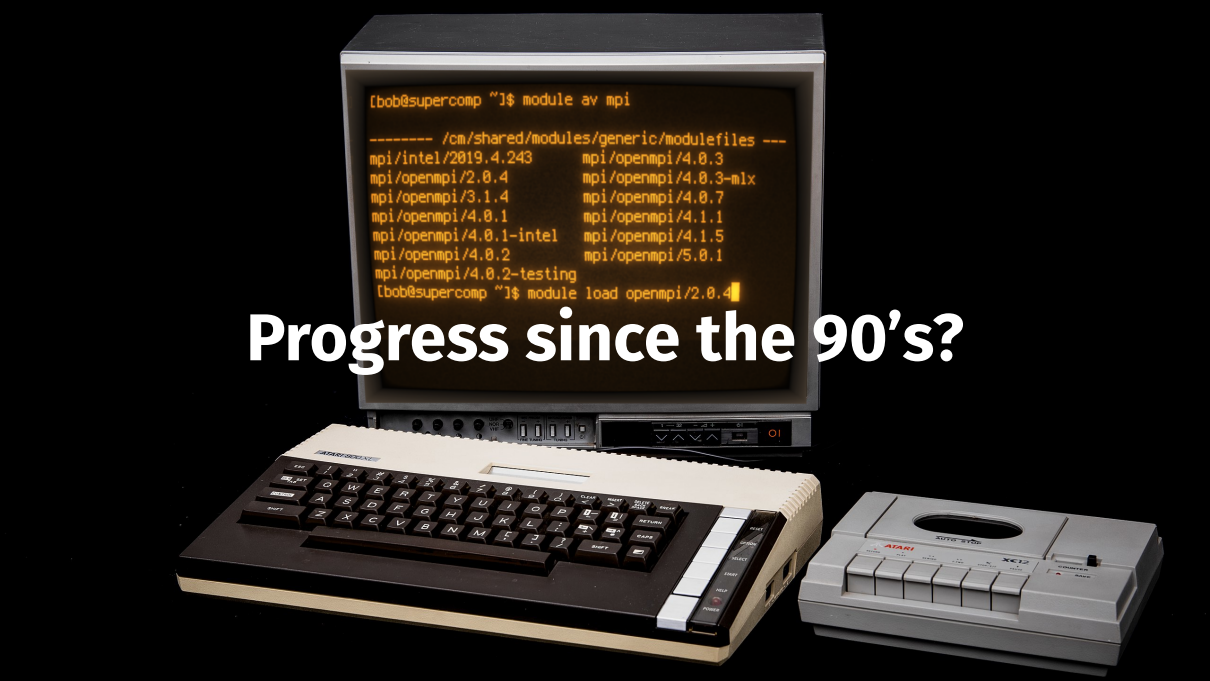
Reproducible software deployment with GNU Guix

Ludovic Courtès

ADAC, Portability, Sustainability & Integration WG Seminar

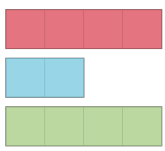
10 January 2025

Inria


[bob@supercomp ~]$ module av mpi
-------- /cm/shared/modules/generic/modulefiles ---
mpi/intel/2019.4.243
mpi/openmpi/2.0.4
mpi/openmpi/3.1.4
mpi/openmpi/4.0.1
mpi/openmpi/4.0.1-intel
mpi/openmpi/4.0.2
mpi/openmpi/4.0.2-testing
mpi/openmpi/4.0.3
mpi/openmpi/4.0.3-mlx
mpi/openmpi/4.0.7
mpi/openmpi/4.1.1
mpi/openmpi/4.1.5
mpi/openmpi/5.0.1
[bob@supercomp ~]$ module load openmpi/2.0.4
Progress since the 90's?
ATARI

Spack

CONDA



208 Open ✓ 308 Closed Author Labels Projects Milestones Assignee Sort

Installation issue: xfd build-error
#11526 opened 18 hours ago by huqy

Installation issue: openmpi (any version) on mac build-error
#11515 opened 4 days ago by luca-heltai

Could not install elfutils build-error
#11501 opened 5 days ago by jczhang07

Installation issue: mumps (serial), error "/bin/sh: line 0: fc: -h: invalid option" build-error
#11498 opened 5 days ago by samfux84

Spack points to incorrect cray-libsci in LANL environment build-error 4
#11491 opened 6 days ago by floquet

Installation issue: range-v3 build-error 2
#11481 opened 6 days ago by chissg

Installation issue: boost build-error 1
#11467 opened 7 days ago by abc19899

A Probabilistic Approach To Selecting Build Configurations in Package Managers

Daniel Nichols
Department of Computer Science, University of Maryland
College Park, Maryland, USA
dnicho@umd.edu

Harshitha Menon
Lawrence Livermore National Laboratory
Livermore, California, USA
harshitha@llnl.gov

Todd Gamblin
Lawrence Livermore National Laboratory
Livermore, California, USA
tgamblin@llnl.gov

Abhinav Bhatele
Department of Computer Science, University of Maryland
College Park, Maryland, USA
bhatele@cs.umd.edu

https://doi.org/10.1109/SC41406.2024.00090

A Pr…ns in

stantly update and test. In this paper we propose a methodology to make use of historical build results in selecting the version for package dependencies. Our method utilizes the flexibility of the Spack package manager's heavily parameterized package configurations to incorporate a machine learning model trained to predict the probability of build outcomes. This work is able to build and install packages with a 13% higher success rate than the default version selection mechanism in Spack.

Departm…

…ryland

tgamblin@llnl.gov

bhatele@cs.umd.edu

https://doi.org/10.1109/SC41406.2024.00090

- Guix started in **2012**
- tools for **reproducible software deployment**
- runs standalone (Guix System) or atop a Linux distro
- **50,000+ packages**
- **100+ monthly contributors**

https://hpc.guix.info

- Guix started in **2012**
- tools for **reproducible software deployment**
- runs standalone (Guix System) or atop a Linux distro
- **50,000+ packages**
- **100+ monthly contributors**
- **Guix-HPC effort (Inria, MDC, UBC, UTHCS) started in 2017**

https://hpc.guix.info

- **PlaFRIM** (FR): Inria Bordeaux (3,000+ cores)
- **GriCAD** (FR): Grenoble (1,000+ cores)
- **GLICID** (FR): Nantes (4,000+ cores)
- **Grid'5000/SLICES** (FR): 8 sites (12,000+ cores)
- **Max Delbrück Center** (DE): 250-node cluster + workstations
- **UMC Utrecht** (NL): 68-node cluster (1,000+ cores)
- ...

5 Inria centers

9 CEA packages

CNRS, univ., ...

LLNL, Stanford, ...

https://numpex.org/exadi-development-and-integration/

CINES

5 Inria centers

TGCC

9 CEA packages

CNRS, univ., ...

IDRIS

LLNL, Stanford, ...

Tier-2 clusters

https://numpex.org/exadi-development-and-integration/

```
guix install gcc-toolchain openmpi hwloc

source ~/.guix-profile/etc/profile

guix package --roll-back
```

```
guix shell python python-numpy
```

```
guix shell python python-numpy \
  -- python3 -c 'import numpy'
```

```
guix shell -D petsc git
```

```
guix shell -D petsc git --container
```

