What's new in pip 26.1 - lockfiles and dependency cooldowns!

pip 26.1 adds support for dependency cooldowns, experimental support for reading/installing from standard lockfiles (pylock.toml), fixes several long-standing limitations of the 2020 resolver, and drops support for Python 3.9.

April 26, 2026 · Updated on April 27, 2026 · 8 minutes

On April 26, 2026, we, the pip team, released pip 26.1.

As always, please consult the changelog to learn about all of the changes contained in this release.

Python 3.9 is no longer supported

pip 26.1 now targets Python 3.10 and higher.

Python 3.9 entered End-of-Life (EOL) status six months ago in October 2025. At this point in the year, many of pip's vendored dependencies have stopped supporting Python 3.9, so it made sense for pip to drop support as well.

New features ✨

Experimental: installing from pylock files

After a year[since the acceptance of PEP 751 which standardized `pylock.toml` lock files, pip 26.1 gains **experimental** (see below for more details on "experimental") support for reading and

installing from such lockfiles. 🎉

The `-r <file>`, `--requirement <file>` option now accepts `pylock.toml` files. You may pass a local lockfile or a remote lockfile as long it is named as expected: `pylock.toml` or `pylock.<name>.toml`.

```
$ pip install -r pylock.toml
WARNING: Using pylock.toml as a requirements source is an experimental feature. I
Collecting six==1.17.0 (from pylock.toml)
  Downloading six-1.17.0-py2.py3-none-any.whl (11 kB)
Collecting urllib3==2.6.3 (from pylock.toml)
  Downloading urllib3-2.6.3-py3-none-any.whl (131 kB)
Installing collected packages: urllib3, six
Successfully installed six-1.17.0 urllib3-2.6.3
```

`pip install`, `pip download`, and `pip wheel` all support reading lockfiles since they accept `-r`.

```
$ pip wheel -r pylock.toml -w ./wheelhouse
WARNING: Using pylock.toml as a requirements source is an experimental feature. I
Collecting six==1.17.0 (from pylock.toml)
  Using cached six-1.17.0-py2.py3-none-any.whl (11 kB)
Collecting urllib3==2.6.3 (from pylock.toml)
  Using cached urllib3-2.6.3-py3-none-any.whl (131 kB)
Saved ./wheelhouse/six-1.17.0-py2.py3-none-any.whl
Saved ./wheelhouse/urllib3-2.6.3-py3-none-any.whl
```

```
$ pip install wheelhouse/*.whl --no-deps --no-index
Processing ./wheelhouse/six-1.17.0-py2.py3-none-any.whl
Processing ./wheelhouse/urllib3-2.6.3-py3-none-any.whl
Installing collected packages: urllib3, six
Successfully installed six-1.17.0 urllib3-2.6.3
```

The intention of this feature is support "environment replication" from a lockfile. Ideally, `pip install -r pylock.toml` is executed within an *empty* environment, although this isn't required. While we are interested in hearing about additional use-cases, the pip team reserves to declare such use-cases as out of scope.

While you may mix `-r pylock.toml` with any other normal requirement or constraint specif file, as long as the total set of requested packages do not conflict, **this is strongly discouraged.**

This is a side-effect of the current implementation, and it is likely to change as the pip project stabilizes its lockfile support.

Limitations and gotchas

- Extras and dependency groups currently cannot be selected from a (multi-use) pylock file[1]
- VCS and local directory entries currently cannot be mixed with external requirements that are hash-locked
- The optional fallback to using `packages.index` to resolve broken URLs is unsupported
- By default, pip will let requirements from lock files pull in additional non-locked requirements. Pass `--no-deps` if this is undesirable.
- Archive file sizes are currently not validated
- Only the format control options `--only-binary` and `--no-binary` will influence locked requirements. All other package selection options (such as `--pre`, `--abi`, etc.) will be ignored

Please note that requirements sourced from a lockfile are treated specially. They function like pinned requirements where the used archive or path is pre-constrained.

If hashes are available for a locked requirement, pip will validate hashes as expected.

What's next?

As pip's lockfile support is experimental, `pip install -r pylock.toml` and `pip lock` should *not* be considered production ready. In the next few releases, the pip team will refine the lockfile UI/UX in response to real-world experience and feedback.[2][3]

Warning

We reserve the right to **modify, deprecate, or remove experimental features without notice**. pip's support for reading pylock files via `-r pylock.toml` is intended as a transitional mechanism until pip can adopt expanded and better UI/UX.

We will be also working towards adding a `pip sync` command which will function similar to `pip-sync` and `uv sync` although in a way that makes sense for pip's operating model. We intend for `pip sync` to be primary way in which pip interacts with lockfiles once available.

Thank you to Stéphane Bidoul who not only championed pip's implementation, but upstreamed pylock.toml support into the `packaging` library so the whole Python ecosystem can benefit.

We need your feedback!

In the meanwhile, if you can, please test out `-r pylock.toml`. If you have any feedback, let us know in the following issues!

- "What's next for `-r pylock.toml`?"
- "What's next for `pip lock`?"
- "First party `pip sync` command"

We'd especially appreciate comments on your use-case for lockfiles and how pip enables or does not enable your use-case.

