Source Code Archiving to the Rescue of Reproducible Deployment

Guix + Software Heritage

Simon Tournier
simon.tournier@inserm.fr
mastodon: @zimoun@sciences.re

Thanks

Antoine R. Dumont
Antoine Eiche
Antoine Lambert
Ludovic Courtès
Stefano Zacchiroli
Timothy Sample

https://doi.org/10.1145/3641525.3663622

GuixHPC

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Previously on “Open Research” dev room. . .

2021 (link)

Previously on “Open Research” dev room. . .

2021 (link)

Guix

2023 (link)

Previously on “Open Research” dev room. . .

2021 (link)

2023 (link)

+

2025

Software is dual

human-readable vs machine-readable

source + transformation ⟶ binary

source + transformation ⟶ binary

Program (source code)

```
/* Hello World program */

#include<stdio.h>

void main()
{
    printf("Hello World");
}
```

source + transformation ⟶ binary

Program (source code)

```
/* Hello World program */

#include<stdio.h>

void main()
{
    printf("Hello World");
}
```

Software is dual

human-readable vs machine-readable

source + transformation ⟶ binary

Program (source code)

```
/* Hello World program */

#include<stdio.h>

void main()
{
    printf("Hello World");
}
```

⟶

Program (excerpt of binary)

```
4004e6: 55
4004e7: 48 89 e5
4004ea: bf 84 05 40 00
4004ef: b8 00 00 00 00
4004f4: e8 c7 fe ff ff
4004f9: 90
4004fa: 5d
4004fb: c3
```

Software is dual

human-readable vs machine-readable

source + transformation ⟶ binary

Program (source code)

```
/* Hello World program */

#include<stdio.h>

void main()
{
    printf("Hello World");
}
```

Program (excerpt of binary)

```
4004e6: 55
4004e7: 48 89 e5
4004ea: bf 84 05 40 00
4004ef: b8 00 00 00 00
4004f4: e8 c7 fe ff ff
4004f9: 90
4004fa: 5d
4004fb: c3
```

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Software is dual

human-readable vs machine-readable

source + transformation ⟶ binary

Program (source code)

```
/* Hello World program */

#include<stdio.h>

void main()
{
    printf("Hello World");
}
```

⟶

Program (excerpt of binary)

```
4004e6: 55
4004e7: 48 89 e5
4004ea: bf 84 05 40 00
4004ef: b8 00 00 00 00
4004f4: e8 c7 fe ff ff
4004f9: 90
4004fa: 5d
4004fb: c3
```

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

long term transparency

Typical scenario

Alice publishes a result in 2022

- The source code is publicly available.
- It is identified as `v0.9`.

Typical scenario

Alice publishes a result in 2022

- The source code is publicly available.
- It is identified as `v0.9`.

Now (2025), Blake needs to scrutinize

- Blake is able to easily install `v1.2` ...
- ... But not `v0.9`.
- Using `v1.2`, the results are different.
- After *sweating* to have `v0.9` installed, the results are different too.

Typical scenario

Alice publishes a result in 2022

- The source code is publicly available.
- It is identified as `v0.9`.

Now (2025), Blake needs to scrutinize

- Blake is able to easily install `v1.2` ...
- ... But not `v0.9`.
- Using `v1.2`, the results are different.
- After *sweating* to have `v0.9` installed, the results are different too.

Why?

Does Blake fully **control over the sources of variations**?

Questions about a computational environment (deployment)

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

Questions about a computational environment (deployment)

- What is the source code?
- 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 (deployment)

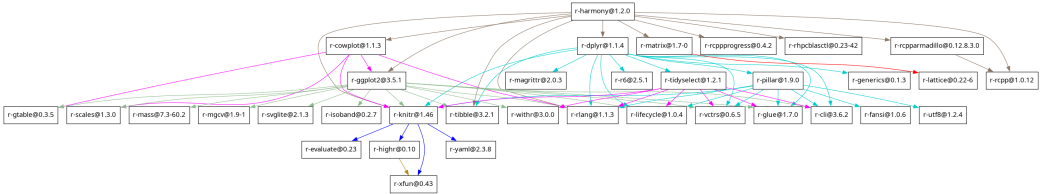
- What is the source code?
- What are the tools required for building?
- What are the tools required for running?
- And recursively for each tool...

