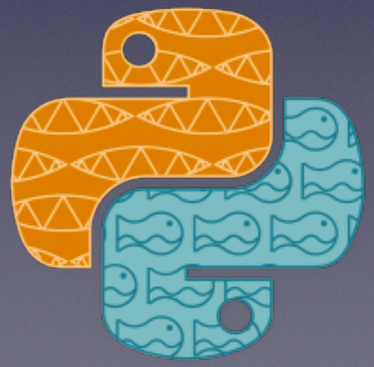
From Script to Open Source Project

Python standards, tools and continuous integration

Michał Karzyński • EuroPython 2019



europython
July 8-14 2019 BASEL

So, you wanna be a

ROCK STAR

Step 1

Master your instrument

Step 2

Learn to play in a band

What can help you play together better?

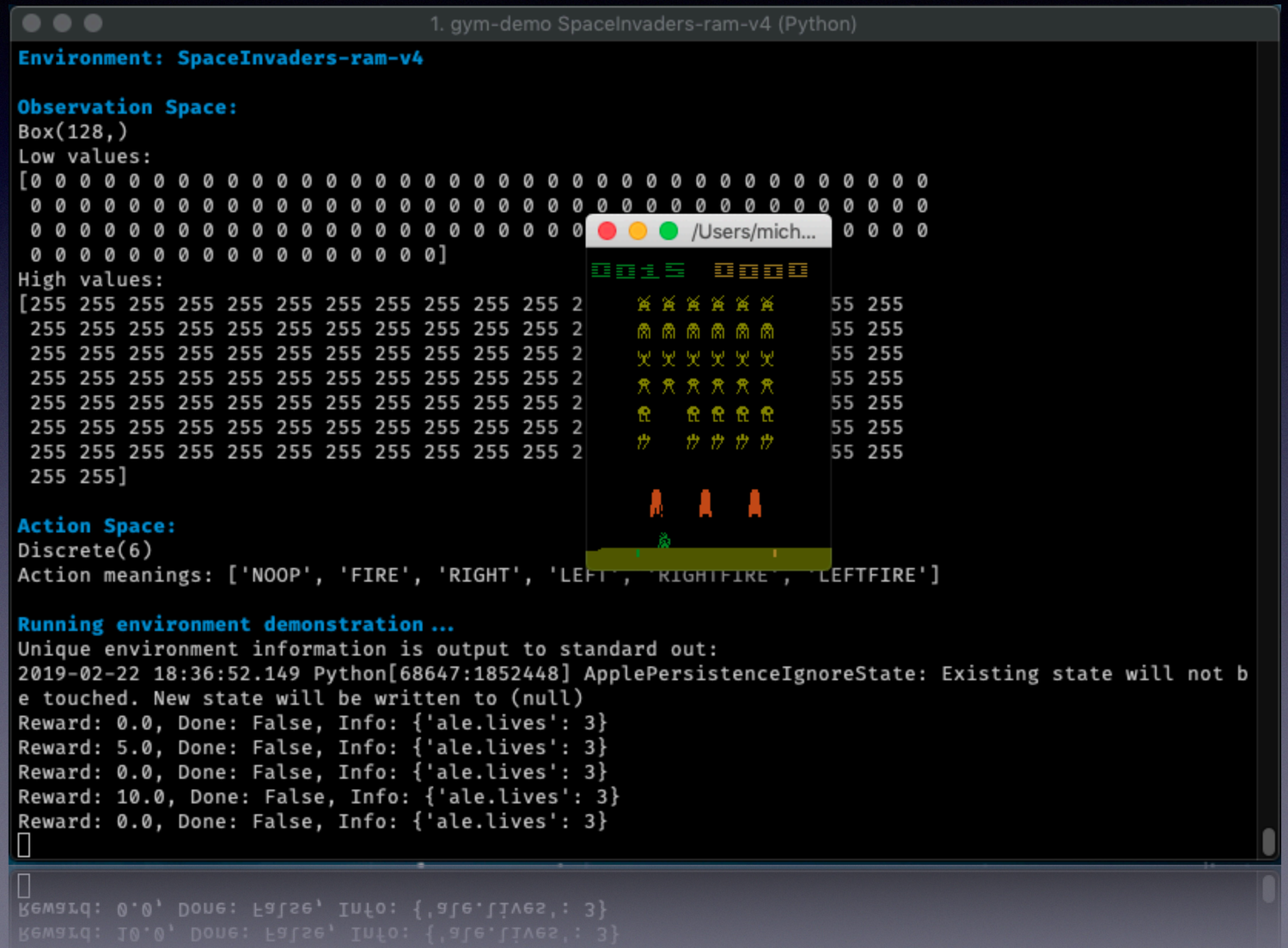
- Standards
- Best practices
- Tools

About me

- Michał Karzyński (@postrational)
- Full stack geek (C++, Python, JavaScript)
- I blog at michal.karzynski.pl
- I'm an architect at 

working on 
nGraph

Open AI Gym Demo



gym-demo

Stages	Code Prep · Automate · CI
Specs	PEP8 · GNU/POSIX · PyPA · PyCQA
pip install	pytest · mypy · docopt · setuptools · tox virtualenv · wheel · black · pre-commit · flake8
Services	↗ GitHub · ↗ TravisCI · ↗ coveralls.io · ↗ Dependabot ↗ codeclimate.com · ↗ codacy.com · ↗ mergify.io · ↗ pyup.io

Your command-line interface (CLI)

```
$ gym-demo
Usage: gym-demo [--steps=NN --no-render --observations] ENV_NAME

$ gym-demo --help

$ gym-demo --steps=5 --no-render Pendulum-v0

$ gym-demo -ns 5 Pendulum-v0
```

GNU/POSIX docopt Code Prep

Your command-line interface (CLI)

```
#!/usr/bin/env python

"""Usage: gym-demo [--steps=NN --no-render --observations] ENV_NAME

Show a random agent playing in a given OpenAI environment.

Arguments:
  ENV_NAME              Name of the Gym environment to run

Options:
  -h --help
  -s --steps=<STEPS>    How many iteration to run for.  [default: 5000]
  -n --no-render        Don't render the environment graphically.
  -o --observations     Print environment observations.
"""
```

