

# **Trusted Publishing & Digital Attestations in the OSS Ecosystem**

**\$ whoami**

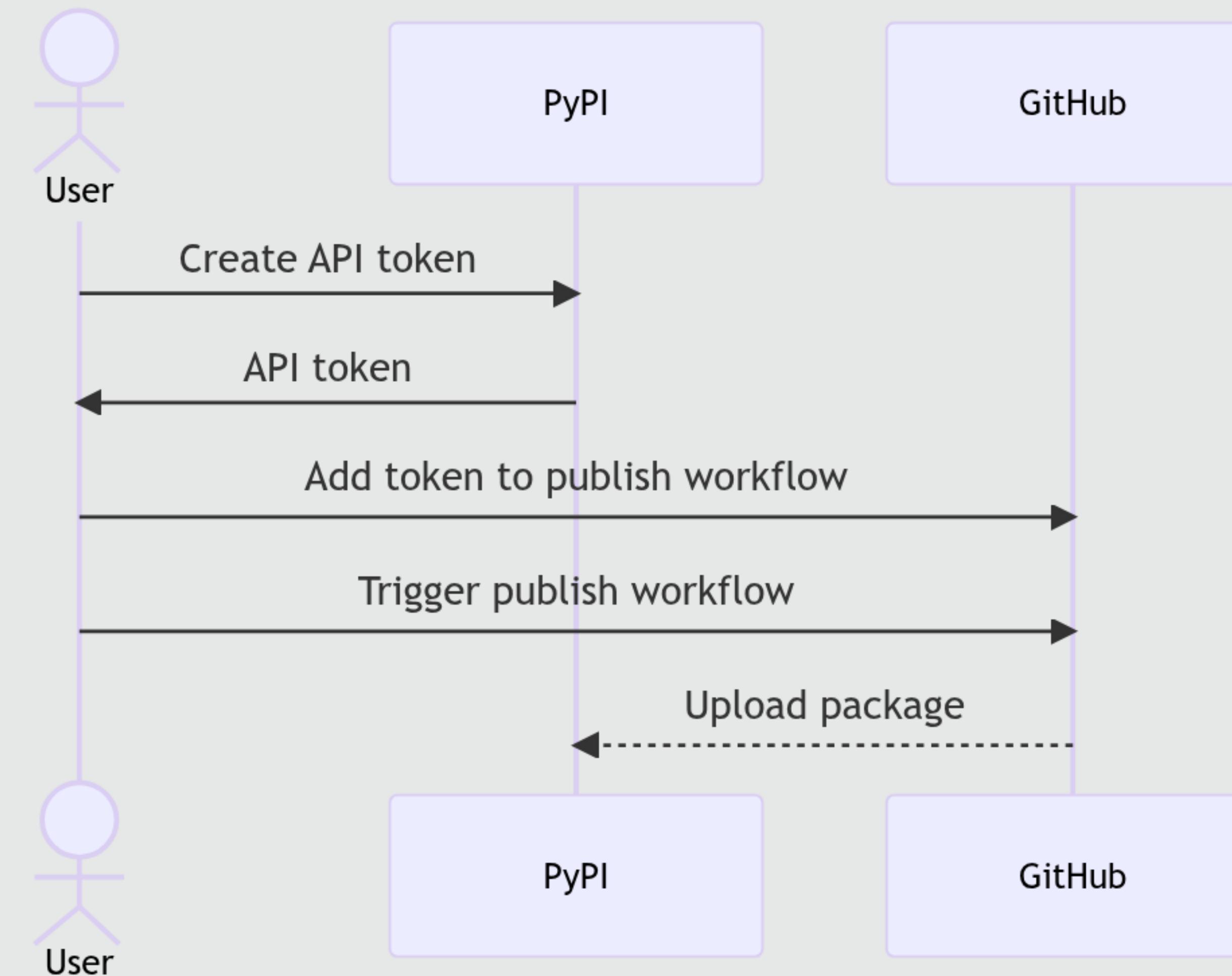


## Jordan Duabe

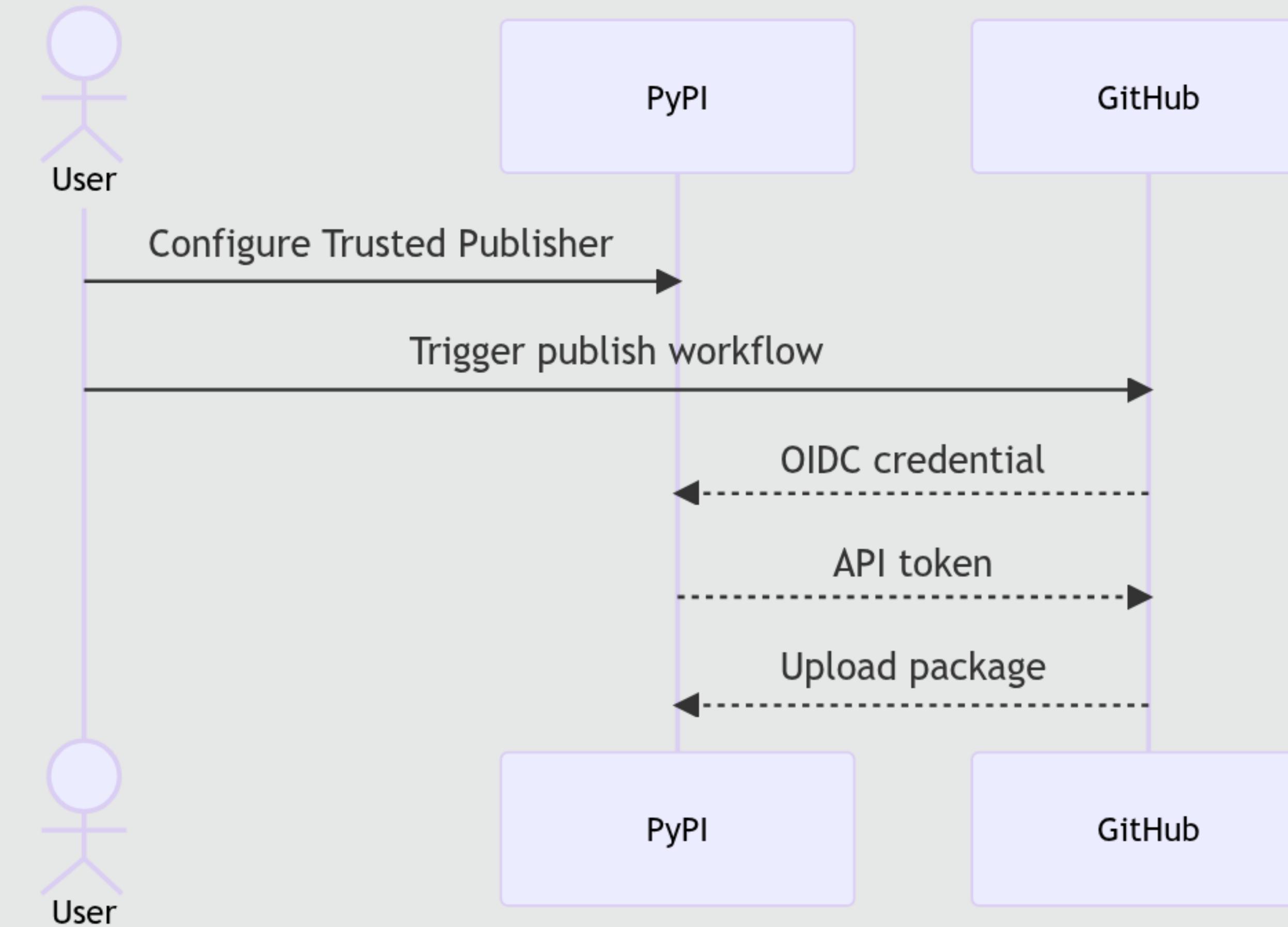
- Software Engineer
- Mechanical keyboard enthusiast
- [jordan@jduabe.dev](mailto:jordan@jduabe.dev)
- [@j4ckofalltrades@infosec.exchange](mailto:@j4ckofalltrades@infosec.exchange)

*Trusted Publishing*

# *API Token publish flow*



# *Trusted Publishing flow*



# Configure a Trusted Publisher in PyPI

Add a new publisher

[GitHub](#)   [GitLab](#)   [Google](#)   [ActiveState](#)

Read more about GitHub Actions' OpenID Connect support [here](#).

