

Skip to content

[![logo](../assets/logo.png)](https://pypi.org "PyPI Docs")

PyPI Docs

Project Metadata

Initializing search

[GitHub](https://github.com/pypi/warehouse "Go to repository")

[![logo](../assets/logo.png)](https://pypi.org "PyPI Docs") PyPI Docs

[GitHub](https://github.com/pypi/warehouse "Go to repository")

- * [Welcome to PyPI User Documentation](..)
- * Organization Accounts Organization Accounts
 - * [About](..organization-accounts/)
 - * [FAQs](..organization-accounts/org-acc-faq/)
 - * [Roles and Entities](..organization-accounts/roles-entities/)
 - * Actions Actions
 - * [Billing Actions](..organization-accounts/actions/billing-actions/)
 - * [Organization Actions](..organization-accounts/actions/org-actions/)
 - * [Project Actions](..organization-accounts/actions/project-actions/)
 - * [Team Actions](..organization-accounts/actions/team-actions/)
 - * [Pricing and Payments](..organization-accounts/pricing-and-payments/)

* Trusted Publishers Trusted Publishers

* [Getting Started](../trusted-publishers/)

* [Adding a Trusted Publisher to an Existing PyPI Project](../trusted-publishers/adding-a-publisher/)

* [Creating a PyPI Project with a Trusted Publisher](../trusted-publishers/creating-a-project-through-oidc/)

* [Publishing with a Trusted Publisher](../trusted-publishers/using-a-publisher/)

* [Security Model and Considerations](../trusted-publishers/security-model/)

* [Troubleshooting](../trusted-publishers/troubleshooting/)

* [Internals and Technical Details](../trusted-publishers/internals/)

* Digital Attestations Digital Attestations

* [Introduction](../attestations/)

* [Producing attestations](../attestations/producing-attestations/)

* [Consuming attestations](../attestations/consuming-attestations/)

* [PyPI Publish Attestation (v1)](../attestations/publish/v1/)

* [Security Model and Considerations](../attestations/security-model/)

* Project Metadata [Project Metadata](./) Table of contents

* Project URLs

* Verified details

* Self-links

* Via Trusted Publishing

* Icons

* General URL

* Hosting Platforms

* Social Media Platforms

* Continuous Integration Services

* Python Ecosystem

* APIs and Datasets APIs and Datasets

- * [Introduction]([../api/](#))
- * [Index API]([../api/index-api/](#))
- * [JSON API]([../api/json/](#))
- * [Upload API]([../api/upload/](#))
- * [Integrity API]([../api/integrity/](#))
- * [Stats API]([../api/stats/](#))
- * [BigQuery Datasets]([../api/bigquery/](#))
- * [RSS Feeds]([../api/feeds/](#))
- * [Secret reporting API]([../api/secrets/](#))

Table of contents

- * Project URLs
 - * Verified details
 - * Self-links
 - * Via Trusted Publishing
 - * Icons
 - * General URL
 - * Hosting Platforms
 - * Social Media Platforms
 - * Continuous Integration Services
 - * Python Ecosystem

[](https://github.com/pypi/warehouse/blob/main/docs/user/project_metadata.md)

"Edit this page")

Project Metadata

Python packages can include additional metadata to provide more information about the project. This document outlines the specific behaviors implemented by PyPI to display project metadata and other details. The comprehensive list of metadata fields is available in the [Python Packaging User Guide](https://packaging.python.org/en/latest/specifications/core-metadata/#core-metadata-specifications).


Project URLs

Packages owners can specify various URLs related to their project using the `[[project.urls]` table[(https://packaging.python.org/en/latest/specifications/pyproject-toml/#urls) in the package's `pyproject.toml`.

PyPI renders these URLs on the project page and splits them into `verified` and `unverified` subgroups. They are also available using the [JSON API](../api/json/).

Verified details

![Verified details](../assets/verified_details.png)

PyPI currently supports several ways of verifying project URLs. When a URL is verified, PyPI highlights it using a green checkmark (.

Warning

A URL being verified only attests that the URL is under control of the PyPI package owner at the time of verification, and does not imply any additional safety about that URL or any other relationship to the project in question.

URL verification occurs when release files are uploaded and is not repeated afterwards. This means the websites that verified URLs point to can change, and the URL will still show up as verified. The verified status only reflects control of the URL ****at the time of file upload**** , not at any later point.

The following subsections specify the different types of URLs that can be verified.

Self-links

PyPI considers any URL pointing to that project on PyPI as verified. For example, the project page for `pip` will mark all of the following as verified:

- * `https://pypi.org/project/pip/`
- * `https://pypi.org/p/pip`
- * `https://pypi.python.org/project/pip`
- * `https://pypi.python.org/p/pip`
- * `https://python.org/pypi/pip`

Via Trusted Publishing

[Trusted Publishing](../trusted-publishers/) allows PyPI to attest that the publishing workflow for a package is coming from a verified source.

