Guix, une usine à conteneurs / VMs (docker, et autres)

Simon Tournier

Inserm US53 - UAR CNRS 2030
simon.tournier@inserm.fr

28 janvier 2025

https://hpc.guix.info

CaféGuix

Université Paris Cité

Dialogue basé sur des faits réels

Alice dit: Ça fonctionne avec `conda install numpy`.
Comme le recommande la doc, d'ailleurs.

Dialogue basé sur des faits réels

Alice dit: Ça fonctionne avec `conda install numpy`.
Comme le recommande la doc, d'ailleurs.

Blake répond: Attends, j'ai vu le Café Guix du 28/05/2024 (transparents) (vidéo).
La commande « magique » est :

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

et il y a l'option `--container`.

Dialogue basé sur des faits réels

Alice dit: Ça fonctionne avec `conda install numpy`.
Comme le recommande la doc, d'ailleurs.

Blake répond: Attends, j'ai vu le Café Guix du 28/05/2024 (transparents) (vidéo).
La commande « magique » est :

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

et il y a l'option `--container`.

Alice dit: Oui mais je n'ai pas le droit d'installer Guix ma machine.

Dialogue basé sur des faits réels

Alice dit: Ça fonctionne avec `conda install numpy`.
Comme le recommande la doc, d'ailleurs.

Blake répond: Attends, j'ai vu le Café Guix du 28/05/2024 (transparents) (vidéo).
La commande « magique » est :

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

et il y a l'option `--container`.

Alice dit: Oui mais je n'ai pas le droit d'installer Guix ma machine.

Carole suggère: Dans mon ancien labo, c'est Docker.

Dialogue basé sur des faits réels

Alice dit: Ça fonctionne avec `conda install numpy`.
Comme le recommande la doc, d'ailleurs.

Blake répond: Attends, j'ai vu le Café Guix du 28/05/2024 (transparents) (vidéo).
La commande « magique » est :

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

et il y a l'option `--container`.

Alice dit: Oui mais je n'ai pas le droit d'installer Guix ma machine.

Carole suggère: Dans mon ancien labo, c'est Docker.

Dan rétorque: Sur mon infra, c'est Singularity/Apptainer !

Dialogue basé sur des faits réels

Alice dit: Ça fonctionne avec `conda install numpy`.
Comme le recommande la doc, d'ailleurs.

Blake répond: Attends, j'ai vu le Café Guix du 28/05/2024 (transparents) (vidéo).
La commande « magique » est :

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

et il y a l'option `--container`.

Alice dit: Oui mais je n'ai pas le droit d'installer Guix ma machine.

Carole suggère: Dans mon ancien labo, c'est Docker.

Dan rétorque: Sur mon infra, c'est Singularity/Apptainer !

Rendez-vous le 29 avril 2025 pour « **Tout savoir sur la commande** `guix pack` »

Dialogue basé sur des faits réels

Alice dit: Ça fonctionne avec `conda install numpy`.
Comme le recommande la doc, d'ailleurs.

Blake répond: Attends, j'ai vu le Café Guix du 28/05/2024 (transparents) (vidéo).
La commande « magique » est :

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

et il y a l'option `--container`.

Alice dit: Oui mais je n'ai pas le droit d'installer Guix ma machine.

Carole suggère: Dans mon ancien labo, c'est Docker.

Dan rétorque: Sur mon infra, c'est Singularity/Apptainer !

Rendez-vous le 29 avril 2025 pour « **Tout savoir sur la commande** `guix pack` »

Parlons-en quand même un peu aujourd'hui...

Dialogue basé sur des faits réels

Alice dit: Je dois encore produire des Machines Virtuelles (VM).

Dialogue basé sur des faits réels

Alice dit: Je dois encore produire des Machines Virtuelles (VM).

Blake répond: Oui moi aussi. Je les enchaîne...

Dialogue basé sur des faits réels

Alice dit: Je dois encore produire des Machines Virtuelles (VM).

Blake répond: Oui moi aussi. Je les enchaîne...

Carol intervient: Pas facile de voir ce qui est différent à quelques mois d'intervalles.

Dialogue basé sur des faits réels

Alice dit: Je dois encore produire des Machines Virtuelles (VM).

Blake répond: Oui moi aussi. Je les enchaîne...

Carol intervient: Pas facile de voir ce qui est différent à quelques mois d'intervalles.

Produire à la chaîne... Vers une industrialisation ?

