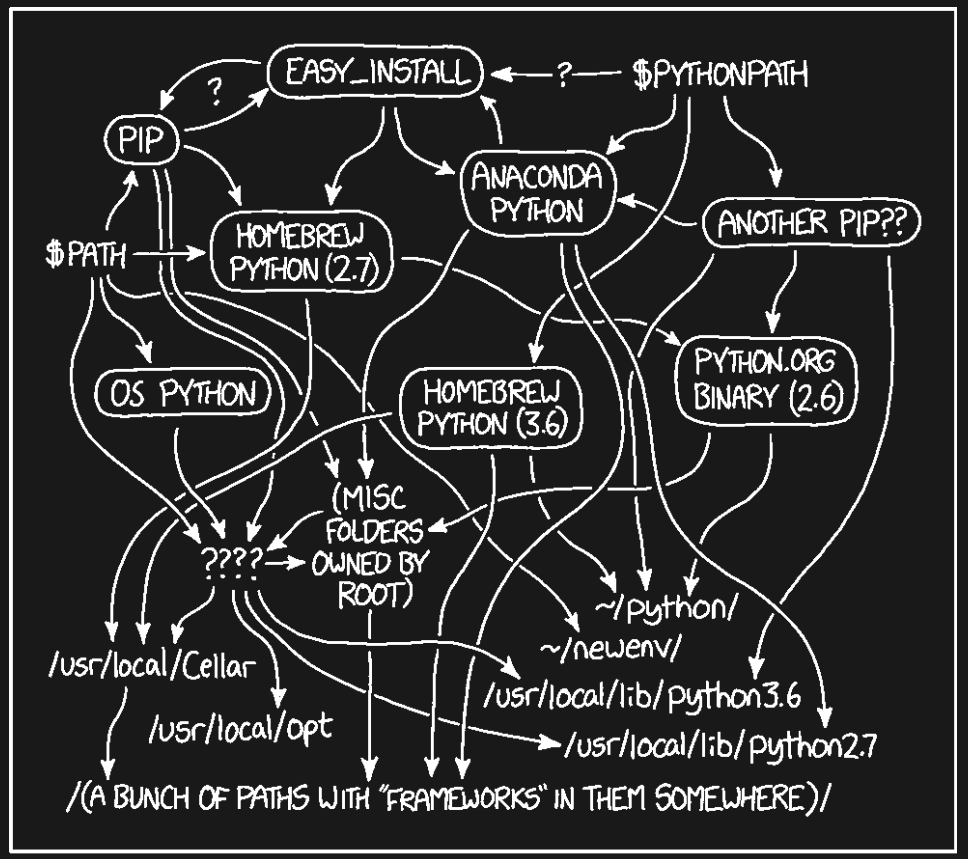
Reproducible environments with Guix

or how to be able to run your current code in ~5 years

Iliya Tikhonenko

(MPE)



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

xkcd #1987 (30.04.2018)

2023-11-13, USM Code Coffee

Functional package management

What's it?

1. Install a package

```
$ guix install firefox
$ apt install firefox
```

2. Delete a package

```
$ guix remove firefox
$ apt remove firefox
```

3. Update database

```
$ guix pull
$ apt update
```

4. Undo

```
$ guix package --roll-back
$ apt 🤷🤷🤷
```

GNU Guix is a

- functional

  (package contents explicitly depend on its dependencies and *isolated* build environment)
- transactional

  (if something does wrong during the upgrade, the system just remains in its original state)
- source-based

  (everything is built from source, but in most cases binary substitutes are used)

package manager with particular focus on *reproducibility*.

```
$ guix install samtools bowtie
```