GNU/POSIX | docopt | Code Prep

Your command-line interface (CLI)

```python
import docopt

arguments = docopt(__doc__)

print_observations = arguments.get("--observations")
steps = int(arguments.get("--steps"))
render_env = not arguments.get("--no-render")
```

GNU/POSIX docopt Code Prep

Code directory layout

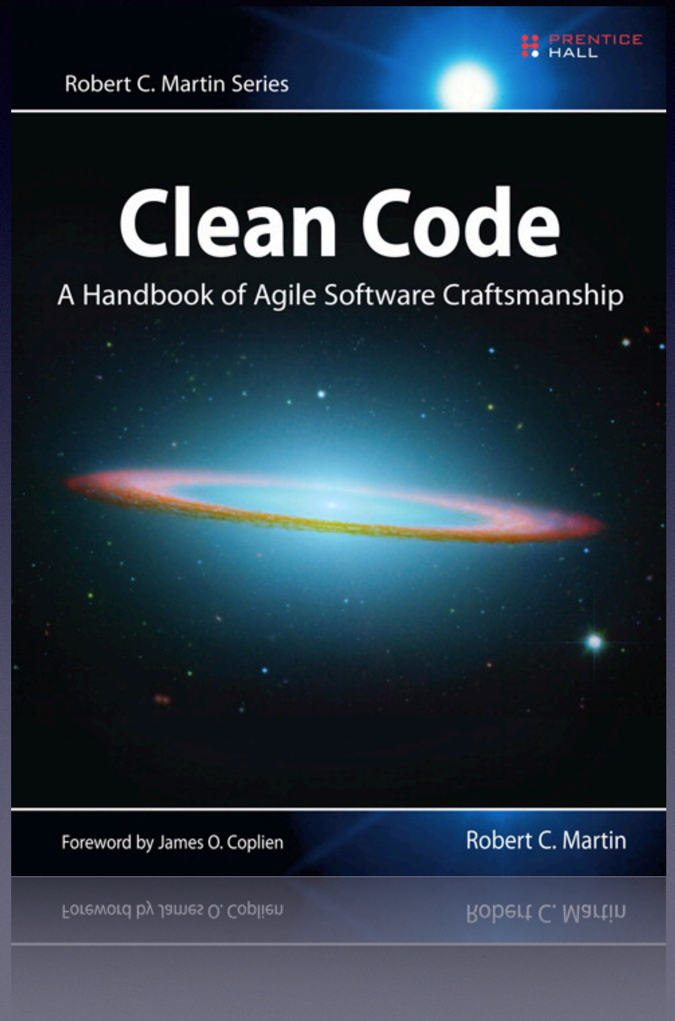
```
package-name
  ├── LICENSE
  ├── README.md
  ├── main_module_name
  │   ├── __init__.py
  │   ├── helpers.py
  │   └── main.py
  ├── docs
  │   ├── conf.py
  │   └── index.rst
  ├── tests
  │   └── test_main.py
  ├── requirements.txt
  └── setup.py
```

src

PEP8

Code Prep

Code structure



- meaningful names
- single responsibility
- up to 2 parameters
- preferably no side-effects
- write unit tests

@unclebobmartin

Code Prep

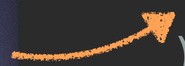
Define your main function

```
if __name__ == "__main__":
    main()
```

Code Prep

Preparing your setup.py file

```
#!/usr/bin/env python
import os
from setuptools import setup

setup(
    name="gym-demo",
    version="0.2.1",
    description="Explore OpenAI Gym environments.",
    long_description=open(
        os.path.join(os.path.abspath(os.path.dirname(__file__)), "README.md")
    ).read(),
    long_description_content_type="text/markdown",
    author="Michal Karzynski",
    packages=["gym_demo"],
    install_requires=["setuptools", "docopt"],
)
```



PyPA | setuptools | wheel | virtualenv | Code Prep

Using your setup.py file

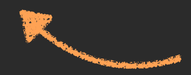
```
$ python setup.py sdist          # Prepare a source package
$ python setup.py bdist_wheel    # Prepare a binary wheel for distribution



# Start local development in a Virtualenv:
$ source my_venv/bin/activate
(my_venv)$ python setup.py develop
or
(my_venv)$ pip install -e .
```

PyPA | setuptools | wheel | virtualenv | Code Prep

Add entry_points to setup.py

setup.py

```
setup(
    # other arguments here...

    # my_module.main:main points to the method main in my_module/main.py
    entry_points={"console_scripts": ["my-command = my_module.main:main"]},
)
```

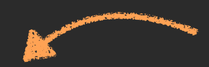


PyPA setuptools Code Prep

Create a requirements.txt file

requirements.txt

```
colorful==0.5.0
docopt==0.6.2
gym==0.12.5
another_package>=1.0,<=2.0
git+https://myvcs.com/some_dependency@sometag#egg=SomeDependency
```



```
$ pip freeze > requirements.txt

$ pip install -r requirements.txt

$ pip install -r requirements_test.txt
```

PyPA pip Code Prep

Use Black to format your code

```
(my_venv) $ black my_module
All done! ✨ 🍰 ✨
1 file reformatted, 7 files left unchanged.
```

📘 PEP8 | black | Automate

Use pre-commit to run formatters

📄 **.pre-commit-config.yaml**

```yaml
repos:
- repo: https://github.com/ambv/black
  rev: stable
  hooks:
    - id: black
```

```
(my_venv) $ pre-commit install
(my_venv) $ git commit
black....................... Failed
hookid: black

Files were modified by this hook.
Additional output:

reformatted gym_demo/demo.py
All done! ✨ 🍰 ✨
1 file reformatted.
```

📘 PEP8 | pre-commit | Automate

Use flake8 to check your code

requirements_test.txt

```
flake8
flake8-blind-except
flake8-bugbear
flake8-builtins
flake8-comprehensions
flake8-debugger
flake8-docstrings
flake8-isort
flake8-quotes
flake8-string-format
```

tox.ini

```
[flake8]
max-line-length=88
max-complexity=6
inline-quotes=double
; ignore:
; C812 - Missing trailing comma
; D104 - Missing docstring in package
ignore=C812,D104
```

```
(my_venv) $ flake8
./my_package/my_module.py:1:1: D100 Missing docstring in public module
```

