Software Sovereignty

From Dependence to Resilience

Roberto Di Cosmo

CEO, Software Heritage
Inria & Université Paris Cité

27 October 2026
Auditorium 921, Praia de Botafogo 190
FGV Direito Rio — Reg Talks



Software Heritage
THE GREAT LIBRARY OF SOURCE CODE

Outline

1. Introduction
2. Software beneath sovereignty
3. Sovereignty as ecumenical resilience
4. Meet Software Heritage
5. From dependence to resilience

Short Bio: Roberto Di Cosmo

Computer Science professor in Paris, now working at INRIA

- *35+ years* of research (Theor. CS, Programming, Software Engineering, Erdos #: 3)
- *25+ years* of Free and Open Source Software
- *15+ years* building and directing structures for the common good



1999 *DemoLinux* – first live GNU/Linux distro

2007 *Free Software Thematic Group*
150 members 40 projects 200Me

2008 *Mancoosi project* www.mancoosi.org

2010 *IRILL* www.irill.org

2015 *Software Heritage* at INRIA

2018 *National Committee for Open Science*, France

2021 *EOSC Task Force on Infrastructures for Software*, European Union

Outline

1. Introduction
2. Software beneath sovereignty
3. Sovereignty as ecumenical resilience
4. Meet Software Heritage
5. From dependence to resilience

Beneath cloud, data and infrastructure

Digital sovereignty

Often framed in terms of *cloud*, *data*, or *infrastructure*.

Beneath lies a more fundamental layer:
software source code

Invisible fabric that powers our economies, administrations, and research systems.

Knowledge is in the source

```
/**
 * @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;
};
```

Covid Sim (excerpt)

Digital sovereignty: governing data, infrastructure and, first of all, *source code*.

Seven perspectives, one layer beneath all of them

"Contesting digital sovereignty"

state • supranational • network • corporate
personal • postcolonial • commons M. Jiang and L. Belli (eds), *Digital Sovereignty in the BRICS Countries*, Cambridge UP, 2025

Digital Commons: community-driven governance of shared digital resources and the production of digital public goods.

Software is all around us



Whichever perspective you take, the layer at the bottom is the same.

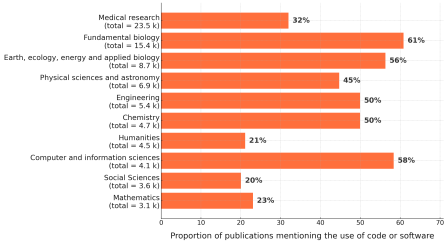
Where source code sits in the DPI stack

layer	**who controls it today**
public services: identity, payments, data exchange	the state, as public infrastructure
the applications that implement them	the state and its contractors
the open source components and registries they are built from	platforms and registries outside the state's control
the source code itself: preserved? identifiable? retrievable?	focus of this talk

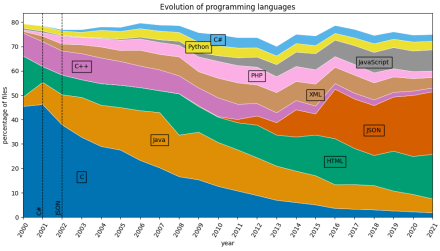
Every DPI is software that has to keep working.

A global undertaking…

Pillar of Science across all research areas



Many programming languages



From all continents

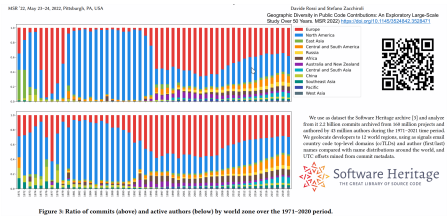
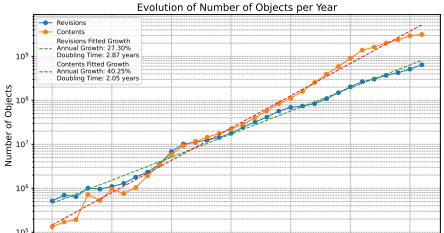


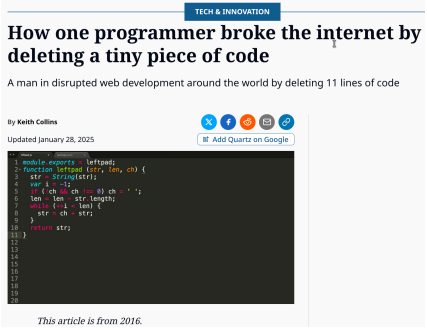
Figure 3: Ratio of commits (above) and active authors (below) by world zone over the 1971–2020 period.