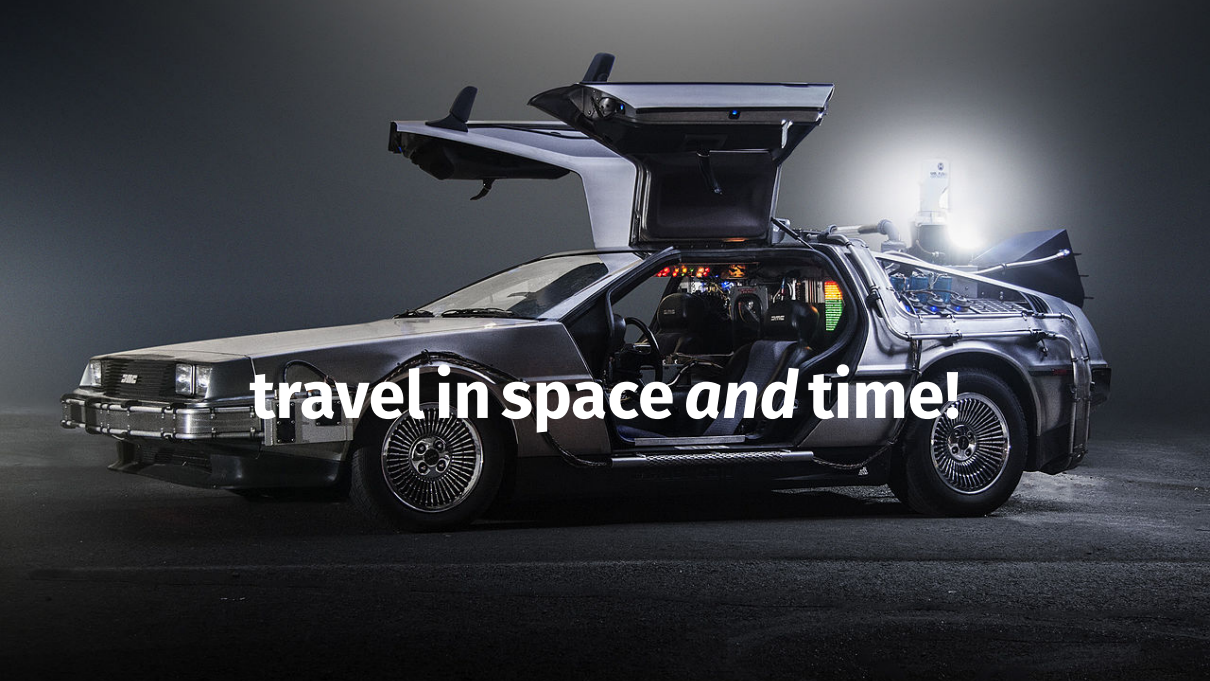
```
guix shell --manifest=manifest.scm -C
```

```
(specifications->manifest
  '("gcc-toolchain" "coreutils" "grep" "sed"
    "openmpi" "openblas" "petsc"))
```

```
bob@laptop$ guix shell --manifest=manifest.scm
bob@laptop$ guix describe
  guix cabba9e
    repository URL: https://git.sv.gnu.org/git/guix.git
    commit: cabba9e15900d20927c1f69c6c87d7d2a62040fe
```

```
bob@laptop$ guix shell --manifest=manifest.scm
bob@laptop$ guix describe
  guix cabba9e
    repository URL: https://git.sv.gnu.org/git/guix.git
    commit: cabba9e15900d20927c1f69c6c87d7d2a62040fe
```

```
alice@supercomp$ guix pull --commit=cabba9e
alice@supercomp$ guix shell --manifest=manifest.scm
```


travel in space *and* time!

Reproducible environments: 2 files, 2 commands

1. `guix describe -f channels > channels.scm`
2. ```
   guix time-machine -C channels.scm -- \
     shell -m manifest.scm
   ```
   ```

- P. Swartvagher, *On the Interactions between HPC Task-based Runtime Systems and Communication Libraries*, PhD thesis, Dec. 2022
- M. Felšöci, *Fast Solvers for High-Frequency Aeroacoustics*, PhD thesis, Feb. 2023
- N. Vallet *et al.*, *Toward practical transparent verifiable and long-term reproducible research using Guix*, Nature Scientific Data, Oct. 2022

Interoperability.

```
$ guix pack \
      python python-numpy python-scipy
…
/gnu/store/…-pack.tar.gz
```

```
$ guix pack --relocatable \
      python python-numpy python-scipy
…
/gnu/store/…-pack.tar.gz
```

https://hpc.guix.info/blog/2020/05/faster-relocatable-packs-with-fakechroot/

```
$ guix pack --format=squashfs \
      python python-numpy python-scipy
…
/gnu/store/…-singularity-image.tar.gz
```

```
$ guix pack --format=docker \
      python python-numpy python-scipy
…
/gnu/store/…-docker-image.tar.gz
```

```
guix module create -o /opt/modules \
  gcc-toolchain openmpi netcdf gromacs
```

https://hpc.guix.info/blog/2022/05/back-to-the-future-modules-for-guix-packages/

```
guix module create -o /opt/modules \
  --manifest=manifest.scm
```

https://hpc.guix.info/blog/2022/05/back-to-the-future-modules-for-guix-packages/

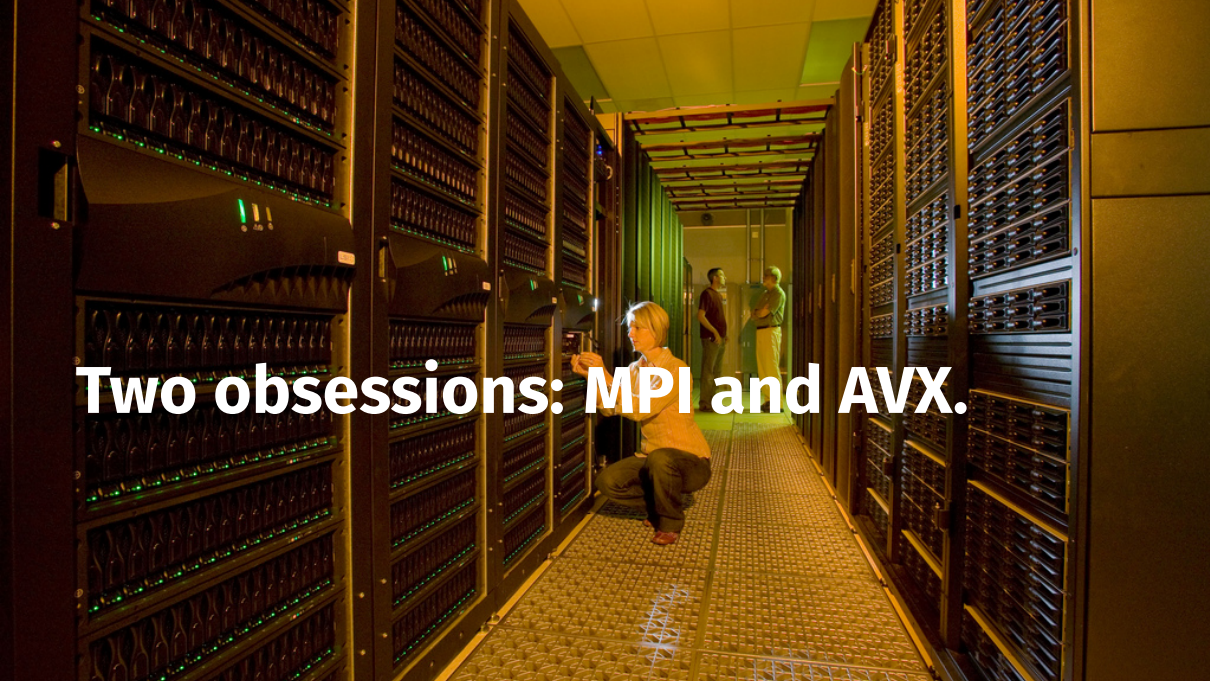

Custom packages

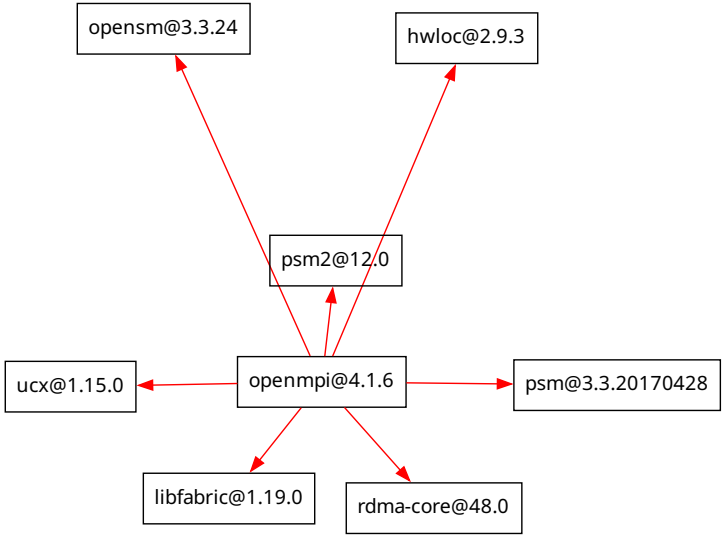
```
guix pack -f squashfs hwloc \
  --with-source=./hwloc-3.0.0rc1.tar.gz


guix shell intel-mpi-benchmarks \
  --with-input=openmpi=mpich
```

