



Reproducible software deployment with Guix in HPC

Ludovic Courtès

CERFACS

6 December 2024

Inria



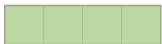
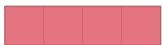
```
[bob@supercomp ~]$ module av mpi
----- /cm/shared/modules/generic/modulefiles ----
mpi/intel/2019.4.243      mpi/openmpi/4.0.3
mpi/openmpi/2.0.4        mpi/openmpi/4.0.3-mlx
mpi/openmpi/3.1.4        mpi/openmpi/4.0.7
mpi/openmpi/4.0.1        mpi/openmpi/4.1.1
mpi/openmpi/4.0.1-intel  mpi/openmpi/4.1.5
mpi/openmpi/4.0.2        mpi/openmpi/5.0.1
mpi/openmpi/4.0.2-testing
[bob@supercomp ~]$ module load openmpi/2.0.4
```

Progress since the 90's?



Spack

CONDA



EASYBUILD.io
building software with ease

🚨 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**

#11491 opened 6 days ago by floquet

🗨️ 4

🚨 **Installation issue: range-v3** **build-error**

#11481 opened 6 days ago by chissg



🗨️ 2

🚨 **Installation issue: boost** **build-error**

#11467 opened 7 days ago by abc19899

🗨️ 1

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

Todd Gamblin

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

Harshitha Menon

Lawrence Livermore National Laboratory
Livermore, California, USA
harshitha@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

Departm

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.

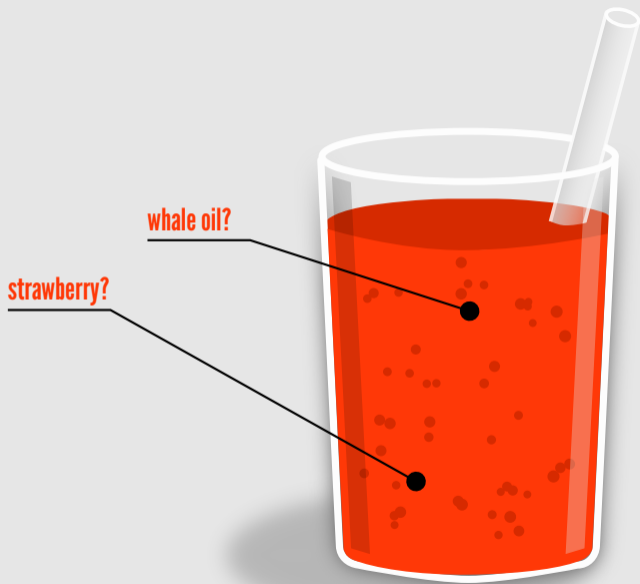
tgamboun@llnl.gov

bhatele@cs.umd.edu

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

Containers to the rescue?





Containers

lack transparency

courtesy of Ricardo Wurmus

Bootstrap: library

From: ubuntu:18.04

%setup

touch /file1

touch \${SINGULARITY_ROOTFS}/file2

%files

/file1

/file1 /opt

%environment

export LISTEN_PORT=12345

export LC_ALL=C

%post

apt-get update && apt-get install -y netcat

NOW=`date`

echo "export NOW=\"\${NOW}\"" >> \$SINGULARITY_ENVIRONMENT

%runscript

echo "Container was created \$NOW"

echo "Arguments received: \$*"

exec echo "\$@"



- ▶ 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** (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



PROGRAMME
DE RECHERCHE

NUMÉRIQUE
POUR L'EXASCALE

CNRS, univ., ...

LLNL, Stanford, ...

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

CINES

5 Inria centers

TGCC

9 CEA packages



PROGRAMME
DE RECHERCHE

NUMÉRIQUE
POUR L'EXASCALE

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 composyx git
```

```
guix shell -D compositum git --container
```

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

```
(specifications->manifest
```

```
  '("gcc-toolchain" "coreutils" "grep" "sed"  
    "openmpi" "openblas" "composyx"))
```

```
$ guix shell -D composyx git \  
  --export-manifest
```

```
$ guix shell -D composyx git \  
  --export-manifest
```

;; What follows is a "manifest" equivalent
;; to the command line you gave.

