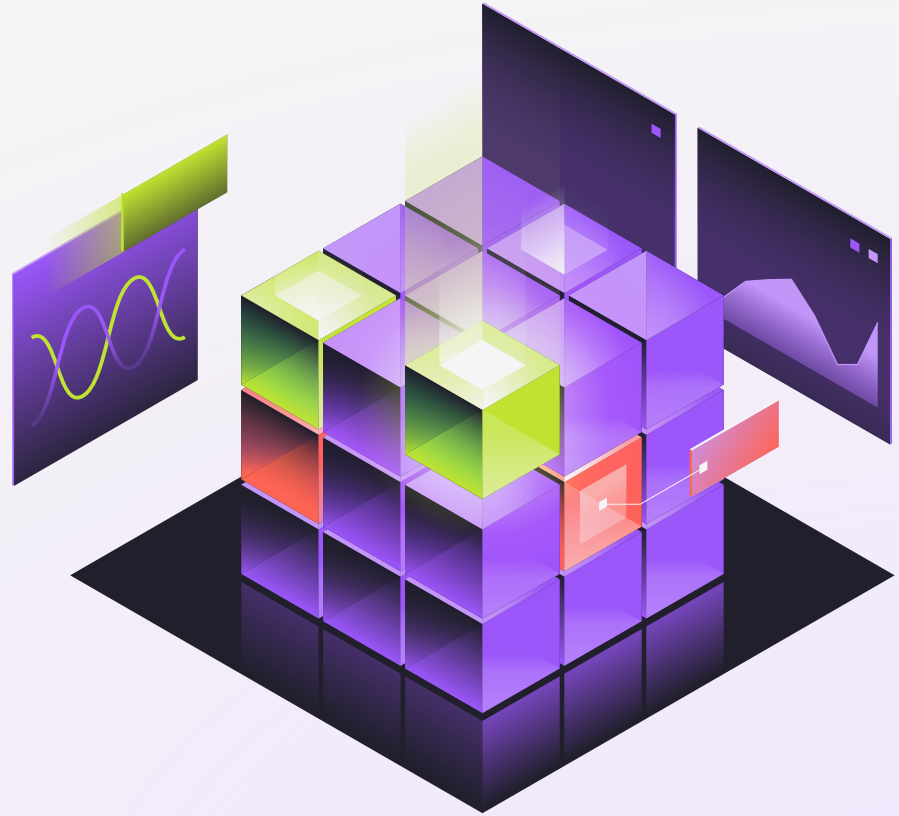
earthmover

virtual stores: from theory to production



ESIP July Meeting 2026

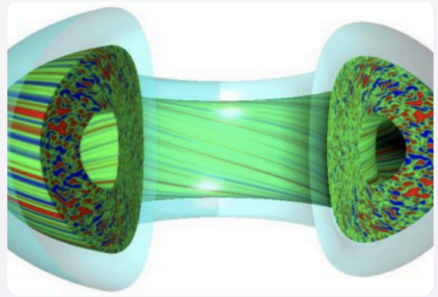
Tom Nicholas

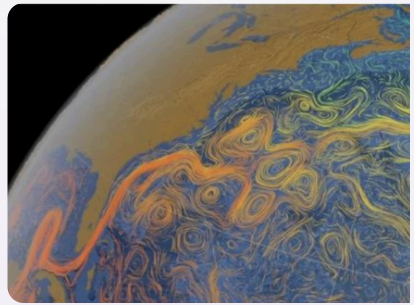
ABOUT ME

Tom Nicholas, PhD

FORWARD ENGINEER

- Plasma Physicist
- Pivoted to geoscience data
- Open source maintainer
  - Xarray
  - xarray.DataTree
  - VirtualiZarr
  - Icechunk





OPEN SOURCE CONTRIBUTIONS

MISSION STATEMENT

To empower people to use scientific data to solve humanity's greatest challenges

TODAY'S TALK

outline

- **What** is a virtual store?
- **Why** is this helpful?
- **How** do you make one?
- **Examples** of virtual stores in the wild
- **History** of geoscientists making Zarr stores
- **Future** of virtual stores

