

# STAC Standards & Roles

*What the standard is, and who steers it*

# Who I am

- ◆ Remote sensing for about thirty years. Open source for the last fifteen years
- ◆ I am one of the seven members of the **STAC Project Steering Committee**
- ◆ I work at LGND AI, but **this is not an LGND talk**. Nothing here is a product

So when I say "we" today, I mean the community that maintains STAC.

# Overview

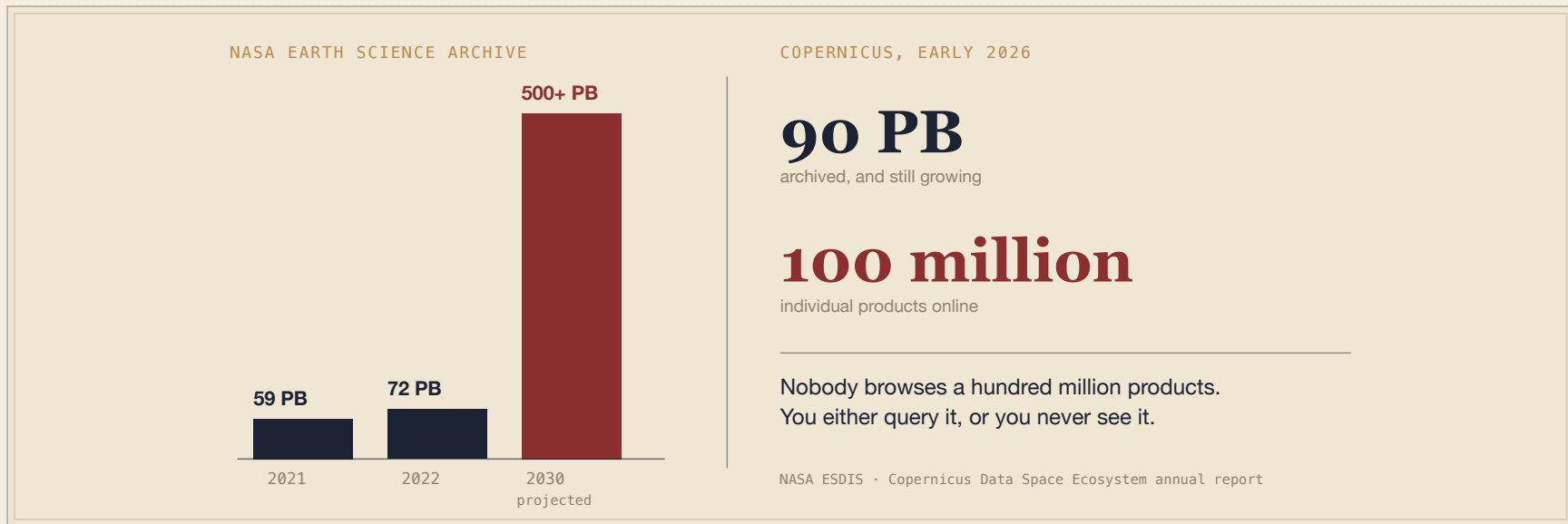
- 
- 1 **What STAC is** — the model, and an Item out of your own catalog
  - 2 **What it enables** — discovery, usage, and workflows
  - 3 **How STAC fits with other standards** — the OGC family, and what STAC deliberately does not do
  - 4 **Who steers it** — the committee, the vote, and how the spec moves
  - 5 **JAXA and STAC** — what you already run, and how to get a say in it
- 

A Q&A with the whole committee follows.

