Guix and long term: difficulties and countermeasures

How to redo later and overthere
what had be done today and here?

Simon Tournier

Inserm US53 - UAR CNRS 2030
`simon.tournier@inserm.fr`

March 6th, 2023
`https://hpc.guix.info`

Replication and reproducibility crisis

More than 70% of researchers have tried and **failed to reproduce** another scientist's experiments, and more than half have failed to reproduce their own experiments.

1,500 scientists lift the lid on reproducibility (Nature, 2016) (link)

Many causes... one solution?
at least, *Open Science* helps

$$\left(\begin{array}{lcl} \text{reproductibility} & = & \text{verification} \\ \text{replicability} & = & \text{validation} \end{array}\right)$$

1905: *Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen*
by A. Einstein

- Only one author, verbal reasoning
- Motivated students are able to check by themselves that all the computations are correct

1905: *Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen*
by A. Einstein

- Only one author, verbal reasoning
- Motivated students are able to check by themselves that all the computations are correct

2022: *Evolutionary-scale prediction of atomic level protein structure with a language model*
by Z. Zin & al.

- 15 authors, references to software
- "[...] we scale language models from 8 million parameters up to ***15 billion* parameters**."
- Code and data seems available. . . but ~~impossible~~^W hard to check that all is correct

1905: *Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen*
by A. Einstein

- Only one author, verbal reasoning
- Motivated students are able to check by themselves that all the computations are correct

2022: *Evolutionary-scale prediction of atomic level protein structure with a language model*
by Z. Zin & al.

- 15 authors, references to software
- "[...] we scale language models from 8 million parameters up to ***15 billion*** **parameters**."
- Code and data seems available. . . but ~~impossible~~^W hard to check that all is correct

Among several questions*, scientific research is evolving,

what does it mean *scientific research* now?

**is 15 billion parameters explanatory?*

Open Science

Science	=	Transparent and Collective
Scientific result	=	Experiment + Numerical treatment

Science at the digital age:

1.	Open Article	HAL, BioArxiv
2.	Open Data	Data Repositories, Zenodo
3.	Open Source	Forges, GitLab, Software Heritage

open science, a tautology?

Open Science

Science	=	Transparent and Collective
Scientific result	=	Experiment + $\boxed{\textit{Numerical treatment}}$

Science at the digital age:

1.	Open Article	HAL, BioArxiv
2.	Open Data	Data Repositories, Zenodo
3.	Open Source	Forges, GitLab, Software Heritage

how to *glue* all that?

open science, a tautology?

Open Science

Science	=	Transparent and Collective
Scientific result	=	Experiment + $\boxed{\textit{Numerical treatment}}$

Science at the digital age:

1.	Open Article	HAL, BioArxiv
2.	Open Data	Data Repositories, Zenodo
3.	Open Source	Forges, GitLab, Software Heritage
4.	$\boxed{\text{Computational env.}}$	?

how to ***glue*** all that?

open science, a tautology?

Open Science

Science	=	Transparent and Collective
Scientific result	=	Experiment + $\boxed{\textit{Numerical treatment}}$

Science at the digital age:

1.	Open Article	HAL, BioArxiv
2.	Open Data	Data Repositories, Zenodo
3.	Open Source	Forges, GitLab, Software Heritage
4.	$\boxed{\text{Computational env.}}$	?

how to ***glue*** all that?

today's topic
considering long-term (3-5 years)

open science, a tautology?

Redo (reproduce or replicate) a result?

		audit		opaque		depend?
result	⟵	paper	+	data	+	analysis
data	⟵	protocol	+	instruments	+	materials
analysis	⟵	script	+	data	+	environment

- audit is the « tractable » part
- opaque is generally the hard part

Redo (reproduce or replicate) a result?

		audit		opaque		depend?
result	⟵	paper	+	data	+	analysis
data	⟵	protocol	+	instruments	+	materials
analysis	⟵	script	+	data	+	environment

- audit is the « tractable » part
- opaque is generally the hard part
- how to evacuate depend? from the equations

Redo (reproduce or replicate) a result?

			audit		opaque		depend?
	result	⟵	paper	+	data	+	analysis
	data	⟵	protocol	+	instruments	+	materials
★	analysis	⟵	script	+	data	+	environment

- audit is the « tractable » part
- opaque is generally the hard part
- how to evacuate depend? from the equations

★ *our issue*

Redo (reproduce or replicate) a result?

			audit		opaque		depend?
	result	⟵	paper	+	data	+	analysis
	data	⟵	protocol	+	instruments	+	materials
★	analysis	⟵	script	+	data	+	$\boxed{\text{environment}}$

- audit is the « tractable » part
- opaque is generally the hard part
- how to evacuate depend? from the equations. . .

