PACKAGING PYTHON PROJECTS

GASP skillshare session

Plan for today

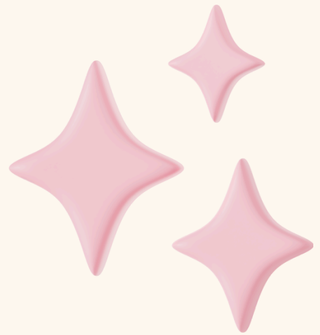
1 Local set-up

2 Publishing to PyPI

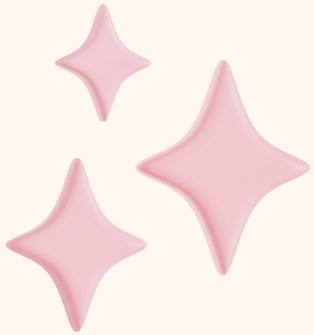
3 GitHub tags and releases

4 GitHub integration

Why do we want to package our project?



`pip install`



Let's practice! [optional]

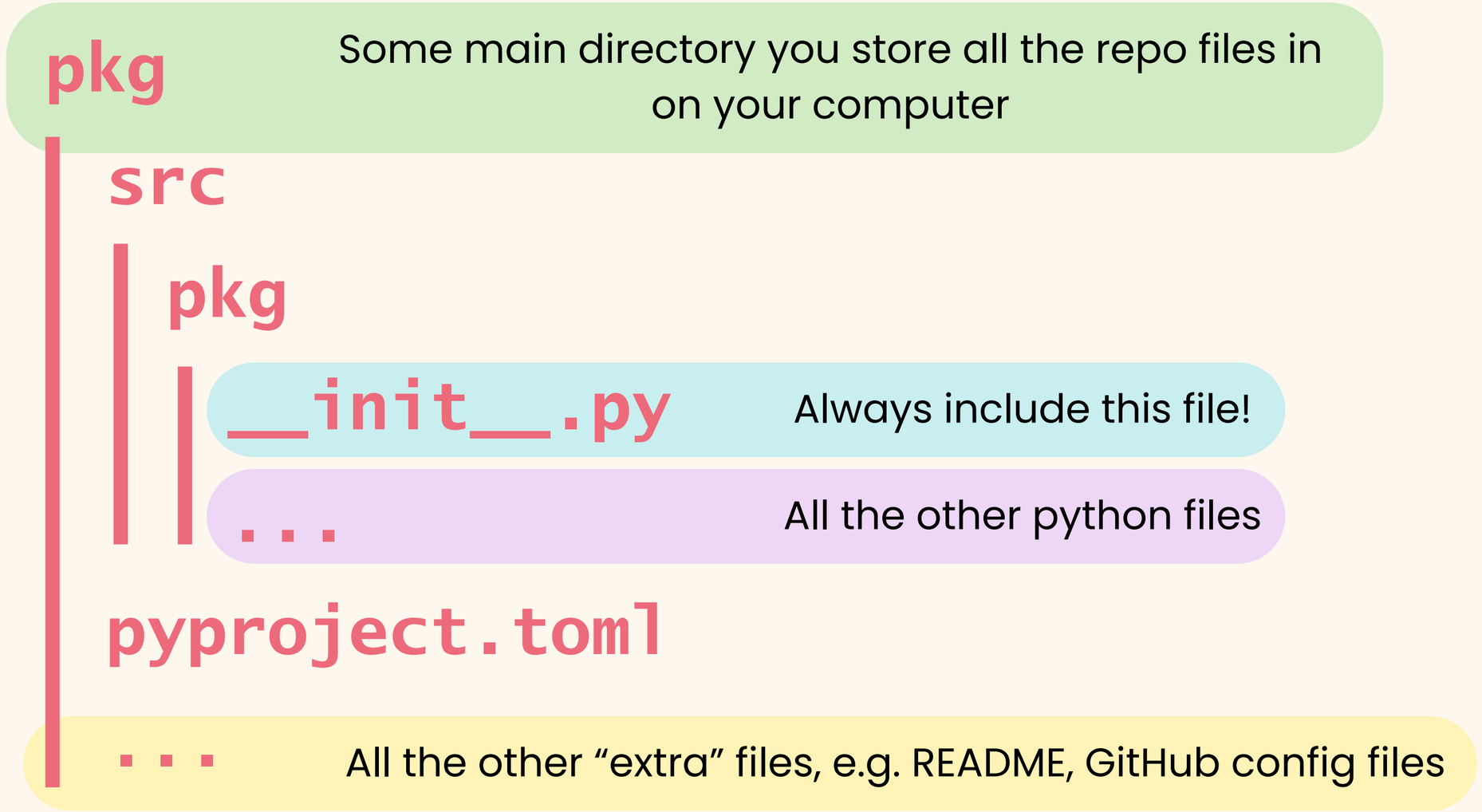
Step 1: install setuptools

```
pip install setuptools
```