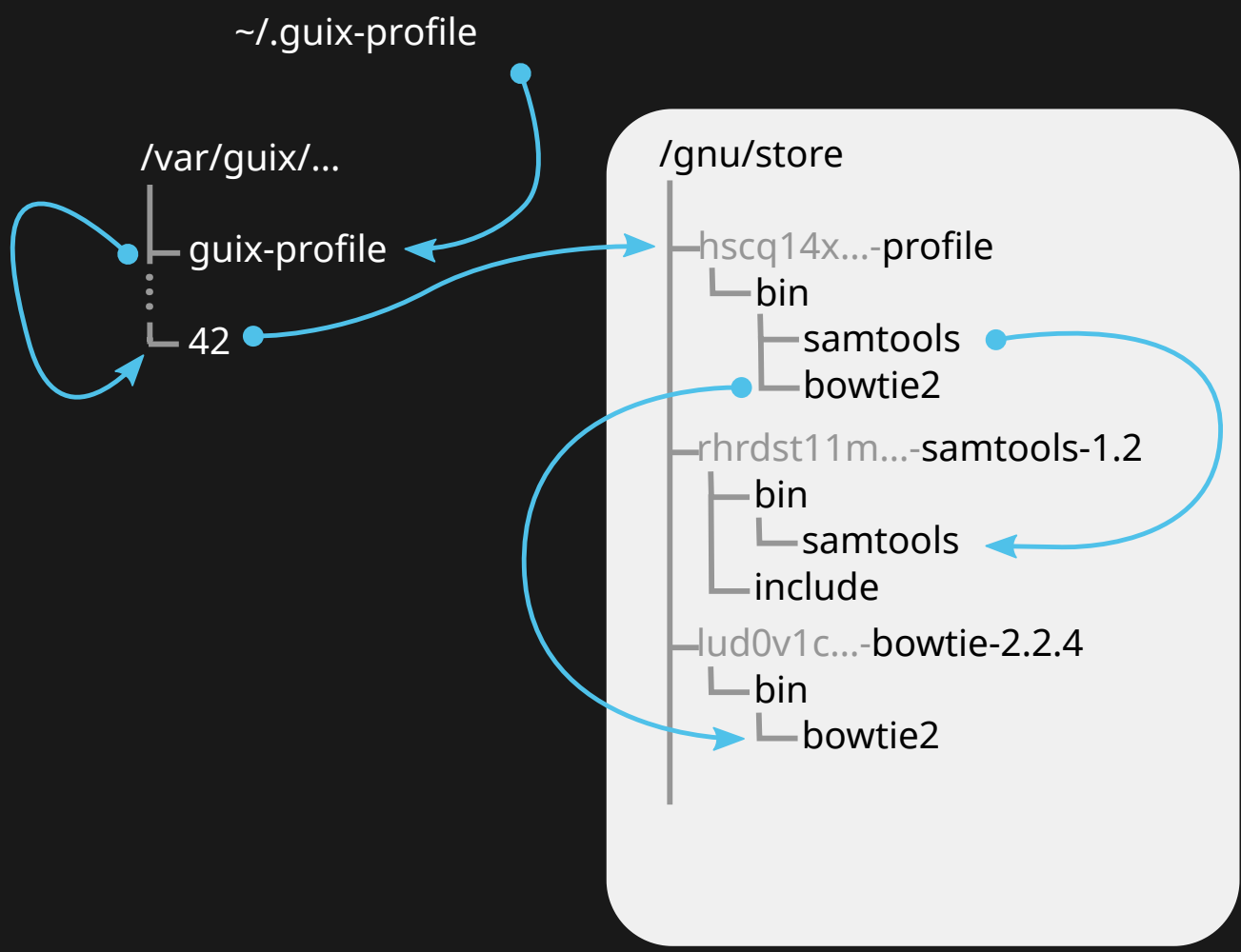


Figure stolen from C. Baines slides who in turn gives credit to Ricardo Wurmus

```
$ guix remove bowtie
```

