

# Software Heritage: Our Software Commons, Forever.

a status update

Nicolas Dandrimont, Stefano Zacchiroli

Inria, Software Heritage

10 August 2017

DebConf17 — Montreal, CA



# Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

- 
- 1 The Software Commons
  - 2 Software Heritage
  - 3 Architecture
  - 4 Gory details
  - 5 Community
  - 6 Conclusion

# Software source code is *special*

Harold Abelson, Structure and Interpretation of Computer Programs

*“Programs must be written for people to read, and only incidentally for machines to execute.”*

## Quake 2 source code (excerpt)

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y = number;
    i = * ( long * ) &y; // evil floating point bit level hacking
    i = 0x5f3759df - ( i >> 1 ); // what the fuck?
    y = * ( float * ) &i;
    y = y * ( threehalfs - ( x2 * y * y ) ); // 1st iteration
    // y = y * ( threehalfs - ( x2 * y * y ) ); // 2nd iteration, this
    // can be removed

    return y;
}
```

## Net. queue in Linux (excerpt)

```
/*
 * SFB uses two B[l][n] : L x N arrays of bins (L levels, N bins per level)
 * This implementation uses L = 8 and N = 16
 * This permits us to split one 32bit hash (provided per packet by rxhash or
 * external classifier) into 8 subhashes of 4 bits.
 */
#define SFB_BUCKET_SHIFT 4
#define SFB_NUMBUCKETS (1 << SFB_BUCKET_SHIFT) /* N bins per Level */
#define SFB_BUCKET_MASK (SFB_NUMBUCKETS - 1)
#define SFB_LEVELS (32 / SFB_BUCKET_SHIFT) /* L */