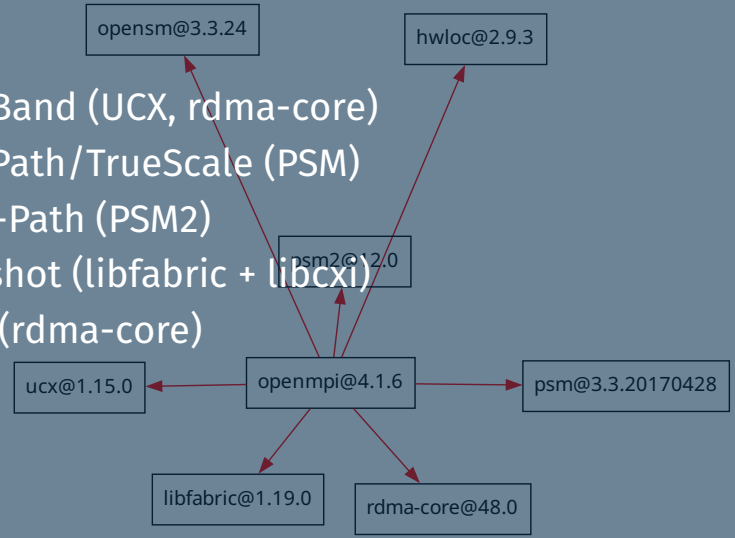
- `--with-commit`
- `--with-branch`
- `--with-patch`
- `--with-input`
- `--with-c-toolchain`
- `--with-graft`
- …

https://guix.gnu.org/manual/en/html_node/Package-Transformation-Options.html

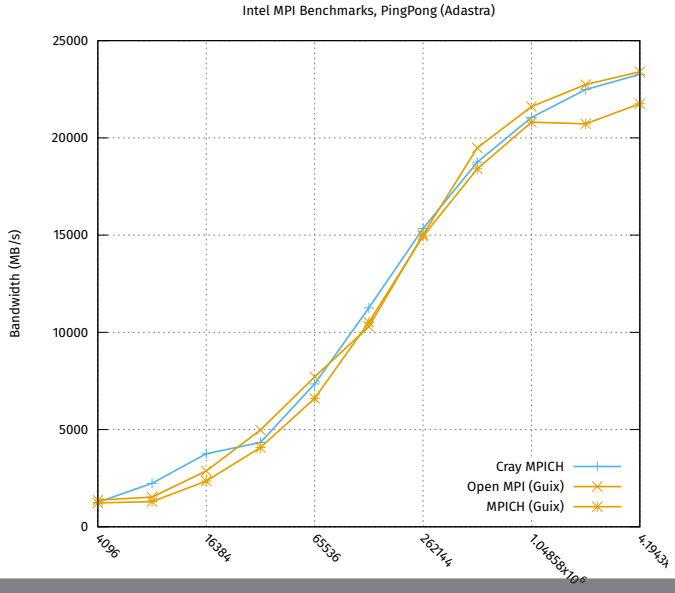

Two obsessions: MPI and AVX.


opensm@3.3.24
hwloc@2.9.3
psm2@12.0
ucx@1.15.0
openmpi@4.1.6
psm@3.3.20170428
libfabric@1.19.0
rdma-core@48.0

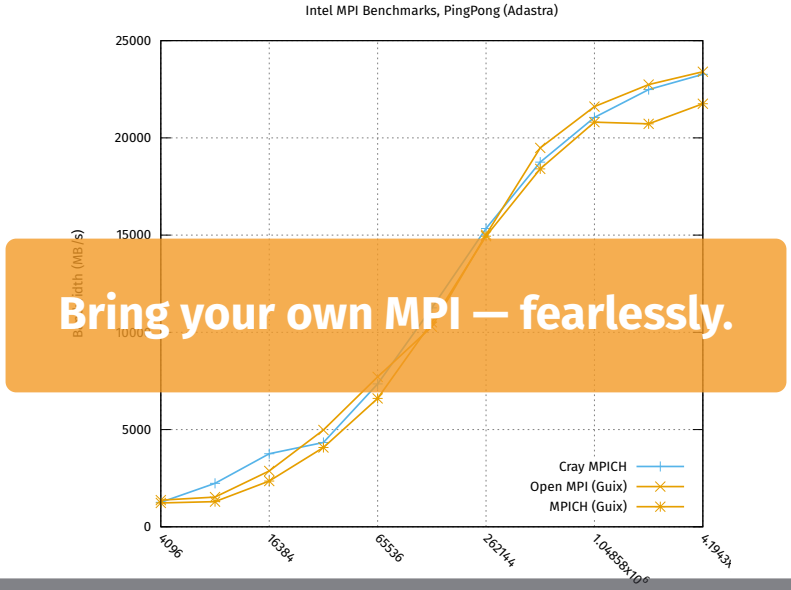
- InfiniBand (UCX, rdma-core)
- InfiniPath/TrueScale (PSM)
- Omni-Path (PSM2)
- Slingshot (libfabric + libcxi)
- RoCE (rdma-core)
- ...



https://hpc.guix.info/blog/2019/12/optimized-and-portable-open-mpi-packaging/



https://hpc.guix.info/blog/2024/11/targeting-the-crayhpe-slingshot-interconnect/



Bring your own MPI — fearlessly.

https://hpc.guix.info/blog/2024/11/targeting-the-crayhpe-slingshot-interconnect/

Design Considerations for Building and Running Containerized MPI Applications

Joshua Hursey
IBM
Rochester, MN, USA
jhursey@us.ibm.com

***Abstract*—Packaging parallel applications in containers has become increasingly popular on High Performance Computing (HPC) systems. These applications depend on the Message Passing Interface (MPI) for communication between processes in their parallel jobs. This paper explores the design considerations that container maintainers must weigh when building and running their applications on HPC systems. This includes highlighting and**

- How close should the library versions inside my container match those on the host system?
- What namespaces and capabilities does MPI care the most about when passing messages both on-node and between nodes?

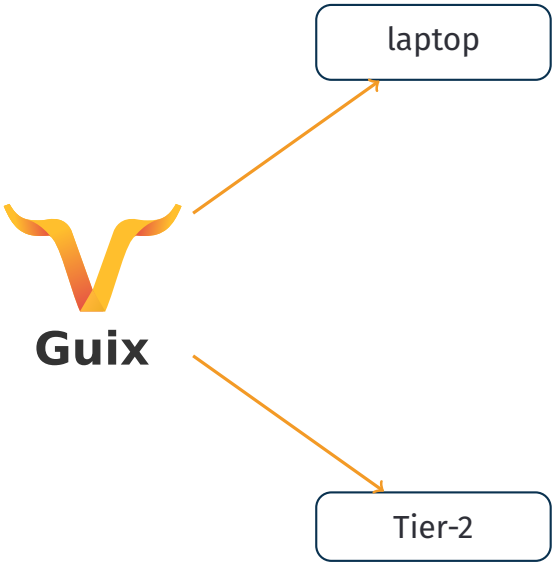
The goal of this paper is to explore each of these questions

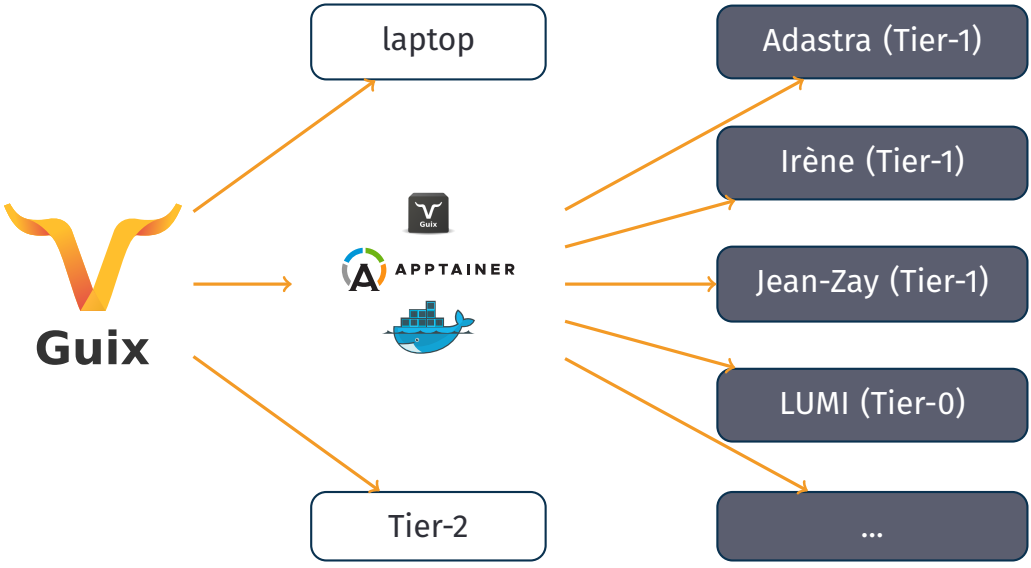
https://doi.org/10.1109/CANOPIEHPC51917.2020.00010

