

# Software Heritage

Archiving Free/Open Source Software for Fun & Profit

Stefano Zacchioli

Software Heritage – [zack@upsilon.cc](mailto:zack@upsilon.cc), [@zacchiro](https://twitter.com/zacchiro)

5 November 2018

Google – New York City, USA



# Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

- Computer Science Professor, University Paris Diderot (France)
- Free Software activist (20+ years)
- Debian Developer & Former 3x Debian Project Leader
- Former Open Source Initiative (OSI) director
- Software Heritage co-founder & CTO

- 
- 1 Software Heritage
  - 2 Accessing the archive
  - 3 R&D challenges
  - 4 Getting involved

# (Free) Software is everywhere



# 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 III 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?



## Fashion victims

- many disparate development platforms
- a myriad places where distribution may happen
- projects tend to migrate from one place to another over time



# Software is spread all around



## Fashion victims

- many disparate development platforms
- a myriad places where distribution may happen
- projects tend to migrate from one place to another over time

## Where is the place ...

where we can find, track and search *all* source code?



A word cloud of terms related to software fragility, including: damage, disaster, malicious, deletion, reference, storage, attack, obsolete, dependencies, dangling, wear, corruption, encryption, format, aging, media, 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', 'attack', 'obsolete', 'deletion', and 'format'. Other smaller words include 'media', 'aging', 'tear', 'dependencies', 'reference', 'storage', 'dangling', 'wear', 'corruption', and 'encryption'. The words are arranged in a cluster, with 'damage' and 'disaster' being the largest.

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?



## 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.*

# Core principles

**Cultural Heritage**



**Industry**



**Research**



**Education**



Software Heritage

**Cultural Heritage**



**Industry**



**Research**



**Education**



**Software Heritage**

Open approach

- 100% Free Software
- transparency



**Cultural Heritage**



**Industry**



**Research**



**Education**



**Software Heritage**

**Open approach**

- 100% Free Software
- transparency

**In for the long haul**

- replication
- non profit

# Archiving goals

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

## We DO archive

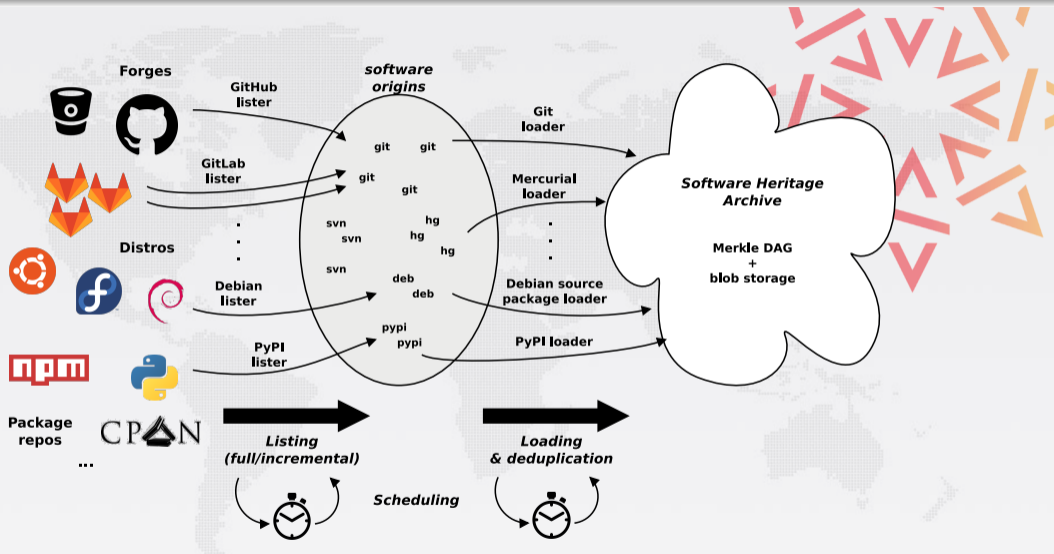
- file **content** (= blobs)
- **revisions** (= commits), with full metadata
- **releases** (= tags), ditto
- where (**origin**) & when (**visit**) we found any of the above

