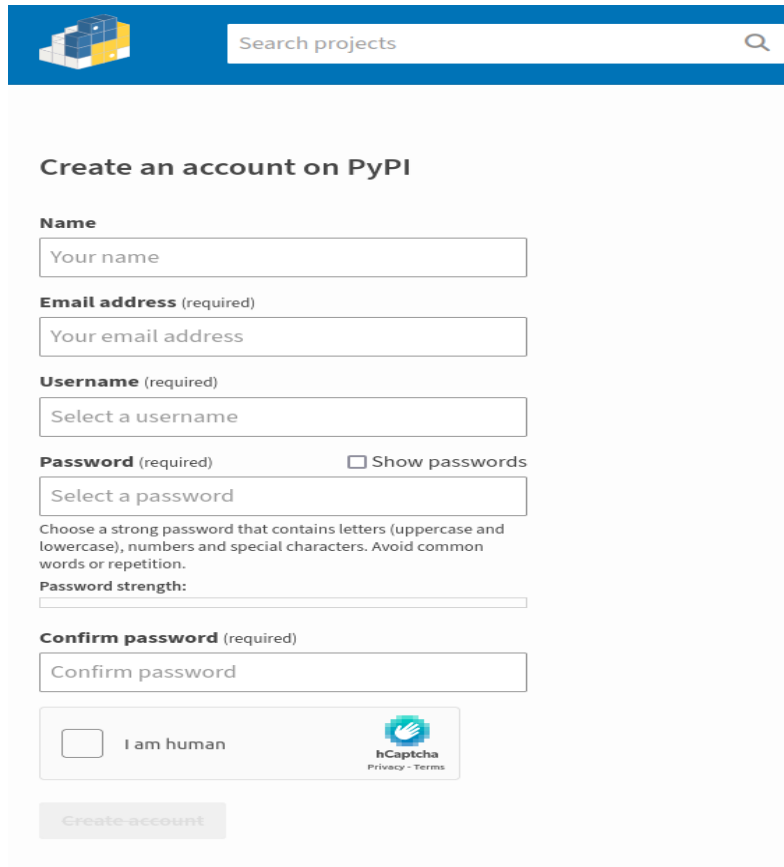


How to set up automatic code uploads from GitHub to PyPI:

Step 1: Create a PyPI account:

- Visit [<https://pypi.org/>](https://pypi.org/) and create a free account.

The image shows the 'Create an account on PyPI' page. At the top is a blue header with the PyPI logo and a search bar. The main content area is white and contains the following fields: 'Name' (text input), 'Email address (required)' (text input), 'Username (required)' (text input), 'Password (required)' (text input) with a 'Show passwords' checkbox, and 'Confirm password (required)' (text input). Below the password field is a note: 'Choose a strong password that contains letters (uppercase and lowercase), numbers and special characters. Avoid common words or repetition.' and a 'Password strength' indicator. At the bottom, there is a checkbox for 'I am human' next to a hCaptcha logo, and a 'Create account' button.

Step 2: Install required tools:

- Install `twine` to securely upload packages to PyPI:

```
```bash
pip install twine
```
```