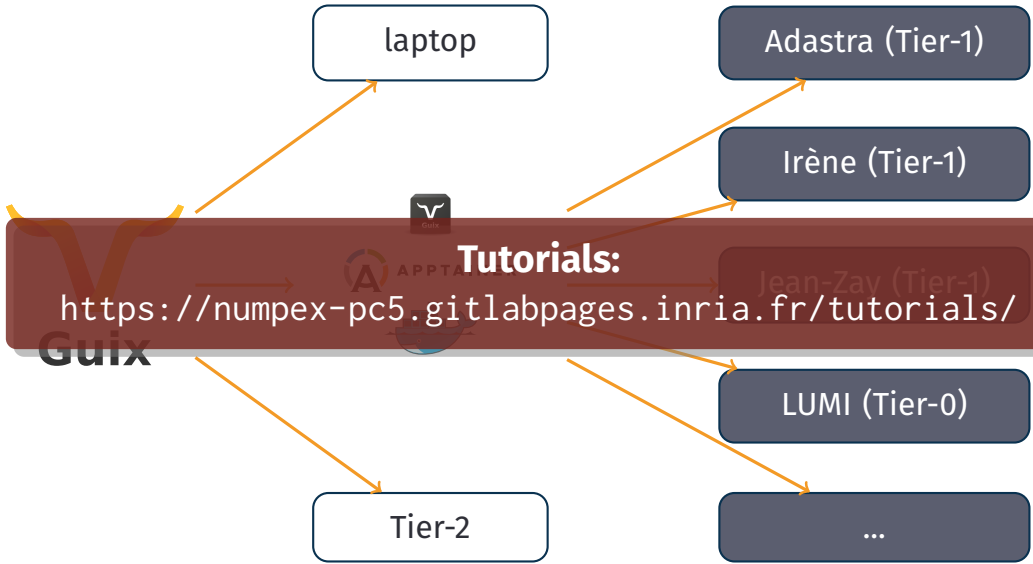
```
laptop$ guix pack -R -S /bin=bin \
      intel-mpi-benchmarks
```

```
laptop$ guix pack -R -S /bin=bin \
      intel-mpi-benchmarks
```

```
supercomp$ tar xf pack.tar.gz
supercomp$ unset LD_LIBRARY_PATH
supercomp$ ./bin/srun -A ... ./bin/IMB-MPI1
```


laptop
Guix
Tier-2


laptop
Guix
Guix
APPTAINER
Tier-2
Adastra (Tier-1)
Irène (Tier-1)
Jean-Zay (Tier-1)
LUMI (Tier-0)
...


laptop
Adastra (Tier-1)
Irène (Tier-1)
Jean-Zay (Tier-1)
LUMI (Tier-0)
...
Tier-2
Guix
Tutorials:
https://numpex-pc5.gitlabpages.inria.fr/tutorials/

```
$ guix shell eigen-benchmarks -- \
    benchBlasGemm 240 240 240
240 x 240 x 240
cblas: 0.20367 (16.289 GFlops/s)
eigen : 0.285149 (11.635 GFlops/s)
```

```
$ guix shell eigen-benchmarks -- \
    benchBlasGemm 240 240 240
240 x 240 x 240
cblas: 0.20367 (16.289 GFlops/s)
eigen : 0.285149 (11.635 GFlops/s)

$ guix shell --tune eigen-benchmarks -- \
    benchBlasGemm 240 240 240
guix shell: tuning for CPU micro-architecture skylake
240 x 240 x 240
cblas: 0.203131 (16.333 GFlops/s)
eigen : 0.0929638 (35.688 GFlops/s)
```

Package multi-versioning

https://hpc.guix.info/blog/2022/01/tuning-packages-for-a-cpu-micro-architecture/


AMD
ROCm

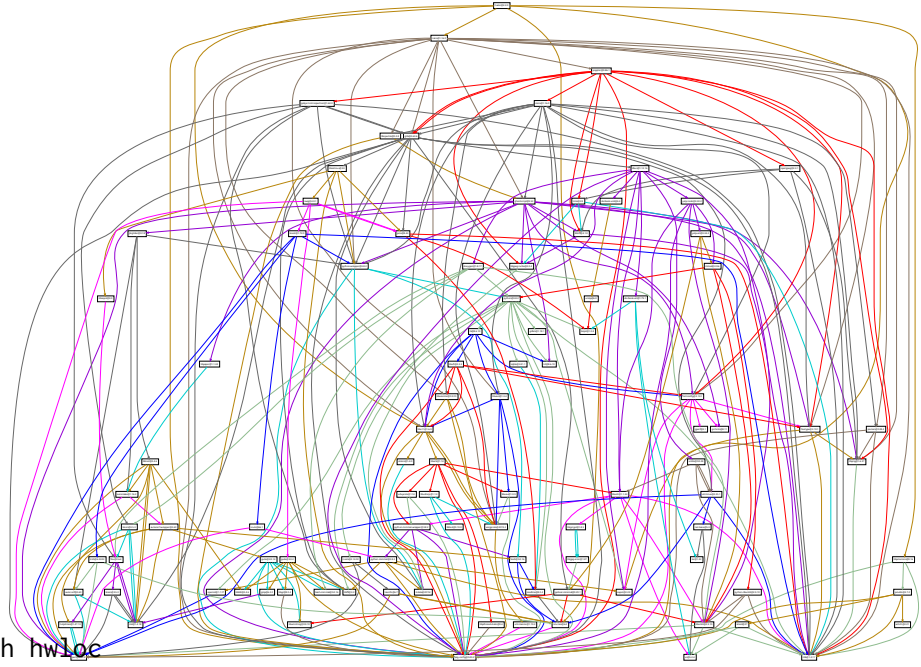
```
laptop$ guix pack -RR hpcg -S /bin=bin
```

```
laptop$ guix pack -RR hpcg -S /bin=bin
```

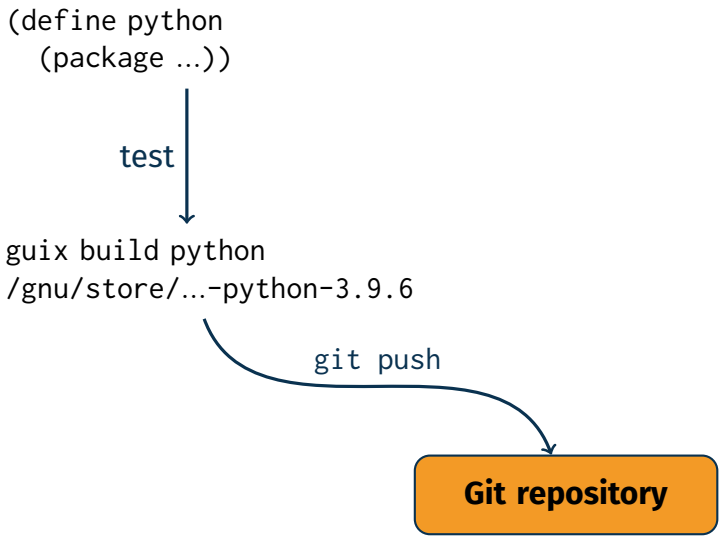
```
adastra$ tar xf pack.tar.gz
adastra$ ./bin/mpirun -n 8 ... \
  ./bin/rochpcg 280 280 280 180
```