An exponential growth



… and a global dependency network

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

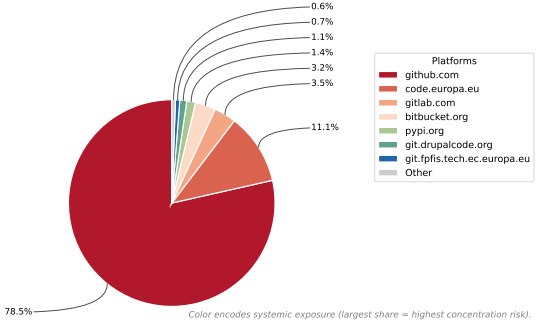
```
1  module.exports = leftpad;
2  function leftpad (str, len, ch) {
3    str = String(str);
4    var i = -1;
5    if (!ch && ch !== 0) ch = ' ';
6    len = len - str.length;
7    while (++i < len) {
8      str = ch + str;
9    }
10   return str;
11 }
12
13
14
15
16
17
18
19
20
```

This article is from 2016.

See the Wikipedia article:

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

A clear and present danger



Used by `europa.eu` devs *(Source: Software Heritage)*

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

Four faces of dependence

Platform the code lives where it is developed, and the developer does not own the place

Registry builds fetch named versions from npm, PyPI, Maven Central… at build time
left-pad, March 2016

Jurisdiction platforms and registries answer to one legal system, not the user's *xz-utils, 30 March to 10 April 2024*

Disappearance projects move, are renamed, are deleted, hosts close *Google Code, Gitorious*

Not on this list: who owns the intellectual property, where the authors were born. Neither keeps a build working.

An example is worth a thousand words

- 29 March 2024: the xz-utils backdoor is disclosed (CVE-2024-3094), the most examined source code of that year
- 30 March: the maintainers' accounts are suspended and the repository becomes unreachable. Software Heritage records a `not_found` visit at 09:18 that day
- 10 April: the repository is back; in between, the full history stayed readable in the archive
- 13 September 2026: two hours of global outage on GitHub, from an internal maintenance job
- Forges are not archives: platforms close (Google Code, Gitorious, about 1 M projects)

Maintainer's own account: https://tukaani.org/xz-backdoor/. Visit log, checkable from this room: https://archive.softwareheritage.org/api/1/origin/https://github.com/tukaani-project/xz/visits/. GitHub published no reason for the suspension, so none is attributed here.

Outline

1. Introduction
2. Software beneath sovereignty
3. Sovereignty as ecumenical resilience
4. Meet Software Heritage
5. From dependence to resilience

Sovereignty is not isolation

What it is *not*

- it is not *autarky*
- it is not *national* silos
- it is not *forking the world*
- it is not *owning everything*

What it is

- shared, open, non-rival infrastructure
- that benefits *all* while preserving autonomy for *each*
- resilient against disruption upstream
- *ecumenical* by design

"Sovereignty is not about owning everything
— it is about ensuring that nothing essential can disappear."

Can we get that for software?

Four properties of resilience

Plural custody answers platform dependence: one collection, several custodians, no custodian can speak for the others

Automatic fallback answers registry dependence: GNU Guix since 2019, Nix next, then the major package registries

Independent mirrors answer jurisdiction dependence: own operator, own jurisdiction, own stack, own decisions

Intrinsic identifiers answer disappearance: SWHID, ISO/IEC 18670:2025, recomputable by anyone

Dependence is when someone else's decision reaches your build.
Resilience is when it cannot.

Outline

1. Introduction
2. Software beneath sovereignty
3. Sovereignty as ecumenical resilience
4. Meet Software Heritage
5. From dependence to resilience

Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better science for all

Reference catalog



find and reference all software source code

Universal archive



preserve and share all software source code

Research infrastructure



enable analysis of all software source code

A *universal* archive, as a shared infrastructure

One infrastructure
open and shared

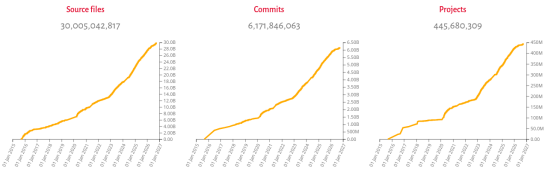
Cultural Heritage | Industry | Research | Public Administration