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

How to capture the answer to these questions?

Usually: package manager (Conda, APT, Brew, ...); Modulefiles; container; etc.

Deployment = 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: more than 477 nodes

Guix comes in!

- Software development tool developed since 2012 by a large community
- Package manager or as a complete standalone system
- Functional software deployment model (pioneered by Nix)

transparent and verifiable

tool of choice when deploying software for reproducible research workflows

Revision = one specific graph

this tool "at version 0.9" = one unambiguous 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
```

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

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

- Alice says "I used Guix at revision `eb34ff1`"
- Blake knows all for deploying the exact same environment

Reproducible deployment (ideally)

- Alice says the tool `r-harmony` from Guix revision `eb34ff1`.
- Blake runs on a different machine or at a different point in time:

```
guix time-machine --commit=eb34ff1 -- install r-harmony
```

and Blake deploys the *exact same software environment*, bit-for-bit.

Under the assumptions

- All the source code is still publicly available. (e.g., more than 477)
- All the intermediary builds are deterministic.

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	Dec. 2022 v1.4.0
#sources	20 184
avail.	96.4%
missing	3.6%
hash mis.	52

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	Dec. 2022 v1.4.0
#sources	20 184
avail.	96.4%
missing	3.6%
hash mis.	52

- `openjdk-9.181.tar.bz2` is unavailable
  from its original upstream URL as it appears in Guix v1.4.0.

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	Dec. 2022 v1.4.0
#sources	20 184
avail.	96.4%
missing	3.6%
hash mis.	52

- `openjdk-9.181.tar.bz2` is unavailable
  from its original upstream URL as it appears in Guix v1.4.0.
- `openjdk@9.181` had 184 dependents
  loosing it $\Longrightarrow$ loosing 185 packages, not one.

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	May 2021 v1.3.0	Dec. 2022 v1.4.0
#sources	15 520	20 184
avail.	95.7%	96.4%
missing	4.3%	3.6%
hash mis.	66	52

- `openjdk-9.181.tar.bz2` is unavailable
  from its original upstream URL as it appears in Guix v1.4.0.
- `openjdk@9.181` had 184 dependents
  loosing it $\Longrightarrow$ loosing 185 packages, not one.

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	Nov. 2020 v1.2.0	May 2021 v1.3.0	Dec. 2022 v1.4.0
#sources	13 609	15 520	20 184
avail.	95.0%	95.7%	96.4%
missing	5.0%	4.3%	3.6%
hash mis.	69	66	52

- `openjdk-9.181.tar.bz2` is unavailable
  from its original upstream URL as it appears in Guix v1.4.0.
- `openjdk@9.181` had 184 dependents
  loosing it $\Longrightarrow$ loosing 185 packages, not one.

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	Apr. 2020 v1.1.0	Nov. 2020 v1.2.0	May 2021 v1.3.0	Dec. 2022 v1.4.0
