Les gestionnaires fonctionnels de paquets Guix et Nix...

...ou le futur des environnements logiciels

ANF UST4HPC - 22/06/2023

Pierre-Antoine Bouttier

D'où je parle

- Ingénieur de recherche, expert en ingénieurie logicielle…
- …directeur adjoint de **l'UAR GRICAD**, basée à Grenoble, fournissant **services, infrastructures et expertise** en soutien à toutes les communautés de recherche grenobloises autour du **calcul scientifique**, du **développement logiciel** et de la **gestion des données de la recherche**.
- Je ne suis pas…
  - …un ASR
  - …un chercheur en informatique

TOC

Parlons reproductibilité numérique

"More than 70% of researchers have tried and failed to reproduce another scientist's experiments, and more than half have failed to reproduce their own experiments."

- 1,500 scientists lift the lid on reproducibility (Nature, 2016) -

Parlons reproductibilité

1905: Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden by A. Einstein

- Un seul auteur, raisonnement principalement verbal
- Un étudiant motivé peut vérifier lui-même l'exactitude des calculs

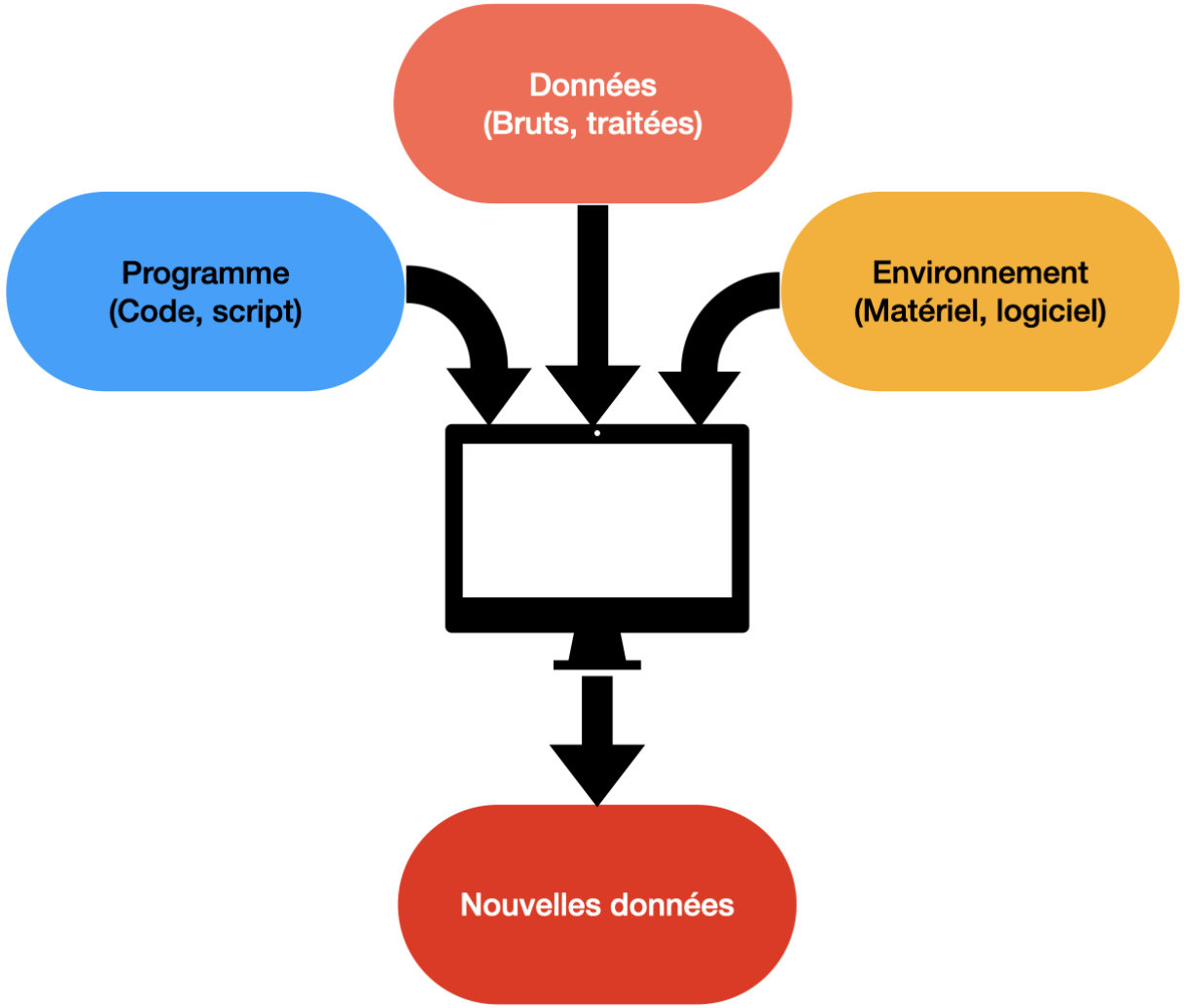
Parlons reproductibilité

2022: Evolutionary-scale prediction of atomic level protein structure with a language model by Z. Zin & al.

- 15 auteurs, références à des logiciels
- “[...] we scale language models from 8 million parameters up to **15 billion parameters.**”
- Code et données semblent disponibles... mais **peut-on réellement vérifier l'exactitude des calculs ?**

Les traitements numériques

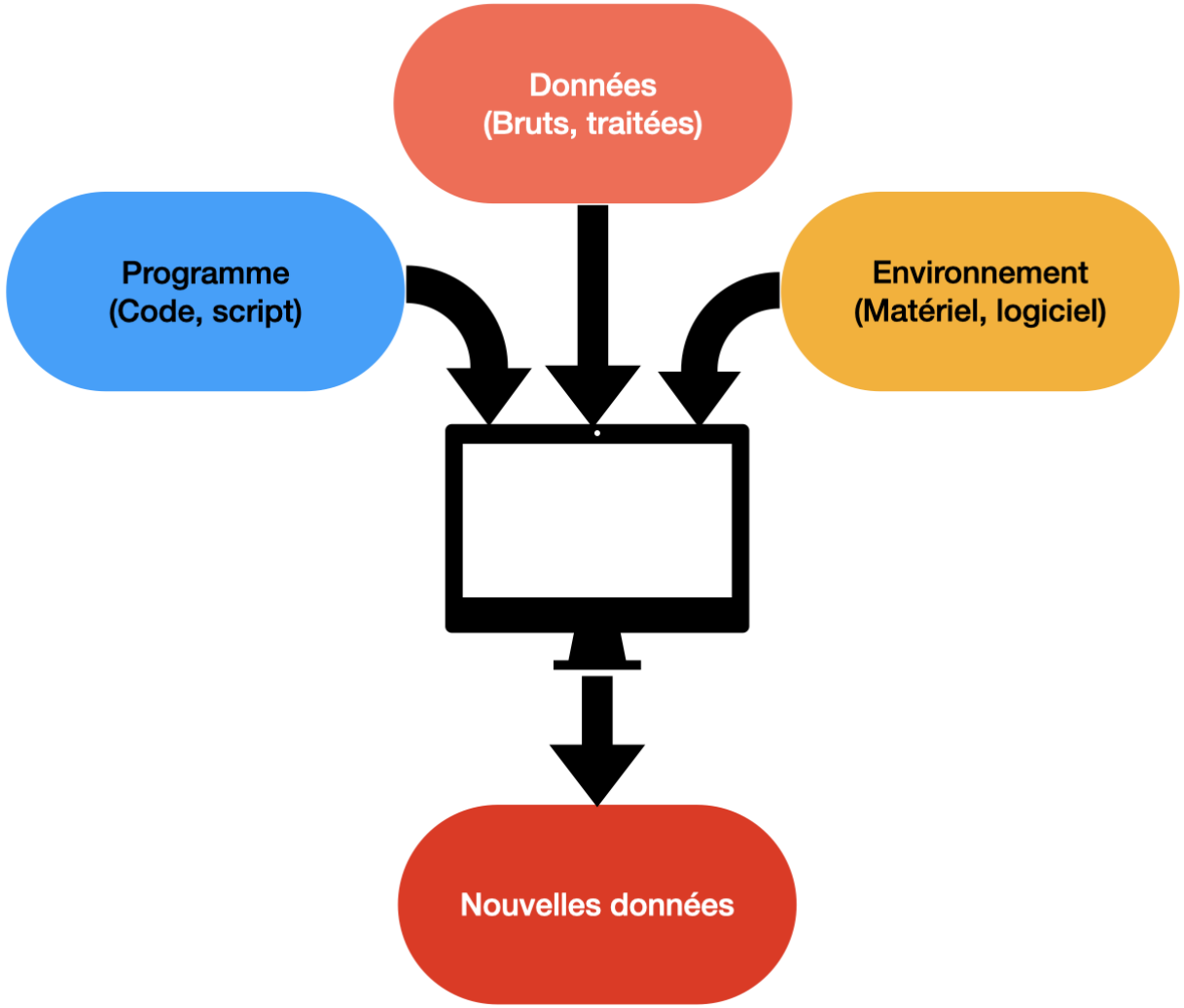
- Une large majorité des résultats scientifiques repose, aujourd'hui, sur un **traitement numérique**
- Un résultat scientifique :
  - Expérience (parfois elle-même numérique)
  - Un **traitement numérique**



