Skip to content

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

uv

Using uv in GitHub Actions

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](../../install-python/)
* [Running scripts](../../scripts/)
* [Using tools](../../tools/)
* [Working on projects](../../projects/)
* [Publishing packages](../../package/)
* [Integrations](../)

Integrations

* [Docker](../docker/)
* [Jupyter](../jupyter/)
* GitHub Actions [GitHub Actions](./) Table of contents
  * Installation
  * Setting up Python
  * Multiple Python versions
  * Syncing and running
  * Caching
  * Using uv pip
* [GitLab CI/CD](../gitlab/)
* [Pre-commit](../pre-commit/)
* [PyTorch](../pytorch/)
* [FastAPI](../fastapi/)
* [Alternative indexes](../alternative-indexes/)
* [Dependency bots](../dependency-bots/)
* [AWS Lambda](../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

Using uv in GitHub Actions

Installation

For use with GitHub Actions, we recommend the official [`astral-sh/setup-uv`](https://github.com/astral-sh/setup-uv) action, which installs uv, adds it to PATH, (optionally) persists the cache, and more, with support for all uv-

supported platforms.

To install the latest version of uv:

example.yml

```
name: Example

jobs:
  uv-example:
    name: python
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: Install uv
        uses: astral-sh/setup-uv@v5
```

It is considered best practice to pin to a specific uv version, e.g., with:

example.yml

```
    name: Example

    jobs:
      uv-example:
        name: python
        runs-on: ubuntu-latest

        steps:
          - uses: actions/checkout@v4

          - name: Install uv
            uses: astral-sh/setup-uv@v5
            with:
              # Install a specific version of uv.
              version: "0.5.29"
```

Setting up Python

Python can be installed with the `python install` command:

example.yml

```
    name: Example
```

```
jobs:
  uv-example:
    name: python
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: Install uv
        uses: astral-sh/setup-uv@v5

      - name: Set up Python
        run: uv python install
```

This will respect the Python version pinned in the project.

Alternatively, the official GitHub `setup-python` action can be used. This can be faster, because GitHub caches the Python versions alongside the runner.

Set the [`python-version-file`](https://github.com/actions/setup-python/blob/main/docs/advanced-usage.md#using-the-python-version-file-input) option to use the pinned version for the project:

example.yml

```
name: Example

jobs:
  uv-example:
    name: python
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: Install uv
        uses: astral-sh/setup-uv@v5

      - name: "Set up Python"
        uses: actions/setup-python@v5
        with:
          python-version-file: ".python-version"
```

Or, specify the `pyproject.toml` file to ignore the pin and use the latest version compatible with the project's `requires-python` constraint:

example.yml

```
name: Example

jobs:
  uv-example:
    name: python
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: Install uv
        uses: astral-sh/setup-uv@v5

      - name: "Set up Python"
        uses: actions/setup-python@v5
        with:
          python-version-file: "pyproject.toml"
```

Multiple Python versions

When using a matrix to test multiple Python versions, set the Python version using `astral-sh/setup-uv`, which will override the Python version specification in the `pyproject.toml` or `.python-version` files:

example.yml

```
jobs:
  build:
    name: continuous-integration
    runs-on: ubuntu-latest
    strategy:
      matrix:
        python-version:
          - "3.10"
          - "3.11"
          - "3.12"

    steps:
      - uses: actions/checkout@v4

      - name: Install uv and set the python version
        uses: astral-sh/setup-uv@v5
        with:
          python-version: ${{ matrix.python-version }}
```

If not using the `setup-uv` action, you can set the `UV_PYTHON` environment variable:

example.yml

```
    jobs:
      build:
        name: continuous-integration
        runs-on: ubuntu-latest
        strategy:
          matrix:
            python-version:
              - "3.10"
              - "3.11"
              - "3.12"
        env:
          UV_PYTHON: ${{ matrix.python-version }}
        steps:
          - uses: actions/checkout@v4
```

Syncing and running

Once uv and Python are installed, the project can be installed with `uv sync`

and commands can be run in the environment with `uv run`:

example.yml

```
    name: Example

    jobs:
      uv-example:
        name: python
        runs-on: ubuntu-latest

        steps:
          - uses: actions/checkout@v4

          - name: Install uv
            uses: astral-sh/setup-uv@v5

          - name: Install the project
            run: uv sync --all-extras --dev

          - name: Run tests
            # For example, using `pytest`
            run: uv run pytest tests
```

Tip

The [`UV_PROJECT_ENVIRONMENT`
setting](../../../concepts/projects/config/#project-environment-path) can be
used to install to the system Python environment instead of creating a virtual

environment.