# 1 What STAC is

---

# Too much to browse



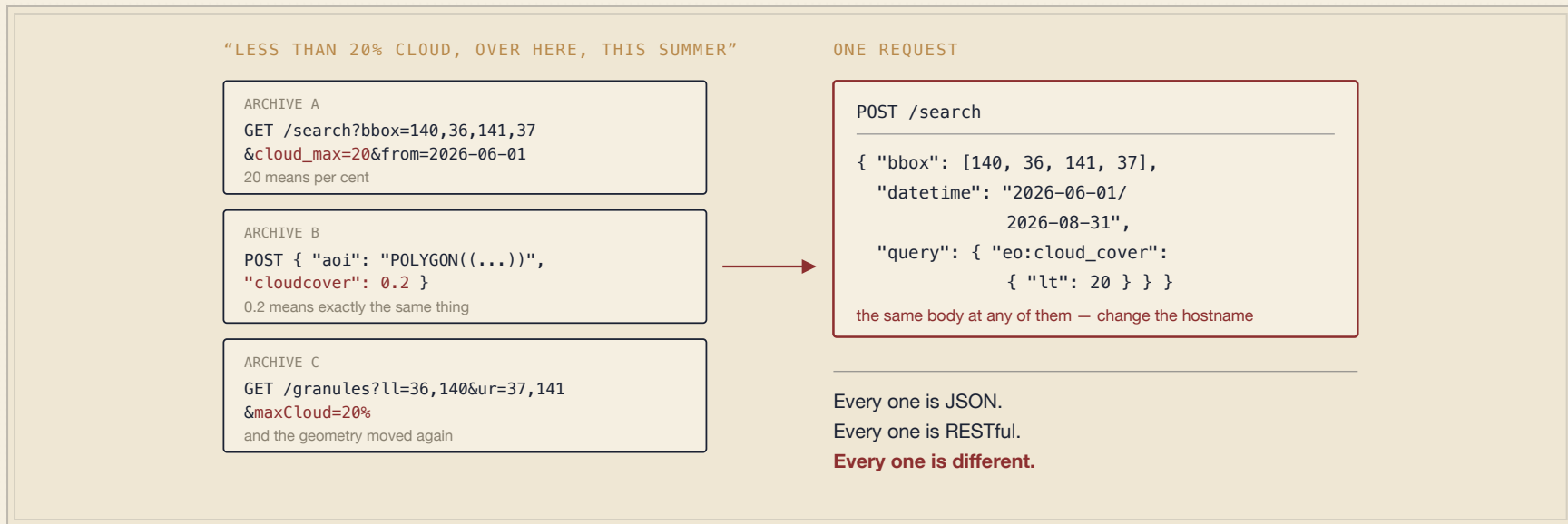
## The archives outgrew browsing

*NASA's Earth science archive went from 59 PB in 2021 to 72 PB a year later, and is projected past 500 PB by 2030. Copernicus already holds more than 90 PB — and over a hundred million separate products.*

If your metadata is not indexed,  
*your data does not exist.*

I have been saying this for years. I still think it is the most important point.

# Every archive had its own API



## The same question, three archives

*Ask three archives for less than 20% cloud and you write it three times. One expects 20, the next 0.2, the third a string with a per-cent sign — and the geometry moves as well.*

STAC is a suite of specifications and an open-source ecosystem for

*describing and searching* **geospatial data.**

# Three pieces

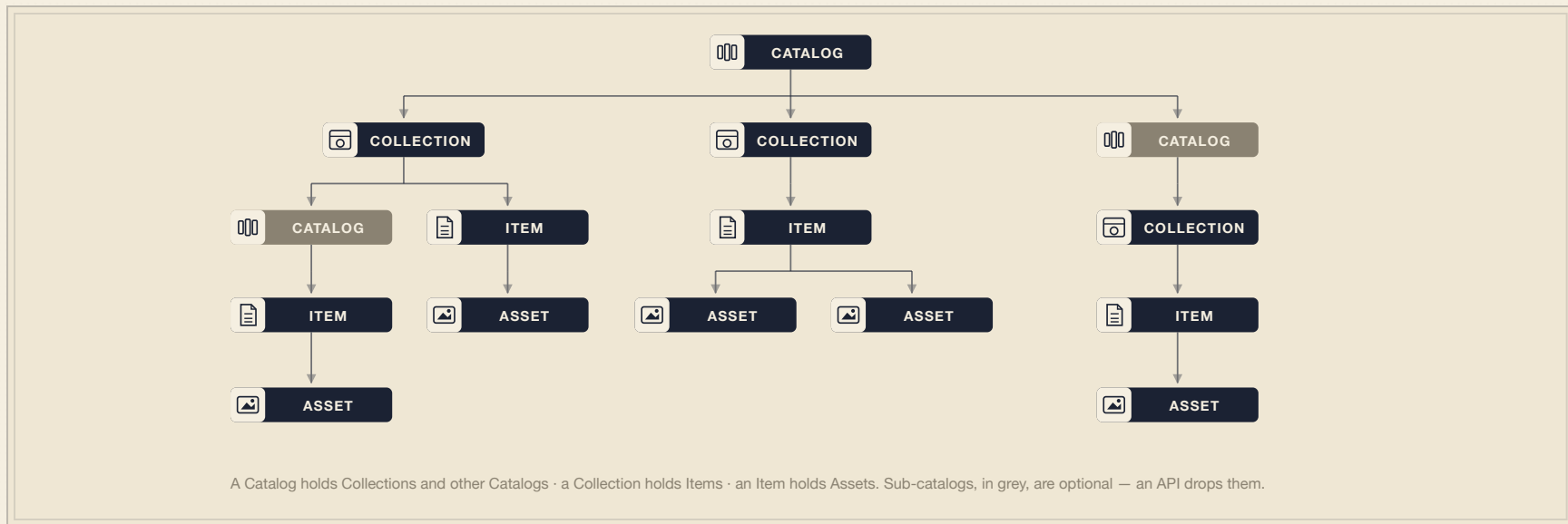
---

|                      |                                                      |
|----------------------|------------------------------------------------------|
| <b>stac-spec</b>     | The metadata language for describing geospatial data |
| <b>stac-api-spec</b> | Searching and accessing that metadata through an API |
| <b>Extensions</b>    | Everything the core deliberately does not cover      |

---

Two specifications share one object model. Extensions are how it grows.