/* SFB also uses a virtual queue, named "bin" */
struct sfb_bucket {
    u16      qlen; /* length of virtual queue */
    u16      p_mark; /* marking probability */
};
```

Len Shustek, Computer History Museum

*“Source code provides a view into the mind of the designer.”*

## Definition (Commons)

The **commons** is the cultural and natural resources accessible to all members of a society, including natural materials such as air, water, and a habitable earth. These resources are held in common, not owned privately. <https://en.wikipedia.org/wiki/Commons>

## Definition (Software Commons)

The **software commons** consists of all computer software which is available at little or no cost and which can be altered and reused with few restrictions. Thus *all open source software and all free software are part of the [software] commons.* [...]

[https://en.wikipedia.org/wiki/Software\\_Commons](https://en.wikipedia.org/wiki/Software_Commons)

## Definition (Commons)

The **commons** is the cultural and natural resources accessible to all members of a society, including natural materials such as air, water, and a habitable earth. These resources are held in common, not owned privately. <https://en.wikipedia.org/wiki/Commons>

## Definition (Software Commons)

The **software commons** consists of all computer software which is available at little or no cost and which can be altered and reused with few restrictions. Thus *all open source software and all free software are part of the [software] commons.* [...]

[https://en.wikipedia.org/wiki/Software\\_Commons](https://en.wikipedia.org/wiki/Software_Commons)

*Source code is a precious part of our commons*

are we taking care of it?



A word cloud of terms related to software fragility, including: damage, disaster, malicious, obsolete, attack, dependencies, deletion, reference, storage, dangling, wear, corruption, encryption, format, media, aging, and tear.

Like all digital information, FOSS is fragile

- inconsiderate and/or malicious code loss (e.g., Code Spaces)
- business-driven code loss (e.g., Gitorious, Google Code)
- for obsolete code: physical media decay (data rot)



A word cloud of terms related to software fragility and digital preservation. The most prominent words are 'damage', 'disaster', 'malicious', 'obsolete', 'attack', 'deletion', and 'format'. Other visible words include 'media', 'aging', 'tear', 'dependencies', 'dangling', 'wear', 'corruption', 'reference', 'storage', and 'encryption'. The words are arranged in a cluster, with some overlapping, and are set against a faint world map background.



Like all digital information, FOSS is fragile

- inconsiderate and/or malicious code loss (e.g., Code Spaces)
- business-driven code loss (e.g., Gitorious, Google Code)
- for obsolete code: physical media decay (data rot)

Where is the archive...

where we go if (a repository on) GitHub or GitLab.com goes away?



A wealth of software research on crucial issues...

- safety, security, test, verification, proof
- software engineering, software evolution
- big data, machine learning, empirical studies



# Software lacks its own research infrastructure



A wealth of software research on crucial issues...

- safety, security, test, verification, proof
- software engineering, software evolution
- big data, machine learning, empirical studies

If you study the stars, you go to Atacama...

... where is the *very large telescope* of source code?

- 
- 1 The Software Commons
  - 2 Software Heritage
  - 3 Architecture
  - 4 Gory details
  - 5 Community
  - 6 Conclusion



## Software Heritage

THE GREAT LIBRARY OF SOURCE CODE



### Our mission

**Collect**, **preserve** and **share** the *source code* of *all the software* that is publicly available.

### Past, present and future

*Preserving the past, enhancing the present, preparing the future.*

