# Software Heritage

## Towards A New Era for Software Engineering, Cybersecurity, and AI

Roberto Di Cosmo

Director, Software Heritage
Inria and Université Paris Cité

May 21st, 2026
Colloquium Sorbonne Université

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

# Outline

# Software *Source Code* is Precious Knowledge

Harold Abelson, Structure and Interpretation of Computer Programs (1st ed.) 1985

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

## Apollo 11 source code (excerpt)

```
P63SPOT3        CA      BIT6            # IS THE LR ANTENNA IN POSITION 1 YET
                EXTEND
                RAND    CHAN33
                EXTEND
                BZF     P63SPOT4        # BRANCH IF ANTENNA ALREADY IN POSITION 1

                CAF     CODE500         # ASTRONAUT:    PLEASE CRANK THE
                TC      BANKCALL        #               SILLY THING AROUND
                CADR    GOPERF1
                TCF     GOTOPOOH        # TERMINATE
                TCF     P63SPOT3        # PROCEED       SEE IF HE'S LYING

P63SPOT4        TC      BANKCALL        # ENTER         INITIALIZE LANDING RADAR
                CADR    SETPOS1

                TC      POSTJUMP        # OFF TO SEE THE WIZARD ...
                CADR    BURNBABY
```

## Covid Sim ( excerpt )

```
/**
 * @brief The basic unit of the simulation and is associated to a geographical location.
 *
 * Interventions (e.g., school closures) are tracked at this level. It contains a list of its
 * members (people), places (schools, universities, workplaces etc.), road networks, links to
 * airports etc.
 */
struct Microcell
{
        /* Note use of short int here limits max run time to USHRT_MAX*ModelTimeStep - e.g. 65536*0.25=16384 days=44 yrs.
           Global search and replace of 'unsigned short int' with 'int' would remove this limit, but use more memory.
        */

        int n; // Number of people in microcell
        int adunit; // admin unit microcell belongs to
        int* members; // array of members/hosts of microcell

        int* places[MAX_NUM_PLACE_TYPES]; // list of places (of various place types) within microcell
        unsigned short int NumPlacesByType[MAX_NUM_PLACE_TYPES]; // number of places (of various place types) within mircocell
        unsigned short int keyworkerproph, move_trig, place_trig, socdist_trig, keyworkerproph_trig;
        unsigned short int move_start_time, move_end_time;
        unsigned short int place_end_time, socdist_end_time, keyworkerproph_end_time;
        TreatStat moverest, treat, vacc, socdist, placeclose;
        unsigned short int treat_trig, vacc_trig;
        unsigned short int treat_start_time, treat_end_time;
        unsigned short int vacc_start_time;
        IndexList* Airportlist;
};
```

Len Shustek, Computer History Museum 2006

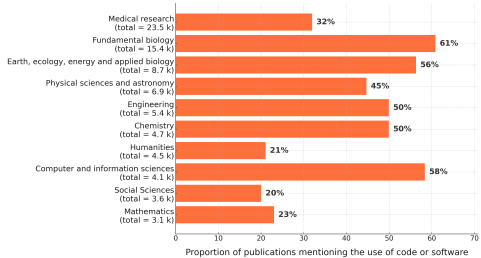
*"Source code provides a view into the mind of the designer."*

# (Open Source) Software is *precious technical and scientific knowledge*

## Yuval Noah Harari (on COVID 19)

*"The real antidote [to epidemic] is* scientific knowledge *and* global cooperation."

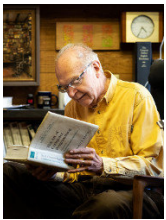
## Software powers modern research




20%+ articles use software, all disciplines

2025 French Open Science Monitor

## We can still talk to the early inventors

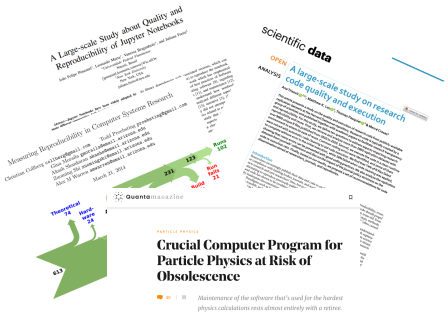


*"Telling historical stories is the best way to teach. It's much easier to understand something if you know the threads it is connected to."*

Donald E. Knuth
Len Shustek
CACM, January 2021

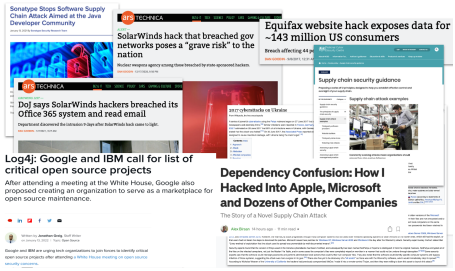
# How are we managing our (open source) software?

## Reproducibility, maintenance in Academia




(articles: here, here, here and here)

## Security, integrity, traceability in Industry




Can they track the software that they

- ship, use, acquire
- has that bug or vulnerability

We need a *dedicated infrastructure* to adress *all* this!

# Outline

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Inria with unesco

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

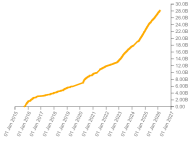
Inria with unesco

# The largest open source code archive: one infrastructure, open, shared, non profit

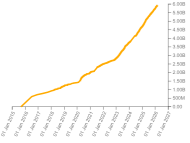
Unique digital common good *built in France since 2015*

**Cultural Heritage** | **Industry** | **Research** | **Public Administration**

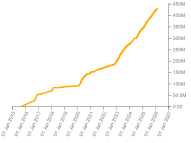
Source files
28,152,651,759



Commits
5,920,787,012



Projects
429,963,770



Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Inria with unesco

# The largest open source code archive: one infrastructure, open, shared, non profit

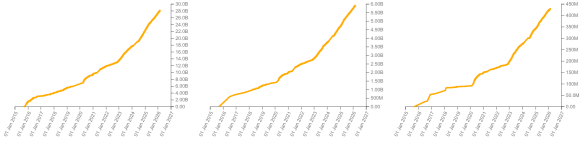
Unique digital common good *built in France since 2015*

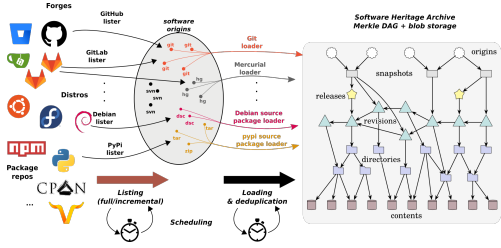
**Cultural Heritage** | **Industry** | **Research** | **Public Administration**

Source files
28,152,651,759

Commits
5,920,787,012

Projects
429,963,770






**5000+ platforms**

**All versions, all history development in a single graph**

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

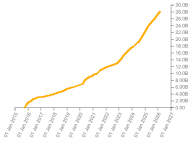
Inria with unesco

# The largest open source code archive: one infrastructure, open, shared, non profit

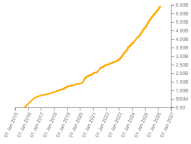
Unique digital common good *built in France since 2015*

**Cultural Heritage** | **Industry** | **Research** | **Public Administration**

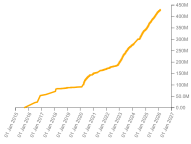
Source files
28,152,651,759

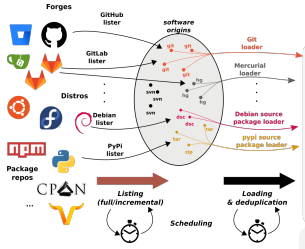


Commits
5,920,787,012



Projects
429,963,770






**5000+ platforms**

**All versions, all history development in a single graph**

- **50 × 10⁹** nodes
- **1000 × 10⁹** edges
- ~ **3 PB** of storage

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Inria with unesco

# The largest open source code archive: one infrastructure, open, shared, non profit

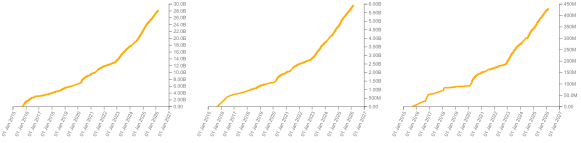
Unique digital common good *built in France since 2015*

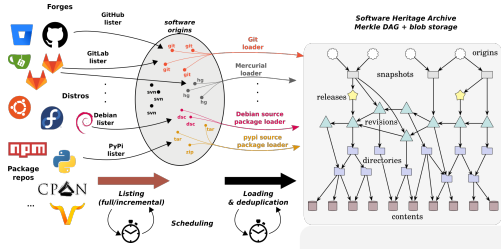
**Cultural Heritage** | **Industry** | **Research** | **Public Administration**

