

02476 Machine Learning Operations
Nicki Skafte Detlefsen

Continuous Integration

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](#)

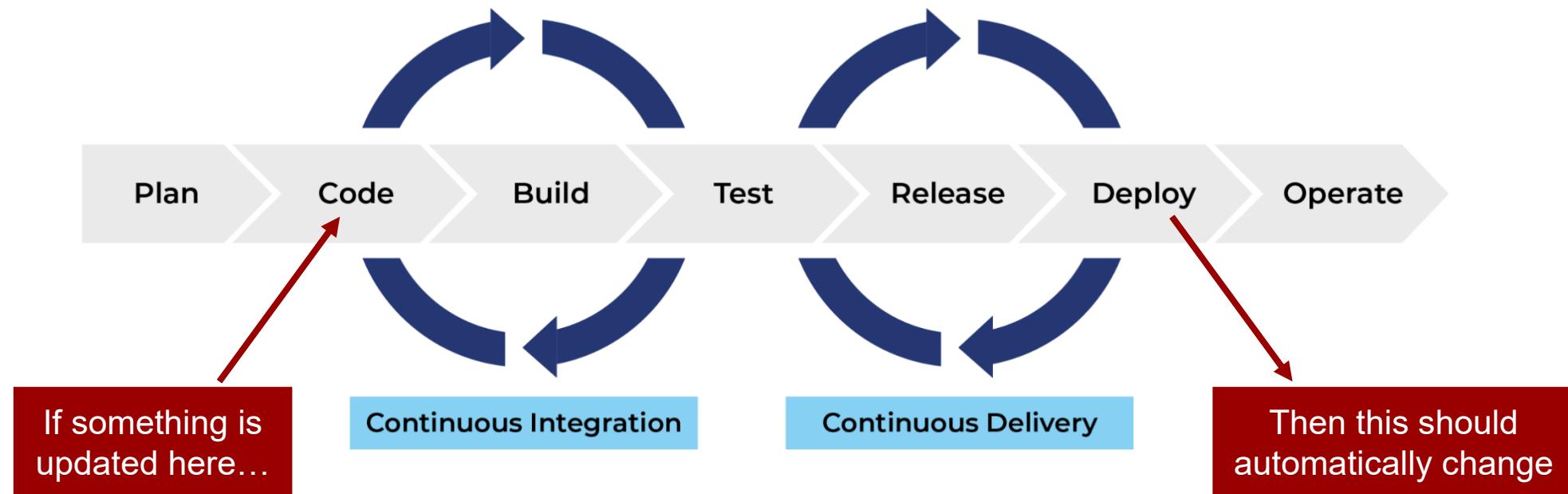
Dev at Fastly : I'll just push this small change to production

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





CD

Continues Integration

Core tasks:

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

 App independent concept

Continues Deployment

Core tasks:

- 💡 How to get your code/application to the user automatically? + monitor life cycle

 App dependent concept

CML

Continues Machine Learning

Core tasks:

- 💡 How to automatically retrain machine learning models when data and code changes?