PyCQA flake8 Automate

Use MyPy for static type analysis

```
from typing import List, Text, Mapping, Union, Optional

def greeting(name: Text) -> Text:
    return "Hello  {}”.format(name)

def my_function(name: Optional[Text] = None) -> Mapping[str, Union[int, float]]:
    ...
```

```
(my_venv) $ mypy --config-file=tox.ini my_module
my_module/main.py:43:27: error: Argument 1 to “my_function" has incompatible
type "int"; expected "List[str]"
```

PyCQA | typing | mypy | Code Prep

Use tox to test all the things

tox.ini

```
[tox]
envlist=py35,py36,py37

[testenv]
deps=
  -Urrequirements.txt
  -Urrequirements_test.txt
commands=
  flake8
  pytest tests/

[pytest]
timeout=300
```

```
$ tox -e py37
GLOB sdist-make: .../setup.py
py37 create: .../.tox/py37
py37 installdeps: -Urrequirements.txt
py37 inst: gym-demo-0.2.2.zip
py37 run-test: commands[1] | flake8
py37 run-test: commands[4] | pytest
...
_____________ summary _____________
  py37: commands succeeded
  congratulations :)
The command exited with 0.
```

PyCQA | tox | Automate

Write unit tests

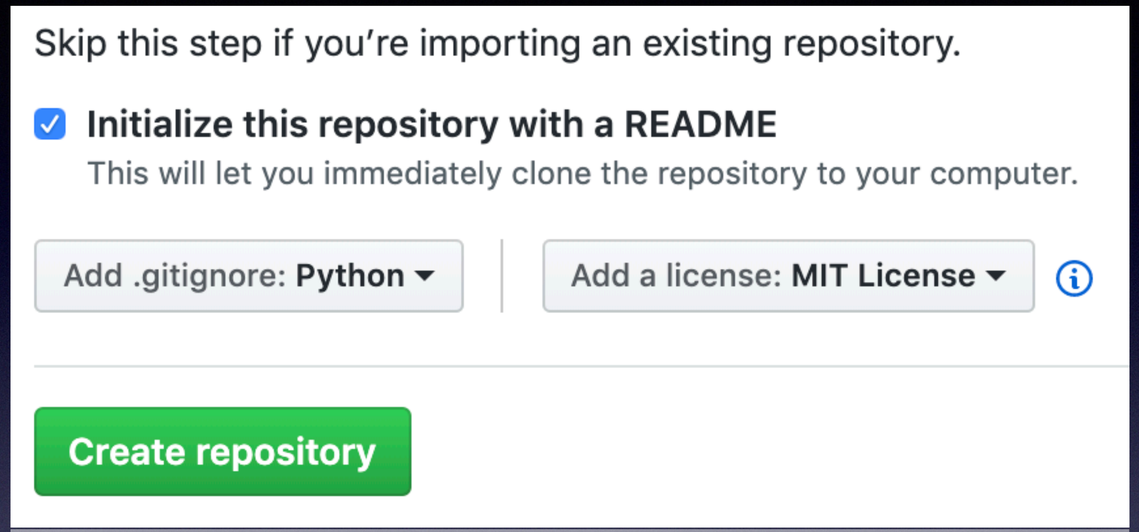
test/test_main.py

```python
"""Test suite for my-project."""
import pytest
from my_project import my_function

def test_my_function():
    result = my_function()
    assert result == "Hello World!"
```

```
$ pytest
======== test session starts ========
rootdir: my-project, inifile: tox.ini
plugins: timeout-1.3.3, cov-2.7.1
timeout: 300.0s
timeout method: signal
timeout func_only: False
collected 7 items

tests/test_main.py .......     [100%]

===== 7 passed in 0.35 seconds ======
```

pytest.org | pytest | Code Prep

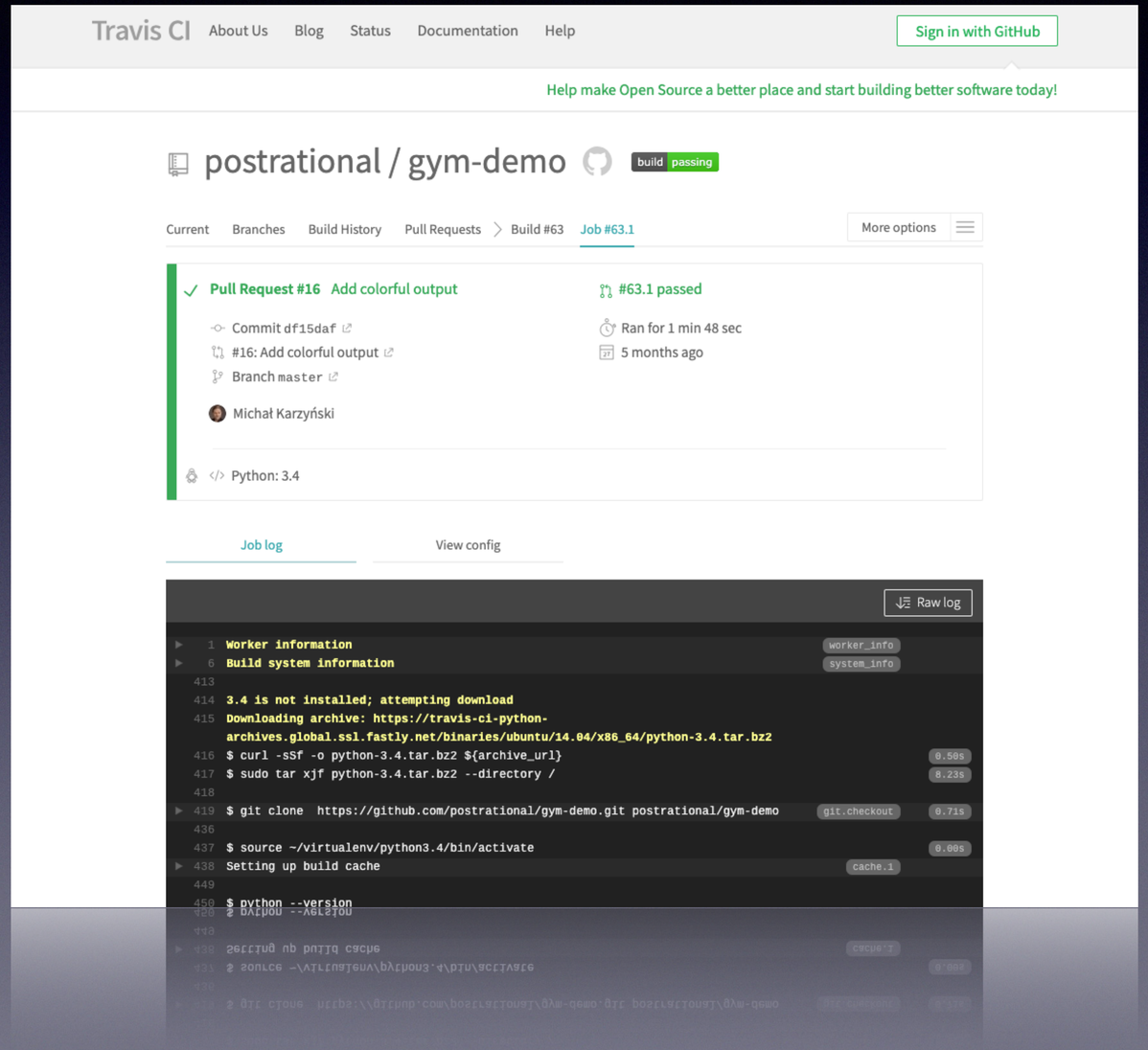
Set up a Git repository




```
$ git init
$ git remote add origin https://github.com/you/your-project.git
$ git pull origin master

$ git add --all
$ git commit -m 'First commit'
$ git push -u origin master
```

