À la découverte de Guix : concepts et applications

Café Guix

Alice Brenon

Sommaire

- Pourquoi Guix ?
- Guix en pratique
- Feuille de route

Pourquoi Guix ?

Guix pour moi

Disciplines scientifiques

- Thèse en informatique (2025)
- Linguistique outillée (TXM, Lexicoscope…)
- Traitement Automatique des Langues
- « Humanités Numériques »

Pratiques

- Développement logiciel
- Ingénierie des données
- Admine sys
- Corrections (des projets de code)

Mais c'est quoi, guix ?

Quelques problèmes courants

- Comment pouvoir refaire ses propres calculs ?
- Comment distribuer son code ?
- Comment contrôler finement les dépendances ?
- Comment partager sa config entre plusieurs machines ?

Gestion des paquets

OS

- `apt`, `yum`, `pacman`…
- responsabilité: admin sys
- stabilité

Langages

- `npm`, `gem`, `opam`…
- responsabilité (?): développeur
- flexibilité

Le cas Python

- `pip`, `conda`, `uv`
- environnements virtuels
  - `venv`
  - modification de variables (`PATH`)
- beaucoup d'effets de bord

Déploiement automatique

Conteneurs

- LXC, Docker, ...
- «on met tout»

Automation

- ansible, chef, puppet...
- un `diff` des opérations

Plateformes de calcul

- modules (, `pip`)
- «expériences»
  - backuper ses fichiers
  - tout scripter
  - ...garder trace de la config système SIDUS (Quemener & Taulelle, 2013)

Guix en pratique

Concepts

paquet	recette pour construire un programme
manifest	collection de plusieurs paquets
channel	source de paquets avec leurs versions
profil	«système» partiel disponible dans un répertoire

Système déclaratif

- tout est en guile (scheme (LISP))
- code ↔ données (homoïconicité)
- paquet, OS, services…

```
(operating-system
  (locale "fr_FR.utf8")
  (timezone "Europe/Paris")
  (keyboard-layout fr-bepo-ctrl_swapcaps)
  (host-name "sallow")
  …)
```

Mes commandes préférées

- `search` : trouver un paquet
- `shell` : «essayer» un paquet
- `build` : compiler ou récupérer un paquet
- `system` : mettre à jour, créer des VMs, des liveUSB…

Quelques cas d'usage

Refaire ses calculs

- retrouver son environnement de travail
- `manifest.scm` : `guix shell --export-manifest > manifest.scm`
- `channels.scm` : `guix describe -f channels > channels.scm`

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

comprendre:

```
[guix time-machine --channels=channels.scm]-> avec cette version de guix
[shell -m manifest.scm]-> appliquer cette commande
```

Distribuer son code

- (décrire son paquet avec un `guix.scm` pour les gens qui ont guix)
- créer une archive avec `guix pack`
  - deb, rpm
  - tarball (`-R`/`-RR` + `-S` liens symboliques)
  - ...docker !
- créer un conteneur, une VM (?!) avec `guix system`

Encore plus de cas d'usage !

Contrôler les dépendances

- paquet: code **et** donnée → facilement modifiable
- commandes de paquets (`shell`, `build`, `pack`) : *transformation* (`--help-transform`)
  - utiliser une autre source, une branche, un commit précis
  - changer des flags de compilation, la toolchain C
  - remplacer un paquet partout dans l'arbre de dépendances

Partager sa config

- `guix home` + structure de donnée dédiée
  - fichiers de config (+ variables)
  - paquets
  - services
- appel depuis la ligne de commande ou intégrée à l'`operating-system`
- test hyper faciles : `container`

```
(home-environment
  (packages (list ghc kakoune pandoc …))
  (services
    (append (list bash
                  channels
                  environment-variables
                  history
                  ssh-agent
                  (sway kbd-layout))))
~)
```

Et par rapport à… ?

Pip, conda…

- peu de contrôle des effets de bord (ok d'écrire dans `$HOME`)
- ne gère pas tout l'OS

Docker

- peu inspectable
- peut contenir plus que nécessaire
- `Dockerfile` peut dépendre de l'état initial, de téléchargements

Nix

- plus de paquets que `guix`
- concepts un peu différents : flakes
- Domain Specific Language (vs. guile)
- documentation peu claire
- partenariats préoccupants (mais alternatives : lix, tvix…)

Our awesome sponsors
aws
TII Technology Innovation Institute
flyingcircus.io
GitHub

Bienvenue dans /gnu/store

```
mybox@~: ls /
bin/   dev/  gnu/   lost+found/  proc/  run/  tmp/  var/
boot/  etc/  home/  mnt/         root/  sys/  usr/

mybox@~: ls /gnu/store/
l0xmnm6s71ixv79xgjl328q5z0wr6cs2-kakoune-2026.05.21
p3ff2dl152ydnfmzaxwxxzdygidqan7v-ghc-pandoc-3.7.0.2
zy1f5yvxr9ka6s26p41wzn85k7p40mry-ghc-unix-time-0.4.17
… (vraiment beaucoup)

mybox@~: ls /gnu/store/l0xmnm6s71ixv79xgjl328q5z0wr6cs2-kakoune-2026.05.21/
bin/  etc/  libexec/  share/

mybox@~: ll /gnu/
total 7876
drwxrwxr-t 9604 root guixbuild 8052736 13 oct.  10:15 store/

mybox@~: mount | grep store
/dev/mapper/PARTITIONS-ROOT on /gnu/store type ext4 (ro,noatime)
```