```
(concatenate-manifests  
  (list (specifications->manifest (list "git"))  
        (package->development-manifest  
          (specification->package "composyx"))))
```

```
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

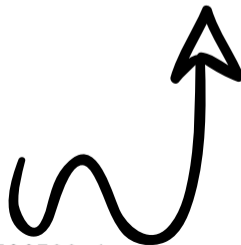
1. `guix describe -f channels > channels.scm`
2. `guix time-machine -C channels.scm -- \`
`shell -m manifest.scm`



Software Heritage



 **Guix**



<https://hal.science/hal-04586520v1>

- ▶ 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
```

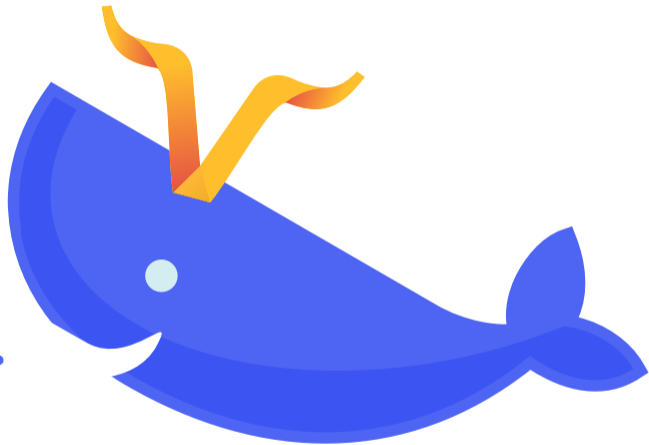
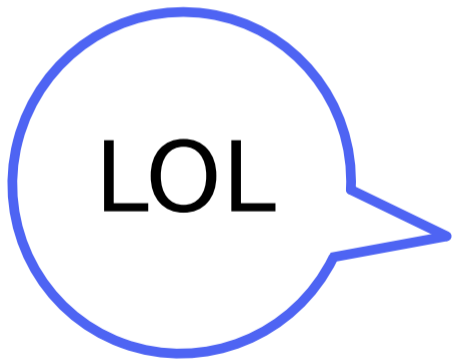
```
$ guix pack --relocatable \  
python python-numpy python-scipy  
...  
/gnu/store/...-pack.tar.gz
```

<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
```

```
guix build composyx \  
  --with-branch=chameleon=devel
```

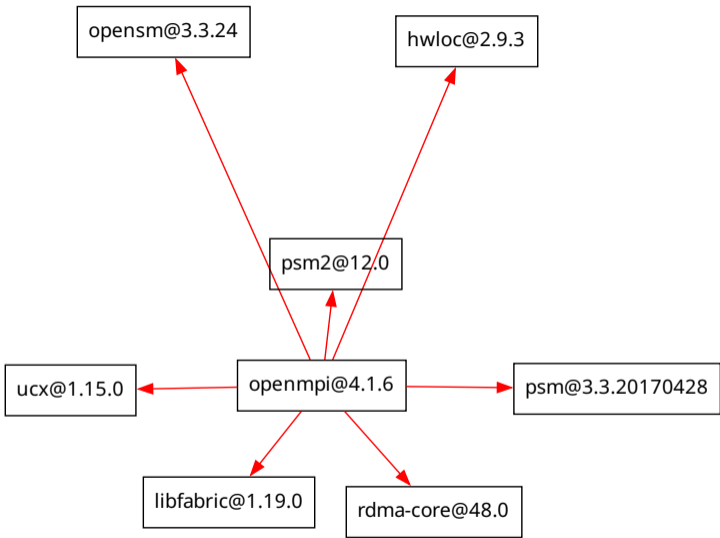
```
guix shell gromacs \  
  --with-latest=python -- \  
  gmx
```

- ▶ `--with-commit`
- ▶ `--with-patch`
- ▶ `--with-input`
- ▶ `--with-c-toolchain`
- ▶ `--with-graft`
- ▶ ...

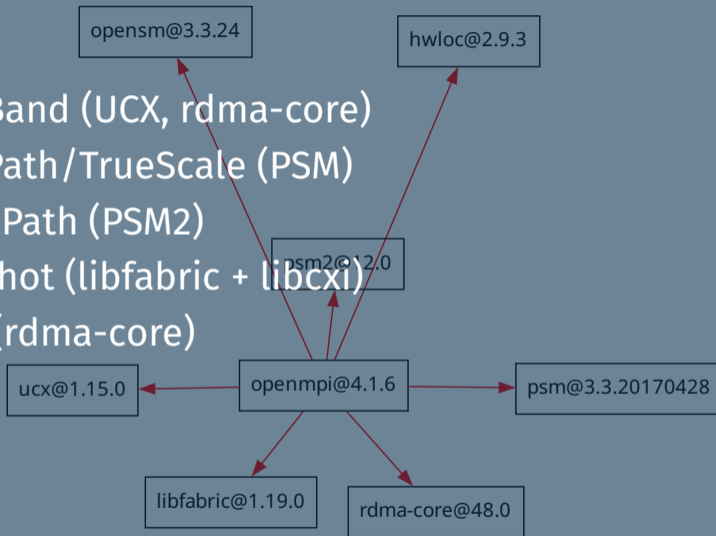
[https://guix.gnu.org/manual/en/html_node/
Package-Transformation-Options.html](https://guix.gnu.org/manual/en/html_node/Package-Transformation-Options.html)