Source files
28,152,651,759

Commits
5,920,787,012

Projects
429,963,770






**5000+ platforms**

**All versions, all history development in a single graph**

- **50 × 10⁹** nodes
- **1000 × 10⁹** edges
- ~ **3 PB** of storage

A revolutionary **infrastructure** ensures **availability** guarantees **integrity** enables **traceability**

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

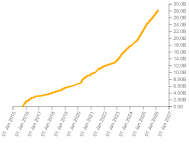
Inria with unesco

# The largest open source code archive: one infrastructure, open, shared, non profit

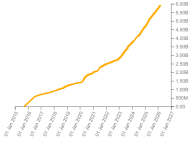
Unique digital common good *built in France since 2015*

**Cultural Heritage** | **Industry** | **Research** | **Public Administration**

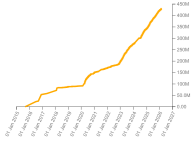
Source files
28,152,651,759

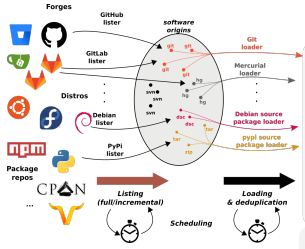


Commits
5,920,787,012



Projects
429,963,770






**5000+ platforms**

**All versions, all history development in a single graph**

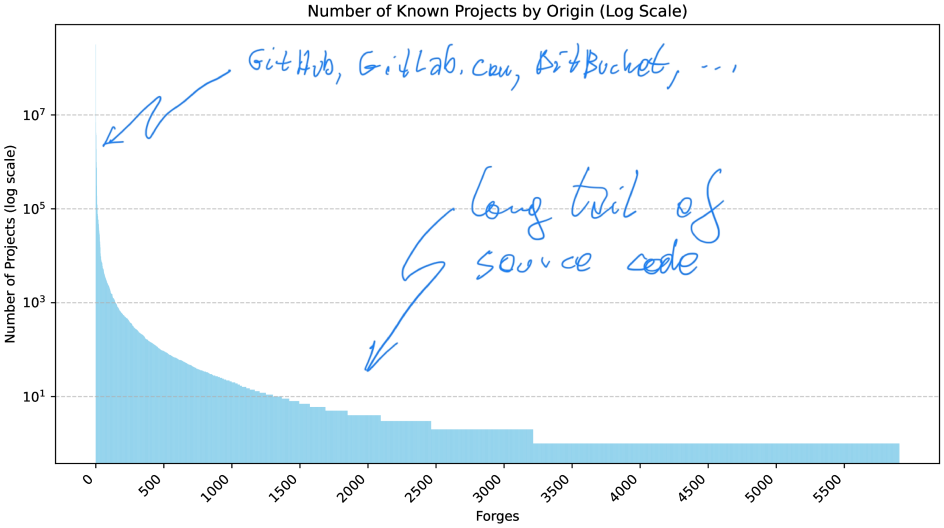
- **50 × 10⁹** nodes
- **1000 × 10⁹** edges
- ~ **3 PB** of storage

A revolutionary **infrastructure** ensures **availability** guarantees **integrity** enables **traceability**

IBM Microsoft TII Technology Innovation Institute HUAWEI cnrs Infosys cea Ministère de l'Enseignement Supérieur et de la Recherche aws Google Université Paris Cité openinventionnetwork

AdaCore GitHub Université di Pisa Sorbonne Université Université de Lorraine Sant'Anna NWO université PARIS-SACLAY

# A peek at the code hosting landscape

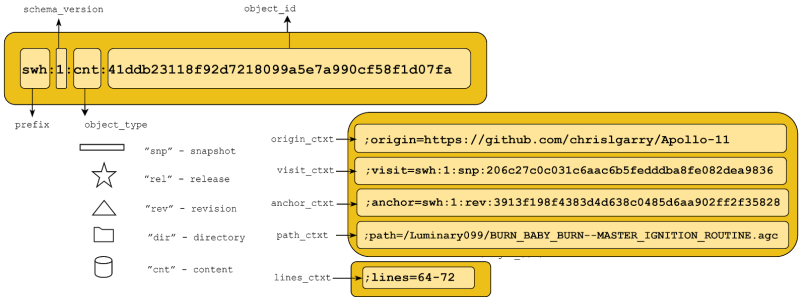



Over 2 million projects in the long tail!

# The Software Hash identifier (SWHID)

## Software Heritage Identifiers (SWHID)

see swhid.org




50+B
intrinsic,
decentralised,
cryptographic

Full fledged *source code references* for traceability, integrity and reproducibility

- Linux Foundation SPDX 2.2
- IANA-registered "swh:"
- WikiData property P6138

Examples: Apollo 11 AGC excerpt, Quake III rsqrt
Guidelines available, see the HOWTO

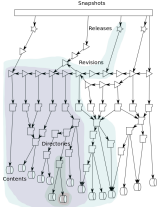
ISO/IEC 18670, see swhid.org

# Outline

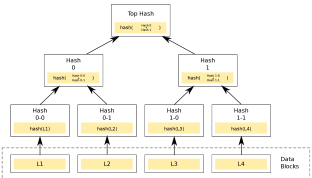
# From archive to revolutionary infrastructure

Modern "Library of Alexandria", *international, non profit, long term* initiative
addressing the needs of *industry, research, culture and society as a whole*

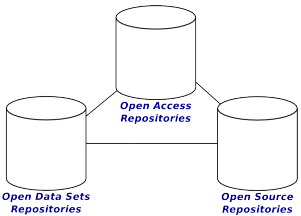
*Software Graph*



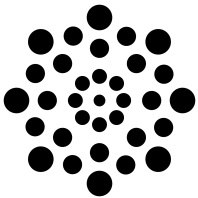

*Software Blockchain*




*Open Science* pillar




*Big Code*



Today we focus on

**Open Science • Large-scale analysis • Cybersecurity • AI**

# Outline

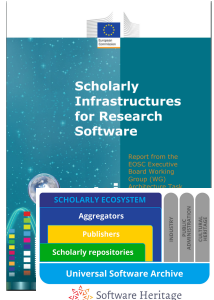
# An example is worth a thousand words

- Browse (e.g. Apollo 11 [excerpt], your work may be already there !)
- Trigger archival, use the updateswh browser extension, configure the webhooks
- Get and use SWHIDs (full specification available online)
- Cite software with biblatex-software package from CTAN
  - Overleaf ACMART template available
- Example in journals: article from IPOL
- Example with Parmap: devel on Github, archive in SWH, curated deposit in HAL
- Extracting all the software products for Inria, for CNRS, for CNES, for LIRMM or for Rémi Gribonval using HalTools
- Curated deposit in SWH via HAL, see for example: LinBox, SLALOM, Givaro, NS2DDV, SumGra, Coq proof, ...
- Example use in research articles:
  - compare Fig. 1 and conclusions in the 2012 version and the updated version
  - SWHID in a replication experiment

# A few adoption indicators

## Policy






- Recommendations in ANR 2023 guidelines (p. 17)
- HAL+SWH in the Open Science software booklet

## Projects

eosc | FAIRCORE4EOSC
Core Components Supporting a FAIR EOSC

The CodeMeta Project

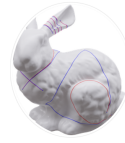
eosc | FAIR-IMPACT
Expanding FAIR solutions across EOSC

## Users and collaborations

### What are they “referencing”?

| source | n | percentage |
|---|---|---|
| **Not available** | 2868 | 46.22 |
| GitHub | 1151 | 18.55 |
| software heritage | 387 | 6.24 |
| zenodo | 142 | 2.29 |
| r package | 70 | 1.13 |
| cran | 56 | 0.90 |
| r package version | 54 | 0.87 |
| gitlab | 35 | 0.56 |

### Graphics Replicability Stamp Initiative



b/Surf: Interactive Bézier Splines on Surface Meshes

Claudio Mancinelli, Giacomo Nazzaro, Fabio Pellacini, Enrico Puppo
IEEE Transactions on Visualization and Computer Graphics (TVCG)

doi | Repository

# Best practices to disseminate

## Artifact Evaluation Committees

- current status
  - many guidelines developed in the 2010's ignore Software Heritage
  - ACM DL or Zenodo deposit of a zip + DOI still required
- state of the art
  - archive on Software Heritage
  - use the qualified SWHID

