Skip to content

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

uv

Using uv with PyTorch

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](../../package/)
* [Integrations](../)

Integrations

* [Docker](../docker/)
* [Jupyter](../jupyter/)
* [GitHub Actions](../github/)
* [GitLab CI/CD](../gitlab/)
* [Pre-commit](../pre-commit/)
* PyTorch [PyTorch](./) Table of contents
  * Installing PyTorch
  * Using a PyTorch index
  * Configuring accelerators with environment markers
  * Configuring accelerators with optional dependencies
  * The uv pip interface
* [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 with PyTorch

The [PyTorch](https://pytorch.org/) ecosystem is a popular choice for deep learning research and development. You can use uv to manage PyTorch projects and PyTorch dependencies across different Python versions and environments, even controlling for the choice of accelerator (e.g., CPU-only vs. CUDA).

Note

Some of the features outlined in this guide require uv version 0.5.3 or later.
If you're using an older version of uv, we recommend upgrading prior to
configuring PyTorch.

Installing PyTorch

From a packaging perspective, PyTorch has a few uncommon characteristics:

* Many PyTorch wheels are hosted on a dedicated index, rather than the Python Package Index (PyPI). As such, installing PyTorch often requires configuring a project to use the PyTorch index.
* PyTorch produces distinct builds for each accelerator (e.g., CPU-only, CUDA). Since there's no standardized mechanism for specifying these accelerators when publishing or installing, PyTorch encodes them in the local version specifier. As such, PyTorch versions will often look like `2.5.1+cpu`, `2.5.1+cu121`, etc.
* Builds for different accelerators are published to different indexes. For example, the `+cpu` builds are published on <https://download.pytorch.org/whl/cpu>, while the `+cu121` builds are published on <https://download.pytorch.org/whl/cu121>.

As such, the necessary packaging configuration will vary depending on both the
platforms you need to support and the accelerators you want to enable.

To start, consider the following (default) configuration, which would be
generated by running `uv init --python 3.12` followed by `uv add torch
torchvision`.

In this case, PyTorch would be installed from PyPI, which hosts CPU-only
wheels for Windows and macOS, and GPU-accelerated wheels on Linux (targeting

CUDA 12.4):

```
[project]
name = "project"
version = "0.1.0"
requires-python = ">=3.12"
dependencies = [
  "torch>=2.6.0",
  "torchvision>=0.21.0",
]
```

Supported Python versions

At time of writing, PyTorch does not yet publish wheels for Python 3.13; as such projects with `requires-python = ">=3.13"` may fail to resolve. See the [compatibility matrix](https://github.com/pytorch/pytorch/blob/main/RELEASE.md#release-compatibility-matrix).

This is a valid configuration for projects that want to use CPU builds on Windows and macOS, and CUDA-enabled builds on Linux. However, if you need to support different platforms or accelerators, you'll need to configure the project accordingly.

Using a PyTorch index

In some cases, you may want to use a specific PyTorch variant across all platforms. For example, you may want to use the CPU-only builds on Linux too.

In such cases, the first step is to add the relevant PyTorch index to your `pyproject.toml`:

CPU-onlyCUDA 11.8CUDA 12.1CUDA 12.4ROCm6Intel GPUs

```
[[tool.uv.index]]
name = "pytorch-cpu"
url = "https://download.pytorch.org/whl/cpu"
explicit = true
```

```
[[tool.uv.index]]
name = "pytorch-cu118"
url = "https://download.pytorch.org/whl/cu118"
explicit = true
```

```
[[tool.uv.index]]
```

```
name = "pytorch-cu121"
url = "https://download.pytorch.org/whl/cu121"
explicit = true

[[tool.uv.index]]
name = "pytorch-cu124"
url = "https://download.pytorch.org/whl/cu124"
explicit = true

[[tool.uv.index]]
name = "pytorch-rocm"
url = "https://download.pytorch.org/whl/rocm6.2"
explicit = true

[[tool.uv.index]]
name = "pytorch-xpu"
url = "https://download.pytorch.org/whl/xpu"
explicit = true
```

We recommend the use of `explicit = true` to ensure that the index is _only_

used for `torch`, `torchvision`, and other PyTorch-related packages, as opposed to generic dependencies like `jinja2`, which should continue to be sourced from the default index (PyPI).