  . . . try to turn environment into audit

★ *our issue*

Redo (reproduce or replicate) a result?

			audit		opaque		depend?
	result	⟵	paper	+	data	+	analysis
	data	⟵	protocol	+	instruments	+	materials
★	analysis	⟵	script	+	data	+	$\boxed{\text{environment}}$

- audit is the « tractable » part
- opaque is generally the hard part
- how to evacuate depend? from the equations. . .

. . . try to turn environment into audit

★ *our issue* (« computer » ≈ instrument and « computation » ≈ measurement
computationnal env. ↔ experimental setup)

Challenges about reproducible research in science

From the « scientific method » viewpoint:

controlling the source of variations

$\Rightarrow$ transparent as with instrument $\approx$ computer

From the « scientific knowledge » viewpoint: (universal?)

- Independant observer must be able to observe the same result.
- The observation must be sustainable (to some extent).

$\Rightarrow$ collective

Challenges about reproducible research in science

From the « scientific method » viewpoint:

controlling the source of variations

⇒ transparent as with instrument ≈ computer

From the « scientific knowledge » viewpoint: (universal?)

- Independant observer must be able to observe the same result.
- The observation must be sustainable (to some extent).

⇒ collective

In a world where (almost) all is *data*

how to redo later and elsewhere what has been done today and here?

(implicitely using a « computer »)

We will speak about. . .

(some examples from `C` programming language but all apply equally to any other computational stack)

Questions (1/2)

Bessel function J_0 using `C` programming language

```
#include <stdio.h>
#include <math.h>

int main(){
  printf("%E\n", j0f(0x1.33d152p+1f));
}
```

Questions (1/2)

Bessel function J_0 using C programming language

```c
#include <stdio.h>
#include <math.h>

int main(){
    printf("%E\n", j0f(0x1.33d152p+1f));
}
```

```
Alice   sees:   5.643440E-08
Blake   sees:   5.963430E-08
```

Determine if the difference is significant or not is let to experts, scientific field by scientific field

Questions (1/2)

Bessel function J_0 using `C` programming language

```c
#include <stdio.h>
#include <math.h>

int main(){
   printf("%E\n", j0f(0x1.33d152p+1f));
}
```

Alice	sees:	5.643440E-08
Blake	sees:	5.963430E-08

Why? In spite of everything being available (*open*)

Determine if the difference is significant or not is let to experts, scientific field by scientific field

Questions (2/2)

Alice and Blake both run « GCC at version 11.2.0 »

Questions (2/2)

Alice and Blake both run « GCC at version 11.2.0 »

still different*

```
       alice@laptop$
                      5.643440E-08
      blake@desktop$
                      5.963430E-08
```

**Not an issue with floating-point computations*

Questions (2/2)

Alice and Blake both run « GCC at version 11.2.0 »

still different*

```
  alice@laptop$  gcc bessel.c                    && ./a.out
                 5.643440E-08
blake@desktop$   gcc bessel.c  -lm -fno-builtin  && ./a.out
                 5.963430E-08
```

(due to *constant folding***)

* *Not an issue with floating-point computations*

** *C language is an example, other source but similar issues with Python, R, Perl, etc.*

Questions (2/2)

Alice and Blake both run « GCC at version 11.2.0 »

still different*

```
 alice@laptop$   gcc bessel.c                     && ./a.out
                 5.643440E-08
blake@desktop$   gcc bessel.c  -lm -fno-builtin   && ./a.out
                 5.963430E-08
```

(due to *constant folding***)

Alice and Blake are running **two different computationnal environments**

**Not an issue with floating-point computations*

***C language is an example, other source but similar issues with Python, R, Perl, etc.*

Questions (2/2)

Alice and Blake both run « GCC at version 11.2.0 »

still different*

```
alice@laptop$   gcc bessel.c                       && ./a.out
                5.643440E-08
blake@desktop$  gcc bessel.c  -lm -fno-builtin     && ./a.out
                5.963430E-08
```

(due to *constant folding***)

Alice and Blake are running **two different computationnal environments**

More than version number is required

* *Not an issue with floating-point computations*

** *C language is an example, other source but similar issues with Python, R, Perl, etc.*

Questions about a computational environment

- What is the code source?
- What are the tools required for building?
- What are the tools required for running?
- And recursively for each tool. . .

