Guide 9: Python environment for offline machines

Python, a virtual environment and pyflightstream on a machine with no internet, Windows and Linux

Written for pyflightstream 0.58.2.

Sources: the package's installation notes and the `pip` documentation [1, 2].

pyflightstream is the work of Geovana Neves: `https://github.com/nevesgeovana/pyflightstream`, concept DOI `10.5281/zenodo.21482924`.
Copyright (c) 2026 Geovana Neves. Licensed under CC BY 4.0
(`https://creativecommons.org/licenses/by/4.0/`). See LICENSE-AND-AUTHORSHIP.md.

Who reads this

- The person who installs pyflightstream on a machine that cannot reach the internet: a Windows workstation behind a closed network, a Linux workstation, or the shared file system of an HPC cluster.
- It happens once per machine and per version. Everybody else activates the environment with one line (Guide 2).
- The same recipe serves both systems: **download on a connected machine, carry one folder, install with the network forbidden**.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Contents

The idea
Python on the offline machine
The wheels
Installing
Running and recording

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 1

The idea

Two machines and one folder



- A **wheel** is a ready-built package. With wheels only, the offline machine needs no compiler and no network.
- The download names the **target's** Python version and platform, not the connected machine's.
- The install forbids the network, so it either installs from what was carried or fails naming what is missing. It never hangs.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

What the target needs to be

Python 3.12 or newer

- The package declares `requires-python = ">=3.12"`, following the scientific-python support schedule SPEC 0 [3].
- 3.9 and 3.10 lack `tomllib` and `enum.StrEnum`, which the package uses. 3.11 is below the declared floor: `pip` refuses it naming `Requires-Python`.
- Download for the same minor version the target runs: 312.

Target	Platform tags to download
Windows, 64 bit	`win_amd64`
Linux x86-64, glibc 2.28 or later	`manylinux_2_28_x86_64` *and* `manylinux2014_x86_64`

Linux: check the C library first

`ldd --version` on a node must print 2.28 or later (RHEL 8 and its rebuilds, and newer). Older systems cannot run the current `numpy` wheels [4].

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Why two Linux tags

- `--platform` tells `pip` which wheels the target accepts. It accepts **exactly the tags you list**, so list every one the target can run.
- The structural solver of the FSI extra, PyNiteFEA, needs `numpy >= 2.4`, whose Linux wheels are `manylinux_2_28` only: with `manylinux2014` alone the download fails.
- Other dependencies (`pydantic-core`) publish `manylinux2014` wheels only: with `manylinux_2_28` alone the download fails too.
- Both tags together resolve. A glibc 2.28 system runs both kinds.

Takeaway *If the download fails with ResolutionImpossible, a platform tag is missing; do not drop `--only-binary`.*

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 2

Python on the offline machine

Windows: one installer, no administrator

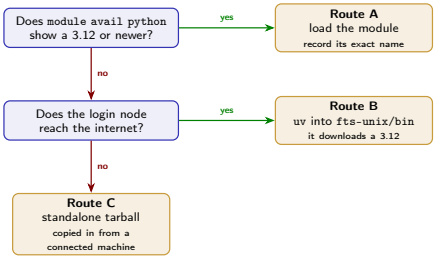
- On the connected machine, download the **Windows installer (64-bit)** of the latest Python 3.12 from `python.org` [5], and carry it with the wheels.
- On the offline machine, run it for the current user only: leave *Install for all users* off. No administrator is needed.
- Keep the `py` launcher on. Check:

```
py -3.12 --version
```

Corporate images

A machine may already carry a managed Python. Use it only if it is 3.12 or newer; otherwise install beside it, per user, and always call it as `py -3.12`.

Linux: three routes to a 3.12, no administrator



Route C always works. The interpreter is relocatable and writes nothing outside `fts-unix`; it is not compiling Python.

