Comment avoir plus de paquets pour Guix?

Simon Tournier

Mastodon: @zimoun@sciences.re
https://tournier.info

16 décembre 2025

https://guix.gnu.org
https://hpc.guix.info

CaféGuix

Scenarii

- Alice utilise `python@3.9` et `numpy@1.20.3`

```
$ sudo apt install python python-numpy
```

- Blake **collabore** avec Alice... mais utilise `python3.8` et `numpy@1.16.5` pour un autre projet

```
$ apt-cache madison python-numpy
python-numpy | 1:1.16.5-2ubuntu7 | ...
```

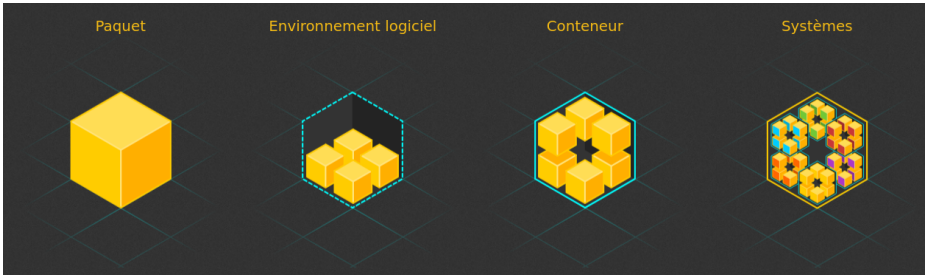
- Charlie **mets à jour** son système et **tout est cassé**

```
$ sudo apt upgrade
The following packages have unmet dependencies:
E: Broken packages
```

- Bob utilise les **mêmes versions** qu'Alice mais n'a **pas le même résultat**
- Dan essaie de **rejouer plus tard** le scénario d'Alice mais rencontre l'enfer des dépendances

C'est quoi Guix déjà ?

un **gestionnaire de paquets** (comme APT, Yum, etc.)
qui produit des ***packs* distribuables** (conteneur Docker ou Singularity)
qui génèrent des ***machines virtuelles* isolées** (*à la* Ansible ou Packer)
sur lequel on construit une distribution Linux **Guix System**
. . .et aussi une bibliothèque Scheme. . .



Ce que nous présentons fonctionne sur n'importe quelle distribution Linux

Mon usage principal

1. gestionnaire de paquets : APT (Debian/Ubuntu), YUM (RedHat), etc.
2. gestionnaire d'environnements : Conda, Pip, Modulefiles, etc.
3. conteneur : Docker, Singularity

Mon usage principal

1. gestionnaire de paquets : APT (Debian/Ubuntu), YUM (RedHat), etc.
2. gestionnaire d'environnements : Conda, Pip, Modulefiles, etc.
3. conteneur : Docker, Singularity

$$\text{Guix} = \#1 + \#2 + \#3$$

Mon usage principal

1. gestionnaire de paquets : APT (Debian/Ubuntu), YUM (RedHat), etc.
2. gestionnaire d'environnements : Conda, Pip, Modulefiles, etc.
3. conteneur : Docker, Singularity

Guix = #1 + #2 + #3

1. Co-installation	*"profil"*	~~`apt-get install`~~
2. Environnement à la volée	`guix shell`	~~`virtualenv`~~ ou ~~`conda`~~
3. Production d'images	`guix pack`	~~`FROM debian:stable`~~

Mon usage principal

1. gestionnaire de paquets : APT (Debian/Ubuntu), YUM (RedHat), etc.
2. gestionnaire d'environnements : Conda, Pip, Modulefiles, etc.
3. conteneur : Docker, Singularity

Guix = #1 + #2 + #3

1. Co-installation	*"profil"*	~~`apt-get install`~~
2. Environnement à la volée	`guix shell`	~~`virtualenv`~~ ou ~~`conda`~~
3. Production d'images	`guix pack`	~~`FROM debian:stable`~~

Redéployer un état avec `guix time-machine`

Mon usage principal

1. gestionnaire de paquets : APT (Debian/Ubuntu), YUM (RedHat), etc.
2. gestionnaire d'environnements : Conda, Pip, Modulefiles, etc.
3. conteneur : Docker, Singularity