# Four entities



## Catalog, Collection, Item, Asset

*A Catalog is a container. A Collection groups Items that are alike. An Item is one scene at one place and time. Assets are the files. If you remember one thing, remember the Item.*

# A Collection is your product

A **Collection** is a dataset — the thing that has a name, a licence, and a citation. It is the level your users think in, and the level they cite.

---

**Required**    `id` · `description` · `license` · `extent` (spatial and temporal) · `links`

---

**Usually**    `title` · `keywords` · `providers` · `summaries` — the range of values across everything in it

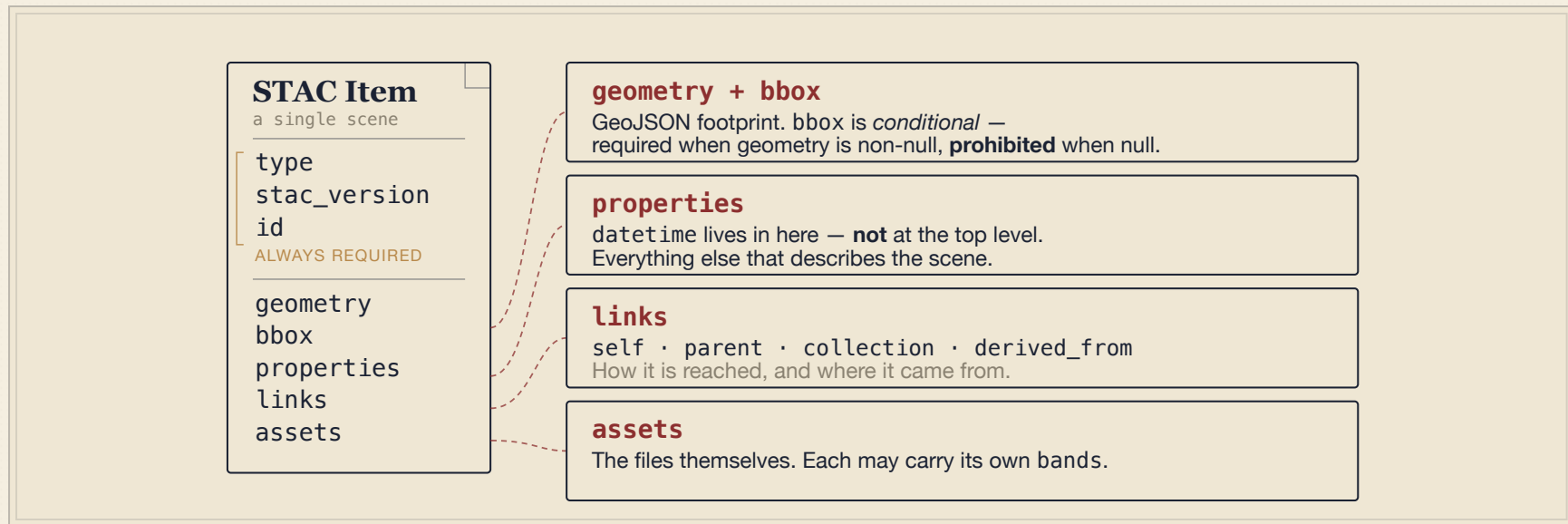
---

**Often**    a DOI, via the `scientific` extension — which your collections already declare

---

This is the unit someone subscribes to, cites, and decides whether to trust. What it holds gets reprocessed and replaced over the years.

# An Item



## The document, exploded

*The required fields, and what each one is for. bbox is conditional on geometry, and datetime is inside properties, not at the top level.*

# A JAXA Item

```
{
  "type": "Feature",
  "id": "N36.00-N37.00",
  "collection": "JAXA.EORC_ALOS.PRISM_AW3D30.v3.2_global",
  "bbox": [140.0, 36.0, 141.0, 37.0],
  "geometry": { "type": "Polygon", "coordinates": [ ... ] },
  "properties": {
    "start_datetime": "2021-02-01T00:00:00Z",
    "end_datetime": "2021-02-28T23:59:59Z"
  },
  "assets": {
    "DSM": { "href": "./E140.00-N36.00-E141.00-N37.00-DSM.tiff",
              "type": "image/tiff; application=geotiff; profile=cloud-optimized",
              "roles": [ "data" ] },
    "MSK": { "href": "./E140.00-N36.00-E141.00-N37.00-MSK.tiff", "roles": [ "data-mask" ] }
  }
}
```

Where, when, what, and a link to each file.

