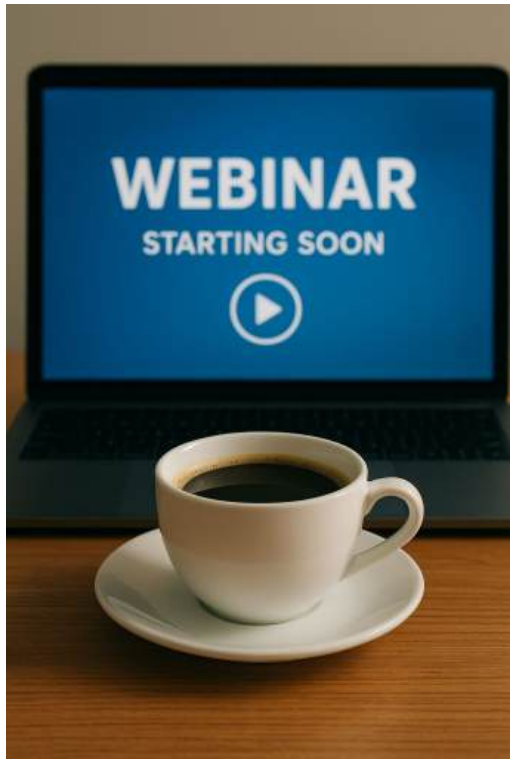
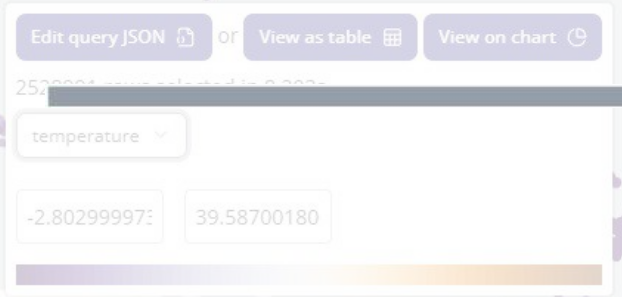
Goedemorgen.

We gaan binnen enkele ogenblikken starten.



DIGISHAPE
DATASCIENCE
FOR
DELTAS

MARIS



Beacon

A high-performance ARCO data lake for accessing & sub-setting large quantities of climate data

Robin Kooyman (MARIS)
Peter Thijsse (MARIS)
Tjerk Krijger (MARIS)

DIGISHAPE DATASCIENCE FOR DELTAS

10 December 2025

MARIS

Mededelingen vooraf

- NL of UK?

- Vragen graag op de chat. Geen interactie tot laatste sessie.
- We hebben tussentijds 2 momenten voor korte vragen, en aan het eind een ruimere sessie (15 mins).

- Het webinar wordt opgenomen, deelnemers blijven anoniem.
- Presentatie en opgenomen webinar worden gedeeld met alle deelnemers.

Agenda

09:00-09:05	**Introduction**	Peter
09:05-09:30	**Beacon Technology**	Robin
09:30-09:35	*Questions*	
09:35-09:40	**Beacon Provider** (demo)	Robin
09:40-09:45	*Questions*	
09:45-09:55	**Beacon User** (demo)	Tjerk
09:55-10:00	*Questions* & closure	Peter
10:00-10:15	*Overtime*	

Introduction (Peter)

Origin of Beacon

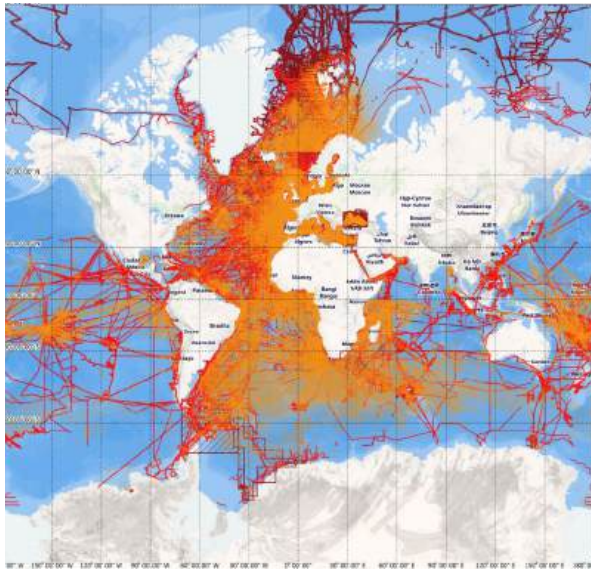
Challenge:

- Large collections of observation data (e.g. SeaDataNet CDI) can contain millions of files.
- Access to files (datasets) is possible, but data sub-setting proves to be difficult.
- How to optimize the systems for Machine2Machine access to subsets, enabling easy access to Jupyter Notebooks and other applications.
- How to go from files to serving applications as an actual “Data lake”?

Example:

- Request: Give me all the temperature data in the North Sea, from 2000-2010, in degrees Celsius, at a depth of 0-5 m.
- Response: One NetCDF file containing exactly this data, which is then on the fly, directly usable in a Jupyter notebook and for HPC.

SeaDataNet CDI

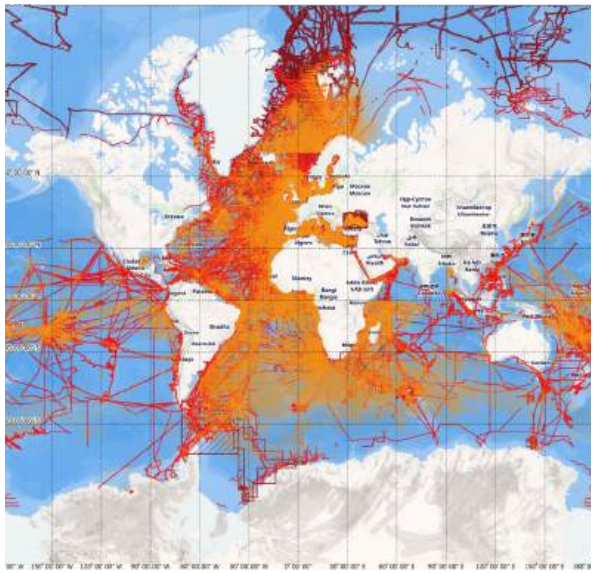


Beacon has been part of various EU projects such as PHIDIAS HPC, FAIR-EASE & EOSC-Future and is currently part of Blue-Cloud 2026 and national initiatives (a.o. IHM).

Beacon data lake

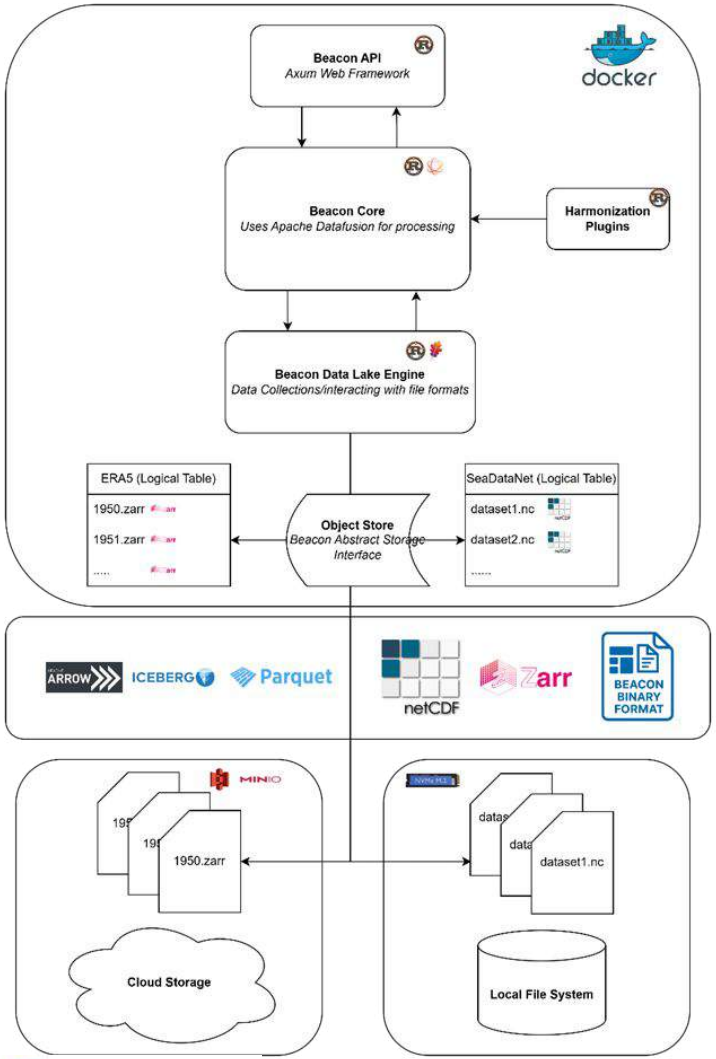
- Beacon (open-source) provides users with **fast and easy access** to multidisciplinary data originating from large collections, on the fly, with high performance, and is able to extract specific data based on the user's request.
- It returns **one single harmonized file as output,** regardless of whether the input contains different data types.
- Allows for:
  - Provider: To set-up their own Beacon 'instance' to enhance the access to their data;
  - User: To use one or more existing Beacon instances from for example well-known data infrastructures such as Euro-Argo, SeaDataNet or the World Ocean Database for fast and easy access to harmonized data subsets.
- Beacon **fast and easy to deploy**: Allows to set-up individual data lakes for any type of large data collection in a very timely and simple manner (less than 1/2 day for 10 million+ datasets).
- Support **harmonised access: Parameter mappings/grouping**, unit conversion, from different data collections, on the fly.
- **High performance**: Gives really fast results on large collections (i.e. 20 seconds request time for source set containing 100 million rows of data).
- Developments started from the perspective of the marine domain Applied to well-know marine data infrastructures But **proven to work in other domains too**.

