

Why GeoID and DynaStore exist

Stable identity for every written feature.
Dynamic data routing for every workload.

Architecture notes from building an OGC/STAC-native geospatial platform



UUIDv7

READ • WRITE • INDEX • UPLOAD

Two names describe one architecture

IDENTITY

GeoID

Written geospatial features receive a UUIDv7 identifier: globally unique, time-sortable and stable across storage, API and indexing paths.

Identity is part of the data model — not an afterthought.

ROUTING

DynaStore

READ, WRITE, INDEX and UPLOAD operations are resolved at runtime against scoped configuration, hints and driver capabilities.

The workload chooses the path; the API stays stable.

GeoID makes identity useful, not incidental

UUIDv7 combines Unix time, version bits, a monotonic counter and randomness — ordered without giving up uniqueness.

01 INGEST

Accept a feature

Preserve a client-supplied public id when present; otherwise expose the geoid.

02 IDENTIFY

Assign UUIDv7

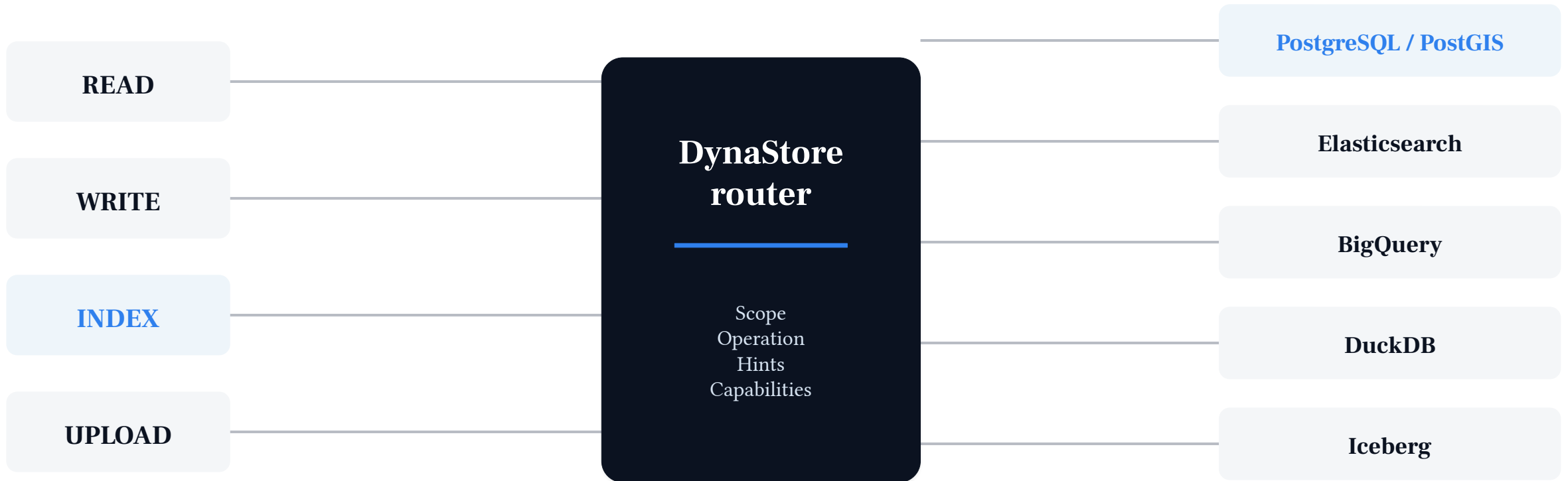
Generate a monotonic UUIDv7 for written features at write time.

03 RESOLVE

Use one identity

Carry stable identity through queries, indexes and asynchronous tasks.

DynaStore chooses the data path at runtime



The API remains consistent while each workload reaches the most suitable backend.

Drivers, formats and access paths stay distinct

PostgreSQL / PostGIS

Durable writes and exact spatial geometry.

Elasticsearch / OpenSearch

Search, indexing and simplified geometry paths.

BigQuery

Analytical reads, counts and aggregation.

DuckDB

Fast local analytical reads across file formats.

Iceberg

Snapshots, time travel and schema evolution.

Parquet / GeoParquet

Columnar files and warehouse data layout.

GDAL / VSI

Raster and cloud-optimized dataset access.

Object storage

GCS, S3 or local warehouse locations.

The same core can run GCP-native or on premises

Cloud integration is added through protocols, drivers and tasks. The core architecture does not require a single deployment environment.

GCP-native

- Direct resumable uploads to GCS
- Pub/Sub object-event processing
- BigQuery and GCS-backed Iceberg paths
- Serverless APIs and isolated task workers

On premises / self-managed

- Containerized APIs and workers
- PostgreSQL / PostGIS as the durable core
- Elasticsearch / OpenSearch and DuckDB
- Local, S3 or configured object storage

SAME CORE CONTRACTS

Open standards turn the platform into an integration surface

01 Discover & query

STAC API, OGC API – Features and Records expose catalogs without bespoke connectors.

02 Visualize

Tiles, Maps and Styles serve vector and raster content through standard routes.

03 Analyze

Coverages, EDR, DGGS and Joins address multidimensional and analytical use cases.

04 Operate

Processes, Connected Systems and Moving Features extend the same modular platform.

Two demos show the architecture under real workloads

The next articles will unpack what happens behind each demonstration — the path, trade-offs and engineering decisions.

01

From a COG to a styled raster map

Raster support through OGC API Tiles, with styles applied through the platform's cartographic surface.

youtu.be/j-lRrwAhWDI

02

Eight million features, uncached

A first-load demonstration that exercises the platform without relying on a warmed cache.

youtu.be/PmtZ37t5HEo

THE SERIES STARTS HERE

I'm documenting the engineering decisions behind GeoID.

Identifiers. Dynamic storage. Open standards. Hybrid cloud.
Follow the series — or join the technical conversation.

Carlo Cancellieri

github.com/un-fao/GeoID

COMING NEXT

**COG → raster tiles +
styles**

**8M features —
uncached first load**