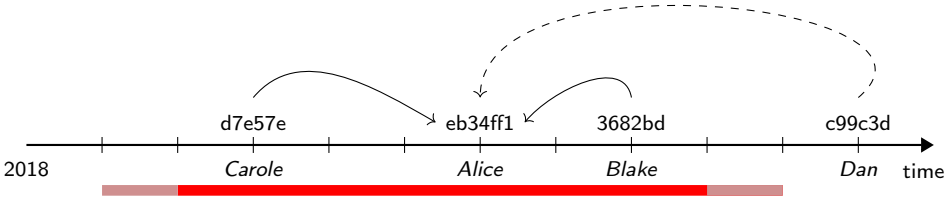
Guix = #1 + #2 + #3

1. Co-installation	*"profil"*	~~`apt-get install`~~
2. Environnement à la volée	`guix shell`	~~`virtualenv`~~ ou ~~`conda`~~
3. Production d'images	`guix pack`	~~`FROM debian:stable`~~

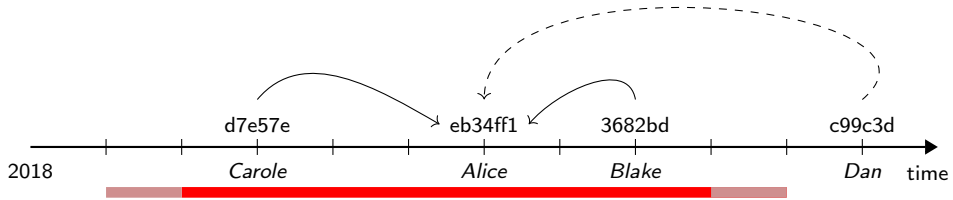
Redéployer un état avec `guix time-machine`

```
guix time-machine -C channels.scm -- shell -m manifest.scm
```

`guix time-machine` = **RE** dans redéploiement



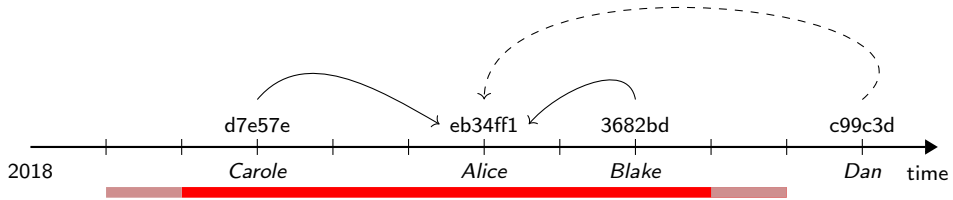
`guix time-machine` = **RE** dans redéploiement



Pour être **reproductible dans le temps**, il faut :

- Une préservation de **tous** les codes source (passerelle vers Software Heritage (lien))
- Une *backward* compatibilité du noyau Linux
- Une compatibilité du *hardware* (p. ex. CPU, disque dur (NVMe), etc.)

`guix time-machine` = **RE** dans redéploiement



Pour être **reproductible dans le temps**, il faut :

- Une préservation de **tous** les codes source (passerelle vers Software Heritage (lien))
- Une *backward* compatibilité du noyau Linux
- Une compatibilité du *hardware* (p. ex. CPU, disque dur (NVMe), etc.)

Quelle est la taille de la fenêtre temporelle avec les trois conditions satisfaites ?

(À ma connaissance, le projet Guix réalise une expérimentation grandeur nature et quasi-unique depuis sa v1.0 en 2019)

Guix s'installe sur **n'importe quelle distribution** Linux récente.

```
$ cd /tmp
$ wget https://codeberg.org/guix/guix/src/branch/master/etc/guix-install.sh
$ chmod +x guix-install.sh
$ sudo ./guix-install.sh
```

Pour commencer :

```
$ guix help
```

Le dernier jour du reste de votre vie

Guix s'installe sur **n'importe quelle distribution** Linux récente.

```
$ cd /tmp
$ wget https://codeberg.org/guix/guix/src/branch/master/etc/guix-install.sh
$ chmod +x guix-install.sh
$ sudo ./guix-install.sh
```

Pour commencer :

```
$ guix help
```

Et si la nouvelle année qui arrive était la bonne convaincre votre collègue ?

Nous allons discuter de. . .

```
$ guix package -A | wc -l
29741
```

30k paquets avec mises à jour et ajouts quotidiens

```
$ guix search julia csv | wc -l
0
```

Que faire ?

1. Canal existant ?
2. Ajout local ?
3. Son canal ?
4. Reproductibilité ?
5. Ce qu'il faut retenir

Qu'est-ce qu'un canal (*channel*) ?