SeaDataNet CDI



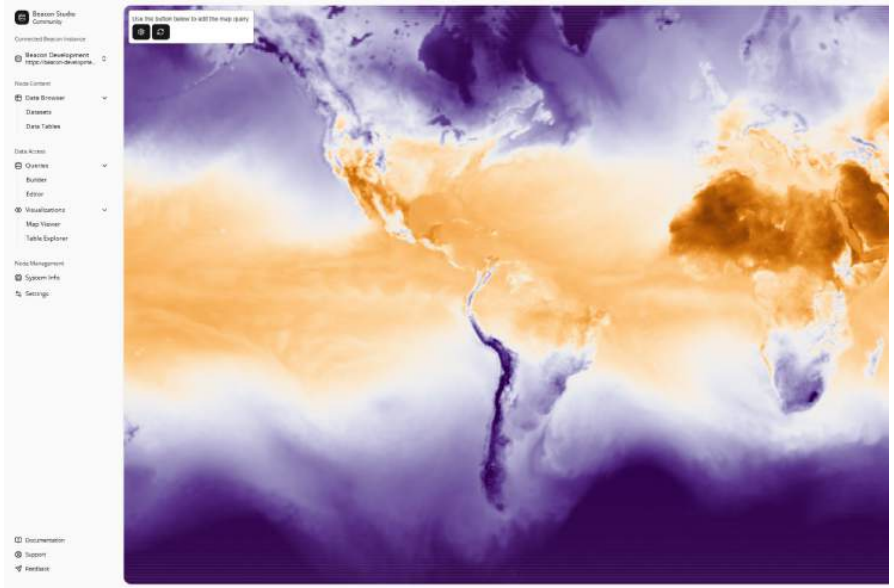
Beacon data lake

Technical details & specifications
Applications, set-up, tooling



Beacon studio

GUI with implementation examples



Beacon notebooks

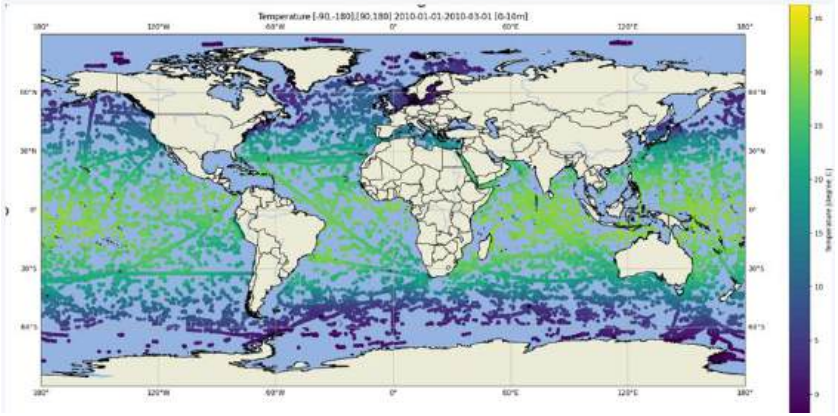
Scripts, implementation examples

```
response = requests.post("https://beacon-wod.maris.nl/api/query", json.dumps(query_body), headers={
    "Authorization": f"Bearer {Token}",
    "Content-type": "application/json"
})
```

datetime	Temperature	Unit	DEPTH	LONGITUDE	LATITUDE	DATASET	cruise-identifier	cast	WMO_ID	country	dataset_id
2010-01-01	24.700001	degree_C	1.0	-95.000000	-5.000000	moored buoy	US015955	12327900	2.115095e+09	UNITED STATES	11382402
2010-01-01	26.120001	degree_C	5.0	-110.000000	0.000000	moored buoy	US015969	15272646	2.115095e+09	UNITED STATES	11382406
2010-01-01	26.110001	degree_C	10.0	-110.000000	0.000000	moored buoy	US015969	15272646	2.115095e+09	UNITED STATES	11382406
2010-01-01	28.959999	degree_C	1.5	147.039993	4.960000	moored buoy	JP020811	12327885	2.115114e+09	JAPAN	11382401
2010-01-01	25.440001	degree_C	1.0	-37.880001	20.040001	moored buoy	US026451	12327924	2.115104e+09	UNITED STATES	11382426
...	...	...	...	...	...	...	...	...	...	...	...
2010-03-01	29.379999	degree_C	1.0	-34.639999	-18.860001	moored buoy	BR001204	12332326	2.115093e+09	BRAZIL	11390007
2010-03-01	27.370001	degree_C	1.0	-155.009995	7.980000	moored buoy	US015934	12332305	2.115114e+09	UNITED STATES	11389980
2010-03-01	29.860001	degree_C	1.0	-170.020004	-2.170000	moored buoy	US015927	12332281	2.115114e+09	UNITED STATES	11389956
2010-03-01	29.540001	degree_C	1.0	89.989998	-1.670000	moored buoy	JP032626	12327389	2.115115e+09	JAPAN	11389989
2010-03-01	28.430000	degree_C	1.0	165.130005	8.050000	moored buoy	US015890	12332307	2.115114e+09	UNITED STATES	11389966

308863 rows × 11 columns

In this example we can then very quickly plot this subset of the temperature data between 0-10 meters depth from the WOD collection of the first months of 2010 produced.



DIGISHAPE DATASCIENCE FOR DELTAS

10 December 2025

MARIS

Beacon Technology (Robin)