Open stuff

La Science Ouverte, une tautologie ?

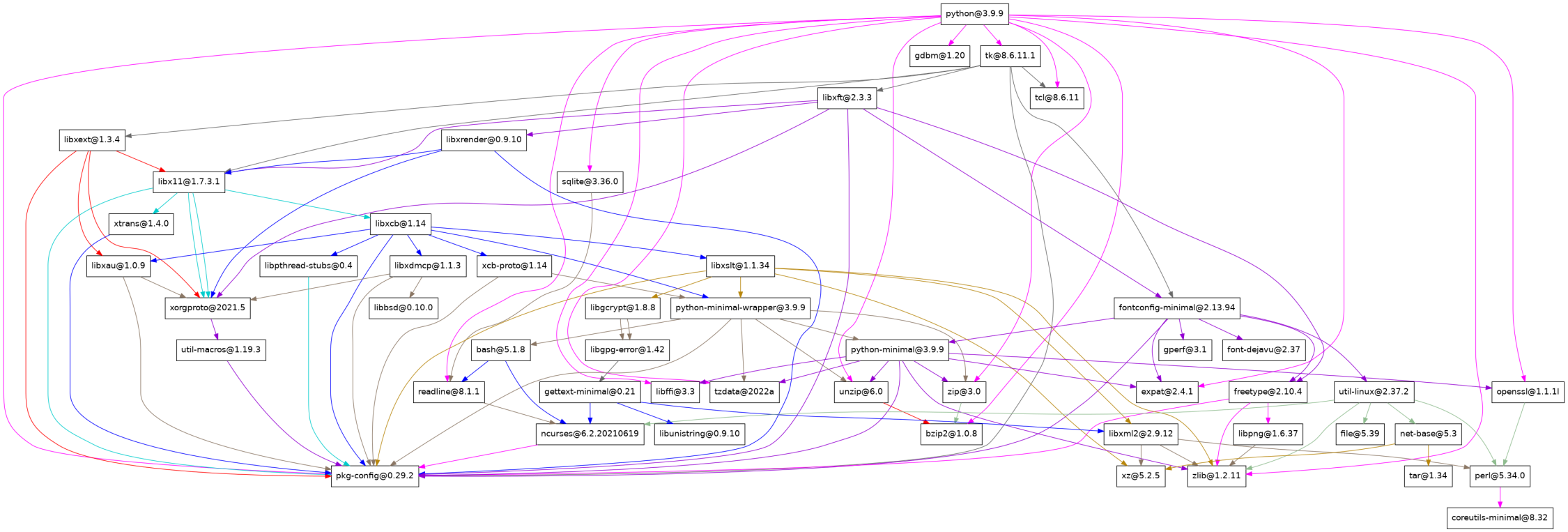
- Les données : Open Data
- Les **programmes** : Open Source
- Les publications : Open Article
- **L'environnement ?**



L'environnement logiciel, une importante source de variabilité

- Nous n'avons qu'un contrôle limité sur l'environnemment matériel
- Reproductibilité, réplicabilité, validation, etc. : l'environnement logiciel comme faiseur de roi

Un programme, vraiment ?