A photograph of a server room aisle. A woman in a light-colored shirt and dark pants is crouching on the metal grating floor, looking at a server rack. In the background, two other people are standing and talking. The room is filled with rows of server racks, and the lighting is a warm, yellowish-green. The text "Two obsessions: MPI and AVX." is overlaid in white, bold font across the center of the image.

Two obsessions: MPI and AVX.



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



```
$ 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)
```

**Package
multi-versioning**

```
$ 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)
```

<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/>

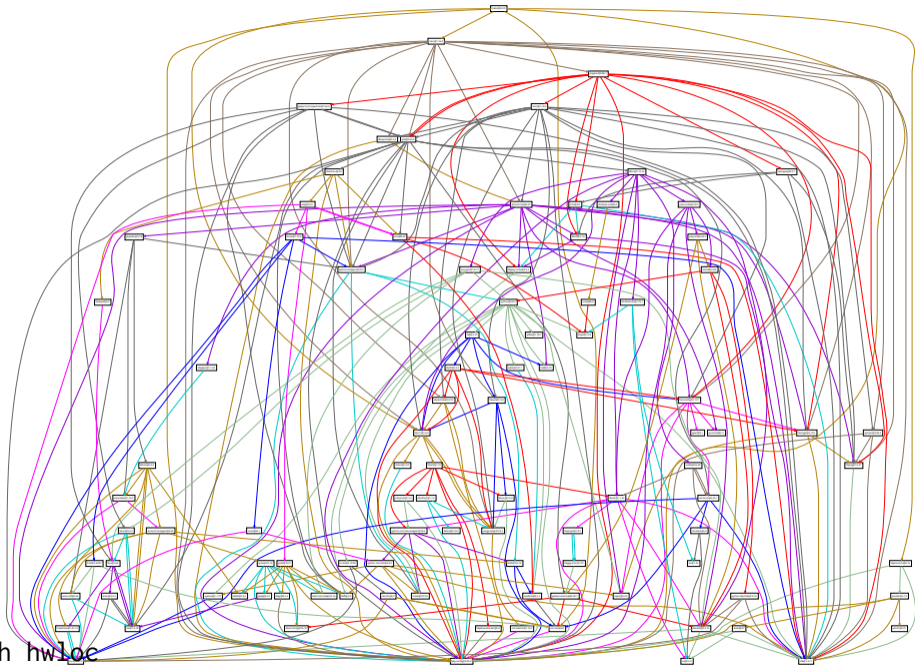

```
laptop$ guix pack -RR hpcg -S /bin=bin
```

**See Romain's talk about
deploying AVBP on Adatastra
and Jean-Zay.**

```
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

```
$ guix build hwloc
```

isolated build: chroot, separate name spaces, etc.

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

hash of **all** the dependencies



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

```
$ guix gc --references /gnu/store/...-hwloc-1.11.2  
/gnu/store/...-glibc-2.33  
/gnu/store/...-gcc-10.3.0-lib  
/gnu/store/...-hwloc-2.9.0
```

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