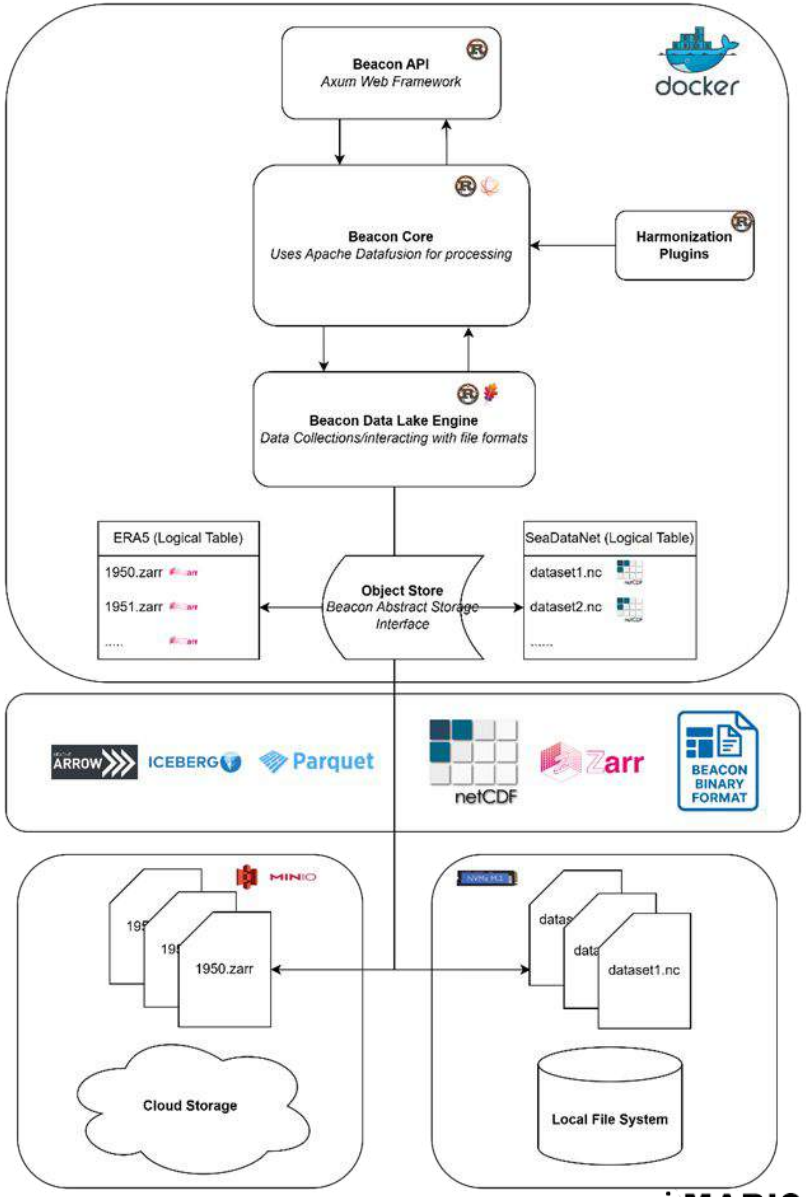
DIGISHAPE DATASCIENCE FOR DELTAS

10 December 2025

MARIS

Beacon in a nutshell

- **Open-Source High Performance ARCO Data Lake**
- **Written in Rust**
  - Performance
  - Memory Safety while having access to pointer arithmetic and assembly instructions such as SIMD.
  - Powerful zero-cost asynchronous programming engine (via tokio.rs).
  - Compiling to machine code
- **Runs on any platform using Docker**
- **Consists of:**
  - Rest API
    - Axum web framework
    - Auto-generated swagger from source code
  - Core Libraries
    - Data Lake Engine
      - Manages all the files & collections
    - Object Store
      - File Operations (reading/writing bytes)
    - Streaming Engine via Datafusion (extended for our needs)
    - Apache Arrow Arrays
  - Data Harmonization Libraries (single output file)

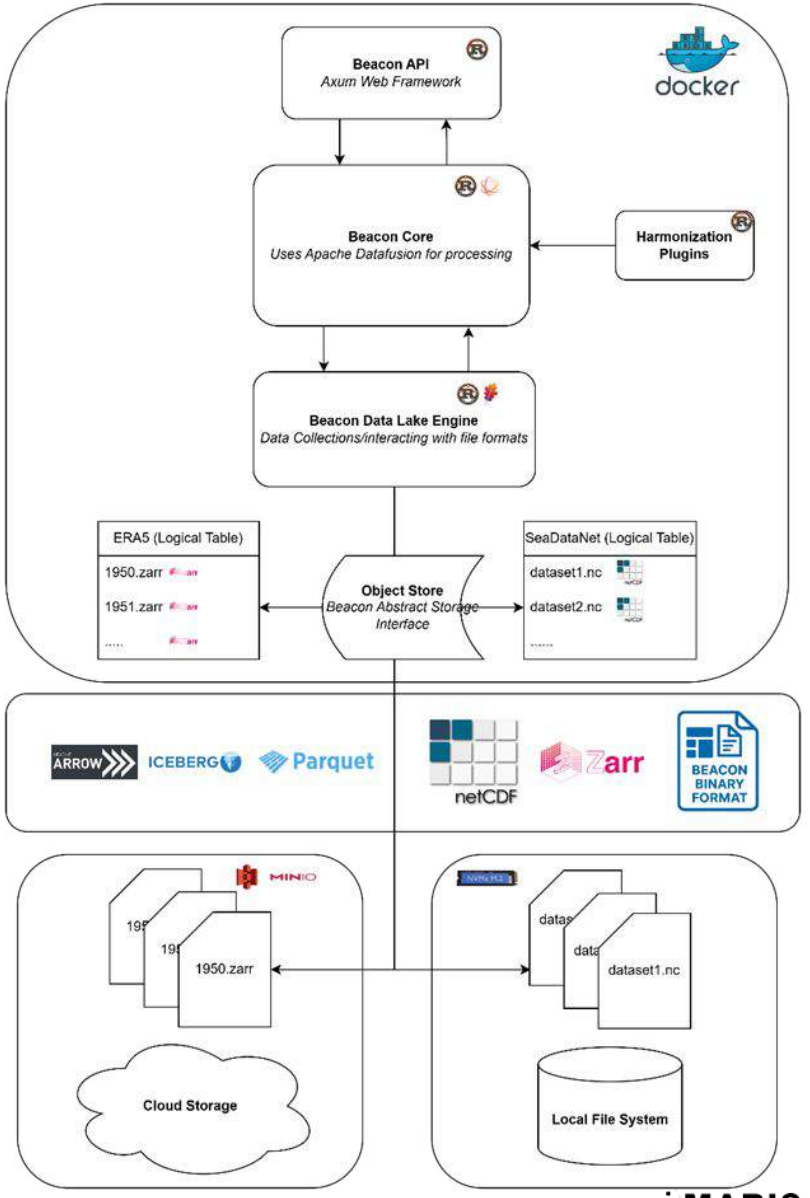


Data lake and cloud support

- **Beacon can create a data lake:**
  - **Directly on a given folder in of your local filesystem (FS)**
    - Eg. all the files in your ‘./data/datasets’
  - **Existing S3 Bucket (using the S3 protocol)**
    - Supports AWS, R2, MinIO, etc.
    - All the files within that bucket will be queryable via Beacon
- **All data lake management operations supported for both local FS and S3**
  - Uploading files
  - Deleting files
  - Downloading files
- **S3 Settings can be changed using Environment Variables**

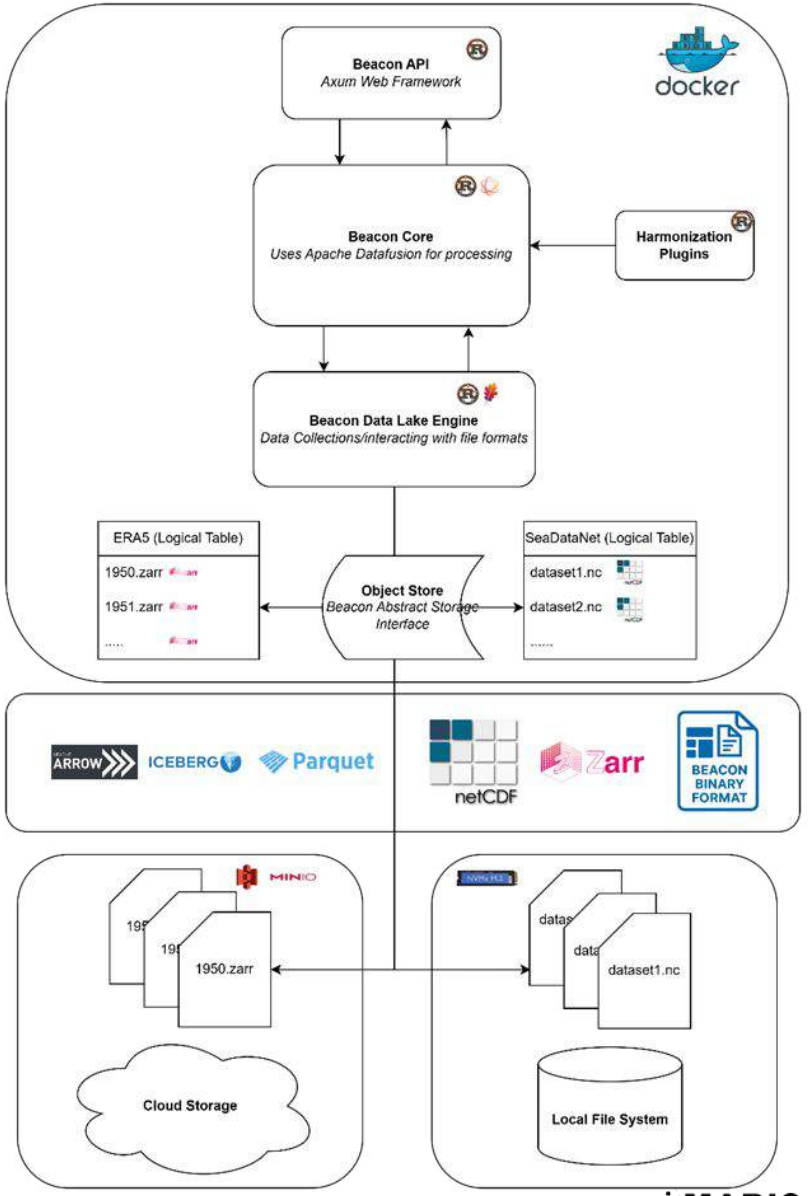
```
- BEACON_S3_DATA_LAKE=true
- BEACON_S3_ENDPOINT=http://minio:9000
- BEACON_S3_BUCKET=beacon-bucket
- BEACON_S3_ACCESS_KEY_ID=minioadmin
- BEACON_S3_SECRET_ACCESS_KEY=minioadmin
```