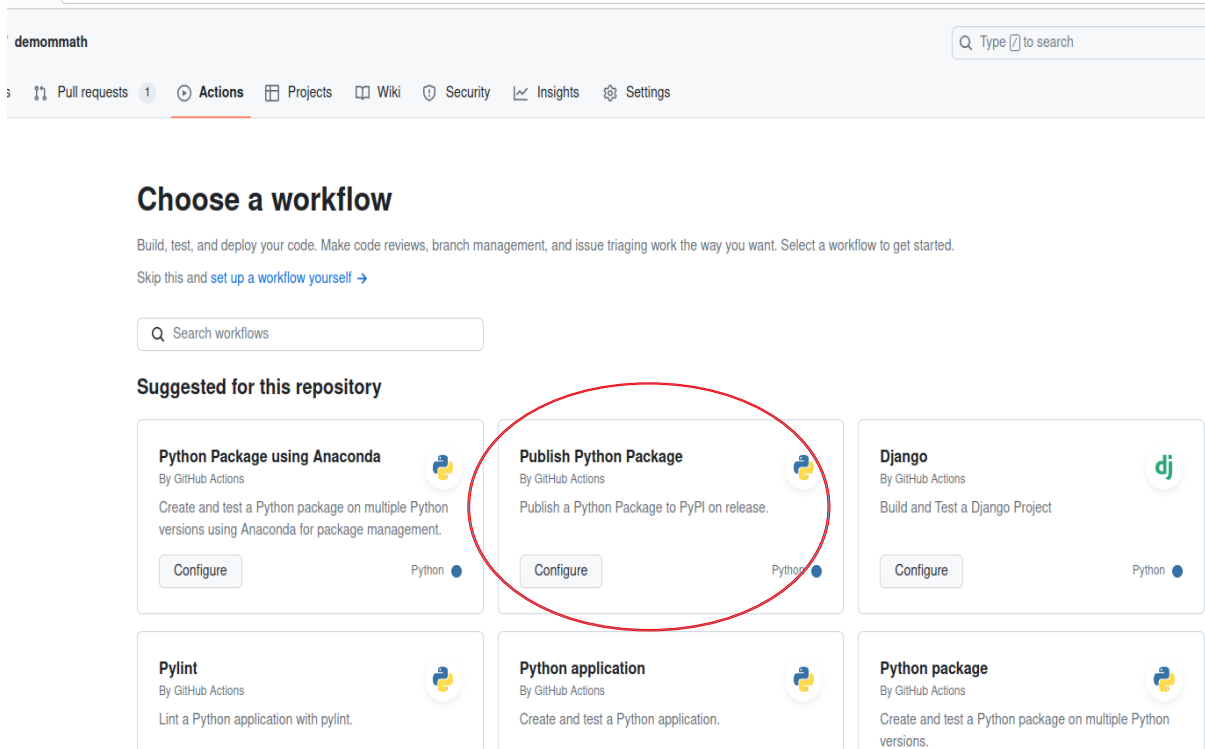
Step 3: Prepare your Python package:

- Ensure your repository has a `setup.py` or `pyproject.toml` file with package metadata.
- Define a version for your package using a version control system (VCS) tag.

Step 4: Create a GitHub Actions workflow:

- In your GitHub repository, navigate to the "Actions" tab.
- Select "Set up a workflow yourself".

- Choose a suitable starter workflow, such as "Publish Python Package".



Step 5: Configure the workflow:

- Replace placeholders with your PyPI credentials and package information.
- Define a trigger, such as pushing a release tag, to initiate the workflow.

- Example workflow using `pypi-action`:

```
``yaml
name: Publish Python Package
on:
  push:
    tags:
      - 'v*' # Trigger on release tags
jobs:
  build-and-publish:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set up Python
        uses: actions/setup-python@v4
        with:
```

```

python-version: '3.x'
- name: Install dependencies
  run: pip install -r requirements.txt
- name: Build package
  run: python setup.py sdist bdist_wheel
- name: Publish to PyPI
  uses: pypa/gh-action-pypi-publish@master
  with:
    password: ${{ secrets.PYPI_API_TOKEN }}
...

```

Code
Blame
32 lines (26 loc) · 699 Bytes
Code 55% faster with G

```

1  name: Publish Python Package
2
3  on:
4    release:
5      types:
6        - created
7
8  jobs:
9    publish:
10     runs-on: ubuntu-latest
11
12     steps:
13       - name: Checkout code
14         uses: actions/checkout@v2
15
16       - name: Set up Python
17         uses: actions/setup-python@v2
18         with:
19           python-version: 3.x
20
21       - name: Install dependencies
22         run: |
23           python -m pip install --upgrade pip
24           pip install setuptools wheel twine
25
26       - name: Build and publish
27         run: |
28           python setup.py sdist bdist_wheel
29           python -m twine upload --repository pypi dist/*
30     env:
31       TWINE_USERNAME: __token__
32       TWINE_PASSWORD: ${{ secrets.PYPI_API_TOKEN }}

```

Step 6: Store PyPI credentials securely:

- In your GitHub repository's "Settings" -> "Secrets", create a secret named `PYPI_API_TOKEN` and store your PyPI API token there.

demommath

Search Type to search

Pull requests 1 Actions Projects Wiki Security Insights **Settings**

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security and analysis

Deploy keys

Secrets and variables

Actions

Codespaces

Dependabot

Integrations

GitHub Apps

Email notifications

Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for **non-sensitive** data. [Learn more about variables](#).