`https://guix.gnu.org/manual/devel/en/guix.html#Channels`

un canal = un dépôt Git

Qu'est-ce qu'un canal (*channel*) ?

https://guix.gnu.org/manual/devel/en/guix.html#Channels

un canal = un dépôt Git

Plomberie

```
$ ls $HOME/.cache/guix/checkouts/
last-expiry-cleanup
7x37p6bub2newlthjx5kk2mco2aq44vxbqp5gwa3sifqxzfb3aqq

$ git -C $HOME/.cache/guix/checkouts/7x37p.../ remote -v
origin https://git.guix.gnu.org/guix.git (fetch)
origin https://git.guix.gnu.org/guix.git (push)
```

Qu'est-ce qu'un canal (*channel*) ? (2)

Ce que fait `guix pull`, c'est :

- mise à jour du dépôt Git (`git pull`)
- « compilation » pour créer le nouveau Guix

Qu'est-ce qu'un canal (*channel*) ? (2)

Ce que fait `guix pull`, c'est :

- mise à jour du dépôt Git (`git pull`)
- « compilation » pour créer le nouveau Guix

Cette compilation, qui semble parfois longue, est ce qui permet
le voyage dans le temps (`guix time-machine`).

https://guix.gnu.org/en/blog/2018/multi-dimensional-transactions-and-rollbacks-oh-my/
https://guix.gnu.org/en/blog/2021/outreachy-guix-git-log-internship-wrap-up/

Qu'est-ce qu'un canal (*channel*) ? (plus de plomberie !)

- le Guix courant est construit à partir de quel *commit* ?

```
$ guix describe
Generation 14 Oct 17 2025 17:27:17 (current)
  guix c787d90
    repository URL: https://git.guix.gnu.org/guix.git
    branch: master
    commit: c787d90fff634e93edaa49c7e8e90dd71668a570
```

Qu'est-ce qu'un canal (*channel*) ? (plus de plomberie !)

- le Guix courant est construit à partir de quel *commit* ?

```
$ guix describe
Generation 14 Oct 17 2025 17:27:17 (current)
  guix c787d90
    repository URL: https://git.guix.gnu.org/guix.git
    branch: master
    commit: c787d90fff634e93edaa49c7e8e90dd71668a570
```

- `guix time-machine` fait comme `guix pull` temporairement

```
$ guix time-machine --commit=3141c2c93f -- describe | grep commit
    commit: 3141c2c93f74b8221729c4123f152ef933f1887b
$ git -C $HOME/.cache/guix/checkouts/7x37p.../ log -1 --format="%h"
3141c2c93f
```

Qu'est-ce qu'un canal (*channel*) ? (plus de plomberie !)

- le Guix courant est construit à partir de quel *commit* ?

```
$ guix describe
Generation 14 Oct 17 2025 17:27:17 (current)
  guix c787d90
    repository URL: https://git.guix.gnu.org/guix.git
    branch: master
    commit: c787d90fff634e93edaa49c7e8e90dd71668a570
```

- `guix time-machine` fait comme `guix pull` temporairement

```
$ guix time-machine --commit=3141c2c93f -- describe | grep commit
    commit: 3141c2c93f74b8221729c4123f152ef933f1887b
$ git -C $HOME/.cache/guix/checkouts/7x37p.../ log -1 --format="%h"
3141c2c93f
```

⇒ l'état interne du dépôt Git n'est pas toujours le même état

Qu'est-ce qu'un canal (*channel*) ? (plus de plomberie !)

- le Guix courant est construit à partir de quel *commit* ?

```
$ guix describe
Generation 14 Oct 17 2025 17:27:17 (current)
  guix c787d90
    repository URL: https://git.guix.gnu.org/guix.git
    branch: master
    commit: c787d90fff634e93edaa49c7e8e90dd71668a570
```

- `guix time-machine` fait comme `guix pull` temporairement

```
$ guix time-machine --commit=3141c2c93f -- describe | grep commit
    commit: 3141c2c93f74b8221729c4123f152ef933f1887b
$ git -C $HOME/.cache/guix/checkouts/7x37p.../ log -1 --format="%h"
3141c2c93f
```

Tout est transparent pour l'utilisateur et il ne faut pas s'en soucier !