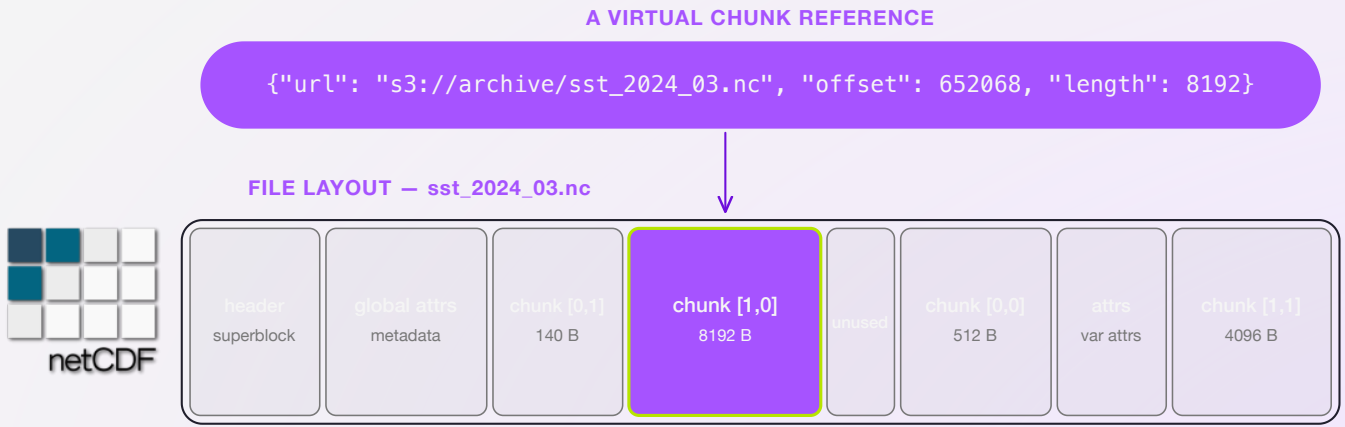
WHAT

what is a virtual store?

VIRTUAL CHUNK

virtual chunks are references to existing data

- Object storage makes it easy to **access** data from anywhere — but hard to **change** it. There's no edit-in-place, only whole-object overwrites. Incentive to **reference existing data.**
- A **virtual chunk reference** is a pointer to bytes that already exist somewhere else — e.g. in a NetCDF file.



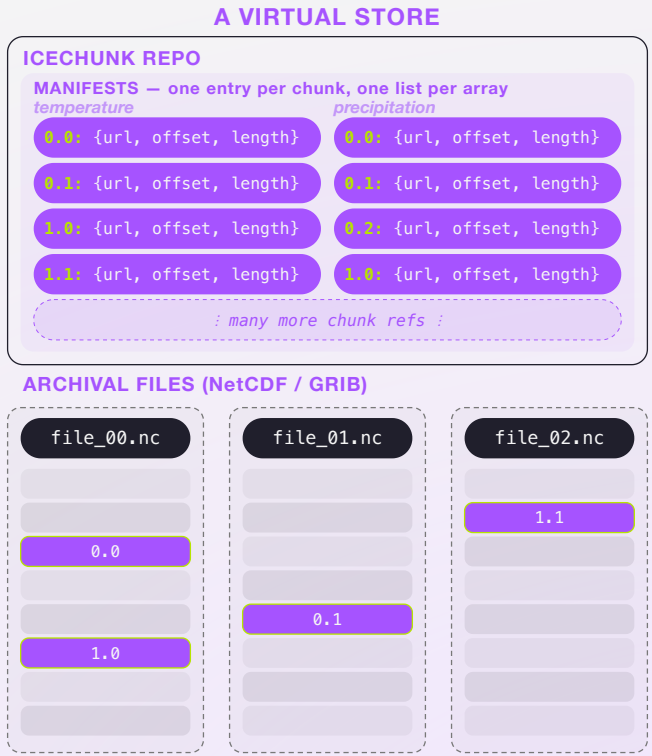
VIRTUAL STORE

a virtual store is a dataset made up of many virtual refs

A **chunk manifest** maps a Zarr array's logical chunk grid to chunk references — wherever the actual bytes live.

A **virtual store** is a data repository (Icechunk repo) whose manifests contain virtual chunks referring to many archival files.

- Looks like an ordinary Zarr store
- **No bytes copied** — the store is just references + metadata
- Many source files (e.g. a directory of NetCDFs) can become **one datacube**



WHY

why is this helpful?

BENEFIT 1: PERFORMANCE & COST

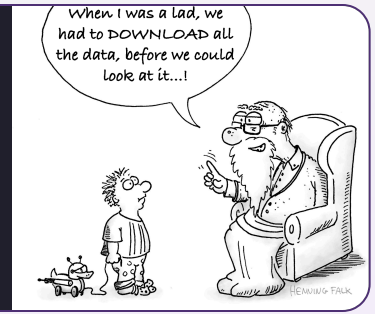
cloud-optimize without copying

- **Cloud-unfriendly formats** → virtual stores make metadata cheaply available
- **Slow downloads** → readers fetch only the bytes they need, not whole files (i.e. efficient random access)
- **Rewriting doubles storage costs** → no need to duplicate binary data as a new Zarr store. Saves 10's of thousands of $ per year for a PB of data!