Step 2: clone the tutorial repository

```
git clone https://github.com/maja-jablonska/pip-tutorial
```

Step 1: file tree structure



pyproject.toml

Check out the TOML file specifications:
toml.io

Generally very helpful:
https://packaging.python.org/

pyproject.toml is read by some **packaging tools**

pyproject.toml

Example packaging tool: **setuptools**

```
pip install setuptools
```

pyproject.toml

```
[build-system]
requires = ["setuptools >= 61.0"]
build-backend = "setuptools.build_meta"
```

pyproject.toml

Section name

```
[build-system]
requires = ["setuptools >= 61.0"]
build-backend = "setuptools.build_meta"
```

pyproject.toml

Setup tool package and version

```
[build-system]
requires = ["setuptools >= 61.0"]
build-backend = "setuptools.build_meta"
```

pyproject.toml

```
[project]
name = "your-project-name"
version = "1.0.0"
readme = "README.md"
dependencies = [
 "jax~=0.4.24",
  ...
]
```

pyproject.toml

```toml
[project]
name = "your-project-name"
version = "1.0.0"
readme = "README.md"
dependencies = [
 "jax~=0.4.24",
  ...
]
```

This is the name under which your package will be accessible in pip

pyproject.toml

```
[project]
name = "your-project-name"
version = "1.0.0"
readme = "README.md"
dependencies = [
 "jax~=0.4.24",
  ...
]
```

This will be updated as you add things

pyproject.toml

```
[project]
name = "your-project-name"
version = "1.0.0"
readme = "README.md"
dependencies = [
  "jax~=0.4.24",
  ...
]
```

List the packages that should be installed before installing this package

“~=” means: a version compatible with 0.4.24 release

pyproject.toml

```
[project]
name = "your-project-name"
version = "1.0.0"
readme = "README.md"
dependencies = [
  "jax~=0.4.24",
  ...
]
```

What's the difference between this and requirements.txt?

requirements.txt is rather for development purposes

pyproject.toml

```
[tool.setuptools.packages.find]
where = ["src"]
```

A **setuptools**-specific setting

Let's practice! [optional]

After filling in the pyproject.toml, try to install the package!

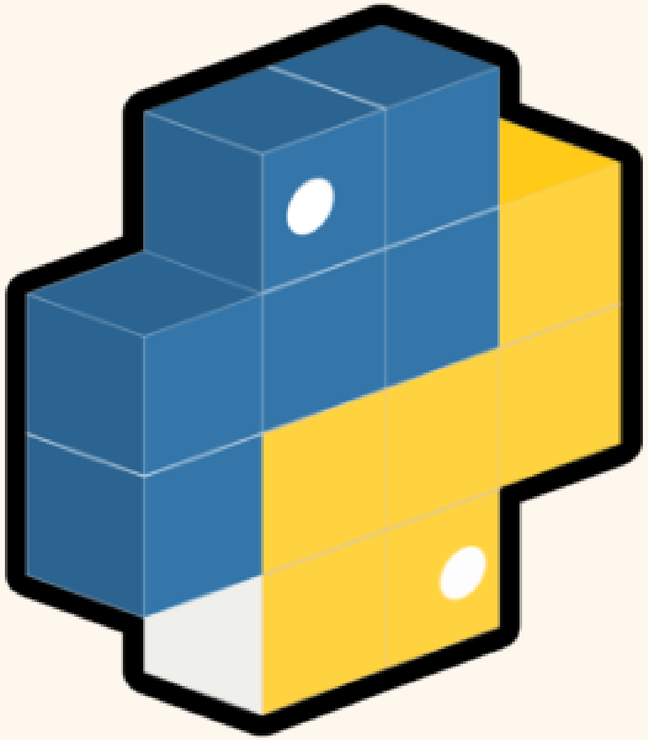
```
pip install .
```

You should see something along the lines of...

```
Installing collected packages: gasp-pip-tutorial
Successfully installed gasp-pip-tutorial-0.1
```