 Specific to ML applications

MLOps levels

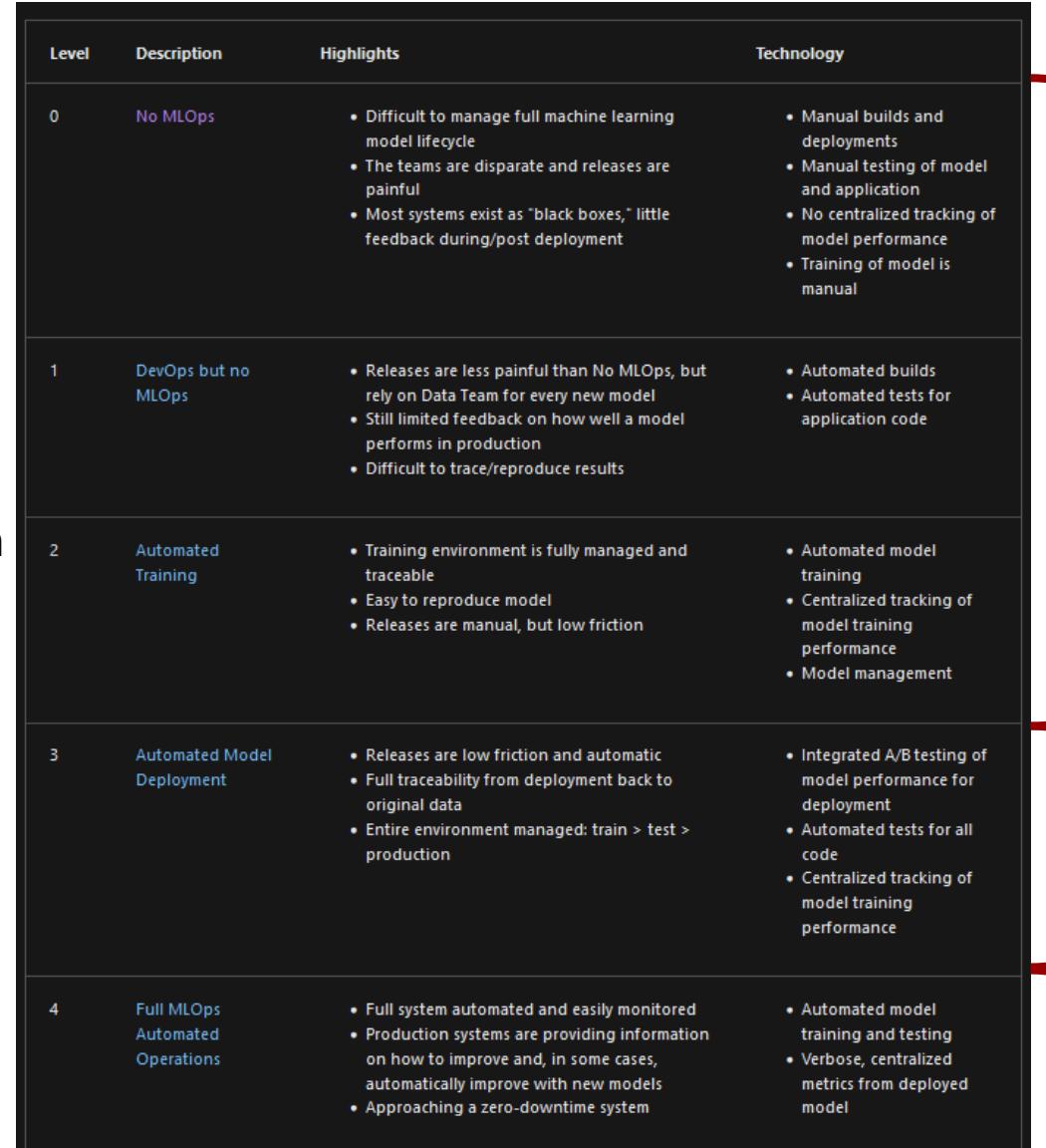
The **Maturity model** overall describes the DevOps practices to run a successful MLOps environment.

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.



A table illustrating the MLOps maturity model levels, categorized by CI (Continuous Integration), CD (Continuous Deployment), and CML (Continuous Model LifeCycle).

Level	Description	Highlights	Technology
0	No MLOps	<ul style="list-style-type: none"> Difficult to manage full machine learning model lifecycle The teams are disparate and releases are painful Most systems exist as "black boxes," little feedback during/post deployment 	<ul style="list-style-type: none"> Manual builds and deployments Manual testing of model and application No centralized tracking of model performance Training 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 model Still limited feedback on how well a model performs in production Difficult to trace/reproduce results 	<ul style="list-style-type: none"> Automated builds Automated tests for application code
2	Automated Training	<ul style="list-style-type: none"> Training environment is fully managed and traceable Easy to reproduce model Releases are manual, but low friction 	<ul style="list-style-type: none"> Automated model training Centralized tracking of model training performance Model management
3	Automated Model Deployment	<ul style="list-style-type: none"> Releases are low friction and automatic Full traceability from deployment back to original data Entire environment managed: train > test > production 	<ul style="list-style-type: none"> Integrated A/B testing of model performance for deployment Automated tests for all code Centralized tracking of model training performance
4	Full MLOps Automated Operations	<ul style="list-style-type: none"> Full system automated and easily monitored Production systems are providing information on how to improve and, in some cases, automatically improve with new models Approaching a zero-downtime system 	<ul style="list-style-type: none"> Automated model training and testing Verbose, 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 displays the GitHub profile of the `TorchMetrics` repository. At the top, a table lists recent commits, all of which are related to Continuous Integration (CI) updates. Below this is the repository's README file, which features a purple hexagonal logo with a bar chart icon and the text "TorchMetrics". The README also includes a subtitle: "Machine learning metrics for distributed, scalable PyTorch applications." A navigation bar at the bottom of the README lists links to "What is Torchmetrics", "Implementing a metric", "Built-in metrics", "Docs", "Community", and "License".

Contributors: 169 (with 158 more)

Languages: Python 99.9% (Other 0.1%)

CI Status: CI testing - complete (Azure Pipelines succeeded, codecov 39%)

Build Status: slack chat (green), docs passing (green), DOI 10.5281/zenodo.5844769 (blue), JOSS 10.21105/joss.04101 (green), pre-commit.ci passed (green)