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

Secrets Variables

Environment secrets

Manage environment secrets

| Name | Environment | Last updated |
|------------|-------------|--------------|
| PYPI_TOKEN | pypi_token | 4 days ago |

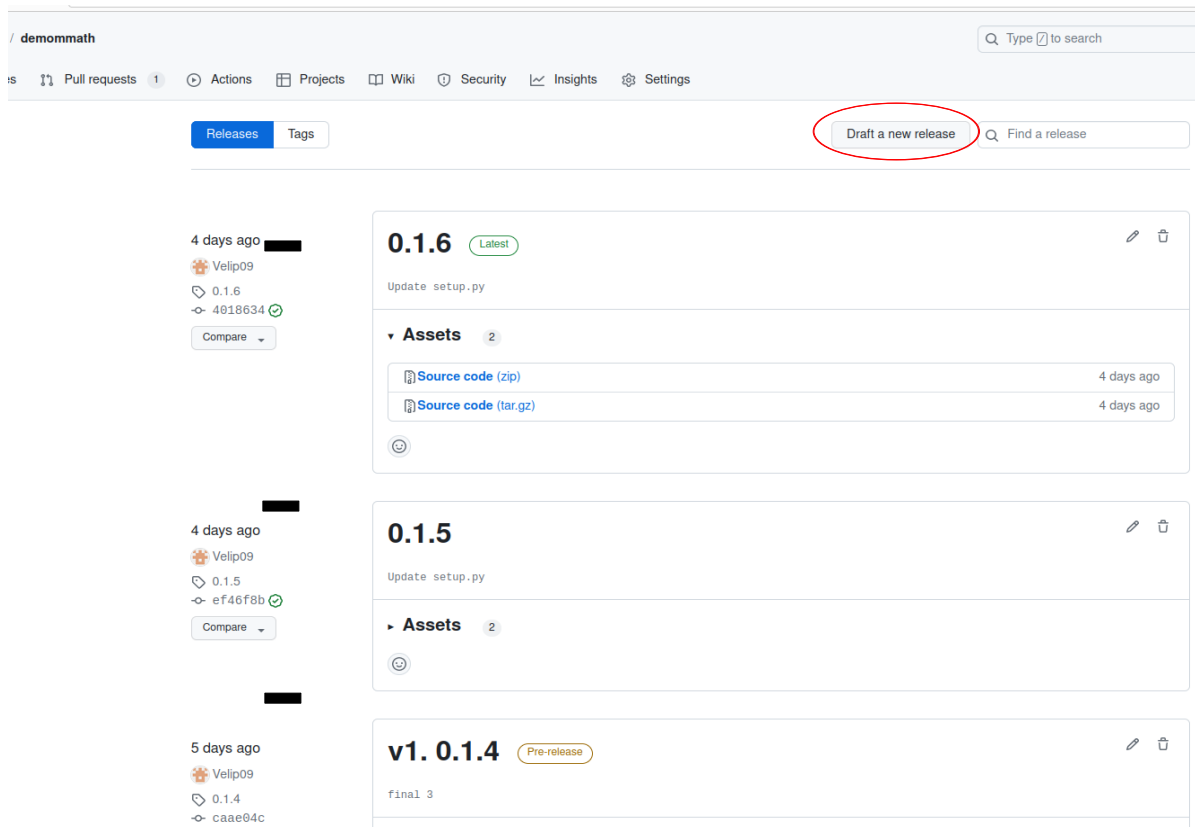
Repository secrets

| Name | Last updated |
|----------------|--------------|
| PYPI_API_TOKEN | 4 days ago |
| PYPI_TOKEN | 4 days ago |

New repository secret

Step 7: Push a release tag:

- When ready, create a new Git tag with the desired version (e.g., `git tag v1.0.0`).
- Push the tag to GitHub: `git push origin v1.0.0`.



Step 8: Monitor the workflow:

- The workflow will automatically execute and upload your package to PyPI.
- You can check progress in the "Actions" tab.

