#1352) 7 days ago
regression	Add LogCosh Error (#1316) 21 days ago
retrieval	[Refactor] Classification 1/n (#1054) 3 months ago
text	Bugfix: Update message regarding connection issues to the HF hub (#1141) 5 months ago
utilities	Set minimum pytorch version to 1.8 + cleanup (#1263) 2 months ago
wrappers	Code cleaning after classification refactor 2/n (#1252) 7 days ago
__init__.py	rename tests/ (#1091) 6 months ago

CI step 2: write tests

💡 In python, we recommend using the **pytest** framework.

💡 Test are simple functions that start with *test_* and uses *assert*

```
import torch
from torch.nn.functional import mse_loss

def test_mse_loss_zeros():
    # (0 - 0)**2 = 0
    assert mse_loss(torch.zeros(1,), torch.zeros(1,)) == 0

def test_mse_loss_ones():
    # (1 - 0)**2 = 1
    assert mse_loss(torch.ones(1,), torch.zeros(1,)) == 0
```

CI step 2: write tests

Test 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')
```

CI step 2: write test

or complicated

```
@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_mccls_prob preds, _input_mccls_prob.target, _sk_cm_multiclass_prob, NUM_CLASSES, False),
     (_input_mccls_logits preds, _input_mccls_logits.target, _sk_cm_multiclass_prob, NUM_CLASSES, False),
     (_input_mccls preds, _input_mccls.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
            }
        )
```

Exercise 1: The data contract

Imagine you are building a model to predict **House Prices**. Aside from 'the code doesn't crash,' list 3 automated checks you would run on the **input data** before training starts to ensure the data is healthy."

Example dataset:



 www.kaggle.com

Housing Prices Dataset

Housing Prices Prediction – Regression Problem



Solution: Exercise 1 - The Data Contract

Automated checks for healthy input data before model training:



Data Type & Range Validation

Confirm numerical features like 'square footage' and 'number of bedrooms' fall within plausible ranges (e.g., non-negative) and maintain their expected data types.



Missing Values Threshold

Verify that no essential columns (e.g., 'price', 'address') contain missing values above a predefined, acceptable percentage, preventing incomplete data from skewing results.



Categorical Consistency

Validate categorical features such as 'neighborhood' or 'property type' against a list of known, expected values to catch typos or new, unhandled categories.

Exercise 2: The Smoke Test

You've just finished training a new model version in your CI pipeline. What is the very first thing you should do with that model file before saving it to a Model Registry? Write down one 'sanity check'.

Solution: Exercise 2 - The Smoke Test

Model Output Validation

Load the newly trained model and perform a prediction on a small, known-good data sample. Verify that the model's output is not empty or erroneous, its data type matches expectations (e.g., float, integer), and it falls within a sensible range for the problem (e.g., a positive house price, a probability between 0 and 1).

Exercise 3: ML Bug Hunting

A developer pushes code that passes all unit tests, but the model's accuracy on the test set drops from 95% to 40%. Discuss with your partner: Should the CI pipeline 'fail' this build? How would you write a test to catch this automatically?"

Solution: Exercise 3 - ML Bug Hunting

Implement a Model Performance Test

Introduce an automated regression test in the CI pipeline. This test should:

- Load the newly trained model.
- Evaluate its performance on a stable, versioned test dataset.
- Compare key metrics (e.g., accuracy, F1-score) against a predefined threshold (e.g., accuracy $\geq 90\%$).
- Fail the build if performance falls below the threshold, blocking deployment.

CI step 2: execute locally

```
(lightning) C:\Users\nsde\Documents\metrics>pytest tests\unittests\regression\test_mean_error.py
=====
platform win32 -- Python 3.8.13, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: C:\Users\nsde\Documents\metrics, configfile: setup.cfg
plugins: cov-4.0.0, doctestplus-0.12.1, timeout-2.1.0
collected 116 items

