Tying software deployment to scientific workflows

Using GNU Guix to make software deployment a first-class citizen

Ludovic Courtès

FOSDEM 2018

Guix in a nutshell.

```
$ guix package --install gcc-toolchain openmpi hwloc
...

$ eval `guix package --search-paths`
...

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

$ guix package --roll-back
...
```

```
bob@laptop$ guix pull --commit=cabba9e
bob@laptop$ guix package -i gcc-toolchain openblas
```

```
bob@laptop$ guix pull --commit=cabba9e
bob@laptop$ guix package -i gcc-toolchain openblas
```

```
alice@supercomp$ guix pull --commit=cabba9e
alice@supercomp$ guix package -i gcc-toolchain openblas
```

```
bob@laptop$ guix pull --commit=cabba9e
bob@laptop$ guix package -i gcc-toolchain openblas
```

reproducible & portable!

```
alice@supercomp$ guix pull --commit=cabba9e
alice@supercomp$ guix package -i gcc-toolchain openblas
```

```
$ guix build hwloc \
    --with-source=./hwloc-2.0rc1.tar.gz
…
```

```
$ guix build hwloc \
    --with-source=./hwloc-2.0rc1.tar.gz
…
```

```
$ guix package -i mumps --with-input=scotch=pt-scotch
…
```

```
$ guix build hwloc \
    --with-source=./hwloc-2.0rc1.tar.gz
…

$ guix package -i mumps --with-input=scotch=pt-scotch
…

$ guix package -i julia --with-input=fftw=fftw-avx
…
```

- started in 2012
- **6,800+ packages**, all free software
- x86_64, i686, ARMv7, AArch64
- binaries at `https://hydra.gnu.org`
- 0.14.0 released in December 2017

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


The archipelago of “tools that do one thing.”

Reproducible deployment at the center of the stage.

“Package management”

```
$ guix package -i openfoam emacs
```

“Virtual environments”

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

Container provisioning

```
$ guix pack hwloc
...
/gnu/store/...-pack.tar.gz
```

Container provisioning

```
$ guix pack --format=docker hwloc
...
/gnu/store/...-docker-image.tar.gz
```