Questions about a computational environment

- What is the code source?
- What are the tools required for building?
- What are the tools required for running?
- And recursively for each tool. . .

Answering these questions enables **control over sources of variations**

Questions about a computational environment

- What is the code source?
- What are the tools required for building?
- What are the tools required for running?
- And recursively for each tool. . .

Answering these questions enables **control over sources of variations**

How to capture the answer of these questions?

Usually: package manager (Conda, APT, Brew, . . .); Modulefiles; container; etc. $\Rightarrow$ **not enough!**

Questions about a computational environment

- What is the code source?
- What are the tools required for building?
- What are the tools required for running?
- And recursively for each tool. . .

Answering these questions enables **control over sources of variations**

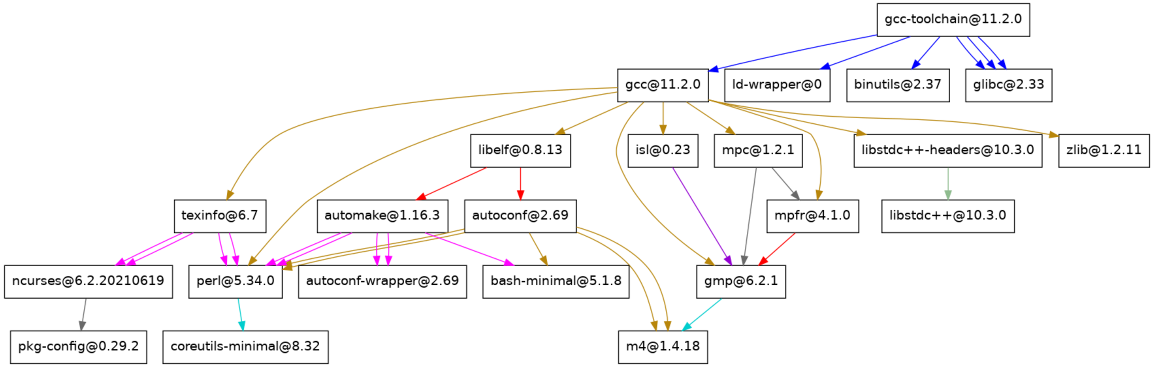
How to capture the answer of these questions?

Usually: package manager (Conda, APT, Brew, . . .); Modulefiles; container; etc. ⇒ **not enough!**

toward a solution: Guix

When Alice says « GCC at version 11.2.0 »

guix graph



Is it the same "version" of GCC if `mpfr` is replaced by version 4.0 ?

complete graph: 43 ou 104 ou 125 ou 218 nodes
(depending what we consider as *binary seed* for *bootstrapping*)

What does reproducing a computational environment mean?

Alice says "GCC at version 11.2.0"

All the tools (node of the graph) must be captured!

Remember

```
#include <stdio.h>
#include <math.h>

int main(){
  printf("%E\n", j0f(0x1.33d152p+1f));
}
```

```
  alice@laptop$  gcc bessel.c                   && ./a.out
                 5.643440E-08
 carole@desktop$  gcc bessel.c  -lm -fno-builtin  && ./a.out
                 5.963430E-08
```

(due to *constant folding*)

What is my version of Guix?

guix describe = state

```
$ guix describe
Generation 76 Apr 25 2022 12:44:37 (current)
  guix eb34ff1
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: eb34ff16cc9038880e87e1a58a93331fca37ad92

$ guix --version
guix (GNU Guix) eb34ff16cc9038880e87e1a58a93331fca37ad92
```

What is my version of Guix?

guix describe = state

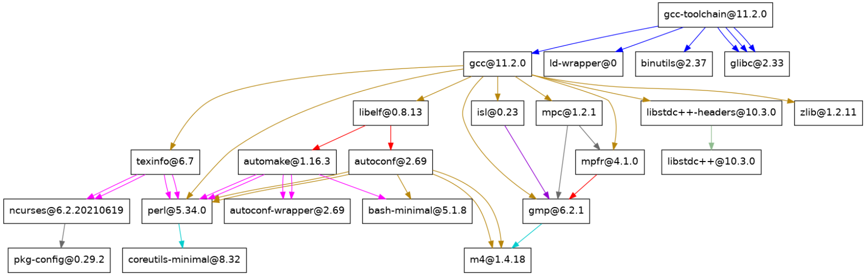
```
$ guix describe
Generation 76 Apr 25 2022 12:44:37 (current)
  guix eb34ff1
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: eb34ff16cc9038880e87e1a58a93331fca37ad92

$ guix --version
guix (GNU Guix) eb34ff16cc9038880e87e1a58a93331fca37ad92
```

one state **pins** the complete collection of packages and Guix itself

A state can refer to several channels (= Git repository), pointing to URL, branches or commits different
A channel contains a list of recipes (code source, how to build the packages, etc.)