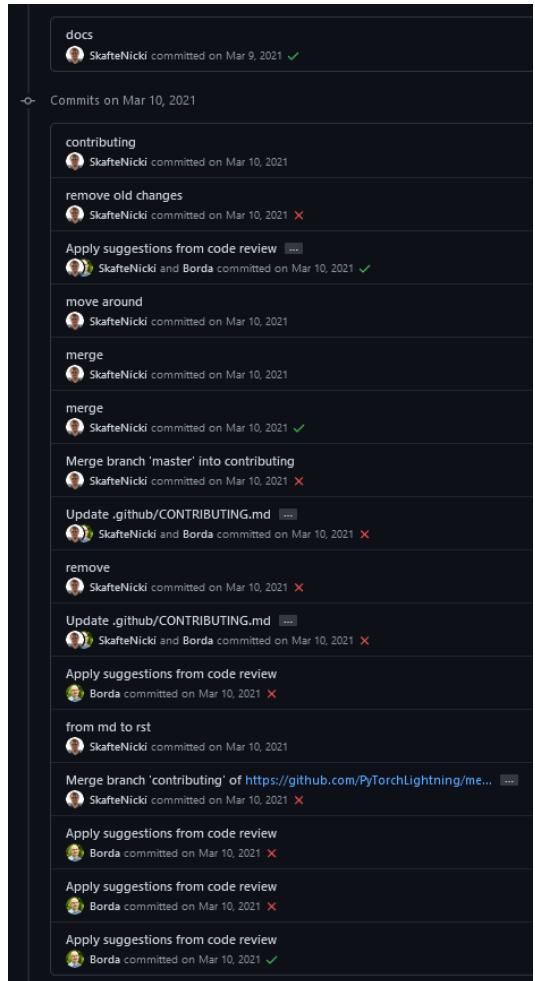
CI step 1: version control

User version control:

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

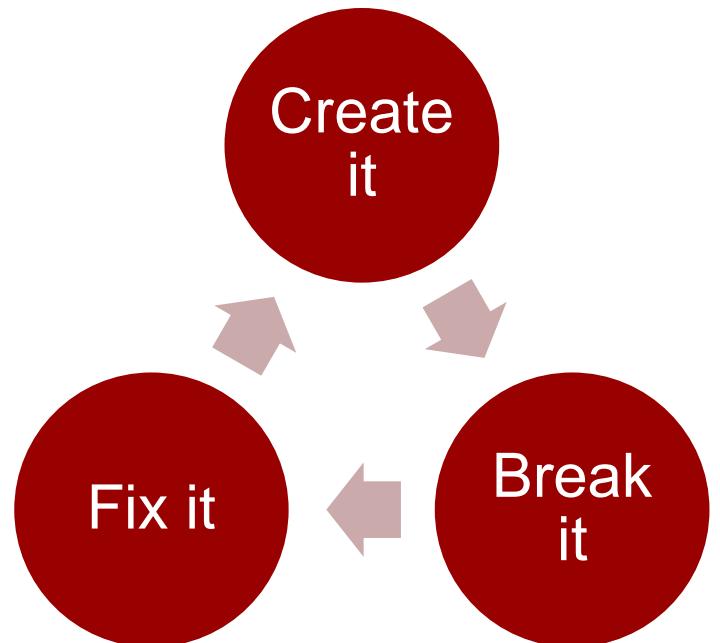
Commit frequently:

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



The screenshot shows a GitHub commit history for a repository named 'docs'. The commits are dated March 10, 2021, and are authored by 'SkafteNicki' and 'Borda'. The commits include:

- docs (SkafteNicki committed on Mar 9, 2021)
- Commits on Mar 10, 2021
 - contributing (SkafteNicki committed on Mar 10, 2021)
 - remove old changes (SkafteNicki committed on Mar 10, 2021)
 - Apply suggestions from code review (SkafteNicki and Borda committed on Mar 10, 2021)
 - move around (SkafteNicki committed on Mar 10, 2021)
 - merge (SkafteNicki committed on Mar 10, 2021)
 - merge (SkafteNicki committed on Mar 10, 2021)
 - Merge branch 'master' into contributing (SkafteNicki committed on Mar 10, 2021)
 - Update .github/CONTRIBUTING.md (SkafteNicki and Borda committed on Mar 10, 2021)
 - remove (SkafteNicki committed on Mar 10, 2021)
 - Update .github/CONTRIBUTING.md (SkafteNicki and Borda committed on Mar 10, 2021)
 - Apply suggestions from code review (Borda committed on Mar 10, 2021)
 - from md to rst (SkafteNicki committed on Mar 10, 2021)
 - Merge branch 'contributing' of https://github.com/PyTorchLightning/me... (SkafteNicki committed on Mar 10, 2021)
 - Apply suggestions from code review (Borda committed on Mar 10, 2021)
 - Apply suggestions from code review (Borda committed on Mar 10, 2021)
 - Apply suggestions from code review (Borda committed on Mar 10, 2021)