Step 2: publishing on PyPI

It’s a good idea to start with TestPyPI!
(https://test.pypi.org)



stellar-spice 0.4

```
pip install stellar-spice
```

Latest version

Released: May 29, 2024

No project description provided

Navigation

Project description

Release history

Download files

Verified details

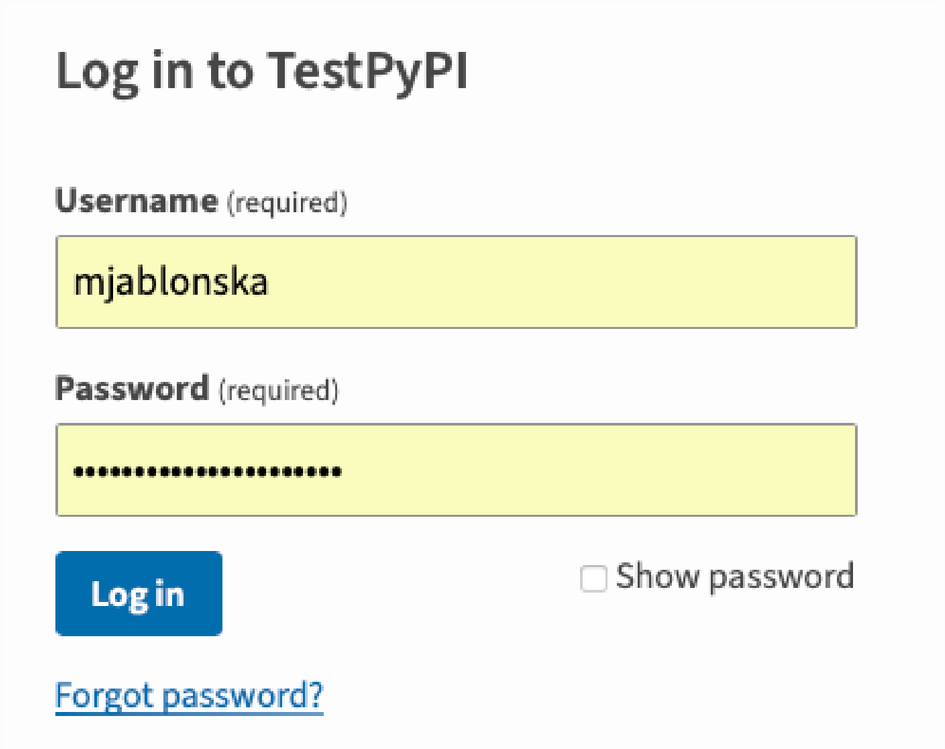
These details have been verified by PyPI

Project description

Let's practice! [optional]

Step 1: Register to Test PyPI

Step 2: publishing on PyPI



Step 3: GitHub tags and releases

Git supports **tags** as a way of marking some specific points in the repository history.

```
git tag -a v1.4 -m "my version 1.4"
```

Let's practice! [optional]