State = Directed Acyclic Graph(*DAG*)



Each node specifies a recipe defining:

- code source — and potentially some *ad-hoc* modifications (`patch`)
- build-time tools — compilers, build automation, configuration flags etc.
- dependencies — other packages ($\rightarrow$recursive $\rightsquigarrow$ graph)

Complete graph : Python = 137 nodes, Numpy = 189, Matplotlib = 915, Scipy = 1439 nodes

Recipe for defining a package

one node of the graph

```
(define python                          ;definition of the node python
  (package
    (name "python")
    (version "3.9.9")
    (source ... )                            ;points to URI source code
    (build-system gnu-build-system)  ;./configure & make
    (arguments ... )                         ; configure flags, etc.
    (inputs (list bzip2                      ;other nodes -> graph (DAG)
                  expat gdbm libffi sqlite ...))))
```

- Each `inputs` is similarly defined (recursion → graph)
- There is no cycle (`bzip2` or its inputs cannot refer to `python`)

What are the roots of the graph? Part of the broad *bootstrapping* (link) problem

Package manager = graph manager

How to capture this information?

- What is the source code ? `source`
- What are the tools required for building? — `inputs`, `propagated-`, `native-inputs`
- What are the tools required for running? — `inputs`, `propagated-`, `native-inputs`
- How is each tool produced? `build-system`, `arguments`

Package manager = graph manager

How to capture this information?

- What is the source code ? `source`
- What are the tools required for building?
- What are the tools required for running?

  } `inputs`, `propagated-`, `native-inputs`
- How is each tool produced? `build-system`, `arguments`

```
(define python                          ;definition of the node python
  (package
    (name "python")
    (version "3.9.9")
    (source ... )                         ;points to URI source code
    (build-system gnu-build-system)       ;./configure & make
    (arguments ... )                      ; configure flags, etc.
    (inputs (list bzip2                   ;other nodes -> graph (DAG)
```

Revision = one specific graph

« GCC at version 11.2.0 » = one pinned graph

```
$ guix describe
Generation 76 Apr 25 2022 12:44:37 (current)
  guix eb34ff1
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: eb34ff16cc9038880e87e1a58a93331fca37ad92
```

this revision `eb34ff1` captures the **complete** graph

- Alice says « I used Guix at revision `eb34ff1` »
- Blake knows all for reproducing the same environment

Collaboration in action

Guix is helping

Alice

describes her environment:

- the list of the tools using the file `manifest.scm`, spawns her environment e.g.,

```
guix shell -m manifest.scm
```

Collaboration in action

Guix is helping

Alice

describes her environment:

- the list of the tools using the file `manifest.scm`, spawns her environment e.g.,

```
guix shell -m manifest.scm
```

- the revision (Guix itself and potentially all the other channels)

```
guix describe -f channels > state-alice.scm
```

Collaboration in action

Guix is helping

collaborate = share one computational environment

Alice

describes her environment:

- the list of the tools using the file `manifest.scm`, spawns her environment e.g.,

```
guix shell -m manifest.scm
```

- the revision (Guix itself and potentially all the other channels)

```
guix describe -f channels > state-alice.scm
```

Collaboration in action

Guix is helping

collaborate = share one computational environment ⇒ share one specific graph

Alice

describes her environment:

- the list of the tools using the file `manifest.scm`, spawns her environment e.g.,

```
guix shell -m manifest.scm
```

- the revision (Guix itself and potentially all the other channels)

```
guix describe -f channels > state-alice.scm
```

Collaboration in action

Guix is helping

collaborate = share one computational environment ⇒ share one specific graph

Alice

describes her environment:

- the list of the tools using the file `manifest.scm`, spawns her environment e.g.,

```
guix shell -m manifest.scm
```