# An Item on a map



## The AW3D30 tile, in STAC Browser

*The same Item from your catalog, opened in STAC Browser. The outline is the Item geometry — E140–E141, N36–N37 — and つくば市 sits just inside the bottom edge. Basemap OpenStreetMap.*

# The STAC API

A static catalog is a tree of files you crawl. An API is the same Items behind a search endpoint.

|                                      |                                                                                                  |
|--------------------------------------|--------------------------------------------------------------------------------------------------|
| <code>/</code>                       | The landing page. Declares <code>conformsTo</code> — which parts this server actually implements |
| <code>/collections</code>            | What the catalog holds                                                                           |
| <code>/search</code>                 | <b>One query across collections</b> — space, time, and properties, in one request                |
| <code>/collections/{id}/items</code> | This part <i>is</i> OGC API - Features                                                           |

A client reads `conformsTo` and knows what it is allowed to ask for. That is how one library works against every STAC API — and the same client reads a static catalog, without the server.

# Ninety-three extensions

An extension advances only when people implement it — not when people agree with it.

| MATURITY      | IMPLEMENTATIONS REQUIRED | HOW MANY |
|---------------|--------------------------|----------|
| Proposal      | 0                        | 42       |
| Pilot         | 1                        | 16       |
| Candidate     | 3                        | 6        |
| <b>Stable</b> | <b>6</b>                 | <b>8</b> |
| Deprecated    | —                        | 8        |

Eighty above, plus ten in progress and three unlabelled.

# Why it has lasted

- ◆ Built on standards that already existed — JSON, GeoJSON, OGC APIs
- ◆ A core that is **small and stable** on purpose
- ◆ Metadata **decoupled** from the data it describes
- ◆ Open-source tools arrived with the specification, not years later

The core spec never tried to describe everything. It was trying to cover about eighty percent of what people needed, and leave the rest to extensions.

## 2 What it enables

---

# Three things people use STAC for

---

|                  |                                                                                                                                                      |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Discovery</b> | <i>Find</i> the data. Space, time, collection — plus <b>content extensions</b> that add searchable fields — Electro-Optical, SAR, View               |
| <b>Usage</b>     | <i>Open</i> the data. The assets, plus <b>usage extensions</b> that describe it well enough to read without opening it — Projection, Raster, Storage |
| <b>Workflows</b> | <i>Chain</i> the processing. STAC in, STAC out — the Items become the interface between steps                                                        |

---

**The two kinds of extension are the split worth remembering.** One makes the search better. The other makes the data usable by a machine that has never seen it before.

# One query, many archives

The same request works against any STAC API. A bounding box and a date range.

```
POST /search
{ "bbox": [140.0, 36.0, 141.2, 37.1],
  "datetime": "2026-06-01/2026-08-31" }
```

Change the hostname and the same request searches somebody else's archive.

# You stop reverse-engineering every dataset

I wrote this a while ago and I still think it is the clearest way to say it:

*Every new dataset, whether from a government agency or a commercial provider, was its own puzzle. You couldn't just find the data; you had to **reverse-engineer the means to use it**.*

An Item tells you the footprint, the date, the cloud cover, the resolution, and the media type of every file — **before you open any of them**. Same fields, every catalog, every mission.

Discovery was the goal.

*Metadata in one place, in one shape* **is**  
**what let the workflows scale.**

Still how you find the files. Also what the processing runs on.

# Nothing is read until you ask

A search result is not a list of files to fetch. It is enough information to plan the whole computation before touching any of them.

---

**Discover** The search returns the Items — which scenes, and the URL of every file

---

**Describe** `proj:` carries the CRS, shape and transform. The band metadata carries data type and nodata. **The cube's geometry is known from the metadata alone**