↗ GitHub ↗ gitignore.io ↗ choosealicense.com Code Prep

Set up continuous integration



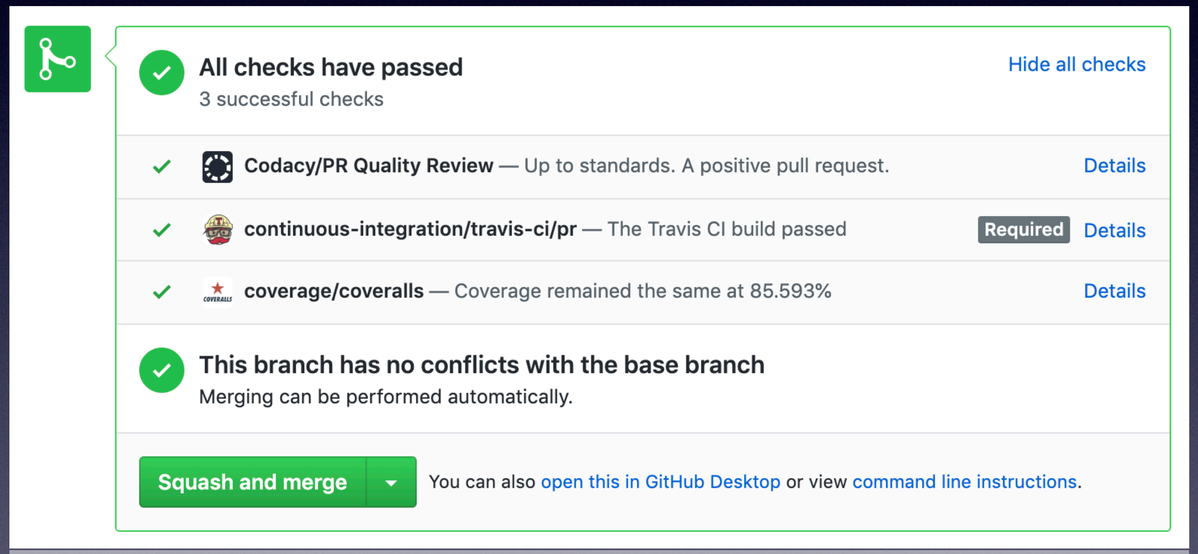
.travis.yml

```yaml
language: python
os: linux
install:
  - pip install tox
script:
  - tox
git:
  depth: false
branches:
  only:
  - "master"
cache:
  directories:
    - $HOME/.cache/pip
```

↗ TravisCI

CI

Set up continuous integration



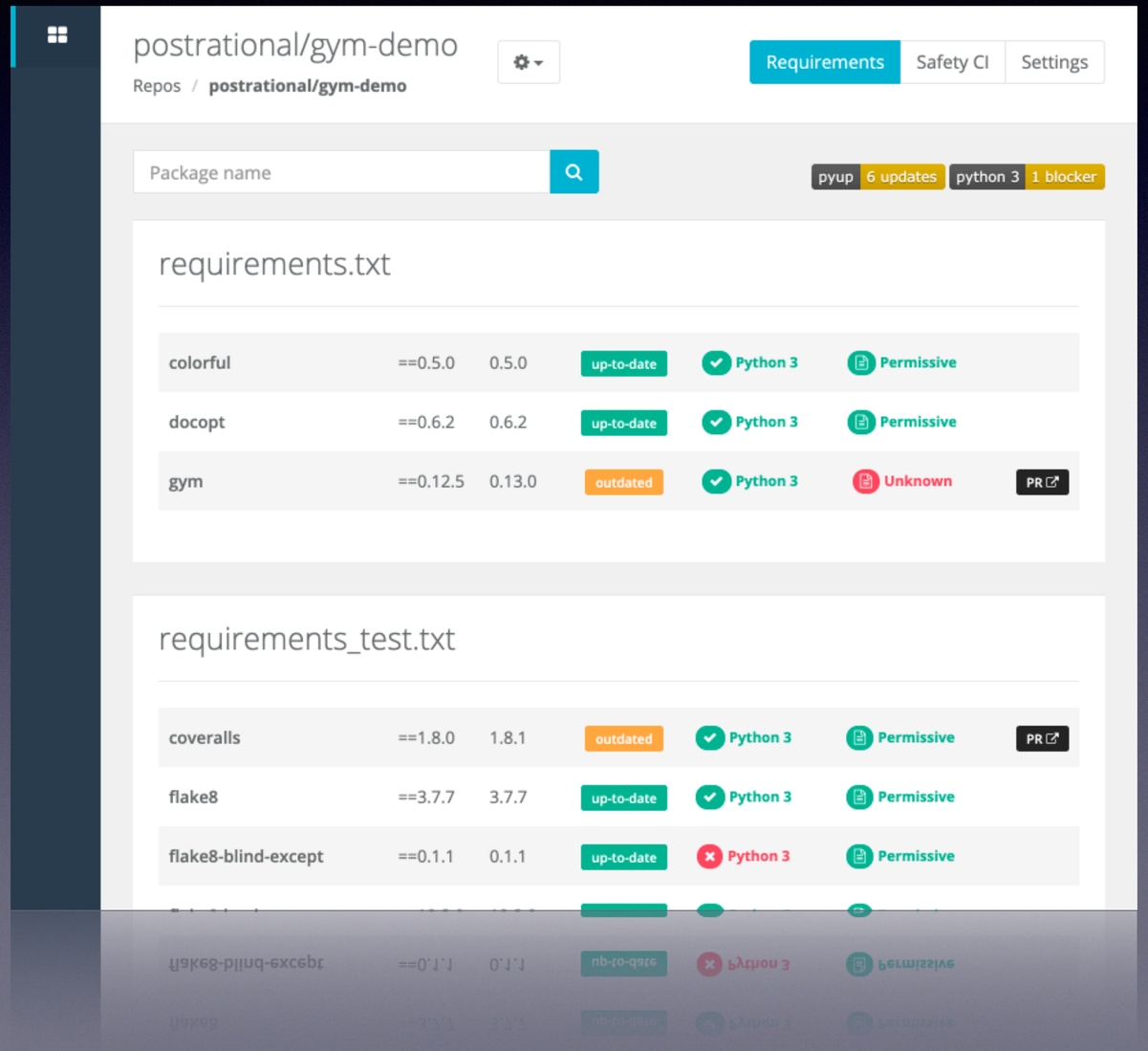
.travis.yml

```yaml
language: python
os: linux
install:
  - pip install tox
script:
  - tox
git:
  depth: false
branches:
  only:
  - "master"
cache:
  directories:
    - $HOME/.cache/pip
```