Intermezzo: the programming language underpinnings

expression

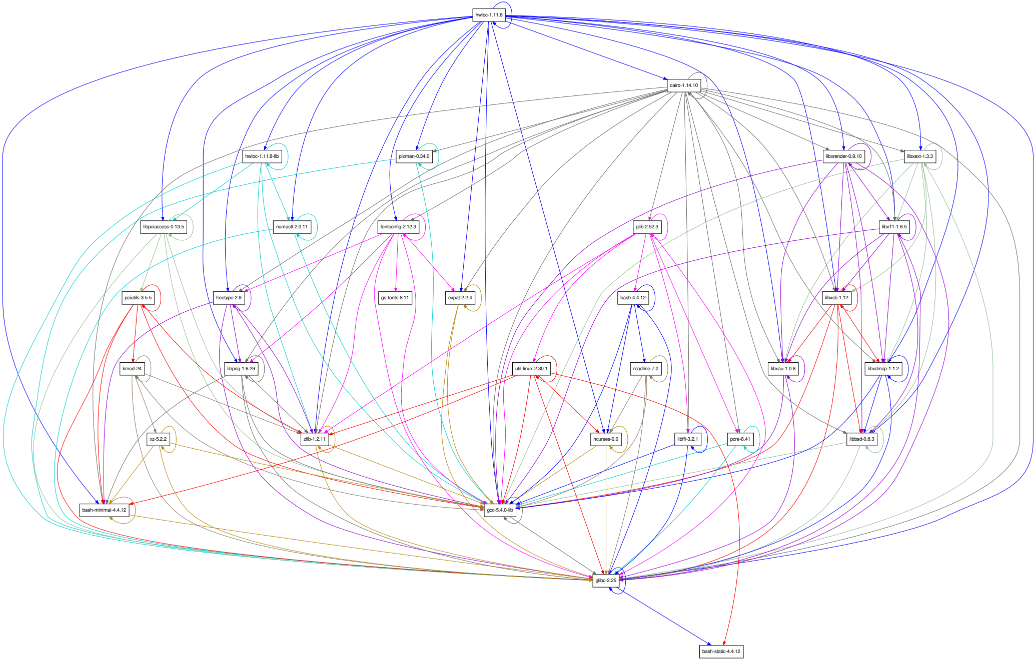
```
(system* "/bin/lstopo")
```

staged expression

```
'(system* "/bin/lstopo")
```

deployment-aware staged expression

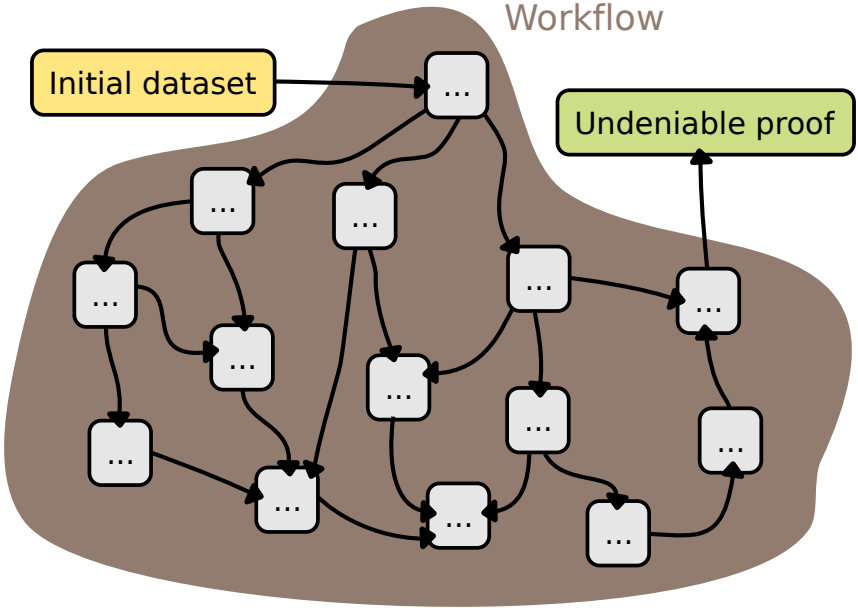
```
#~(system* #$(file-append hwloc "/bin/lstopo"))
```



hwloc-1.11.8
cairo-1.14.10
hwloc-1.11.8-lib
pixman-0.34.0
libxrender-0.9.10
libxext-1.3.3
libpciaccess-0.13.5
numactl-2.0.11
fontconfig-2.12.3
glib-2.52.3
libx11-1.6.5
pciutils-3.5.5
freetype-2.8
gs-fonts-8.11
expat-2.2.4
bash-4.4.12
libxcb-1.12
kmod-24
libpng-1.6.29
util-linux-2.30.1
readline-7.0
libxau-1.0.8
libxdmcp-1.1.2
xz-5.2.2
zlib-1.2.11
ncurses-6.0
libffi-3.2.1
pcre-8.41
libbsd-0.8.3
bash-minimal-4.4.12
gcc-5.4.0-lib
glibc-2.25
bash-static-4.4.12

Guix Workflow Language

http://www.guixwl.org


Workflow
Initial dataset
Undeniable proof
...

```
define-module
    test
use-modules
    guix workflows
    guix processes
    gnu packages bioinformatics
    gnu packages python

process: simple-test
    package-inputs
        list python samtools
    data-inputs
        list "sample.bam" "hg38.fa" "abc"
    procedure #---python
import os
print "hello from python 3"
---
```

```
guix workflow --run=example \
     -i input.dat -o output.dat \
     --engine=grid-engine
```

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[1–3]. And still, when a fourth one appeared[4], 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[5].

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.

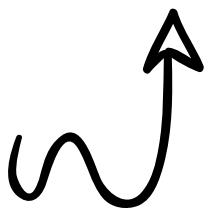
To achieve this goal, the whole publishing chain is radically different from other traditional scientific journals. ReScience lives on GitHub where each new implementation of a computational study is made available together with comments, explanations and tests. Each submission takes the form of a pull 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 ReScience Journal

Software Heritage

The ReScience Journal

Software Heritage



The ReScience Journal

Let’s connect the bits!

guix-hpc@gnu.org

https://hpc.guixsd.org/