#sources	11 659	13 609	15 520	20 184
avail.	92.4%	95.0%	95.7%	96.4%
missing	7.6%	5.0%	4.3%	3.6%
hash mis.	63	69	66	52

- `openjdk-9.181.tar.bz2` is unavailable
  from its original upstream URL as it appears in Guix v1.4.0.
- `openjdk@9.181` had 184 dependents
  loosing it $\Longrightarrow$ loosing 185 packages, not one.

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	May 2019 v1.0.0	Apr. 2020 v1.1.0	Nov. 2020 v1.2.0	May 2021 v1.3.0	Dec. 2022 v1.4.0
#sources	8 794	11 659	13 609	15 520	20 184
avail.	91.5%	92.4%	95.0%	95.7%	96.4%
missing	8.5%	7.6%	5.0%	4.3%	3.6%
hash mis.	87	63	69	66	52

- `openjdk-9.181.tar.bz2` is unavailable
  from its original upstream URL as it appears in Guix v1.4.0.
- `openjdk@9.181` had 184 dependents
  loosing it $\Longrightarrow$ loosing 185 packages, not one.

Software Heritage comes in!

Like all digital information, source code is fragile

link rot: projects are created, moved around, removed

"too big to fail": e.g., Gitorious, Google Code, Bitbucket

If a website disappears, you go to the Internet Archive...

Where do you do if (a repository on) GitHub or GitLab goes away?

Software Heritage comes in!

Like all digital information, source code is fragile

link rot: projects are created, moved around, removed

"too big to fail": e.g., Gitorious, Google Code, Bitbucket

If a website disappears, you go to the Internet Archive...

Where do you do if (a repository on) GitHub or GitLab goes away?

Answer: **Software Heritage**

Software Heritage comes in!

Like all digital information, source code is fragile

link rot: projects are created, moved around, removed

"too big to fail": e.g., Gitorious, Google Code, Bitbucket

collect, preserve and share source code

If a website disappears, you go to the Internet Archive...

Where do you do if (a repository on) GitHub or GitLab goes away?

Answer: **Software Heritage**

The SWH archive is the largest publicly available archive of software source code.

Software Heritage: international + non-profit

built with long term in mind

Sharing the vision

United Nations Educational, Scientific and Cultural Organization

www.softwareheritage.org/support/testimonials

Donors, members, sponsors

Inria

Diamond sponsor

C2A

Platinum sponsors

CNRS · intel · Microsoft · HUAWEI

Gold sponsors

Hugging Face · openinventionnetwork · servicenow · SOCIETE GENERALE · SORBONNE UNIVERSITÉ · Université Paris Cité

Silver sponsors

AdaCore · GitHub · Google · UNIVERSITÀ DI PISA

Bronze sponsors

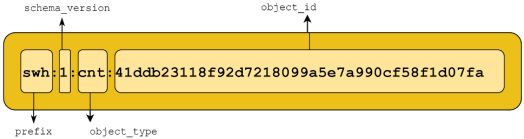
SCANOSS

www.softwareheritage.org/support/sponsors

widely supported

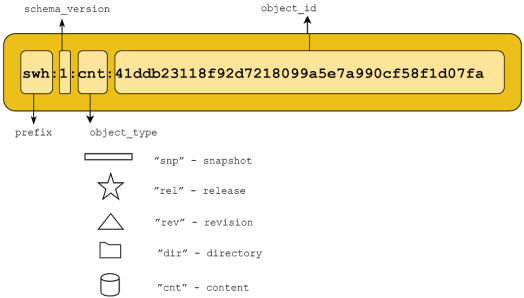
Software Heritage Identifiers (SWHID)

swhid.org



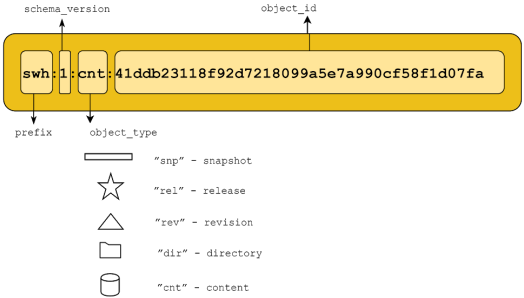
Software Heritage Identifiers (SWHID)

swhid.org



Software Heritage Identifiers (SWHID)

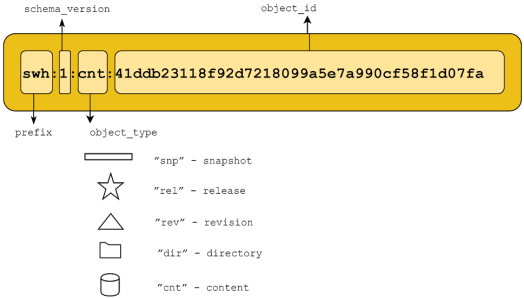
swhid.org



https://archive.softwareheritage.org/swh:1:cnt:41ddb23118f92d7218099a5e7a990cf58f1d07fa