- the revision (Guix itself and potentially all the other channels)

```
guix describe -f channels > state-alice.scm
```

- then **shares these two files**: `state-alice.scm` and `manifest.scm`

Collaboration in action

Guix is helping

collaborate = share one computational environment ⇒ share one specific graph

Alice

describes her environment:

- the list of the tools using the file `manifest.scm`, spawns her environment e.g.,

```
guix shell -m manifest.scm
```

- the revision (Guix itself and potentially all the other channels)

```
guix describe -f channels > state-alice.scm
```

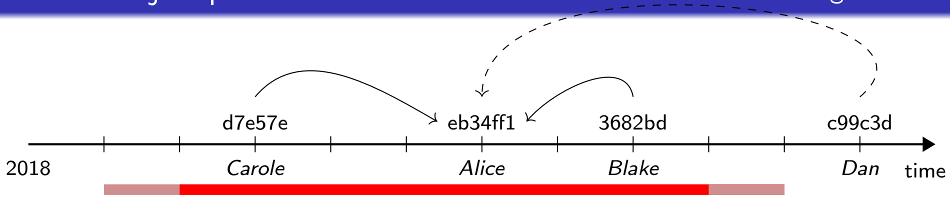
- then **shares these two files**: `state-alice.scm` and `manifest.scm`

Blake

spawns the same computational environment **from these two files**

```
guix time-machine -C state-alice.scm -- shell -m manifest.scm
```

Reproducible = jump to different states `guix time-machine`



Requirements for being reproductible with the passing of time using Guix:

- Preservation of the **all** source code (≈ 75% archived (link) in Software Heritage (link))
- *Backward* compatibility of the Linux kernel
- Compatibility of *hardware* (to some extent)
- (No time-bomb!)

What is the size of this temporal window where these 3 conditions are satisfied?

To my knowledge, the Guix project is quasi-unique by experimenting since v1.0 in 2019.

how to redo later and elsewhere what has been done today and here?

Traceability and transparency

being collectively able to study bug-to-bug

Guix should manage everything about the **environment**

```
guix time-machine -C state.scm -- cmd -m list-software.scm
```

if it is specified

« how to build »	`channels.scm`	(*state*)
« what to build »	`manifest.scm`	(*software list*)

how to redo later and elsewhere what has been done today and here?

Traceability and transparency

being collectively able to study bug-to-bug

Guix should manage everything about the **environment**

```
guix time-machine -C state.scm -- cmd -m list-software.scm
```

if it is specified

« how to build » `channels.scm` (*state*)

« what to build » `manifest.scm` (*software list*)

What is required in addition to these 2 files?

Preservation of what? (1/3)

```
guix time-machine -C channels.scm -- shell -m manifest.scm
```

★	each channel	used by	`channels.scm`	(= Git repository defining packages)
★	code source	used by	`manifest.scm`	(= URI pointing to upstream)

```
(define python                          ;package definition
  (package
    (name "python")
    (version "3.9.9")
    (source ... )                       ;package source
    (build-system gnu-build-system)
    (arguments ... )
    (inputs (list ...))))
```

Preservation of what? (2/3)

example of source

```
(source
 (origin
   (method url-fetch)
   (uri (string-append "https://www.python.org/ftp/python/"
                       version "/Python-" version ".tar.xz"))
   (patches (search-patches ...))
   (sha256
    (base32
     "09vd7g71i11iz5ydqghwc8kaxr0vgji94hhwwnj77h3kll28r0h6"))))
```

Préservation de quoi ? (3/3)

- Git repository (channel)
- `source`
  - archive *tarballs* (compressed) `url-fetch`
  - Git repository `git-fetch`
  - Subversion repository `svn-fetch`
  - Mercurial repository `hg-fetch`
  - CVS repository `cvs-fetch`

Préservation de quoi ? (3/3)

▶ Git repository	(channel)
▶ `source`	
▶ archive *tarballs* (compressed)	`url-fetch`
▶ Git repository	`git-fetch`
▶ Subversion repository	`svn-fetch`
▶ Mercurial repository	`hg-fetch`
▶ CVS repository	`cvs-fetch`

```
$ guix repl -- sources.scm | sort | uniq -c | sort -nr
                 13432  url-fetch
                  6691  git-fetch
                   391  svn-fetch
                    43  other
                    31  hg-fetch
                     3  cvs-fetch
```

Why preserving?