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

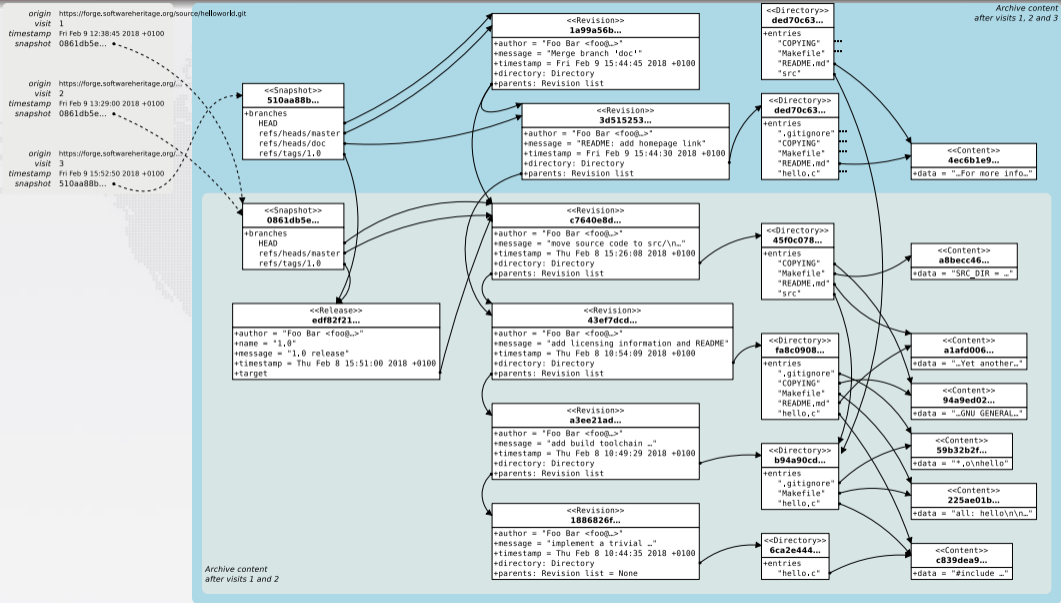
## We DON'T archive

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

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

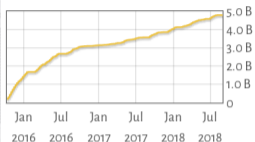


# The archive: a (giant) Merkle DAG



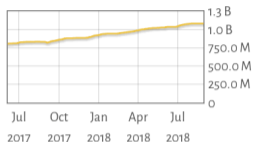
## Source files

5,011,613,861



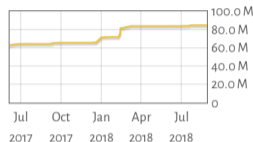
## Commits

1,126,348,335



## Projects

85,202,432



## Current sources

- live: GitHub, Debian, GitLab.com, PyPI
- one-off: Gitorious, Google Code, GNU
- WIP: Bitbucket



## Current sources

- live: GitHub, Debian, GitLab.com, PyPI
- one-off: Gitorious, Google Code, GNU
- WIP: Bitbucket

200 TB (compressed) blobs, 6 TB database (as a graph: 10 B nodes + 100 B edges)



## Current sources

- live: GitHub, Debian, GitLab.com, PyPI
- one-off: Gitorious, Google Code, GNU
- WIP: Bitbucket

200 TB (compressed) blobs, 6 TB database (as a graph: 10 B nodes + 100 B edges)

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

- 
- 1 Software Heritage
  - 2 Accessing the archive
  - 3 R&D challenges
  - 4 Getting involved



RESTful API to programmatically access the Software Heritage archive

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

## Features

- pointwise **browsing** of the archive
  - ... snapshots → 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?*

## 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
- raw download available for textual contents

## Vault service

- source code is thoroughly deduplicated within the Software Heritage archive
- bulk download of large artefacts (e.g., a Linux kernel release) requires collecting millions of objects
- the **Software Heritage Vault** cooks and caches source code bundles for bulk download needs

## Tech bits

- **RESTful API** to request downloads, notifications, and monitoring
- `docs.softwareheritage.org/devel/swh-vault`

Browser-based interface to browse the Software Heritage archive

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

## Features

- all **REST API features**, but good looking :-)
  - browsing: snapshots → revisions → directories → contents ...
  - access to metadata and crawling information
- **origin search**, as full text indexing of origin URLs
- bulk **download**, via integration with the Vault



Browser-based interface to browse the Software Heritage archive

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

## Features

- all **REST API features**, but good looking :-)
  - browsing: snapshots → revisions → directories → contents ...
  - access to metadata and crawling information
- **origin search**, as full text indexing of origin URLs
- bulk **download**, via integration with the Vault

# Demo

- 
- 1 Software Heritage
  - 2 Accessing the archive
  - 3 R&D challenges
  - 4 Getting involved

## The real world sucks

- corrupted repositories
- takedown notices
- partial irrecoverable data losses

## The real world sucks

- corrupted repositories
- takedown notices
- partial irrecoverable data losses

## *Incomplete Merkle DAGs*

- nodes can go missing at archival time or disappear later on
- top-level hash(es) no longer capture the full state of the archive

## The real world sucks

- corrupted repositories
- takedown notices
- partial irrecoverable data losses

## *Incomplete* Merkle DAGs

- nodes can go missing at archival time or disappear later on
- top-level hash(es) no longer capture the full state of the archive

## Open questions

- how do you capture such full state then?
- how do you efficiently check if something is to be re-archived?
- ultimately, what's your notion of having "fully archived" something?