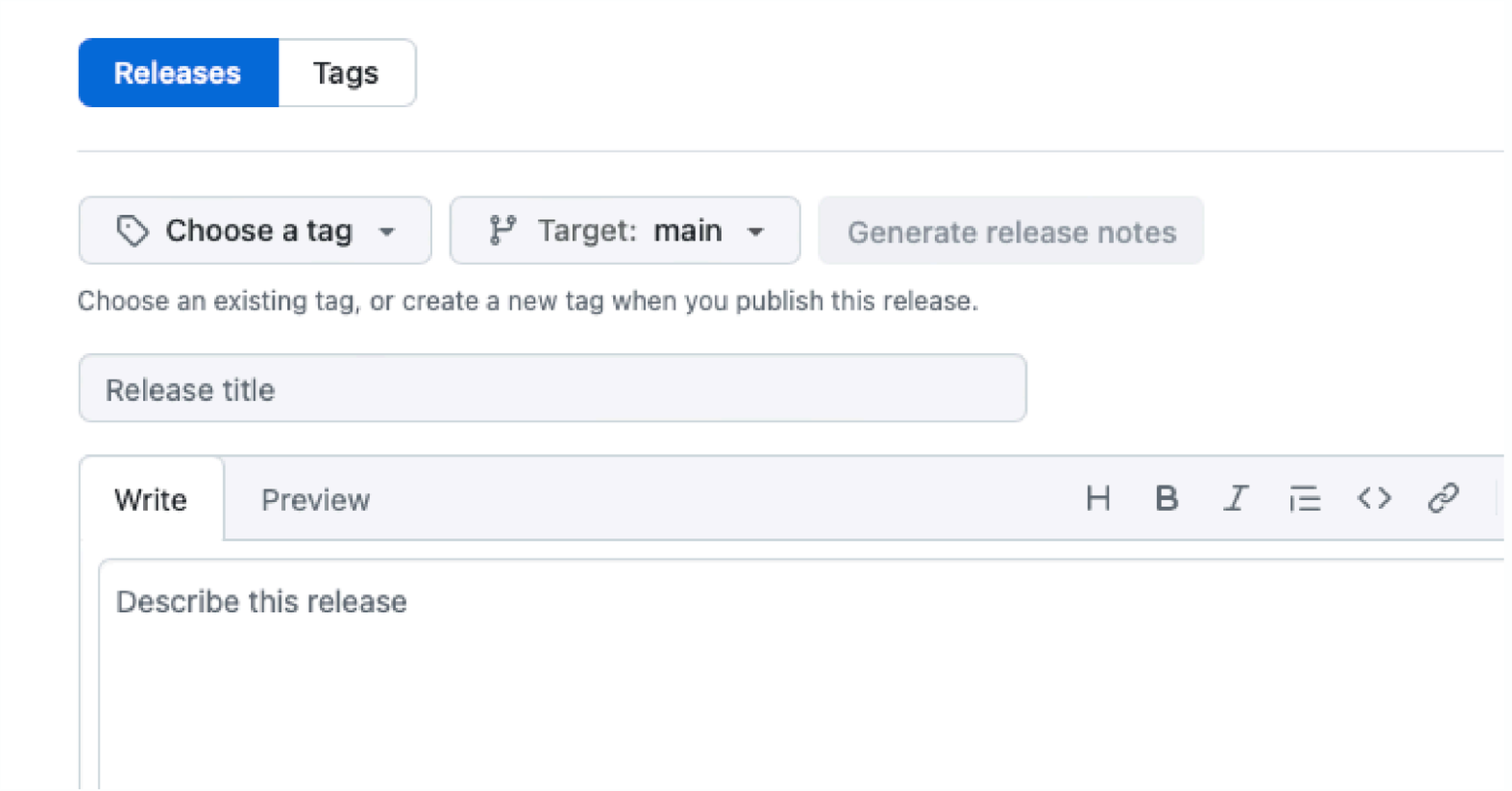
Step 1: create a new annotated tag

```
git tag -a v0.1 -m "version 0.1"
```