```
$ guix gc --references /gnu/store/...-hwloc-1.11.2  
/gnu/store/...-glibc-2.33  
/gnu/store/...-gcc-10.3.0-lib  
/gnu/store/...-hwloc-2.9.0
```

bit-identical for everyone*

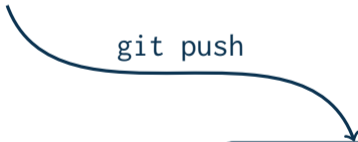
```
(define python  
  (package ...))
```

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

Git repository

guix pull

user

(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

Buils

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	rocbas.x86_64-linux	rocbas-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	rocbas.x86_64-linux	rocbas-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	rocbas.x86_64-linux	rocbas-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	938839
Evaluation	8086586 (guix-hpc)
Status	✔ Succeeded
System	x86_64-linux
Name	rocblas-5.6.1
Duration	8302 seconds
Finished	15 Sep 01:23 +0200
Weather	● Still succeeding
Log file	pretty, raw
Derivation	/gnu/store/01rp7gjfvgv5m9dfm1579hrk812aqqc0y-rocblas-5.6.1.drv
Dependencies	<ul style="list-style-type: none">✔ cmake-minimal-3.24.2✔ tar-1.34✔ gzip-1.13✔ bzip2-1.0.8✔ file-5.45✔ diffutils-3.10✔ patch-2.7.6

instance at <https://guix.bordeaux.inria.fr>

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 : <hr/> <table><tr><td>guix</td><td>c0d4bd5 → 034eb1b</td></tr></table>	guix	c0d4bd5 → 034eb1b
guix	c0d4bd5 → 034eb1b		
Log file	pretty, raw		
Derivation	/gnu/store/kh77cdzsfxmh7h9bbr1nk2vzqw7dqb6b-maphys-1.0.0.drv		
Dependencies	✔ hwloc-2.11.1		

Wrapping up.

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

build daemon

client commands

```
guix build hwloc
```

build processes
chroot, separate UIDs

client commands

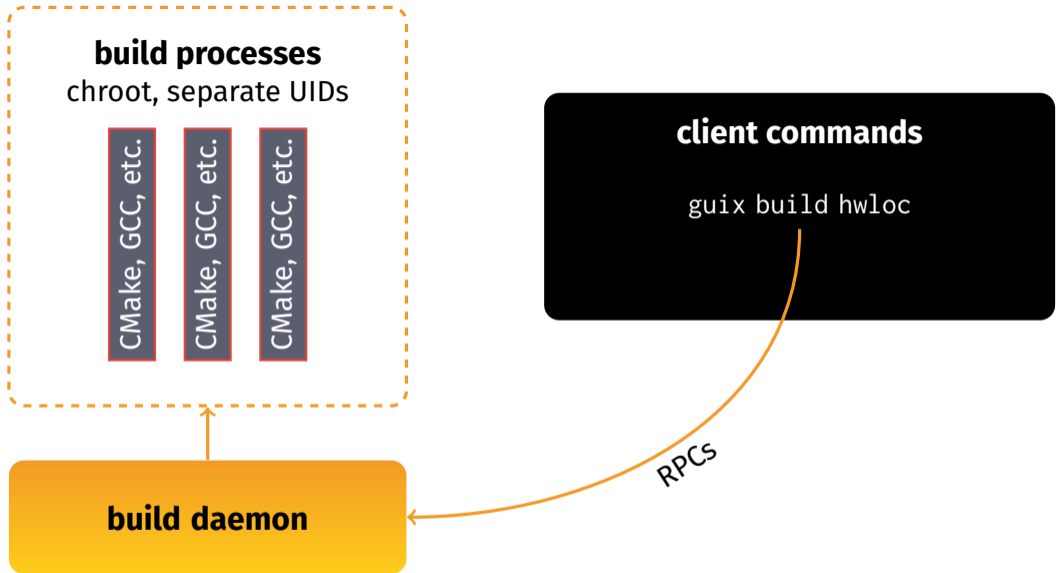
`guix build hwloc`

build daemon

RPCs

```
graph TD; A[client commands] -- RPCs --> B[build daemon];
```

The diagram illustrates the interaction between a client and a build daemon. On the right, a black rounded rectangle labeled "client commands" contains the code `guix build hwloc`. An orange curved arrow labeled "RPCs" points from this box to a yellow rounded rectangle on the left labeled "build daemon". Above the daemon, a dashed orange box labeled "build processes" contains the text "chroot, separate UIDs", indicating the environment in which the daemon operates.



common package
collections

use & publish
binaries

Sharing!

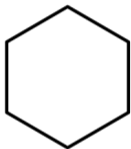
`/etc/guix/channels.scm`

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
```