Déployer oui, Redéployer aussi !

Ce dont je voudrais vous convaincre

lignes de production
conteneur, VM

Ce dont je voudrais vous convaincre

Guix est une usine (presque) parfaite pour fabriquer des conteneurs et/ou des machines virtuelles

lignes de production
conteneur, VM

Ce dont je voudrais vous convaincre

Guix est une usine (presque) parfaite pour fabriquer des conteneurs et/ou des machines virtuelles

lignes de production conteneur, VM

- Qu'est-ce que Guix ? — *gestionnaire d'environnement computationnel*
- Pourquoi Guix ? — *transparent et reproductible*
- Concrètement, un exemple ?

Pour fixer les idées

Logiciel	code source ou programme *binaire* associé
Paquet	recette pour configurer, construire, installer un logiciel
Dépendance	autre paquet nécessaire
Gestionnaire de paquets	automatisation du processus traitant la recette du paquet (et ses dépendances)
Environnement computationnel	pile de tous les logiciels nécessaires pour la configuration, construction et installation d'une collection de logiciels
Déploiement	environnement computationnel configuré
Redéploiement	déploiement sur une autre infra et plus tard (6 mois, 1 an, plus ?)

Solution(s) / Difficulté(s)

1. gestionnaire de paquets : APT (Debian/Ubuntu), YUM (RedHat), etc.
2. conteneur : Docker, Singularity

Solution(s) / Difficulté(s)

1. gestionnaire de paquets : APT (Debian/Ubuntu), YUM (RedHat), etc.
2. conteneur : Docker, Singularity

Guix = #1 + #2

Solution(s) / Difficulté(s)

1. gestionnaire de paquets : APT (Debian/Ubuntu), YUM (RedHat), etc.
2. conteneur : Docker, Singularity

Guix = #1 + #2

1. Comment réinstaller le même ensemble de paquets ? `apt-get update`
2. Comment inspecter l'image de base ? `FROM debian:stable`

Solution(s) / Difficulté(s)

1. gestionnaire de paquets : APT (Debian/Ubuntu), YUM (RedHat), etc.
2. conteneur : Docker, Singularity

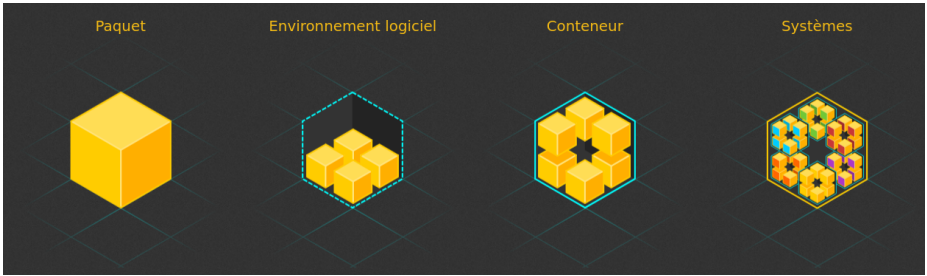
Guix = #1 + #2

1. Comment réinstaller le même ensemble de paquets ? `apt-get update`
2. Comment inspecter l'image de base ? `FROM debian:stable`

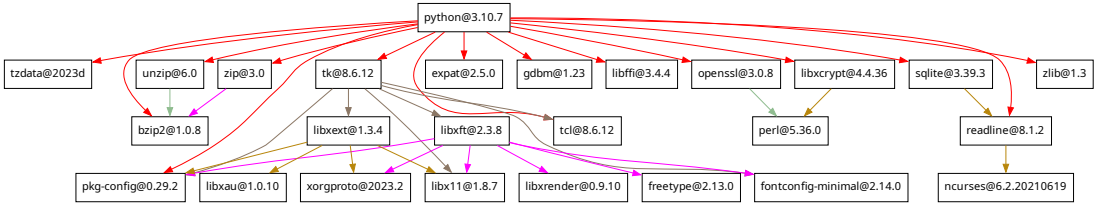
Et si répondre à l'une répondait à l'autre ?

Guix, c'est...

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
...et aussi une bibliothèque Scheme...