BLOG POST

Fundamentals: What is Cloud-Optimized Scientific Data?

earthmover earthmover.io



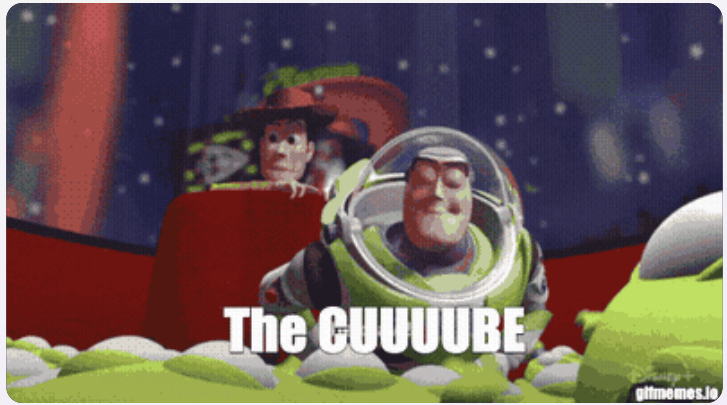
BENEFIT 2: CATALOGING

forget about files and formats

- **Too many files** → aggregate thousands of small files into a single large, queryable store

- **Too many formats** → Zarr becomes a **universal interface** for array data, regardless of what's underneath

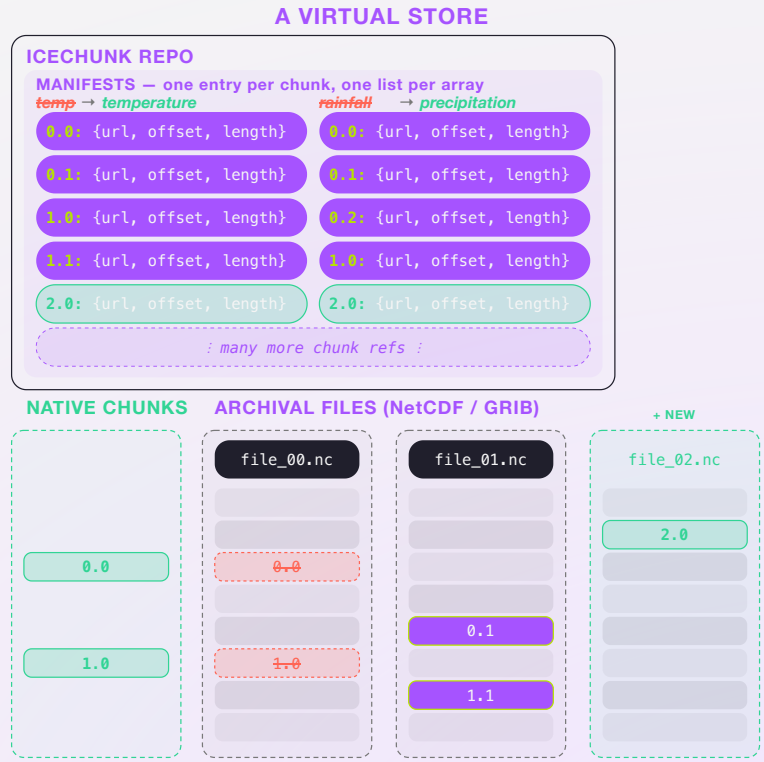
```
import arraylake as al
import xarray as xr

ic_repo = al.Client().get_repo("nasa/imerg")
session = ic_repo.readonly_session("main")
ds = xarray.open_zarr(session.store)
# do science!
```



BENEFIT 3: CONSISTENCY

fix problems without rewriting archives

- **Metadata inconsistencies** → the new store can arbitrarily rewrite metadata
- **Bad files** → corrupt or inconsistent files can have their chunks selectively overwritten with new native chunks
- **Updating operational data** → Icechunk provides safe, atomic commits for all updates, including virtually ingesting new files



BENEFIT 4: AGILITY

sidestep inertia of existing systems

Building virtual references only requires reading files you already have access to.

- **Data producer moves slowly** → *you don't need permission* to create a virtual store!
- **Data delivery coupled to location** → abstracting over the real data location allows all sorts of tricks...

Lowers the barrier for anyone to make an existing archive cloud-optimized.

HOW

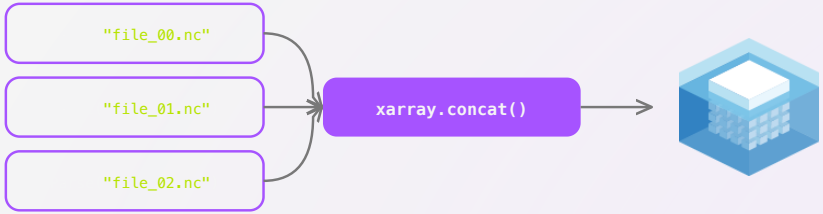
how do you make one?

HOW

how do you make a virtual store?