Software Heritage Identifiers (SWHID)

swhid.org



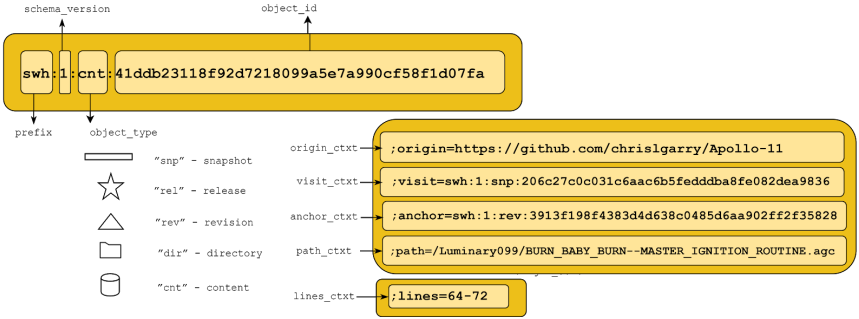
https://archive.softwareheritage.org/swh:1:cnt:41ddb23118f92d7218099a5e7a990cf58f1d07fa

an emerging standard

specification (link)

Software Heritage Identifiers (SWHID)

swhid.org



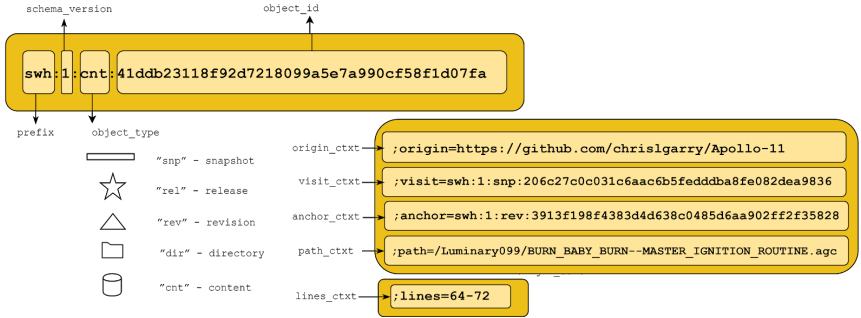
https://archive.softwareheritage.org/swh:1:cnt:41ddb23118f92d7218099a5e7a990cf58f1d07fa

an emerging standard

specification (link)

Software Heritage Identifiers (SWHID)

swhid.org



https://archive.softwareheritage.org/swh:1:cnt:41ddb23118f92d7218099a5e7a990cf58f1d07fa

content-addressed server

an emerging standard

specification (link)

Source code identification

`origin` specifies:

- `method`: *tarball*, VCS as Git, Mercurial, Subversion, etc.
- `uri`: upstream location (URL)
- `sha256`: cryptographic hash

Source code identification

`origin` specifies:

- `method`: *tarball*, VCS as Git, Mercurial, Subversion, etc.
- `uri`: upstream location (URL)
- `sha256`: cryptographic hash

source code is *essentially* **content-addressed**

Source code identification

`origin` specifies:

- `method`: *tarball*, VCS as Git, Mercurial, Subversion, etc.
- `uri`: upstream location (URL)
- `sha256`: cryptographic hash

source code is *essentially* **content-addressed**

If `uri` becomes stale, e.g., URL no longer available or tempered (hash mismatch)

Source code identification

`origin` specifies:

- `method`: *tarball*, VCS as Git, Mercurial, Subversion, etc.
- `uri`: upstream location (URL)
- `sha256`: cryptographic hash

source code is *essentially* **content-addressed**

If `uri` becomes stale, e.g., URL no longer available or tempered (hash mismatch)
Then Blake can work around. (if a copy is available elsewhere)

Source code identification

`origin` specifies:

- `method`: *tarball*, VCS as Git, Mercurial, Subversion, etc.
- `uri`: upstream location (URL)
- `sha256`: cryptographic hash

source code is *essentially* **content-addressed**

If `uri` becomes stale, e.g., URL no longer available or tempered (hash mismatch)
Then Blake can work around. (if a copy is available elsewhere)

Elsewhere might be

- a new URL

`guix download` finds the source with the expected hash and proceeds.

Source code identification

`origin` specifies:

- `method`: *tarball*, VCS as Git, Mercurial, Subversion, etc.
- `uri`: upstream location (URL)
- `sha256`: cryptographic hash

source code is *essentially* **content-addressed**

If `uri` becomes stale, e.g., URL no longer available or tempered (hash mismatch)
Then Blake can work around. (if a copy is available elsewhere)

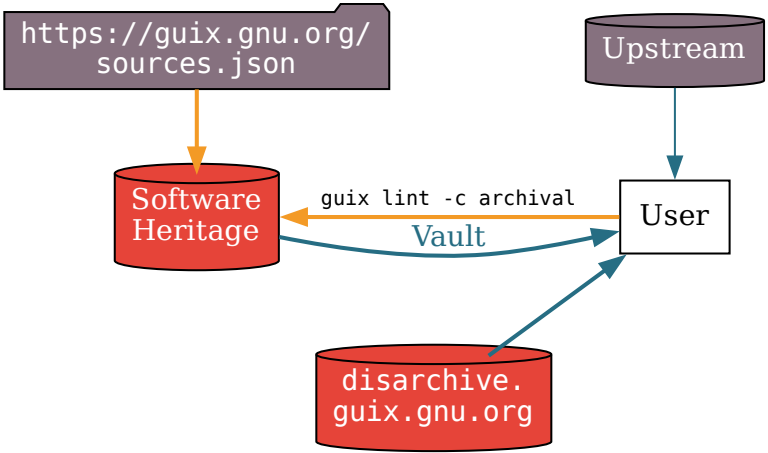
Elsewhere might be

- a new URL

  `guix download` finds the source with the expected hash and proceeds.
- a **content-addressed server**

  as served by the Guix project or the Nix project, or the Software Heritage initiative.

Architecture = connecting Guix and Software Heritage



Why Disarchive? = issue with compressed *tarballs*