tests\unittests\regression\test_mean_error.py sssssssssssss.....ssssssssssssss.....xxxxxxxxx.....w
=====
..\\.\Anaconda3\envs\lightning\lib\site-packages\_pytest\config\_init__.py:1183
  C:\Users\nsde\Anaconda3\envs\lightning\lib\site-packages\_pytest\config\_init__.py:1183: PytestDeprecationWarning: The --strict option is deprecated, use --strict-markers instead.
    self.issue_config_time_warning(
-- Docs: https://docs.pytest.org/en/stable/warnings.html
=====
88 passed, 28 skipped, 8 xfailed, 1 warning in 16.44s =====
```

- Test passed
- ✗ Test failed
- ⠚ Test skipped (`pytest.skipif`, `pytest.skip`)
- ✗ Test was expected to fail (`pytest.xfail`)

Do you remember to do this before each commit?

Let's automate doing it instead

CI step 3: Automating stuff

What can be automated: EVERYTHING 

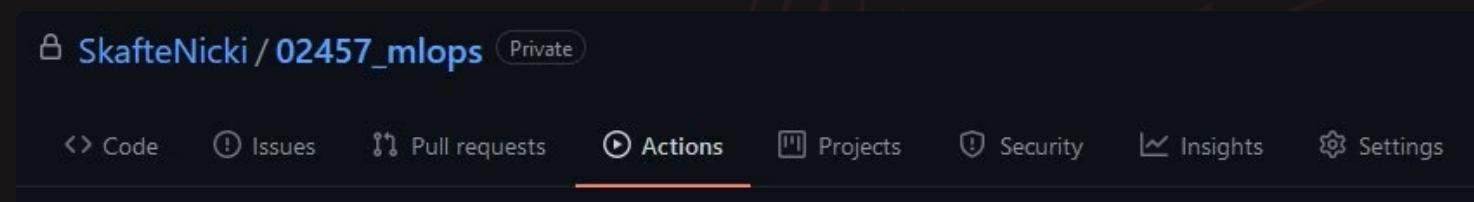
- 💡 Unit testing
- 💡 Integration testing
- 💡 Documentation creation
- 💡 Linters (style formatting)
- 💡 Security checks
- 💡 Code coverage
- 💡 Custom checks...

Only your imagination is the limit...

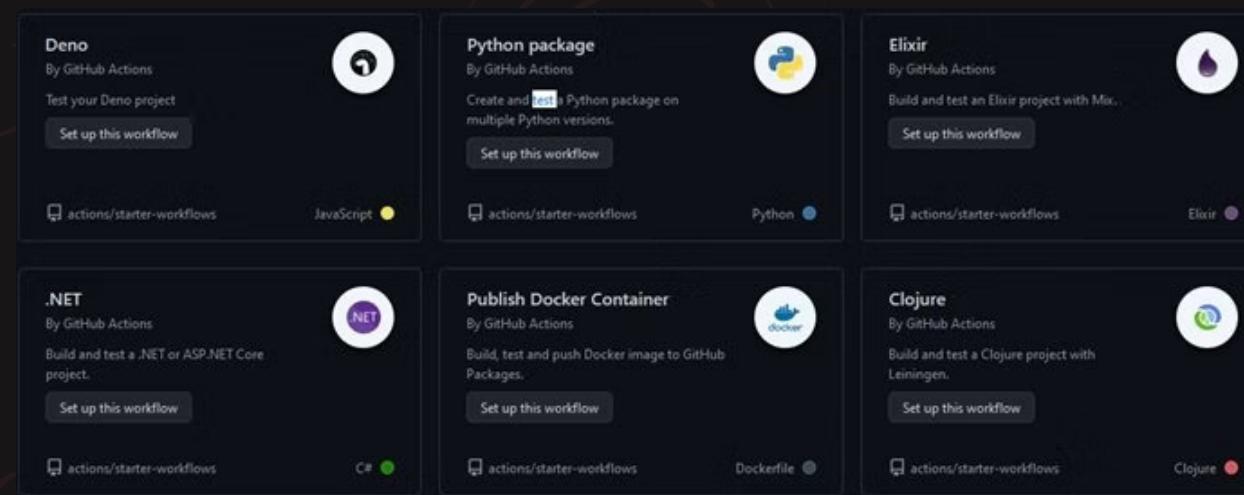
CI step 3: Github actions

Build-in continuous integration in Github.

Free 2000 automation minutes/month (public repository)



Many ready to go workflows



CI step 3: workflow files

Workflow files are a set of instructions that should be executed on a virtual machine hosted by Github

You can have one or many workflow files (runs in parallel)

When should workflow be triggered

Define OS + python

Clones code

Setup Python

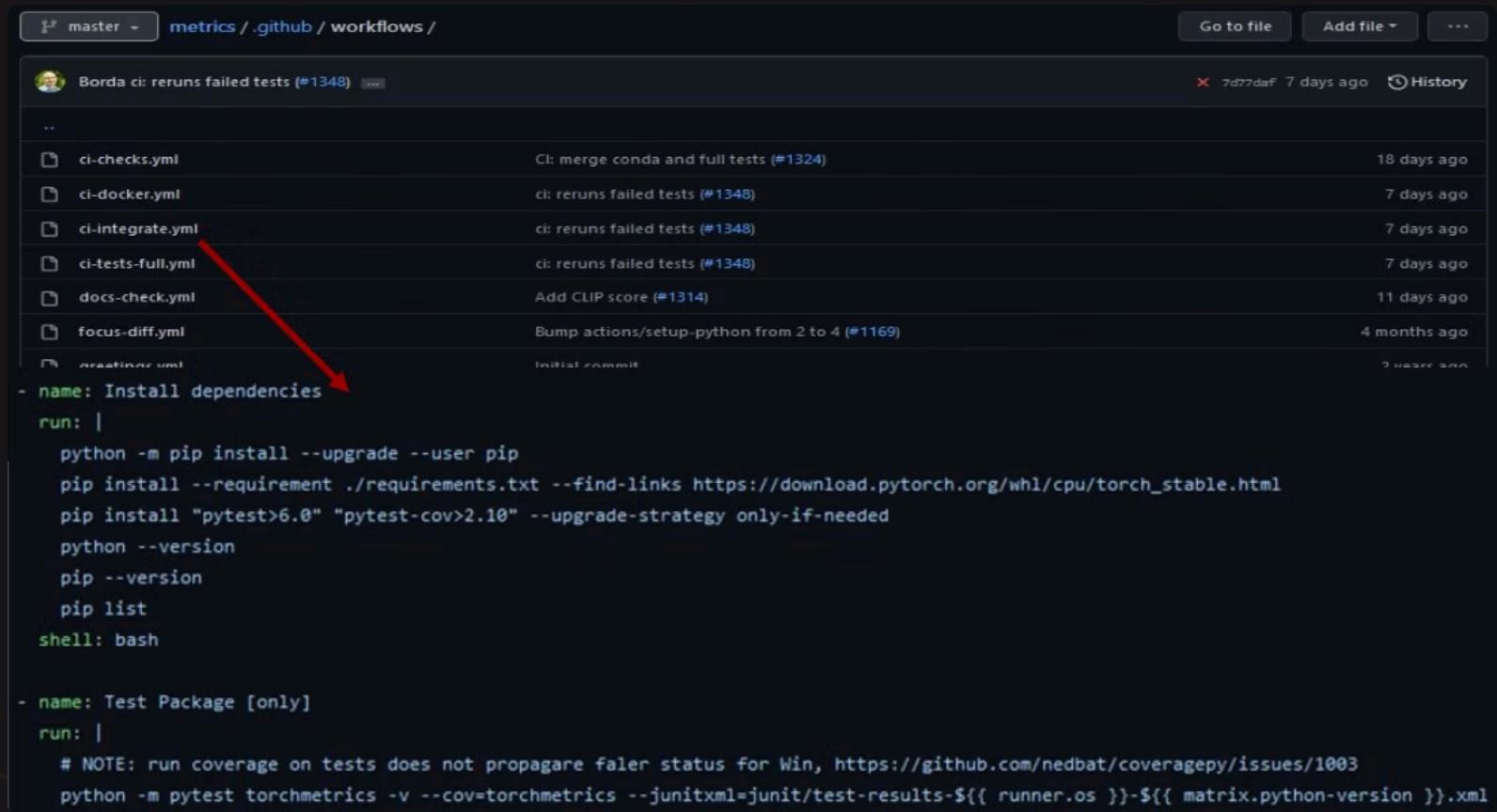
Install dependencies

Check formatting

Run tests