- **Also supports running Beacon on top of S3 buckets without write access (anonymous)**



Supporting many file types

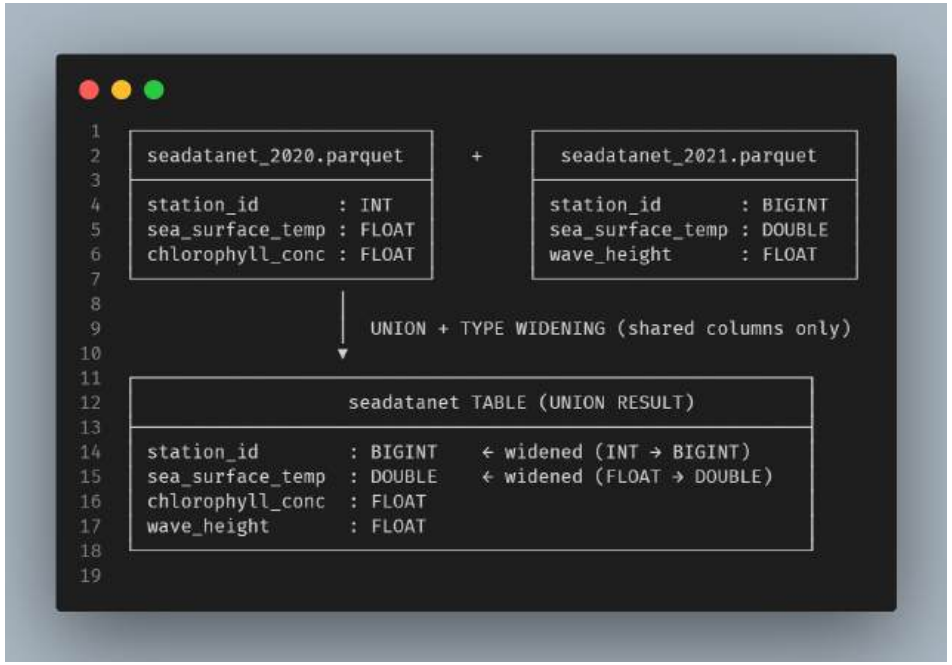
- **Beacon can read from an existing file or multiple at the same**
  - Eg. read column *a,b,c* from example1.parquet
  - Eg. read columns *x,y,z* from *.parquet
- **Supported file formats:**
  - CSV
  - Zarr (any structure, timeseries, gridded, cruises & supports parallel chunked reading)
    - Reads only the relevant chunks
      - E.g. given *lon*, *lat* as statistics options when reading means it will pushdown filters to only read relevant chunks based on filters given for the columns *lon*, *lat*
  - Parquet
  - Apache Iceberg (integrated via parquet)
  - Delta Table (integrated via parquet)
  - Arrow IPC
  - NetCDF (any structure, timeseries, gridded, cruises & supports chunked reading)
  - Beacon Binary Format (also ARCO)
  - Icechunk (**Work in Progress)



Poll #1

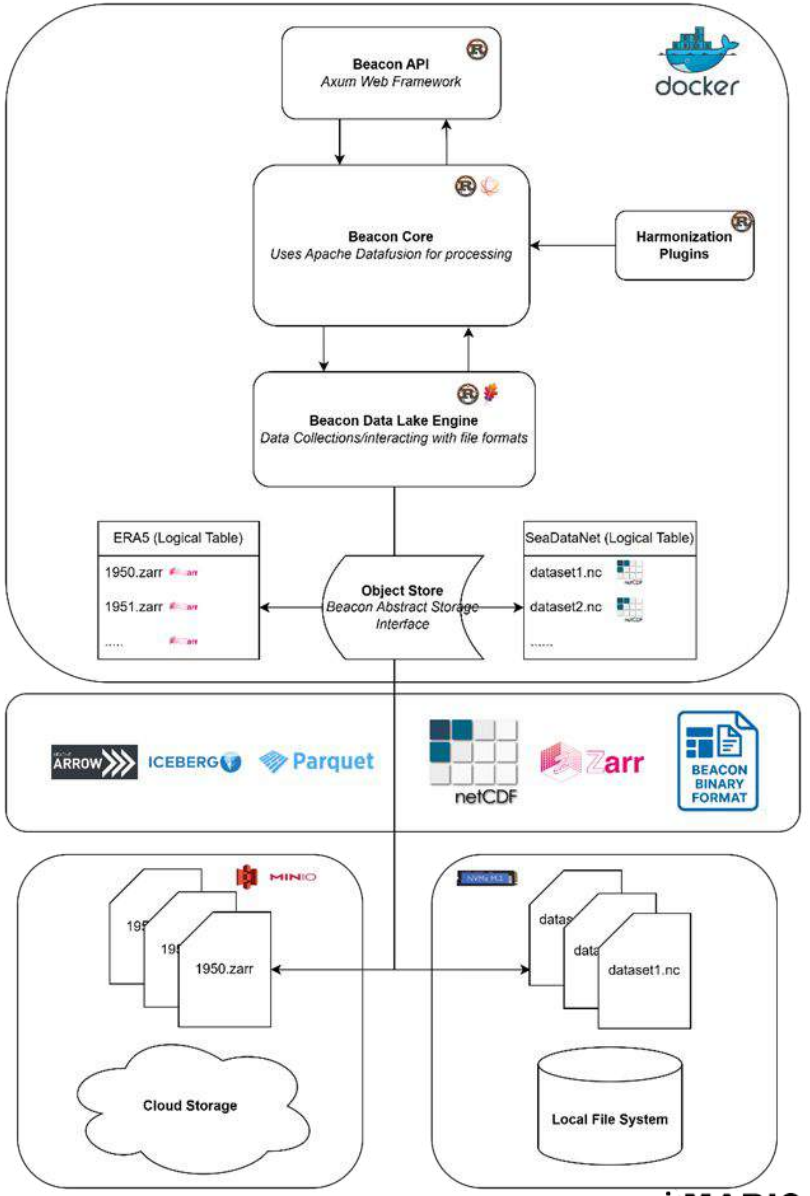
Schema and data type support

- **Reading files with different schemas (columns/datatypes)**
  - Harmonizes the schema
    - missing columns are filled with NULL
    - columns with different data types are widened to their supertype
      - (eg. int32 + float32 -> float64)
  - This happens during planning so almost no performance impact
  - Can be overridden with explicit cast in a query
- Most important data types are supported
  - bool
  - int8,int16,int32,int64
  - uint8,uint16,uint32,uint64
  - float32,float64
  - timestamps
  - strings (fixed-size + variable-sized)
  - binary (fixed-size + variable-sized)
  - *geo-types (point, polygon)
    - is still work in progress but basic support (eg. st_makepoint)
    - improving them as we go
    - compatibility with postgis but more performance