```
(sha256
   (base32 "1df7b...")
```

```
$ guix hash harmony_1.2.0.tar.gz
1df7b...

$ guix hash \                    # Fast compression
   <(gzip -dc harmony_1.2.0.tar.gz | gzip -1 -c)
03v29...

$ guix hash \                    # Best compression
   <(gzip -dc harmony_1.2.0.tar.gz | gzip -9 -c)
10j87...
```

```
# Extract the compressed tarball
$ guix hash harmony-1.2.0 \
    --serializer=nar
b7900...

$ guix hash harmony-1.2.0 \
    --serializer=git
3a46b...

$ guix hash harmony-1.2.0 \
    --serializer=git --hash=sha1
75b43...
```

Why Disarchive? = issue with compressed *tarballs*

```
(sha256
   (base32 "1df7b..."))
```

```
$ guix hash harmony_1.2.0.tar.gz
1df7b...

$ guix hash \                    # Fast compression
   <(gzip -dc harmony_1.2.0.tar.gz | gzip -1 -c)
03v29...

$ guix hash \                    # Best compression
   <(gzip -dc harmony_1.2.0.tar.gz | gzip -9 -c)
10j87...
```

Process of creating compressed tarballs might vary.
(compression, timestamps, file properties, etc.)

```
# Extract the compressed tarball
$ guix hash harmony-1.2.0 \
    --serializer=nar
b7900...

$ guix hash harmony-1.2.0 \
    --serializer=git
3a46b...

$ guix hash harmony-1.2.0 \
    --serializer=git --hash=sha1
75b43...
```

Why Disarchive? = issue with compressed *tarballs*

```
(sha256
   (base32 "1df7b..."))
```

```
$ guix hash harmony_1.2.0.tar.gz
1df7b...

$ guix hash \                    # Fast compression
   <(gzip -dc harmony_1.2.0.tar.gz | gzip -1 -c)
03v29...