# Our principles

**Cultural Heritage**



**Industry**



**Research**



**Education**



Software Heritage

**Cultural Heritage**



**Industry**



**Research**



**Education**



## Software Heritage

### Open approach

- 100% FOSS
- transparency

### In for the long haul

- replication
- non profit

- 
- 1 The Software Commons
  - 2 Software Heritage
  - 3 Architecture**
  - 4 Gory details
  - 5 Community
  - 6 Conclusion

# Archiving goals

Targets: VCS repositories & source code releases (e.g., tarballs)

## We DO archive

- file **content** (= blobs)
- **revisions** (= commits), with full metadata
- **releases** (= tags), ditto
- where (**origin**) & when (**visit**) we found any of the above

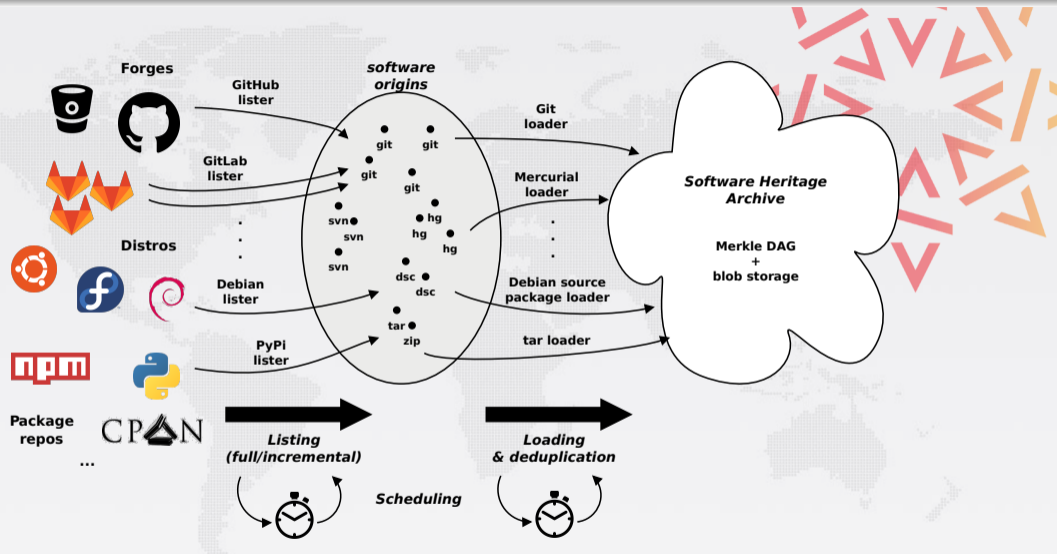
... in a VCS-/archive-agnostic **canonical data model**

## We DON'T archive

- homepages, wikis
- BTS/issues/code reviews/etc.
- mailing lists

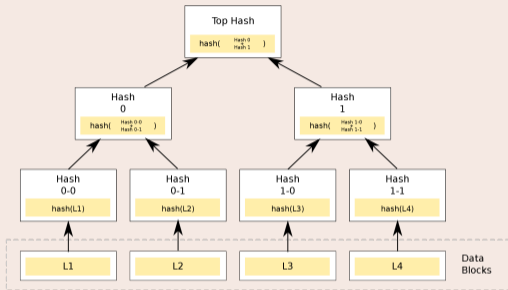
Long term vision: play our part in a *"semantic wikipedia of software"*

# Data flow





## Merkle tree (R. C. Merkle, Crypto 1979)

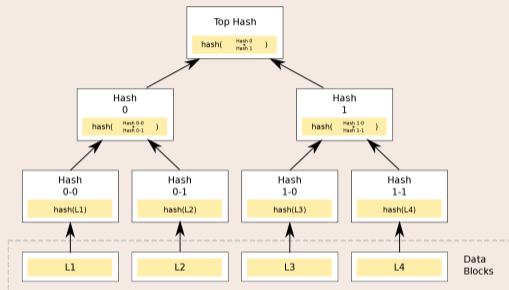


Combination of

- tree
- hash function

# Merkle trees

## Merkle tree (R. C. Merkle, Crypto 1979)



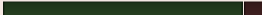
Combination of

- tree
- hash function

## Classical cryptographic construction

- fast, parallel signature of large data structures
- widely used (e.g., Git, blockchains, IPFS, ...)
- built-in deduplication

## Revisions

Details	Changes	Files
SHA: 963634dca6ba5dc37e3ee426ba091092c267f9f6		
Author: <a href="#">Nicolas Dandrimont &lt;nicolas@dandrimont.eu&gt;</a> (Thu Sep 1 14:26:13 2016)		
Committer: <a href="#">Nicolas Dandrimont &lt;nicolas@dandrimont.eu&gt;</a> (Thu Sep 1 14:26:13 2016)		
Subject: <code>provenance.tasks: add the revision -&gt; origin cache task</code>		
Parent: <a href="#">fc3a8b59ca1df424d860f2c29ab07fee4dc35d10</a> : <i>test...storage: property pipeline origin and cont...</i>		
<code>provenance.tasks: add the revision -&gt; origin cache task</code>		
<a href="#">swh/storage/provenance/tasks.py</a>  77		

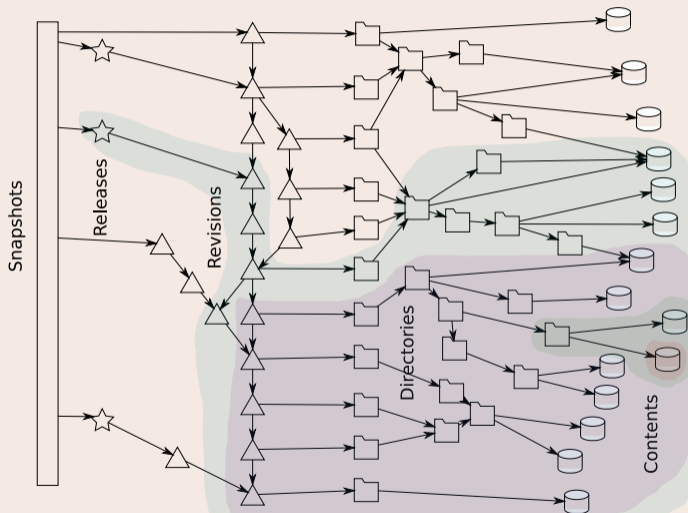


tree [515f00d44e92c65322aaa9bf3fa097c00ddb9c7d](#)  
parent [fc3a8b59ca1df424d860f2c29ab07fee4dc35d10](#)  
author Nicolas Dandrimont <nicolas@dandrimont.eu> 1472732773 +0200  
committer Nicolas Dandrimont <nicolas@dandrimont.eu> 1472732773 +0200

`provenance.tasks: add the revision -> origin cache task`

id: [963634dca6ba5dc37e3ee426ba091092c267f9f6](#)

# The archive: a (giant) Merkle DAG



# Archive coverage

## Our sources

- GitHub — full, up-to-date mirror
- Debian, GNU — one shot ingestion experiment (up to Aug 2015)
- Gitorious, Google Code — processing (Archive Team & Google)
- Bitbucket — WIP

# Archive coverage

## Our sources

- GitHub – full, up-to-date mirror
- Debian, GNU – one shot ingestion experiment (up to Aug 2015)
- Gitorious, Google Code – processing (Archive Team & Google)
- Bitbucket – WIP

## Some numbers



150 TB blobs, 5 TB database (as a graph: 7 B nodes + 60 B edges)

# Archive coverage

## Our sources

- GitHub – full, up-to-date mirror
- Debian, GNU – one shot ingestion experiment (up to Aug 2015)
- Gitorious, Google Code – processing (Archive Team & Google)
- Bitbucket – WIP

## Some numbers



150 TB blobs, 5 TB database (as a graph: 7 B nodes + 60 B edges)

The *richest* source code archive already, ... and growing daily!

First public version of our Web API (Feb 2017)

<https://archive.softwareheritage.org/api/>

## Features

- pointwise **browsing** of the Software Heritage archive
  - ... releases → revisions → directories → contents ...
- full access to the **metadata** of archived objects
- **crawling** information
  - *when have you last visited this Git repository I care about?*
  - *where were its branches/tags pointing to at the time?*

## Complete endpoint index

<https://archive.softwareheritage.org/api/1/>



# A tour of the Web API — origins & visits

