

Analysis as Applications: Quick introduction to lockfiles

Matthew Feickert

(@HEPfeickert)

Reproducibility is a hard problem in science

Lorena A. Barba group

Barbagroup reproducibility syllabus



Lockheed P-80A airplane (1946). Credit: NASA Commons. —A reminder to test your code.

Posted on 10.31.2016

Also published on the Medium publication "[Hacker Noon](#)."

After my short piece, "[A hard road to reproducibility](#)," appeared in *Science*, I received several emails and Twitter mentions asking for more specific tips—both about tools and documents we use in the group to train the team about reproducibility.

“ In answer to popular demand, then, I have collected here what we could call the “Barba-group Reproducibility Syllabus.”

”

Your environment is
part of your science.

Treat your analysis like
a Python application.

There are problems with
what I'm about to show.

But this is to get thinking.

Lock files are a thing

I'm running out of time, so won't iterate here (conda-lock, pdm, poetry, pipenv, pip-tools). Not sure what to do? Here's a "get going and do something" introduction to lock files with pip-tools.

```
# In a virtual environment of course  
$ python -m pip install pip-tools
```

high level requirements.txt

```
$ cat requirements.txt
```

```
numpy
```

```
matplotlib
```

Compile high level dependencies into a lock file

```
$ pip-compile \  
    --generate-hashes \  
    --output-file requirements.lock \  
    requirements.txt
```

pip-secure-install with pip-tools lock files



Brett Cannon

☰ README.md

pip-secure-install

A GitHub action to have [pip](#) install from a requirements file as securely as possible.

Inputs

python

The command to run Python (as `-m` is used to run pip). Defaults to `python`.

requirements-file

The path to the requirements file. Defaults to `requirements.txt`.

options

Additional command-line options to pass to pip (e.g. `--target`).

Design

A few options are turned on for pip to make sure [installations are secure](#) and reproducible:

- A requirements file must be specified to make sure all dependencies are known statically for auditing purposes (`-F`).
- No dependency resolution is done to make sure the requirements file is complete (`--no-deps`).
- All requirements must have a hash provided to make sure the files have not been tampered with (`--require-hashes`).
- Only wheels are allowed to have reproducible installs (`--only-binary :all:`).

Install with pip-secure-install

```
$ python -m pip install \  
    --no-deps \  
    --require-hashes \  
    --only-binary :all: \  
    --requirement requirements.lock
```

Pin high level requirements.txt

```
$ cat requirements.txt
```

```
numpy==1.23.1
```

```
matplotlib==3.5.2
```

Commit the high level requirements.txt and the requirements.lock lock file to version control