Because **online services sometimes stop**

- Google Code (link) early 2016
- Alioth (Debian) in 2018 replaced by Salsa
- Gna! in 2017 after 13 years
- Gitourious in 2015 (the second most popular service for hosting Git repository in 2011)
- etc.

Why preserving?

Because **online services sometimes stop**

- Google Code (link) early 2016
- Alioth (Debian) in 2018 replaced by Salsa
- Gna! in 2017 after 13 years
- Gitourious in 2015 (the second most popular service for hosting Git repository in 2011)
- etc.
- `gforge.inria.fr` for example Guix issue #42162 (link)

```
Believe it or not, gforge.inria.fr was finally phased out on
Sept. 30th.  And believe it or not, despite all the work and all the
chat :-), we lost the source tarball of Scotch 6.1.1 for a short period
of time (I found a copy and uploaded it to berlin a couple of hours
ago).
```

How to preserve?

Forge $\neq$ Archive

collaborative software platform for developing

L'objectif d'une forge est de permettre à plusieurs développeurs de **participer ensemble au développement** d'un ou plusieurs logiciels, le plus souvent à travers le réseau Internet.

https://fr.wikipedia.org/wiki/Forge_(informatique)

(no English wikipedia entry)

L'archivage est un ensemble d'actions qui a pour but de garantir l'accessibilité sur le long terme d'informations (dossiers, documents, données) que l'on doit ou souhaite conserver pour des raisons juridiques

https://fr.wikipedia.org/wiki/Archivage

Software Heritage *« are building the universal software archive »* (link)

Online service sometimes stop. . .

Why would it be different for Software Heritage?

No guarantee but. . .

Software Heritage is an open, non-profit initiative unveiled in 2016 by Inria. It is supported by a broad panel of institutional and industry partners, in collaboration with UNESCO.

The long term goal is to collect all publicly available software in source code form together with its development history, replicate it massively to ensure its preservation, and share it with everyone who needs it.

- Strong support by national and international institutes
- With the mission to specifically archive all the open source code

(SWH demo?)

Preservation with Software Heritage

`https://www.softwareheritage.org/`

collect and **preserve** software in source code form in the very **long term**
(not a forge!)

Guix is able:

- save source code from Guix package definition and the Guix package definition itself
- use Software Heritage archive as fallback if upstream source disappears

Questions:

- How to cite a software? Reference to source code only? Dependencies? Build options?
- **Intrinsic** identifier *(depends only on the object; as checksum)*
  vs **Extrinsic** identifier
  (depends on a register to keep the correspondence between identifier and object; as label version)

Guix and Software Heritage (SWH): status

Guix is able to:

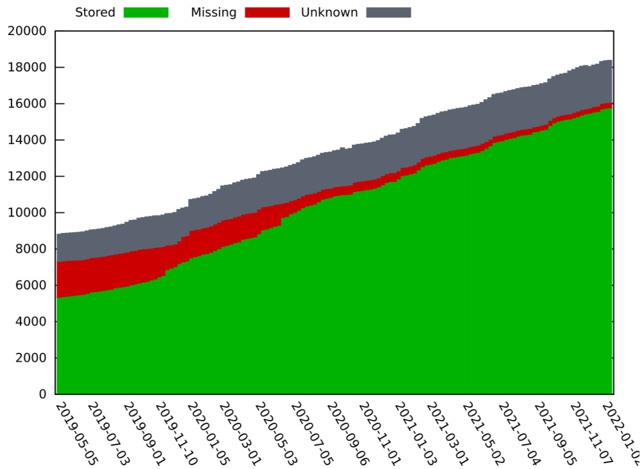
- save (automatically) source code to SWH
- find back (automatically) source code from SWH

But not for all the type of source!

- Current tooling for Guix ecosystem:
  - Git repository `git-fetch`
  - archive *tarballs* (compressed) `url-fetch`
- . . . and all the rest is still missing (help is welcome :-))

Some stats : preservation of Guix

https://ngyro.com/pog-reports/latest/



Saved in Software Heritage:

- Git = 98.3%
- *tarballs* ≈ 70%

(need some love for resuming :-))

As an practitioner

- Save the source code of the package

```
guix lint -c archival some-package
```

As an practitioner

- Save the source code of the package

```
guix lint -c archival some-package
```

- Manually save the Git repository of third-party channels (the complete recipe = graph)

As an practitioner

- Save the source code of the package

```
guix lint -c archival some-package
```