---

**Defer** `chunks=` builds a Dask graph. A cube spanning terabytes exists as a plan, not as pixels

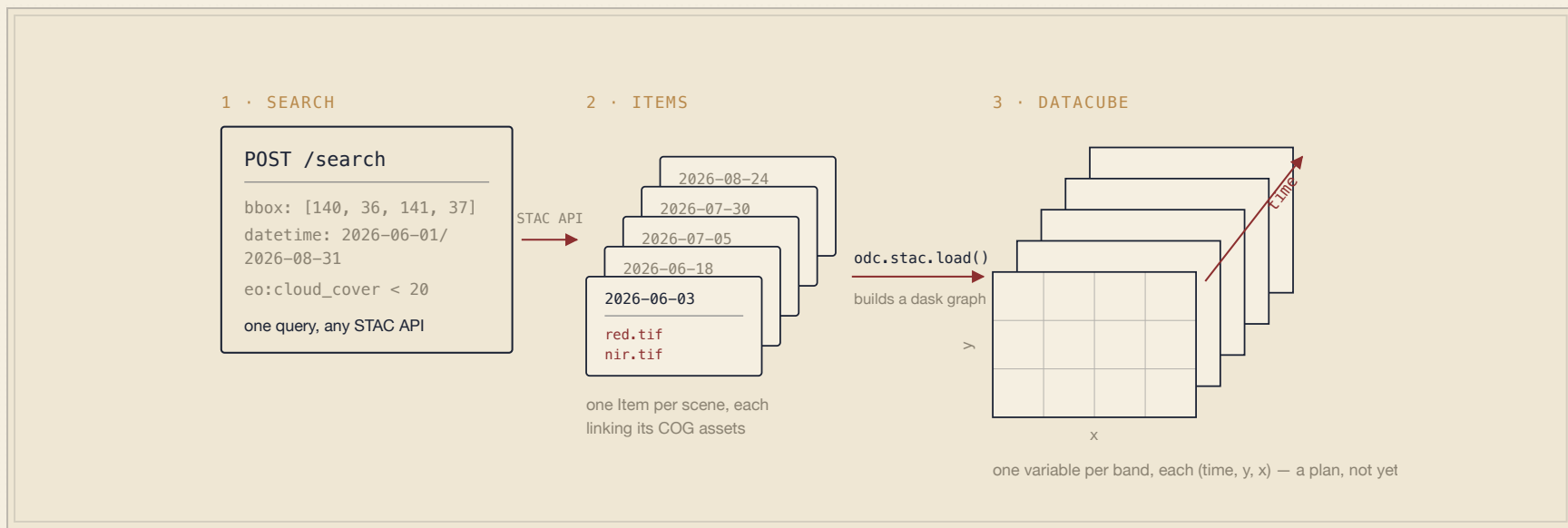
---

**Read** Only when you compute — and each chunk is a range request for the tiles it needs

---

**This is what the extensions are for.** Without `proj:`, the loader has to open every file to find out where it is — which is the download you were trying to avoid.

# One search, one cube



## From one search to one cube

*The query goes to any STAC API, comes back as one Item per scene, and odc-stac stacks those Items into an xarray cube with a time dimension. No download step anywhere in the chain.*

# Search, plan, compute

```
from pystac_client import Client
import odc.stac

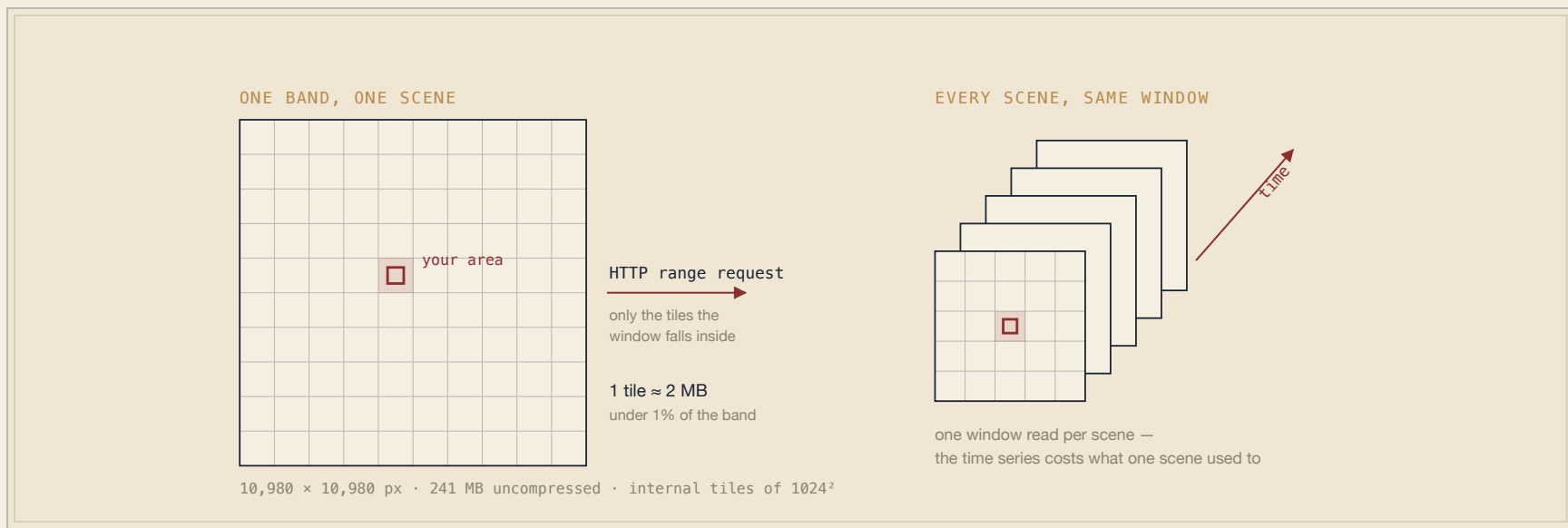
items = Client.open("https://earth-search.aws.element84.com/v1").search(
    collections=["sentinel-2-l2a"],
    bbox=[140.0, 36.0, 141.0, 37.0],
    datetime="2026-06-01/2026-08-31",
    query={"eo:cloud_cover": {"lt": 20}},
).item_collection()

cube = odc.stac.load(items, bands=["red", "nir"], resolution=10,
    bbox=[140.0, 36.0, 141.0, 37.0],
    chunks={"time": 1, "x": 2048, "y": 2048})

ndvi = (cube.nir - cube.red) / (cube.nir + cube.red)
ndvi.mean("time").compute()
```