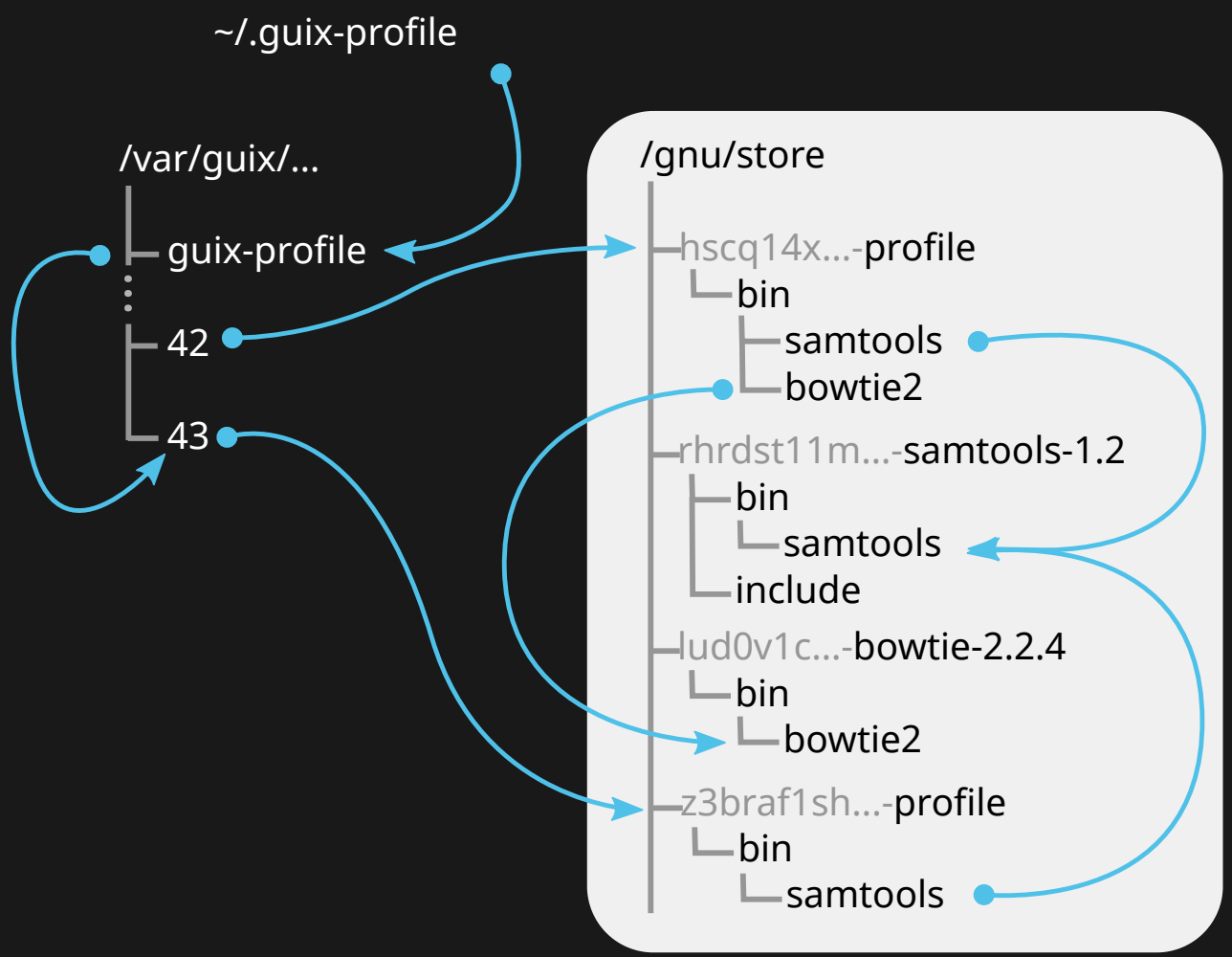


Figure stolen from C. Baines slides who in turn gives credit to Ricardo Wurmus

```
$ guix package --roll-back
```

