Data Resilience with Existing Tools

Connecting what already works instead of building from scratch

Bernardo Gameiro

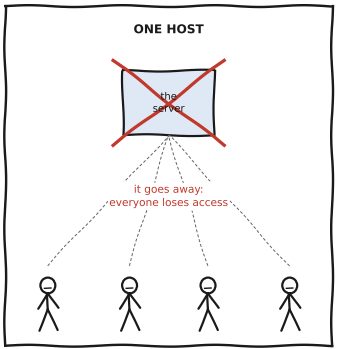
Instituto de Física Corpuscular (IFIC), Valencia, Spain

Nights of open Knowledge (NooK) 2026, Lübeck

20–21 November 2026

Introduction: data resilience

- Research depends on public data (climate, health, economic), most of it published by one institution, at one address
- Access can change for reasons unrelated to the science: new policies, funding cuts, bad actors
- Since January 2025, roughly **3,000 to 4,000** US federal datasets were removed from public access



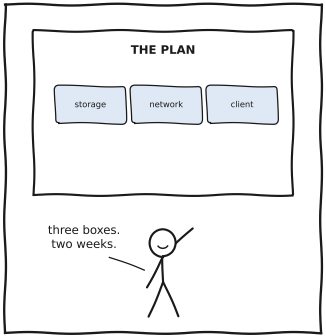
Data Rescue Project, *Understanding Federal Data Loss* (2025).

Resilient data stays available, and provably unchanged, when any one host goes away.

Every analysis that starts by downloading from that address stops with it.

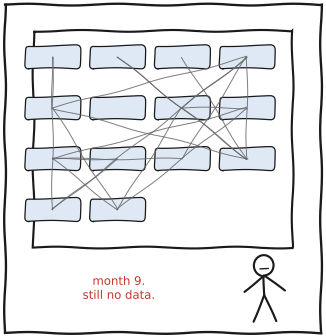
Usual approach: design a new system

- Write the requirements: **available**, **verifiable**, **partial** reads, **adopted**, **usable** in the tools analysts know
- Then build each one from zero: a network of nodes, content addressing, signatures, a new client, a new query language
- Nothing protects any data until all of it works **and** people install it
- This habit has a name: *not invented here*



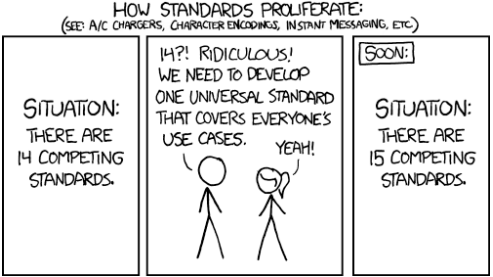
Usual approach: design a new system

- Write the requirements: **available**, **verifiable**, **partial** reads, **adopted**, **usable** in the tools analysts know
- Then build each one from zero: a network of nodes, content addressing, signatures, a new client, a new query language
- Nothing protects any data until all of it works **and** people install it
- This habit has a name: *not invented here*



Usual approach: design a new system

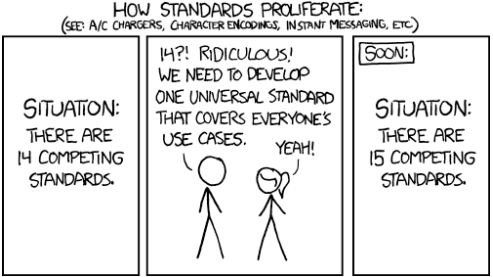
- Write the requirements: **available**, **verifiable**, **partial** reads, **adopted**, **usable** in the tools analysts know
- Then build each one from zero: a network of nodes, content addressing, signatures, a new client, a new query language
- Nothing protects any data until all of it works **and** people install it
- This habit has a name: *not invented here*



xkcd 927, "Standards", Randall Munroe. CC BY-NC 2.5.

