Guides

Using tools

Many Python packages provide applications that can be used as tools. uv has specialized support for easily invoking and installing tools.

Running tools

The `uvx` command invokes a tool without installing it.

For example, to run `ruff`:

```
$ uvx ruff
```

Note

This is exactly equivalent to:

```
$ uv tool run ruff
```

`uvx` is provided as an alias for convenience.

Arguments can be provided after the tool name:

```
$ uvx pycowsay hello from uv

  -------------
< hello from uv >
  -------------
   \   ^__^
    \  (oo)\_______
       (__)\       )\/\
           ||----w |
           ||     ||
```

Tools are installed into temporary, isolated environments when using `uvx`.

Note

If you are running a tool in a *project* and the tool requires that your project is installed, e.g., when using `pytest` or `mypy`, you'll want to use `uv run` instead of `uvx`. Otherwise, the tool will be run in a virtual environment that is isolated from your project.

If your project has a flat structure, e.g., instead of using a `src` directory for modules, the project itself does not need to be installed and `uvx` is fine. In this case, using `uv run` is only beneficial if you want to pin the version of the tool in the project's dependencies.

Commands with different package names

When `uvx ruff` is invoked, uv installs the `ruff` package which provides the `ruff` command. However, sometimes the package and command names differ.

The `--from` option can be used to invoke a command from a specific package, e.g., `http` which is provided by `httpie`:

```
$ uvx --from httpie http
```

Requesting specific versions

To run a tool at a specific version, use `command@<version>`:

```
$ uvx ruff@0.3.0 check
```

To run a tool at the latest version, use `command@latest`:

```
$ uvx ruff@latest check
```

The `--from` option can also be used to specify package versions, as above:

```
$ uvx --from 'ruff==0.3.0' ruff check
```

Or, to constrain to a range of versions:

```
$ uvx --from 'ruff>0.2.0,<0.3.0' ruff check
```

Note the `@` syntax cannot be used for anything other than an exact version.

Requesting extras

The `--from` option can be used to run a tool with extras:

```
$ uvx --from 'mypy[faster-cache,reports]' mypy --xml-report mypy_report
```

This can also be combined with version selection:

```
$ uvx --from 'mypy[faster-cache,reports]==1.13.0' mypy --xml-report
mypy_report
```

Requesting different sources

The `--from` option can also be used to install from alternative sources.

For example, to pull from git:

```
$ uvx --from git+https://github.com/httpie/cli httpie
```

You can also pull the latest commit from a specific named branch:

```
$ uvx --from git+https://github.com/httpie/cli@master httpie
```

Or pull a specific tag:

```
$ uvx --from git+https://github.com/httpie/cli@3.2.4 httpie
```

Or even a specific commit:

```
$ uvx --from git+https://github.com/httpie/cli@2843b87 httpie
```

Commands with plugins

Additional dependencies can be included, e.g., to include `mkdocs-material` when running `mkdocs`:

```
$ uvx --with mkdocs-material mkdocs --help
```

Installing tools

If a tool is used often, it is useful to install it to a persistent environment and add it to the `PATH` instead of invoking `uvx` repeatedly.

Tip

`uvx` is a convenient alias for `uv tool run`. All of the other commands for interacting with tools require the full `uv tool` prefix.

To install `ruff`:

```
$ uv tool install ruff
```

When a tool is installed, its executables are placed in a `bin` directory in the `PATH` which allows the tool to be run without uv. If it's not on the `PATH`, a warning will be displayed and `uv tool update-shell` can be used to add it to the `PATH`.

After installing `ruff`, it should be available:

```
$ ruff --version
```

Unlike `uv pip install`, installing a tool does not make its modules available in the current environment. For example, the following command will fail:

```
$ python -c "import ruff"
```

This isolation is important for reducing interactions and conflicts between dependencies of tools, scripts, and projects.

Unlike `uvx`, `uv tool install` operates on a *package* and will install all executables provided by the tool.

For example, the following will install the `http`, `https`, and `httpie` executables:

```
$ uv tool install httpie
```

Additionally, package versions can be included without `--from`:

```
$ uv tool install 'httpie>0.1.0'
```

And, similarly, for package sources:

```
$ uv tool install git+https://github.com/httpie/cli
```

As with `uvx`, installations can include additional packages:

```
$ uv tool install mkdocs --with mkdocs-material
```

Upgrading tools

To upgrade a tool, use `uv tool upgrade`:

```
$ uv tool upgrade ruff
```

Tool upgrades will respect the version constraints provided when installing the tool. For example, `uv tool install ruff >=0.3,<0.4` followed by `uv tool upgrade ruff` will upgrade Ruff to the latest version in the range `>=0.3,<0.4`.

To instead replace the version constraints, re-install the tool with `uv tool install`:

```
$ uv tool install ruff>=0.4
```

To instead upgrade all tools:

```
$ uv tool upgrade --all
```

Requesting Python versions

By default, uv will use your default Python interpreter (the first it finds) when running, installing, or upgrading tools. You can specify the Python interpreter to use with the `--python` option.

For example, to request a specific Python version when running a tool:

```
$ uvx --python 3.10 ruff
```

Or, when installing a tool:

```
$ uv tool install --python 3.10 ruff
```

Or, when upgrading a tool:

```
$ uv tool upgrade --python 3.10 ruff
```

For more details on requesting Python versions, see the Python version concept page.

Legacy Windows Scripts

Tools also support running legacy setuptools scripts. These scripts are available via `$(uv tool dir)\<tool-name>\Scripts` when installed.

Currently only legacy scripts with the `.ps1`, `.cmd`, and `.bat` extensions are supported.

For example, below is an example running a Command Prompt script.

```
$ uv tool run --from nuitka==2.6.7 nuitka.cmd --version
```

In addition, you don't need to specify the extension. `uvx` will automatically look for files ending in `.ps1`, `.cmd`, and `.bat` in that order of execution on your behalf.

```
$ uv tool run --from nuitka==2.6.7 nuitka --version
```

Next steps

To learn more about managing tools with uv, see the Tools concept page and the command reference.

Or, read on to learn how to work on projects.