**Owner** (required)

The GitHub organization name or GitHub username that owns the repository

**Repository name** (required)

The name of the GitHub repository that contains the publishing workflow

**Workflow name** (required)

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

**Environment name** (optional)

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**

Source: <https://docs.pypi.org/trusted-publishers/adding-a-publisher/>

# *Publishing with a Trusted Publisher*

```
# https://github.com/pypa/gh-action-pypi-publish
jobs:
  pypi-publish:
    name: upload release to PyPI
    runs-on: ubuntu-latest
    + # Specifying a GitHub environment is optional, but strongly encouraged
    + environment: pypi
    + permissions:
    + # IMPORTANT: this permission is mandatory for Trusted Publishing
    + id-token: write
  steps:
    # retrieve your distributions here
    - name: Publish package distributions to PyPI
      uses: pypa/gh-action-pypi-publish@v1.11.0
    - with:
    -   username: __token__
    -   password: ${{ secrets.PYPI_TOKEN }}
```

Source: <https://docs.pypi.org/trusted-publishers/using-a-publisher/>

# *Benefits over long-lived API tokens*

- Can be accidentally exposed in CI logs or configuration files
- Require manual rotation and management
- If compromised, they provide persistent access until revoked
- Oftentimes do not have granularity over permissions

# *Benefits over long-lived API tokens*

- Can be accidentally exposed in CI logs or configuration files
- Require manual rotation and management
- If compromised, they provide persistent access until revoked
- Oftentimes do not have granularity over permissions



# *Adoption in the OSS ecosystem*

- Ruby's RubyGems
- Dart's Pub Package Registry
- Rust's Crates
- JavaScript's npm and JSR

# Supply Chain Attack on Nx build system package

## Remediation and Preventative Measures Taken

We have taken the following actions to remediate this issue, prevent further issues, also ensure validity of future packages.

- Deprecated all malicious package versions
- Restored `21.4.1` (a valid version) as `latest`
- Revoked possibly compromised personal account access, even though single compromised token seems most likely at this time
- Rotated all team NPM and GitHub tokens
- Audit GitHub and NPM activities across the organization for suspicious activities
- Updated Publish access for `nx` to require 2FA or automation
- Posted this advisory
- The `nx` package now requires [Trusted Providers](#) methodology of publishing via our `.github/publish.yml` workflow in the `nrwl/nx` repo.
- Remove NPM tokens from our pipeline now that we're using Trusted Providers on NPM
- All NPM packages under Nx (the company) including `nx` have been set to require 2FA and cannot be published with access tokens
- All Github secrets on the `nrwl/nx` repo have been rotated. If any other secrets were compromised, they are no longer valid.
- We are continuing to assess how if any other malicious activity was done with the other secrets we have in Github but have not found any further malicious activity at this time.
- All branches containing the vulnerable pipeline on `nrwl/nx` have been updated to not have the vulnerable pipeline.
- The `nx` repo now also will require team members to approve workflows triggered by external contributors. This will block workflows unknown to us from slipping through.
- CodeQL was enabled in the Nx repo. This will catch similar vulnerabilities in PRs before they are merged.
- We have added `SECURITY.md` instructions detailing the proper way to notify us privately about future security issues in Nx.

This advisory will be updated when there is more information available.