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

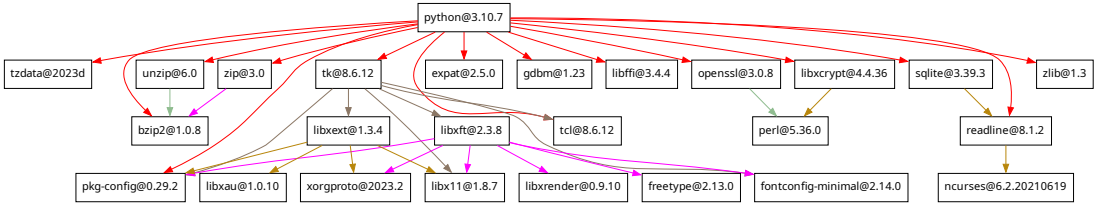


(graphe des dépendances tronqué, sinon 145 nœuds)

Est-ce le même Python 3.10.7 si la version de Perl est 5.34.3 ou 5.36.1 ?

Alice installe Python version 3.10.7

gestionnaire de paquets

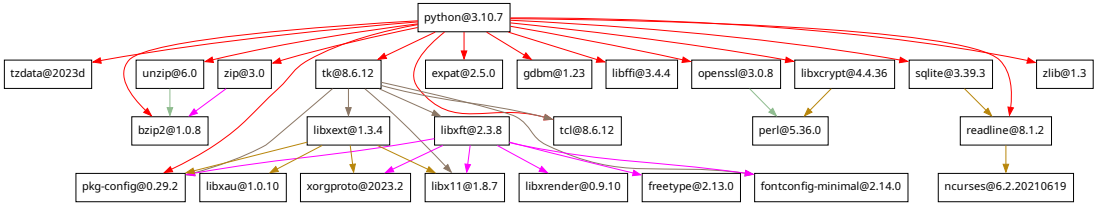


(graphe des dépendances tronqué, sinon 145 nœuds)

Est-ce le même Python 3.10.7 si la version de Perl est 5.34.3 ou 5.36.1 ?
Est-ce le même Python 3.10.7 s'il est compilé avec GCC 11.4.0 ou GCC 10.5.0 ?

Alice installe Python version 3.10.7

gestionnaire de paquets



(graphe des dépendances tronqué, sinon 145 nœuds)

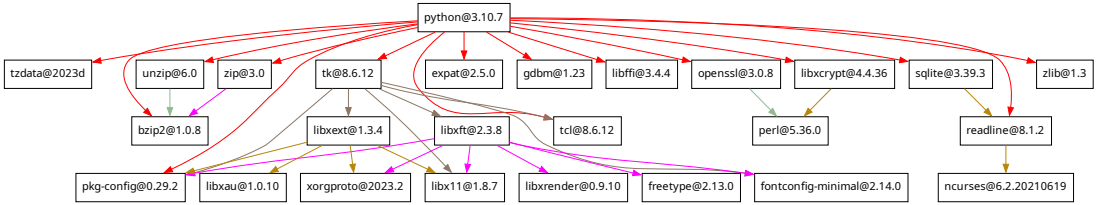
Est-ce le même Python 3.10.7 si la version de Perl est 5.34.3 ou 5.36.1 ?
Est-ce le même Python 3.10.7 s'il est compilé avec GCC 11.4.0 ou GCC 10.5.0 ?

Chaque nœud décrit

- le code source à une version spécifique

Alice installe Python version 3.10.7

gestionnaire de paquets



(graphe des dépendances tronqué, sinon 145 nœuds)

Est-ce le même Python 3.10.7 si la version de Perl est 5.34.3 ou 5.36.1 ?
Est-ce le même Python 3.10.7 s'il est compilé avec GCC 11.4.0 ou GCC 10.5.0 ?

Chaque nœud décrit

- le code source à une version spécifique
- et aussi des options de compilation, parfois des modifications (*patch*) et autres

Définition d'un paquet (nœud du graphe)

```
(define python
  (package
    (name "python")
    (version "3.10.7")
    (source ... )
    (build-system gnu-build-system)
    (arguments ... )
    (inputs (list bzip2 expat gdbm libffi sqlite
                  openssl readline zlib tcl tk))))
```

- Chaque « `inputs` » est une définition similaire (récursion → graphe)
- Il n'y a pas de cycle (`bzip2` ou ses `inputs` ne peuvent pas utiliser `python`)

(Quel sont les nœuds qui enracine le graphe ? Problème du *bootstrap*)

Une histoire de version, en résumé

version = graphe[1]

Guix fournit les outils pour

- Manipuler le graphe
- Inspecter chaque nœud (nœud = paquet)
- Réécrire des nœuds à la volée

[1] « déployer = décrire un graphe »

Une histoire de version, en résumé

version = graphe[1]

Guix fournit les outils pour

- Manipuler le graphe
- Inspecter chaque nœud (nœud = paquet)
- Réécrire des nœuds à la volée

Guix fonctionne par état (révision)

un état fixe toute la collection des paquets et de Guix lui-même

comment capture-t-on un état ?