- Manually save the Git repository of third-party channels (the complete recipe = graph)

- Find the source of one package (if "missing")

  ~~`guix help`~~ . . . bah that's automatic :-)

  If all is working as expected, this does all the job,

```
guix time-machine -C channels.scm -- shell -m manifest.scm
```

As an practitioner

- Save the source code of the package

```
guix lint -c archival some-package
```

- Manually save the Git repository of third-party channels (the complete recipe = graph)

- Find the source of one package (if "missing")

~~`guix help`~~ . . . bah that's automatic :-)

If all is working as expected, this does all the job,

```
guix time-machine -C channels.scm -- shell -m manifest.scm
```

Guix tries, in that order,

- is it available in the substitute servers?
- is it available upstream (defined by `source`)?
- is it in Software Heritage?

Reproducible researcher point of view

- Alice implements a software where the source code is at:

  `https://gitlab.inria.fr/projet/un-outil.git`

  and package it using this channel:

  `https://gitlab.inria.fr/projet/un-canal.git`

Git channel repository (`channels.scm`) fetched from SWH too!

Reproducible researcher point of view

- Alice implements a software where the source code is at:

  `https://gitlab.inria.fr/projet/un-outil.git`

  and package it using this channel:

  `https://gitlab.inria.fr/projet/un-canal.git`
- Alice saves in Software Heritage these both Git repositories

Git channel repository (`channels.scm`) fetched from SWH too!

Reproducible researcher point of view

- Alice implements a software where the source code is at:

  `https://gitlab.inria.fr/projet/un-outil.git`

  and package it using this channel:

  `https://gitlab.inria.fr/projet/un-canal.git`
- Alice saves in Software Heritage these both Git repositories
- Alice publishes along to the paper the two file `channels.scm` and `manifest.scm`

Git channel repository (`channels.scm`) fetched from SWH too!

Reproducible researcher point of view

- Alice implements a software where the source code is at:

  `https://gitlab.inria.fr/projet/un-outil.git`

  and package it using this channel:

  `https://gitlab.inria.fr/projet/un-canal.git`
- Alice saves in Software Heritage these both Git repositories
- Alice publishes along to the paper the two file `channels.scm` and `manifest.scm`
- Blake is able to redeploy the same computational environment as Alice

  . . . although the `inria.fr` server machine are down!

```
guix time-machine -C channels.scm -- shell -m manifest.scm
```

Git channel repository (`channels.scm`) fetched from SWH too!

Fallback in action

```
$ guix time-machine -C channels.scm -- shell -m manifest.scm
Updating channel 'guix' from Git repository at 'https://git.savannah.gnu.org/gi
Updating channel 'example' from Git repository at 'https://whatever-here.org/do
SWH: found revision 67c9f2143aa6f545419ae913b4ae02af4cd3effc with directory at
SWH vault: requested bundle cooking, waiting for completion...
swh:1:rev:67c9f2143aa6f545419ae913b4ae02af4cd3effc.git/
[...]
fatal: could not read Username for 'https://github.com': No such device or addr
Trying content-addressed mirror at berlin.guix.gnu.org...
Trying to download from Software Heritage...
SWH: found revision e1eefd033b8a2c4c81babc6fde08ebb116c6abb8 with directory at'
[...]
```

https://simon.tournier.info/posts/2021-10-25-software-heritage.html

Cool, but. . .

- How to identify / reference my code?

  intrinsic vs extrinsic identifier (link)

  tag (v1.2.3) vs hash (20303c0), then which hash? etc.
- How to be sure that the complete graph is saved and preserved?
- How to deal if one node is failing to be rebuild?
- What about *binary bootstrap* (the roots of the graph)?
- etc.

Links:
https://lists.gnu.org/archive/html/guix-devel/2023-02/msg00398.html
https://lists.gnu.org/archive/html/guix-devel/2023-03/msg00007.html
https://lists.gnu.org/archive/html/guix-devel/2023-03/msg00025.html

Redo the past

Being able to redeploy from now the same computational environment as 3 years ago

It requires:

- Exact same source code
- Rebuild on compatible hardware
- Deterministic rebuild

hard engineering tasks

```
$ cp $(guix build -S hello) hello.tar.gz

$ gzip -d $(guix build -S hello) -c | gzip -c > re-hello.tar.gz
```

```
$ guix hash {,re-}hello.tar.gz
086vqwk2wl8zfs47sq2xpjc9k066ilmb8z6dn0q6ymwjzlm196cd
063mn4h9mr4hqipc29dsa0a5bm330n2db8qy6hb5w5qs75mgldpb
```

```
guix shell disarchive guile-lzma guile
```

```
$ disarchive disassemble hello.tar.gz
      (sha256
        "8d99142afd92576f30b0cd7cb42a8dc6809998bc5d607d88761f512e26c7db20"))
    (header (mtime 0) (extra-flags 2) (os 3))
    (compressor gnu-best-rsync)
```

```
$ disarchive disassemble re-hello.tar.gz
      (sha256
        "eb36fa6a391a175e16341ea3d5840563d4551450ba25c16ec490e49a20b17518"))
    (header (mtime 0) (extra-flags 0) (os 3))
    (compressor gnu)
```

```
$ guix hash -S nar -H sha256 -f nix-base32 $(guix build julia-zygote -S)
02bgj6m1j25sm3pa5sgmds706qpxk1qsbm0s2j3rjlrz9xn7glgk

$ EDITOR=cat guix edit julia-zygote | grep base32 | tail -1
        (base32 "02bgj6m1j25sm3pa5sgmds706qpxk1qsbm0s2j3rjlrz9xn7glgk"))))
```

```
$ guix hash -S git -H sha1 -f hex $(guix build julia-zygote -S)
3cfdb31b517eec4173584fba2b1aa65daad46e09
```

```
$ guix hash -S nar -H sha256 -f nix-base32 $(guix build julia-zygote -S)
02bgj6m1j25sm3pa5sgmds706qpxk1qsbm0s2j3rjlrz9xn7glgk

$ EDITOR=cat guix edit julia-zygote | grep base32 | tail -1
        (base32 "02bgj6m1j25sm3pa5sgmds706qpxk1qsbm0s2j3rjlrz9xn7glgk"))))
```

```
$ guix hash -S git -H sha1 -f hex $(guix build julia-zygote -S)
3cfdb31b517eec4173584fba2b1aa65daad46e09
```

Search with `swh:1:dir:3cfdb31b517eec4173584fba2b1aa65daad46e09` returns,

https://archive.softwareheritage.org/browse/directory/3cfdb31b517eec...

see API

https://archive.softwareheritage.org/api/1/directory/3cfdb31b517eec...

Work in progress to bridge various intrinsic identifiers?

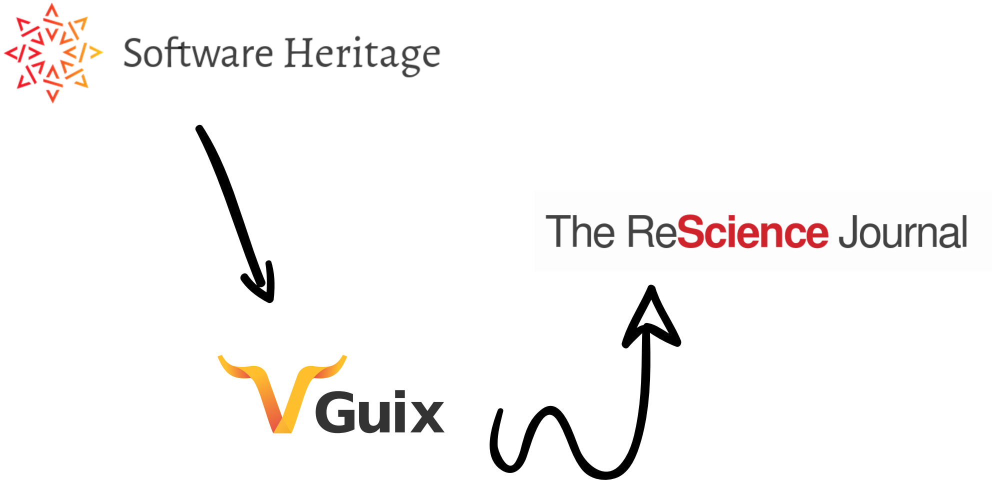
Discussion about considering NAR hashes:

https://gitlab.softwareheritage.org/swh/meta/-/issues/4538

in this thread:

- explanation of NAR format (link)
- simple Python implementation of NAR format (link)

The vision to *reach*



Questions?

`guix-science@gnu.org`

dedicated Mattermost (chat) (link)

CaféGuix

`https://hpc.guix.info/events/2022/café-guix/`