↗ TravisCI

CI

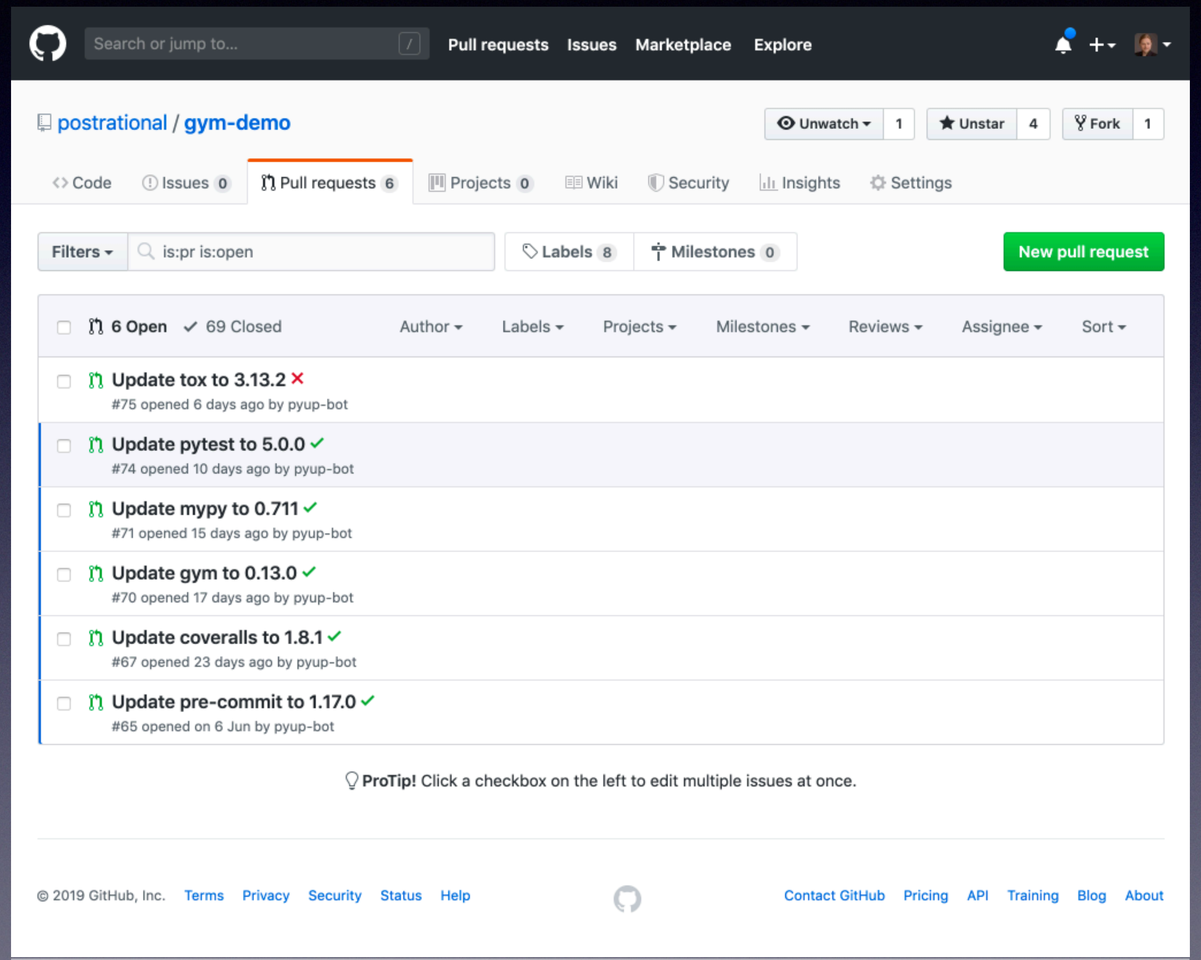
Requirements updater



- No configuration
- Just log in with GitHub and give the bot access permissions
- Bot will find your requirements files