[1] « déployer = décrire un graphe »

```
$ guix describe
Generation 76 Apr 25 2022 12:44:37 (current)
  guix eb34ff1
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: eb34ff16cc9038880e87e1a58a93331fca37ad92
```

```
$ guix describe
Generation 76 Apr 25 2022 12:44:37 (current)
  guix eb34ff1
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: eb34ff16cc9038880e87e1a58a93331fca37ad92
```

- Il est possible de spécifier un *état* :

```
alice@laptop$  guix describe --format=channels > state.scm
blake@desktop$  guix pull --channels=state.scm
```

```
$ guix describe
Generation 76 Apr 25 2022 12:44:37 (current)
  guix eb34ff1
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: eb34ff16cc9038880e87e1a58a93331fca37ad92
```

- Il est possible de spécifier un *état* :

```
alice@laptop$  guix describe --format=channels > state.scm
blake@desktop$  guix pull --channels=state.scm
```

- Il est possible de se placer **temporairement** dans un état spécifique pour exécuter une commande (comme créer un *pack*)

```
guix time-machine --commit=eb34ff1 -- pack --manifest=software-list.scm
```

Spécification des paquets = fichier manifeste `software-list.scm`

```
(specifications->manifest
  (list "python" "python-numpy"))
```

[2] programmation déclarative

Spécification des paquets = fichier manifeste `software-list.scm`

```
(specifications->manifest
  (list "python" "python-numpy"))
```

```
(use-modules (guix transformations))

(define transform
  (options->transformation
    '((with-c-toolchain . "python=gcc-toolchain@10.5.0"))))

(packages->manifest
 (map (compose transform specification->package)
      (list "python" "python-numpy")))
```

[2] programmation déclarative

Spécification des paquets = fichier manifeste `software-list.scm`

```
(specifications->manifest
  (list "python" "python-numpy"))
```

```
(use-modules (guix transformations))

(define transform
  (options->transformation
    '((with-c-toolchain . "python=gcc-toolchain@10.5.0"))))

(packages->manifest
 (map (compose transform specification->package)
      (list "python" "python-numpy")))
```

Domain-Specific Language (DSL) pour décrire[2] l'environnement computationnel

[2] programmation déclarative

Empaqueter des paquets

Guix est *agnostique* sur le format du « conteneur »

- `tar` (*tarballs*)
- Docker
- Singularity
- paquet binaire Debian `.deb`
- paquet binaire RPM `.rpm`

- archives repositionnables
- sans `Dockerfile`
- via `squashfs`
- sans fichier `debian/rules`
- sans fichier `spec`

Empaqueter des paquets

Guix est *agnostique* sur le format du « conteneur »

- `tar` (*tarballs*)
- Docker
- Singularity
- paquet binaire Debian `.deb`
- paquet binaire RPM `.rpm`

- archives repositionnables
- sans `Dockerfile`
- via `squashfs`
- sans fichier `debian/rules`
- sans fichier `spec`

```
alice@laptop$   guix time-machine -C state.scm -- pack -f docker -m list.scm
blake@desktop$  guix time-machine -C state.scm -- pack -f squashfs -m list.scm
```

Empaqueter des paquets

Guix est *agnostique* sur le format du « conteneur »

- `tar` (*tarballs*)
- Docker
- Singularity
- paquet binaire Debian `.deb`
- paquet binaire RPM `.rpm`

- archives repositionnables
- sans `Dockerfile`
- via `squashfs`
- sans fichier `debian/rules`
- sans fichier `spec`

```
alice@laptop$  guix time-machine -C state.scm -- pack -f docker -m list.scm
blake@desktop$ guix time-machine -C state.scm -- pack -f squashfs -m list.scm
```

- Alice et Blake utilisent les **exacts mêmes** binaires
- Les exacts mêmes binaires sont déployés dans deux formats différents de conteneur

Options spécifiques au format Docker

```
$ guix pack --help-docker-format

      --image-tag=NAME
                         Use the given NAME for the Docker image repository

      -A, --entry-point-argument=COMMAND/PARAMETER
                         Value(s) to use for the Docker ENTRYPOINT arguments.
                         Multiple instances are accepted. This is only valid
                         in conjunction with the --entry-point option

      --max-layers=N
                         Number of image layers
```