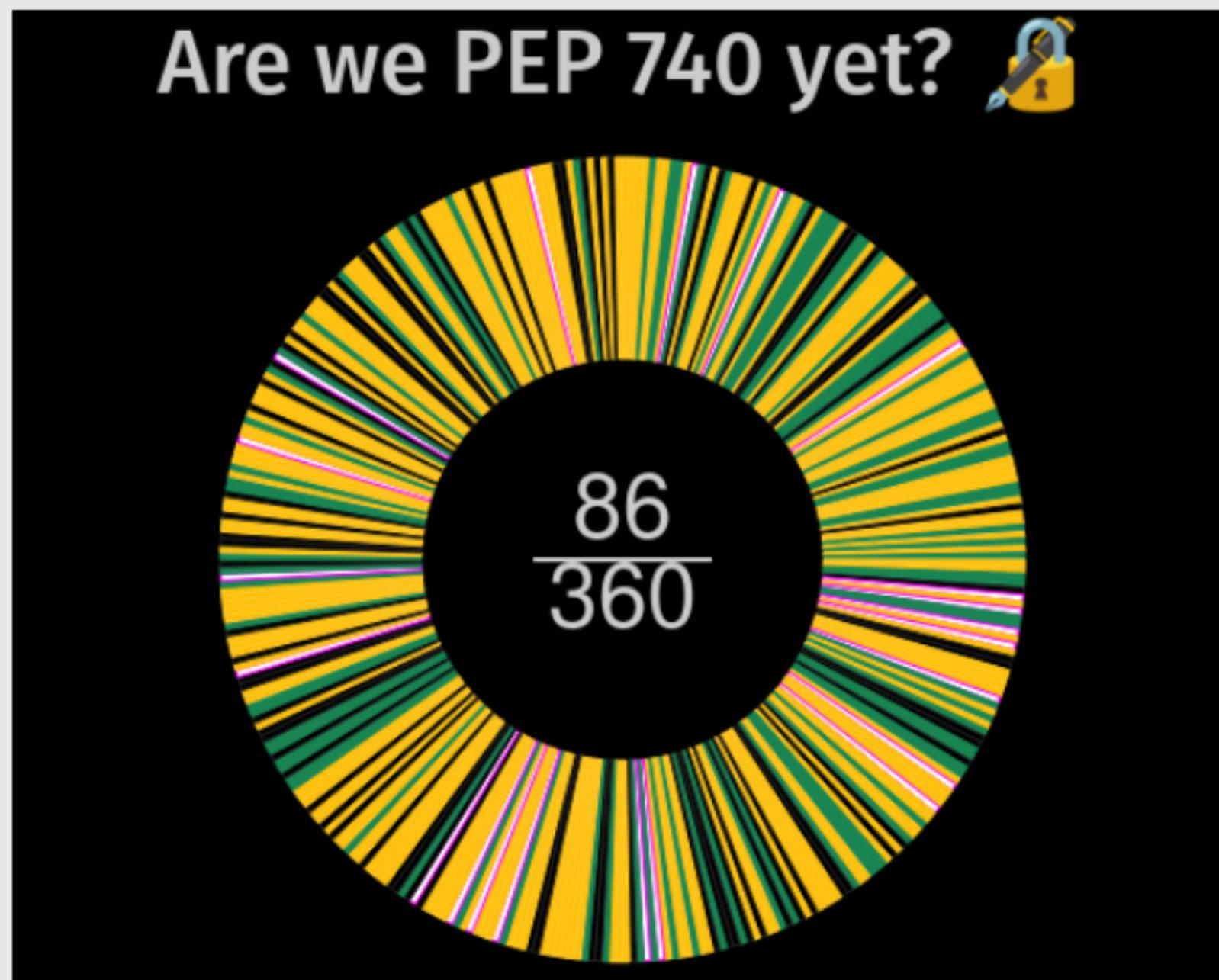
Source: <https://github.com/nrwl/nx/security/advisories/GHSA-cxm3-wv7p-598c>