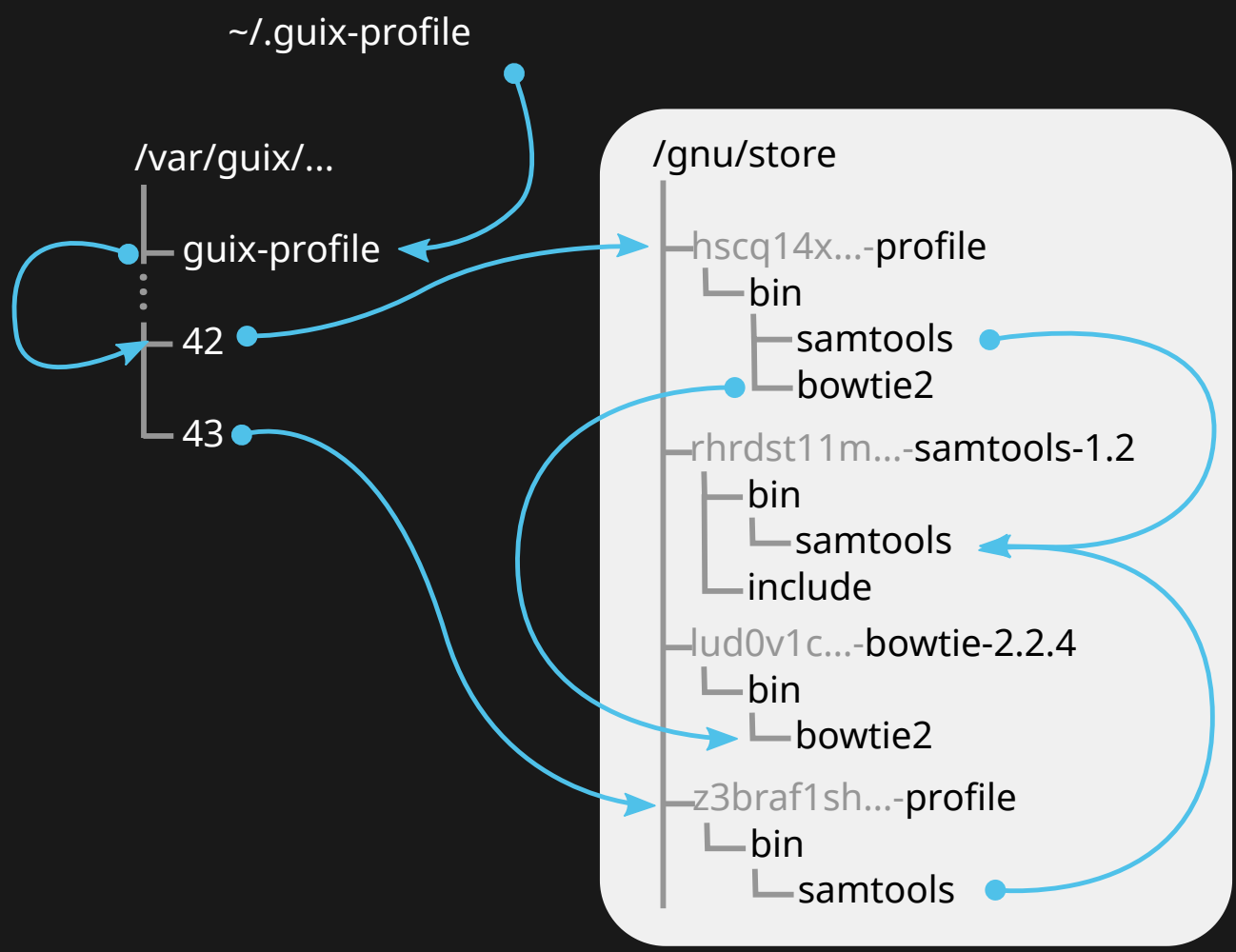


Figure stolen from C. Baines slides who in turn gives credit to Ricardo Wurmus

Managing computational environments

0. pip and virtualenv

1. Manages python dependencies
2. Does not ensure compatibility of dependencies (just installs them)
3. Makes assumptions about outside environment (python, C/C++/Fortran libraries)

An example

```
$ pip install scipy
 ... lots of errors ...
 numpy.distutils.system_info.NotFoundError: No BLAS/LAPACK libraries
found. Note: Accelerate is no longer supported.
  To build Scipy from sources, BLAS & LAPACK libraries need to be
installed.
  See site.cfg.example in the Scipy source directory and
  https://docs.scipy.org/doc/scipy/reference/building/index.html for
details.
  ----------------------------------------
```

1/2. environment modules

```
$ module purge
$ module load intel/19.1.3 impi/2019.9 mkl/2019.4 \
         anaconda/3/2021.11 cmake/3.24 hdf5-mpi/1.12.1
```

1. Adjusts environment variables (PATH etc) to make additional software available.
2. Consistency of dependencies – ensured ‘by hand’