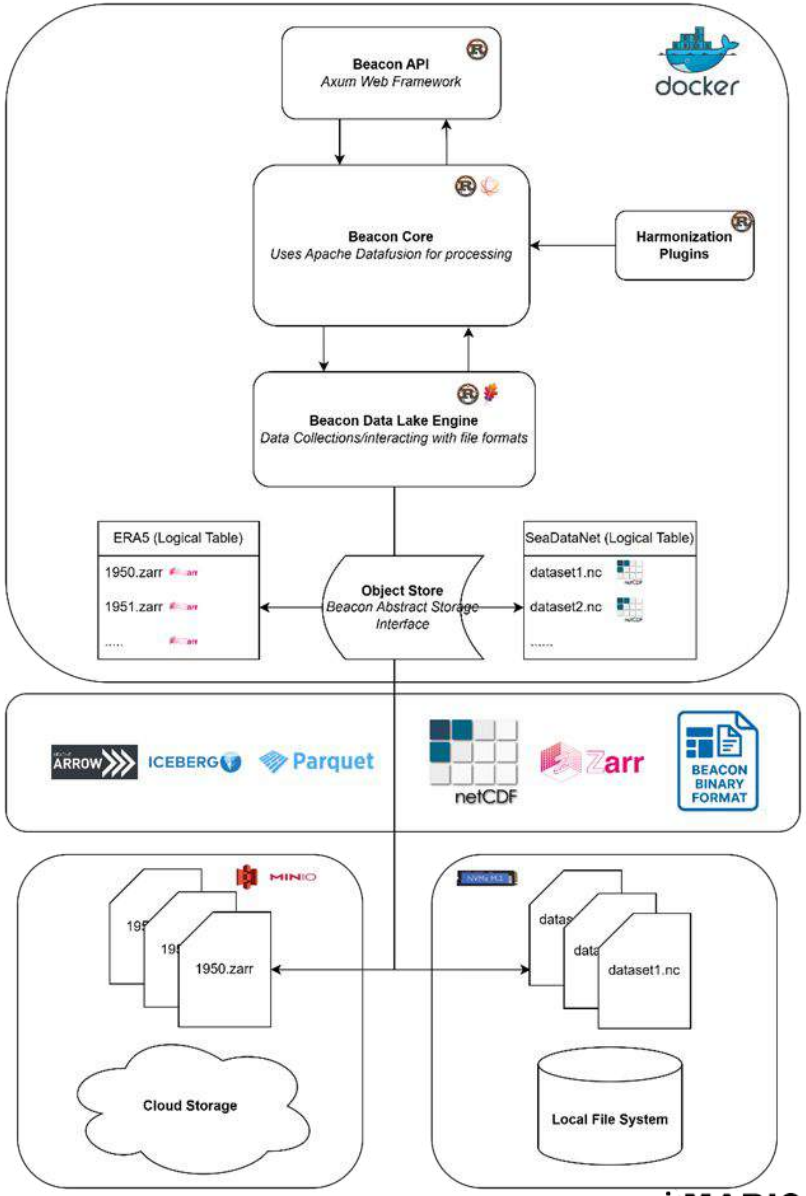
Virtual collections

- **Create virtual collections (logical tables)**
  - Eg. expose *'file1.parquet', 'some_dir/file2.parquet'* as a collection named: *'example'*
  - Eg. expose multiple files using glob '*.parquet' or 'some_dir/*.parquet' as a collection
  - Works for any file format (zarr, parquet etc…)
  - Also harmonizes the schema using the rules from the previous slide
- **Create preset collection on top off virtual collections**
  - It is possible to stack tables on top of each other, adding more and more functionality.
  - Allows providers to set:
    - predefined columns
    - predefined filters
    - predefined drop down options
    - renaming columns (no performance impact)
    - overriding/changing data types



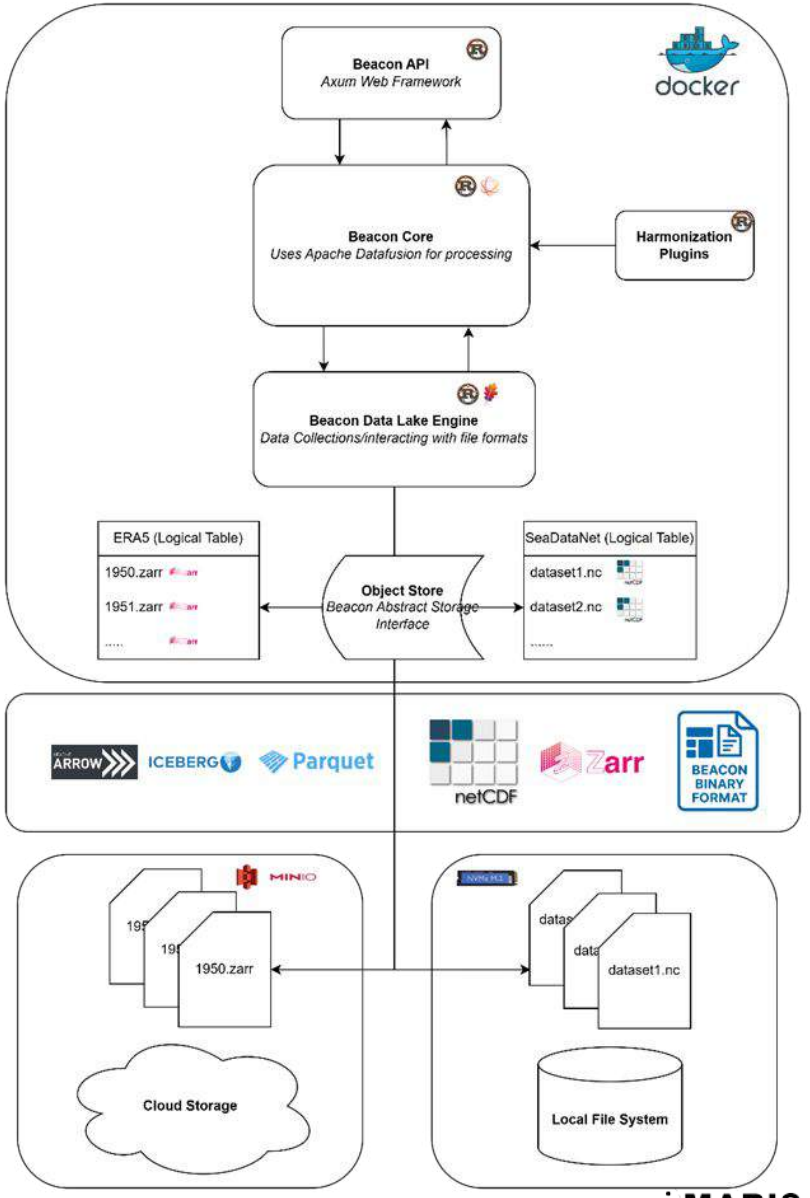
Querying

- **Querying Data**
  - JSON Query
    - Select columns
    - Add filters on columns
      - Equals/ not equals
      - Range filters (*gt/gteq/lt/lteq*)
      - Null/not null filters
      - Polygon filters
    - Sort
    - Distinct
    - Functions
  - SQL Query (optional via environment variable)
    - No DML statements allowed (read-only)
    - Full SQL Support
- **Query Output:**
  - Streaming (arrow ipc)
  - Arrow ipc file
  - Parquet (GeoParquet)
  - CSV
  - NetCDF (optional n-dimensional)
  - ODV Ascii