hash of **all** the dependencies



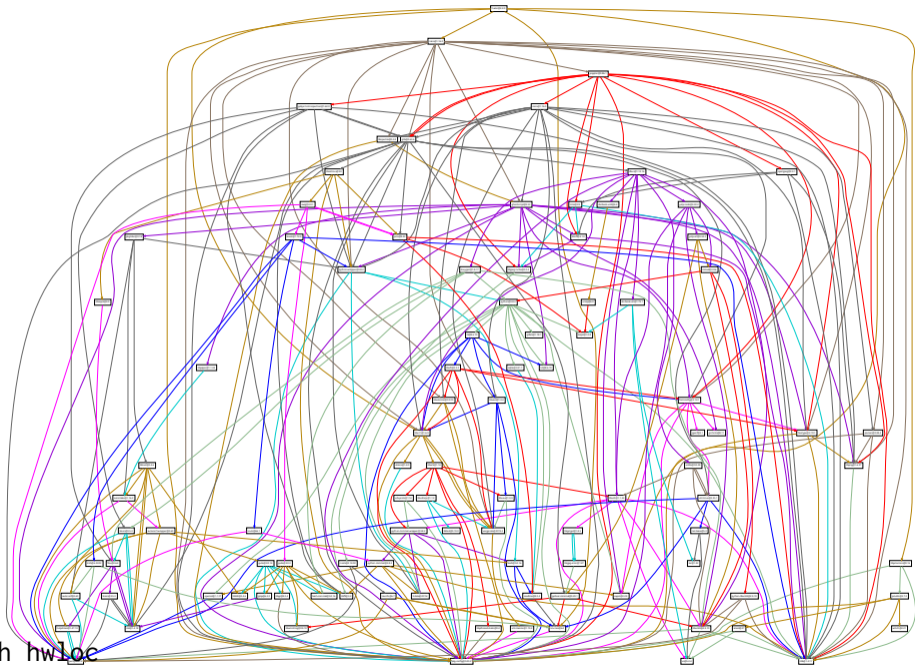
```
$ guix build hwloc  
/gnu/store/h2g4sf72...-hwloc-1.11.2
```

```
$ 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


```
(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...")))))
  ...))
```



Software Heritage

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

(operating-system

```
(host-name "guibox")
```

```
(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 "guibox")
  (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 "guibox")
  (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 "guibox")
  (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 "guibox")
  (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 "guibox")
  (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 "/"))
                              (file-system
                                (device "vda4")
                                (mount-point "/var"))
                              %base-file-systems))
                %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:gnupl3+)))
```

```
(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" <img alt="arrow" data-bbox="615 545 630 560"/>))))))

(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:gnpl3+)))
```

```
(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:gp13+)))
```



```
(define-public hello-cerfacs
  (package
    (name "hello-cerfacs")
    (version "1.0")
    (source (origin
              (method url-fetch)
              (uri (string-append
                    dependencies"http://example.org/hello-" version
                    ".tar.gz"))
              (sha256 (base32 "0wqd...dz6" )))
            reference to a variable)
    (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:gp13+)))
```

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

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

for a file called cerfacts.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

build daemon

client commands

```
guix build hello
```

build processes

chroot, separate UIDs

client commands

```
guix build hello
```

build daemon

RPCs

```
graph TD; A[client commands] -- RPCs --> B[build daemon];
```

build processes

chroot, separate UIDs

Guile, make, etc.

Guile, make, etc.

Guile, make, etc.

build daemon

client commands

```
guix build hello
```

RPCs

```
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

Copyright © 2010, 2012–2024 Ludovic Courtès ludo@gnu.org.

GNU Guix logo, CC-BY-SA 4.0, <https://gnu.org/s/guix/graphics>.

Smoothie image and hexagon image © 2019 Ricardo Wurmus, CC-BY-SA 4.0.

Atari computer picture © winkelnkemper, CC-BY-SA 2.0,
https://commons.wikimedia.org/wiki/File:Atari_800_XL_home_computer_with_monitor_and_tape_program_recorder_XC12.jpg

Guix groupe photo by Tess Gobain, <https://hpc.guix.info/events/2023/workshop>

DeLorean time machine picture © 2014 Oto Godfrey and Justin Morton, CC-BY-SA 4.0,
https://commons.wikimedia.org/wiki/File:TeamTimeCar.com-BTTF_DeLorean_Time_Machine-OtoGodfrey.com-JMortonPhoto.com-07.jpg.

Copyright of other images included in this document is held by their respective owners.

This work is licensed under the **Creative Commons Attribution-Share Alike 3.0** License. To view a copy of this license, visit <https://creativecommons.org/licenses/by-sa/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

At your option, you may instead copy, distribute and/or modify this document under the terms of the **GNU Free Documentation License, Version 1.3 or any later version** published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at <https://www.gnu.org/licenses/gfdl.html>.

The source of this document is available from <https://gitlab.inria.fr/lcourtes/talks>.