Maîtriser l'environnement logiciel

Contrôler l'environnement logiciel, du moins ses variabilités, revient à maîtriser une machinerie impliquant des millions de rouages, des milliers de concepteurs et dont les plans sont modifiés quotidiennement.

Les outils sont indispensables.

Aujourd'hui, comment fait-on ?

Les gestionnaires d'environnement logiciels sont nombreux :

- Ceux associés à des systèmes d'exploitations : e.g. `apt-get`, `yum`
- Ceux associés à un langage : `pip`, `npm`, etc.
- Les "généralistes" : `spack`, `easybuild`, etc.
- Ceux faits pour la reproductibilité : `nix`, `guix`.

Quelques remarques

- La catégorisation précédente n'est pas stricte : un gestionnaire d'env. log. fait souvent plusieurs choses.
- **Les conteneurs ne gèrent pas un environnement logiciel.**

Welcome to the jungle

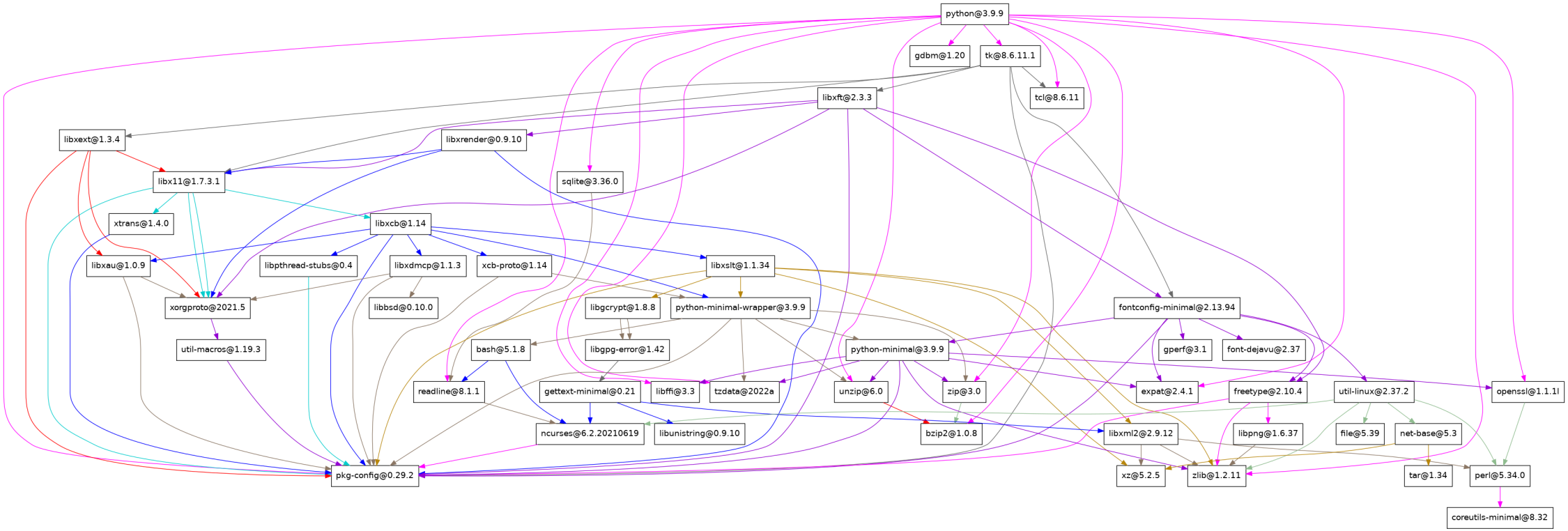
	easybuild	Spack	/ GuixHPC
Build Reproducibility	up to system libs	up to system libs	almost binary
User Environments	module-based	module-based	pure, isolated
Runtime env	module/os-based	module/os-based	static deps
Portability	no	no	yes
Packaging Language	Python/Easyconfig/tcl	Python	Nix
Multiple versions	yes	yes	rebuild/grafts
Binary packages	no	yes	yes
Isolated build env.	no	no	yes
External/Legacy Modules	yes	?	no
Who packages	anyone?	no	anyone
Who builds	admin and user?	admin or user	admin and user on head (daemon+store)
Community/Doc	HPC	HPC	large
Custom Packages	?	?	source-based