$ guix hash \                    # Best compression
   <(gzip -dc harmony_1.2.0.tar.gz | gzip -9 -c)
10j87...
```

Process of creating compressed tarballs might vary.
(compression, timestamps, file properties, etc.)

```
# Extract the compressed tarball
$ guix hash harmony-1.2.0 \
    --serializer=nar
b7900...

$ guix hash harmony-1.2.0 \
    --serializer=git
3a46b...

$ guix hash harmony-1.2.0 \
    --serializer=git --hash=sha1
75b43...
```

xkcd #927 ^W^W

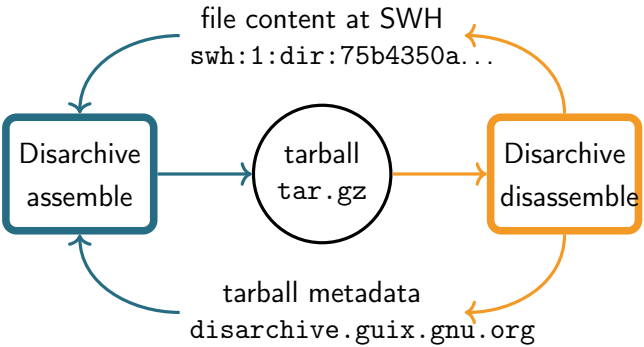
Cryptographic hash algorithm and data serializer are important for retrieving.

Disarchive dissemble output = description of metadata

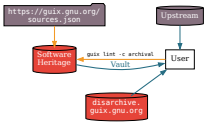
```
(disarchive
  (version 0)
  (gzip-member
    (name "harmony_1.2.0.tar.gz")
    (digest (sha256 "a63c7d7..."))
    (header (mtime 1701246604)
        (extra-flags 0) (os 3))
    (footer (crc 2567676087)
            (isize 6225920))
    (compressor gnu)
    (input
     (tarball
       (name "harmony_1.2.0.tar")
       (digest (sha256 "6c50a34..."))
       (default-header
         (uid 1010)
         (gid 100)
         (chksum (trailer " "))
         (magic "ustar ")
         ;; more omitted
```

```
(headers
  ("harmony/"
   (mode 493)
   (mtime 1701246604)
   (chksum 5084)
   (typeflag 53))
   ;; many headers omitted
   ("harmony/inst/doc/Seurat.R"
    (size 4130)
    (mtime 1701214143)
    (chksum 6701)))
(padding 0)
(input
  (directory-ref
    (version 0)
    (name "harmony_1.2.0")
    (addresses
      (swhid
          "swh:1:dir:75b4350a..."))
    (digest (sha256
        "3a46bbf..."))))))))
```

Separate storage



Retrieving source code = SWH Vault + Disarchive



content-address

Guix: "*normalized archived*" (`nar`) + `sha256`

SWH: SWHID = Git compatible `sha1`

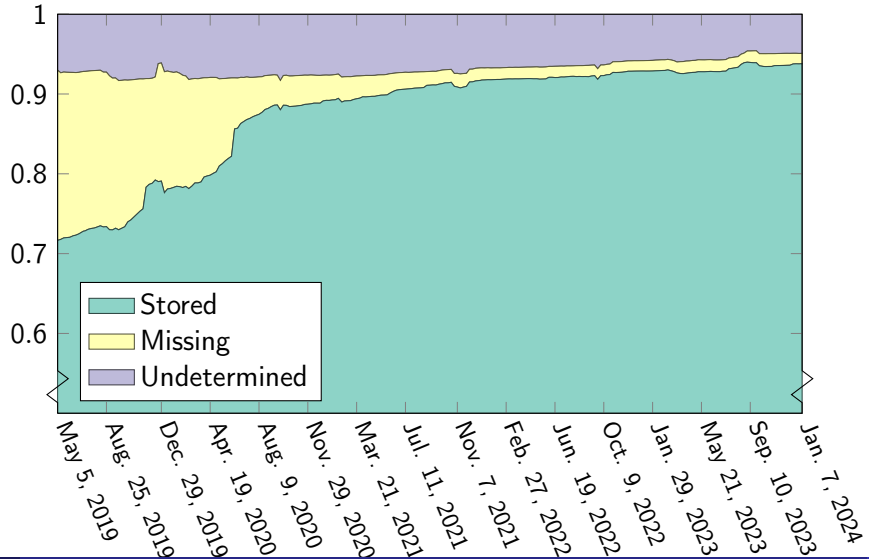
Case: VCS checkouts

- SWH addresses content as SWHID and associates the `nar-sh256` as *external identifier*.
- Guix queries using `nar-sh256` and gets back SWHID.
- Guix asks SWH Vault to "cook" the files and fetch them.

Case: *tarballs*

- Using Disarchive disassemble output, from `nar-sha256`, Guix gets SWHID.
- Guix asks SWH Vault to "cook" the files and fetch them.
- Using Disarchive disassemble output, Guix assembles bit-identical *compressed tarball*.

Coverage by sampled Guix revision



Work in progress