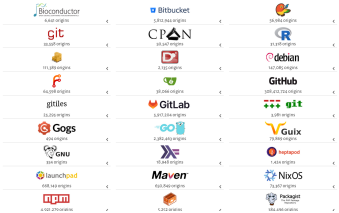
Software Heritage

Inria

unesco

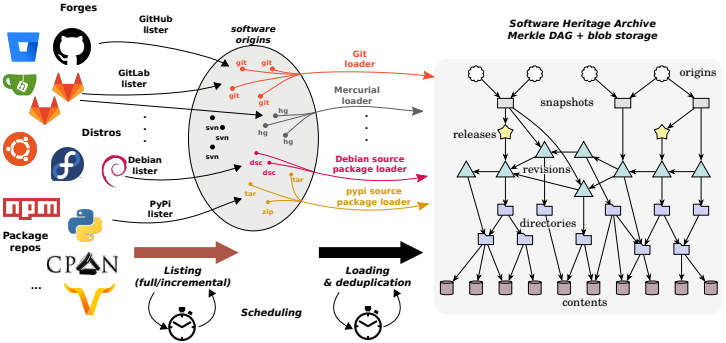
The largest archive ever built





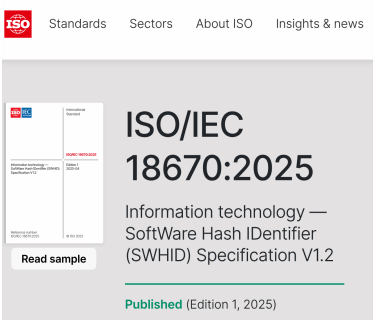
IBM Microsoft TII Technology Innovation Institute HUAWEI cnrs Infosys cea Ministère de l'Enseignement Supérieur et de la Recherche aws Google Université Paris Cité openinventionnetwork

AdaCore GitHub Inria Université de Pise Sorbonne Université Scuola Normale Superiore Université de Lorraine Sant'Anna NWO université Paris-Saclay



Global development history permanently archived in a uniform data model

- over 30 billion unique source files from over 445 million software projects
- ~3PB (compressed) blobs, ~60 B nodes, over 1 trillion edges



- computed from the content, not assigned by a registry
- the same bytes give the same identifier on every mirror, in every jurisdiction
- an international standard since 23 April 2025

The OpenChain CRA compliance checklist (v1.0, September 2026) names the SWHID among acceptable component identifiers.

A URL says where code was. A SWHID says what it is.

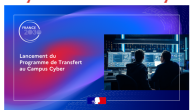
Groundbreaking services

Data for Transparent AI

Traceable source code for LLMs

CodeCommons

CyberSecurity



Make open-source more secure

Support Open Source Program Offices

OSPO RADAR

Track, find and reuse research software

Strategic Metrics and Insights

Software Heritage

Strategic Software Insights Report

The Public Code Observatory with UN and UNESCO

4 520 research articles cite Software Heritage *(Google Scholar, 2025)* • 2 M+ Save Code Now deposits

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.

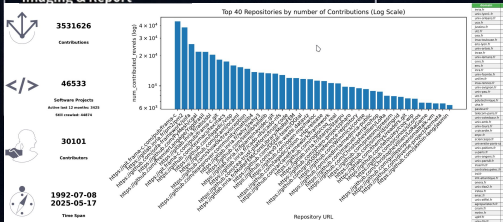
Phase 6: High-Resolution Imaging & Report

3531626
Contributions

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

30101
Contributors

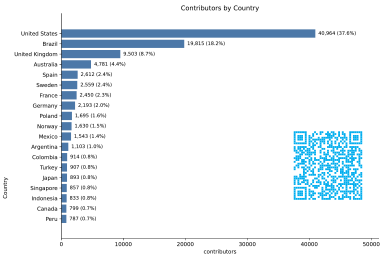
1992-07-08
2025-05-17
Time Span



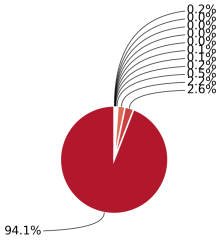
Brazil builds a lot but keeps almost all of it in one place

94 % of it lives on GitHub *all of .br*

Second in the world *government code*

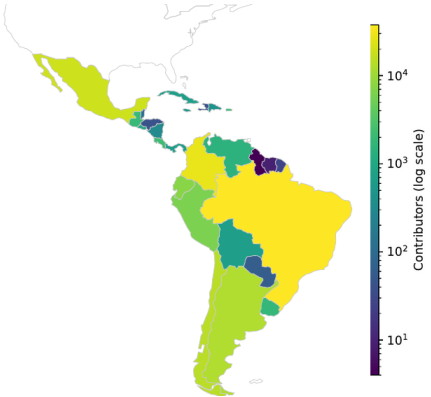


Public Code Observatory — UN, UNESCO, SWH
Brazil 2nd of 193 UN member states, July 2026

