Skip to content

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

uv

Reproducible examples

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/)

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

* [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

    * [Commands](../../cli/)
    * [Settings](../../settings/)
    * [Troubleshooting](../)

Troubleshooting

      * [Build failures](../build-failures/)
      * Reproducible examples [Reproducible examples](./) Table of contents
        * Why reproducible examples are important
        * How to write a reproducible example
        * Strategies for reproducible examples
          * Docker image
          * Script
          * Git repository
    * [Resolver](../../resolver-internals/)
    * [Benchmarks](../../benchmarks/)
    * [Policies](../../policies/)

Policies

* [Versioning](../../policies/versioning/)
* [Platform support](../../policies/platforms/)
* [License](../../policies/license/)

Table of contents

Reproducible examples

Why reproducible examples are important

A minimal reproducible example (MRE) is essential for fixing bugs. Without an example that can be used to reproduce the problem, a maintainer cannot debug it or test if it is fixed. If the example is not minimal, i.e., if it includes

lots of content which is not related to the issue, it can take a maintainer much longer to identify the root cause of the problem.

How to write a reproducible example

When writing a reproducible examaple, the goal is to provide all of the context necessary for someone else to reproduce your example. This includes:

* The platform you're using (e.g., the operating system and architecture)
* Any relevant system state (e.g.,)
* The version of uv
* The version of other relevant tools
* The relevant files (the `uv.lock`, `pyproject.toml`, etc.)
* The commands to run

To ensure your reproduction is minimal, remove as many dependencies, settings, and files as possible. Be sure to test your reproduction before sharing it. We recommend including verbose logs from your reproduction; they may differ on your machine in a critical way. Using a [Gist](https://gist.github.com) can be helpful for very long logs.

Below, we'll cover several specific strategies for creating and sharing reproducible examples.

Tip

There's a great guide to the basics of creating MREs on [Stack

Overflow](https://stackoverflow.com/help/minimal-reproducible-example).

Strategies for reproducible examples

Docker image

Writing a Docker image is often the best way to share a reproducible example because it is entirely self-contained. This means that the state from the reproducer's system does not affect the problem.

Note

Using a Docker image is only feasible if the issue is reproducible on Linux. When using macOS, it's prudent to ensure your image is not reproducible on Linux but some bugs _are_ specific to the operating system. While using Docker to run Windows containers is feasible, it's not commonplace. These sorts of bugs are expected to be reported as a script instead.

When writing a Docker MRE with uv, it's best to start with one of [uv's Docker images](../../../guides/integration/docker/#available-images). When doing so, be sure to pin to a specific version of uv.

```
FROM ghcr.io/astral-sh/uv:0.5.24-debian-slim
```

While Docker images are isolated from the system, the build will use your system's architecture by default. When sharing a reproduction, you can explicitly set the platform to ensure a reproducer gets the expected behavior. uv publishes images for `linux/amd64` (e.g., Intel or AMD) and `linux/arm64` (e.g., Apple M Series or ARM)

```
FROM --platform=linux/amd64 ghcr.io/astral-sh/uv:0.5.24-debian-slim
```

Docker images are best for reproducing issues that can be constructed with commands, e.g.:

```
FROM --platform=linux/amd64 ghcr.io/astral-sh/uv:0.5.24-debian-slim

RUN uv init /mre
WORKDIR /mre
RUN uv add pydantic
RUN uv sync
RUN uv run -v python -c "import pydantic"
```

However, you can also write files into the image inline:

```
FROM --platform=linux/amd64 ghcr.io/astral-sh/uv:0.5.24-debian-slim

COPY <<EOF /mre/pyproject.toml
[project]
name = "example"
version = "0.1.0"
description = "Add your description here"
readme = "README.md"
requires-python = ">=3.12"
dependencies = ["pydantic"]
EOF

WORKDIR /mre
RUN uv lock
```

If you need to write many files, it's better to create and publish a Git repository. You can combine these approaches and include a `Dockerfile` in the repository.

When sharing a Docker reproduction, it's helpful to include the build logs. You can see more output from the build steps by disabling caching and the fancy output:

```
docker build . --progress plain --no-cache
```

Script

When reporting platform-specific bugs that cannot be reproduced in a
container, it's best practice to include a script showing the commands that
can be used to reproduce the bug, e.g.:

```
uv init
uv add pydantic
uv sync
uv run -v python -c "import pydantic"
```

If your reproduction requires many files, use a Git repository to share them.

In addition to the script, include _verbose_ logs (i.e., with the `-v` flag)
of the failure and the complete error message.

Whenever a script relies on external state, be sure to share that information.
For example, if you wrote the script on Windows and it uses a Python version
that you installed with `choco` and runs on PowerShell 6.2, please include
that in the report.

Git repository

When sharing a Git repository reproduction, include a script that reproduces the problem or, even better, a Dockerfile. The first step of the script should be to clone the repository and checkout a specific commit:

```
$ git clone https://github.com/<user>/<project>.git
$ cd <project>
$ git checkout <commit>
$ <commands to produce error>
```

You can quickly create a new repository in the [GitHub UI](https://github.com/new) or with the `gh` CLI:

```
$ gh repo create uv-mre-1234 --clone
```

When using a Git repository for a reproduction, please remember to _minimize_ the contents by excluding files or settings that are not required to reproduce your problem.

January 27, 2025

Back to top [Previous Build failures](../build-failures/) [Next Resolver](../../resolver-internals/)

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")