Usual approach: design a new system

- Write the requirements: **available**, **verifiable**, **partial** reads, **adopted**, **usable** in the tools analysts know
- Then build each one from zero: a network of nodes, content addressing, signatures, a new client, a new query language
- Nothing protects any data until all of it works **and** people install it
- This habit has a name: *not invented here*



xkcd 927, "Standards", Randall Munroe. CC BY-NC 2.5.

The alternative: first check which requirements existing tools already meet.

Existing tools: candidates

	Available no single host	**Verifiable** copy provably genuine	**Partial** read part of a file	**Adopted** widely deployed	**Usable** in analysis tools
one web server	✗	✗	✓	✓	✓
repository with a DOI	✗	checksum	✓	in research	✓
git-annex / DataLad	if you add several	✓	whole files	some fields	after download
IPFS	✓	✓	✓	rarely	via a gateway
Dat / Hypercore	✓	✓	✓	very rarely	✗

IPFS was built for exactly this. Brave shipped an IPFS node inside its browser and removed it in August 2024: fewer than 0.1% of its users had used it.

Tools built for this are rarely used; tools used everywhere have one host.

Existing tools: candidates

	Available no single host	**Verifiable** copy provably genuine	**Partial** read part of a file	**Adopted** widely deployed	**Usable** in analysis tools
one web server	✗	✗	✓	✓	✓
repository with a DOI	✗	checksum	✓	in research	✓
git-annex / DataLad	if you add several	✓	whole files	some fields	after download
IPFS	✓	✓	✓	rarely	via a gateway
Dat / Hypercore	✓	✓	✓	very rarely	✗
BitTorrent	✓	✓	✓ (pieces)	**everywhere**	✗ **all first**

Adoption matters more than design here: data survives only while someone holds a copy.

One candidate built for something else meets four of five: sharing large files.

Existing tools: BitTorrent's track record

- Designed in 2001 to share large files between many people
- Data is named by a hash of its content, so any copy can be checked
- No central server needed: anyone holding the data can serve it, including a normal web server
- One well-known index: raided by police in 2006 and 2014, domains seized, blocked in many countries. Still online

The Pirate Bay, reply to DreamWorks
23 August 2004

"As you may or may not be aware, Sweden is not a state in the United States of America. Sweden is a country in northern Europe. Unless you figured it out by now, **US law does not apply here**. For your information, no Swedish law is being violated."

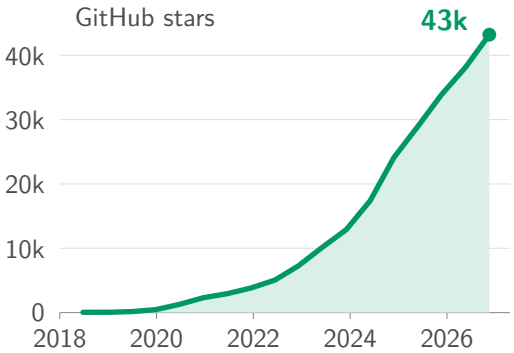
The last line of the reply is not suitable for a slide.

Excerpt. The site published the takedown notices it received, together with its replies.

Two decades of well-funded attempts to take files offline, largely unsuccessful.

Connecting them: the missing piece

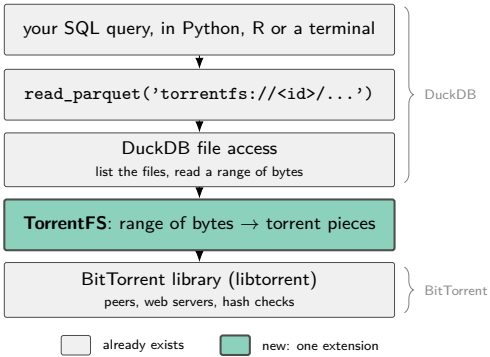
- **DuckDB** is already widely used: standalone SQL, inside Python, R or C/C++, and many more APIs. It reads Parquet and CSV directly, and extensions add new formats to read from
- Research data is already shared over BitTorrent: Academic Torrents has done it since 2013
- But, usually, a torrent client delivers the **whole dataset** before any of it can be used



GitHub stars of duckdb/duckdb, data from star-history.com, retrieved 19 November 2026.