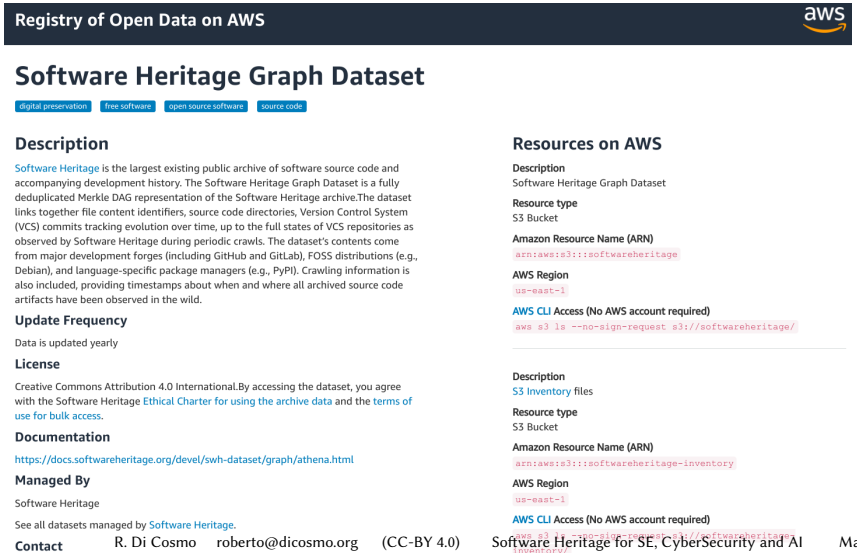
## Software citation

- current status
  - cursory mention in the text, sometimes citation of a research article, or manual ad hoc entries
- state of the art
  - include codemeta.json (or citation.cff, better if generated from codemeta.json)
  - generate biblatex entry, and use biblatex-software (CTAN, acmart.cls, etc.)

# Outline

# The full graph in the AWS Open Data collection

https://registry.opendata.aws/software-heritage/



Registry of Open Data on AWS

aws

## Software Heritage Graph Dataset

digital preservation | free software | open source software | source code

### Description

Software Heritage is the largest existing public archive of software source code and accompanying development history. The Software Heritage Graph Dataset is a fully deduplicated Merkle DAG representation of the Software Heritage archive.The dataset links together file content identifiers, source code directories, Version Control System (VCS) commits tracking evolution over time, up to the full states of VCS repositories as observed by Software Heritage during periodic crawls. The dataset's contents come from major development forges (including GitHub and GitLab), FOSS distributions (e.g., Debian), and language-specific package managers (e.g., PyPI). Crawling information is also included, providing timestamps about when and where all archived source code artifacts have been observed in the wild.

### Update Frequency

Data is updated yearly

### License

Creative Commons Attribution 4.0 International.By accessing the dataset, you agree with the Software Heritage Ethical Charter for using the archive data and the terms of use for bulk access.

### Documentation

https://docs.softwareheritage.org/devel/swh-dataset/graph/athena.html

### Managed By

Software Heritage

See all datasets managed by Software Heritage.

### Contact

### Resources on AWS

**Description**
Software Heritage Graph Dataset

**Resource type**
S3 Bucket

**Amazon Resource Name (ARN)**
`arn:aws:s3:::softwareheritage`

**AWS Region**
`us-east-1`

**AWS CLI Access (No AWS account required)**
`aws s3 ls --no-sign-request s3://softwareheritage/`

**Description**
S3 Inventory files

**Resource type**
S3 Bucket

**Amazon Resource Name (ARN)**
`arn:aws:s3:::softwareheritage-inventory`

**AWS Region**
`us-east-1`

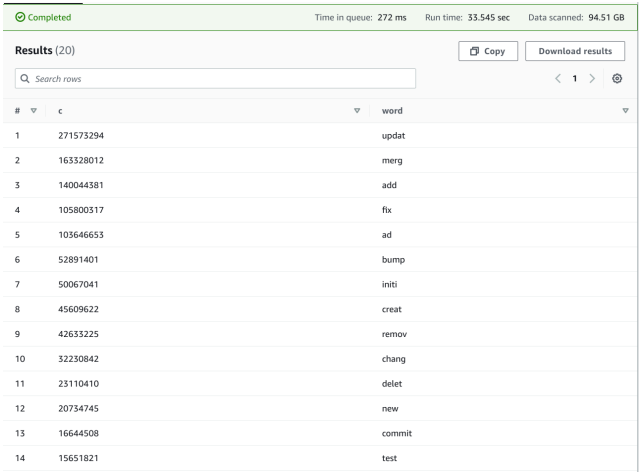
**AWS CLI Access (No AWS account required)**
`aws s3 ls --no-sign-request s3://softwareheritage-inventory/`

# Example: most popular commit verbs (stemmed)

## Query using Amazon Athena

```
SELECT COUNT(*) AS C, word FROM (
        SELECT word_stem(lower(split_part(
         trim(from_utf8(message)),' ', 1)))
        AS word FROM revision
   WHERE length(message) < 1000000)
WHERE word != ''
GROUP BY word
ORDER BY C
DESC LIMIT 20;
```

## Results



Completed — Time in queue: 272 ms — Run time: 33.545 sec — Data scanned: 94.51 GB

**Results** (20) — Copy — Download results

Search rows — 1

| # | c | word |
|---|---|---|
| 1 | 271573294 | updat |
| 2 | 163328012 | merg |
| 3 | 140044381 | add |
| 4 | 105800317 | fix |
| 5 | 103646653 | ad |
| 6 | 52891401 | bump |
| 7 | 50067041 | initi |
| 8 | 45609622 | creat |
| 9 | 42633225 | remov |
| 10 | 32230842 | chang |
| 11 | 23110410 | delet |
| 12 | 20734745 | new |
| 13 | 16644508 | commit |
| 14 | 15651821 | test |

# Going beyond SQL

State-of-the-art graph compression from social networks

Paolo Boldi, Antoine Pietri, Sebastiano Vigna, Stefano Zacchiroli

Ultra-Large-Scale Repository Analysis via Graph Compression

SANER 2020, 27th Intl. Conf. on Software Analysis, Evolution and Reengineering. IEEE

Results

Full graph structure (50 B nodes, 800 B edges) in 400 GiB RAM

- traversal time is tens of ns per edge
- bidirectional traversals implemented
- beware: metadata access is still *off RAM*

~~Java~~ Rust and gRPC APIs available …

docs.softwareheritage.org/devel/swh-graph/grpc-api.html

## What it does

- User-space POSIX filesystem (FUSE) exposing the entire SWH archive as a directory tree.
- Navigate by SWHID: `archive/swh:1:rev:...`, `archive/swh:1:dir:...`, `archive/swh:1:cnt:...`
- New backend on top of swh-graph: traverse history, parents, snapshots at full graph speed.
- Lazy content fetch + local cache — you only pay for what you read.

## Use case: Whisper team / Coccinelle

*Inria Whisper (LIP6, Sorbonne Université)*

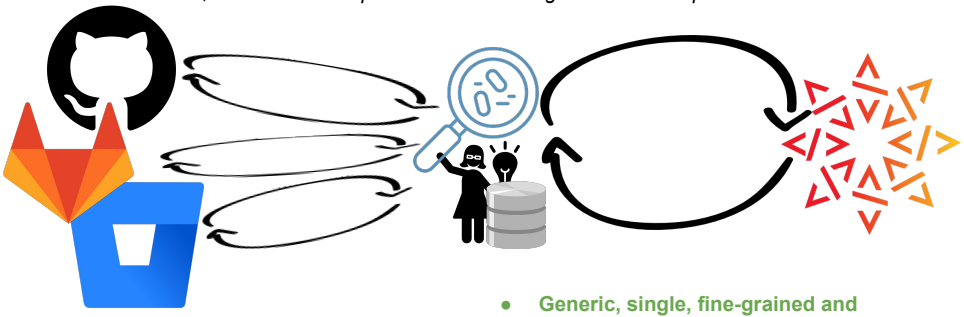
Coccinelle — semantic-patch engine for the Linux kernel.
With swh-fuse, Coccinelle can run *across the full history of public code*, not just the latest checkout: study patch propagation, vulnerability fix coverage, code-evolution patterns at SWH scale.

docs.softwareheritage.org/devel/swh-fuse • gitlab.softwareheritage.org/swh/devel/swh-fuse

# Outline

# Mining Android Applications on Software Heritage

*RQ: how to build a specific dataset for a given research question?*



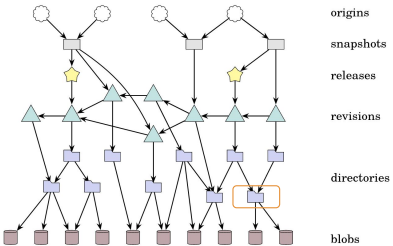
- **Specific and limited API**
- **Hardly reproducible**

- **Generic, single, fine-grained and unlimited API**
- **Growing number of source codes**
- **Easy to update the dataset**

*(from the Inria/IRISA DiverSE team)*

# Using the SWH merkle dag to identify android repositories

Identify android application repositories = Find the AndroidManifest.xml among the sources

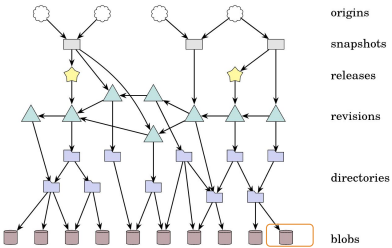



SWH Merkle DAG, Antoine Pietri

1) Iterate over the graph nodes until you find a directory node containing a file named "AndroidManifest.xml".