Next, update the `pyproject.toml` to point `torch` and `torchvision` to the desired index:

CPU-onlyCUDA 11.8CUDA 12.1CUDA 12.4ROCm6Intel GPUs

```
[tool.uv.sources]
torch = [
  { index = "pytorch-cpu" },
]
torchvision = [
  { index = "pytorch-cpu" },
]
```

PyTorch doesn't publish CUDA builds for macOS. As such, we gate on `sys_platform` to instruct uv to use the PyTorch index on Linux and Windows, but fall back to PyPI on macOS:

```
[tool.uv.sources]
```

```
    torch = [
      { index = "pytorch-cu118", marker = "sys_platform == 'linux' or sys_platform == 'win32'" },
    ]
    torchvision = [
      { index = "pytorch-cu118", marker = "sys_platform == 'linux' or sys_platform == 'win32'" },
    ]
```

PyTorch doesn't publish CUDA builds for macOS. As such, we gate on
`sys_platform` to instruct uv to limit the PyTorch index to Linux and Windows,
falling back to PyPI on macOS:

```
    [tool.uv.sources]
    torch = [
      { index = "pytorch-cu121", marker = "sys_platform == 'linux' or sys_platform == 'win32'" },
    ]
    torchvision = [
      { index = "pytorch-cu121", marker = "sys_platform == 'linux' or sys_platform == 'win32'" },
    ]
```

PyTorch doesn't publish CUDA builds for macOS. As such, we gate on
`sys_platform` to instruct uv to limit the PyTorch index to Linux and Windows,
falling back to PyPI on macOS:

```
    [tool.uv.sources]
    torch = [
      { index = "pytorch-cu124", marker = "sys_platform == 'linux' or sys_platform == 'win32'" },
    ]
    torchvision = [
      { index = "pytorch-cu124", marker = "sys_platform == 'linux' or sys_platform == 'win32'" },
    ]
```

PyTorch doesn't publish ROCm6 builds for macOS or Windows. As such, we gate on
`sys_platform` to instruct uv to limit the PyTorch index to Linux, falling
back to PyPI on macOS and Windows:

```
    [tool.uv.sources]
    torch = [
      { index = "pytorch-rocm", marker = "sys_platform == 'linux'" },
    ]
    torchvision = [
      { index = "pytorch-rocm", marker = "sys_platform == 'linux'" },
    ]
```

PyTorch doesn't publish Intel GPU builds for macOS. As such, we gate on

`sys_platform` to instruct uv to limit the PyTorch index to Linux and Windows,

falling back to PyPI on macOS:

```
    [tool.uv.sources]
    torch = [
      { index = "pytorch-xpu", marker = "sys_platform == 'linux' or sys_platform == 'win32'" },
    ]
    torchvision = [
      { index = "pytorch-xpu", marker = "sys_platform == 'linux' or sys_platform == 'win32'" },
    ]
     # Intel GPU support relies on `pytorch-triton-xpu` on Linux, which should also be installed from the PyTorch index
    # (and included in `project.dependencies`).
    pytorch-triton-xpu = [
      { index = "pytorch-xpu", marker = "sys_platform == 'linux'" },
    ]
```

As a complete example, the following project would use PyTorch's CPU-only

builds on all platforms:

```
    [project]
    name = "project"
```

```
version = "0.1.0"
requires-python = ">=3.12.0"
dependencies = [
  "torch>=2.6.0",
  "torchvision>=0.21.0",
]

[tool.uv.sources]
torch = [
    { index = "pytorch-cpu" },
]
torchvision = [
    { index = "pytorch-cpu" },
]

[[tool.uv.index]]
name = "pytorch-cpu"
url = "https://download.pytorch.org/whl/cpu"
explicit = true
```

Configuring accelerators with environment markers

In some cases, you may want to use CPU-only builds in one environment (e.g., macOS and Windows), and CUDA-enabled builds in another (e.g., Linux).

