Skip to content

[![logo](../../assets/logo-letter.svg)](../.. "uv")

uv

Building and publishing a package

Initializing search

[uv](https://github.com/astral-sh/uv "Go to repository")

[![logo](../../assets/logo-letter.svg)](../.. "uv") uv

[uv](https://github.com/astral-sh/uv "Go to repository")

  * [Introduction](../..)
  * [Getting started](../../getting-started/)

Getting started

    * [Installation](../../getting-started/installation/)
    * [First steps](../../getting-started/first-steps/)
    * [Features](../../getting-started/features/)
    * [Getting help](../../getting-started/help/)
  * [Guides](../)

Guides

* [Installing Python](../install-python/)
* [Running scripts](../scripts/)
* [Using tools](../tools/)
* [Working on projects](../projects/)
* Publishing packages [Publishing packages](./) Table of contents
  * Preparing your project for packaging
  * Building your package
  * Publishing your package
  * Installing your package
  * Next steps
* [Integrations](../integration/)

Integrations

  * [Docker](../integration/docker/)
  * [Jupyter](../integration/jupyter/)
  * [GitHub Actions](../integration/github/)
  * [GitLab CI/CD](../integration/gitlab/)
  * [Pre-commit](../integration/pre-commit/)
  * [PyTorch](../integration/pytorch/)
  * [FastAPI](../integration/fastapi/)
  * [Alternative indexes](../integration/alternative-indexes/)
  * [Dependency bots](../integration/dependency-bots/)
  * [AWS Lambda](../integration/aws-lambda/)
* [Concepts](../../concepts/)

Concepts

* [Projects](../../concepts/projects/)

Projects

* [Structure and files](../../concepts/projects/layout/)
* [Creating projects](../../concepts/projects/init/)
* [Managing dependencies](../../concepts/projects/dependencies/)
* [Running commands](../../concepts/projects/run/)
* [Locking and syncing](../../concepts/projects/sync/)
* [Configuring projects](../../concepts/projects/config/)
* [Building distributions](../../concepts/projects/build/)
* [Using workspaces](../../concepts/projects/workspaces/)

* [Tools](../../concepts/tools/)
* [Python versions](../../concepts/python-versions/)
* [Resolution](../../concepts/resolution/)
* [Caching](../../concepts/cache/)

* [Configuration](../../configuration/)

Configuration

* [Configuration files](../../configuration/files/)
* [Environment variables](../../configuration/environment/)
* [Authentication](../../configuration/authentication/)
* [Package indexes](../../configuration/indexes/)

    * [Installer](../../configuration/installer/)
  * [The pip interface](../../pip/)

The pip interface

    * [Using environments](../../pip/environments/)
    * [Managing packages](../../pip/packages/)
    * [Inspecting packages](../../pip/inspection/)
    * [Declaring dependencies](../../pip/dependencies/)
    * [Locking environments](../../pip/compile/)
    * [Compatibility with pip](../../pip/compatibility/)
  * [Reference](../../reference/)

Reference

    * [Commands](../../reference/cli/)
    * [Settings](../../reference/settings/)
    * [Troubleshooting](../../reference/troubleshooting/)

Troubleshooting

      * [Build failures](../../reference/troubleshooting/build-failures/)
      * [Reproducible examples](../../reference/troubleshooting/reproducible-examples/)
    * [Resolver](../../reference/resolver-internals/)
    * [Benchmarks](../../reference/benchmarks/)
    * [Policies](../../reference/policies/)

Policies

* [Versioning](../../reference/policies/versioning/)
* [Platform support](../../reference/policies/platforms/)
* [License](../../reference/policies/license/)

Table of contents

Building and publishing a package

uv supports building Python packages into source and binary distributions via `uv build` and uploading them to a registry with `uv publish`.

Preparing your project for packaging

Before attempting to publish your project, you'll want to make sure it's ready to be packaged for distribution.

If your project does not include a `[build-system]` definition in the `pyproject.toml`, uv will not build it by default. This means that your project may not be ready for distribution. Read more about the effect of declaring a build system in the [project concept](../../concepts/projects/config/#build-systems) documentation.

Note

If you have internal packages that you do not want to be published, you can mark them as private:

```
[project]
classifiers = ["Private :: Do Not Upload"]
```

This setting makes PyPI reject your uploaded package from publishing. It does not affect security or privacy settings on alternative registries.

We also recommend only generating per-project tokens: Without a PyPI token matching the project, it can't be accidentally published.

Building your package

Build your package with `uv build`:

```
$ uv build
```

By default, `uv build` will build the project in the current directory, and
place the built artifacts in a `dist/` subdirectory.

Alternatively, `uv build <SRC>` will build the package in the specified
directory, while `uv build --package <PACKAGE>` will build the specified
package within the current workspace.

Info

By default, `uv build` respects `tool.uv.sources` when resolving build
dependencies from the `build-system.requires` section of the `pyproject.toml`.
When publishing a package, we recommend running `uv build --no-sources` to
ensure that the package builds correctly when `tool.uv.sources` is disabled,
as is the case when using other build tools, like
[`pypa/build`](https://github.com/pypa/build).

Publishing your package

Publish your package with `uv publish`:

```
$ uv publish
```

Set a PyPI token with `--token` or `UV_PUBLISH_TOKEN`, or set a username with `--username` or `UV_PUBLISH_USERNAME` and password with `--password` or `UV_PUBLISH_PASSWORD`. For publishing to PyPI from GitHub Actions, you don't need to set any credentials. Instead, [add a trusted publisher to the PyPI project](https://docs.pypi.org/trusted-publishers/adding-a-publisher/).

Note

PyPI does not support publishing with username and password anymore, instead you need to generate a token. Using a token is equivalent to setting `--username __token__` and using the token as password.

If you're using a custom index through `[[tool.uv.index]]`, add `publish-url` and use `uv publish --index <name>`. For example:

```
[[tool.uv.index]]
name = "testpypi"
url = "https://test.pypi.org/simple/"
publish-url = "https://test.pypi.org/legacy/"
```

Note

When using `uv publish --index <name>`, the `pyproject.toml` must be present, i.e. you need to have a checkout step in a publish CI job.

Even though `uv publish` retries failed uploads, it can happen that publishing fails in the middle, with some files uploaded and some files still missing. With PyPI, you can retry the exact same command, existing identical files will be ignored. With other registries, use `--check-url <index url>` with the index URL (not the publish URL) the packages belong to. When using `--index`, the index URL is used as check URL. uv will skip uploading files that are identical to files in the registry, and it will also handle raced parallel uploads. Note that existing files need to match exactly with those previously uploaded to the registry, this avoids accidentally publishing source distribution and wheels with different contents for the same version.

Installing your package

Test that the package can be installed and imported with `uv run`:

```
$ uv run --with <PACKAGE> --no-project -- python -c "import <PACKAGE>"
```

The `--no-project` flag is used to avoid installing the package from your local project directory.

Tip

If you have recently installed the package, you may need to include the `--refresh-package <PACKAGE>` option to avoid using a cached version of the package.

Next steps

To learn more about publishing packages, check out the [PyPA guides](https://packaging.python.org/en/latest/guides/section-build-and-publish/) on building and publishing.

Or, read on for [guides](../integration/) on integrating uv with other software.

January 15, 2025