# Using the SWH merkle dag to identify android repositories

Identify android application repositories = Find the AndroidManifest.xml among the sources

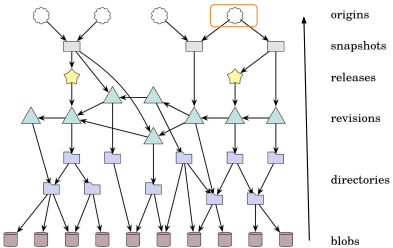



SWH Merkle DAG, Antoine Pietri

2) Extract the SWH identifier of the blob corresponding to the AndroidManifest.xml and download the corresponding file through the SWH Web API

# Using the SWH merkle dag to identify android repositories

Identify android application repositories = Find the AndroidManifest.xml among the sources




SWH Merkle DAG, Antoine Pietri

3) Traverse the graph in backward direction to the origin node and get the repository url

# Bottomline

Broad variety of sources in *one open dataset*

reduces usual GH bias

Reference simple *standard data format*

VCS and forge details are abstracted away

Simplifies reproducibility packages

no need to create a full copy, *just list the SWHIDs!*

Software Heritage does the heavy lifting for you

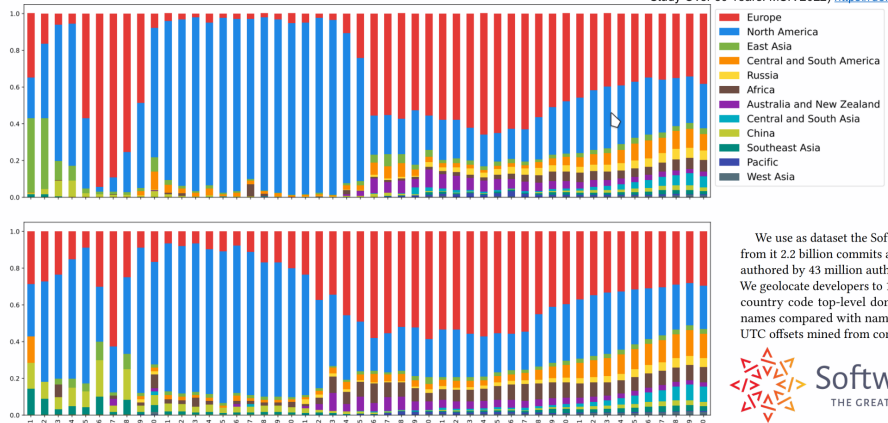
no need to scrape/download repositories all over again

# Example: (Open) Source Code comes from all over the world

MSR '22, May 23–24, 2022, Pittsburgh, PA, USA

Davide Rossi and Stefano Zacchiroli

Geographic Diversity in Public Code Contributions: An Exploratory Large-Scale Study Over 50 Years. MSR 2022) https://doi.org/10.1145/3524842.3528471

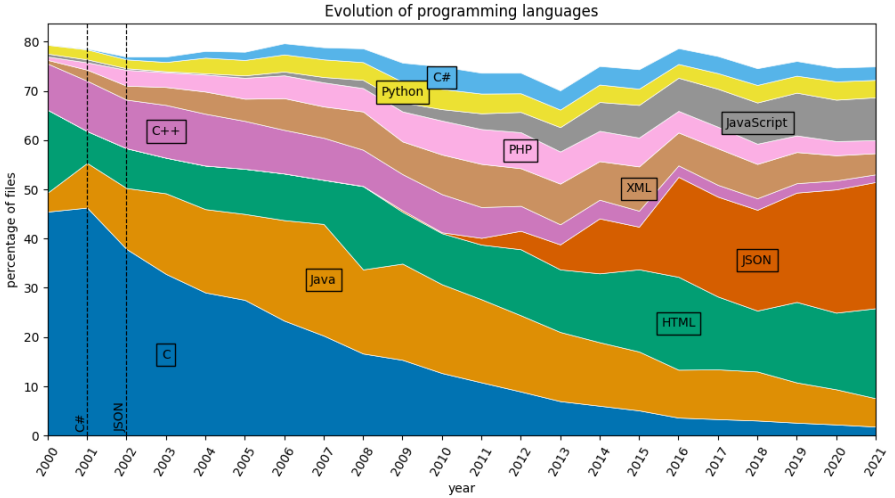



We use as dataset the Software Heritage archive [3] and analyze from it 2.2 billion commits archived from 160 million projects and authored by 43 million authors during the 1971–2021 time period. We geolocate developers to 12 world regions, using as signals email country code top-level domains (ccTLDs) and author (first/last) names compared with name distributions around the world, and UTC offsets mined from commit metadata.

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

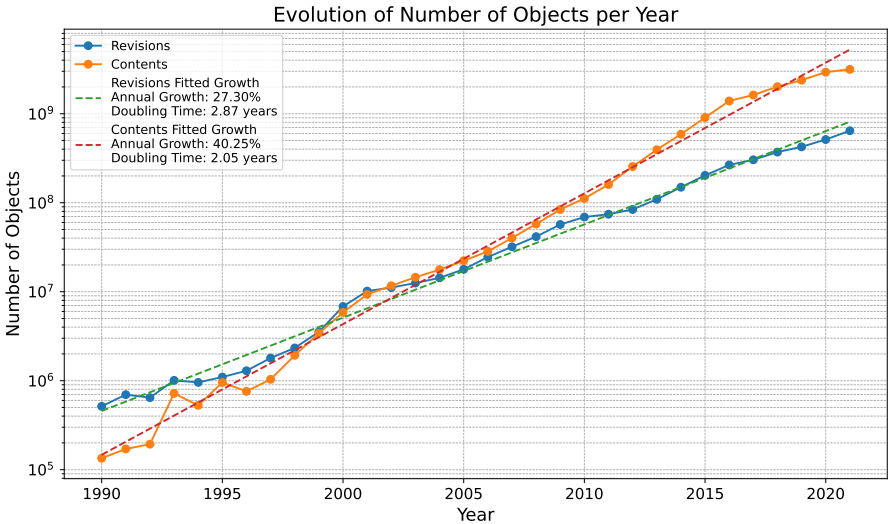
**Figure 3: Ratio of commits (above) and active authors (below) by world zone over the 1971–2020 period.**

# Example: Programming language evolution in (Open) Source




A. Desmazières, R. Di Cosmo, V. Lorentz
50 years of programming language evolution through the Software Heritage Looking Glass
International Conference on Mining Softwre Repositories (MSR) 2025.
https://hal.science/hal-04924849/

# Example: (Open) Source Code grows at an exponential rate




Software Heritage

Get the data and the code

# Example: Institutional contribution analysis

Question:

can we leverage the Software Heritage graph to map contributions from a research institution in the galaxy of software development?

Answer: let's try

- Extract: projects and contributions using email domain matching in the Software Heritage graph.
- Deduplicate: Cluster forks/copies using commit metadata, select a canonical repository.
- Enrich: Add GitHub metadata (topics, README, etc.) for context and visibility.
- Analyze: Compute key metrics on impact, contributor domains, and research relevance.

# Outline

# From the **Software Sky** to **Institutional Insight:** Anatomy of an Observational Pipeline

Observation parameters (config file)

## Stage 0: The Software Sky

The process begins with a complete map of the software universe, built from the Software Heritage archive, containing all observable software 'stars'. This stage is configured by observation parameters and a spectral filter definition (email domain selection).

## Phase 1: Target Selection

Identify the stars associated with a specific institution by selecting contributors using institutional identifiers, which isolates the relevant region of the software sky for focused observation.

## Quality Gate: Signal Strength Check

Before proceeding, the pipeline verifies that the selected region contains enough observable signals (software contributions) to produce statistically meaningful results.

Egeait to Nous Kaba

## Phase 2: Gravitational Cleaning (Deforking)

The process renvents gravitational artifects such as mitrers and fures from the data to retain only the physically meaningful celestial objects (original software projects).

## Phase 4: Spectral Enrichment

Each observed object is euriched with spectral data—such as activits, domains, and licenses—collected from multiple observation instruments to provide a deeper understanding.

Activity

Domains

Licenses

## Adeptive Observation Scheduling

This fecture ensures the automatic relation of observation tokens to bypass iassument (DPQ limits, ensuring continuous data collection.

## Observation Cache

A oerWixeen observation log is maintained, preventing redandant measurements and speeding up subsequent analyzes.