Connecting them: the missing piece

- **DuckDB** is already widely used: standalone SQL, inside Python, R or C/C++, and many more APIs. It reads Parquet and CSV directly, and extensions add new formats to read from
- Research data is already shared over BitTorrent: Academic Torrents has done it since 2013
- But, usually, a torrent client delivers the **whole dataset** before any of it can be used



The transport exists, the analysis tool exists: only the connection is missing.

Connecting them: running example

- Monthly land temperatures of 3,448 cities, 1743–2013: 8.6 million rows in 227 Parquet files, 96 MB, one torrent
- The folder name is a column: a `Country` filter skips the other folders
- Each Parquet file ends with an index: its row count, and where each column is
- Question: *how many records are there for Portugal?*

Berkeley Earth, land temperatures by city, as published on Kaggle.

```
temperatures/  (one torrent, 96 MB)
  |- Country=Afghanistan/ data_0.parquet
  |- Country=Albania/data_0.parquet
  |- ...
  |- Country=Portugal/ data_0.parquet
  |- ...
  `- Country=Zimbabwe/data_0.parquet
```

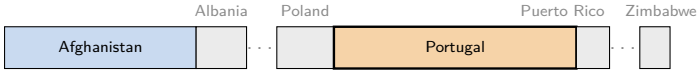
227 files in 159 folders

Answering it needs one file out of 227, and only part of that file.

Connecting them: how a query is served

TorrentFS is a DuckDB extension. It makes paths like `torrentfs://<id>/...` readable, and turns every read into a request for the torrent pieces that hold those bytes.

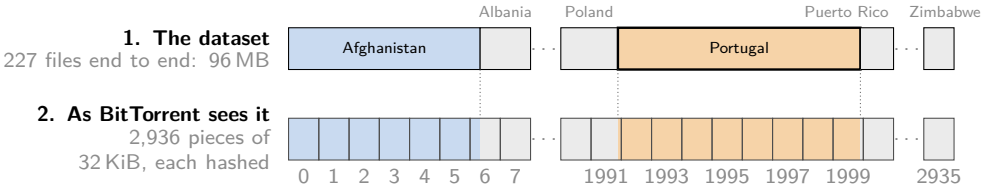
1. The dataset
227 files end to end: 96 MB



Nothing else changes: same SQL, same files, same DuckDB.

Connecting them: how a query is served

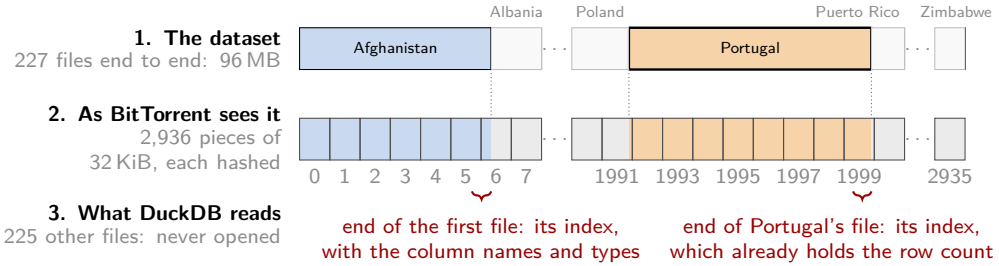
TorrentFS is a DuckDB extension. It makes paths like `torrentfs://<id>/...` readable, and turns every read into a request for the torrent pieces that hold those bytes.



Nothing else changes: same SQL, same files, same DuckDB.

