Skip to content

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

uv

Building distributions

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")

Guides

* [Installing Python](../../../guides/install-python/)

* [Running scripts](../../../guides/scripts/)

* [Using tools](../../../guides/tools/)

* [Working on projects](../../../guides/projects/)

* [Publishing packages](../../../guides/package/)

* [Integrations](../../../guides/integration/)

Integrations

* [Docker](../../../guides/integration/docker/)

* [Jupyter](../../../guides/integration/jupyter/)

* [GitHub Actions](../../../guides/integration/github/)

* [GitLab CI/CD](../../../guides/integration/gitlab/)

* [Pre-commit](../../../guides/integration/pre-commit/)

* [PyTorch](../../../guides/integration/pytorch/)

* [FastAPI](../../../guides/integration/fastapi/)

* [Alternative indexes](../../../guides/integration/alternative-indexes/)

* [Dependency bots](../../../guides/integration/dependency-bots/)

* [AWS Lambda](../../../guides/integration/aws-lambda/)

* [Concepts](../../)

Concepts

* [Projects](../)

Projects

Configuration

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/)

Table of contents

Building distributions

To distribute your project to others (e.g., to upload it to an index like
PyPI), you'll need to build it into a distributable format.

Python projects are typically distributed as both source distributions
(sdists) and binary distributions (wheels). The former is typically a
`.tar.gz` or `.zip` file containing the project's source code along with some
additional metadata, while the latter is a `.whl` file containing pre-built
artifacts that can be installed directly.

Important

When using `uv build`, uv acts as a [build
frontend](https://peps.python.org/pep-0517/#terminology-and-goals) and only

determines the Python version to use and invokes the build backend. The details of the builds, such as the included files and the distribution filenames, are determined by the build backend, as defined in [`[build-system]`](../config/#build-systems). Information about build configuration can be found in the respective tool's documentation.

Using `uv build`

`uv build` can be used to build both source distributions and binary distributions for your project. By default, `uv build` will build the project in the current directory, and place the built artifacts in a `dist/` subdirectory:

```
$ uv build
$ ls dist/
example-0.1.0-py3-none-any.whl
example-0.1.0.tar.gz
```

You can build the project in a different directory by providing a path to `uv build`, e.g., `uv build path/to/project`.

`uv build` will first build a source distribution, and then build a binary distribution (wheel) from that source distribution.

You can limit `uv build` to building a source distribution with `uv build --sdist`, a binary distribution with `uv build --wheel`, or build both distributions from source with `uv build --sdist --wheel`.

Build constraints

`uv build` accepts `--build-constraint`, which can be used to constrain the versions of any build requirements during the build process. When coupled with `--require-hashes`, uv will enforce that the requirement used to build the project match specific, known hashes, for reproducibility.

For example, given the following `constraints.txt`:

setuptools==68.2.2 --hash=sha256:b454a35605876da60632df1a60f736524eb73cc47bbc9f3f1ef1b644de74fd2a

Running the following would build the project with the specified version of `setuptools`, and verify that the downloaded `setuptools` distribution matches the specified hash:

```
$ uv build --build-constraint constraints.txt --require-hashes
```

January 30, 2025

Made with [Material for MkDocs Insiders](https://squidfunk.github.io/mkdocs-

material/)

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

](https://discord.com/invite/astral-sh "discord.com") [

](https://pypi.org/project/uv/ "pypi.org") [ ](https://x.com/astral_sh

"x.com")