How to get started writing Guix packages

Andreas Enge

CANARI project-team
INRIA Bordeaux
andreas.enge@inria.fr
https://enge.math.u-bordeaux.fr/

Journées du réseau français de recherche reproductible
Bordeaux
19 mai 2026

Connect to Grid'5000

- If you brought your own device – fine!
- Otherwise add this to `.ssh/config`:

```
Host g5k
  User LOGIN
  Hostname access.grid5000.fr
  ForwardAgent no
Host *.g5k
  User LOGIN
  ProxyCommand ssh g5k -W "$(basename %h .g5k):%p"
  ForwardAgent no
```

- Connect to a reserved machine:

```
ssh lille.g5k
oarsub -I -t inner=2145873 \
   -l host=1/core=1,walltime=02:00:00 \
   --project lab-2026-jrfrr
```

How to get started writing Guix packages

1. Using an importer
2. Updating a package
3. Creating a package from scratch
4. Channelling your energy
5. Contributing to Guix

Import packages from language repositories

- Before you start: Check the channels!

  https://hpc.guix.info/browse

- Digression
  - Many Guix developers use Emacs.
  - Many potential Guix contributors think that Emacs is a requirement.
  - This is a myth. You can also use a text editor.
  - Difficulty: Guix is so flexible that there is no canonical way.
- Available importers:

  ```
  guix import --help
  ```

  composer, cpan, cran, crate, egg, elm, elpa, gem, gnu, go, hackage, hexpm, json, luanti, minetest, npm-binary, nuget, opam, pypi, stackage, texlive
- Warning: Rust is... special.

  https://guix.gnu.org/cookbook/en/html_node/Common-Workflow-for-Rust-Packaging.html

Import a PyPI package

- Find an interesting package.
  https://pypi.org/project/robotics-numpy/
- Import it

  ```
  guix import pypi robotics-numpy
  ```

  This prints a package record.
- Put it into a file.

  ```
  mkdir proj
  guix import pypi robotics-numpy > proj/bordeaux.scm
  ```
- Compile it

  ```
  guix build -f proj/bordeaux.scm
  ```
- Boilerplate missing!
  Copy-paste it from elsewhere, or let the system help you.

Import a PyPI package

- While we are at it, turn the value into a public variable and encapsulate it into its own module:

```
(define-module (bordeaux)
  #:use-module  ((guix licenses) #:prefix license:)
  #:use-module  (guix download)
  #:use-module  (guix packages)
  #:use-module  (guix build-system pyproject)
  #:use-module  (guix build-system python)
  #:use-module  (gnu packages check)
  #:use-module  (gnu packages python-build)
  #:use-module  (gnu packages python-check)
  #:use-module  (gnu packages python-science)
  #:use-module  (gnu packages python-xyz))
```

Import a PyPI package

- Additional difficulty:
  We do not have `python-ruff`.
- Test framework in `native-inputs`.
- Solution (?): Drop it and disable tests!

```
(define-public python-robotics-numpy
  (package
    ...
    (arguments
     (list #:tests? #f))
    ...
    (license license:expat)))
```

- Compile the package using its name

```
guix build -L proj python-robotics-numpy
```

Import a PyPI package

- Is it really there?

```
guix package -L proj -A numpy
```

- Check the package.

```
guix lint -L proj python-robotics-numpy
```

- Improve the metadata...
- Solution
-
```
$ guix shell -L proj python-robotics-numpy python
[env]$ python3
>>> import robotics_numpy as rn
>>> import numpy as np
...
```

- Useful tip: recursive importer

```
guix import pypi -r robotics-numpy
```

How to get started writing Guix packages

1. Using an importer
2. Updating a package
3. Creating a package from scratch
4. Channelling your energy
5. Contributing to Guix

Update a package

- Even easier! Download an outdated package.
- Check for updates.

```
guix refresh -L $PWD/proj ruby-mysql2
```

- Update in place.

```
guix refresh -L $PWD/proj -u ruby-mysql2
```

(For a reason I do not understand, we need an absolute path here.)

- Admire the change.

```
diff exercises/bordeaux-2.scm proj/bordeaux-2.scm
```

- Build the package.

```
guix build -L proj ruby-mysql2
```

Exercise

Import your favourite package.

How to get started writing Guix packages

1. Using an importer
2. Updating a package
3. Creating a package from scratch
4. Channelling your energy
5. Contributing to Guix

Start easily

- Find an interesting package.
  Simon Tatham's Portable Puzzle Collection
  https://www.chiark.greenend.org.uk/~sgtatham/puzzles/
- Check the license.
  https://www.chiark.greenend.org.uk/~sgtatham/puzzles/doc/licence.html#licence
- Find the source code.
  https://git.tartarus.org/simon/puzzles.git
- Create a package skeleton (do not use Icecat...).
  https://guix-hpc.gitlabpages.inria.fr/guix-packager/
  Download result to `proj/bordeaux-3.scm`
- Try to build.

```
guix build -L proj puzzles -K
```

Complete the package

- Add `native-inputs`.
  Here: pkg-config
- Add `inputs`.
  Here: gtk+
- Disable tests.
- ...
- Solution
- Run the games!

```
guix shell -L proj puzzles -- mines
```

How to get started writing Guix packages

1. Using an importer
2. Updating a package
3. Creating a package from scratch
4. Channelling your energy
5. Contributing to Guix

Create a custom channel

- Move your package to git.

```
cd proj
git init .
git add .
git commit
cd ..
```

- Create a channel.

```
cat > channels.scm << EOF
(cons* (channel
         (name 'bordeaux)
         (url "file://`pwd`/proj"))
       %default-channels)
EOF
```

- Use the channel.

```
guix time-machine -C channels.scm -- \
   build puzzles
```

Make things permanent

Several options.

1. Work for yourself.
   - Move the `channels.scm` file to `$HOME/.config/guix/channels.scm`
   - Update the Guix view of the world.

     ```
     guix pull
     ```
2. Work with a group.
   - Move the git repository to a shared server.
   - Replace

     `file:///...`

     by something like

     `https://gitlab.inria.fr/guix-hpc/guix-hpc.git`
3. Work with the world.
   - Contribute your package to Guix upstream.

Exercise

Create your favourite package.

How to get started writing Guix packages

1. Using an importer
2. Updating a package
3. Creating a package from scratch
4. Channelling your energy
5. Contributing to Guix

Set up Guix

- See here:
  https://guix.gnu.org/manual/devel/en/guix.html#Contributing
- Clone the repository.

```
git clone https://codeberg.org/guix/guix.git
```

- Put yourself in an environment with all Guix dependencies.

```
guix shell -D guix --pure
```

- Build Guix.

```
cd guix
./bootstrap
./configure --localstatedir=/var --sysconfdir=/etc
make -j 4
```

... and wait

Test the package in situ

- Create a branch.

```
git checkout -b bordeaux
```

- Put the package into an existing module; say, into `gnu/packages/games.scm`.

```
./pre-inst-env guix edit puzzles
```

- Build it.

```
make
./pre-inst-env guix build puzzles
```

- Commit it.

```
./etc/committer.scm
```

Beautify the package, and go!

- Fix the indentation.

  ```
  ./pre-inst-env guix style puzzles
  ```

  (Result may or may not be beautiful.)
- Check for real problems.

  ```
  ./pre-inst-env guix lint puzzles
  ```
- Submit!
  Pull request on Codeberg.