The URLs that can be verified depend on the Trusted Publisher used:

* [GitHub Actions](../trusted-publishers/creating-a-project-through-oidc/#github-actions): Packages uploaded using GHA from a repository will have the GitHub URLs for that repository verified. For example, for the `pypa/pip` repository, the following URLs will be verified:

- * `https://github.com/pypa/pip`
- * `https://github.com/pypa/pip/*` (all subpaths)
- * `https://github.com/pypa/pip.git`
- * `https://pypa.github.io/pip`
- * `https://pypa.github.io/pip/*` (all subpaths)

* [GitLab CI/CD](../trusted-publishers/creating-a-project-through-oidc/#gitlab-cicd): Packages uploaded using GitLab CI/CD from a repository will have the GitLab URLs for that repository verified. For example, for the `pypa/pip` repository, the following URLs will be verified:

- * `https://gitlab.com/pypa/pip`
- * `https://gitlab.com/pypa/pip/*` (all subpaths)
- * `https://gitlab.com/pypa/pip.git`


* [Google Cloud](../trusted-publishers/creating-a-project-through-oidc/#google-cloud): No Google-specific URLs are currently verified.

* [ActiveState](../trusted-publishers/creating-a-project-through-oidc/#activestate): Packages uploaded using ActiveState will have URLs linked to the project in ActiveState verified:

- * `https://platform.activestate.com/pypa/pip`
- * `https://platform.activestate.com/pypa/pip/*` (all subpaths)

Icons

![Unverified details](../assets/unverified_details.png)

While the labels or URLs can be arbitrary, PyPI recognizes the ones from the lists below and changes the default icon from  to a customized one.

General URL

To display a custom icon, an entry must match one of the pattern. The recognition patterns are case-insensitive. Items marked with an asterisk (*) indicate a prefix. It means that any name starting with the specified pattern will be recognized.

Name	Icon	Description	Aliases
------	------	-------------	---------

---	---	---	---
-----	-----	-----	-----

Homepage		For the project homepage	
----------	--	--------------------------	--

Download		A download link	
----------	--	-----------------	--

Changelog		Changelog information	Change log, Changes, Release notes, News, What's new, History
-----------	--	-----------------------	---

Documentation*		Project documentation	Docs* , a URL pointing to [Read the Docs](https://about.readthedocs.com/) domains or a URL starting with `docs.` or `documentation.`
----------------	--	-----------------------	--

Bug*		Bug/Issue report location	Issue*, Tracker*, Report*
------	--	---------------------------	---------------------------

Funding*		Sponsoring information	Sponsor*, Donation*, Donate*
----------	--	------------------------	------------------------------

Hosting Platforms

An entry URL must point to a domain below to display a custom icon. Custom subdomains are also matched. For instance, if `domain.com` is listed, a URL ending in `.domain.com` will also match.

Service	Icon	Domain
---------	------	--------

---	---	---
-----	-----	-----

Bitbucket		`bitbucket.org`
-----------	--	-----------------

GitHub		`github.com`, `github.io`
--------	--	---------------------------

GitLab		`gitlab.com`
--------	--	--------------

Google		`google.com`
--------	--	--------------

Social Media Platforms

To display a custom icon, an entry must either :

- * have its name match the case-insensitive pattern listed
- * a URL pointing to a listed domain (custom subdomains are supported)

Platform	Icon	Name	Domain
----------	------	------	--------

---	---	---	---
-----	-----	-----	-----

Discord			`discord.com`, `discordapp.com`, `discord.gg`
---------	--	--	---

Gitter			`gitter.im`
--------	--	--	-------------

Mastodon			Mastodon
----------	--	--	----------

Reddit			`reddit.com`
--------	--	--	--------------

Slack			Slack* `slack.com`
-------	--	--	----------------------

Youtube			`youtube.com`, `youtu.be`
---------	--	--	---------------------------

Twitter			`twitter.com`, `x.com`
---------	--	--	------------------------

Continuous Integration Services

To display a custom icon (), an entry URL must point to one of the service provider domains listed below. Custom subdomains are supported.

Service	Domain
---------	--------

---	---
-----	-----

AppVeyor	`ci.appveyor.com`
----------	-------------------

CircleCI	`circleci.com`
----------	----------------

Codecov	`codecov.io`
---------	--------------

Coveralls	`coveralls.io`
-----------	----------------

Travis CI	`travis-ci.com`, `travis-ci.org`
-----------	----------------------------------

Python Ecosystem

To display a custom icon, an entry URL must point to one of the domain listed below.

Name	Icon	Domain
------	------	--------

---	---	---
-----	-----	-----

PyPI		`cheeseshop.python.org`, `pypi.io`, `pypi.org`, `pypi.python.org`
------	--	---

Python		`python.org`, `*.python.org`
--------	--	------------------------------

Made with [Material for MkDocs](<https://squidfunk.github.io/mkdocs-material/>)

[](<https://github.com/pypi> "github.com") [](<https://twitter.com/pypi>
"twitter.com")