```
GET https://archive.softwareheritage.org/api/1/origin/ \
    git/url/https://github.com/hylang/hy
{ "id": 1,
  "origin_visits_url": "/api/1/origin/1/visits/",
  "type": "git",
  "url": "https://github.com/hylang/hy"
}
```

```
GET https://archive.softwareheritage.org/api/1/origin/ \
    1/visits/
[ ...,
  { "date": "2016-09-14T11:04:26.769266+00:00",
    "origin": 1,
    "origin_visit_url": "/api/1/origin/1/visit/13/",
    "status": "full",
    "visit": 13
  }, ...
]
```



# A tour of the Web API — snapshots

```
GET https://archive.softwareheritage.org/api/1/origin/ \
  1/visit/13/
{ ...,
  "occurrences": { ...,
    "refs/heads/master": {
      "target": "b94211251...",
      "target_type": "revision",
      "target_url": "/api/1/revision/b94211251.../"
    },
    "refs/tags/0.10.0": {
      "target": "7045404f3...",
      "target_type": "release",
      "target_url": "/api/1/release/7045404f3.../"
    }, ...
  }, ...
},
"origin": 1,
"origin_url": "/api/1/origin/1/",
"status": "full",
"visit": 13
}
```



# A tour of the Web API — revisions

```
GET https://archive.softwareheritage.org/api/1/revision/  
6072557b6c10cd9a21145781e26ad1f978ed14b9/  
{  
  "author": {  
    "email": "tag@pault.ag",  
    "fullname": "Paul Tagliamonte <tag@pault.ag>",  
    "id": 96,  
    "name": "Paul Tagliamonte"  
  },  
  "committer": { ... },  
  "date": "2014-04-10T23:01:11-04:00",  
  "committer_date": "2014-04-10T23:01:11-04:00",  
  "directory": "2df4cd84e...",  
  "directory_url": "/api/1/directory/2df4cd84e.../",  
  "history_url": "/api/1/revision/6072557b6.../log/",  
  "merge": false,  
  "message": "0.10: The Oh f*ck it's PyCon release",  
  "parents": [ {  
    "id": "10149f66e...",  
    "url": "/api/1/revision/10149f66e.../"
```

# A tour of the Web API — contents

```
GET https://archive.softwareheritage.org/api/1/content/ \
  adc83b19e793491b1c6ea0fd8b46cd9f32e592fc/
{
  "data_url": "/api/1/content/sha1:adc83b19e.../raw/",
  "filetype_url": "/api/1/content/sha1:.../filetype/",
  "language_url": "/api/1/content/sha1:.../language/",
  "length": 1,
  "license_url": "/api/1/content/sha1:.../license/",
  "sha1": "adc83b19e...",
  "sha1_git": "8b1378917...",
  "sha256": "01ba4719c...",
  "status": "visible"
}
```