VirtualiZarr

- Parses archival files
- Organizes multiple files into one datacube (uses flexible Xarray API)
- Writes virtual references into Icechunk

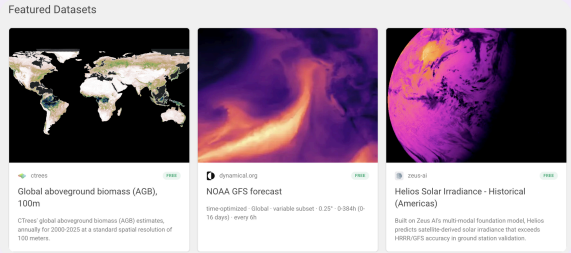


Icechunk

- Stores virtual references in object storage, as manifests
- Allows safe, efficient updates and version history
- User interacts with generic Zarr interface
- Reads data back *as fast as physically possible*

Arraylake (optional step)

- Catalog and share the data, publicly or privately



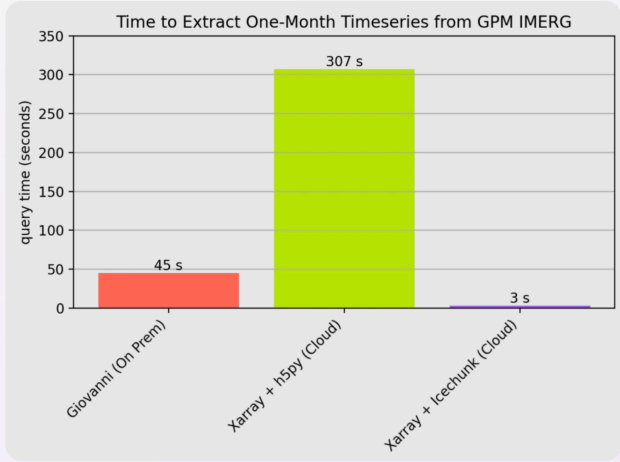
EXAMPLES

virtual stores in the wild

CASE STUDY 1

nasa imerg

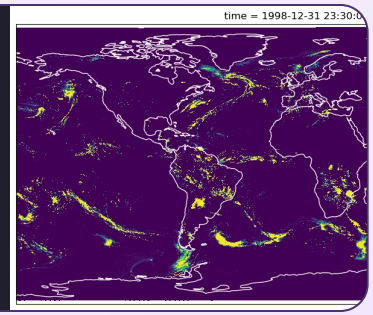
- Global precipitation estimates, historically distributed as one file per timestep. Great for a global map, painful for a timeseries...
- Fetching only necessary chunks → **100x faster** timeseries access once virtualized (compared to naive Xarray + h5py)!



BLOG POST

Solving NASA's Cloud Data Dilemma: How Icechunk Revolutionizes Earth Data...

earthmover earthmover.io

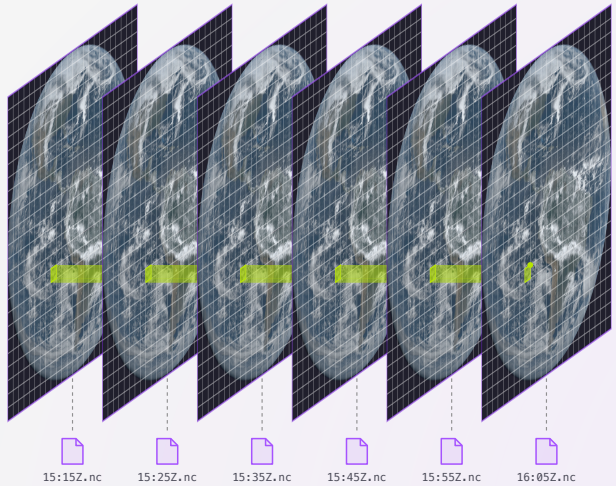


CASE STUDY 2

goes-16

Scaling test: 7-year geostationary satellite imagery archive. 300k NetCDF4 files, **7.1 billion chunks**, one Icechunk store!

Allows **simple timeseries access** across scenes.

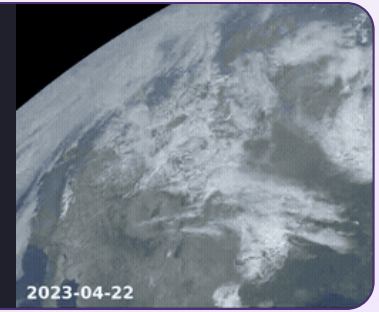


> **Note**
>
> - Did not need to ask the GOES-16 team's permission to build this.
> - Read from the public AWS archive, wrote references into our own store in a different bucket.

BLOG POST

Old format, no problem!: Cloud-optimizing the GOES-16 archive as Virtual Zarr

earthmover earthmover.io



CASE STUDY 3

grib

GRIB is a notoriously fiddly format to work with — format-specific parsers make virtualizing it tractable.

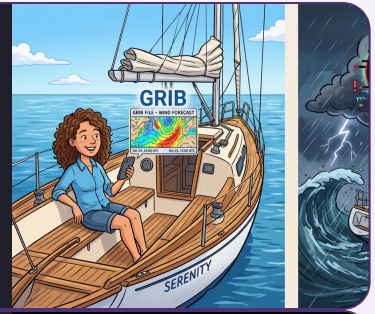
- **NOAA NBM**, virtualized via "Gribberish" package. No one correct way to map GRIB → Zarr, so uses flexibility of VirtualiZarr to handle the different data model
- **NOAA HRRR**, virtualized by **dynamical.** 14PB, ~1 billion GRIB messages, 150+ variables!