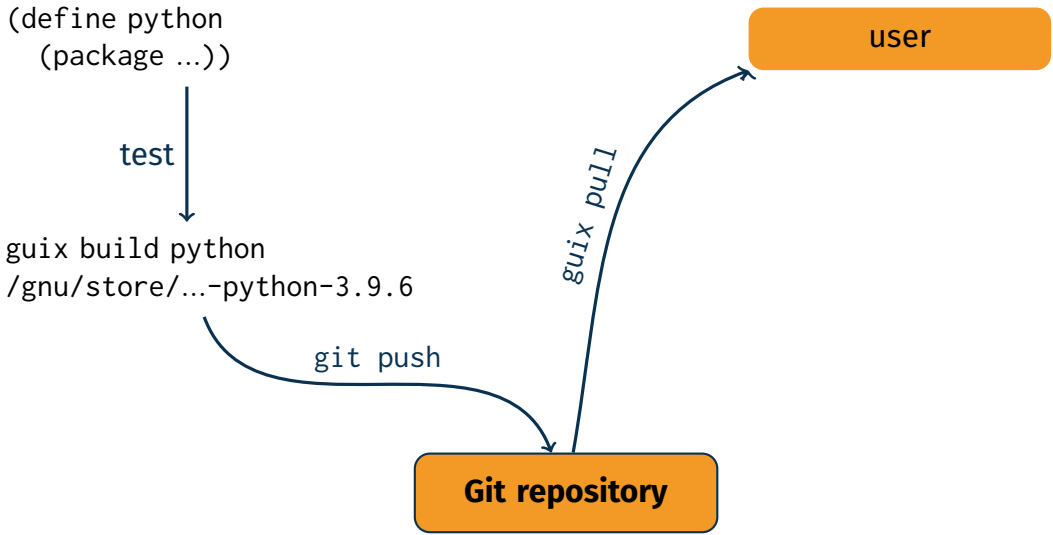
https://hpc.guix.info/blog/2024/01/hip-and-rocm-come-to-guix/

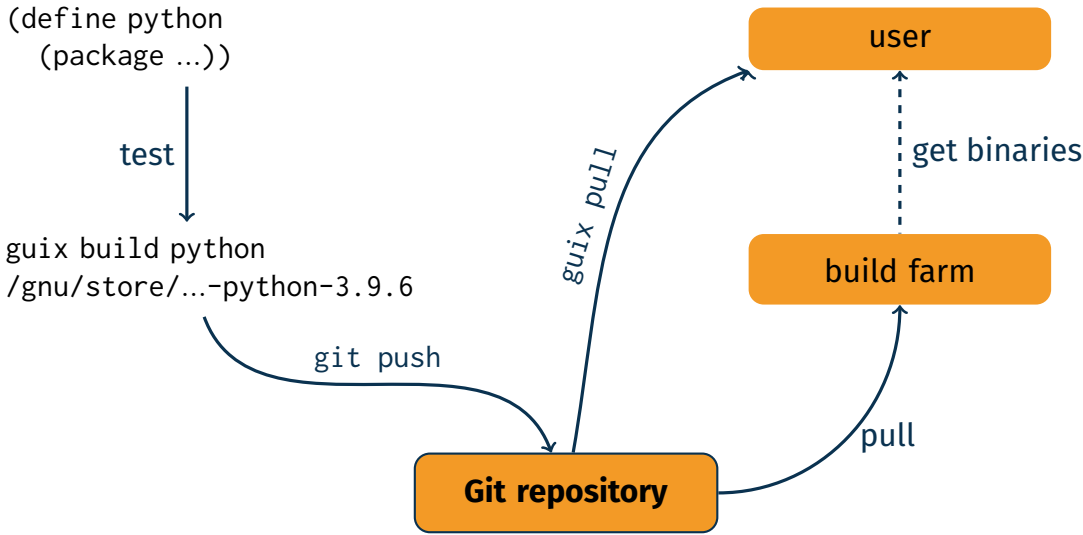
Reproducible deployment as a foundation for robust CI/CD.



guix graph hwloc


(define python
(package ...))
test
guix build python
/gnu/store/...-python-3.9.6
git push
Git repository


(define python
(package ...))
test
guix build python
/gnu/store/...-python-3.9.6
git push
guix pull
user
Git repository


(define python
(package ...))
test
guix build python
/gnu/store/...-python-3.9.6
git push
guix pull
user
get binaries
build farm
pull
Git repository

	Channel	Commit
	https://gitlab.inria.fr/guix-hpc/guix-hpc.git	f002aad7b66cd215b6a1720c48f08ed73f2661d5
⬆	https://git.savannah.gnu.org/git/guix.git	7b0f145802f0c2c785014293d748721678fef824

Builds

All 36 | Scheduled 0 | Succeeded 32 | Failed 4

		ID	Completion time	Job	Name	System
✔	●	714970	1 Jun 18:19 +0200	rochpl.x86_64-linux	rochpl-5.7.1	x86_64-linux
✔	●	714967	1 Jun 18:17 +0200	rocblas.x86_64-linux	rocblas-5.7.1	x86_64-linux
✔	●	714971	27 May 13:48 +0200	rochpl.x86_64-linux	rochpl-5.6.1	x86_64-linux
✔	●	714964	27 May 13:44 +0200	rocblas.x86_64-linux	rocblas-5.6.1	x86_64-linux
✔	●	714973	27 May 11:30 +0200	rochpl.x86_64-linux	rochpl-5.4.4	x86_64-linux
✔	●	714972	27 May 11:06 +0200	rochpl.x86_64-linux	rochpl-5.5.1	x86_64-linux
✔	●	714966	27 May 11:04 +0200	rocblas.x86_64-linux	rocblas-5.5.1	x86_64-linux
✔	⬆	714948	27 May 09:50 +0200	py-melissa-core.x86_64-linux	py-melissa-core-1.0.0-2.d8de4d5	x86_64-linux

https://hpc.guix.info/blog/2023/03/contiguous-integration-and-continuous-delivery-for-hpc/

Build details

Action

Build ID	952555
Evaluation	8092988 (guix-hpc)
Status	✗ Failed
System	x86_64-linux
Name	maphys-1.0.0
Duration	211 seconds
Finished	16 Sep 13:25 +0200
Weather	↓ New failure Channel changes compared to the previous (successful) build: guix `c0d4bd5 → 034eb1b`
Log file	pretty, raw
Derivation	`/gnu/store/kh77cdzsfxmh7h9bbr1nk2vzqw7dqb6b-maphys-1.0.0.drv`
Dependencies	✓ hwloc-2.11.1

Wrapping up.

1. robustness
2. flexibility
3. performance portability
4. interoperability
5. reproducible research

Reproducible deployment
can be achieved
without sacrificing performance.

Let’s add
reproducible deployment
to our best practices book.

ludovic.courtes@inria.fr | @civodul@toot.aquilenet.fr

https://hpc.guix.info

Bonus slides!

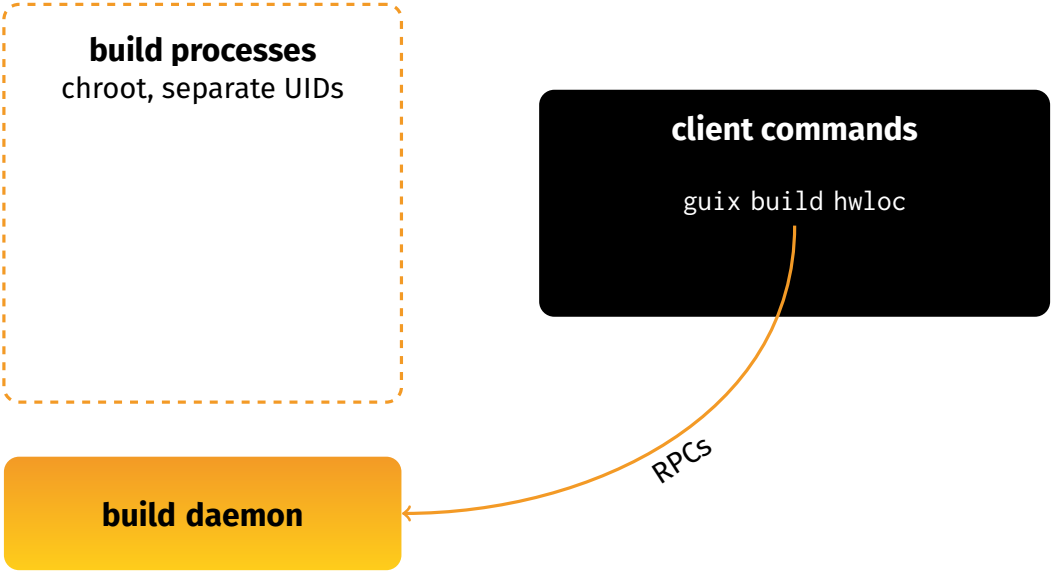
What about admins?