## Phase 5: Astrophysical Analysis

The entiched observations are analysed to compute metrics, classily objects into scientific domains, and generate a synthetic, comprehensive view of the observed system.

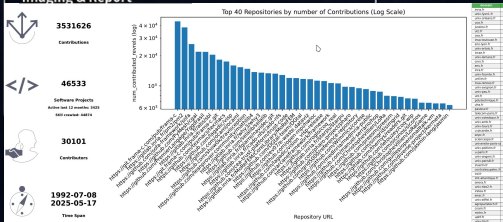
## Phase 6: High-Resolution Imaging & Report

**3531626**
Contributions

**46533**
Software Projects
Active last 12 months: 3025
Still-created: 16874

**30101**
Contributors

**1992-07-08**
**2025-05-17**
Time Span



# First SWH VLT pictures : SU contribution to Open Source

**155686**

**Contributions**

**2739**

**Software Projects**

**Active last 12 months: 91**

**Still crawled: 2469**

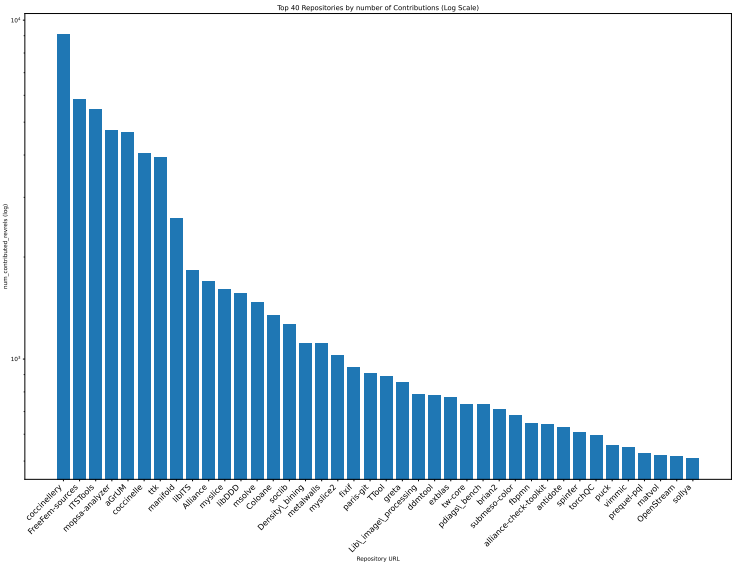
**1096**

**Contributors**

**1998-10-02**
**2026-03-04**

**Time Span**

# First SWH VLT pictures: SU contribution to Open Source



# Outline

## Programme

*France 2030 PTCC — 5 years (2024–2028) — 8 research teams*

Co-leaders: Olivier Barais (Inria, DiverSE) · Stefano Zacchiroli (Télécom Paris, ACES)

### Partners — academic

- APR — LIP6, Sorbonne Université *(prog. languages, static analysis MOPSA, verification)*
- ACES — Télécom Paris *(cybersecurity, software supply chain)*
- CEA List *(static analysis: Frama-C, Binsec)*

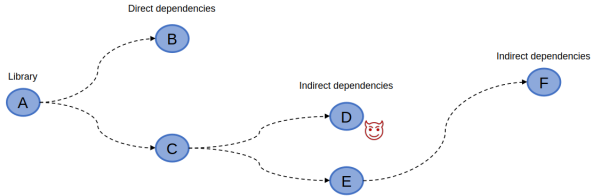
### Partners — Inria

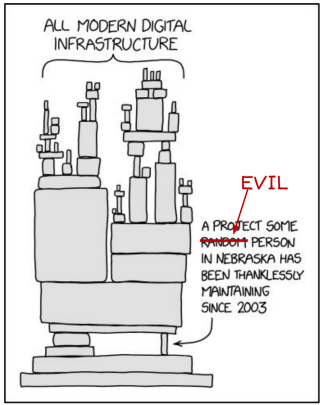
- DiverSE — supply chain security
- RMOD — reverse engineering (Moose)
- Software Heritage — archive, infrastructure
- Spirals — self-protective systems, web security
- Whisper — Linux kernel, Coccinelle

# Software supply chain attacks

## Reusing OSS via dependencies

- Software dependencies: a popular way of reusing open source software.
- Software product $A$ uses functionalities implemented in OSS product $B$ . . . and so on.




based on xkcd.com/2347

## Attacking the software supply chain

- Attacking undermaintained "leaf" packages (e.g., D) → efficient attack strategy
- Many documented attacks: event-stream (2018), node-ipc (2022), XZ utils (2024), . . .

# An universal knowledge base about public code vulnerabilities

## Vision

- Software Heritage is the perfect (and only) place where to build an universal knowledge base that maps known vulnerabilities to public code artifacts.
- SWH can provide an open data API mapping SWHIDs to CVEs, that knows about *all public commits* and can be leveraged to increase open source security.

## EU Cyber Resilience Act (CRA)

- Key helper to abide to CRA obligations, coming into effect ~Q3 2026.
- SWH funding member of the Open Regulatory Compliance Working Group.

## Roadmap

- Context: SWHSec project (PTCC-funded, 2023-2027).
- Current status: working prototype that processes OSV.dev data and use it to "color" the entire SWH commit graph (~5 billion commits) with vulnerability information.


Syful Islam, Stefano Zacchiroli.
On the Informativeness of Security Commit Messages: A Large-scale Replication Study.
To appear in EASE 2026, ACM, Glasgow, UK.


Syful Islam, Stefano Zacchiroli.
On the Use of Commit Messages for Corrective Software Maintenance: A Systematic Mapping Study.
To appear in EASE 2026, ACM, Glasgow, UK.


Romain Robbes, Théo Matricon, Thomas Degueule, André Hora, Stefano Zacchiroli.
Promises, Perils, and (Timely) Heuristics for Mining Coding Agent Activity.
To appear in MSR 2026, ACM, Rio de Janeiro, Brazil.


Luís Soeiro, Thomas Robert, Stefano Zacchiroli.
Finding Software Supply Chain Attack Paths with Logical Attack Graphs.
FPS 2025, Brest, France. Springer, 2026.


Solal Rapaport, Laurent Pautet, Samuel Tardieu, Stefano Zacchiroli.
Altered Histories in Version Control System Repositories: Evidence from the Trenches.
ASE 2025, IEEE, Seoul, South Korea. Pages 2183–2194.

# Outline

# Software Heritage and Generative AI, first contacts

October 19, 2023

## Software Heritage Statement on Large Language Models for Code

### Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

# Software Heritage and Generative AI, first contacts

October 19, 2023

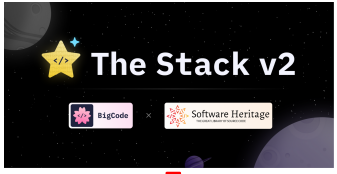
## Software Heritage Statement on Large Language Models for Code

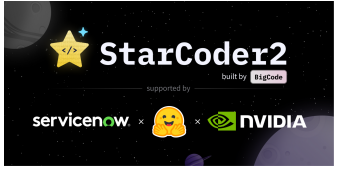
### Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

February 2024

Yes, it's possible!



# Software Heritage and Generative AI, first contacts

October 19, 2023

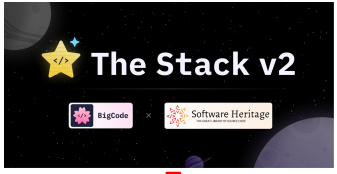
## Software Heritage Statement on Large Language Models for Code

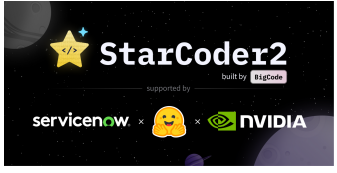
### Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

February 2024

Yes, it's possible!




But it's hard…

# Lessons learned

## Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

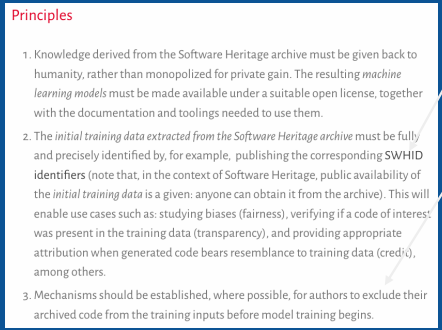
# Lessons learned

**Transparency is easy**: SWHID (undergoing ISO standardisation) and Software Heritage
N.B. : may be mandated by regulations!

Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

# Lessons learned



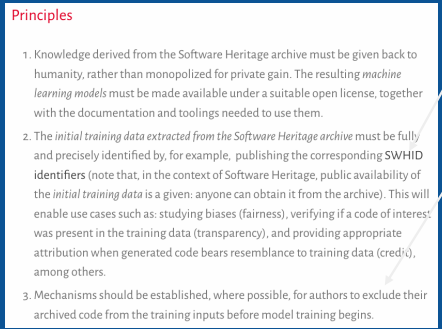
Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

**Transparency is easy**: SWHID (undergoing ISO standardisation) and Software Heritage
N.B. : may be mandated by regulations!