Same bounding box as the Item earlier. Everything above `compute()` is a plan — no pixel is read until the last line, and then only the pixels that line needs.

# What one chunk reads

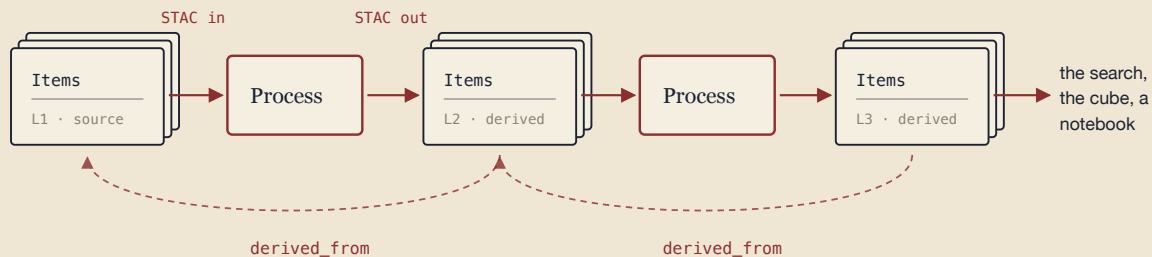


## The window read

*The bottom of the chain. A 200 × 200 pixel field lands inside a single 1024 × 1024 block, so a range request fetches 2 MB of a 241 MB band. Repeat per scene: the time series costs what one scene used to.*

# STAC in, STAC out

## METADATA IN, METADATA OUT

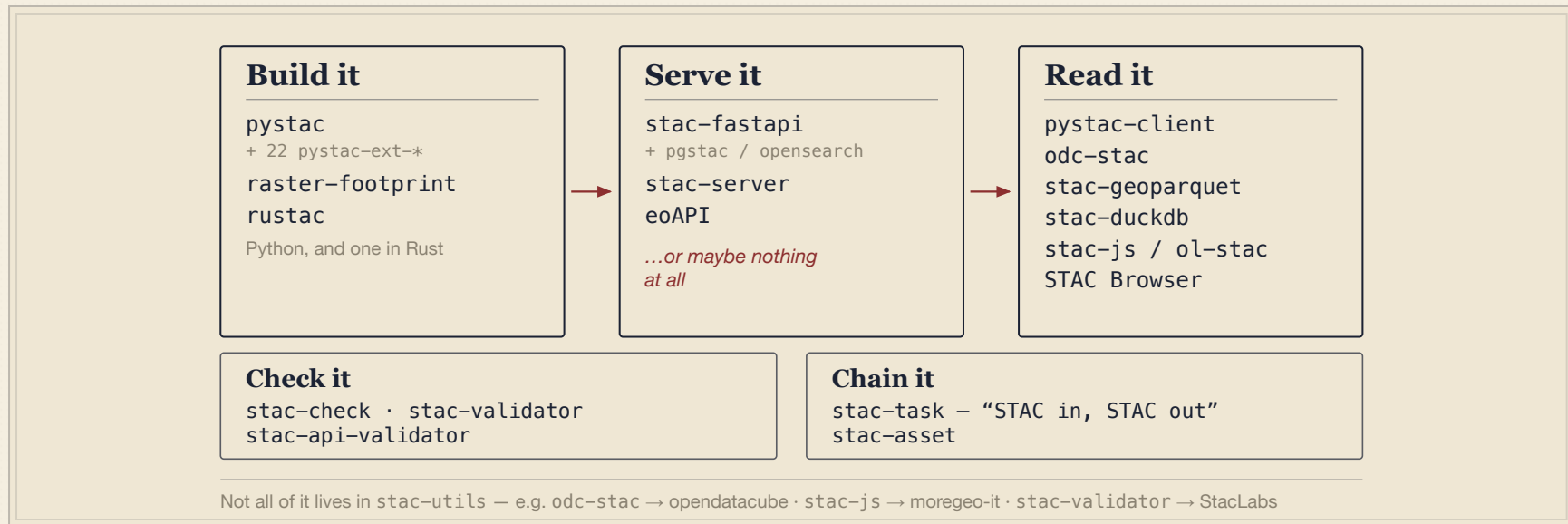


The file paths live inside the Items. The interface between steps is the metadata —  
so every step writes a catalog, and the chain records its own lineage as it goes.