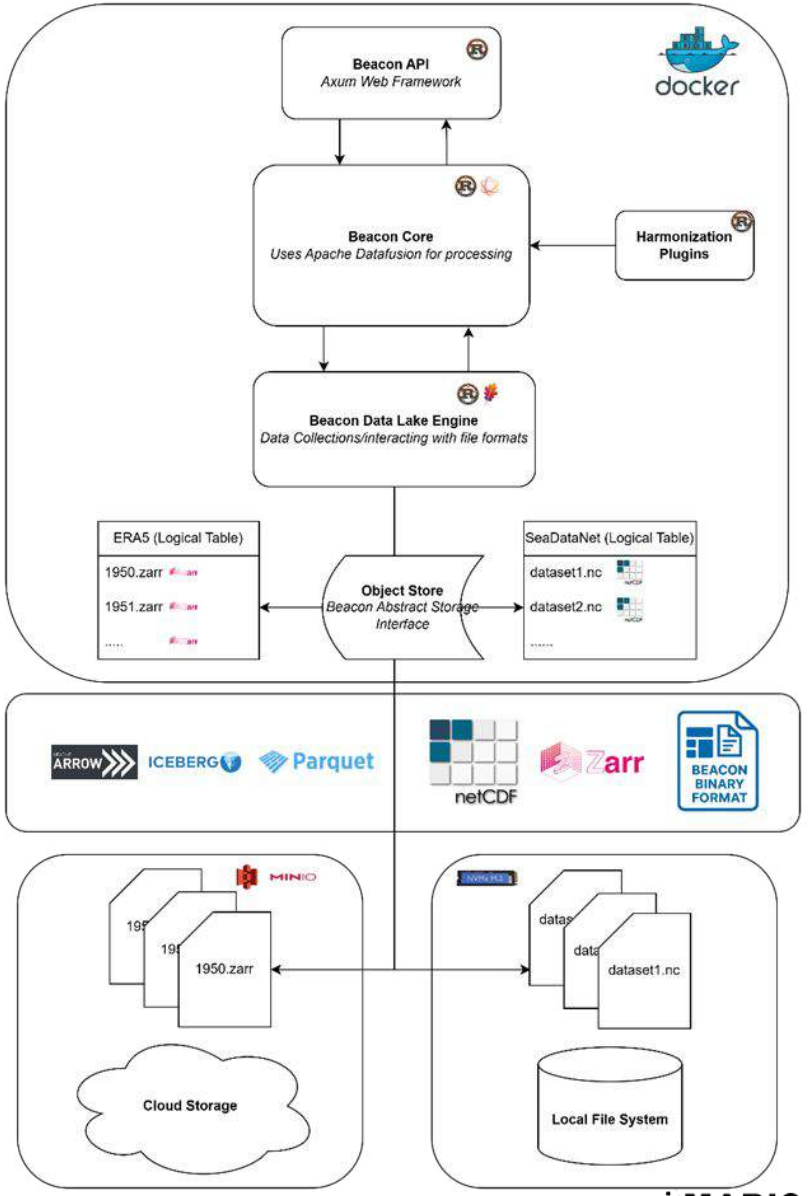
High performance

- **Excellent out-of-the box performance**
  - Apache Arrow
  - Datafusion (one of highest performing OLAP libraries https://benchmark.clickhouse.com)
  - Smart indexes (based on which file format)
    - all indexes are on disk (not in memory) or build during planning
    - pushdown filters efficiently for most file formats
  - Only reading relevant columns
    - using pushdown projections
  - Full Streaming Engine
    - Can stream data back to the client over http while the query is running using arrow ipc
  - Runs fully parallel utilizing all computational cores
    - Also when reading from S3

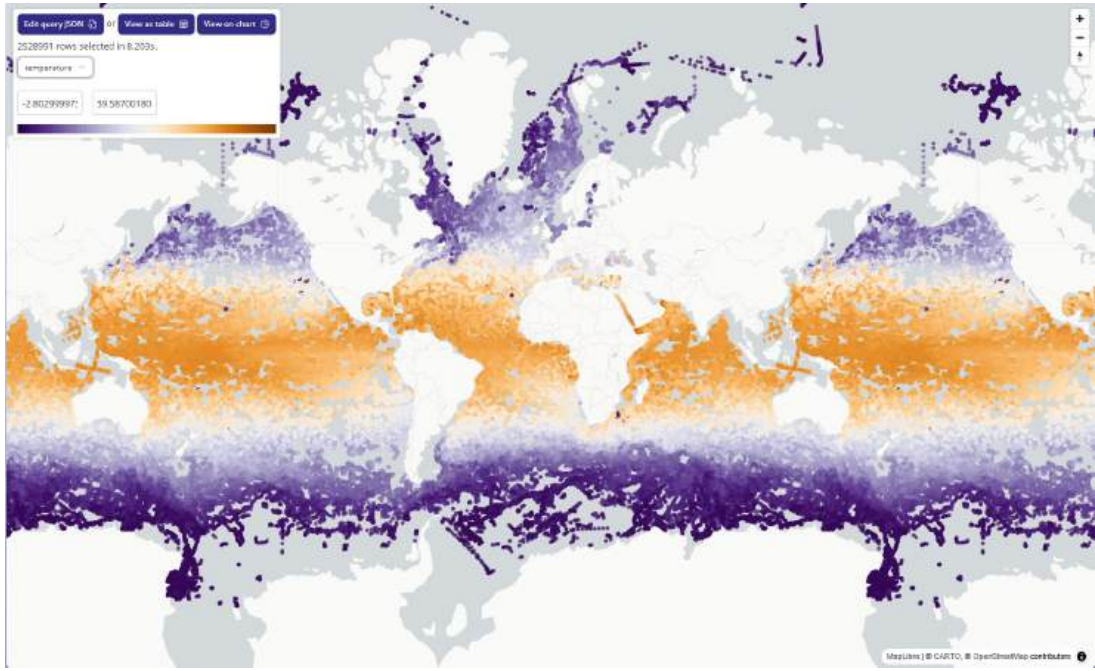


Beacon Binary Format (BBF)

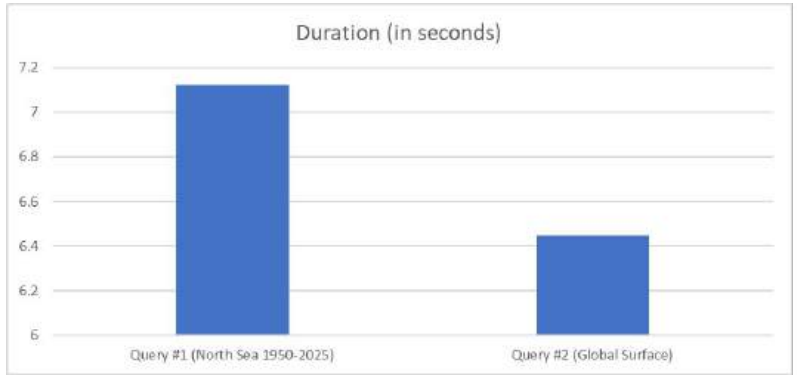
- **High Performance Binary Data Format**
- **Store many datasets (netcdf,csv,parquet) into a single bbf file.**
  - example.bbf
    - dataset1.nc
    - dataset2.nc
    - dataset3.csv
    - dataset4.parquet
- **Creates a custom binary encoded version of the dataset**
  - Data is stored in n-dimensional columns using a Arrow Array Layout
  - Columns are stored inline making compression much more effective
  - A pruning index is created on each column
- **Also supports storing datasets in chunks within a BBF file**
- **In simple terms:**
  - A ZIP file containing many different datasets with an index on top
- **Reading is done using a custom parallel async scheduler**
  - No blocking
  - Data is retrieved in blocks (by default 1MB)
  - Threads will schedule efficiently when both read the same block
    - Only 1 thread will retrieve the block from disk/S3, the scheduler will pre-fetch the next block already

