

IMPERIAL

Department of Aeronautics

Engineering a Scalable, FAIR Data Infrastructure for Resource-Efficient Research

Irufan Ahmed

Acknowledgements

Foundational work

Initial design and development by Lee Benson as part of the AI for NetZero project - establishing custom InvenioRDM

Current work

Custom metadata and developing the storage backend via Ceph distributed storage

Case for a FAIR data repository

Valuable datasets are on inaccessible storage - creating loss of scientific knowledge and barriers to collaboration

Manual FAIR compliance is error-prone and unsustainable - automated schemas and DOIs are essential

Need for future retrieval, not just archiving - data must be machine-readable and queryable

Hidden compute cost - without data sharing, groups independently re-run expensive simulations that already exist

Poor metadata = wasted compute: if a dataset cannot be found or trusted, researchers re-run the simulation. A missed discovery can cost thousands of CPU-hours.

Our solution architecture

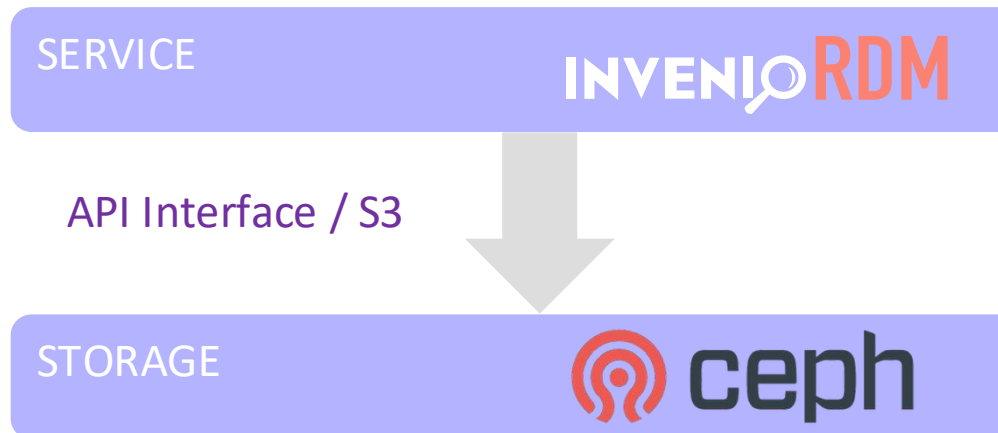
A persistent, citable home for research output

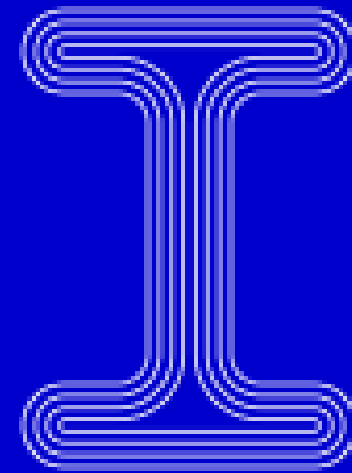
InvenioRDM

- Developed by CERN, built for scientific data
- FAIR compliance: automated DOIs, DataCite integration
- OAI-PMH harvesting for discoverability
- Extensible: allows custom metadata schemes

Ceph

- Horizontally scalable object storage
- Self-healing architecture (no single point of failure)
- S3-compatible for direct AI tool integration
- Long-term versioned preservation





Data Repository using InvenioRDM

Domain-specific metadata

Extension to InvenioRDM's metadata

Required Information

Core

- Title, description, date
- Resource type

Authors

- At least 1 name
- Affiliation (optional)
- ORCID (optional)

Study

Field

Keywords

Data

Type (numerical/experimental)

Facility

What was measured

Study Details

Research field

- Jet
- Boundary layer
- Wake
- Combustion
- Other

Flow characteristics

- Regime (compressible, incompressible)
- Subfield details
- Model name (e.g., Ahmed body)

Observation info

- Facility: code or experimental rig
- Methods: LES, PIV, RANS, DNS
- Operating conditions (Re, Mach)
- Coordinate system

File information

File-level metadata

- File formats (vtk, csv, hdf5)
- Variable names
- Variable descriptions
- Units for each variable
- Storage format (binary/ASCII)

Data uploading workflow

Using API to upload data

1

Fill metadata (web UI)

User-friendly – no need to fill in a JSON file

2

Upload to InvenioRDM (CLI)

Schema validation, faster and reliable using API

3

Submit for review

Initially uploaded as a draft

4

Review and approve

Generates a DOI

Searching for data

1

Simple keyword search

 turbulence

How

Type broad terms

Scope

Searches titles, descriptions, and authors

Use case

Quick discovery when you know the project name or topic

2

Metadata search



`custom_fields.domain.study.keywords:turbulence`

How

Use field:value syntax to target specific schema fields

Syntax

`custom_fields.domain.study.keywords:turbulence`

Use case

Precise queries when you need exact metadata matches

3

Faceted filtering



How

Click checkboxes on the sidebar

Mechanism

Maps queries to simple UI filters

Use case

Exploratory browsing and refining results interactively

4

Programmatic search (API)