With `tool.uv.sources`, you can use environment markers to specify the desired

index for each platform. For example, the following configuration would use PyTorch's CPU-only builds on Windows (and macOS, by way of falling back to PyPI), and CUDA-enabled builds on Linux:

```
[project]
name = "project"
version = "0.1.0"
requires-python = ">=3.12.0"
dependencies = [
  "torch>=2.6.0",
  "torchvision>=0.21.0",
]

[tool.uv.sources]
torch = [
  { index = "pytorch-cpu", marker = "sys_platform == 'win32'" },
  { index = "pytorch-cu124", marker = "sys_platform == 'linux'" },
]
torchvision = [
  { index = "pytorch-cpu", marker = "sys_platform == 'win32'" },
  { index = "pytorch-cu124", marker = "sys_platform == 'linux'" },
]

[[tool.uv.index]]
name = "pytorch-cpu"
```

```
url = "https://download.pytorch.org/whl/cpu"
explicit = true

[[tool.uv.index]]
name = "pytorch-cu124"
url = "https://download.pytorch.org/whl/cu124"
explicit = true
```

Similarly, the following configuration would use PyTorch's Intel GPU builds on Windows and Linux, and CPU-only builds on macOS (by way of falling back to PyPI):

```
[project]
name = "project"
version = "0.1.0"
requires-python = ">=3.12.0"
dependencies = [
  "torch>=2.6.0",
  "torchvision>=0.21.0",
  "pytorch-triton-xpu>=3.2.0 ; sys_platform == 'linux'",
]

[tool.uv.sources]
torch = [
```

```
    { index = "pytorch-xpu", marker = "sys_platform == 'win32' or sys_platform == 'linux'" },
]
torchvision = [
    { index = "pytorch-xpu", marker = "sys_platform == 'win32' or sys_platform == 'linux'" },
]
pytorch-triton-xpu = [
    { index = "pytorch-xpu", marker = "sys_platform == 'linux'" },
]

[[tool.uv.index]]
name = "pytorch-xpu"
url = "https://download.pytorch.org/whl/xpu"
explicit = true
```

Configuring accelerators with optional dependencies

In some cases, you may want to use CPU-only builds in some cases, but CUDA-enabled builds in others, with the choice toggled by a user-provided extra (e.g., `uv sync --extra cpu` vs. `uv sync --extra cu124`).

With `tool.uv.sources`, you can use extra markers to specify the desired index for each enabled extra. For example, the following configuration would use PyTorch's CPU-only for `uv sync --extra cpu` and CUDA-enabled builds for `uv sync --extra cu124`:

```
[project]
name = "project"
version = "0.1.0"
requires-python = ">=3.12.0"
dependencies = []

[project.optional-dependencies]
cpu = [
  "torch>=2.6.0",
  "torchvision>=0.21.0",
]
cu124 = [
  "torch>=2.6.0",
  "torchvision>=0.21.0",
]

[tool.uv]
conflicts = [
  [
    { extra = "cpu" },
    { extra = "cu124" },
  ],
]

[tool.uv.sources]
torch = [
```

```
    { index = "pytorch-cpu", extra = "cpu" },
    { index = "pytorch-cu124", extra = "cu124" },
  ]
  torchvision = [
    { index = "pytorch-cpu", extra = "cpu" },
    { index = "pytorch-cu124", extra = "cu124" },
  ]

  [[tool.uv.index]]
  name = "pytorch-cpu"
  url = "https://download.pytorch.org/whl/cpu"
  explicit = true

  [[tool.uv.index]]
  name = "pytorch-cu124"
  url = "https://download.pytorch.org/whl/cu124"
  explicit = true
```

Note

Since GPU-accelerated builds aren't available on macOS, the above configuration will fail to install on macOS when the `cu124` extra is enabled.

The `uv pip` interface

While the above examples are focused on uv's project interface (`uv lock`, `uv

sync`, `uv run`, etc.), PyTorch can also be installed via the `uv pip`
interface.

PyTorch itself offers a [dedicated interface](https://pytorch.org/get-
started/locally/) to determine the appropriate pip command to run for a given
target configuration. For example, you can install stable, CPU-only PyTorch on
Linux with:

```
$ pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu
```

To use the same workflow with uv, replace `pip3` with `uv pip`:

```
$ uv pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu
```

January 30, 2025

Back to top [Previous Pre-commit](../pre-commit/) [Next FastAPI
](../fastapi/)

[ ](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")