Que signifie faits pour la reproductibilité ?

- Indépendance du système hôte à la construction des binaires

Que signifie faits pour la reproductibilité ?

- **Indépendance du système hôte à la construction des binaires**



Que signifie faits pour la reproductibilité ?

- Indépendance du système hôte à la construction des binaires
- **Construction des binaires reproductible**
  - À partir des sources, clairement identifiées...
  - ... suivant des instructions précises...
  - ... pour l'ensemble du graphe de dépendance

Que signifie faits pour la reproductibilité ?

- Indépendance du système hôte à la construction des binaires
- Construction des binaires reproductible
- Indépendance du système hôte à l'exécution des binaires

Corollaires

Un gestionnaire d'env. log reproductible est également :

- Très portable
- Très pratique : **e.g.** multiplicité d'environnement isolé sur un même système

Les gestionnaires fonctionnels de paquets

L'approche des FPMs - les paquets

- Les instructions de construction d'un paquet sont écrites dans un langage fonctionnel (DSL dérivé de Haskell pour Nix, Guile pour Guix).
- Les fichiers source contenant les définitions de paquet sont versionnés dans un dépôt `git`, appelé **channel**
- L'unicité d'un paquet est assurée (et identifiée) par l'état de sa définition et celle de ses dépendances : **n° de commit du channel** (Reproductibilité +1)

Example de paquet Nix

```
{ lib, stdenv, fetchurl }:

stdenv.mkDerivation rec {
    pname = "hello"; version = "2.10";
    src = fetchurl {
        url = "mirror://gnu/hello/${pname}-${version}.tar.gz";
        sha256 = "0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kzl7c9lng89ndq1i";
    };

    doCheck = true;

    meta = with lib; {
        description = "A program that produces a familiar, friendly greeting";
        longDescription = ''
        GNU Hello is a program that prints "Hello, world!" when you run it.
        It is fully customizable. '';
        homepage = "https://www.gnu.org/software/hello/manual/";
        changelog = "https://git.savannah.gnu.org/cgit/hello.git/plain/NEWS?h=v${version}"; license = licenses.gpl3Plus;
        maintainers = [ maintainers.eelco ];
        platforms = platforms.all;
    };
}
```

Exemple de paquet Guix

```
(define-public hello (package
    (name "hello")
    (version "2.10")
    (source (origin
        (method url-fetch)
        (uri (string-append
              "mirror://gnu/hello/hello-" version
              ".tar.gz"))
        (sha256 (base32
            "0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kzl7c9lng89ndq1i"))))
    (build-system gnu-build-system)
    (synopsis "Hello, GNU world: An example GNU package")
    (description "GNU Hello prints the message \"Hello, world!\" and then exits.
    It serves as an example of standard GNU coding practices.
    As such, it supports command-line arguments, multiple languages, and so on.")
    (home-page "https://www.gnu.org/software/hello/")
    (license gpl3+)))
```

Les mécanismes de construction

Sur une machine cible, lorsqu'il y a une demande build (e.g. `guix install hello` ou `nix-env -i hello`) :

- Le build se fait dans un env. complètement isolé du système
- Téléchargement de sorties pré-construites si dispo, construction from scratch sinon
- Les fichiers résultants du build sont stockés dans un seul répertoire en lecture seule contenant l'ensemble des builds (`/nix/store` ou `/gnu/store`)
- Contrôle crypto (avant et après construction)
- En espace utilisateur : liens symboliques dans `$HOME` vers `/*/store` et positionnement des variables d'environnements

Des trucs cools

- **Garantie de forte reproductibilité**, (re-)construction avec variation
- **Génération très simple** de VMs, conteneurs, ramdisk, archives…
- Des environnements d'exécution isolés du système (`shell`)
- **Envergures techniques et applicatives plus vastes que les approches comme Conda, Spack, Flatpack, Snap, etc.**
- Réelle activité communautaire
- Lien à double sens avec **Software Heritage**
- etc.