Ajout d'autres canaux

Fichier `channels.scm`

```
(list
 (channel
  (name 'guix-science)
  (url "https://codeberg.org/guix-science/guix-science.git")
  (branch "master"))
 (channel
  (name 'guix-science-nonfree)
  (url "https://codeberg.org/guix-science/guix-science-nonfree.git")
  (branch "master"))
 %default-channels)    <-- Guix lui-même
```

Ajout d'autres canaux

Fichier `channels.scm`

```
(list
 (channel
  (name 'guix-science)
  (url "https://codeberg.org/guix-science/guix-science.git")
  (branch "master"))
 (channel
  (name 'guix-science-nonfree)
  (url "https://codeberg.org/guix-science/guix-science-nonfree.git")
  (branch "master"))
 %default-channels)    <-- Guix lui-même
```

https://hpc.guix.info/channels

Oui, il est encore difficile de chercher dans les canaux. :-)

Trois façons pour prendre en compte

- `guix pull`
- avec la liste des canaux sauvegardée là `$HOME/.config/guix/channels.scm`
- **puis** `guix search julia csv`

Trois façons pour prendre en compte

- `guix pull`
- avec la liste des canaux sauvegardée là `$HOME/.config/guix/channels.scm`
- **puis** `guix search julia csv`

- `guix pull -C channels.scm`
- **puis** `guix search julia csv`

Trois façons pour prendre en compte

- `guix pull`
- avec la liste des canaux sauvegardée là `$HOME/.config/guix/channels.scm`
- **puis** `guix search julia csv`

- `guix pull -C channels.scm`
- **puis** `guix search julia csv`

- `guix time-machine -C channels.scm -- search julia csv`

Ma façon de faire *(peut-être pas la bonne)*

`guix pull`

```
$ cat $HOME/.config/guix/channels.scm
(use-modules (guix ci))

(list (channel-with-substitutes-available
       %default-guix-channel
       "https://ci.guix.gnu.org"))
```

`guix time-machine -C extra-channels.scm --` *commande*

(Je ne fais très peu `guix pull` mais presque exclusivement `guix time-machine`, surtout avec l'efficacité des mécanismes de *cache*.)

Quand le paquet n'est pas disponible ?

```
$ guix time-machine -C all-channels-on-Earth.scm \
       -- search julia csv | wc -l
0
```

Il faut donc empaqueter !

Empaqueter est un vrai *artisanat*

L'artisanat est la transformation de produits ou la mise en œuvre de services grâce à un savoir-faire particulier et hors contexte industriel de masse : l'artisan assure en général tous les stades de sa transformation [...]
https://fr.wikipedia.org/wiki/Artisanat

Points clés

- Il n'y a pas de secret, empaqueter prend du temps
- La difficulté est très variable, et dépend aussi l'écosystème

  p. ex. R est plus facile que Python
- Il y a deux situations :
  - un paquet totalement nouveau
  - un paquet variant

  p. ex. une autre version
- L'option `--load-path`/`-L` facilite le cycle de tests/erreurs
- `guix import --help`

```
guix import cran Seurat --recursive                    (- :
guix import pypi pytorch                               )- :
```

Exemple : empaqueter `julia-csv`

Oups, il n'y pas d'*importer*.

Le boulot fastidieux commence...

- https://github.com/JuliaData/CSV.jl
- Julia central *registery*
- Dépend des paquets Julia Parsers, PooledArrays, etc.

Exemple : init

```
$ cat path/to/more-packages/custom.scm
(define-module (custom)
  #:use-module (guix packages)
  #:use-module (guix git-download)
  #:use-module (guix build-system julia)
  #:use-module ((guix licenses) #:prefix license:)
  #:use-module (gnu packages julia-xyz)
[...]
```

guix *cmd* -L path/to/more-packages

cmd = `show`
ou `build`
ou `lint`
ou autres

```
(define-public julia-parsers
  (package
    (name "julia-parsers")
    (version "2.2.4")
    (source (origin
              (method git-fetch)
              (uri (git-reference
                    (url "https://github.com/JuliaData/Parsers.jl"
                    (commit (string-append "v" version))))
              (file-name (git-file-name name version))
              (sha256
               (base32 "09v2x9yd1wdp74hzsf6218dpamlf2hb5nkmixqb4bc
    (build-system julia-build-system)
    (home-page "https://github.com/JuliaData/Parsers.jl")
    (synopsis "Fast parsing machinery for basic types in Julia")
    (description "@code{Parsers.jl} is a collection of type parser
utilities for Julia.")
    (license license:expat)))
```

Après du travail...

```
$ ag --scheme define-public more-packages/ | wc -l
5
```

```
$ guix search -L more-packages/ julia csv | recsel -p name,version
name: julia-csv
version: 0.10.4

$ guix shell -L more-packages/ julia julia-csv \
       -- julia -e 'using CSV; print(methods(CSV.File))'
# 6 methods for type constructor:
[1] CSV.File(ctx::CSV.Context) in CSV at /gnu/store/...-profile/share/julia/loa
[2] CSV.File(ctx::CSV.Context, chunking::Bool) in CSV at /gnu/store/...-profile
...
```