↗ Dependabot ↗ pyup.io CI

Requirements updater



- The bot will start making update PRs
- Which your CI process will test

↗ Dependabot ↗ pyup.io CI

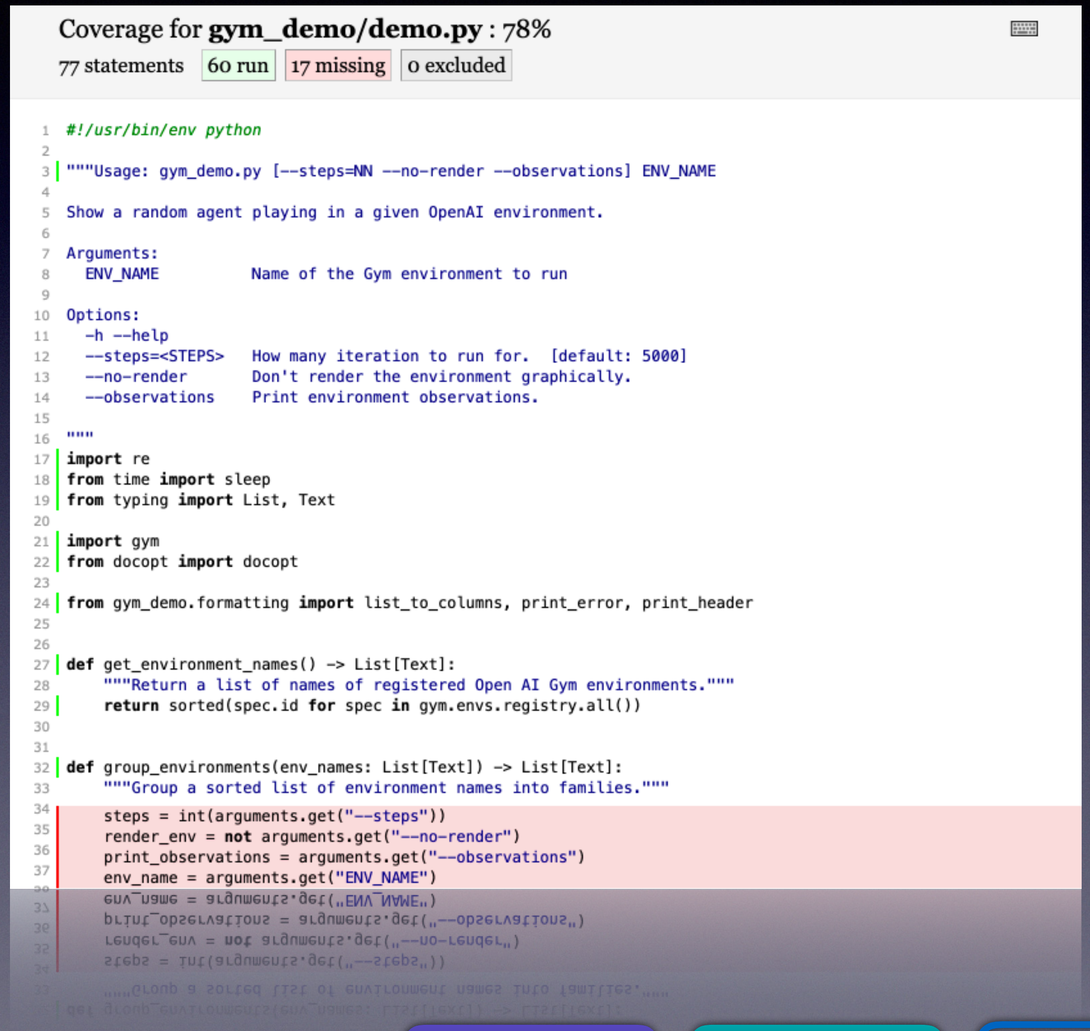
Test coverage checker

```
$ pytest --cov=my_module tests/
========================== test session starts ==========================

tests/test_main.py ..............................                [100%]

------------ coverage: platform darwin, python 3.7.2-final-0 ------------
Name                         Stmts   Miss  Cover
------------------------------------------------
my_module/__init__.py            0      0   100%
my_module/main.py               77     17    78%
my_module/utils.py              41      0   100%
------------------------------------------------
TOTAL                          118     17    86%

======================== 255 passed in 1.25 seconds =====================
```

pytest pytest-cov Automate

Test coverage checker

```
$ pytest --cov=my_module \
         --cov-report=html tests/

$ open htmlcov/index.html
```