Quand `Dockerfile` est plus qu'une liste de paquet...

```
FROM debian:stable
RUN apt-get update && apt-get install python3 python3-numpy
VOLUME vol1 vol2
EXPOSE 8080
```

Quand `Dockerfile` est plus qu'une liste de paquet...

```
FROM debian:stable
RUN apt-get update && apt-get install python3 python3-numpy
VOLUME vol1 vol2
EXPOSE 8080
```

- Difficile de savoir ce que contient `debian:stable`
- Difficile de savoir ce que fait ou fera `apt-get update`
- Difficile d'avoir un variant pour `python3` ou `python3-numpy`

Quand `Dockerfile` est plus qu'une liste de paquet...

```
FROM debian:stable
RUN apt-get update && apt-get install python3 python3-numpy
VOLUME vol1 vol2
EXPOSE 8080
```

- Difficile de savoir ce que contient `debian:stable`
- Difficile de savoir ce que fait ou fera `apt-get update`
- Difficile d'avoir un variant pour `python3` ou `python3-numpy`

- Sauvegarde des fichiers `channels.scm` et `manifest.scm` (*état et liste de paquets*)
- `guix time-machine -C channels.scm -- pack -f docker -m manifest.scm`

```
FROM construit-par-guix
VOLUME vol1 vol2
EXPOSE 8080
```

Inspection en pratique I

Alice J'ai ce conteneur construit en 2021.

```
$ docker images
REPOSITORY            TAG     IMAGE ID       CREATED      SIZE
python-python-numpy   latest  e0a073dfa1ec   51 years ago 431MB
```

Inspection en pratique I

Alice J'ai ce conteneur construit en 2021.

```
$ docker images
REPOSITORY              TAG     IMAGE ID       CREATED      SIZE
python-python-numpy     latest  e0a073dfa1ec   51 years ago 431MB
```

Blake Et qu'est-ce qu'elle contient cette image?

Inspection en pratique I

Alice J'ai ce conteneur construit en 2021.

```
$ docker images
REPOSITORY             TAG     IMAGE ID       CREATED      SIZE
python-python-numpy    latest  e0a073dfa1ec   51 years ago 431MB
```

Blake Et qu'est-ce qu'elle contient cette image?

Alice Regardons...

```
$ docker run -d python-python-numpy:latest python3
$ docker export -o /tmp/re-pack.tar $(docker ps ...)
$ tar -xf /tmp/re-pack.tar $(tar -tf /tmp/re-pack.tar \
                              | grep 'profile/manifest')
$ tree gnu/store
gnu/store
    |__ vdf5c49kzsmdm70134fdgy418ifxd7k-profile
        |__ manifest
```

Inspection en pratique II

Alice Extraction des fichiers `channels.scm` et `manifest.scm`.

```
$ guix package
         -p gnu/store/vdf5c49kzsmdm70134fdgy418ifxd7k-profile \
         --export-channels
$ guix package -p gnu/store/vdf5c49kzsmdm70134fdgy418ifxd7k-profile
               --export-manifest
```

Et partage.

Inspection en pratique II

Alice Extraction des fichiers `channels.scm` et `manifest.scm`.

```
$ guix package
         -p gnu/store/vdf5c49kzsmdm70134fdgy418ifxd7k-profile \
         --export-channels
$ guix package -p gnu/store/vdf5c49kzsmdm70134fdgy418ifxd7k-profile
               --export-manifest
```

Et partage.

Blake Aujourd'hui

```
$ docker load \
    < $(guix time-machine -C channels.scm \
          -- pack -f docker --save-provenance -m manifest.scm)
```

Inspection en pratique II

Alice Extraction des fichiers `channels.scm` et `manifest.scm`.

```
$ guix package
         -p gnu/store/vdf5c49kzsmdm70134fdgy418ifxd7k-profile \
         --export-channels
$ guix package -p gnu/store/vdf5c49kzsmdm70134fdgy418ifxd7k-profile
               --export-manifest
```

Et partage.

Blake Aujourd'hui

```
$ docker load \
    < $(guix time-machine -C channels.scm \
          -- pack -f docker --save-provenance -m manifest.scm)
```

Pourquoi ne pas partager directement le conteneur Docker ?