- A cluster's system Python is usually older than 3.12. Put a 3.12 **inside the shared install folder** (`<cluster-root>/fts-unix`); none of the routes touches the system Python.
- Route C always works on an offline machine: a relocatable CPython from the python-build-standalone project [6], copied in.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Linux, Route C: choosing the standalone archive

Choice	Take	Why not the other
version	`3.12`	the wheels are cp312; a 3.13 rejects them
CPU level	`x86_64`	`x86_64_v3` needs AVX2 and dies on an older node
contents	`install_only_stripped`	the same interpreter, about a third of the size
libc	`linux-gnu`	`musl` is for Alpine

Copy the archive into the shared folder, then on the Linux machine:

```
export FTS=<cluster-root>/fts-unix
cd $FTS
sha256sum cpython-3.12*-install_only_stripped.tar.gz   # compare with the release
tar -xzf  cpython-3.12*-install_only_stripped.tar.gz   # creates $FTS/python
$FTS/python/bin/python3.12 --version
```

The interpreter finds its standard library relative to its own binary and writes nothing outside `$FTS`. Nothing is compiled.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 3

The wheels

Download on the connected machine

For a Linux target (one command, written for the Windows command prompt, where ^ continues a line; use \ in a Linux shell and a backtick in PowerShell):

```
python -m pip download "pyflightstream[fsi]==0.58.2" -d pfs0410-linux ^
    --python-version 312 --only-binary=:all: ^
    --platform manylinux_2_28_x86_64 --platform manylinux2014_x86_64
```

For a Windows target:

```
python -m pip download "pyflightstream[fsi]==0.58.2" -d pfs0410-win ^
    --python-version 312 --only-binary=:all: --platform win_amd64
```

Option	What it prevents
`--python-version 312`	downloading for the connected machine's Python
`--platform ...`	downloading the connected machine's own wheels
`--only-binary=:all:`	downloading source, which would need a compiler

Copyright (c) 2026 Geovana Neves, CC BY 4.0

A development build, and the check before carrying

A development build (a version ending in `.dev<N>`) is not on the package index. Name its wheel file instead of the version; `pip` copies it and downloads its dependencies:

```
python -m pip download ^
  "pyflightstream-<version>.dev<N>-py3-none-any.whl[fsi]" ^
  -d pfs0410-win --python-version 312 ^
  --only-binary=:all: --platform win_amd64
```

The failure that gives no warning

Without `--platform` and `--python-version`, `pip` quietly downloads for *your* machine. The folder looks right and copies fine, and the install fails on the target after everything was moved.

Check

Every file must end in `.whl`; one `.tar.gz` means source got in. The platform files must carry the target's tag.

Part 4

Installing

Linux: the environment at its final path

```
$FTS/python/bin/python3.12 -m venv --copies $FTS/py-venv/pfs0410
source $FTS/py-venv/pfs0410/bin/activate
ls $FTS/pfs-wheels/pfs0410-linux | wc -l      # the folder pip will read
pip install --no-index --find-links $FTS/pfs-wheels/pfs0410-linux "pyflightstream[fsi]"
python -c "import pyflightstream; print(pyflightstream.__version__)"
pyfs-matrix --help
```

- `--no-index` forbids the internet; `--find-links` says where to look instead.
- `--copies` puts the interpreter inside the environment, so it survives anybody tidying up `python/` later.
- The version line must print the version you downloaded, and `pyfs-matrix --help` must answer.

Windows: the same, with the launcher

```
cd C:/<work-folder>/fts-workspace
py -3.12 -m venv .venv
.venv\Scripts\activate
pip install --no-index --find-links <folder>\pfs0410-win "pyflightstream[fsi]"
python -c "import pyflightstream; print(pyflightstream.__version__)"
pyfs-matrix --help
```

- `<folder>` is where the carried wheels are; any local or network folder works.
- `deactivate` leaves the environment.
- A development build installs the same way: its wheel is in the folder.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Rules of a shared environment

Build it at its final path

A virtual environment cannot be moved: its absolute path is written into `activate` and into every command. Built in `$HOME` and copied, it breaks for a colleague days later with `bad interpreter`.

The interpreter is recorded, not configured

A run writes the path of the Python that ran it into the solver's action programs. The environment must therefore sit at a path valid on every node.