Des «calques» de liens symboliques

```
mybox@~: which pandoc
/home/abrenon/.guix-home/profile/bin/pandoc

mybox@~: ll `which pandoc`
…/bin/pandoc -> /gnu/store/dnwn4k47w3mniqhvhrwlafx26xaswmzl-pandoc-3.7.0.2/bin/pandoc

mybox@~: ll /home/abrenon/.guix-home/profile
/home/abrenon/.guix-home/profile -> /gnu/store/9x449n0h0bs8psfsicg8gdff65lvw8x6-profile

mybox@~: ll /home/abrenon/.guix-home
/home/abrenon/.guix-home -> /gnu/store/m9881bxn0249bv306gxq36n9ngvifn2h-home

mybox@~: ll /etc
asound.conf -> /gnu/store/lj1ks7y607gzmwwxpyh7xr8nf2h2649j-asound.conf
bashrc -> /gnu/store/gw0zr7lkidi8iifxqcp907yc75j3l5jf-bashrc
bashrc.d -> /gnu/store/3smshm54sgsmkjnrlg7p65alsnx446ik-bashrc.d
bluetooth -> /gnu/store/wzdyrka2lpiirp0m5q40f0sjc4n4sg3n-etc-bluetooth
dbus-1 -> /gnu/store/f29h9fi86sn6s04s21rcvp9qljwx4rhg-dbus-configuration
elogind -> /gnu/store/x63n7sn6avpf506pb25mw2xwhz9xxl5l-etc-elogind
…
```

Quelques bonnes propriétés

Accès simultané à des versions différentes

- hash différents, pas de conflits
- chaque paquet utilise celle qu'il a besoin

Partage de dépendances

- pas de recopie inutile (vs. npm)
- moins d'espace disque occupé

Mises à jour «atomiques»

- tout préparer *puis* changer le lien
- `guix system : build` vs. `reconfigure`

Retours en arrière

- possibilité de garder les liens précédents
- différentes options dans GRUB
- `rollback` (ou `--rollback`)
  - `guix system`
  - `guix package`
  - `guix pull`
  - …

Feuille de route

Récemment

Communauté

- 1ère release de Guix-Science !! (07 septembre 2026)
- `time-machine` vers une URL[1]
  - résultat d'une évaluation
  - plus facile pour les démos
  - relève les contraintes de *tout* contrôler

[1] https://guix.gnu.org/en/blog/2026/time-travel-without-borders/

Cuirass (Intégration Continue)

- nouvelle version 1.4 !
- beaucoup d'améliorations UX (merci Romain Garbage)
- Cuirass fournit des channels.scm !

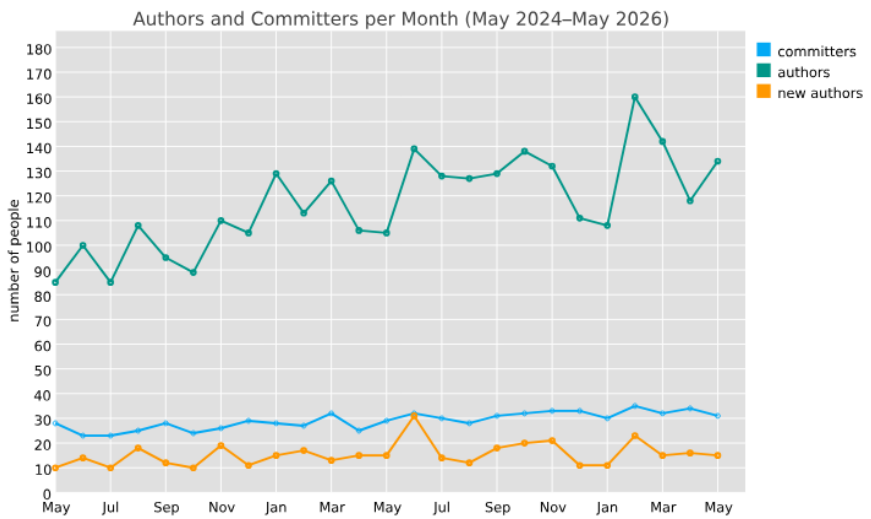
Processus de développement

Historiquement

- plateforme principale: courriel `<guix-devel@gnu.org>`
  - discussions
  - patches
- discussion mais pas de mécanisme de prise de décision collective
- demandes fortes de la communauté pour des interfaces web / issues / pull-request

Depuis bientôt 2 ans

- décembre 2024
- Guix Consensus Document (GCD)
- GCD 002 : migrer à Codeberg (25 mai 2025)
- GCD 008 : résolution contre l'usage d'IA génératives (finalement pas approuvée mais discussion importante dans la communauté)



À suivre

Quelques critiques actuelles

- lenteur de `guix pull`
- disponibilité des paquets : dans certains commit de guix, certains paquets cassés

Pistes futures

- «évaluations parfaites» (Vivien Kraus)
  - lien avec les `time-machine` dynamiques
  - vers des releases de guix ?
- git vs. libgit2 (Sergio Pastor Pérez)
- avancées régulières sur Hurd

Conclusion

- un gestionnaire d'«environnements»
  - paquets
  - **et** OS
  - même la configuration `user`
- déclaratif
- dans un langage sympa