## Archive stats

- as a graph: ~10 B nodes, ~100 B edges
- nodes breakdown: ~40% contents, ~40% directories, ~10% commits
- content size: ~400 TB (raw), ~200 TB compressed (content by content)
- median compressed size: 3 KB
- i.e., **a lot of very small files**

## Archive stats

- as a graph: ~10 B nodes, ~100 B edges
- nodes breakdown: ~40% contents, ~40% directories, ~10% commits
- content size: ~400 TB (raw), ~200 TB compressed (content by content)
- median compressed size: 3 KB
- i.e., **a lot of very small files**

## Current storage solution (unsatisfactory)

- contents: ad hoc object storage with multiple backends
  - file-system, Azure, AWS, etc.
- rest of the graph: Postgres (~6 TB)
  - rationale: recursive queries to traverse the graph
  - (no, it doesn't work at this scale)

## Requirements

- long-term storage
- suitable for distribution/replica
- suitable for scale-out processing



## Requirements

- long-term storage
- suitable for distribution/replica
- suitable for scale-out processing

## Graph

- early experiences with Ceph (RADOS)
  - not a good fit out of the box
  - 7x size increase over target retention policy due to large minimum chunk size (64 KB)
- ad-hoc object packing (?)
- .oO( do we really have to re-invent a file-system? )

## Contents — size considerations

- a few hundreds TB is not *that* big, but it cuts off volunteer mirrors

## Contents — size considerations

- a few hundreds TB is not *that* big, but it cuts off volunteer mirrors

## Content compression

- low compression ration (2x) with 1-by-1 compression
- typical Git/VCS packing heuristics do not work here, because contents occur in many different contexts
- early experiences with Rabin-style compression & co. were unsatisfactory

## Contents — size considerations

- a few hundreds TB is not *that* big, but it cuts off volunteer mirrors

## Content compression

- low compression ration (2x) with 1-by-1 compression
- typical Git/VCS packing heuristics do not work here, because contents occur in many different contexts
- early experiences with Rabin-style compression & co. were unsatisfactory

## Distributed archival

- massively distributed archival (e.g., P2P) would be nice
- but most P2P techs are more like CDNs than archives and do not offer retention policy guarantees (e.g., self-healing)

## Use cases


- Vault: recursive visits to collect archived objects
- Provenance: single-destination shortest path

## Use cases

- Vault: recursive visits to collect archived objects
- Provenance: single-destination shortest path

## Technology

- beyond the capabilities of off-the-shelf graph DBs
- graph topology: scale-free, but not small world
- *probably* bad fit for Pregel/Chaoss/etc
- are web graph style compression techniques suitable for storing and processing the Merkle DAG in memory? (unclear)

- 
- 1 Software Heritage
  - 2 Accessing the archive
  - 3 R&D challenges
  - 4 Getting involved

## Features...

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



## Features...

- (done) **lookup** by content hash
- (done) **browsing**: "wayback machine" for source code (API + UI)
- (early access) **deposit** of source code bundles directly to the archive
- (early access) **save code now**, on-demand archive
- (done) **download**: `wget / git clone` from the archive
- (todo) **provenance** lookup 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 at hand's reach!

## Links

- [www.softwareheritage.org](http://www.softwareheritage.org) – general information
- [archive.softwareheritage.org](http://archive.softwareheritage.org) – the archive
- [forge.softwareheritage.org](http://forge.softwareheritage.org) – our own code

## Bibliography



Jean-François Abramatic, Roberto Di Cosmo, Stefano Zacchiroli  
Building the Universal Archive of Source Code  
Communication of the ACM, October 2018



Roberto Di Cosmo, Stefano Zacchiroli  
Software Heritage: Why and How to Preserve Software Source Code  
iPRES 2017: Intl. Conf. on Digital Preservation



Roberto Di Cosmo, Morane Gruenpeter, Stefano Zacchiroli  
Identifiers for Digital Objects: the Case of Software Source Code Preservation  
iPRES 2018: Intl. Conf. on Digital Preservation