  Requires a special codec to decode the GRIB contents at read-time.

BLOG POST

Virtually Gribberish: Bringing Icechunk clarity to GRIB archives

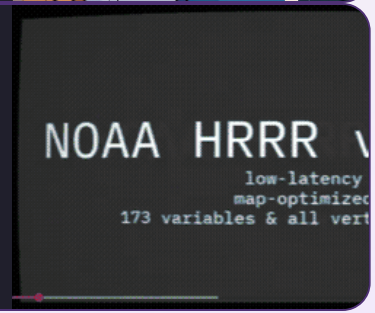
earthmover earthmover.io



GUEST POST

Having it all: virtual and materialized data products

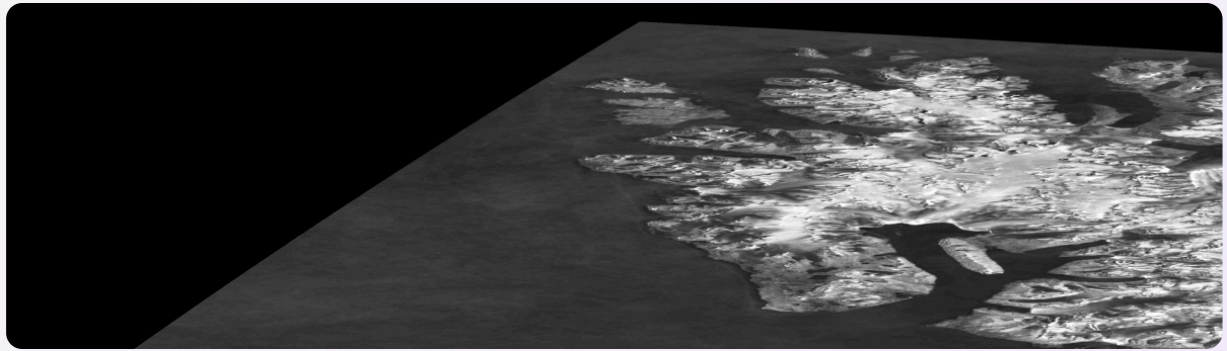
earthmover earthmover.io



CASE STUDY 4

tiff / cog

- TIFF/COG can be virtualized too (thanks to `virtual-tiff` package.)
- For example the `sentinel-1-global-coherence` dataset on AWS (~1 million TIFF files, 70TB).

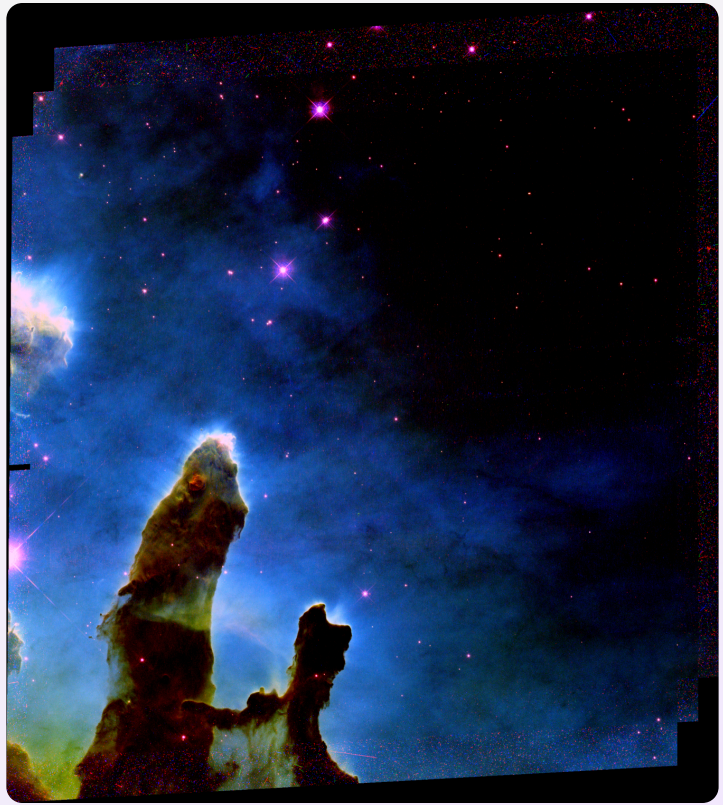


CASE STUDY 5

fits — hubble

Even astronomy archives benefit: Hubble's Pillars of Creation, virtualized from FITS files in the HST archive on AWS Open Data Registry into a Zarr store.

Image made using a few lines of Xarray code!