Caching

It may improve CI times to store uv's cache across workflow runs.

The [`astral-sh/setup-uv`](https://github.com/astral-sh/setup-uv) has built-in support for persisting the cache:

example.yml

```
    - name: Enable caching
      uses: astral-sh/setup-uv@v5
      with:
        enable-cache: true
```

You can configure the action to use a custom cache directory on the runner:

example.yml

```
    - name: Define a custom uv cache path
      uses: astral-sh/setup-uv@v5
```

```
      with:
        enable-cache: true
        cache-local-path: "/path/to/cache"
```

Or invalidate it when the lockfile changes:

example.yml

```
    - name: Define a cache dependency glob
      uses: astral-sh/setup-uv@v5
      with:
        enable-cache: true
        cache-dependency-glob: "uv.lock"
```

Or when any requirements file changes:

example.yml

```
    - name: Define a cache dependency glob
      uses: astral-sh/setup-uv@v5
      with:
```

```
      enable-cache: true
      cache-dependency-glob: "requirements**.txt"
```

Note that `astral-sh/setup-uv` will automatically use a separate cache key for each host architecture and platform.

Alternatively, you can manage the cache manually with the `actions/cache` action:

example.yml

```
  jobs:
    install_job:
      env:
        # Configure a constant location for the uv cache
        UV_CACHE_DIR: /tmp/.uv-cache

      steps:
        # ... setup up Python and uv ...

        - name: Restore uv cache
          uses: actions/cache@v4
          with:
            path: /tmp/.uv-cache
```

```
      key: uv-${{ runner.os }}-${{ hashFiles('uv.lock') }}
      restore-keys: |
        uv-${{ runner.os }}-${{ hashFiles('uv.lock') }}
        uv-${{ runner.os }}

    # ... install packages, run tests, etc ...

    - name: Minimize uv cache
      run: uv cache prune --ci
```

The `uv cache prune --ci` command is used to reduce the size of the cache and is optimized for CI. Its effect on performance is dependent on the packages being installed.

Tip

If using `uv pip`, use `requirements.txt` instead of `uv.lock` in the cache key.

Note

When using non-ephemeral, self-hosted runners the default cache directory can grow unbounded. In this case, it may not be optimal to share the cache between jobs. Instead, move the cache inside the GitHub Workspace and remove it once the job finishes using a [Post Job Hook](https://docs.github.com/en/actions/hosting-your-own-runners/managing-

self-hosted-runners/running-scripts-before-or-after-a-job).

```
    install_job:
      env:
        # Configure a relative location for the uv cache
        UV_CACHE_DIR: ${{ github.workspace }}/.cache/uv
```

Using a post job hook requires setting the `ACTIONS_RUNNER_HOOK_JOB_STARTED`
environment variable on the self-hosted runner to the path of a cleanup script
such as the one shown below.

clean-uv-cache.sh

```
    #!/usr/bin/env sh
    uv cache clean
```

Using `uv pip`

If using the `uv pip` interface instead of the uv project interface, uv
requires a virtual environment by default. To allow installing packages into
the system environment, use the `--system` flag on all `uv` invocations or set

the `UV_SYSTEM_PYTHON` variable.

The `UV_SYSTEM_PYTHON` variable can be defined in at different scopes.

Opt-in for the entire workflow by defining it at the top level:

example.yml

```
env:
  UV_SYSTEM_PYTHON: 1

jobs: ...
```

Or, opt-in for a specific job in the workflow:

example.yml

```
jobs:
  install_job:
    env:
      UV_SYSTEM_PYTHON: 1
    ...
```

Or, opt-in for a specific step in a job:

example.yml

```
    steps:
      - name: Install requirements
        run: uv pip install -r requirements.txt
        env:
          UV_SYSTEM_PYTHON: 1
```

To opt-out again, the `--no-system` flag can be used in any uv invocation.

February 6, 2025

Back to top [Previous Jupyter](../jupyter/) [Next GitLab CI/CD](../gitlab/)

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")