Step 2: push the tagged version

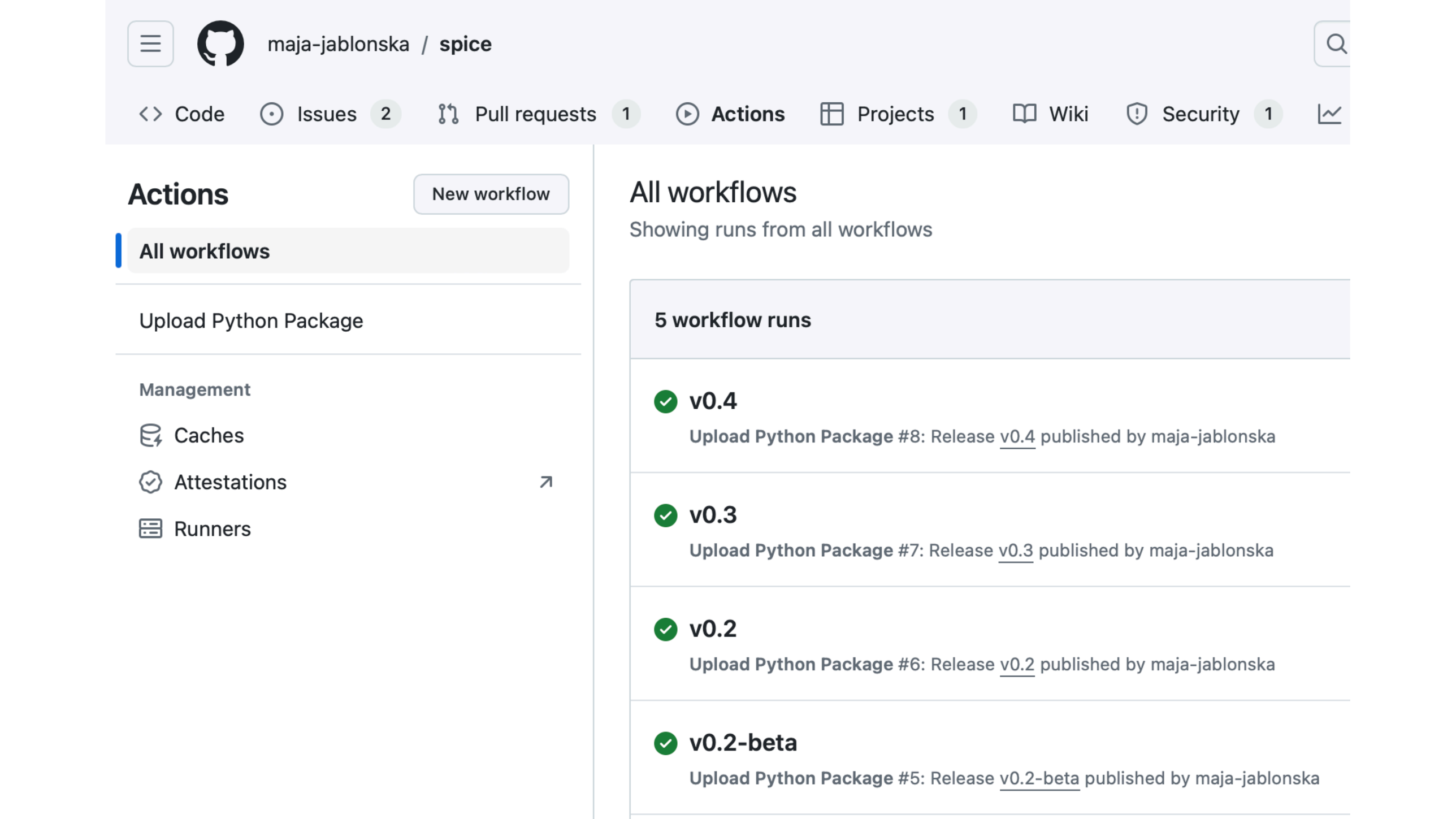
```
git push origin tag v0.1
```

Step 3: GitHub tags and releases

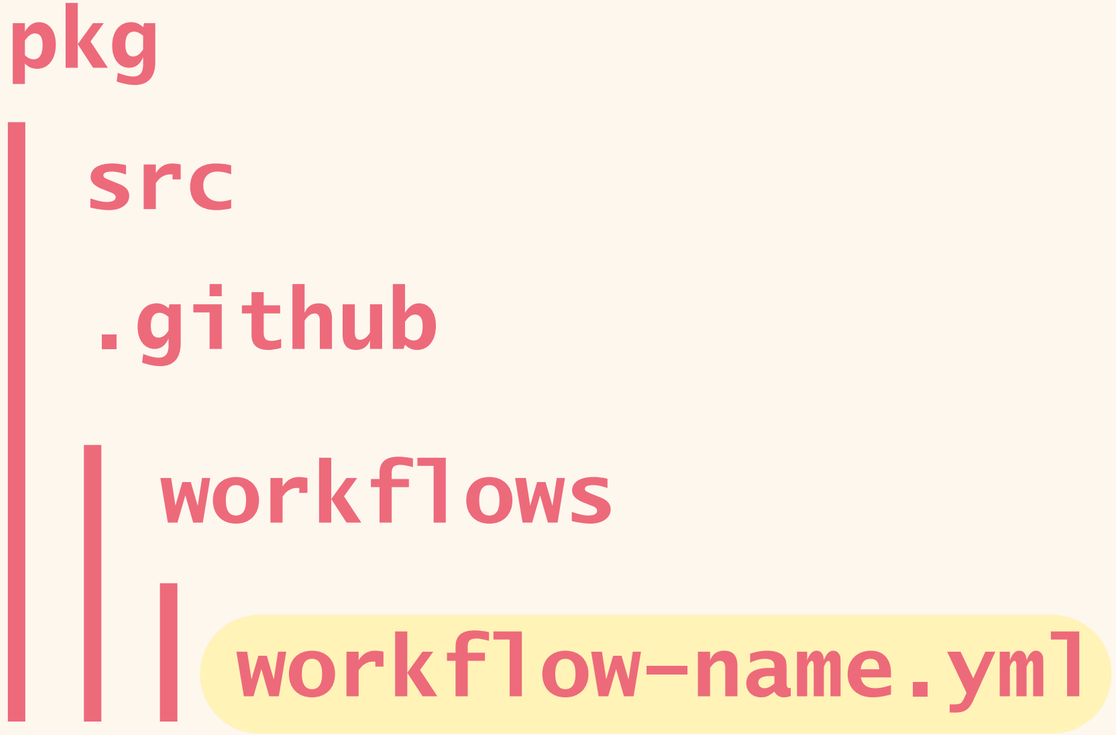


Step 4: GitHub integration

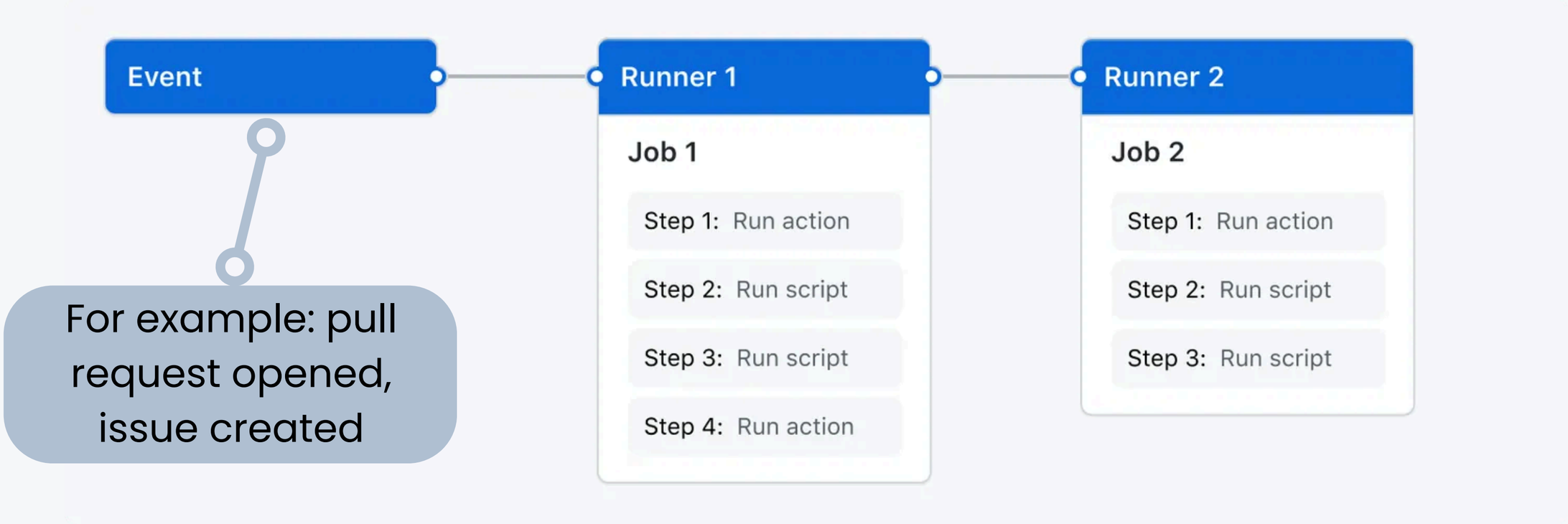
We can use **GitHub workflows** to automatically run tests, publish our package etc...


maja-jablonska / spice
Code
Issues 2
Pull requests 1
Actions
Projects 1
Wiki
Security 1
Actions
New workflow
All workflows
Upload Python Package
Management
Caches
Attestations
Runners
All workflows
Showing runs from all workflows
5 workflow runs
v0.4
Upload Python Package #8: Release v0.4 published by maja-jablonska
v0.3
Upload Python Package #7: Release v0.3 published by maja-jablonska
v0.2
Upload Python Package #6: Release v0.2 published by maja-jablonska
v0.2-beta
Upload Python Package #5: Release v0.2-beta published by maja-jablonska

Step 4: GitHub integration



Workflows



source: docs.github.com

Step 4: GitHub integration

```yaml
name: learn-github-actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm install -g bats
      - run: bats -v
```

workflow.yml

Step 4: GitHub integration

```yaml
name: learn-github-actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm install -g bats
      - run: bats -v
```

Name as it will appear in the “Actions” tab on GitHub

workflow.yml

Step 4: GitHub integration

```yaml
name: learn-github-actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm install -g bats
      - run: bats -v
```