Connecting them: how a query is served

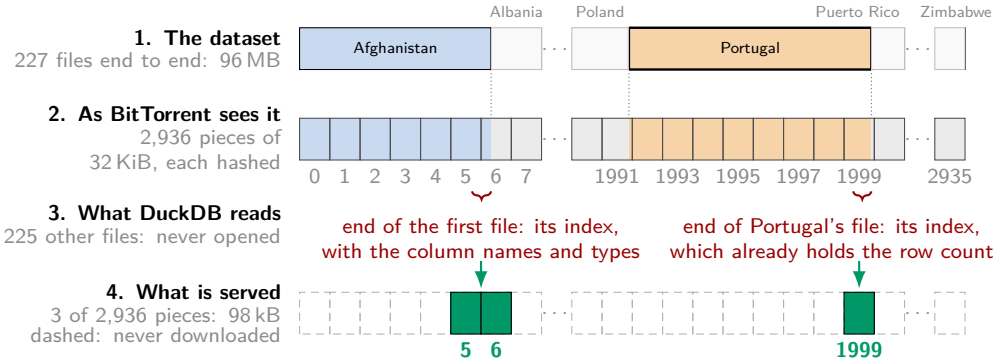
TorrentFS is a DuckDB extension. It makes paths like `torrentfs://<id>/...` readable, and turns every read into a request for the torrent pieces that hold those bytes.



Nothing else changes: same SQL, same files, same DuckDB.

Connecting them: how a query is served

TorrentFS is a DuckDB extension. It makes paths like `torrentfs://<id>/...` readable, and turns every read into a request for the torrent pieces that hold those bytes.



Nothing else changes: same SQL, same files, same DuckDB.

Connecting them: the example query in DuckDB

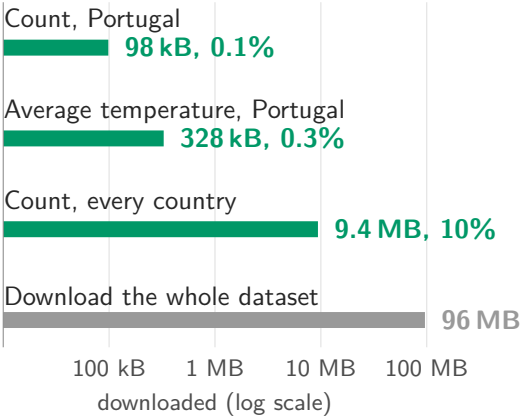
```
D LOAD torrentfs;
D SELECT count(*) FROM torrent_files('0172544aafd0...');  -- the file list only
227
D SELECT count(*)
  FROM read_parquet('torrentfs://0172544aafd0.../*/*.parquet',
                    hive_partitioning=true)
  WHERE Country = 'Portugal';
22233
D SELECT torrent_io_stats();  -- what crossed the network
{..., "bytes_downloaded":  98304, ..., "piece_hash_failures":  0, "peers_banned":  0}
```

98 kB of 96 MB: pieces 5, 6 and 1999 of 2,936, with a cold cache.

Local test network: one tracker and one seeder on the same machine.

Every piece is checked against its hash, then kept locally for the next query.

Connecting them: what each query downloads



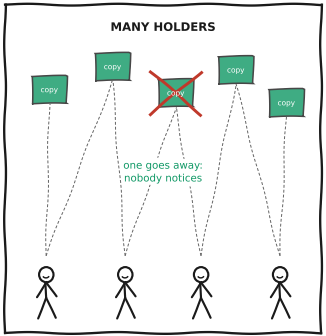
Cold cache, median of 3 runs each. Tested locally.

- **Count, Portugal**: only file indexes, where Parquet keeps the row counts
- **Average temperature**: the column is spread over the file's 6 blocks of rows, so 8 of its 9 pieces are needed
- **Count, every country**: the index of all 227 files, 10% of the dataset

The saving comes from the files a query never has to open.