Retour d'expérience de Guix à GRICAD

À propos de nos serveurs de calculs

- Centre de calcul avec 3 calculateurs reliés entre eux par une grille
- Environnements logiciels déployés avec `guix` ...
- ...directement accessible aux utilisateurs (pas de module)

Les besoins

- `guix` : battery-included…
- Mais pas complètement chargée : nécessité de faire des paquets
  - Logiciels pas encore empaquetés (il y en a !)
  - Besoins spécifiques de versions, d'options de compilations, etc

Les forces en présence

- Quelques centaines d'utilisateurs uniques par an
- Entre 3 et 4 IT à 10% de leur temps pour l'empaquetage `guix`*
- Toutes les communautés et les besoins logiciels associés :
  - HPC traditionnel**
  - Workflows python, R
  - IA
  - Codes communautaires
  - etc.

> * : estimation grossière
> ** : ne signifie pas logiciels simples à empaqueter

L'installation de Guix

- Montages NFS `/gnu/store` et `/var/guix` sur tous les clusters
- Install "classique" depuis la doc de Guix, en root, sur un cluster (`luke`)
- Utilisateurs guixbuilder01 à guixbuilder10 et le groupe guixbuild sur `luke`
- Lancement du démon guix sur `luke` et écoute en tcp depuis les autres clusters.
- Un script à sourcer, côté utilisateur, sur chaque cluster à chaque connexion

Guix, côté utilisateurs

- **Débutants**
  - Pas de problèmes spécifiques, ils suivent la doc.
- **Historiques**
  - Adaptation par rapport à module (assez rapide)
  - Mais quelques soucis pour certains softs…
- **Power users**
  - Prise en main OK (et souvent enthousiaste)
  - Appropriations de l'outil (`guix shell` , `manifest.scm` , `guix time-machine` & `guix describe` , etc.)

Guix, côté support

Les débuts

- Liberté des utilisateurs = moins de demandes d'installation d'environnement…
- …mais demandes quand même !
- Mise au niveau un peu longue :
  - Adapation au langage (DSL pour Nix, Guile pour Guix)
  - Gestion des logiciels propriétaires

Et maintenant ?

Après montée en compétence, plein de bénéfices et aucun regret !

- Grande base de paquets existants
- `guix shell`, `guix pack`, options de transformation de paquets…
- Portabilité
- Reproductibilité accentuée pour nos utilisateurs

Cons

- Demande de la ressource et de l'ingéniosité pour certaines suites logicielles
  - Logiciels propriétaires (e.g. Intel OneAPI)
  - Frameworks complexes (PyTorch, TensorFlow)
- Communauté HPC encore modeste (mais active et croissante)

La communauté Guix-HPC

Regroupement d'utilisateurs ou mainteneurs de Guix pour le HPC.

- Retour et partage d'expérience
- Partage de paquets !
- Animation et dissémination autour de l'utilisation de Guix et de ses possibilités
- Des besoins qui se transforment en features !

- SAVE THE DATE : Guix-HPC days, 8-10 novembre, Montpellier

Conclusion

- D'un point de vue purement **exploitation**, **nous n'avons aucune envie de changer d'outil !**
- **Guix** (ou Nix) reste **le meilleur outil** pour inscrire pleinement nos utilisateurs dans le cadre de **la Science Ouverte**
- **On peut aller encore plus loin dans la vie communautaire** autour de GUIX pour le rendre encore plus accessible

Merci de votre attention

Place à la pratique