Trigger for the workflow

workflow.yml

Step 4: GitHub integration

```yaml
name: learn-github-actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm install -g bats
      - run: bats -v
```

Runner configuration

workflow.yml

Step 4: GitHub integration

```yaml
name: learn-github-actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm install -g bats
      - run: bats -v
```

Reusing some existing workflow bits

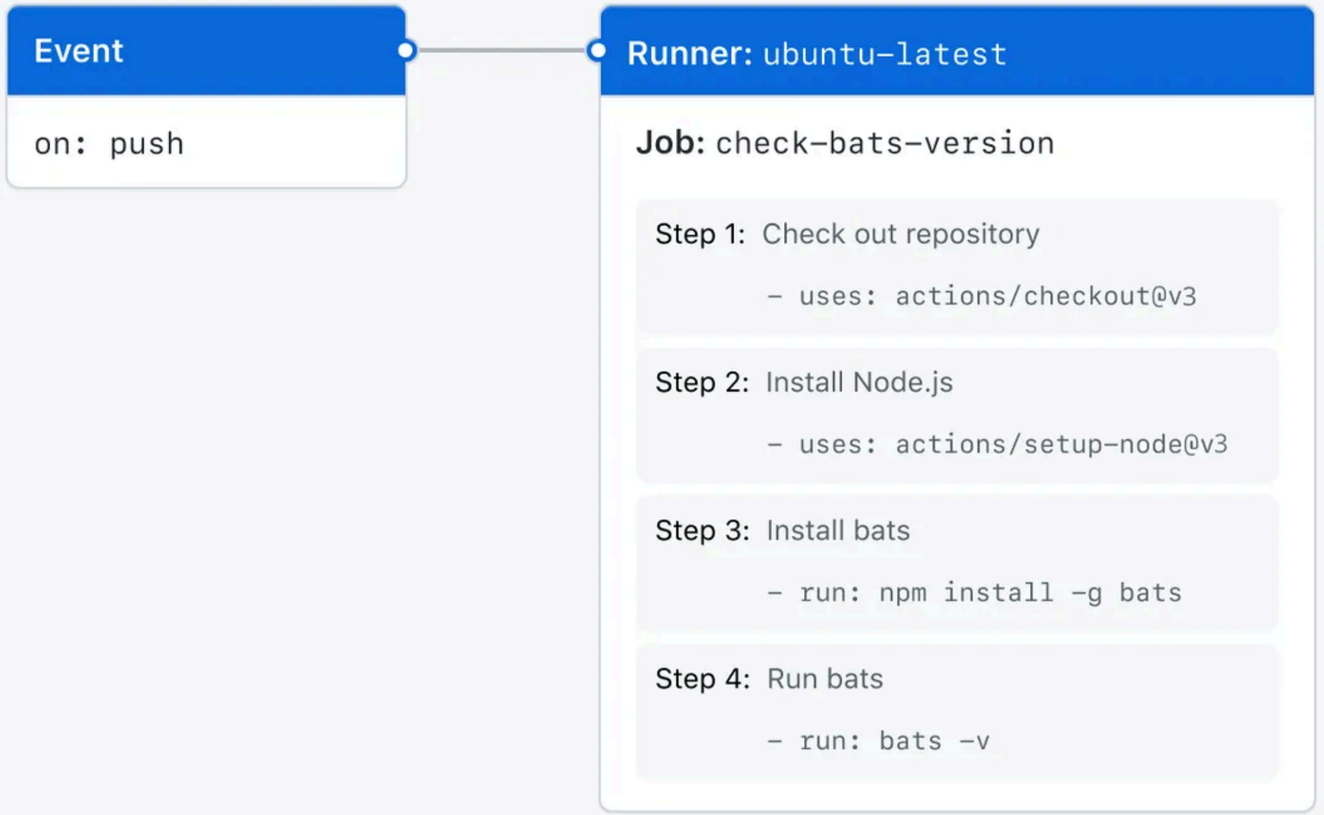
workflow.yml

Step 4: GitHub integration

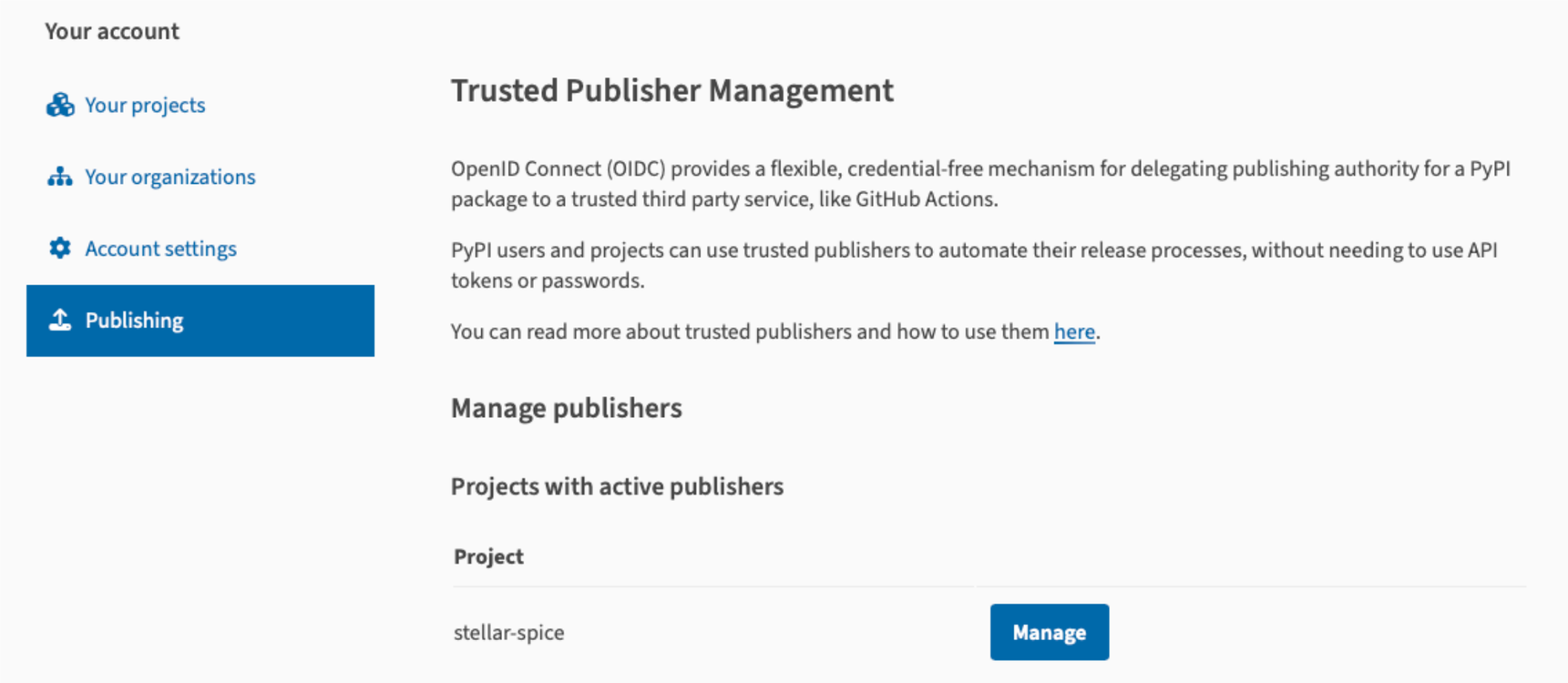
```yaml
name: learn-github-actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm install -g bats
      - run: bats -v
```

Commands to be run

workflow.yml


Event
on: push
Runner: ubuntu-latest
Job: check-bats-version
Step 1: Check out repository
- uses: actions/checkout@v3
Step 2: Install Node.js
- uses: actions/setup-node@v3
Step 3: Install bats
- run: npm install -g bats
Step 4: Run bats
- run: bats -v

Step 4: GitHub integration



Step 4: GitHub integration

GitHub | GitLab | Google | ActiveState

Read more about GitHub Actions' OpenID Connect support here.

PyPI Project Name (required)

project name

The project (on PyPI) that will be created when this publisher is used

Owner (required)

owner

The GitHub organization name or GitHub username that owns the repository

Repository name (required)

repository

The name of the GitHub repository that contains the publishing workflow

Workflow name (required)

workflow.yml

The filename of the publishing workflow. This file should exist in the `.github/workflows/` directory in the repository configured above.

Environment name (optional)

release

The name of the GitHub Actions environment that the above workflow uses for publishing. This should be configured under the repository's settings. While not required, a dedicated publishing environment is **strongly** encouraged, **especially** if your repository has maintainers with commit access who shouldn't have PyPI publishing access.

Add

Final step: trigger the workflow!

As long as the workflow.yml is connected to PyPI, the action should upload the package!

Thank you!