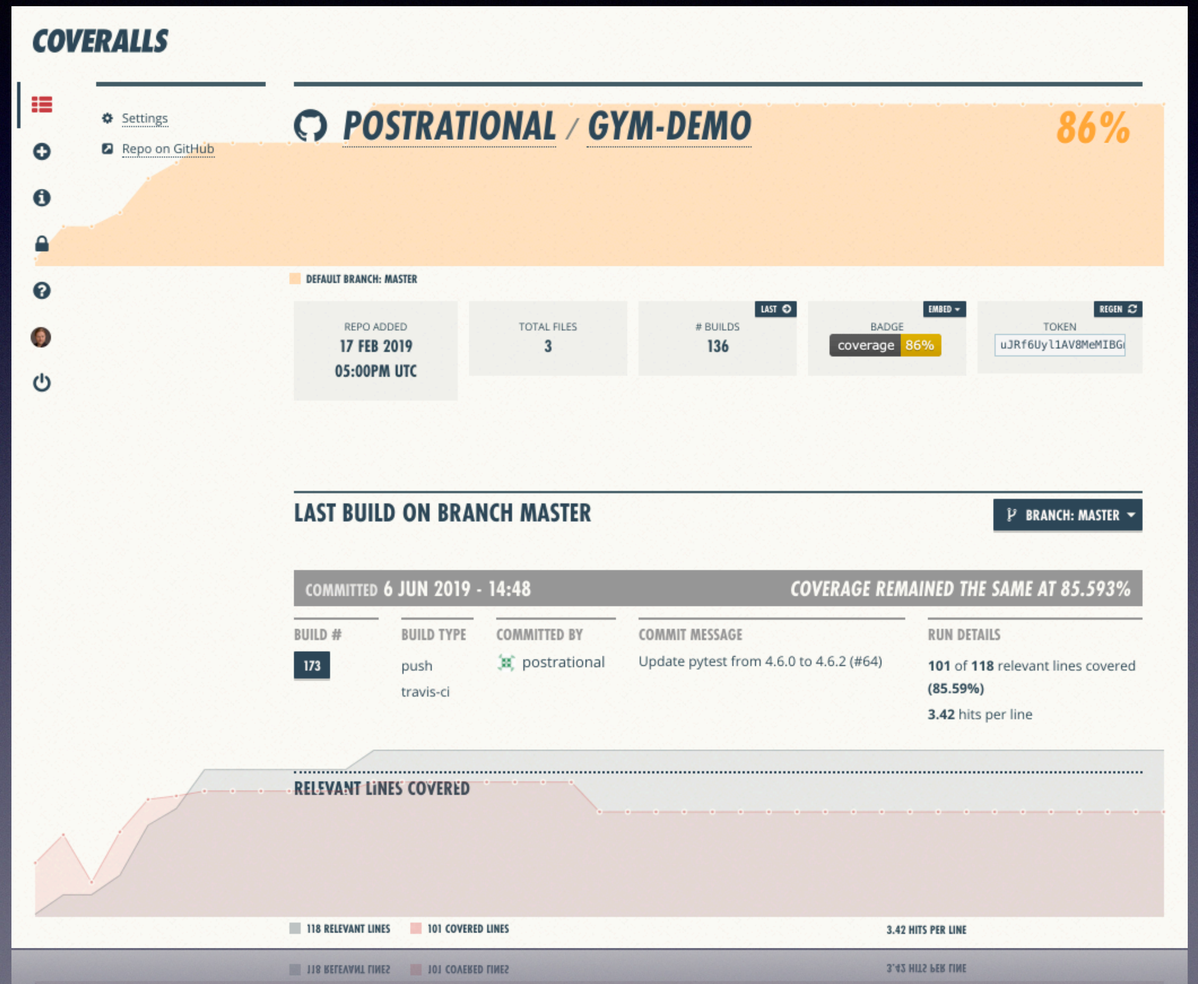


pytest

pytest-cov

Automate

Test coverage checker

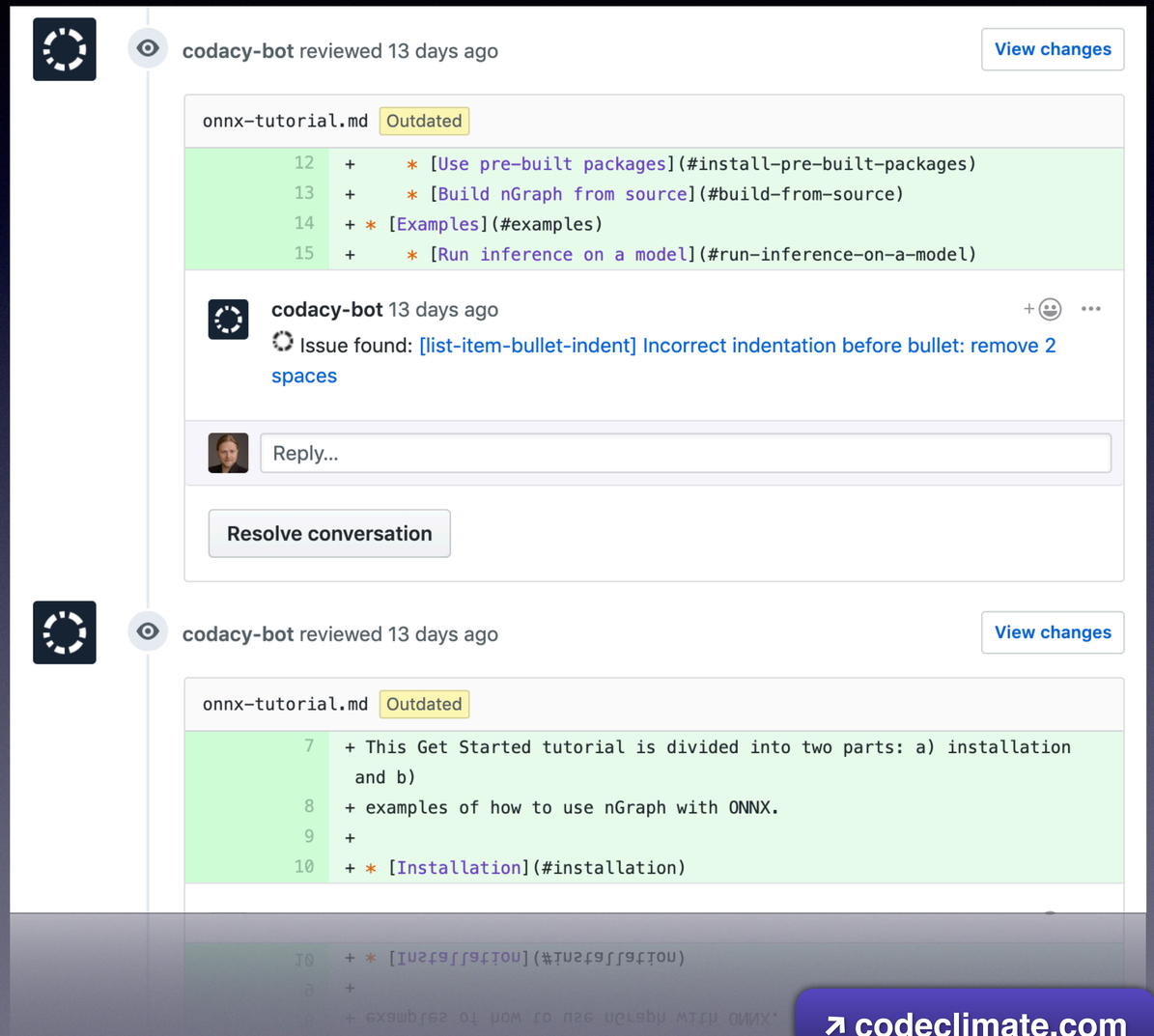


tox.ini

```
[testenv]
...
commands=
  ...
  pytest --cov=my_module tests/
  - coveralls
```