```
GET https://archive.softwareheritage.org/api/1/content/ \
    adc83b19e793491b1c6ea0fd8b46cd9f32e592fc/
{
  "data_url": "/api/1/content/sha1:adc83b19e.../raw/",
  "filetype_url": "/api/1/content/sha1:.../filetype/",
  "language_url": "/api/1/content/sha1:.../language/",
  "length": 1,
  "license_url": "/api/1/content/sha1:.../license/",
  "sha1": "adc83b19e...",
  "sha1_git": "8b1378917...",
  "sha256": "01ba4719c...",
  "status": "visible"
}
```

## Caveats

- rate limits apply throughout the API
- blob download available for selected contents

## Features...

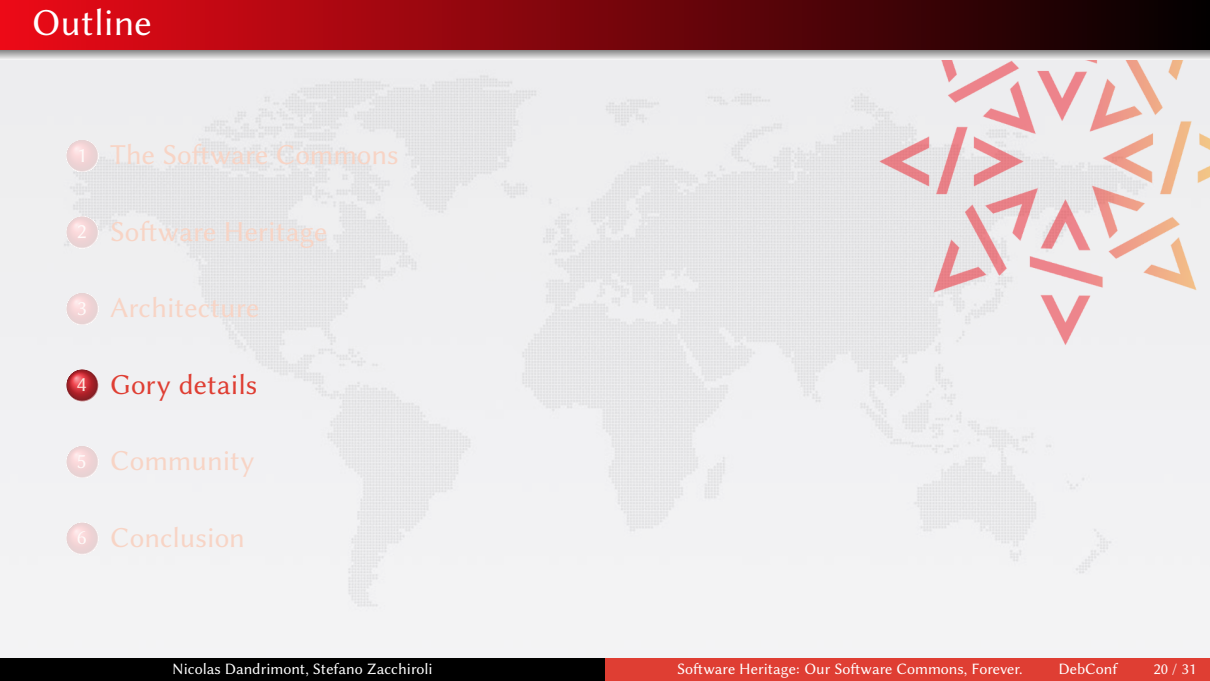
- (done) **lookup** by content hash
- **browsing**: "wayback machine" for archived code
  - (done) via Web API
  - (todo) via Web UI
- (todo) **download**: `wget / git clone` from the archive
- (todo) **deposit** of source code bundles directly to the archive
- (todo) **provenance** information for all archived content
- (todo) **full-text search** on all archived source code files

## Features...

- (done) **lookup** by content hash
- **browsing**: "wayback machine" for archived code
  - (done) via Web API
  - (todo) via Web UI
- (todo) **download**: `wget / git clone` from the archive
- (todo) **deposit** of source code bundles directly to the archive
- (todo) **provenance** information for all archived content
- (todo) **full-text search** on all archived source code files

... and much more than one could possibly imagine

all the world's software development history in a single graph!

- 
- 1 The Software Commons
  - 2 Software Heritage
  - 3 Architecture
  - 4 Gory details
  - 5 Community
  - 6 Conclusion



# Technology: how do you store the SWH DAG?

## Problem statement

- How would you store and query a graph with 10 billion nodes and 60 billion edges?
- How would you store the contents of more than 3 billion files, 300TB of raw data?
- on a limited budget (100 000 € of hardware overall)

# Technology: how do you store the SWH DAG?

## Problem statement

- How would you store and query a graph with 10 billion nodes and 60 billion edges?
- How would you store the contents of more than 3 billion files, 300TB of raw data?
- on a limited budget (100 000 € of hardware overall)

## Our hardware stack

- two hypervisors with 512GB RAM, 20TB SSD each, sharing access to a storage array (60 x 6TB spinning rust)
- one backup server with 48GB RAM and another storage array

## Our software stack

- A RDBMS (PostgreSQL, what else?), for storage of the graph nodes and edges
- filesystems for storing the actual file contents

## Metadata storage

- Python module `swh.storage`
- thin Python API over a pile of PostgreSQL functions
- motivation: keeping relational integrity at the lowest layer

## Content ("object") storage

- Python module `swh.objstorage`
- very thin object storage abstraction layer (PUT, APPEND and GET) over regular storage technologies
- separate layer for asynchronous replication and integrity management (`swh.archiver`)
- motivation: stay as technology neutral as possible for future mirrors

## Current primary deployment

- Storage on 16 sharded XFS filesystems; key = *sha1* (content), value = *gzip* (content)
- if sha1 = **abcdef01234...**, file path = / srv / storage / **a** / **ab** / **cd** / **ef** / **abcdef01234...**
- 3 directory levels deep, each level 256-wide = 16 777 216 directories (1 048 576 per partition)

## Secondary deployment

- Storage on Azure blob storage
- 16 storage containers, objects stored in a flat structure there

## Generic model is fine

The abstraction layer is fairly simple and generic, and the implementation of the upper layers (replication, integrity checking) was a breeze.