Actually there's no time...to the asciinema!

```
--hash=sha256:50dff9cc21826d2977ef2d2a205504034e3a4563ca6f5db739b0d1026658e004 \
--hash=sha256:510cef4a3f401c246cfd8227b300828715dd055463cdca6176c2e4036df8bd4f \
--hash=sha256:5aed7dce90403cd9db06a115c78d0145c83070e864c1de1064f52ee6f6b1b20 \
--hash=sha256:690d1a15d7ba3694631e08f046e65ab031911c11f44920e97fed5eb0b0c1d \
--hash=sha256:6b7088c1ce160f50ea40764f825ec9b72ed9da25346216b91361ee1f8ad1b08f6 \
--hash=sha256:6e8c66f70fb539301e064f6478d7453e820d8a2c631da948a23384065cd95544 \
--hash=sha256:727dd1389bc5cb9827cbd1f9d40d2c2a1a0c9b32dd2261db522d2a004a6e0cc \
--hash=sha256:74a04183e6e4930b667d321524e3c5361094bb4af9083db5c301db64cd341f3 \
--hash=sha256:75e36f9d3e0fb872693f232cb8a5ff2cd578801251f3a4f6854c6a5d437d3c04 \
--hash=sha256:7761afe0126d046974a01e030ae7529e0ca6a196de3ec6937c11df0d1b1bc91c \
--hash=sha256:7888310f6214f19ab2b6df90f3f06afa3df7ef7355fc025e78a3044737fab1f5 \
--hash=sha256:7b0554af24df2bf96618dac71ddada02420f9460e943b181108cac55a7a2dcd4 \
--hash=sha256:7c7b502bc34f6e32ba022b4a209638f9e097d7a9098104ae420eb8186217ebbb \
--hash=sha256:808add66ea764ed97d44dda1ac4f2cfec4c1867d9efb16a33d158be79f32b8a4 \
--hash=sha256:831e648102c8f152e14c1a0938689dbb22480c548c484b8b248b350967b88c \
--hash=sha256:93689632949aff41199090eff5f474f3990b6823404e45d66a544304e9cdc467 \
--hash=sha256:96b5e6874431df16aee0c1ba237574cb6df1dc173798faa6a9d8b399a05d0e \
--hash=sha256:9a54614049a18a2d6fe156e68e188da02a046a4a93cf24f373b7fd977e943421 \
--hash=sha256:a138441e95562b3c078746a22f8fca8ff1c22c014f856278b0dbd89ca36cfff1b \
--hash=sha256:a647c044780995c5e54615a2e5360ccedd2f85e70ab57f7be817ca61305e630b \
--hash=sha256:a940bc48970ab30906d7a85afac4b944a572a7432e0698a7239f444a4e6f7b \
--hash=sha256:ad2277b185ebce4763f44dc3802e30f057c2b680f6d34e550bae4651872df3 \
--hash=sha256:b6d5e92df2b77665e07ddb2e4dbd6d644b7e4c0d2e972a852627c0ba0d75cf \
--hash=sha256:bc431b665722a5ad1fbd4f354fb933b7a582a5ee39a906ffffb680d72f27a1 \
--hash=sha256:bdd0de2d6468ecae88dd8935012c4a72681e5df632af903a1dca8c567aa871a \
--hash=sha256:c79690d4cd9318d9481d89a77e2d3fcaeff5486be641e60a4b49f3d2ccae420 \
--hash=sha256:cb6259196a589123d755380b65127ddc0f4c64b21f3bb46ce3a6ea6636590b \
--hash=sha256:d5b87da5a00ac5b6bad5c3aa3b86505f559b84f39035b233d5bf844b0834b1 \
--hash=sha256:dcd7b9cf7139dc8258d164b55696ecd16c04607f1cc33ba7af86613801ffe4ac0 \
--hash=sha256:dfe4c1fedfde4e2fbc009d5ad420647f7730d719786388b7de0999bf32c0d9fd \
--hash=sha256:ea98f633d45f7e815db648df7f0f19e328302c36427343e4432c84432e7ff4 \
--hash=sha256:ec52c351b35ca269cb1f8069d610fc45c5bd38c3e91f9ab4cbbf0aebc136d9c8 \
--hash=sha256:ee7f592281f7c174d3d6cbfbb7ee5984a671fcd77e3f78e973d492e9bf0eb3f \
--hash=sha256:f07f1f00e22b231dd3d9b9208692042e29792d6bd4f6639415d2f23158a0013 \
--hash=sha256:f3fac744f9b540148fa7715a435d2283b71f68bfb6d4aae24482a890aed18b59 \
--hash=sha256:fa768f5f9f958270b081bb33581b4b569faabf8774726b283ed06617101dc \
--hash=sha256:fac2d65901fb0fdf20363fbd345c01958a742f2dc2a8dd4495af66e3ff502a4

# via matplotlib
pyarsing==3.0.9 \
--hash=sha256:2b020ecf7d21b687f219b71ecad3631f644a4f0b1403fa1d1036b0c6416d70fb \
--hash=sha256:5026bae9a10eeaf6b1dab2f09052b9f4307d44aee4eda64b309723d8d206bbc

# via
# matplotlib
# packaging
python-dateutil==2.8.2 \
--hash=sha256:0123cacc1627ae19ddf3c27a5de5bd67ee4586fdd6440d9748fabb483d3e06 \
--hash=sha256:961d03dc3453ebbc594ddea9e4e11c5651520a876d0f4db161e8674aae935da9

# via matplotlib
six==1.16.0 \
--hash=sha256:1e61c37477a1626458e36f7b1d82aa5c9b094fa4802892072e49de9c60c4c926 \
--hash=sha256:8abb2f1d86890a2fbb999999a77cfd3e47c2a354b0111771326f8aa26e0254

# via python-dateutil
(envv) root@d8436cd8d409a: /#
```



<https://asciinema.org/a/508642>

Understand your environment.

Do science.

Don't forget to be awesome!

pip-secure-install

Brett Cannon's "pip-secure-install" recipe:

Use pip flags to make installs secure and reproducible:

- A requirements file must be specified to make sure all dependencies are known statically for auditing purposes (`--requirement`).
- No dependency resolution is done to make sure the requirements file is complete (`--no-deps`).
- All requirements must have a hash provided to make sure the files have not been tampered with (`--require-hashes`).
- Only wheels are allowed to have reproducible installs (`--only-binary :all:`).