IN PRODUCTION AT EARTHMOVER

virtual chunks allow interesting tricks

- **Filtered subscriptions** - customer A sells subset of their store to customer B. Works by shipping modified manifests to B, which point to subset of chunks in A's storage.
- **Embargoed ERA5** - public and private versions. Storage indirection allows us to move embargoed data from private to public storage on a rolling basis.

CAVEATS

what's the catch?

- **Can't rechunk** without using more storage — stuck with the original chunk pattern of the source files.
- **Assumes no referenced files are altered** - Icechunk error upon read if any changes, but link now broken
- **Not every dataset** is supported - Some archives are just too messy. Note **Level 3/4** data works well today; not yet **Level 2**.
- **Non-trivial security concerns** - arbitrary indirection can be dangerous - Arraylake solves this but Icechunk alone structurally cannot.

BLOG POST

Virtual chunks that just work — but securely

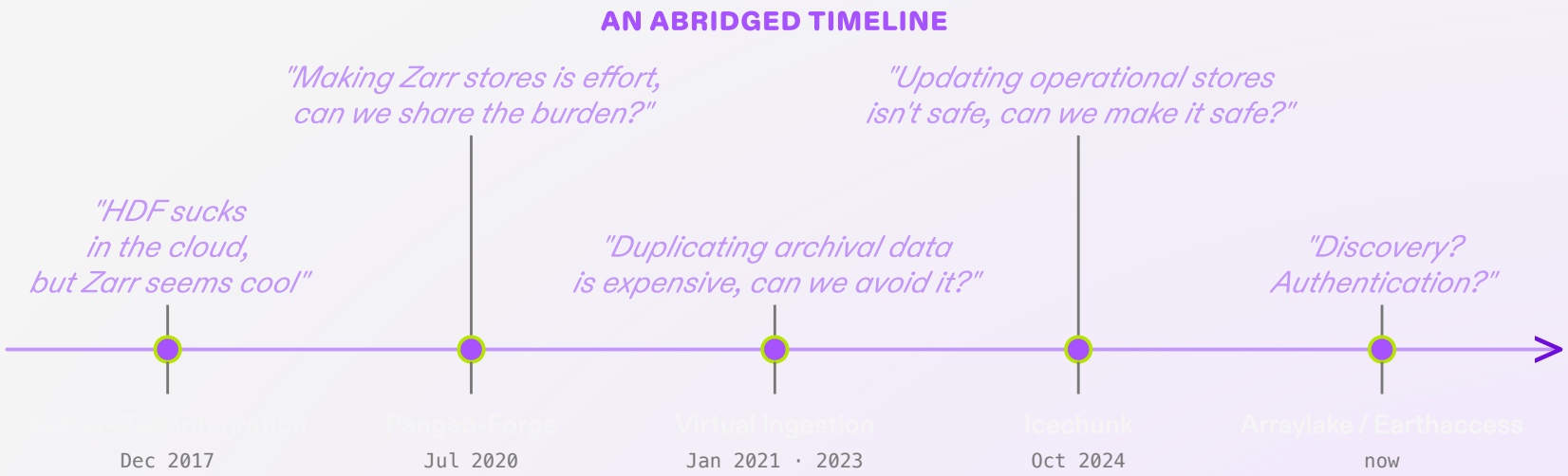
earthmover earthmover.io

HISTORY

how did we get here?