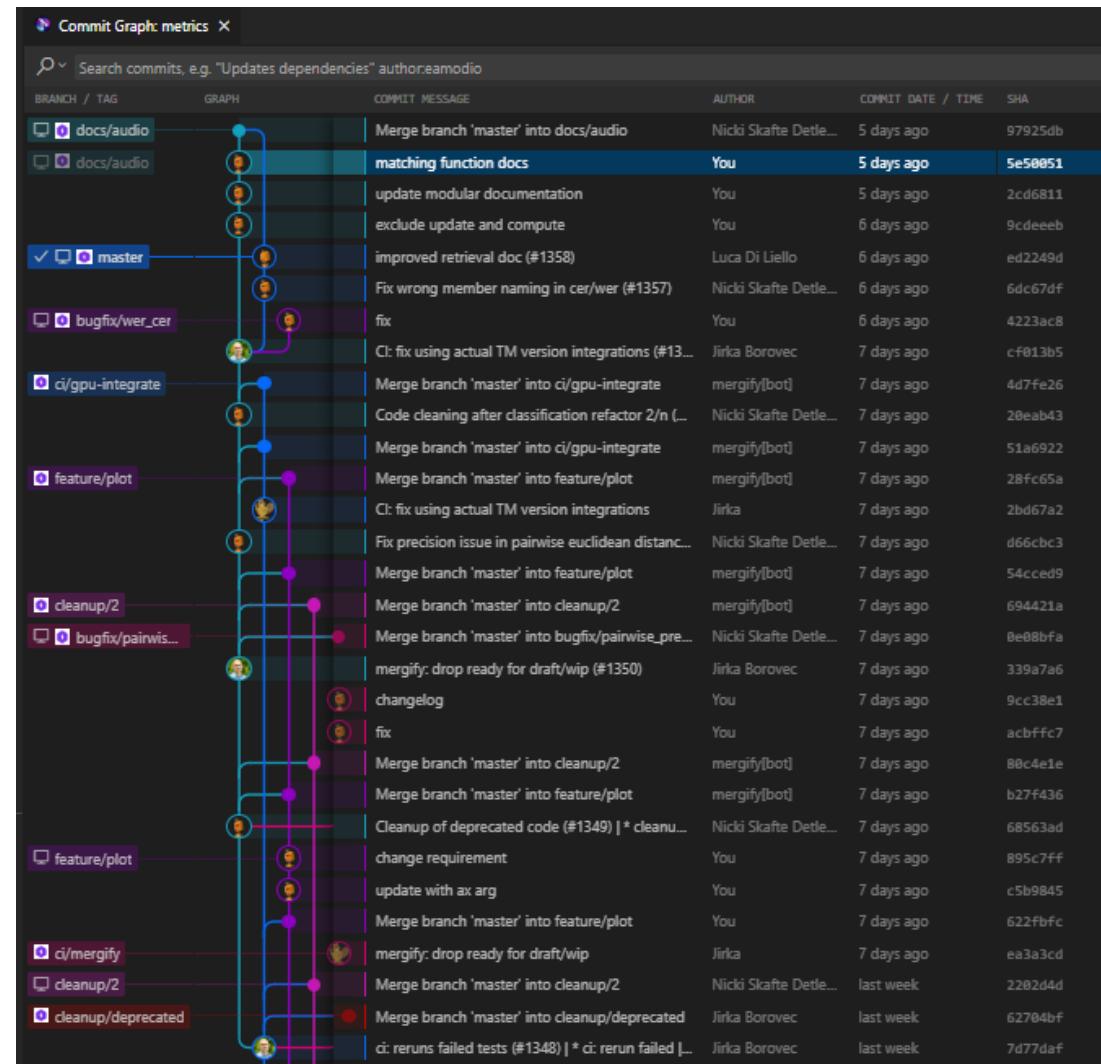
CI step 1: Use branches

Parallel workflow

Experimental features changes are kept away from master/main

Recommend extensions for VS code:

- 💡 [Gitlens](#) or [GitGraph](#)
- 💡 [Github PR and issues](#)



CI step 1: Use pull requests

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

The screenshot shows a GitHub pull request interface with the following steps highlighted:

- 1. Find PR**: The "Pull requests" tab is selected, indicated by a red box around the tab bar.
- 2. Check changes**: The "Files changed" section is highlighted with a red box, showing a diff view of the code changes.
- 3. Make one or more comments**: A comment from user "SkafeNicki" is shown, suggesting to add an entry to the file. The entire comment area is highlighted with a red box.
- 4. Send review**: The "Review changes" button at the top right of the diff view is highlighted with a red box.

The pull request details shown are for "Added MinkowskiDistance support #1362". The code changes involve imports for mean_squared_log_error, mean_absolute_error, and minkowski_distance from torchmetrics.functional.regression and torchmetrics.functional.mape.