```
guix time-machine --commit=v1.0.0 -- install r-harmony
```

Installs (and potentially rebuilds) Harmony defined in Guix 1.0.0 from 2019.

- This command exploits SWH support as it was in 2019: in its infancy.
- Recovery mechanism is itself improving over time.
- Mitigations:
  - Delegate downloading to the Guix build daemon.
    (special and dedicated "builders" as `builtin:download` or `builtin:git-download`)
  - Recover source code referenced by past revisions using present-day techniques.

- Support more archive formats including lzip, Zip, unusual gzip compression.
- Deal with (long) *cooking* time by SWH Vault,
  from minutes to days depending on artifact size and service load.

Conclusion

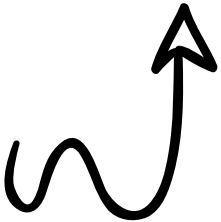
- Cite and reference source code using SWHID
- Use Guix!

The source of these slides is archived.
(Software Heritage id swh:1:dir:f5ed6543eabd6c5fb989c76412760088601b064b (link))

Toward better “Open Research”

Software Heritage

The ReScience Journal



Questions?

guix-science@gnu.org

swh-devel@inria.fr

https://hpc.guix.info

https://www.softwareheritage.org

These slides are archived.
(Software Heritage id swh:1:cnt:5926f6bb438a9b5c9e4914c137ad15101522179d (link))

Appendix

Example of VCS retrieval

```
building /gnu/store/agsi5ynwvmyfscc2avxkf7i3089m1p0i-scons-3.0.4-checkout.drv...
Initialized empty Git repository in /gnu/store/2p3cb96q8zk2pnarcnkwaifqw3l8gc70-scons-3.0.4-checkout/.git
fatal: unable to access 'https://github.com/SCons/scons.git/': Could not resolve host: github.com
Failed to do a shallow fetch; retrying a full fetch...
fatal: unable to access 'https://github.com/SCons/scons.git/': Could not resolve host: github.com
git-fetch: '/gnu/store/lcygm0p2d59acvwi12lwldg5c0d4czpr-git-minimal-2.41.0/bin/git fetch origin'
failed with exit code 128
Trying content-addressed mirror at bordeaux.guix.gnu.org...
Unable to fetch from bordeaux.guix.gnu.org, getaddrinfo-error: (-2)
Trying content-addressed mirror at ci.guix.gnu.org...
Unable to fetch from ci.guix.gnu.org, getaddrinfo-error: (-2)
Trying content-addressed mirror at bordeaux.guix.gnu.org...
Unable to fetch from bordeaux.guix.gnu.org, getaddrinfo-error: (-2)
Trying to download from Software Heritage...
SWH: found directory
with nar-sha256 hash 16a209173f87735020b29d84f497d44204cbcf86a451066342c51ff47996c8f7
at 'swh:1:dir:d3d1330dfc409be4624a01d384868fea0427c4c3'
swh:1:dir:d3d1330dfc409be4624a01d384868fea0427c4c3/
swh:1:dir:d3d1330dfc409be4624a01d384868fea0427c4c3/.appveyor.yml
...
successfully built /gnu/store/agsi5ynwvmyfscc2avxkf7i3089m1p0i-scons-3.0.4-checkout.drv
```

Example of *tarball* retrieval