## Metadata in, metadata out

*Each step takes Items and emits Items, so L1 to L2 to L3 stops being three bespoke scripts that agree by convention. stac-task is built around exactly this, and a derived\_from link on each output means the chain records its own lineage.*

# The ecosystem



## The ecosystem, one picture

*The open-source tools, laid out by what they do to a catalog — build it, serve it, read it, check it, chain it — with the organisations that maintain them.*

# 3 How STAC fits with other standards

---

# What STAC is not

## NOT A DATA FORMAT

STAC describes files. It does not replace GeoTIFF, COG, NetCDF or Zarr.

## NOT A DATABASE

It is a document model. You will still choose how to index and serve it.

## NOT A PROCESSING SYSTEM

It tells you what exists and where. Processing the data is a separate job.

## NOT A PORTAL

There is no single STAC website. There are many catalogs, and an ecosystem of tooling to use them.

# The OGC API family

| OGC API   | STATUS            | WHAT IT IS FOR                                                                  |
|-----------|-------------------|---------------------------------------------------------------------------------|
| Features  | Approved          | Serving vector features — <b>STAC API - Features is built on it</b>             |
| Records   | Approved (Part 1) | Searching metadata. The closest neighbour to STAC                               |
| Coverages | Draft             | Subsetting gridded data — the natural partner for an EO raster archive          |
| EDR       | Approved 1.1.0    | Values at a point, area, trajectory or corridor. A time series without the file |
| Tiles     | Approved          | Serving data as tiles                                                           |
| Maps      | Approved          | Serving rendered map layers                                                     |
| Processes | Approved          | Invoking processing                                                             |

Also approved: Common, DGGS, Connected Systems, Moving Features. Still draft: Styles, Routes, 3D GeoVolumes. Together they replace the WFS / WCS / WMS / WMTS / WPS / CSW generation.

A STAC Item *is* a valid OGC Feature — the geometry is the area the data covers.

# Who does what

The family divides the work. STAC only does the first row.

|               |                                                                                                  |
|---------------|--------------------------------------------------------------------------------------------------|
| Find it       | STAC · Records — what exists, where, and when                                                    |
| Get the data  | Features (vector) · Coverages (a subset of a grid) · EDR (values at a point, area or trajectory) |
| Look at it    | Tiles · Maps                                                                                     |
| Compute on it | Processes                                                                                        |

STAC tells you a file exists and hands you a URL. It does not serve you pixels, render a map, or run anything. That is the design, not a gap. It is why STAC sits *in front of* these rather than competing with them.

# STAC and OGC API - Records

Two catalog standards. This is the one place people genuinely get confused.

|                | OGC API - RECORDS                                                                                 | STAC                                                               |
|----------------|---------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|
| Catalogs       | Any resource — features, coverages, tiles, styles, services, processes, machine models, notebooks | Data files you are going to open                                   |
| The unit       | A <i>record</i> : metadata about a thing                                                          | An <i>Item</i> : a GeoJSON Feature with a time and a set of assets |
| The constraint | A record needs an id. Geometry and time are optional                                              | An Item <b>must</b> have a time, and <b>must</b> link to assets    |
| Status         | Part 1 Core approved 1.0.0. Parts 2 and 3 draft                                                   | Core 1.1.0 and API 1.0.0, both OGC Community Standards             |

That third row is the whole difference. STAC is Features with time made mandatory and a file on the end.

# STAC is an OGC Community Standard

Core v1.1.0 and API v1.0.0 were approved in September 2025 and published on 14 October 2025.

- ◆ Formal international recognition — a document number, a citable reference
- ◆ **Not a handover.** OGC adopted the releases as fixed-content standards
- ◆ Day-to-day work stays on GitHub, under the PSC

The standards documents say so themselves: the originating party *"may continue to update their work; however, this document is fixed in content."*

# Who already runs STAC

A few of the large EO catalogs — nowhere near a complete list.

---

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <b>Element 84</b> | Earth Search — Sentinel-2, Landsat, and more on AWS |
|-------------------|-----------------------------------------------------|

---

|                  |                                     |
|------------------|-------------------------------------|
| <b>Microsoft</b> | Planetary Computer, 136 collections |
|------------------|-------------------------------------|

---

|             |                                  |
|-------------|----------------------------------|
| <b>USGS</b> | landsatlook — serving STAC 1.1.0 |
|-------------|----------------------------------|

---

|             |                    |
|-------------|--------------------|
| <b>NASA</b> | CMR-STAC, and VEDA |
|-------------|--------------------|

---

|             |                                        |
|-------------|----------------------------------------|
| <b>JAXA</b> | 112 collections — more on this shortly |
|-------------|----------------------------------------|

Beyond Earth observation: **Overture Maps** distributes building footprints through a STAC catalog, and the **UK Met Office** publishes weather forecasts through one.