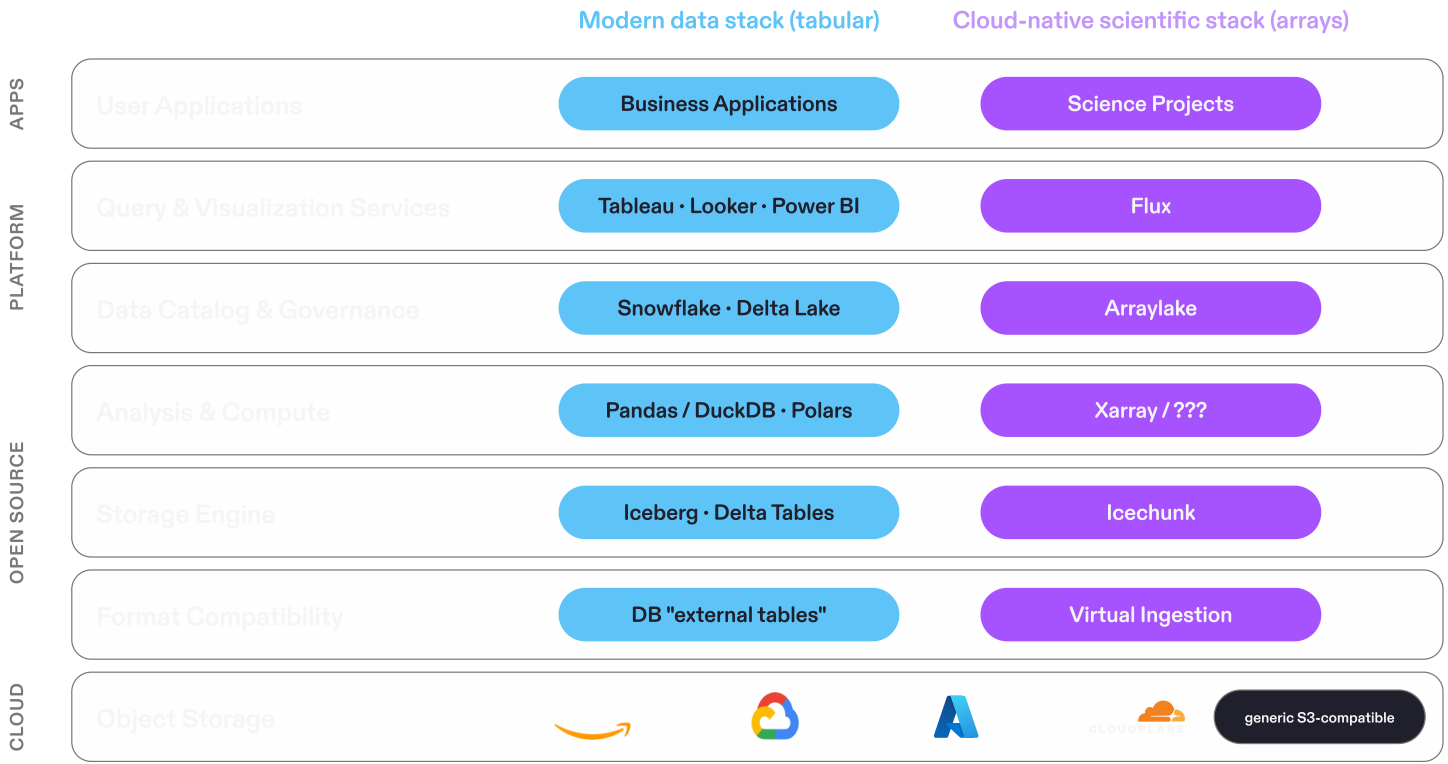
HISTORY

how did we get here?



PRESENT

are we modern yet?



FUTURE

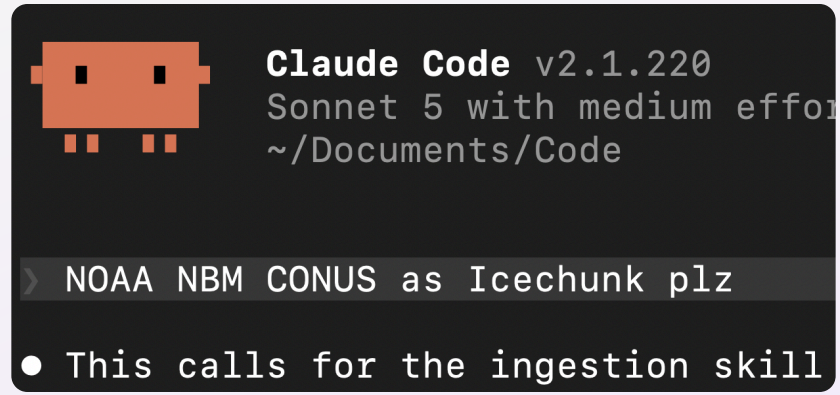
what next?

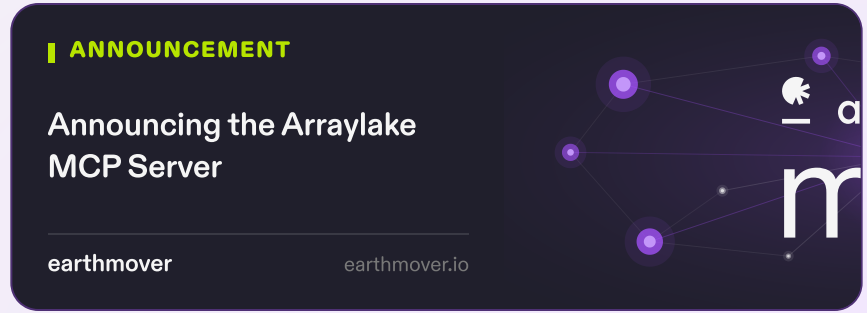
AI-ASSISTED INGESTION

avoid hand-writing dataset-specific ingestion code?

AI agents can do it.

- Works best when agents have powerful tools to call — e.g. Xarray, VirtualiZarr and Icechunk. (Arraylake MCP also helps)
- Great, but needs supervision at PB-scale
- LLMs can also recognize when virtual ingest can't work
- Try the icechunk-datacube-ingestion skill (in Rich's workshop!)