Takeaway *Put the version in the folder name (`pfs0410`). Older environments stay beside the new one, and an old study stays reproducible.*

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Two folders with the same name

`py-venv/pfs0410` is the environment, and `venv` creates it. `pfs-wheels/pfs0410-linux` is the folder of wheels, and only the copy from the connected machine creates it. When the second is missing, `pip` answers:

```
WARNING: Location '.../pfs-wheels/pfs0410-linux' is ignored: it is either
a non-existing path or lacks a specific scheme.
ERROR: No matching distribution found for pyflightstream
```

Takeaway *An `ls` of `py-venv/` shows a folder that looks like proof. It is the other folder: list the one `pip` reads.*

Copyright (c) 2026 Geovana Neves, CC BY 4.0

One activation file for everybody (Linux)

```
# $FTS/bin/pfs0410-env
export PATH=<cluster-root>/fts-unix/py-venv/pfs0410/bin:$PATH
```

- Users write `source $FTS/bin/pfs0410-env` at the top of a session or of a job script.
- It exports `PATH` rather than defining an alias, because a job script does not expand aliases.
- The files of older environments stay beside it.
- To leave it: a new shell, or take the folder off `PATH` again.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Part 5

Running and recording

Running a matrix locally on Linux

The rule

To run a matrix locally on Linux, change only the executable path in executables.toml.

- The row's `FS_BUILD` id maps to the executable in `inputs/executables.toml`; a path true of one machine only goes in `inputs/executables.local.toml`, read over it. Nothing else changes.
- With a profile in `inputs/hpc/` the run would submit; keep it here:

```
source $FTS/bin/pfs0410-env
pyfs-matrix run matriz.fs --workspace . --local
```

Records say `forced_local`; no `collect` needed afterwards.

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Record what you did

Fact	Why the next person needs it
where the 3.12 came from	the module name, or the standalone archive and its sha256
the environment folder	`py-venv/pfs0410`, and the activation file
the package version	what `pyflightstream.__version__` printed
the platform tags used	the download command, verbatim
when, and by whom	so the next installation is the same one

Takeaway *Two facts exist only in the head of whoever installed. Write them down, beside the environment.*

Copyright (c) 2026 Geovana Neves, CC BY 4.0

Command summary

Windows target

```
pip download "pyflightstream[fsi]==0.58.2" ^
  -d W --python-version 312 ^
  --only-binary=:all: --platform win_amd64
py -3.12 -m venv .venv
.venv\Scripts\activate
pip install --no-index --find-links W ^
  "pyflightstream[fsi]"
pyfs-matrix --help
```

Linux target

```
pip download "pyflightstream[fsi]==0.58.2" \
  -d L --python-version 312 \
  --only-binary=:all: \
  --platform manylinux_2_28_x86_64 \
  --platform manylinux2014_x86_64
$FTS/python/bin/python3.12 -m venv --copies $FTS/py-
      venv/E
source $FTS/py-venv/E/bin/activate
pip install --no-index --find-links L \
  "pyflightstream[fsi]"
```

Copyright (c) 2026 Geovana Neves, CC BY 4.0

References

[1] Geovana Neves, *pyflightstream documentation*, page "Getting started". `https://nevesgeovana.github.io/pyflightstream/latest/getting-started/`, and the repository's `README.md`: `https://github.com/nevesgeovana/pyflightstream`.

[2] Python Packaging Authority, *pip documentation*: `pip download` and `pip install`, 2026. `https://pip.pypa.io/en/stable/cli/pip_download/`

[3] Scientific Python, *SPEC 0: minimum supported dependencies*. `https://scientific-python.org/specs/spec-0000/`

REF-57 N. J. Smith, T. Kluyver, *PEP 600: future manylinux platform tags for portable Linux built distributions*, 2019. `https://peps.python.org/pep-0600/`

[4] Python Software Foundation, *Python releases for Windows*. `https://www.python.org/downloads/windows/`

[5] Astral, *python-build-standalone*: relocatable CPython builds. `https://github.com/astral-sh/python-build-standalone`

Copyright (c) 2026 Geovana Neves, CC BY 4.0