Skip to content

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

uv

Configuration files

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/)

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 files [Configuration files](./) Table of contents
  * Settings
  * .env
  * Configuring the pip interface
* [Environment variables](../environment/)
* [Authentication](../authentication/)
* [Package indexes](../indexes/)
* [Installer](../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

Configuration files

uv supports persistent configuration files at both the project- and user-level.

Specifically, uv will search for a `pyproject.toml` or `uv.toml` file in the current directory, or in the nearest parent directory.

Note

For `tool` commands, which operate at the user level, local configuration files will be ignored. Instead, uv will exclusively read from user-level configuration (e.g., `~/.config/uv/uv.toml`) and system-level configuration (e.g., `/etc/uv/uv.toml`).

In workspaces, uv will begin its search at the workspace root, ignoring any configuration defined in workspace members. Since the workspace is locked as a single unit, configuration is shared across all members.

If a `pyproject.toml` file is found, uv will read configuration from the `[tool.uv]` table. For example, to set a persistent index URL, add the following to a `pyproject.toml`:

pyproject.toml

```
[[tool.uv.index]]
url = "https://test.pypi.org/simple"
default = true
```

(If there is no such table, the `pyproject.toml` file will be ignored, and uv will continue searching in the directory hierarchy.)

uv will also search for `uv.toml` files, which follow an identical structure, but omit the `[tool.uv]` prefix. For example:

uv.toml

```
[[index]]
url = "https://test.pypi.org/simple"
default = true
```

Note

`uv.toml` files take precedence over `pyproject.toml` files, so if both `uv.toml` and `pyproject.toml` files are present in a directory, configuration will be read from `uv.toml`, and `[tool.uv]` section in the accompanying `pyproject.toml` will be ignored.

uv will also discover user-level configuration at `~/.config/uv/uv.toml` (or `$XDG_CONFIG_HOME/uv/uv.toml`) on macOS and Linux, or `%APPDATA%\uv\uv.toml` on Windows; and system-level configuration at `/etc/uv/uv.toml` (or `$XDG_CONFIG_DIRS/uv/uv.toml`) on macOS and Linux, or `%SYSTEMDRIVE%\ProgramData\uv\uv.toml` on Windows.

User-and system-level configuration must use the `uv.toml` format, rather than the `pyproject.toml` format, as a `pyproject.toml` is intended to define a Python _project_.

If project-, user-, and system-level configuration files are found, the settings will be merged, with project-level configuration taking precedence over the user-level configuration, and user-level configuration taking precedence over the system-level configuration. (If multiple system-level

configuration files are found, e.g., at both `/etc/uv/uv.toml` and `$XDG_CONFIG_DIRS/uv/uv.toml`, only the first-discovered file will be used, with XDG taking priority.)

For example, if a string, number, or boolean is present in both the project- and user-level configuration tables, the project-level value will be used, and the user-level value will be ignored. If an array is present in both tables, the arrays will be concatenated, with the project-level settings appearing earlier in the merged array.

Settings provided via environment variables take precedence over persistent configuration, and settings provided via the command line take precedence over both.

uv accepts a `--no-config` command-line argument which, when provided, disables the discovery of any persistent configuration.

uv also accepts a `--config-file` command-line argument, which accepts a path to a `uv.toml` to use as the configuration file. When provided, this file will be used in place of _any_ discovered configuration files (e.g., user-level configuration will be ignored).

Settings

See the [settings reference](../../reference/settings/) for an enumeration of the available settings.

`.env`

`uv run` can load environment variables from dotenv files (e.g., `.env`, `.env.local`, `.env.development`), powered by the [`dotenvy`](https://github.com/allan2/dotenvy) crate.

To load a `.env` file from a dedicated location, set the `UV_ENV_FILE` environment variable, or pass the `--env-file` flag to `uv run`.

For example, to load environment variables from a `.env` file in the current working directory:

```
$ echo "MY_VAR='Hello, world!'" > .env
$ uv run --env-file .env -- python -c 'import os; print(os.getenv("MY_VAR"))'
Hello, world!
```

The `--env-file` flag can be provided multiple times, with subsequent files overriding values defined in previous files. To provide multiple files via the `UV_ENV_FILE` environment variable, separate the paths with a space (e.g., `UV_ENV_FILE="/path/to/file1 /path/to/file2"`).

To disable dotenv loading (e.g., to override `UV_ENV_FILE` or the `--env-file` command-line argument), set the `UV_NO_ENV_FILE` environment variable to `1`, or pass the`--no-env-file` flag to `uv run`.

If the same variable is defined in the environment and in a `.env` file, the value from the environment will take precedence.

Configuring the pip interface

A dedicated [`[tool.uv.pip]`](../../reference/settings/#pip) section is provided for configuring _just_ the `uv pip` command line interface. Settings in this section will not apply to `uv` commands outside the `uv pip` namespace. However, many of the settings in this section have corollaries in the top-level namespace which _do_ apply to the `uv pip` interface unless they are overridden by a value in the `uv.pip` section.

The `uv.pip` settings are designed to adhere closely to pip's interface and are declared separately to retain compatibility while allowing the global settings to use alternate designs (e.g., `--no-build`).

As an example, setting the `index-url` under `[tool.uv.pip]`, as in the following `pyproject.toml`, would only affect the `uv pip` subcommands (e.g., `uv pip install`, but not `uv sync`, `uv lock`, or `uv run`):

pyproject.toml

```
[tool.uv.pip]
index-url = "https://test.pypi.org/simple"
```

November 12, 2024

Back to top [Previous Index](../) [Next Environment variables](../environment/)

[ ](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")