FUTURE WORK

remaining challenges

- **Make ingestion even easier**
  - Ingestion as a service?
  - Expertise still required for extremely large datasets (e.g. manifest splitting, coordinated writes)
- **Expand what we can virtualize**
  - Variable-length chunks + other types of inter-file inconsistencies (e.g. varying codecs, cf encoding)
  - Pathological file formats (e.g. Radar formats)
  - Non-datacube-like datasets (e.g. Level 2 data)
- **Improve access and discoverability**
  - Arraylake free tier coming soon
  - NASA Earthdata is an island - Earthdata Login integration?
  - Better non-Python tooling support

CONCLUSION

virtual stores solve many problems

- Simple idea of referencing existing data solves:
  - Query performance and storage costs
  - Cataloging and accessibility
  - Consistency
  - Agility
- This technology is the result of learning-by-doing
- State of the art: AI-assisted virtual ingestion into Icechunk, distributed via Arraylake

BONUS SLIDES

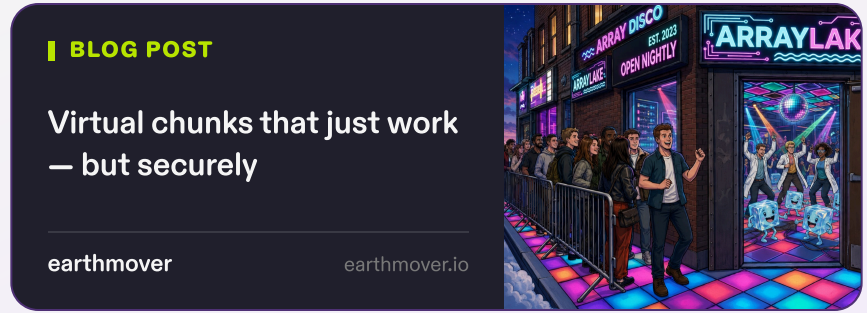
in case there’s time (or questions)

AUTHENTICATION SUBTLETIES

virtual chunks cross trust boundaries

Virtual chunks reference **arbitrary locations** containing **arbitrary data** — that has real security implications.

- **Exfiltration** — a malicious manifest could reference `file:///bitcoin-wallet`, then just ask you to "load this Icechunk repo"
- **Overexposure** — a misconfigured prefix can expose an entire private bucket
- **Tampering** — archival files updated in place invalidate assumptions baked into a manifest
- Fix: a **trusted server** validates chunk storage locations and dispenses credentials only for buckets a user is



KERCHUNK VS VIRTUALIZARR

manifest formats side-by-side

	Icechunk	VirtualiZarr	Kerchunk	DMR++	LanceDB	Iceberg
Logical data model	Zarr (but flexible)	Zarr	Zarr (but flexible)	HDF5?	tabular	tabular
Mixed chunk types	✅	✅ (per-array)	✅ (not native)	❌	✅	?
Efficient at scale	✅	✅	❌ (JSON/dict), ✅ (Parquet)	❌	✅	✅
Prefix management	✅ (VCCs)	✅ (registry)	❌	❌	✅	?
Runtime I/O layer	✅	✅ (obstore)	✅ (fsspec)	✅ (Hyrax)	✅	✅
Serverless data access	✅	✅	✅	❌	✅	✅
Change detection	✅	n/a	❌	❌	✅	✅
Transactions / versioning	✅	n/a	❌	❌	✅	✅
Serialized format	IC flatbuffers	n/a	JSON / Parquet	XML	lance	parquet

HOW DOES VIRTUALIZARR SCALE?

parallelizing manifest construction

Building references over huge archives (millions of files) needs to be parallelized like any other big-data job.

- Parallelization backends: **Dask** and **Lithops** — same manifest-construction logic, just fanned out across workers
- In production on GOES-16 (7 billion chunks)**: writing the manifests dominated,** not parsing files or concatenating references
- The final `to_icechunk()` commit is a serial tail — Amdahl's law bites no matter how wide you parallelize the rest
- Manifest splitting is one of the few tuning knobs left for virtual stores — split along time for ingestion, along space for timeseries, but not so much that a snapshot ends up with too many manifests

MORE ANIMATIONS

see the access patterns for yourself

Illustrated, interactive comparisons of native vs. virtual vs. cloud-optimized access patterns, click through at:

https://earth-mover.github.io/icechunk-illustrated/

ZAX

query engine for array data

Zax is a modern, pure-Rust query engine for NetCDF-style data, built directly on Icechunk.

- **Schema-first** — datasets are fully described before any I/O occurs
- **Lazy by default** — builds a query plan, no I/O until `.collect()`
- **Chunk-aware execution** — bounded-memory, innermost-first tiling
- SQL on array data via **DataFusion** — variables become columns, dimensions broadcast to rows
- **Client / server** — develop locally with Xarray or Zax, execute remotely when scale demands it

pandas : polars :: xarray : zax

```
ds = zax.open_icechunk(repo, branch="main")
ds_region = ds[['temperature']].sel(
    lat=slice(45, 60), lon=slice(0, 90)
)
fragment = ds_region.collect()  # bounded memory
```

earthmover

earthmover.io