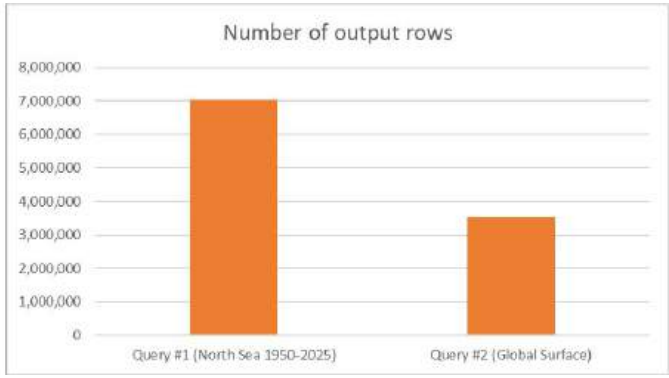


Beacon in-situ Performance 🚀

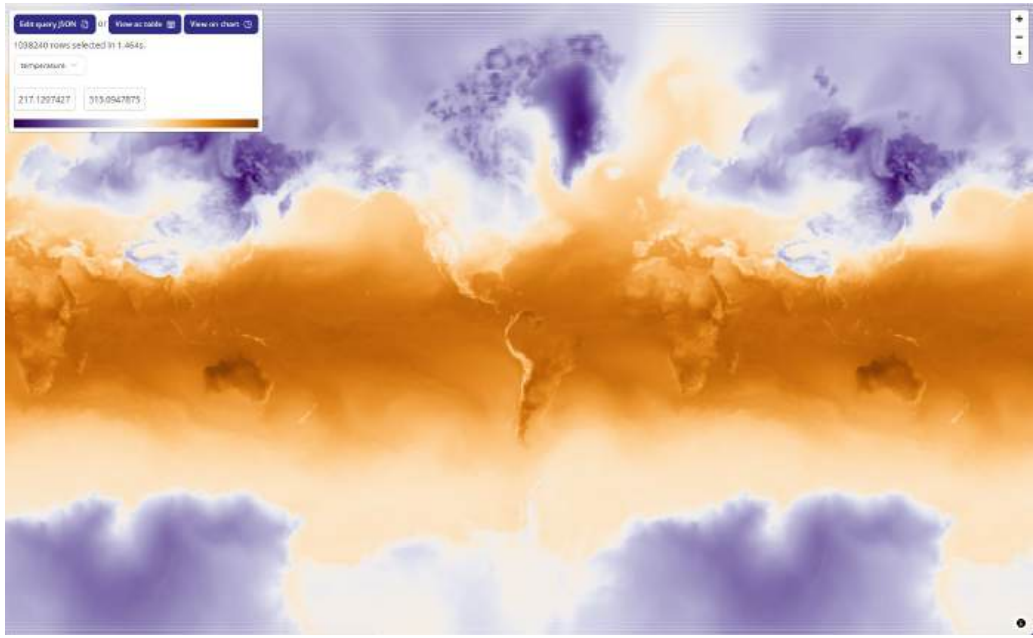


- **World Ocean Database**
  - ~20 Million individual NetCDF Files stored as multiple BBF files
  - ~600GB of data, compressed to 50GB BBF files inside a Cloud Bucket
- **Benchmark Queries**
  - Selected Columns:
    - Temperature + QC
    - Longitude
    - Latitude
    - Depth
    - Time
    - Platform
    - Institute
  - Query #1 Range:
    - North Sea
    - 1950 - 2025
  - Query #2 Range:
    - Global Surface Level (0-10m)
    - 1 entire year

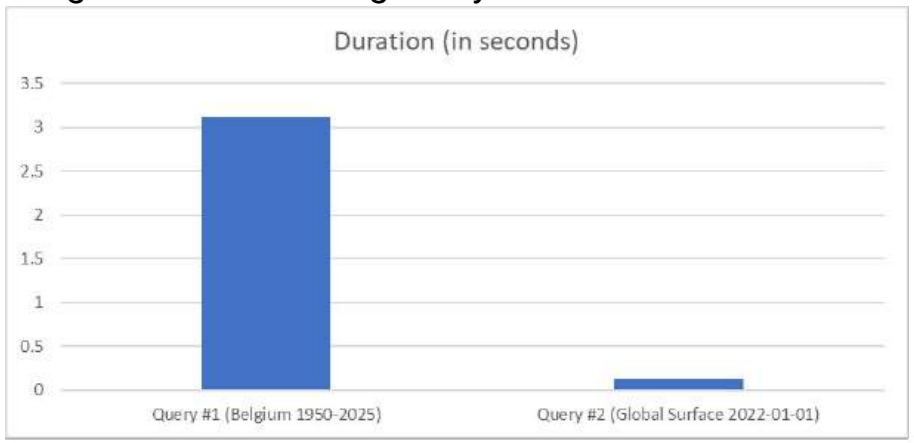


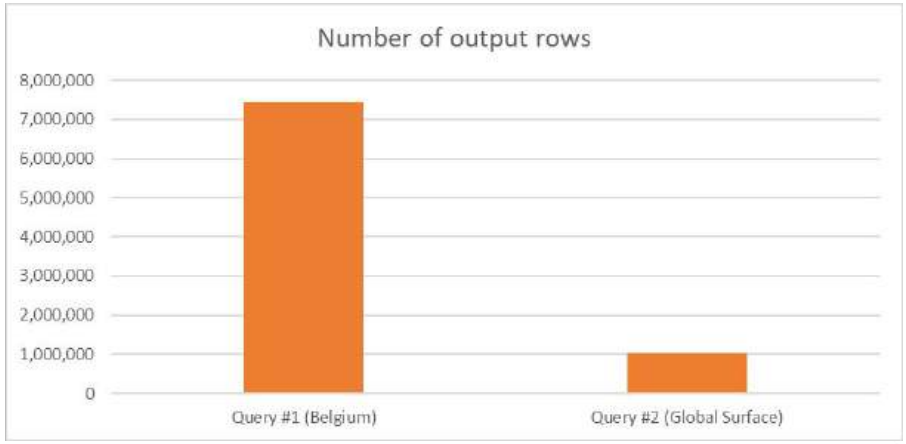


Beacon 🤝 Zarr Performance ⚡



- **ERA5 Daily Reanalysis**
  - Stored as ZARR files inside a Cloud Bucket
  - Multiple TB's (Global, 1950 - 2025)
- **Benchmark Queries**
  - Selected Columns:
    - Temperature
    - Longitude
    - Latitude
    - Time
  - Query #1 Range Belgium, 1950 - 2025
  - Query #2 Range Global for a Single Day