**Opt out is complex**: who is *the real right owner*?
(similar issues to license compliance)

# Lessons learned



Principles

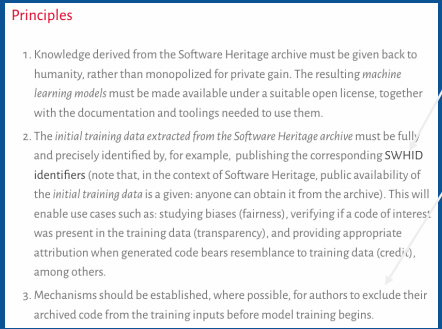
1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

+

**Transparency is easy**: SWHID (undergoing ISO standardisation) and Software Heritage
N.B. : may be mandated by regulations!

**Opt out is complex**: who is *the real right owner*?
(similar issues to license compliance)

# Lessons learned



Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

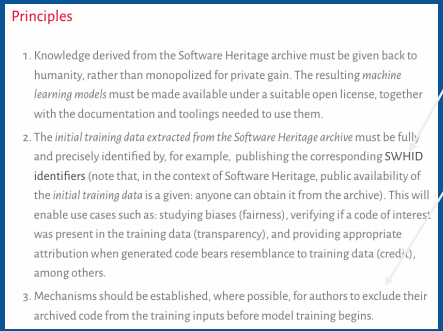
+

**Transparency is easy**: SWHID (undergoing ISO standardisation) and Software Heritage
N.B. : may be mandated by regulations!

**Opt out is complex**: who is *the real right owner*? (similar issues to license compliance)

- **Building the training set** is complex: e.g. includes **license compliance** alike work **at massive scale**

# Lessons learned



Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

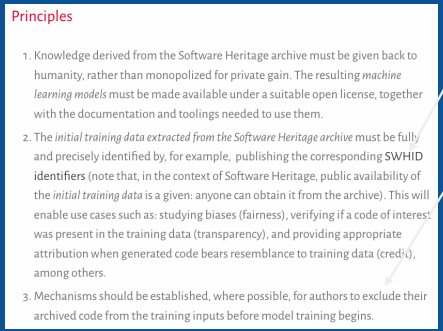
**Transparency is easy**: SWHID (undergoing ISO standardisation) and Software Heritage
N.B. : may be mandated by regulations!

**Opt out is complex**: who is *the real right owner*? (similar issues to license compliance)

+

- **Building the training set** is complex: e.g. includes **license compliance** alike work **at massive scale**
- Generating **attribution information** on model output **is more complex** than license compliance

# Lessons learned



Principles

1. Knowledge derived from the Software Heritage archive must be given back to humanity, rather than monopolized for private gain. The resulting *machine learning models* must be made available under a suitable open license, together with the documentation and toolings needed to use them.
2. The *initial training data extracted from the Software Heritage archive* must be fully and precisely identified by, for example, publishing the corresponding SWHID identifiers (note that, in the context of Software Heritage, public availability of the *initial training data* is a given: anyone can obtain it from the archive). This will enable use cases such as: studying biases (fairness), verifying if a code of interest was present in the training data (transparency), and providing appropriate attribution when generated code bears resemblance to training data (credit), among others.
3. Mechanisms should be established, where possible, for authors to exclude their archived code from the training inputs before model training begins.

**Transparency is easy**: SWHID (undergoing ISO standardisation) and Software Heritage
N.B. : may be mandated by regulations!

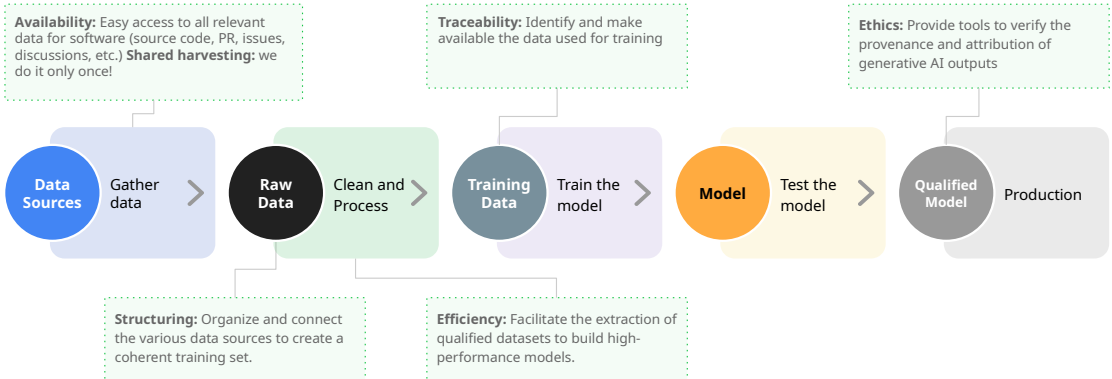
**Opt out is complex**: who is *the real right owner*?
(similar issues to license compliance)

+

- **Building the training set** is complex: e.g. includes **license compliance** alike work **at massive scale**
- Generating **attribution information** on model output **is more complex** than license compliance

We need **a coordinated effort** to ensure fully open models will succeed!

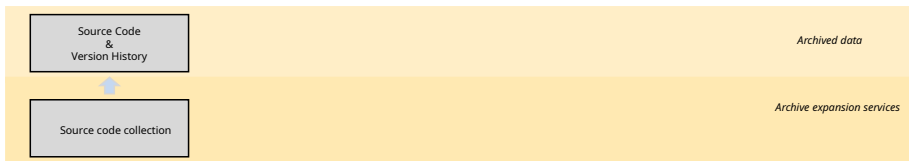
# A STEP FORWARD: CodeCommons





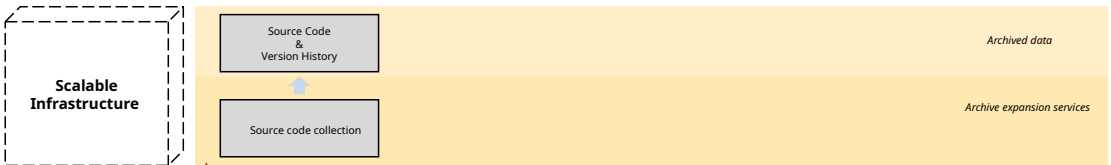

Solutions

# CodeCommons: bird’s eye view (technical focus)



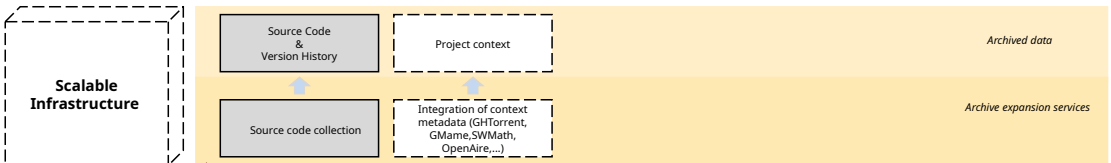

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

# CodeCommons: bird’s eye view (technical focus)



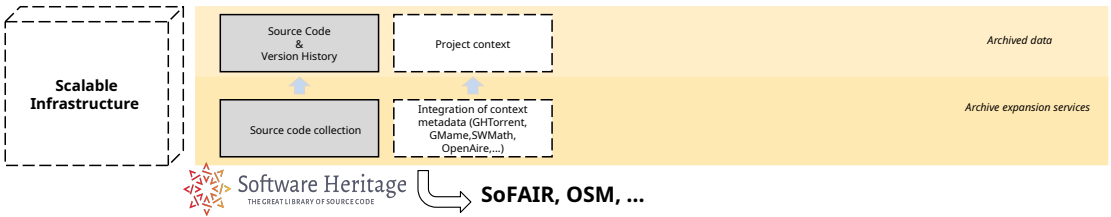

Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

# CodeCommons: bird’s eye view (technical focus)

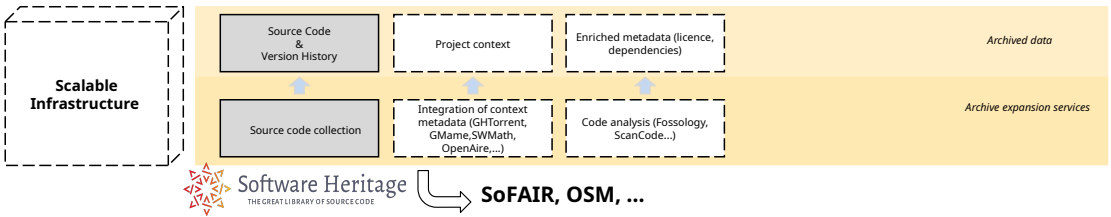



Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

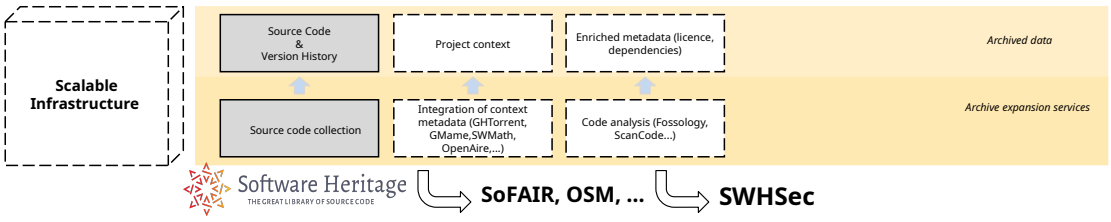
# CodeCommons: bird’s eye view (technical focus)