CI step 1: pre-commit



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

```
! .pre-commit-config.yaml
! .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_mlops on 🐫 main [!??] via 🐕 v3.11.5 🐕 mlops
> 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
```

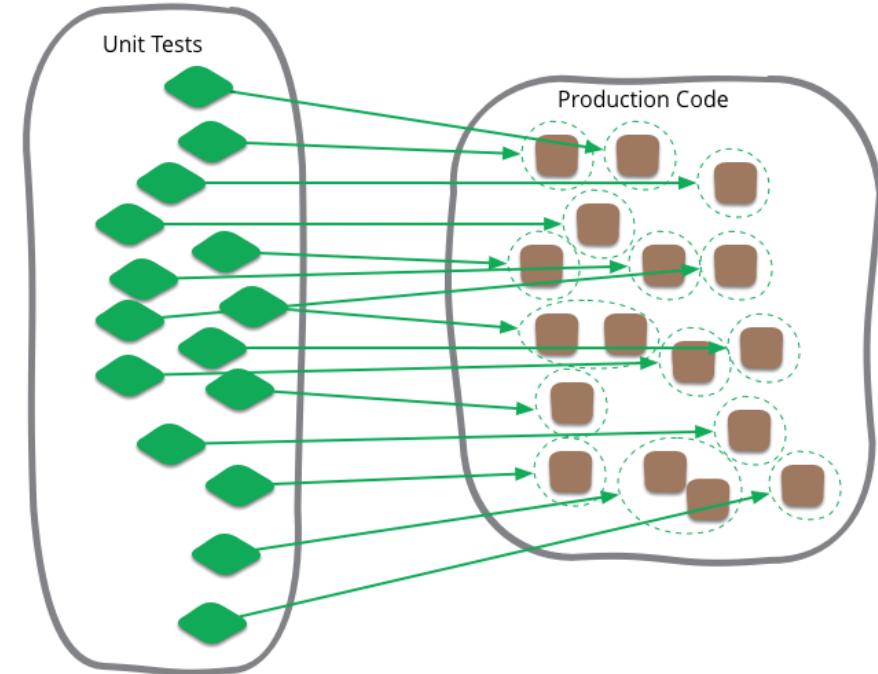
CI step 2: write tests

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)

File / Commit Message	Date
..	6 days ago
audio	2 months ago
classification	7 days ago
detection	19 days ago
functional	7 days ago
image	21 days ago
multimodal	11 days ago
nominal	10 days ago
regression	21 days ago
retrieval	2 months ago
text	6 days ago
utilities	7 days ago
wrappers	7 days ago
__about__.py	11 days ago
__init__.py	8 days ago
aggregation.py	6 months ago
collections.py	7 days ago
metric.py	7 days ago
py.typed	6 months ago

CI step 2: write tests

For tests, the convention is to either

```
├── README.md
├── src/
│   ├── __init__.py
│   └── important_functions.py
└── tests/
    └── index.md
```

CI step 2: write tests

CI step 2: write tests

CI step 2: execute locally

CI step 3: Automating stuff

What can be automated: EVERYTHING 

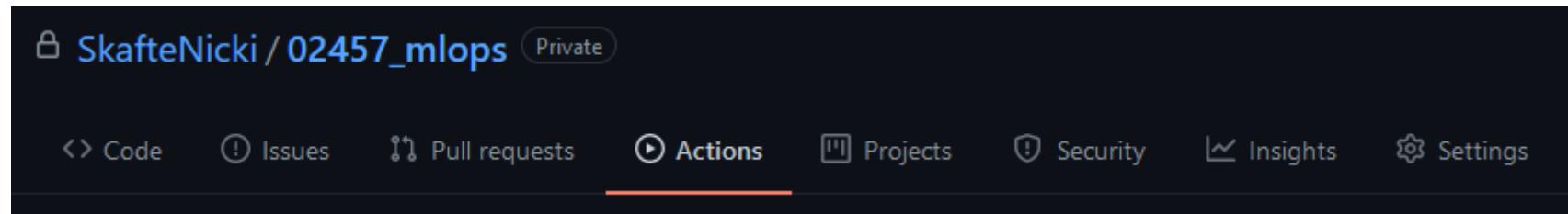
- 💡 Unit testing
- 💡 Integration testing
- 💡 Documentation creation
- 💡 Linters (style formatting)
- 💡 Security checks
- 💡 Code coverage
- 💡 Custom checks...

Only your imagination is the limit...