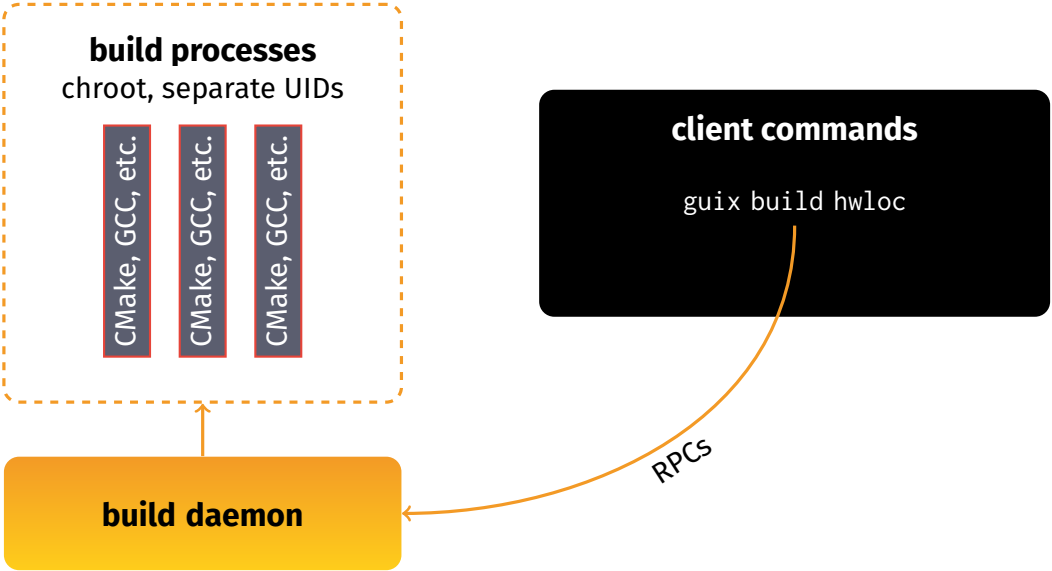
build processes
chroot, separate UIDs

client commands

```
guix build hwloc
```

build daemon


build processes
chroot, separate UIDs
client commands
guix build hwloc
RPCs
build daemon



https://hpc.guix.info/blog/2017/09/reproducibility-and-root-privileges/

common package collections

use & publish binaries

Sharing!

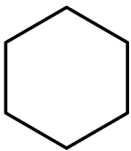
`/etc/guix/channels.scm`

software stored once, deduplicated

remove unused
software

`guix gc`

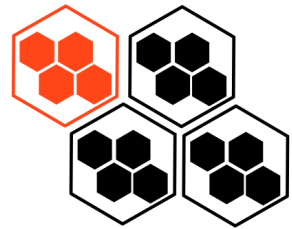
find uses of
vulnerable software



package

environments

containers

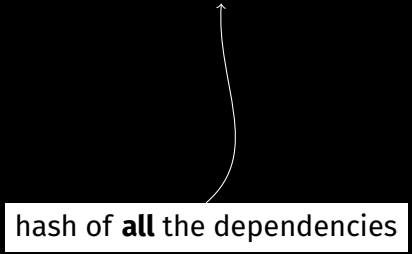


systems

```
$ guix build hwloc
```

isolated build: chroot, separate name spaces, etc.

```
$ guix build hwloc
/gnu/store/ h2g4sf72... -hwloc-1.11.2
```



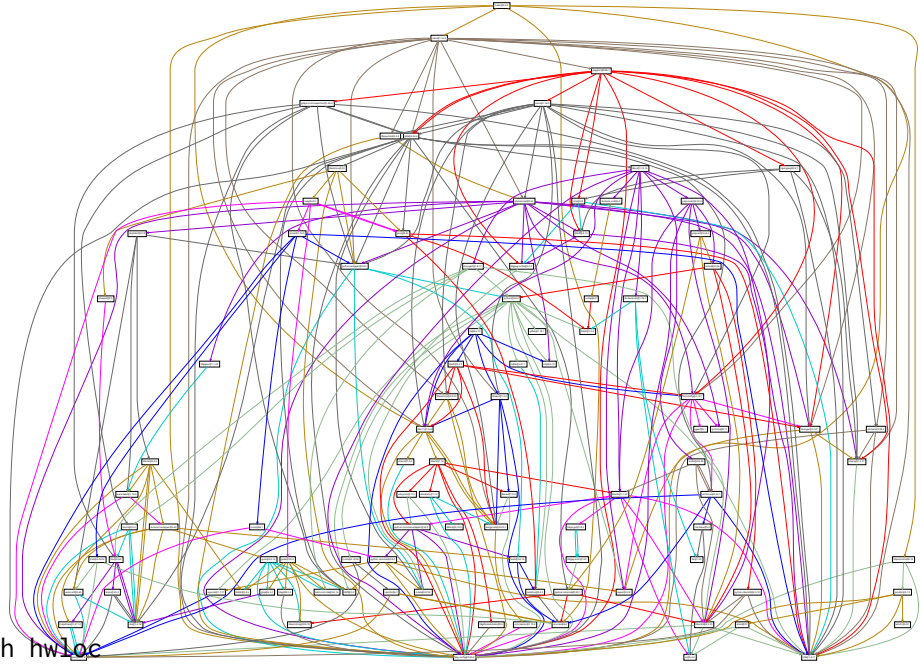
```
$ guix build hwloc
/gnu/store/ h2g4sf72… -hwloc-1.11.2

$ guix gc --references /gnu/store/…-hwloc-1.11.2
/gnu/store/…-glibc-2.24
/gnu/store/…-gcc-4.9.3-lib
/gnu/store/…-hwloc-1.11.2
```

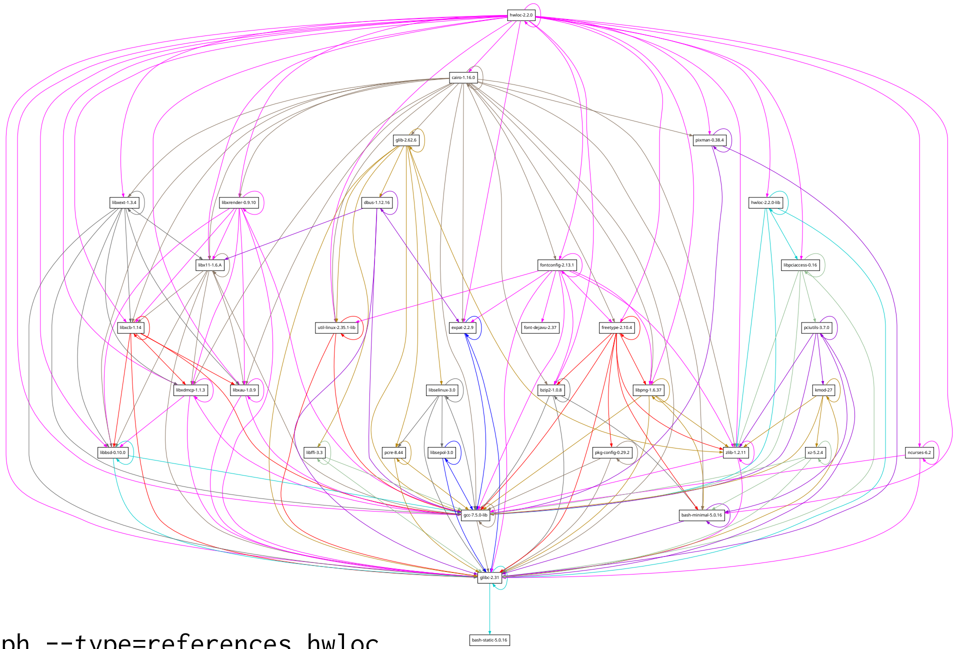
```
$ guix build hwloc
/gnu/store/ h2g4sf72… -hwloc-1.11.2

$ guix gc --references /gnu/store/…-hwloc-1.11.2
/gnu/store/…-glibc-2.24
/gnu/store/…-gcc-4.9.3-lib
/gnu/store/…-hwloc-1.11.2
```