# *Digital Attestations*

# PEP 740 – Index support for digital attestations

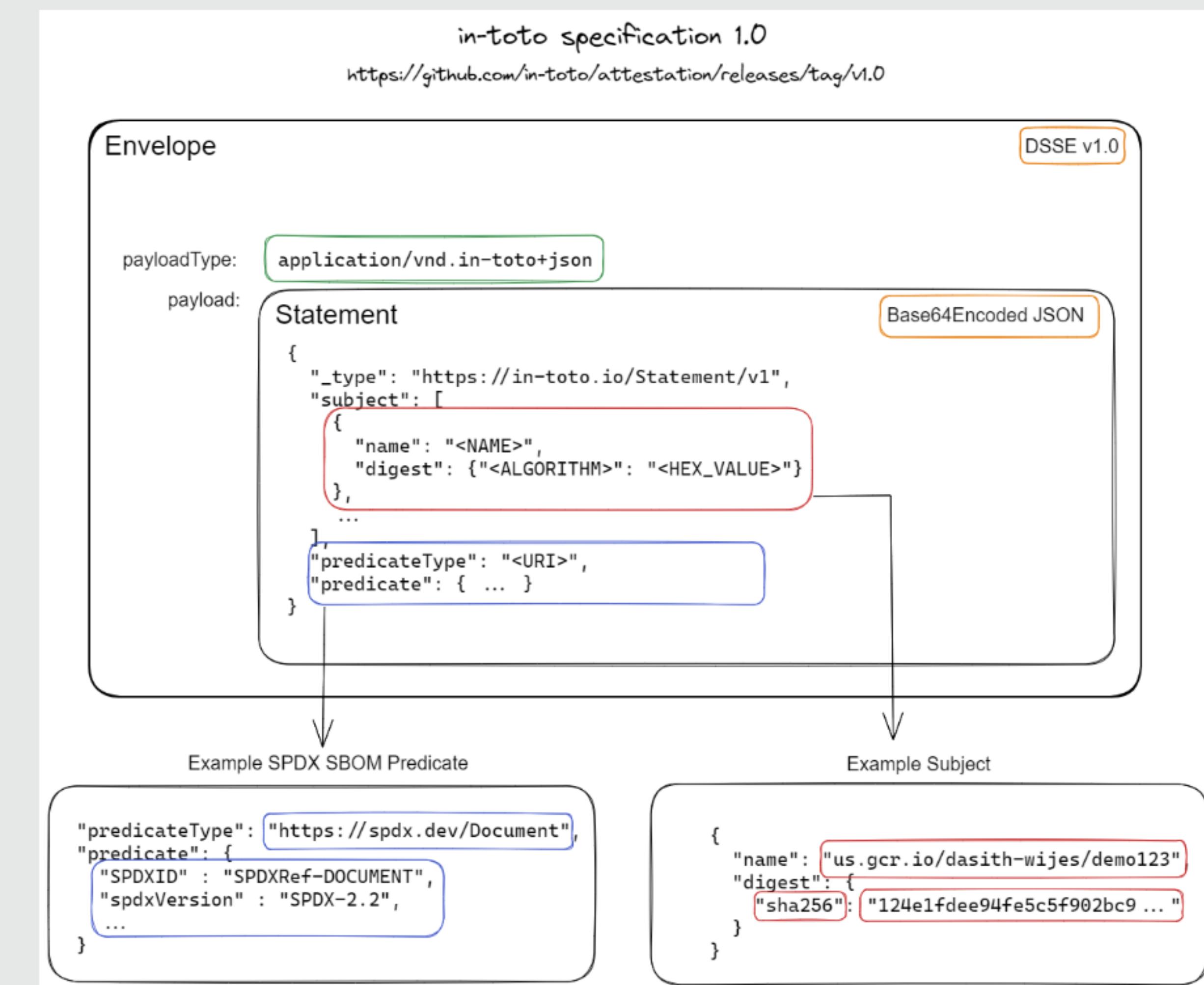
PyPI uses the [in-toto Attestation Framework](#) for the attestations it accepts.