```
1 name: Python package
2
3   on:
4     push:
5       branches: [ main ]
6     pull_request:
7       branches: [ main ]
8
9   jobs:
10    build:
11      runs-on: ubuntu-latest
12      strategy:
13        matrix:
14          python-version: ["3.7", "3.8", "3.9", "3.10"]
15
16    steps:
17      - uses: actions/checkout@v3
18      - name: Set up Python ${{ matrix.python-version }}
19        uses: actions/setup-python@v4
20        with:
21          python-version: ${{ matrix.python-version }}
22      - name: Install dependencies
23        run:
24          - python -m pip install --upgrade pip
25          - pip install flake8 pytest
26          - pip install -r requirements.txt
27          - python setup.py install
28
29      - name: Lint with flake8
30        run:
31          - flake8 src/
32
33      - name: Test with pytest
34        run:
35          - pytest tests/
```

CI step 3: workflow files



The screenshot shows a GitHub repository interface with the path `metrics/.github/workflows/`. The list of files includes:

- ci-checks.yml (CI: merge conda and full tests (#1324), 18 days ago)
- ci-docker.yml (ci: reruns failed tests (#1348), 7 days ago)
- ci-integrate.yml (ci: reruns failed tests (#1348), 7 days ago)
- ci-tests-full.yml (ci: reruns failed tests (#1348), 7 days ago)
- docs-check.yml (Add CLIP score (#1314), 11 days ago)
- focus-diff.yml (Bump actions/setup-python from 2 to 4 (#1169), 4 months ago)
- create-new.yml (initial commit, 2 years ago)

A red arrow points to the `name: Install dependencies` section of the `ci-tests-full.yml` file, which contains the following YAML code:

```
- 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 failure 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
```

CI step 3: workflow files

✓ 43 checks in total

Test a combination of

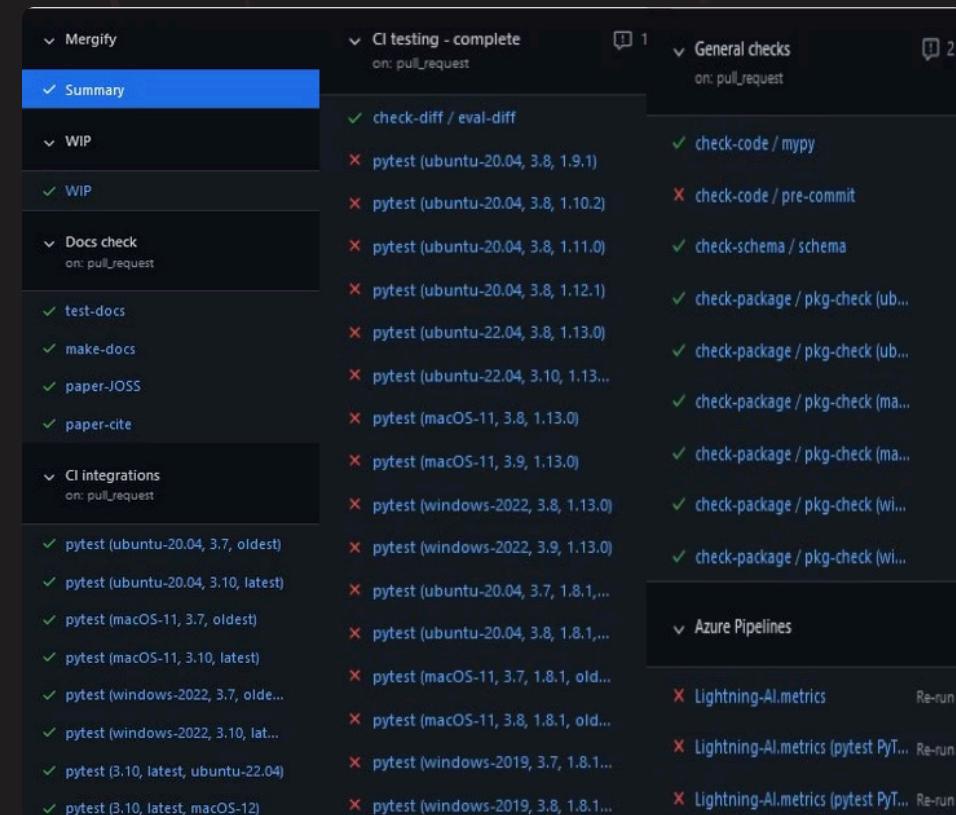
💡 Hardware setup

💡 Operating system

💡 Python version

💡 Core dependencies

Runs unit tests, build documentation, test coverage, linting of code, package installer etc.

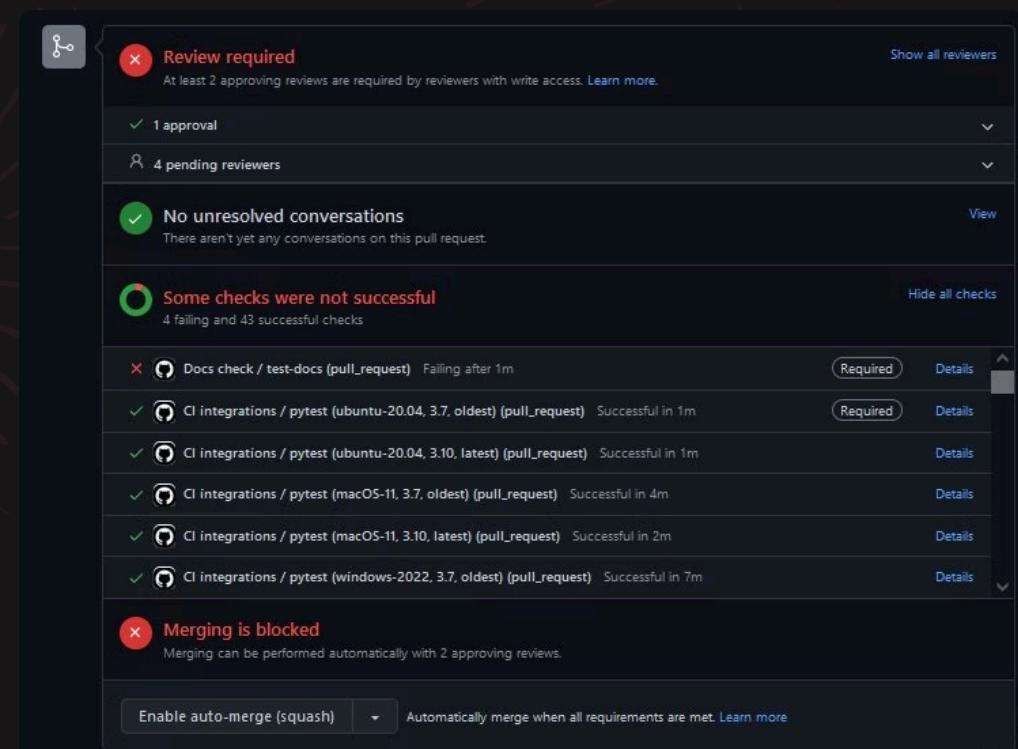


CI step 3: Code is checked before merging

Branch protection rules:

- ⚠ All/some tests should pass
- ⚠ At least x core members need to approve
- ⚠ Comments should be taken care of

View more [here](#)



CI step 3: Automate tedious tasks with bots

The screenshot shows a GitHub commit history and a code diff tool interface.

Commit History:

- justusschock and others added 5 commits 7 days ago
 - Update test_auc.py (Verified) 674e7ea
 - [pre-commit.ci] auto fixes from pre-commit.com hooks ... (X) 07a4b85
 - Update test_auc.py (Verified) 85e4458
 - [pre-commit.ci] auto fixes from pre-commit.com hooks ... (X) 0a94ba5
 - Update test_auc.py (Verified) 0527efd

Code Diff:

```
diff --git a/tests/classification/test_auc.py b/tests/classification/test_auc.py
@@ -93,9 +93,9 @@ def test_auc_differentiability(self, x, y, reorder):
93     def test_auc(x, y, expected, unsqueeze_x, unsqueeze_y):
94         if unsqueeze_x:
95             x = x.unsqueeze(-1)
96 -        if unsqueeze_y:
97             y = y.unsqueeze(-1)
98 -    # Test Area Under Curve (AUC) computation
99 -    assert auc(tensor(x), tensor(y), reorder=True) == expected
100 +        if unsqueeze_y:
101             y = y.unsqueeze(-1)
102 +    # Test Area Under Curve (AUC) computation
103 +    assert auc(tensor(x), tensor(y), reorder=True) == expected
```

CI step 3: Automate tedious tasks with bots

🤖 Dependabot can help auto checking new releases of dependencies in your project

The screenshot shows a GitHub pull request and its corresponding configuration file.

Pull Request: build(deps): update kornia requirement from <0.7.1,>=0.6.7 to >=0.6.7, <0.7.2 in /requirements #2293

Dependabot Comment: Updates the requirements on `main` to permit the latest version.

Code Snippet (requirements/image_test.txt):

```
@@ -2,7 +2,7 @@
 2 # in case you want to preserve/enforce restrictions on the latest
 3 # compatible version, add "strict" as an in-line comment
 4
 5 scikit-image >=0.19.0, <=0.21.0
 6 kornia >=0.6.7, <0.7.1
 7 pytorch-msssim ==1.0.0
 8 sewar >=0.4.4, <=0.4.6
 9 numpy <1.25.0
```

.github/dependabot.yml:

```
version: 2
updates:
  - package-ecosystem: "pip"
    directory: "/requirements"
    schedule:
      interval: "weekly"
```

Summary of continues integration

Use version control



Write (unit-)test for your code



Automate build + test



The agents are here

Agents can help you solve issues, review code and solve bugs...

SkaftNicki/dtu_mlops

#480 [bug] CLI exercise classifier dataset mismatch

0 comments

iarata opened on January 6, 2026



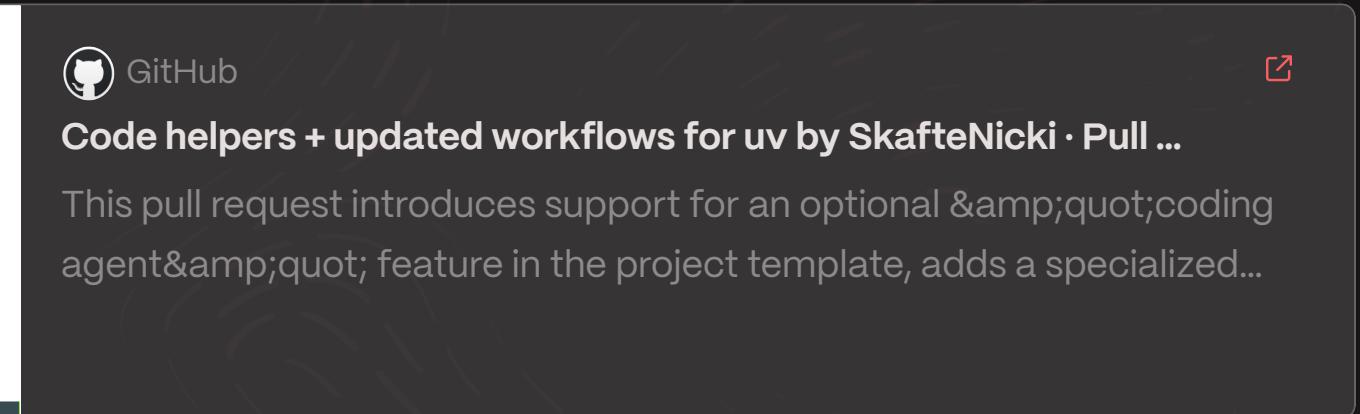
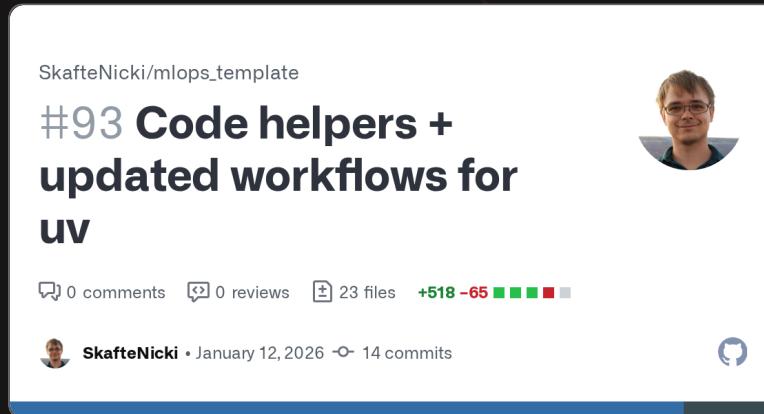
GitHub

[bug] CLI exercise classifier dataset mismatch · Issue #480 · S...

In exercise of Command line arguments at
s2_organisation_and_version_control/cli/, part 3 mentions: ... trains a...



A few corrections to last weeks template



- Smaller updates to agents
- Specific works for pip and uv workflows
- Correction of Docker uv files

Meme of the day

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

The screenshot shows a GitHub pull request page for a repository. At the top, a comment from 'dependabot' is visible, indicating a bump from version 1.3.1 to 1.3.2 of the 'mixin-deep' dependency. Below this, there's a section for 'Dependabot commands and options'. A series of commits are listed, starting with 'dependabot' adding a 'dependencies' label, followed by a commit to bump the dependency, a force-push from 'dependabot', and finally a merge from 'mergify'. At the bottom of the pull request, a comment from 'nemoBot' includes a celebratory GIF of a person cheering.