(nearly) bit-identical for everyone



guix graph hwloc



guix graph --type=references hwloc

```
(define pastix
  (package
    (name "pastix")
    (home-page "https://gitlab.inria.fr/solverstack/pastix")
    (source (origin
              (method git-fetch)
              (uri (git-reference
                     (url home-page)
                     (commit "2f30ff07a")
                     (recursive? #t)))
              (sha256
               (base32
                "106rf402cvfdhc2yf...")))))
    ...))
```

```
(define pastix
  (package
    (name "pastix")
    (home-page "https://gitlab.inria.fr/solverstack/pastix")
    (source (origin
              (method git-fetch)
              (uri (git-reference
                     (url home-page)
                     (commit "2f30ff07a")
                     (recursive? #t)))
              (sha256
               (base32
                "106rf402cvfdhc2yf...")))))
    ...))
```

https://www.softwareheritage.org/2019/04/18/software-heritage-and-gnu-guix-join-forces-to-enable-long-term-reproducibility/

```
(operating-system
  (host-name "guixbox")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (bootloader-configuration
                (bootloader grub-efi-bootloader)
                (target "/boot/efi")))
  (file-systems (append (list (file-system
                                (device (file-system-label "my-root"))
                                (mount-point "/")
                                (type "ext4")))
                        %base-file-systems))
  (users (append (list (user-account
                         (name "charlie")
                         (group "users")
                         (home-directory "/home/charlie")))
                 %base-user-accounts))
  (services (append (list (service dhcp-client-service-type)
                          (service openssh-service-type))
                    %base-services)))
```

```
(operating-system
  (host-name "guixbox")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (bootloader-configuration
                (bootloader grub-efi-bootloader)
                (target "/boot/efi")))
  (file-systems (append (list (file-system
                                (device (file-system-label "my-root"))
                                (mount-point "/")
                                (type "ext4")))
                        %base-file-systems))
  (users (append (list (user-account
                         (name "charlie")
                         (group "users")
                         (home-directory "/home/charlie")))
                 %base-user-accounts))
  (services (append (list (service dhcp-client-service-type)
                          (service openssh-service-type))
                    %base-services)))
```

guix system vm config.scm

```
(operating-system
  (host-name "guixbox")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (bootloader-configuration
                (bootloader grub-efi-bootloader)
                (target "/boot/efi")))
  (file-systems (append (list (file-system
                                (device (file-system-label "my-root"))
                                (mount-point "/")
                                (type "ext4")))
                        %base-file-systems))
  (users (append (list (user-account
                         (name "charlie")
                         (group "users")
                         (home-directory "/home/charlie")))
                 %base-user-accounts))
  (services (append (list (service dhcp-client-service-type)
                          (service openssh-service-type))
                    %base-services)))
```

guix system **docker-image** config.scm

```
(operating-system
  (host-name "guixbox")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (bootloader-configuration
               (bootloader grub-efi-bootloader)
               (target "/boot/efi")))
  (file-systems (append (list (file-system
                               (device (file-system-label "my-root"))
                               (mount-point "/")
                               (type "ext4")))
                        %base-file-systems))
  (users (append (list (user-account
                        (name "charlie")
                        (group "users")
                        (home-directory "/home/charlie")))
                 %base-user-accounts))
  (services (append (list (service dhcp-client-service-type)
                          (service openssh-service-type))
                    %base-services)))
```

guix system **container** config.scm

```
(operating-system
  (host-name "guixbox")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (bootloader-configuration
                (bootloader grub-efi-bootloader)
                (target "/boot/efi")))
  (file-systems (append (list (file-system
                                (device (file-system-label "my-root"))
                                (mount-point "/")
                                (type "ext4")))
                        %base-file-systems))
  (users (append (list (user-account
                         (name "charlie")
                         (group "users")
                         (home-directory "/home/charlie")))
                 %base-user-accounts))
  (services (append (list (service dhcp-client-service-type)
                          (service openssh-service-type))
                    %base-services)))
```

guix system **reconfigure** config.scm

```
(operating-system
  (host-name "guixbox")
  (timezone "Europe/Brussels")
  (locale "fr_BE.utf8")
  (bootloader (bootloader-configuration
                (bootloader grub-efi-bootloader)
                (target "/boot/efi")))
  (file-systems (append (list (file-system
                                (device (file-system-label "my-root"))
                                (mount-point "/")
                                (type "ext4")))
                        %base-file-systems))
  (users (append (list (user-account
                         (name "charlie")
                         (group "users")
                         (home-directory "/home/charlie")))
                 %base-user-accounts))
  (services (append (list (service dhcp-client-service-type)
                          (service openssh-service-type))
                    %base-services)))
```

The next step?

Your first package.

0. **Install** Guix: `https://guix.gnu.org/en/download`
1. Create a **Git repository**—a *channel*
2. **Write** (or generate) a *package definition*
3. **Test** it with `guix build`
4. **Iterate** :-)
5. **Commit**, push, enjoy!

0. **Install** Guix: `https://guix.gnu.org/en/download`
1. Create a **Git repository**—a *channel*
2. **Write** (or generate) a *package definition*
3. **Test** it with `guix build`
4. **Iterate** :-)
5. **Commit**, push, enjoy!
6. (*optional*) **Publish** binaries with `guix publish`

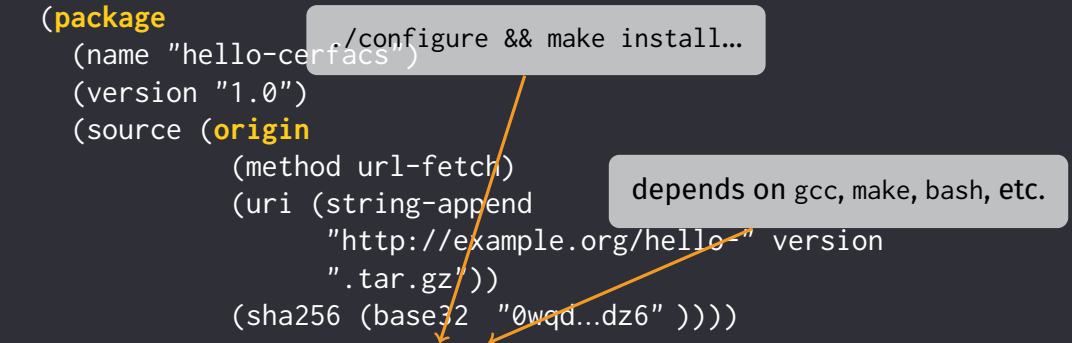
Feeling lucky?

```
guix import pypi my-package > ~/my-def.scm
```

```
(define-public hello-cerfacs
  (package
    (name "hello-cerfacs")
    (version "1.0")
    (source (origin
              (method url-fetch)
              (uri (string-append
                    "http://example.org/hello-" version
                    ".tar.gz"))
              (sha256 (base32  "0wqd...dz6" ))))
    (build-system  gnu-build-system )
    (synopsis "The hello package")
    (description "Greetings, CERFACS!")
    (home-page "https://example.org")
    (license license:gpl3+)))
```

```
(define-public hello-cerfacs
  (package
    (name "hello-cerfacs")
    (version "1.0")
    (source (origin
              (method url-fetch)
              (uri (string-append
                    "http://example.org/hello-" version
                    ".tar.gz"))
              (sha256 (base32  "0wqd...dz6" ))))
    (build-system  gnu-build-system )
    (synopsis "The hello package")
    (description "Greetings, CERFACS!")
    (home-page "https://example.org")
    (license license:gpl3+)))
```