- [SLSA Provenance](#) (*build provenance*)
- [PyPI Publish](#) (*publish provenance*)



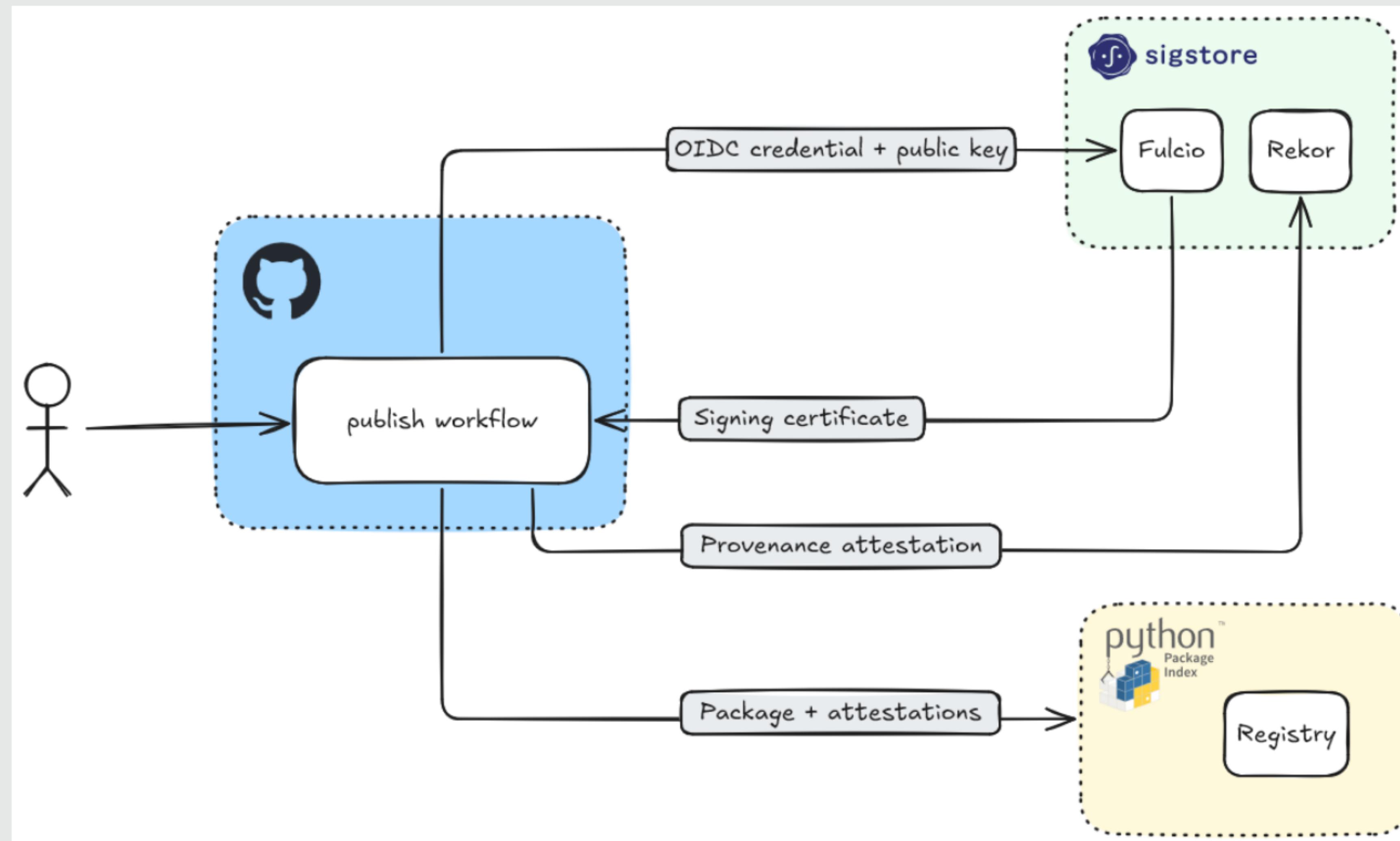
Source: <https://trailofbits.github.io/are-we-pep740-yet/>