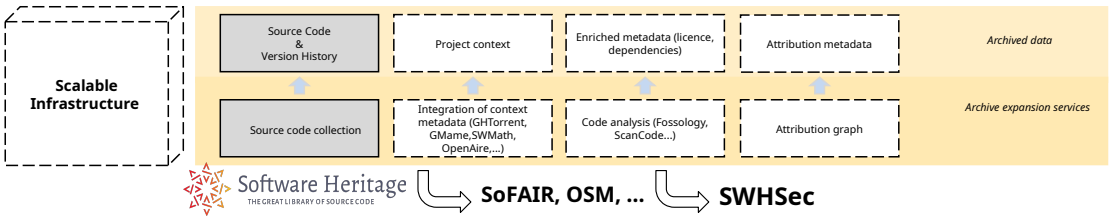
# CodeCommons: bird’s eye view (technical focus)



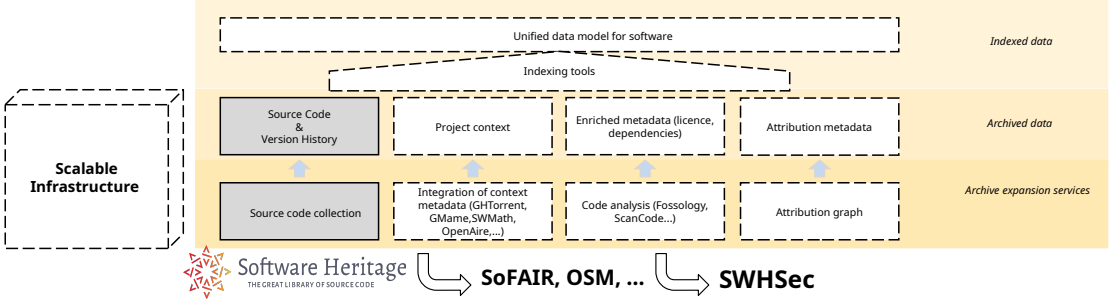
# CodeCommons: bird’s eye view (technical focus)



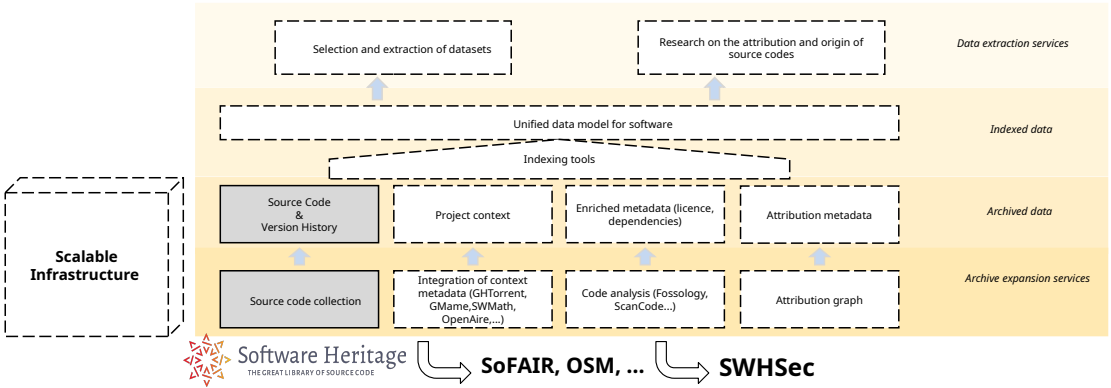
# CodeCommons: bird’s eye view (technical focus)



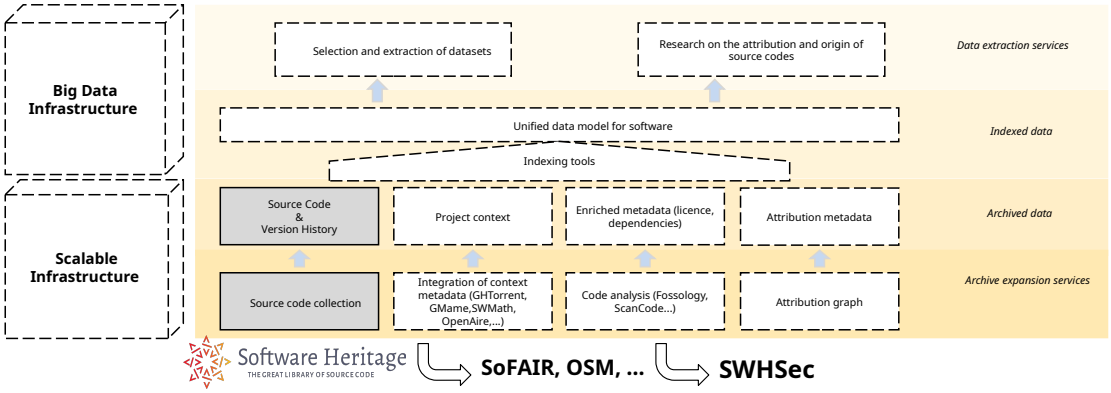
# CodeCommons: bird’s eye view (technical focus)



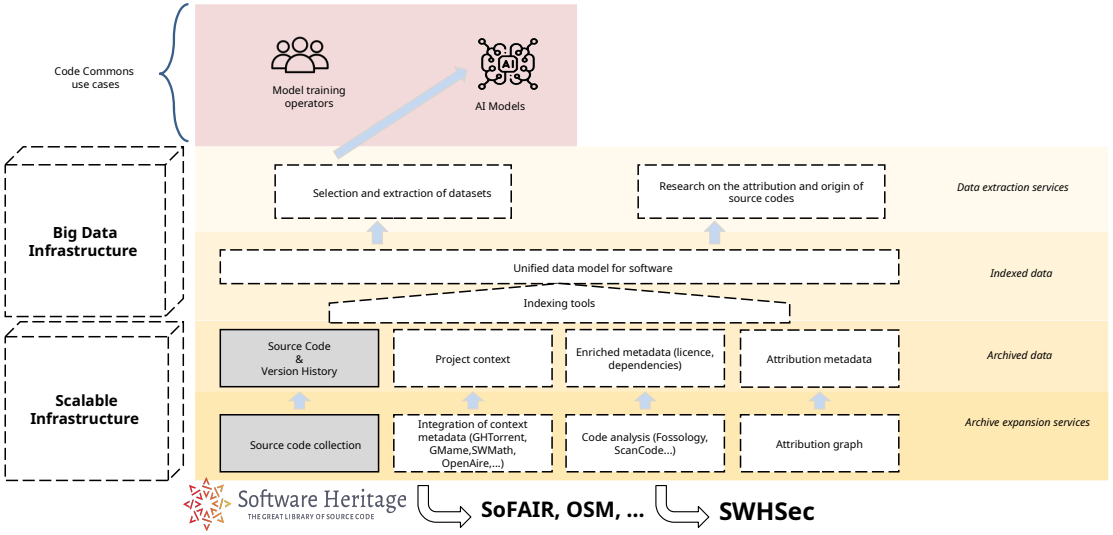
# CodeCommons: bird’s eye view (technical focus)