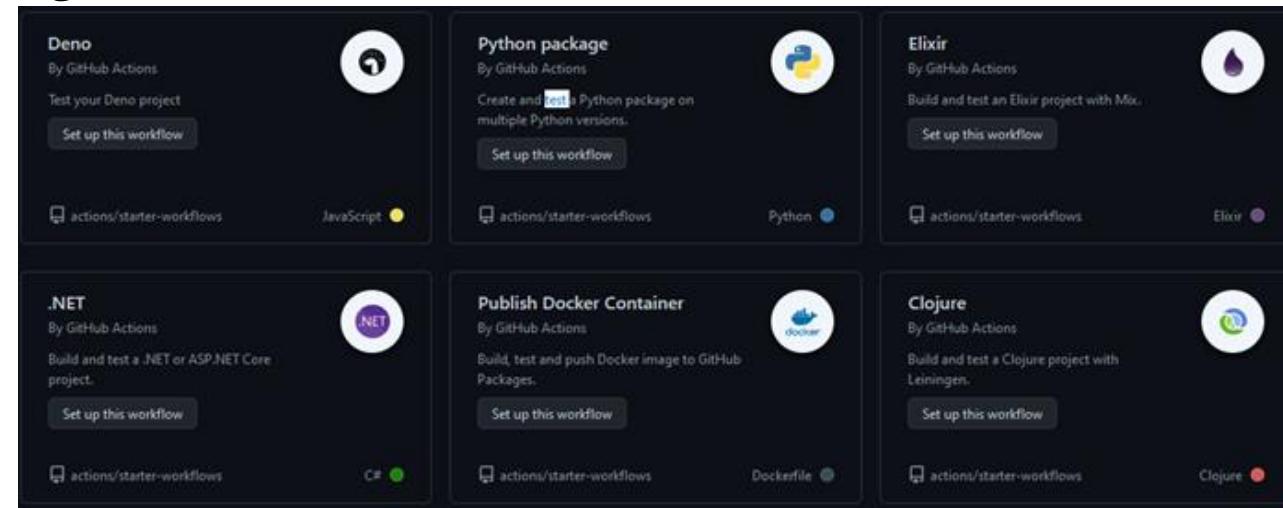
CI step 3: Github actions

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

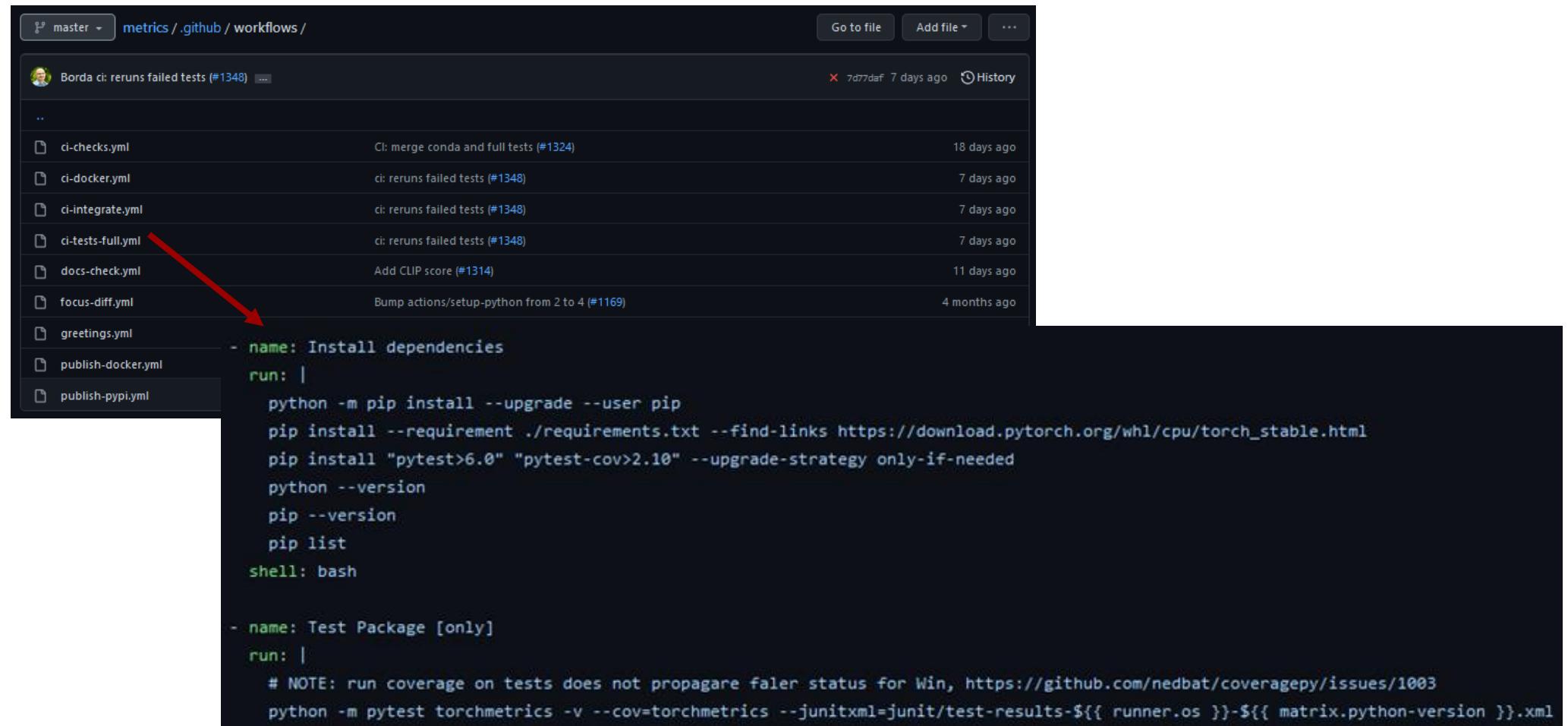
Install dependencies

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

CI step 3: workflow files



The screenshot shows a GitHub repository interface with the following details:

- Repository: metrics / .github / workflows /
- Branch: master
- Last commit: 7d77ddf 7 days ago
- Commit message: History
- File list:
 - ci-checks.yml (Cl: 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
 - greetings.yml
 - publish-docker.yml
 - publish-pypi.yml

A red arrow points to the 'ci-tests-full.yml' file in the list. The file content is displayed below:

```
- name: Install dependencies
  run: |
    python -m pip install --upgrade --user pip
    pip install --requirement ./requirements.txt --find-links https://download.pytorch.org/wheel/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.

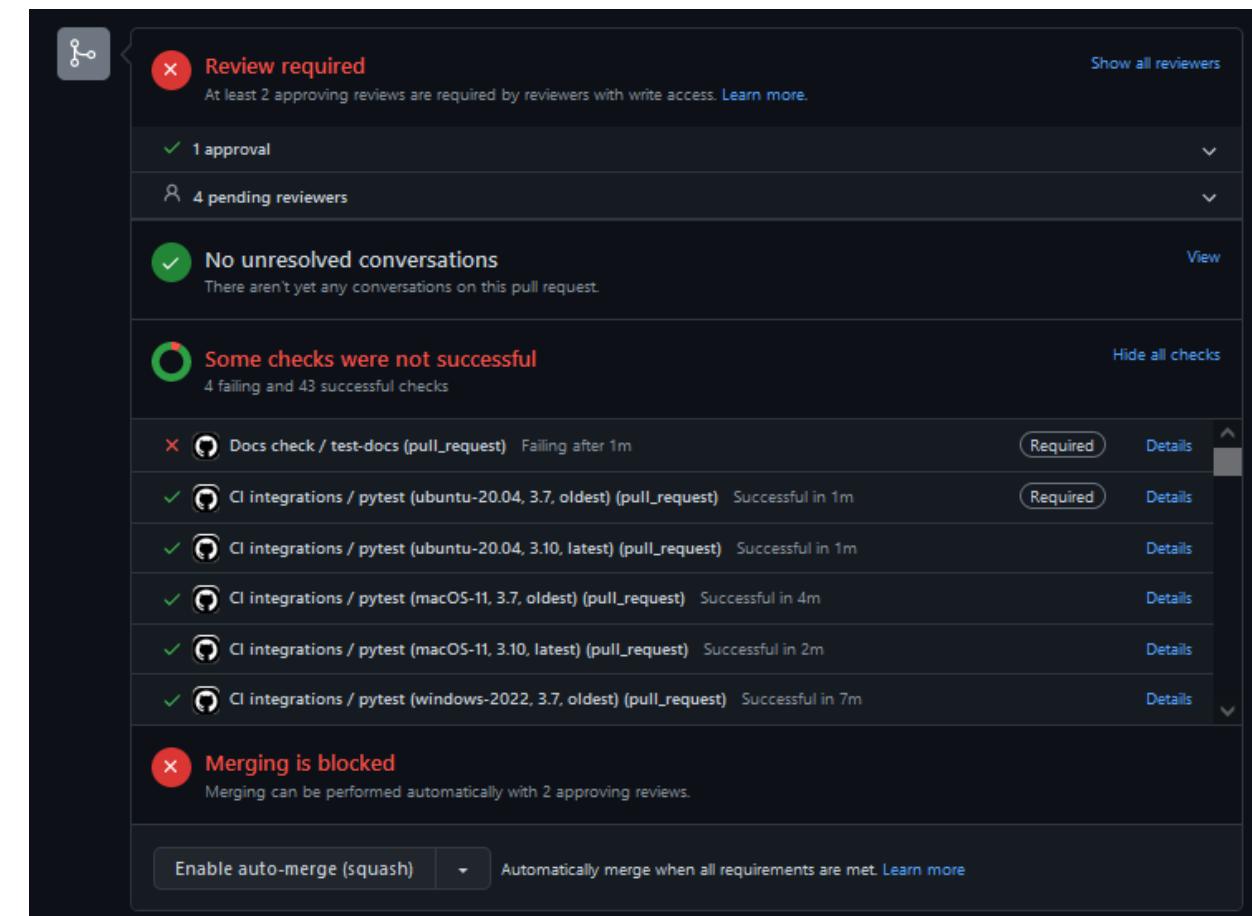
<p>✓ Mergify</p> <p>✓ Summary</p> <p>✗ WIP</p> <p>✓ WIP</p> <p>✗ Docs check on: pull_request</p> <p>✓ test-docs</p> <p>✓ make-docs</p> <p>✓ paper-JOSS</p> <p>✓ paper-cite</p> <p>✗ CI integrations on: pull_request</p> <p>✓ pytest (ubuntu-20.04, 3.7, oldest)</p> <p>✓ pytest (ubuntu-20.04, 3.10, latest)</p> <p>✓ pytest (macOS-11, 3.7, oldest)</p> <p>✓ pytest (macOS-11, 3.10, latest)</p> <p>✓ pytest (windows-2022, 3.7, oldest)</p> <p>✓ pytest (windows-2022, 3.10, latest)</p> <p>✓ pytest (3.10, latest, ubuntu-22.04)</p> <p>✓ pytest (3.10, latest, macOS-12)</p>	<p>✗ CI testing - complete on: pull_request 18</p> <p>✓ check-diff / eval-diff</p> <p>✗ pytest (ubuntu-20.04, 3.8, 1.9.1)</p> <p>✗ pytest (ubuntu-20.04, 3.8, 1.10.2)</p> <p>✗ pytest (ubuntu-20.04, 3.8, 1.11.0)</p> <p>✗ pytest (ubuntu-20.04, 3.8, 1.12.1)</p> <p>✗ pytest (ubuntu-22.04, 3.8, 1.13.0)</p> <p>✗ pytest (ubuntu-22.04, 3.10, 1.13...)</p> <p>✗ pytest (macOS-11, 3.8, 1.13.0)</p> <p>✗ pytest (macOS-11, 3.9, 1.13.0)</p> <p>✗ pytest (windows-2022, 3.8, 1.13.0)</p> <p>✗ pytest (windows-2022, 3.9, 1.13.0)</p> <p>✗ pytest (ubuntu-20.04, 3.7, 1.8.1,...)</p> <p>✗ pytest (ubuntu-20.04, 3.8, 1.8.1,...)</p> <p>✗ pytest (macOS-11, 3.7, 1.8.1, old...)</p> <p>✗ pytest (macOS-11, 3.8, 1.8.1, old...)</p> <p>✗ pytest (windows-2019, 3.7, 1.8.1...)</p> <p>✗ pytest (windows-2019, 3.8, 1.8.1...)</p>	<p>✗ General checks on: pull_request 2</p> <p>✓ check-code / mypy</p> <p>✗ check-code / pre-commit</p> <p>✓ check-schema / schema</p> <p>✓ check-package / pkg-check (ub...</p> <p>✓ check-package / pkg-check (ub...</p> <p>✓ check-package / pkg-check (ma...</p> <p>✓ check-package / pkg-check (ma...</p> <p>✓ check-package / pkg-check (wi...</p> <p>✓ check-package / pkg-check (wi...</p> <p>✓ Azure Pipelines</p> <p>✗ Lightning-AI.metrics Re-run</p> <p>✗ Lightning-AI.metrics (pytest PyT... Re-run</p> <p>✗ Lightning-AI.metrics (pytest PyT... Re-run</p>
---	---	---

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



A screenshot of a code editor showing a diff between two versions of a file named "tests/classification/test_auc.py". The left side shows the original code, and the right side shows the modified code. A red arrow points from the GitHub commit history above to the diff in the code editor.