# *in-toto* Attestation Framework Spec



Source: <https://github.com/in-toto/attestation/blob/main/spec/README.md>

# Trusted Publishing with Sigstore



# *Generating and uploading attestations*

```
# https://github.com/pypa/gh-action-pypi-publish
# Attestations enabled by default as of v.1.11.0 and newer
jobs:
  pypi-publish:
    name: upload release to PyPI
    runs-on: ubuntu-latest
    environment: pypi
    permissions:
      id-token: write
    steps:
      # retrieve your distributions here
      - name: Publish package distributions to PyPI
        uses: pypa/gh-action-pypi-publish@v1.11.0
```

Source: <https://docs.pypi.org/trusted-publishers/using-a-publisher/>

# Provenance in PyPI

## Provenance

The following attestation bundles were made for `powerline_k8s-1.5.4.tar.gz`:

Publisher:  [pypi-publish.yml](#) on [j4ckofalltrades/powerline-k8s](#)

### Attestations:

*Values shown here reflect the state when the release was signed and may no longer be current.*

- Statement:
  - Statement type: <https://in-toto.io/Statement/v1>
  - Predicate type: <https://docs.pypi.org/attestations/publish/v1>
  - Subject name: `powerline_k8s-1.5.4.tar.gz`
  - Subject digest: `4c491b88f3cb872720e38f5890565f6f1c93f228a51f52174ee486a30417807d`
  - Sigstore transparency entry: [245631303](#)
  - Sigstore integration time: Jun 21, 2025, 7:22:04 PM



### Source repository:

- Permalink: [j4ckofalltrades/powerline-k8s@8ee093f25d3d6515db6d28d41d8e4a935f2845c1](#)
- Branch / Tag: [refs/tags/v1.5.4](#)
- Owner: <https://github.com/j4ckofalltrades>
- Access: `public`

### Publication detail:

- Token Issuer: <https://token.actions.githubusercontent.com>
- Runner Environment: `github-hosted`
- Publication workflow: [pypi-publish.yml@8ee093f25d3d6515db6d28d41d8e4a935f2845c1](#)
- Trigger Event: `push`

Source: [https://pypi.org/project/powerline-k8s/#powerline\\_k8s-1.5.4.tar.gz](https://pypi.org/project/powerline-k8s/#powerline_k8s-1.5.4.tar.gz)

# *Verifying provenance with PyPI's Integrity API*

```
# GET /integrity/<project>/<version>/<filename>/provenance
$ http get https://pypi.org/integrity/powerline-k8s/1.5.4/powerline_k8s-1.5.4.tar.gz/provenance \
> | jq -r '.attestation_bundles[0].attestations[0].envelope.statement' \
> | base64 -d \
> | jq

{
  "_type": "https://in-toto.io/Statement/v1",
  "subject": [
    {
      "name": "powerline_k8s-1.5.4.tar.gz",
      "digest": {
        "sha256": "4c491b88f3cb872720e38f5890565f6f1c93f228a51f52174ee486a30417807d"
      }
    }
  ],
  "predicateType": "https://docs.pypi.org/attestations/publish/v1",
  "predicate": null
}
```

# *Adoption in the OSS ecosystem: npm*

## Provenance

Built and signed on



**GitHub Actions**

[View build summary](#)

Source Commit

[github.com/sigstore/sigstore-js@26d1651](https://github.com/sigstore/sigstore-js@26d1651)

Build File

<.github/workflows/release.yml>

Public Ledger

[Transparency log entry](#)

```
$ npm audit signatures  
audited 507 packages in 7s
```

507 packages have verified registry signatures

10 packages have verified attestations

Source: <https://docs.npmjs.com/generating-provenance-statements>

# npm package provenance stats

## npm package provenance stats

24 / 500

**What is this list?**

This site shows the top 500 most-downloaded packages on [npm](#) showing which have been uploaded with attestations.

- **Green** packages (5%) with a 📜 have attestations for their latest release
- Grey packages (29%) with a 🕒 come from a supported CI/CD provider but were uploaded before attestations were available
- **Yellow** packages (66%) with a ━ come from a supported CI/CD provider but have no attestations (yet!)
- **Pink** packages (0%) with a ✗ come from an unsupported CI/CD provider

Additionally packages with a 📄 use Trusted Publishing instead of long-lived API tokens.