## Filesystem implementation is bad

Slow spinning storage + little RAM (48GB) + 16 million dentries = (very) bad performance

## Current deployment

- PostgreSQL deployed in primary/replica mode, using pg\_logical for replication: different indexes on primary (tuned for writes) and replicas (tuned for reads).
- most logic done in SQL
- thin Pythonic API over the SQL functions

## end goals

- proper handling of relations between objects at the lowest level
- doing fast recursive queries on the graph (e.g. find the provenance info for a content, walking up the whole graph, in one single query)

Limited resources

PostgreSQL works really well

Limited resources

PostgreSQL works really well ... until your indexes don't fit in RAM



## Limited resources

PostgreSQL works really well ... until your indexes don't fit in RAM

Our recursive queries jump between different object types, and between evenly distributed hashes. Data locality doesn't exist. Caches break down.

## Limited resources

PostgreSQL works really well ... until your indexes don't fit in RAM

Our recursive queries jump between different object types, and between evenly distributed hashes. Data locality doesn't exist. Caches break down.

Massive deduplication = efficient storage

## Limited resources

PostgreSQL works really well ... until your indexes don't fit in RAM

Our recursive queries jump between different object types, and between evenly distributed hashes. Data locality doesn't exist. Caches break down.

Massive deduplication = efficient storage

**but** Massive deduplication = exponential width for recursive queries

## Limited resources

PostgreSQL works really well ... until your indexes don't fit in RAM

Our recursive queries jump between different object types, and between evenly distributed hashes. Data locality doesn't exist. Caches break down.

Massive deduplication = efficient storage

**but** Massive deduplication = exponential width for recursive queries

## Reality check

Referential integrity?

## Limited resources

PostgreSQL works really well ... until your indexes don't fit in RAM

Our recursive queries jump between different object types, and between evenly distributed hashes. Data locality doesn't exist. Caches break down.

Massive deduplication = efficient storage

**but** Massive deduplication = exponential width for recursive queries

## Reality check

Referential integrity? Real repositories downloaded from the internet are all kinds of broken.

## Object storage

Our azure prototype shows that using a scale-out "cloudy" technology for our object storage works really well. Plain filesystems on spinning rust, not so much.

## Object storage

Our azure prototype shows that using a scale-out "cloudy" technology for our object storage works really well. Plain filesystems on spinning rust, not so much. We need to investigate other storage tech (ceph, swift, ...) for our main copy of the archive as our budget ramps up.

## Object storage

Our azure prototype shows that using a scale-out "cloudy" technology for our object storage works really well. Plain filesystems on spinning rust, not so much. We need to investigate other storage tech (ceph, swift, ...) for our main copy of the archive as our budget ramps up.

## Metadata storage

Our initial assumption that we wanted referential integrity and built-in recursive queries was wrong.



## Object storage

Our azure prototype shows that using a scale-out "cloudy" technology for our object storage works really well. Plain filesystems on spinning rust, not so much. We need to investigate other storage tech (ceph, swift, ...) for our main copy of the archive as our budget ramps up.