\*\*And lots more! [stacindex.org](https://stacindex.org) lists **144** catalogs today — **83 of them static files**, not APIs. Small catalogs are the normal case, not the exception.

# Slow change is deliberate

- ◆ **The STAC API has not changed since April 2023** — three years. Core has been at v1.1.0 since September 2024
- ◆ Breaking changes can only happen in a 2.0, so a catalog you write today stays valid
- ◆ **Implementations do not go stale.** Tools written three years ago still work against catalogs published this morning

A specification that can be taught the same way two years later is not stalled. It is finished enough to build on. For an agency planning a ten-year archive, I think that stability is the whole value.

# Three ways to publish

The spec has been stable since 2024. How you *serve* it has kept moving — and that is the good news: the same Items scale from a folder to a petabyte.

---

## Static JSON

A tree of files on object storage. No server, no database, nothing to operate. This is how most catalogs start

---

## stac-geoparquet

The same Items as one Parquet file. One range request answers a query instead of thousands. The entire Sentinel-2 L2A archive is a **7 GB file** you can point DuckDB at

---

## A database and an API

`pgstac` + `stac-fastapi` . Live search over an archive that changes constantly

---

Three deployments, one specification. **You choose by the size of your catalog, not by rewriting your metadata** — and you can move between them later without asking your users to change anything.

# 4 Who steers it

---

# The Project Steering Committee

|                                |                  |
|--------------------------------|------------------|
| Matthias Mohr ( <i>chair</i> ) | moreGeo          |
| Alex Leith                     | Auspatious       |
| Emmanuel Mathot                | Development Seed |
| Julia Signell                  | Development Seed |
| Matthew Hanson                 | LGND AI          |
| Michelle Roby                  | Radiant Earth    |
| Pete Gadomski                  | Development Seed |

Seven seats, and **no company holds a majority of them**. Members are nominated by the PSC or by contributors and admitted on a majority vote; a seat inactive for twelve months can be reassigned.

**Disclosure:** I am one of them, so I am not a neutral observer of STAC governance. Guidelines accepted March 2025 · roster and rules at [radiantearth/stac-psc](https://radiantearth/stac-psc)

# How a decision gets made

Written down, in public, in the same repository as the specification.

---

## An ordinary change

A pull request. It needs **two core-team reviews** — a release needs three

---

## When consensus fails

It goes to a PSC vote, open a minimum of **five business days**

---

## The ballot

**+1** support and willing to help · **-1** veto · **±0** mild view · **0** no opinion

---

## To pass

A majority including the proposer, **and no vetoes**

---

## A veto

Comes with reasons *and* an alternative. Overriding one needs a majority of **all** eligible voters, not just those who turned up

---

The chair tracks membership and arbitrates when the process breaks down — and votes like everyone else. A proposal nobody votes on lapses after twenty business days.

# How direction gets set

All of it happens in the open, in the same repositories anyone can read. Direction comes out of community sprints and the milestone buckets on GitHub.

- ◆ The specification, the API, the extensions and the tools are separate repos running the same process
- ◆ Best practices were moved out of the spec into **their own repo** in August 2025 — seven topic files plus an implementation guide
- ◆ Sprints are where direction actually gets decided. **This week is one of them**

If you want to change STAC: *open an issue, and come to one of the community calls.*

# 5 JAXA and STAC

---

Open science needs data that is  
*machine-findable* ,  
not just publicly downloadable.

# You already run STAC

`data.earth.jaxa.jp/stac/cog/v1/catalog.json` — a static STAC 1.0.0 catalog, **112 collections**.

- ◆ **96 JAXA collections.** Plus **14 NASA, 1 Copernicus, 1 JMA**
- ◆ Anyone can read all of it without an account or an API key

The NASA, Copernicus and JMA collections are the part I would point at — you are indexing other agencies' data alongside your own. That only works because it is the same standard on both sides.

Counted from the live catalog, 24 August 2026.

# How JAXA can influence STAC

You do not need permission, and you do not need to be a member. You only need to turn up.

---

## The calls

Every other Monday, 11:00 ET — midnight here. And an **Asia-Oceania call, first Friday of the month, 12:00 Japan time**, which exists so this region does not have to join in the night

---

## The channels

`#stac` on the Cloud-Native Geospatial Slack, and the `stac-community` Google Group

---

## Your own extension

An extension can live in your own organisation. You do not need our permission to write one

---

## This week

Most of the people who maintain this specification are in this building

---

The thing that helps most is telling us what does not work. **We only learn about a gap when someone reports it** — so if STAC does not do what JAXA needs, that is the useful thing to say. We are here until Thursday, and we should use the time.

# Thank you