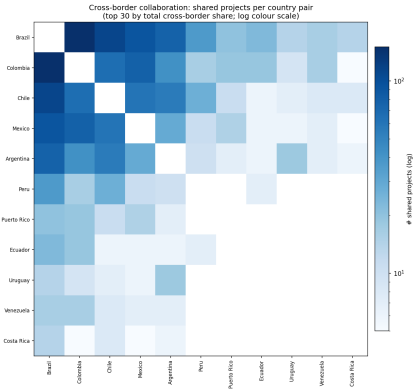


All Brazilian open source — not just government or academic: 256 908 projects, every `.br` domain.

A top-two producer of public code — on one platform, in one jurisdiction.
Global concentration = global risk.



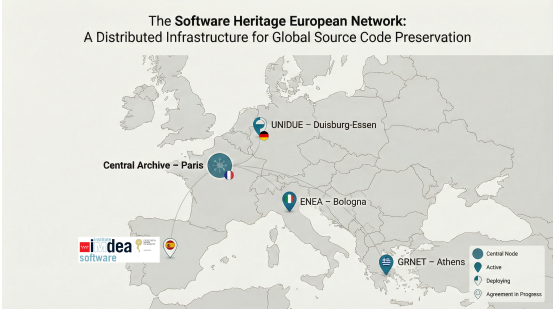
139 221 people, by the country of their institutional domain (log scale)



projects shared by each pair of countries (pairs with 5 or more)

the region already builds software *as a region*

The mirror network today



Four mirrors, all in Europe. Brazil would be the first beyond.

Two decisions, one moment

1. Fund the mirror at LNCC
   continuity of world source code, on Brazilian soil
2. Join the Foundation as a major sponsor
   join the governance

Each country can host it.

No country controls it.

10 years report

Outline

1. Introduction
2. Software beneath sovereignty
3. Sovereignty as ecumenical resilience
4. Meet Software Heritage
5. From dependence to resilience

The whole talk in one table

dependence	seen in	resilience
platform	76% of Europe's research code on one platform	plural custody
registry	left-pad, 2016	automatic fallback (Guix)
jurisdiction	xz-utils, 2024	independent mirrors
disappearance	5.4% already gone from where published	intrinsic identifiers (SWHID)

All four properties run today. The question for Brazil is whether to hold a seat.

True digital sovereignty is not control — it is resilience.

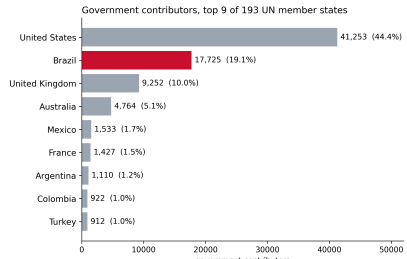
Soberania sem rivalidade. Controle sem exclusividade.

https://www.softwareheritage.org



Brazil builds a lot *second in the world for government code*

Government contributors



Public Code Observatory, edition 1.1: UN, UNESCO, Software Heritage

snapshot of 2 March 2026

Brazil, 2nd of 193 UN member states

- 17,725 government contributors
  19.1% of 92,831 worldwide
- 77,456 government projects touched
  25.6% of 288,411 worldwide

A top-two producer of public code. Worldwide, 94.5% of government projects sit on one platform.

Global concentration is global risk.

Redundancy and resilience

Copies

- under the control of one organisation
- one legal or commercial decision reaches all of them
- protect against failures

Mirrors

- under independent control, in their own jurisdiction
- removal from the archive does not propagate automatically: each operator decides
- protect against decisions

Resilience, demonstrated: Guix 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

Next: npm, PyPI, Maven Central, Cargo, CPAN, RubyGems, Packagist, … When a registry fails, the build continues.

Software Heritage as digital public infrastructure for software

What DPI is expected to be

- open
- non-profit
- universal
- standards-based
- multi-stakeholder
- interoperable

What the archive already is

- open code, open data, open APIs
- non-profit, Inria and UNESCO since 2017, partnership renewed January 2026
- all public code, every platform, every language
- SWHID, an ISO/IEC standard
- sponsors across sectors, mirrors run by independent institutions
- Guix, Nix, HAL, the Public Code Observatory build on it