On souhaite partager

Le plus souhaitable est de contribuer à un canal déjà existant

un canal = un dépôt Git

```
cd path/to/more-packages
git init
git add *.scm
git commit -m 'Message utile'
```

Et voilà !

https://guix.gnu.org/manual/devel/en/guix.html#Creating-a-Channel
https://guix.gnu.org/manual/devel/en/guix.html#Channel-Authentication
https://doi.org/10.22152/programming-journal.org/2023/7/1

Reproductibilité

```
alice@laptop$ guix pull -C channels.scm
alice@laptop$ guix shell -m manifest.scm -- julia
alice@laptop$ guix describe -f channels > channels-pin.scm
```

Il faut partager les paquets **et** la révision de tous les canaux.
(`manifest.scm` **et** `channels-pin.scm`)

```
blake@cluster$ guix time-machine -C channels-pin.scm -- shell -m manifest.scm
```

Les canaux sont des dépôts Git qui peuvent donc être archivés dans Software Heritage (SWH) et Guix va automatiquement y piocher si le serveur du canal a disparu.

- Ajout automatique du code `source` p. ex. `guix lint -c archival julia-csv`
- Ajout manuel du dépôt Git correspondant au canal dans SWH

Pour (re)jouer

https://archive.softwareheritage.org/swh:1:rev:74811b87a1f7c4fba146ba011e96cdf3017c0b07 (lien)

```
(list (channel (name 'mon-canal)
               (url "https://serveur-canal-disparu.org")
               (branch "main")
               (commit "1648b9d39a46fb19678d0c269c38bf93d496bdd4"))
      (channel (name 'guix)
               (url "https://git.guix.gnu.org/guix.git")
               (branch "master")
               (commit "06493e738825598447f5b45d7100ca7eff8b669d")))
```

```
$ guix time-machine -C channels-pin.scm -- build --source julia-csv
```

- Contenu `https://serveur-canal-disparu.org` automatiquement récupéré depuis SWH
- Contenu `https://serveur-source-disparu.org` idem, depuis SWH

Le message

- Empaqueter est un artisanat.
- Et si vous deveniez des artisans ?

autonomie utilisateur

- Il manque un paquet, essayez de l'empaqueter...

...c'est plus « facile » qu'on imagine !

Le message

- Empaqueter est un artisanat.
- Et si vous deveniez des artisans ?

autonomie utilisateur

- Il manque un paquet, essayez de l'empaqueter...

...c'est plus « facile » qu'on imagine !

Empaqueter c'est bien, relire pour fusionner c'est mieux !

Le message

- Empaqueter est un artisanat.
- Et si vous deveniez des artisans ?

autonomie utilisateur

- Il manque un paquet, essayez de l'empaqueter...

...c'est plus « facile » qu'on imagine !

Empaqueter c'est bien, relire pour fusionner c'est mieux !

Ne soyez pas timide.

Le message

- Empaqueter est un artisanat.
- Et si vous deveniez des artisans ?

autonomie utilisateur

- Il manque un paquet, essayez de l'empaqueter...

...c'est plus « facile » qu'on imagine !

Empaqueter c'est bien, relire pour fusionner c'est mieux !

Ne soyez pas timide.

Au delà des paquets, un projet collaboratif entre humains

En pratique

– Vous avez empaqueter pour vous et ça fait le boulout… chouette !
– Ça vit dans canal… chouette !
– Trop timide ?

1. je ne vise personne, quoique ;-)

En pratique

– Vous avez empaqueter pour vous et ça fait le boulout… chouette !
– Ça vit dans canal… chouette !
– Trop timide ?

1. Mentionner l'URL dans Mattermost
2. Ouvrir un ticket *issue* avec l'URL sur le canal `guix-science`
3. Soumettre *pull request*

1. je ne vise personne, quoique ;-)

En pratique

– Vous avez empaqueter pour vous et ça fait le boulout… chouette !
– Ça vit dans canal… chouette !
– Trop timide ?

1. Mentionner l'URL dans Mattermost
2. Ouvrir un ticket *issue* avec l'URL sur le canal `guix-science`
3. Soumettre *pull request*

- Moins vous considérez vos paquets "prêts" à la soumission parce que *bof ? !*
  Plus vous devriez les soumettre !
- Moins vous avez le temps de soumettre un paquet déjà fait
  Plus vous devriez le soumettre !

peut-être que les résolutions de 2026 ouvriront des tickets ? [1]

1. je ne vise personne, quoique ;-)

Des questions ?