```
Trying to use Disarchive to assemble /gnu/store/zwmrwlb2l53xbllxcw3axad54kyqcplp-Python-3.12.2.tar.xz...
Retrieving Disarchive spec
from https://disarchive.guix.gnu.org/sha256/be28112dac813d2053545c14bf13a16401a21877f1a69eb6ea5d84c4a0f3d
Assembling the directory Python-3.12.2
Downloading /gnu/store/zwmrwlb2l53xbllxcw3axad54kyqcplp-Python-3.12.2.tar.xz from Software Heritage...
SWH vault: requested bundle cooking, waiting for completion...
SWH vault: Processing...
swh:1:dir:72d77318a8c52ddfc004251fb7297799135704e6/
swh:1:dir:72d77318a8c52ddfc004251fb7297799135704e6/Python-3.12.2/pyconfig.h.in
...
Checking Python-3.12.2 digest... ok
Assembling the tarball Python-3.12.2.tar
Checking Python-3.12.2.tar digest... ok
Assembling the XZ file Python-3.12.2.tar.xz
Checking Python-3.12.2.tar.xz digest... ok
Copying result to /gnu/store/zwmrwlb2l53xbllxcw3axad54kyqcplp-Python-3.12.2.tar.xz
successfully built /gnu/store/nx97h7yr21l04nn60mqlf1yzfyxj06jh-Python-3.12.2.tar.xz.drv
source is at 'Python-3.12.2'
applying '/gnu/store/cdla0h7pcnckxlk3aflik3zsmbsfxzfp-python-3-deterministic-build-info.patch'...
applying '/gnu/store/ns40bs4bs19syckgh7v37rbxax0wfq01-python-3.12-fix-tests.patch'...
applying '/gnu/store/d7xlln76p372rssay94xkwbzf9p9p1rn-python-3-hurd-configure.patch'...
successfully built /gnu/store/y6s4xxsk5fc5909wafcvp2mxh5lc6c65-Python-3.12.2.tar.xz.drv
```

Example of *bugs* (read Missing in Coverage)

```
Trying to download from Software Heritage...
SWH: found directory
with nar-sha256 hash c98bd6991721d60b9a79600428bfbe8db0aaac3d383cd2df803ed7867c7cb63b
at 'swh:1:dir:218d95849f10fc0691d7dfa80999ce5061e654ef'
swh:1:dir:218d95849f10fc0691d7dfa80999ce5061e654ef/
...
swh:1:dir:218d95849f10fc0691d7dfa80999ce5061e654ef/wisp.py
r:sha256 hash mismatch for /gnu/store/7pcac04x82wyhknyfkdwhk3j958n2r75-guile-wisp-1.0.7-checkout:
  expected hash: 0fxngiy8dmryh3gx4g1q7nnamc4dpszjh130g6d0pmi12ycxd2y9
  actual hash:   0z7y487nnmw22xry82bb75shwp50gacm4kbwn01vhhli2bchpx37
hash mismatch for store item '/gnu/store/7pcac04x82wyhknyfkdwhk3j958n2r75-guile-wisp-1.0.7-checkout'
build of /gnu/store/8lcrd6n6m18hzh9dszm8c1xhjyfd54d9-guile-wisp-1.0.7-checkout.drv failed
View build log at '/var/log/guix/drvs/8l/crd6n6m18hzh9dszm8c1xhjyfd54d9-guile-wisp-1.0.7-checkout.drv.gz'
guix build: error: build of `/gnu/store/8lcrd6n6m18hzh9dszm8c1xhjyfd54d9-guile-wisp-1.0.7-checkout.drv' f
```

Discussion (SWH) #5093 (link)
(Guix) #71631 (link)

Fixed by `bd908af0c619cb1b74afeeb07839d7af08de9d91`