Connecting them: publishing a dataset

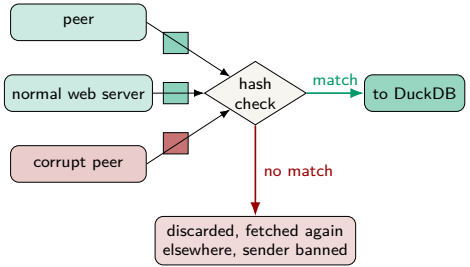
- Make a torrent of the folder you already publish, with standard tools
- Keep your web server and list it as one of the sources: it stops being the *only* one
- Publish the torrent identifier next to the DOI: it names the exact content
- Anyone can keep a copy: colleagues, libraries, other labs. No account, no vendor, and the extension will be open source



Compare with the first slide: one host there, many holders here.

Validation: every piece is checked, whoever sends it

We tried	What happened
a peer sending corrupted pieces	hash check fails; piece fetched again, peer banned
a web server returning zeros, garbage or the wrong offset	rejected by the same hash check
a hostile peer next to an honest one	results stay correct



Every piece is checked, so no source has to be trusted.

Summary

Problem

Public research data often has a single host. Since January 2025, 3,000 to 4,000 US federal datasets were removed from public access.

Approach

Write the requirements down, check which existing tools meet them, build only the connection. BitTorrent keeps data available and verifiable, DuckDB is where analysis happens, TorrentFS connects them.

Result

Datasets shared over BitTorrent can be queried in place with plain SQL: only the parts a query needs are downloaded, from any holder, and every piece is checked.

Next

Works today on Linux, tested on a local network. Next: public networks, Windows and macOS, a one-line install.



Thank you!

Special thanks to
Hanaan Shafi, Marcel Juhász
and Praggya Jeyakumar

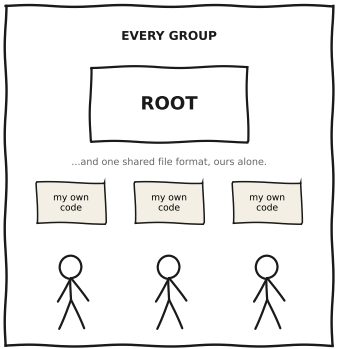
bgameiro@ific.uv.es

Backup: beyond this example

The approach, in three steps:

1. write the requirements down
2. check which existing tools already meet them, including tools built for something else
3. build only the connection

It applies well beyond data resilience. In nuclear and particle physics, many groups write their own analysis code and most data is stored in ROOT files, while Parquet and SQL tools are maintained by far larger communities. A possible next step: a DuckDB extension that reads ROOT files.



The less you build, the more of your effort goes to the actual problem.

Backup: status and next steps

Works today

- Parquet and CSV read straight from torrents in DuckDB
- Only the pieces a query needs are downloaded, each verified
- Normal web servers as sources alongside peers
- Both versions of the BitTorrent format

Next

- Tests on public networks (so far: a local test network)
- Windows and macOS (so far: Linux)
- Per-connection settings and a cache size limit
- Packaging for a one-line install

Sources

- US federal datasets removed since January 2025: Data Rescue Project, *Understanding Federal Data Loss and Future Directions* datarescueproject.org/data-loss-report
- Running example: Berkeley Earth, land temperatures by city (3,448 cities, 1743–2013), as published on Kaggle kaggle.com/datasets/berkeleyearth
- BitTorrent specifications, including web servers as sources and the second version of the format bittorrent.org
- Brave removed its built-in IPFS node in August 2024, used by fewer than 0.1% of its users github.com/brave/brave-browser/issues/37735
- git-annex and DataLad handbook.datalad.org; Academic Torrents (since November 2013), developing Human Connectome Project releases academictorrents.com
- The Pirate Bay: reply to DreamWorks of 23 August 2004, published on the site; police raids on 31 May 2006 and 9 December 2014 en.wikipedia.org/wiki/The_Pirate_Bay_raid
- DuckDB GitHub stars: star-history.com, retrieved 19 November 2026 star-history.com/#duckdb/duckdb
- Apache Parquet parquet.apache.org; DuckDB duckdb.org; xkcd 927, “Standards”, Randall Munroe, CC BY-NC 2.5 xkcd.com/927
- TorrentFS (repository not public yet): tested locally