↗ coveralls.io | coveralls | pytest-cov | CI

Automated code review



↗ codeclimate.com

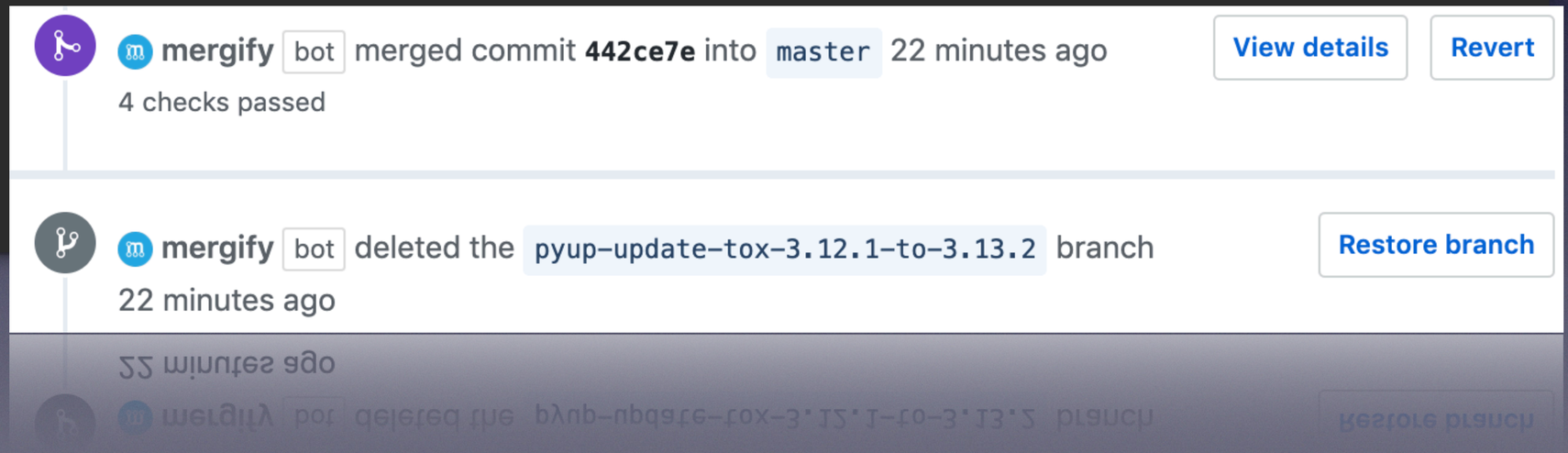
↗ codacy.com

CI

Automated PR merge

.mergify.yml

```yaml
pull_request_rules:
  - name: merge when CI passes and 1 positive review
    conditions:
      - "#approved-reviews-by>=1"
      - status-success=continuous-integration/travis-ci/pr
      - base=master
    actions:
      merge:
        method: squash
        strict: true
```



↗ mergify.io

CI

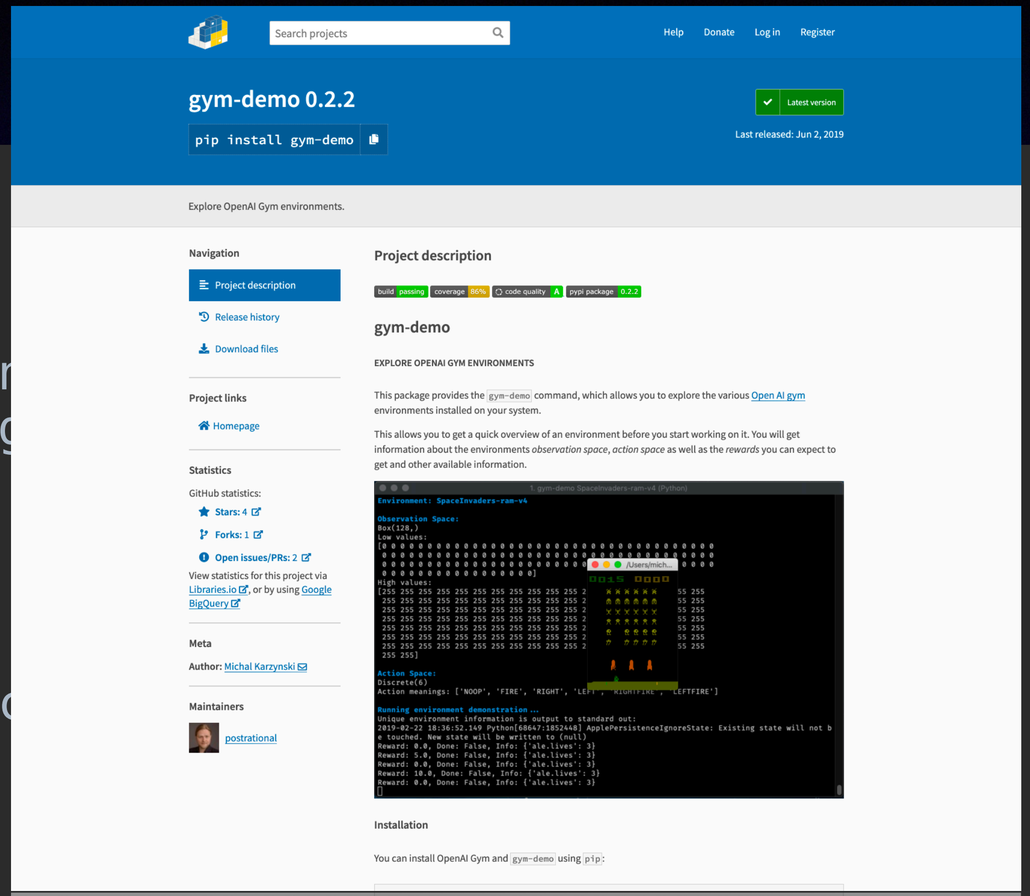
Bots working for you

- PyUP bot finds updates on PyPI
- Travis CI tests your code against the new package version
- Mergify merges the PR if tests pass

CI

Publish your project on PyPI

```
(my_venv) $ python setup.py bdist_wheel
(my_venv) $ python setup.py sdist
(my_venv) $ twine check dist/*
Checking distribution dist/my-package-0.2.2-py3-n
Checking distribution dist/my-package-0.2.2.tar.g

(my_venv) $ twine upload dist/*
Enter your username: my_username
Enter your password:
Uploading distributions to https://pypi.org/legac
Uploading my_package-0.2.2-py3-none-any.whl
Uploading my-package-0.2.2.tar.gz
```



↗ PyPI

wheel

twine

Publish!

More details available on my blog:

michal.karzynski.pl

THANK YOU