Inspection en pratique III

Commençons par comparer les deux conteneurs Docker

Alice
```
$ md5sum $(readlink -f gnu/store/vdf5c4...-profile/bin/python3)
c2fc669...  /gnu/store/76sq5...-python-3.8.2/bin/python3.8
```

Blake
```
$ md5sum $(readlink -f gnu/store/ia1sxr...-profile/bin/python3)
c2fc669...  /gnu/store/76sq5...-python-3.8.2/bin/python3.8
```

Inspection en pratique III

Commençons par comparer les deux conteneurs Docker

Alice
```
$ md5sum $(readlink -f gnu/store/vdf5c4...-profile/bin/python3)
c2fc669...  /gnu/store/76sq5...-python-3.8.2/bin/python3.8
```
Blake
```
$ md5sum $(readlink -f gnu/store/ia1sxr...-profile/bin/python3)
c2fc669...  /gnu/store/76sq5...-python-3.8.2/bin/python3.8
```

Les *profils* sont différents.
`vdf5c4` vs `ia1sxr`

Mais les binaires sont exactement les mêmes!

Inspection en pratique III

Commençons par comparer les deux conteneurs Docker

Alice
```
$ md5sum $(readlink -f gnu/store/vdf5c4...-profile/bin/python3)
c2fc669...  /gnu/store/76sq5...-python-3.8.2/bin/python3.8
```
Blake
```
$ md5sum $(readlink -f gnu/store/ia1sxr...-profile/bin/python3)
c2fc669...  /gnu/store/76sq5...-python-3.8.2/bin/python3.8
```

Les *profils* sont différents.
`vdf5c4` vs `ia1sxr`

Mais les binaires sont exactement les mêmes!

Le lecteur attentif de « When Docker images become fixed-point »

– paru le 22 Octobre 2021 –

remarquera que Blake a le même *profil*.

Reproduire à l'identique c'est bien, mais...

Le but d'Alice et Blake est de contrôler leur environnement computationnel.

Reproduire à l'identique c'est bien, mais...

Le but d'Alice et Blake est de contrôler leur environnement computationnel.

- Comment est-ce que la bibliothèque NumPY a-t-elle était construite?
  Quelle options de compilation?
  Et pour la bibliothèque d'algèbre linéaire OpenBLAS?

Reproduire à l'identique c'est bien, mais...

Le but d'Alice et Blake est de contrôler leur environnement computationnel.

- Comment est-ce que la bibliothèque NumPY a-t-elle était construite?
  Quelle options de compilation?
  Et pour la bibliothèque d'algèbre linéaire OpenBLAS?

```
guix time-machine -C channels.scm -- edit python-numpy
guix time-machine -C channels.scm -- build python-numpy --derivation
```

Reproduire à l'identique c'est bien, mais...

Le but d'Alice et Blake est de contrôler leur environnement computationnel.

- Comment est-ce que la bibliothèque NumPY a-t-elle était construite?
  Quelle options de compilation?
  Et pour la bibliothèque d'algèbre linéaire OpenBLAS?

```
guix time-machine -C channels.scm -- edit python-numpy
guix time-machine -C channels.scm -- build python-numpy --derivation
```

- Le projet a évolué, il y a besoin d'autres outils ou des variants...

...même si Alice n'est plus au labo

...ou que personne n'a conservé la recette du conteneur Docker qui "va-bien".

Reproduire à l'identique c'est bien, mais...

Le but d'Alice et Blake est de contrôler leur environnement computationnel.

- Comment est-ce que la bibliothèque NumPY a-t-elle était construite?
  Quelle options de compilation?
  Et pour la bibliothèque d'algèbre linéaire OpenBLAS?

```
guix time-machine -C channels.scm -- edit python-numpy
guix time-machine -C channels.scm -- build python-numpy --derivation
```

- Le projet a évolué, il y a besoin d'autres outils ou des variants...

  ...même si Alice n'est plus au labo

  ...ou que personne n'a conservé la recette du conteneur Docker qui "va-bien".

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

Attention au petit MAIS

La reconstruction d'un fichier `manifest.scm`
à partir d'un fichier `profile/manifest` (option `--export-manifest`)
est toujours correcte.

Attention au petit MAIS

La reconstruction d'un fichier `manifest.scm`
à partir d'un fichier `profile/manifest` (option `--export-manifest`)

est (presque) toujours correcte.