```
diff --git a/tests/classification/test_auc.py b/tests/classification/test_auc.py
index 00-93,9+93,9..00 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
101 +    if unsqueeze_x:
102 +        x = x.unsqueeze(-1)
103 +    if unsqueeze_y:
104 +        y = y.unsqueeze(-1)
105 +    # Test Area Under Curve (AUC) computation
106 +    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 image shows three screenshots illustrating Dependabot's functionality:

- GitHub Pull Request View:** A screenshot of a GitHub pull request (#2293) showing Dependabot's automated update of the `kornia` dependency from version 0.7.1 to 0.6.7. The PR has been merged. A red arrow points from this screenshot to the diff view below.
- Code Diff View:** A screenshot of a code diff between two versions of a requirements file (`requirements/image_test.txt`). The diff highlights the update of the `kornia` dependency from version 0.7.1 to 0.6.7. The line `- kornia >=0.6.7, <0.7.1` is shown in red, and the updated line `+ kornia >=0.6.7, <0.7.2` is shown in green.
- GitHub Repository View:** A screenshot of a GitHub repository showing the configuration file `dependabot.yml`. It contains the following configuration:version: 2
updates:
 - package-ecosystem: "pip"
 directory: "/requirements"
 schedule:
 interval: "weekly"

Summary of continues integration

1. Use version control



2. Write (unit-)test for your code



3. Automate build + test



Meme of the day

The image shows a GitHub pull request (PR) page. At the top, a user profile picture of Gabriele Petronella (@gabro27) is shown next to the text "So this just happened:". Below this, a bulleted list details the sequence of events:

- 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 GitHub interface shows the PR title "Bumps mixin-deep from 1.3.1 to 1.3.2.", the commit message "Bump mixin-deep From 1.3.1 to 1.3.2.", and the merge commit message "force-pushed the dependabot/npm_and_yarn/mixin-deep-1.3. d795b84 to cf6ab99". A comment from "mergify" merges the commit. At the bottom, a GIF of a group of people cheering is displayed.