How

Send HTTP requests via terminal(curl) or code (Python)

Mechanism

Bypasses UI to return JSON data

Use case

Automating data retrieval, bulk analysis or pipeline integration

Accessing data

Using the API to download files

Full download

- Local workstation and storage
- Best for full-scale processing where local storage is available
- Unfeasible for large data

High energy cost: transferring TBs
consumes significant network and storage
resources

Partial download

- Download local subset
- Best for quick inspection, plotting specific time-steps, feeding data loaders

Reduced transfer = lower energy costs
and faster iteration

Partial download

Cloud-native access

- Use virtualisation (via kerchunk) to map remote files
- HDF5/NetCFD – full support for random access
- CSV files – standard streaming support
- A 10 TB dataset → download only the 100 GB slice needed. **99% less transfer** for targeted analyses.

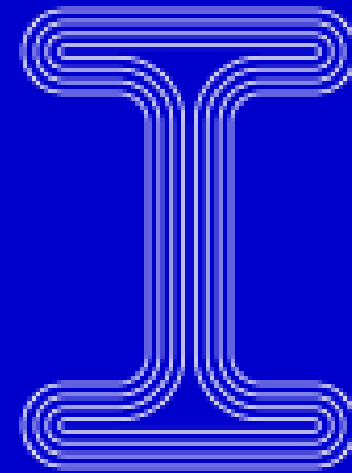
```
import fsspec
import zarr
from kerchunk.hdf import SingleHdf5ToZarr

# index hdf5 (build a lightweight JSON map)
with fsspec.open("https://data.ae.ic.ac.uk/sim_01.h5?download=1",
"rb") as f:
    reference = SingleHdf5ToZarr(f, url).translate()

# access data
fs = fsspec.filesystem("reference", fo=reference)
ds = zarr.open(fs.get_mapper(""))

# loading data
print(f"Dataset Shape: {ds['pressure'].shape}")
# >> (10000, 10000, 100)

# obtain a slice of the data (via HTTP range requests)
slice_data = ds['pressure'][0:100, 0:100, 0:10]
```



Ceph cluster as storage backend

One Ceph cluster, two storage services

- A single Ceph cluster is used as storage for two services:
 1. InvenioRDM for research data repository
 2. CephFS for day-to-day user storage via a POSIX file system mount
- Shared backend: both are backed by the same Ceph system, which supports file and object storage from one platform

Erasure Coding – more storage for the same hardware

4

Storage nodes

1 PB

Raw capacity

600 TiB

Usable (60% eff.)

- Ceph stores multiple full copies of the same object on different disks or nodes
- Replication size of 3 means every piece of data is stored 3 times, which is simple and fast but uses 3x raw storage
- For erasure coding, instead of storing full copies, Ceph splits data into k data chunks and m parity chunks then spreads them across different disks.
- Erasure coding delivers 1.8× more usable storage per unit of raw hardware compared to 3× replication - same fault tolerance, fewer disks required.

Embodied carbon: manufacturing storage hardware can exceed its operational power draw in lifetime CO₂. Maximising utilisation via erasure coding + compression + tiering *amortises that embodied carbon across far more data* - without buying new nodes.

CephFS vs per-group external drives

E-waste and hardware lifecycle

Per-group external drives

- Whole device is replaced when full
- Functional hardware is decommissioned regularly, generating significant e-waste.

Shared CephFS cluster

- Grow incrementally by adding a single disk or node - even with different specs.
- Old hardware stays in the cluster until components individually fail.

Failure and premature e-waste

- RAID controller failure can render all disks unreadable - requiring an identical replacement model (often unavailable for older hardware).
- The entire disk set becomes e-waste.

- Ceph self-heals across the cluster automatically when device fails
- Recovery load distributed - no single point of failure, no premature hardware death.

Energy per TB of active data

- Each group runs its own appliance - independently powered 24/7, even when mostly idle.
- Energy not shared or amortised.

- Infrastructure power shared across all contributing groups.
- A single well-utilised cluster consumes far less energy per TB than many small idle appliances combined.

Conclusion and next steps

What we've built

- A departmental data-stewardship platform using InvenioRDM + Ceph
- 600 TiB usable storage from 1PB raw via erasure coding
- FAIR data automated with DOIs
- Cloud-native HTTP range requests minimise data transfer energy

Next steps

- Enhance upload tool with smart resume capabilities for unstable connections
- Establish disaster recovery
- Develop API-driven pipelines for data analysis
- Tiered data storage - hot, warm, cold tiers for example
- Recommend or automate compression-aware submission workflows so that large datasets are archived in smaller formats

IMPERIAL

Department of Aeronautics

Thank you