Dependency cooldowns

`--uploaded-prior-to` now supports accepts a relative duration in the `PnD` format, where `n` is the number of days.[4] This is meant to enable dependency cooldowns, which aim to reduce the impact of compromised upstream packages.

```
$ pip install --uploaded-prior-to=P3D pip --force-reinstall  # exclude the recent
Collecting pip
  Downloading pip-26.0.1-py3-none-any.whl.metadata (4.7 kB)
Downloading pip-26.0.1-py3-none-any.whl (1.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 2.3 MB/s  0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 26.1.dev0
    Uninstalling pip-26.1.dev0:
      Successfully uninstalled pip-26.1.dev0
Successfully installed pip-26.0.1
```

A dependency cooldown involves configuring a set period of time before a newly released package is eligible for installation, providing time for package registries and third-party security firms to discover and package authors to recover from a compromise.

It's worth noting that while a cooldown can minimize the impact of supply chain attacks, it will also delay security fixes from reaching your environment. If you use this option, pair it with a vulnerability scanning tool such as Dependabot or pip-audit so that you are notified of security issues independently of your update schedule.

For more information, please consult William Woodruff's excellent post "We should all be using dependency cooldowns". Thank you also to Norbert Manthey for contributing this feature to pip!

Lifting several 2020 resolver limitations

The modern pip resolver ("the 2020 resolver") is miles better than the legacy resolver, capable of resolving complex dependency trees without producing an invalid environment. However, pip has kept its legacy resolver available as a deprecated feature via `--use-deprecated=legacy-resolver` to support use-cases that are unsupported by the 2020 resolver.

Starting in pip 26.1, the 2020 resolver now supports most of these previously unsupported use-cases:

- Fix a crash when installing a requirement with an extra that has a corresponding URL constraint
- Properly apply URL constraints to requirements with extras
- Allow unpinned requirements to use hashes from constraints. Constraints like `{name}=={version} --hash=...` feeds into hash verification for a corresponding requirement.

With a few remaining planned changes, pip will be in a position to fully remove the legacy resolver, which we would like to do sometime in 2027.

Security fixes

Avoid mistreating a .tar.gz archive as a zip file (CVE-2026-3219)

It's possible for a `.tar.gz` archive to look like a zip file, i.e. when given to the stdlib `zipfile.is_zipfile()` function, it will return true. On pip 26.0 and older, pip will ignore the `.tar.gz` contents and use the zip contents instead. This can be abused to easily obfuscate malicious code.

pip 26.1 updates the logic used to disambiguate tar and zip archives. pip will now use the following signals, in decreasing priority:

- Content type returned by the remote server
- File extension

- Python standard library magic detection functions **if only** they unambiguously agree

Thank you to Dmitrii Sutiagin for contributing a fix and to Caleb Brown for reporting this issue responsibly to the pip team.

Fix ACE caused by self check deferred imports (CVE-2026-6357)

pip will issue a notice if there is a newer version of pip available. For performance reasons this check occurs at the end of a command invocation, with the responsible code being imported when needed. If pip's own modules are overwritten during a `pip install`, this may result in arbitrary code execution.

pip 26.1 fixes this vulnerability by performing the self check eagerly before any command runs, deferring only the emitting of the notice to the very end.

Thank you to Damian Shaw for discovering and fixing this issue.

Upgrade to urllib3 2.x

Now that pip requires Python 3.10, pip 26.1 ships with urllib3 2.6.3 instead of 1.26.20. Unlike the 1.x series, 2.x is actively supported by the urllib3 project.

This upgrade fixes a number of CVEs associated with the urllib3 library:

- CVE-2026-21441
- CVE-2025-66471
- CVE-2025-50182

Please note that we consider these CVEs to be largely inconsequential for pip. The main benefit for end-users is that any downstream security scanners will stop complaining about pip's vendored copy of urllib3.

Deprecations & upcoming removals

Here's the current list of current deprecations with the release in which they are scheduled f removal. As always, any given removal may be **pushed to a future release as needed**.

`PIP_CONSTRAINT` for build dependencies

To be removed in pip 26.2

The `PIP_CONSTRAINT` envvar will eventually stop taking effect for build dependencies. We're making this change to accommodate the planned transition to installing build dependencies in-process. Affected users should use `--build-constraint` or `PIP_BUILD_CONSTRAINT`.

Acknowledgements

None so far. I did write this on a compressed timeline, however, so please forgive any typos.

---

1. `--group` will not do what you want. It will read from the local `pyproject.toml`. ↩
2. If you take a look at the linked issues, there many open design questions left to resolved. I have my own opinion and the other pip committers may have different opinions. We want to get the UI/UX right, so please forgive us as we cycle on pip's lockfile support. ↩
3. In general, we don't want pip to be an *innovator* when it comes to new features or new UX flows. We will aim to match pre-existing features from other package managers where feasible. ↩
4. We chose to only support the simple `PnD` ISO 8601 duration format to keep the parsing code as minimal as feasible. We did not want to vendor a human time parsing library or roll our own. ↩

Pip Release

OLDER »
What's new in pip 26.0 - prerelease and upload-time filtering!