`find-module hdf5-mpi | grep 1.14.2` on cobra.mpcdf.mpg.de

```
hdf5-mpi/1.14.1 (after loading gcc/10 impi/2021.6) hdf5-mpi/1.14.1 (after loading gcc/10 impi/2021.7)
hdf5-mpi/1.14.1 (after loading gcc/10 impi/2021.9) hdf5-mpi/1.14.1 (after loading gcc/10 openmpi/4)
hdf5-mpi/1.14.1 (after loading gcc/10 openmpi/4.1) hdf5-mpi/1.14.1 (after loading gcc/11 impi/2021.6)
hdf5-mpi/1.14.1 (after loading gcc/11 impi/2021.7) hdf5-mpi/1.14.1 (after loading gcc/11 impi/2021.9)
hdf5-mpi/1.14.1 (after loading gcc/11 openmpi/4)   hdf5-mpi/1.14.1 (after loading gcc/11 openmpi/4.1)
hdf5-mpi/1.14.1 (after loading gcc/12 impi/2021.6) hdf5-mpi/1.14.1 (after loading gcc/12 impi/2021.7)
hdf5-mpi/1.14.1 (after loading gcc/12 impi/2021.9) hdf5-mpi/1.14.1 (after loading gcc/12 openmpi/4)
hdf5-mpi/1.14.1 (after loading gcc/12 openmpi/4.1) hdf5-mpi/1.14.1 (after loading gcc/13 impi/2021.6)
hdf5-mpi/1.14.1 (after loading gcc/13 impi/2021.7) hdf5-mpi/1.14.1 (after loading gcc/13 impi/2021.9)
hdf5-mpi/1.14.1 (after loading gcc/13 openmpi/4)   hdf5-mpi/1.14.1 (after loading gcc/13 openmpi/4.1)
hdf5-mpi/1.14.1 (after loading intel/2023.1.0.x impi/2021.9) hdf5-mpi/1.14.1 (after loading intel/2023.1.0.x openmpi/4)
hdf5-mpi/1.14.1 (after loading intel/2023.1.0.x openmpi/4.1) hdf5-mpi/1.14.1 (after loading intel/21.6.0 impi/2021.6)
hdf5-mpi/1.14.1 (after loading intel/21.6.0 openmpi/4)   hdf5-mpi/1.14.1 (after loading intel/21.6.0 openmpi/4.1)
hdf5-mpi/1.14.1 (after loading intel/21.7.1 impi/2021.7) hdf5-mpi/1.14.1 (after loading intel/21.7.1 openmpi/4)
hdf5-mpi/1.14.1 (after loading intel/21.7.1 openmpi/4.1)
```

… but what if I need `hdf5-mpi/1.14` and `intel/19`?

1. conda (or mamba)

1. Manages *packages* (not limited to python)
2. Solves entire *environments*

```
$ conda create -n test astropy

The following packages will be downloaded:
    package                    |            build
    ---------------------------|-----------------
    ...
    astropy-5.3.4              |  py312ha883a20_0         8.8 MB
    blas-1.0                   |              mkl           6 KB
    bzip2-1.0.8                |       h7b6447c_0          78 KB
    ...
------------------------------------------------------------
                                           Total:       269.6 MB
```

1. conda (or mamba) – hidden assumptions

1. Manages *binaries* [1]:
   - compiler versions?
   - architecture-specific optimizations?
2. `ldd $HOME/opt/micromamba/envs/…/bin/gfortran`

```
linux-vdso.so.1 (0×00007ffcf874f000)
libm.so.6 ⇒ /lib/x86_64-linux-gnu/libm.so.6 (0×00007f7058af8000)
libc.so.6 ⇒ /lib/x86_64-linux-gnu/libc.so.6 (0×00007f7058906000)
/lib64/ld-linux-x86-64.so.2 (0×00007f7058c6b000)
```

… still depends on the underlying system

1. However, see conda-forge and feedstocks

2. Docker or Apptainer

1. Manages self-contained *images*
2. Builds *isolated* (using Linux kernel features) environments
3. Environments are immutable, but can be extended

Example (non-scientific..)

```
$ docker run --name … -e POSTGRES_PASSWORD=… -d postgres
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
578acb154839: Downloading
[==========================================>    ]  27.28MB/29.15MB
8a9a8dd839ec: Download complete
9a28d48e5d8f: Download complete
…
```

Dockerfiles (and reproducibility)