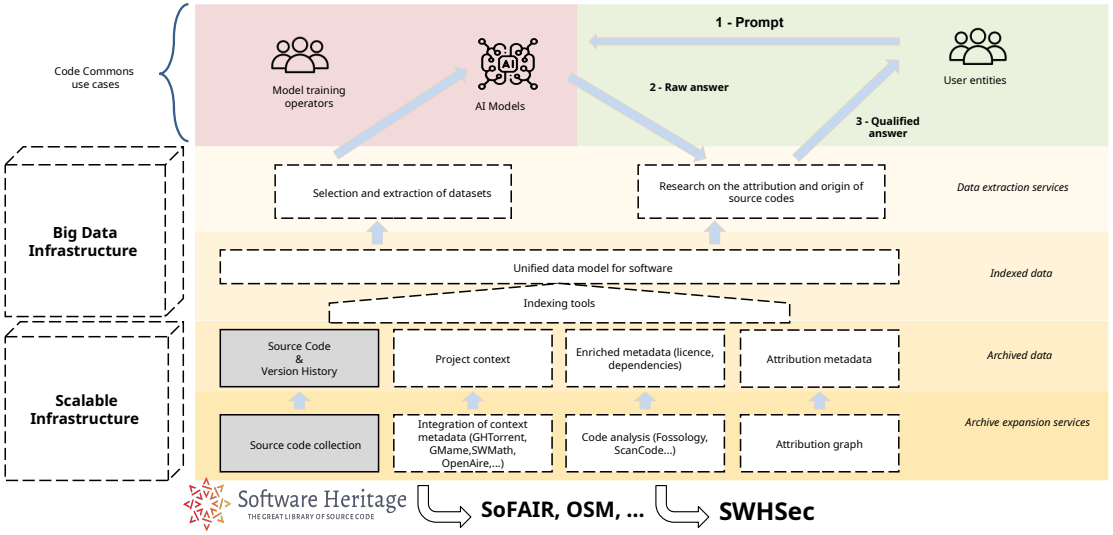
# CodeCommons: bird’s eye view (technical focus)



# CodeCommons: bird’s eye view (technical focus)



# CodeCommons: bird's eye view (technical focus)



# CodeCommons

## Open, responsible, and transparent AI: Our shared goal

**CodeCommons** is an ambitious project to create the world's most comprehensive digital commons for code

Building on the existing foundation of Software Heritage, the largest publicly available source code archive, CodeCommons aims to bring into one place all the **critical** and **qualified** information needed to create **smaller**, **better** datasets for the next generation of AI tools.

At its core, the project prioritizes transparency and traceability, enabling model builders and users to **respect creators' rights** while promoting **sovereign** and **sustainable** AI.

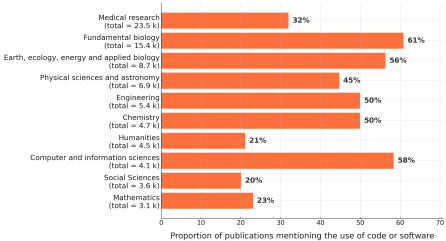
**Learn more**

**Meet the teams**

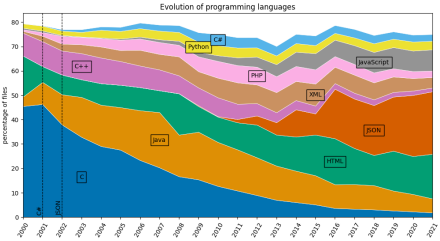
# Outline

# Software is a global undertaking…

## Pillar of Science across all research areas




## Many programming languages




## From all continents

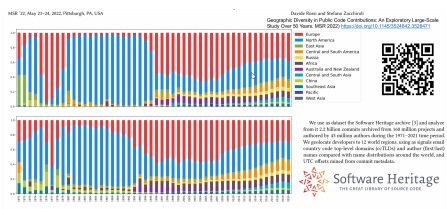
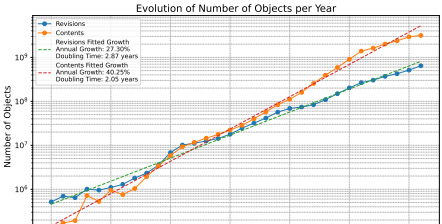


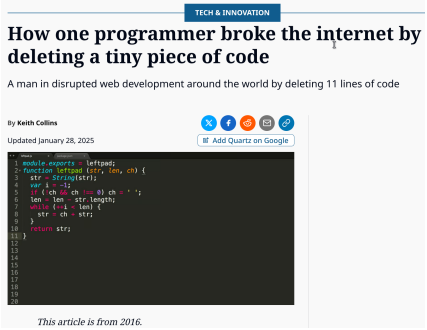

Figure 3: Ratio of commits (above) and active authors (below) by world zone over the 1971–2020 period.

## An exponential growth



# … with deep (infra)structural dependencies!

## An early warning: left-pad (March 2016)



TECH & INNOVATION

**How one programmer broke the internet by deleting a tiny piece of code**

A man in disrupted web development around the world by deleting 11 lines of code

By **Keith Collins**

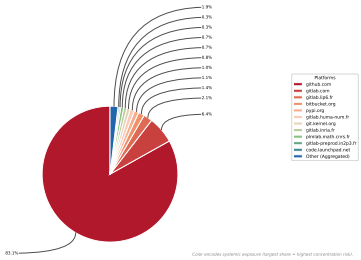
Updated January 28, 2025

Add Quartz on Google

```
1 module.exports = leftpad;
2 function leftpad (str, len, ch) {
3   str = String(str);
4   var i = -1;
5   if (!ch && ch !== 0) ch = ' ';
6   len = len - str.length;
7   while (++i < len) {
8     str = ch + str;
9   }
10  return str;
11 }
12
13
14
15
16
17
18
19
20
```

*This article is from 2016.*

## A clear and present danger




Used by `Sorbonne University` devs *(Source: Software Heritage)*

See the Wikipedia article:

- Facebook, PayPal, Netflix, Spotify affected
- web broken worldwide ~2.5 hours

"*If GitHub or PyPI disappeared tomorrow,* research would not just slow down *— it would stop.*"

# Resilience, demonstrated: Guix (and Nix) integration

Software Heritage and GNU Guix join forces to enable long term reproducibility

## Connecting reproducible deployment to a long-term source code archive

Ludovic Courtès — March 29, 2019

GNU Guix can be used as a "package manager" to install and upgrade software packages as is familiar to GNU/Linux users, or as an environment manager, but it can also provision containers or virtual machines, and manage the operating system running on your machine.

One foundation that sets it apart from other tools in these areas is *reproducibility*. From a high-level view, Guix allows users to *declare* complete software environments and instantiate them. They can share those environments with others, who can replicate them or adapt them to their needs. This aspect is key to reproducible computational experiments: scientists need to reproduce software environments before they can reproduce experimental results, and this is one of the things we are focusing on in the context of the Guix-HPC effort. At a lower level, the project, along with others in the Reproducible Builds community, is working to ensure that software build outputs are reproducible, bit for bit.

Work on reproducibility at all levels has been making great progress. Guix, for instance, allows you to travel back in time. That Guix can travel back in time *and* build software reproducibly is a great step forward. But there's still an important piece that's missing to make this viable: a stable source code archive. This is where Software Heritage (SWH for short) comes in.

**When source code vanishes**

Guix demonstrates automated recovery using the Software Heritage archive

When a platform or registry fails… Guix fallbacks to Software Heritage and continues to build!

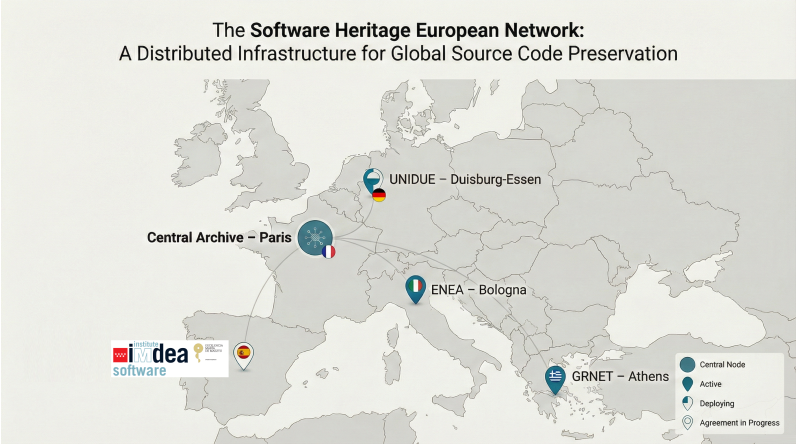
Since 2019

# An ecumenical approach to sovereign fallback of build chains is possible

## Extend to key package managers

*npm, PyPI, Maven Central, Cargo, CPAN, RubyGems, Packagist, …*

## Realize ecumenical resilience



# Outline

# A growing and active community

## Core Team

## Ambassadors

## All together, 2026 Symposium

# A rally flag for a grand vision

Bring together academia, industry, governments, communities

*"to build a reference, global infrastructure for open and better software"*

## Software Heritage is

- vendor neutral, open source
- a worldwide, long term initiative

## Software Heritage enables

- archival, reference, integrity
- qualification, sharing and reuse

## Call to action

- leverage for AI, Cyber, SwEng
- get involved research, code
- promote for AEC, Journals, etc.
- fund the long term mission

## Annual report 2025


## 5 years in 5 minutes


## Get these slides