Refer to the npm docs for more details about [Trusted Publishers](#) and [generating provenance statements](#).

Search packages...

| Package        | Status | Release Date                   |
|----------------|--------|--------------------------------|
| semver         | 📜      | v7.7.2 released on 2025-05-12  |
| ansi-styles    | 🕒      | v6.2.1 released on 2022-10-12  |
| debug          | ━      | v4.4.1 released on 2025-05-13  |
| chalk          | ━      | v5.6.0 released on 2025-08-17  |
| supports-color | ━      | v10.2.0 released on 2025-08-17 |
| minimatch      | ━      | v10.0.3 released on 2025-06-12 |
| ms             | 🕒      | v2.1.3 released on 2020-12-08  |

Last updated Saturday, August 30, 2025 at 10:23:54 UTC. (Updated daily.)

Inspired by [Are we PEP 740 yet?](#). Source on [GitHub](#).

Source: <https://jduabe.dev/npm-package-provenance-stats>

# *Sigstore usage in the OSS ecosystem*

# *Verifying Sigstore signatures in the OSS ecosystem*

- PEP 761
- GitHub Artifact Attestations
- Homebrew provenance

# *PEP 761 – Deprecating PGP signatures for CPython artifacts*

| Release | PEP                     | Release manager      | OIDC Issuer   |
|---------|-------------------------|----------------------|---|
| 3.7     | <a href="#">PEP 537</a> | nad@python.org       | <a href="https://github.com/login/oauth">https://github.com/login/oauth</a> |
| 3.8     | <a href="#">PEP 569</a> | lukasz@langa.pl      | <a href="https://github.com/login/oauth">https://github.com/login/oauth</a> |
| 3.9     | <a href="#">PEP 596</a> | lukasz@langa.pl      | <a href="https://github.com/login/oauth">https://github.com/login/oauth</a> |
| 3.10    | <a href="#">PEP 619</a> | pablogsal@python.org | <a href="https://accounts.google.com">https://accounts.google.com</a>       |
| 3.11    | <a href="#">PEP 664</a> | pablogsal@python.org | <a href="https://accounts.google.com">https://accounts.google.com</a>       |
| 3.12    | <a href="#">PEP 693</a> | thomas@python.org    | <a href="https://accounts.google.com">https://accounts.google.com</a>       |
| 3.13    | <a href="#">PEP 719</a> | thomas@python.org    | <a href="https://accounts.google.com">https://accounts.google.com</a>       |
| 3.14    | <a href="#">PEP 745</a> | hugo@python.org      | <a href="https://github.com/login/oauth">https://github.com/login/oauth</a> |

```
# download release artifact and verification bundle
$ wget https://www.python.org/ftp/python/3.14.0/Python-3.14.0rc1.tgz
$ wget https://www.python.org/ftp/python/3.14.0/Python-3.14.0rc1.tgz.sigstore

# verify using a sigstore client
$ uvx --prerelease=allow sigstore verify identity \
> --bundle Python-3.14.0rc1.tgz.sigstore \
> --cert-identity hugo@python.org \
> --cert-oidc-issuer https://github.com/login/oauth \
> Python-3.14.0rc1.tgz
$ OK: Python-3.14.0rc1.tgz
```

Source: <https://www.python.org/downloads/metadata/sigstore/>

# GitHub attestations and brew provenance



```
# github artifact attestations can be verified using the gh cli  
$ gh attestation verify my-artifact.tar.gz --owner my-organization
```

```
# homebrew provenance is still in beta  
$ export HOMEBREW_VERIFY_ATTESTATIONS=1  
$ brew install uv
```

```
# use verify manually (depends on gh cli)  
$ brew verify uv
```

Source: <https://github.blog/news-insights/product-news/introducing-artifact-attestations-now-in-public-beta/>

# Takeaways

- Trusted publishing improves security by removing the need for a preexisting shared secret
- Digital attestations provide verifiable provenance for Python package distributions
- Downstream verification -- integration with tooling like pip and uv
- Not a *panacea* -- one step that improves software supply chain security

*Thank You*