slothai/numpy does not use apk update (i.e. a dependency on a remote resource), but some Dockerfiles do

```
 1  FROM slothai/openblas as openblas
 2  FROM python:3.6.5-alpine3.7
 3  # Metadata as defined at http://label-schema.org
 4  ARG BUILD_DATE
 5  ARG VCS_REF
 6  ARG NUMPY_VERSION=1.14.3
 7
 8  COPY --from=openblas /opt/OpenBLAS/ /opt/OpenBLAS/
 9  LABEL org.label-schema.build-date=$BUILD_DATE \
10      org.label-schema.name="NumPy" \
11      org.label-schema.vcs-ref=$VCS_REF \
12      org.label-schema.vcs-url="https://github.com/slothai/docker-numpy" \
13      org.label-schema.vendor="SlothAI <https://slothai.github.io/>" \
14      org.label-schema.schema-version="1.0"
15  RUN apk add --no-cache --virtual .meta-build-dependencies \
16          gcc \
17          musl-dev && \
18      apk add --no-cache gfortran && \
19      wget -O numpy.tar.gz "https://github.com/numpy/numpy/archive/v$NUMPY_VERSION.tar.gz" && \
20      tar xzf numpy.tar.gz && rm -f numpy.tar.gz && \
21      echo -e "[openblas]\nlibraries = openblas\n\
22  library_dirs = /opt/OpenBLAS/lib\n\
23  include_dirs = /opt/OpenBLAS/include\n\
24  runtime_library_dirs = /opt/OpenBLAS/lib\
25  " > /numpy-$NUMPY_VERSION/site.cfg && \
26      pip install Cython==0.28.2 && \
27      cd /numpy-$NUMPY_VERSION/ && python setup.py build --parallel=$(nproc) --fcompiler=gfortran && \
28      python setup.py install && cd / && \
29      rm -rf /numpy-$NUMPY_VERSION/ && \
30      pip uninstall --yes Cython && \
31      apk del .meta-build-dependencies && \
32      find / -type d -name __pycache__ -exec rm -r {} +
```

Guix Examples

Creating an environment

Declaring everything at once (manifests)

```
(specifications→manifest
  (list "openblas" "openmpi"
        ;; …
        "gcc-toolchain"))
```

and then

```
$ guix shell -m manifest.scm strace
                            #^^^^ additional package for debugging
```

Adding packages iteratively (profiles)

```
$ guix install openblas -p ./myenv
$ guix install openmpi -p ./myenv
$ guix package --export-manifest -p ./myenv > manifest.scm
```

Running code in the environment

```
$ guix shell -m manifest.scm
```

Pure environments

```
$ guix shell --pure --preserve=^OMP_NUM_THREADS …
```

unsets all environment variables (`PATH` and friends) except those that are `--preserve`d.

Containers

```
$ guix shell --container \
  --share=output-dir \
  --expose=…=/input-dir \
  -m manifest.scm \
  coreutils strace grep sed -- ./run -i /input-dir -o output-dir
```

process, filesystem (unless `--exposed` or `--shared`) and network isolation (unless `-N` is used) using Linux cgroups & namespaces.

< manifest for the current project >

Reproducing environments

What we need

1. Code and data
2. `manifest.scm`
3. A description the current Guix and used channels:

```
guix describe -f channels > guix-version-for-reproduction.txt
```

How to reproduce

```
guix time-machine -C guix-version-for-reproduction.txt -- shell -C ...
```

< code from 2021 example >

Missing software?

Custom channels

- E.g. https://github.com/guix-science/guix-science

```
;; add to ~/.config/guix/channels.scm
(channel
  (name 'guix-science)
  (url "https://github.com/guix-science/guix-science.git")
  (introduction
   (make-channel-introduction
        "b1fe5aaff3ab48e798a4cce02f0212bc91f423dc"
        (openpgp-fingerprint
         "CA4F 8CF4 37D7 478F DA05  5FD4 4213 7701 1A37 8446"))))
```

Import from another PM

```
$ guix import … >> /path/to/custom/definitions/file.scm
$ guix shell -L /path/to/custom/definitions …
```

Caveats

- Guix does not natively work on MacOS (but can work on a Linux VM inside)
- Requires root for installation, not available on an average HPC cluster (but there is `guix pack`);
- There is a lot of software still missing / outdated;
- Not as easy and fast as `conda install`;
- Scheme for writing packages / manifests;
- Build farm (ci.guix.gnu.org) sometimes fails to build everything in time and Guix starts building packages, which can take a while (use `guix weather …` to check);
- `/gnu/store` can weight a few hundred of gigabytes (but there is `guix gc` to clean currently unused package versions).

Links

- Nix, another functional package manager with similar features.
- Guix official documentation and blog.
- A community package search in non-official channels.
- Reproducible Software Environments in HPC workshop webpage with slides and recordings.
- What's in a package, a (bit sad) story about conda, nix, spack, guix and packaging `pytorch`
- "Toward practical transparent verifiable and long-term reproducible research using Guix", a paper in Nature Scientific Data.
- https://www.cbaines.net/projects/guix/freenode-live-2017/presentation
- Revealjs version of this presentation

Questions?