## Metadata storage

Our initial assumption that we wanted referential integrity and built-in recursive queries was wrong. We could probably migrate to "dumb" object storages for each type of object, with another layer to check metadata integrity regularly.

- 
- 1 The Software Commons
  - 2 Software Heritage
  - 3 Architecture
  - 4 Gory details
  - 5 **Community**
  - 6 Conclusion

# You can help!

## Coding

- [www.softwareheritage.org/community/developers/](http://www.softwareheritage.org/community/developers/)
- [forge.softwareheritage.org](http://forge.softwareheritage.org) – our own code

## Current development priorities

- ★★★ listers for unsupported forges, distros, pkg. managers
- ★★★ loaders for unsupported VCS, source package formats
- ★★ Web UI: eye candy wrapper around the Web API
- ★ content indexing and search

... *all* contributions equally welcome!

# You can help!

## Coding

- [www.softwareheritage.org/community/developers/](http://www.softwareheritage.org/community/developers/)
- [forge.softwareheritage.org](http://forge.softwareheritage.org) – **our own code**

## Current development priorities

- ★★★ listers for unsupported forges, distros, pkg. managers
- ★★★ loaders for unsupported VCS, source package formats
- ★★ Web UI: eye candy wrapper around the Web API
- ★ content indexing and search

... *all* contributions equally welcome!

## Join us

- [www.softwareheritage.org/jobs](http://www.softwareheritage.org/jobs) – **job openings**
- [wiki.softwareheritage.org](http://wiki.softwareheritage.org) – **internships**

# Sharing the Software Heritage vision



See more

<http://www.softwareheritage.org/support/testimonials>

# Sponsoring Software Heritage work

*Inria*  
INVENTEURS DU MONDE NUMÉRIQUE



Microsoft



SOCIÉTÉ  
GÉNÉRALE



HUAWEI

Data Archiving and Networked Services

DANS

NOKIA Bell Labs



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA  
DIPARTIMENTO DI INFORMATICA - SCIENZA E INGEGNERIA

April 3rd, 2017: landmark UNESCO/Inria agreement...



[www.softwareheritage.org/?p=11623](http://www.softwareheritage.org/?p=11623)

**Next step:** 27-28 Sep 2017: UNESCO/Inria conference in Paris

- 
- 1 The Software Commons
  - 2 Software Heritage
  - 3 Architecture
  - 4 Gory details
  - 5 Community
  - 6 Conclusion



## Software Heritage is

- a *reference archive* of all FOSS ever written
- a unique *complement* for *development platforms*
- an international, open, nonprofit, *mutualized infrastructure*
- at the service of our community, at the service of society

## References

Roberto Di Cosmo, Stefano Zacchiroli. *Software Heritage: Why and How to Preserve Software Source Code*. To appear, iPRES 2017, Kyoto, Sep 2017. Preprint: <http://deb.li/swhipres17>

## Come in, we're open!

[www.softwareheritage.org](http://www.softwareheritage.org) – *sponsoring, job openings*

[wiki.softwareheritage.org](http://wiki.softwareheritage.org) – *internships, leads*

[forge.softwareheritage.org](http://forge.softwareheritage.org) – *our own code*

## Q: how about SHA1 collisions?

```
create domain sha1 as bytea
  check (length(value) = 20);
create domain sha1_git as bytea
  check (length(value) = 20);
create domain sha256 as bytea
  check (length(value) = 32);

create table content (
  sha1      sha1 primary key,
  sha1_git  sha1_git not null,
  sha256    sha256 not null,
  length    bigint not null,
  ctime     timestampz not null default now(),
  status    content_status not null default 'visible',
  object_id bigserial
);

create unique index on content(sha1_git);
create unique index on content(sha256);
```