guix hash hello-1.0.tar.gz


(define-public hello-cerfacs
(package
(name "hello-cerfacs")
(version "1.0")
(source (origin
(method url-fetch)
(uri (string-append
"http://example.org/hello-" version
".tar.gz"))
(sha256 (base32 "0wqd...dz6"))))
(build-system gnu-build-system)
(synopsis "The hello package")
(description "Greetings, CERFACS!")
(home-page "https://example.org")
(license license:gpl3+)))
./configure && make install…
depends on gcc, make, bash, etc.

```
(define-public hello-cerfacs
  (package
    (name "hello-cerfacs")
    (version "1.0")
    (source (origin
              (method url-fetch)
              (uri (string-append
                    "http://example.org/hello-" version
                    ".tar.gz"))
              (sha256 (base32  "0wqd…dz6" ))))
    (build-system cmake-build-system)
    (synopsis "The hello package")
    (description "Greetings, CERFACS!")
    (home-page "https://example.org")
    (license license:gpl3+)))
```

```
(define-public hello-cerfacs
  (package
    (name "hello-cerfacs")
    (version "1.0")
    (source (origin
              (method url-fetch)
              (uri (string-append
                    "http://example.org/hello-" version
                    ".tar.gz"))
              (sha256 (base32  "0wqd...dz6" ))))
    (build-system cmake-build-system)
    (inputs (list  rocm-cmake  rocr-runtime ))
    (synopsis "The hello package")
    (description "Greetings, CERFACS!")
    (home-page "https://example.org")
    (license license:gpl3+)))
```

```
(define-public hello-cerfacs
  (package
    (name "hello-cerfacs")
    (version "1.0")
    (source (origin
              (method url-fetch)
              (uri (string-append
                    "http://example.org/hello-" version
                    ".tar.gz"))
              (sha256 (base32  "0wqd...dz6" ))))
    (build-system cmake-build-system)
    (inputs (list  rocm-cmake  rocr-runtime ))
    (synopsis "The hello package")
    (description "Greetings, CERFACS!")
    (home-page "https://example.org")
    (license license:gpl3+)))
```

dependencies

reference to a variable

```
(define-module ( cerfacs )
  #:use-module (guix)
  #:use-module (guix build-system cmake)
  #:use-module ((guix licenses) #:prefix license:))

(define-public hello-cerfacs
  (package
    ...))
```

```
(define-module ( cerfacs )
  #:use-module (guix)
  #:use-module (guix build-system cmake)
  #:use-module ((guix licenses) #:prefix license:))

(define-public hello-cerfacs
  (package
    ...))
```

for a file called `cerfacs.scm`

```
(define-module ( cerfacs hello )
  #:use-module (guix)
  #:use-module (guix build-system cmake)
  #:use-module ((guix licenses) #:prefix license:))

(define-public hello-cerfacs
  (package
    ...))
```

for a file called `cerfacs/hello.scm`

Time to build it!

```
$ guix build -L ~/src/cerfacs hello-cerfacs
```

```
$ guix build -L ~/src/cerfacs hello-cerfacs
ice-9/eval.scm:223:20: In procedure proc:
error: rocm-cmake: unbound variable
hint: Did you forget ‘(use-modules (gnu packages rocm))’?
```

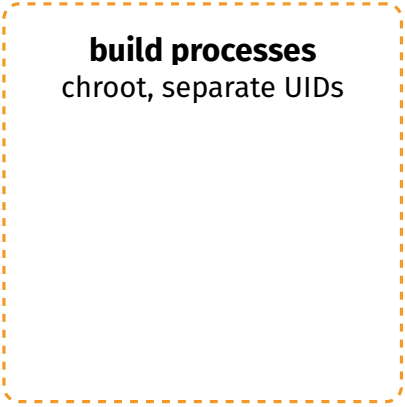
```
$ guix build -L ~/src/cerfacs hello-cerfacs
ice-9/eval.scm:223:20: In procedure proc:
error: rocm-cmake: unbound variable
hint: Did you forget ‘(use-modules (gnu packages rocm))’?
```

Edit file, add `#:use-module (gnu packages rocm)`, save…

```
$ guix build -L ~/src/cerfacs hello-cerfacs
ice-9/eval.scm:223:20: In procedure proc:
error: rocm-cmake: unbound variable
hint: Did you forget ‘(use-modules (gnu packages rocm))’?
```

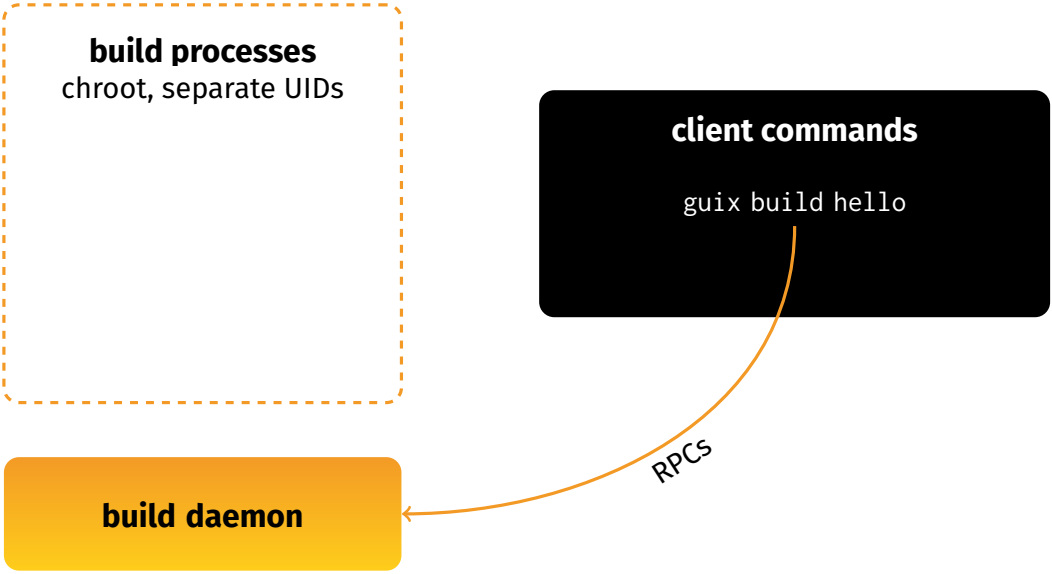
Edit file, add `#:use-module (gnu packages rocm)`, save...

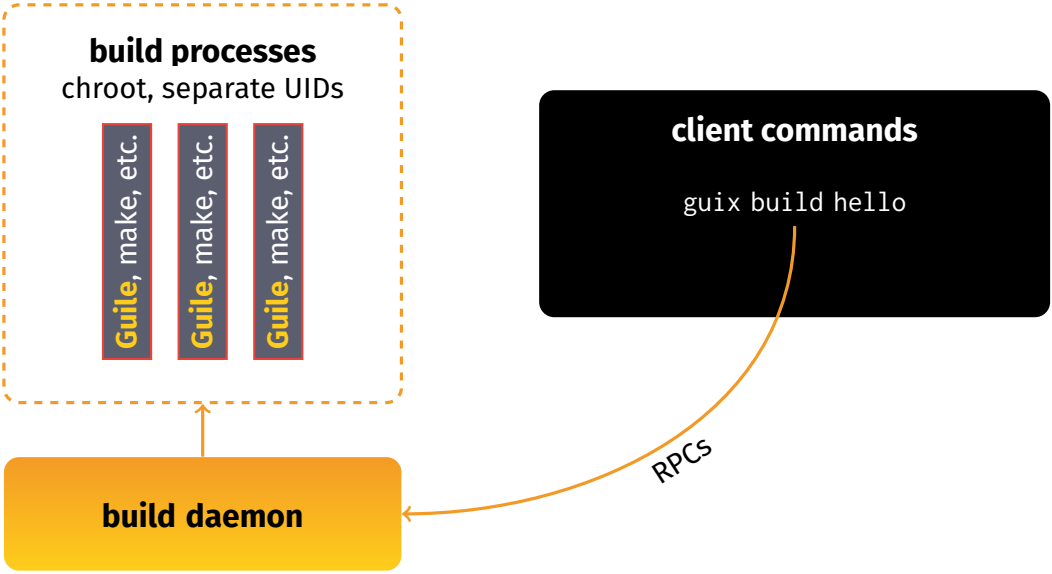
```
$ guix build -L ~/src/cerfacs hello-cerfacs
...
/gnu/store/...-hello-cerfacs-1.0
```


build processes
chroot, separate UIDs


client commands
guix build hello


build daemon


build processes
chroot, separate UIDs
client commands
guix build hello
RPCs
build daemon


build processes
chroot, separate UIDs
Guile, make, etc.
Guile, make, etc.
Guile, make, etc.
client commands
guix build hello
RPCs
build daemon

```
guix install -L ~/src/cerfacs hello-cerfacs

guix shell -L ~/src/cerfacs -D hello-cerfacs

guix pack -f docker -L ~/src/cerfacs hello-cerfacs

...
```

Last steps

- Publish Git repository
- Have users extend `~/.config/guix/channels.scm`:

```
(append (list (channel
                (name 'my-channel)
                (url "https://example.org/my-channel.git")))
        %default-channels)
```

- … and run `guix pull`

Need help?

- https://guix.gnu.org/en/help
- https://guix.gnu.org/manual/devel/en/html_node/Defining-Packages.html