Bug#74015
`--export-manifest` **fails for some transformations**

Attention au petit MAIS

La reconstruction d'un fichier `manifest.scm`
à partir d'un fichier `profile/manifest` (option `--export-manifest`)

est (presque) toujours correcte.

Bug#74015
`--export-manifest` **fails for some transformations**

Le fichier `profile/manifest` sauvegarde le résultat de l'évaluation,
donc quelques **transformations complexes** peuvent passer entre les mailles.

TODO: implémenter le plan décrit dans Bug#74015

Et une machine virtuelle ?

sur le même principe que précédemment

```
guix time-machine -C state.scm -- system image -t docker config.scm
```

Ici, `docker` est un type de *format* parmi d'autres:

`qcow2`, `wsl2`, `efi-raw`, `uncompressed-iso9660`, etc.

Et une machine virtuelle ?

sur le même principe que précédemment

```
guix time-machine -C state.scm -- system image -t docker config.scm
```

Ici, `docker` est un type de *format* parmi d'autres:

`qcow2`, `wsl2`, `efi-raw`, `uncompressed-iso9660`, etc.

`config.scm` est basé sur le même DSL, un exemple ?

Et une machine virtuelle ?

sur le même principe que précédemment

```
guix time-machine -C state.scm -- system image -t docker config.scm
```

Ici, `docker` est un type de *format* parmi d'autres:

`qcow2`, `wsl2`, `efi-raw`, `uncompressed-iso9660`, etc.

```
guix time-machine -C state.scm -- system vm config.scm
```

Gènère un script qui lors de son exécution démarre une VM via QEMU

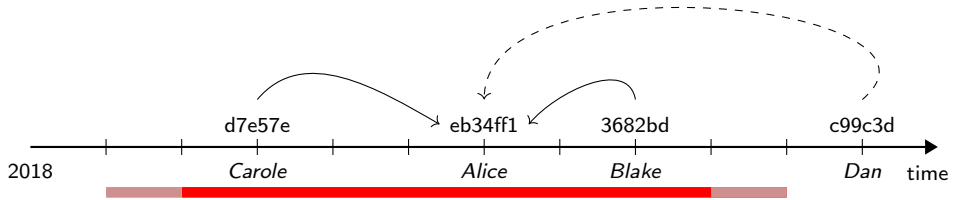
`config.scm` est basé sur le même DSL, un exemple ?

```
(use-modules (gnu))
(use-package-modules python python-xyz)

(operating-system
(host-name "komputilo")
(timezone "Europe/Paris")
(locale "fr_FR.utf8")
(keyboard-layout (keyboard-layout "gb"))

(users (append (list (user-account
                      (name "alice")
                      (group "users")))
               %base-user-accounts))

(packages (append (list python python-numpy)
                  %base-packages)))
```

Ajout de services

```
(define fichiers
  (list "/root/plot"))

(operating-system
  ...
  (services (cons (simple-service 'rotate-mes-trucs
                                  rottlog-service-type
                                  (list (log-rotation
                                         (frequency 'daily)
                                         (files fichiers))))
                  %base-services)))
```

`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)

Nos ami·e·s Grecs Anciens

Mnemosyne
perles
mémoire

Cronus
faux
temps

> Nous ne pouvons pas prédire
> ce que la faux coupera

Perles

- simple rendu facile (simple made easy)
- efficace signifiant robuste
- adressage-par-contenu, identifiant intrinsèque, référence inhérente
- environnement computationnel transparent et auditable
- orientation sur l'autonomie utilisateur

Est-ce que Guix est une perle face à la faux ?

lignes de production

- Conteneur Docker ou Singularity
- Images de base pour `Dockerfile`

- Machine virtuelle
- Image Docker

lignes de production

- Conteneur Docker ou Singularity
- Images de base pour `Dockerfile`

- Machine virtuelle
- Image Docker

`guix time-machine -C state.scm -- `*commande options* `config.scm`

Transparent et vérifiable

lignes de production

- Conteneur Docker ou Singularity
- Images de base pour `Dockerfile`

- Machine virtuelle
- Image Docker

`guix time-machine -C state.scm -- `*commande options* `config.scm`

Transparent et vérifiable

Toujours pas convaincu?

Workshop novembre 2023, Montpellier

- Reproducible virtual machine management with Guix – Yann Dupont
- Reconciling high-performance computing with the use of third-party libraries? – Emmanuel Agullo

Des questions ?

guix-science@gnu.org

https://hpc.guix.info/