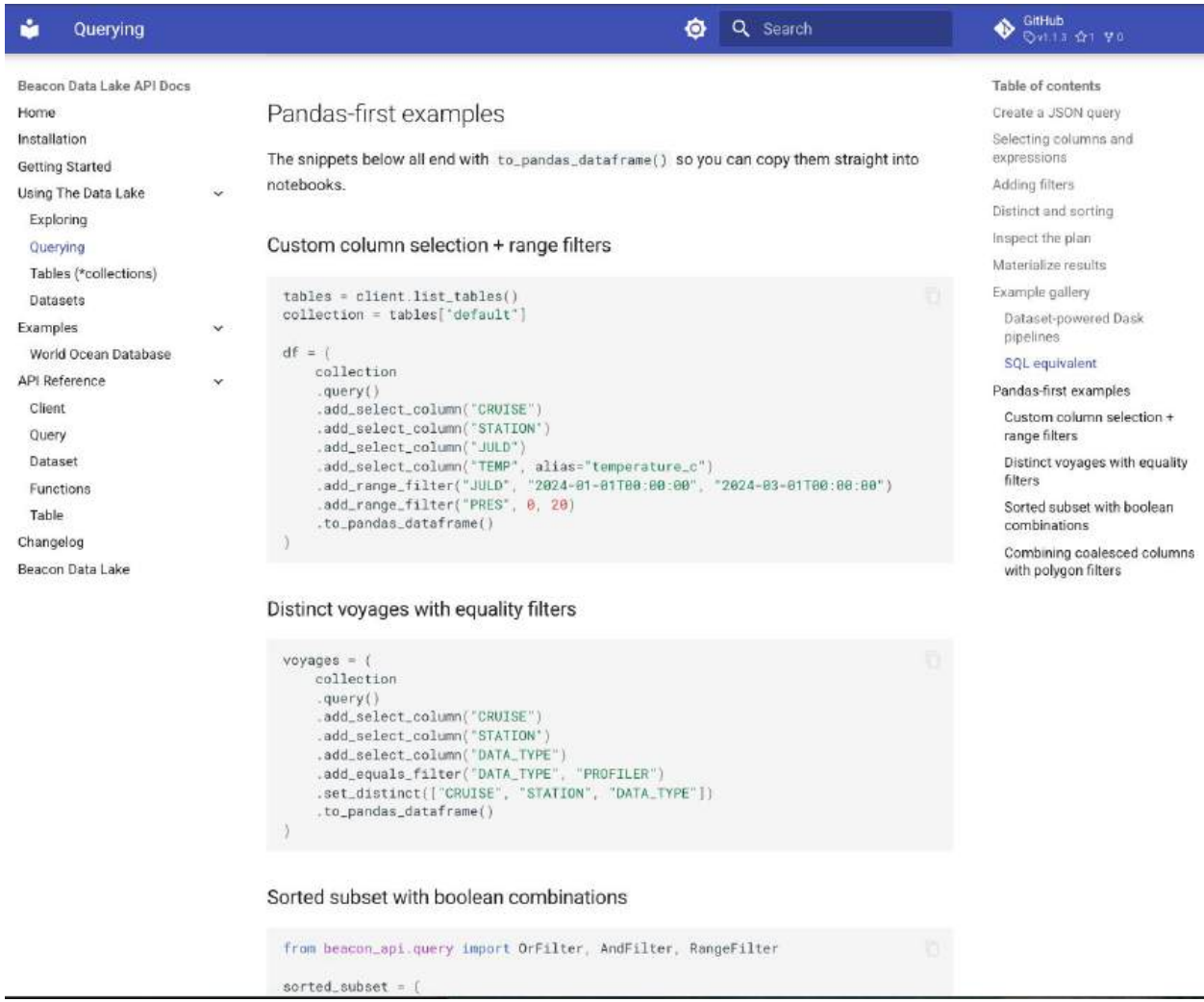
DIGISHAPE DATASCIENCE FOR DELTAS

10 December 2025

MARIS

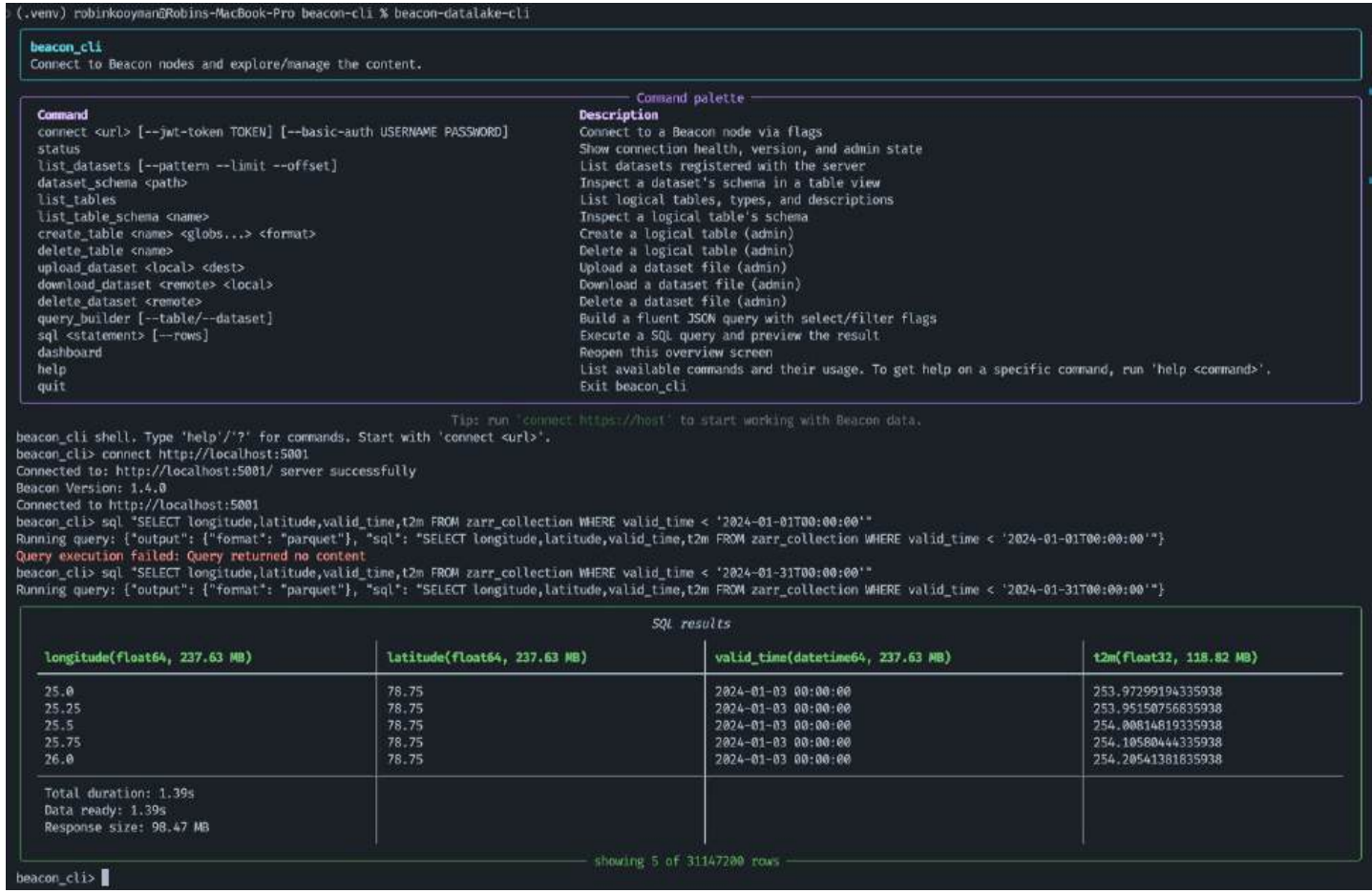
Beacon Tooling

- **Beacon Python SDK**
  - pip install beacon-api
  - Python Package
    - https://pypi.org/project/beacon-api/
  - GitHub:
    - https://github.com/maris-development/beacon-py
  - Documentation:
    - https://maris-development.github.io/beacon-py/latest/
  - Works with all beacon nodes
  - Export directly to:
    - Streaming (pollable through iterator of arrow batches)
    - Pandas
    - GeoPandas
    - Xarray (n-dimensional optional)
    - Parquet
    - GeoParquet
    - NetCDF
    - CSV
    - Zarr



Beacon Tooling

- **Beacon Data Lake CLI (WIP)**
  - Build on top of the Beacon Python SDK
  - pip install beacon-data-lake-cli
  - Python Package
    - https://pypi.org/project/beacon-data-lake-cli/
  - Works with all beacon nodes
  - Easy management
    - uploading files
    - downloading raw files
    - deleting files
    - creating collections
    - querying using SQL
    - preview results
    - list datasets
    - etc…
  - Querying
    - export results to local file
      - parquet, csv, arrow, etc.



Poll #2

Poll #3

Questions

Beacon Provider

DIGISHAPE DATASCIENCE FOR DELTAS

10 December 2025

MARIS

Beacon Provider

- Setting up a Beacon Instance (2 minutes)
  - git clone https://github.com/maris-development/beacon-example.git
  - Pick either and navigate to that folder:
    - MinIO S3 example
    - Local file system example
  - run the command: docker compose up
  - navigate to: http://localhost:8080 and the swagger page will appear
  - done, start uploading/moving files to your data directory and start querying immediately

Beacon Provider

```
C:\Users\robin\Documents\demo>git clone https://github.com/maris-development/beacon-example.git
Cloning into 'beacon-example'...
remote: Enumerating objects: 35, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 35 (delta 8), reused 28 (delta 5), pack-reused 0 (from 0)
Receiving objects: 100% (35/35), 98.11 KiB | 16.35 MiB/s, done.
Resolving deltas: 100% (8/8), done.

C:\Users\robin\Documents\demo>cd beacon-example

C:\Users\robin\Documents\demo\beacon-example>cd cloud-storage-minio

C:\Users\robin\Documents\demo\beacon-example\cloud-storage-minio>docker compose up
time="2025-12-09T14:49:41+01:00" level=warning msg="C:\\Users\\robin\\Documents\\demo\\beacon-example\\cloud-storage-minio\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid
 potential confusion"
[+] Running 4/4
  ✔Network cloud-storage-minio_default           Created                                                                                                                                                0.1s
  ✔Container minio                               Created                                                                                                                                                0.1s
  ✔Container cloud-storage-minio-createbuckets-1 Created                                                                                                                                                0.1s
  ✔Container cloud-storage-minio-beacon-1        Created                                                                                                                                                0.1s
Attaching to beacon-1, createbuckets-1, minio
minio  | INFO: Formatting 1st pool, 1 set(s), 1 drives per set.
minio  | INFO: WARNING: Host local has more than 0 drives of set. A host failure will result in data becoming unavailable.
minio  | MinIO Object Storage Server
minio  | Copyright: 2015-2025 MinIO, Inc.
minio  | License: GNU AGPLv3 - https://www.gnu.org/licenses/agpl-3.0.html
minio  | Version: RELEASE.2025-09-07T16-13-09Z (go1.24.6 linux/amd64)
minio  |
minio  | API: http://172.18.0.2:9000  http://127.0.0.1:9000
minio  | WebUI: http://172.18.0.2:9001 http://127.0.0.1:9001
minio  |
minio  | Docs: https://docs.min.io
minio  | WARN: Detected default credentials 'minioadmin:minioadmin', we recommend that you change these values with 'MINIO_ROOT_USER' and 'MINIO_ROOT_PASSWORD' environment variables
createbuckets-1  | Added `local` successfully.
createbuckets-1  | Bucket created successfully `local/beacon-bucket`.
createbuckets-1  | Bucket beacon-bucket ready.
createbuckets-1 exited with code 0
beacon-1         | 2025-12-09T13:49:48.941509Z  INFO beacon_api: Beacon API v1.4.1
beacon-1         | 2025-12-09T13:49:48.975764Z  INFO beacon_data_lake: Configuring S3 object store for datasets
beacon-1         | 2025-12-09T13:49:48.976780Z  INFO object_store::aws::builder: Using Static credential provider
beacon-1         | 2025-12-09T13:49:49.059557Z DEBUG beacon_data_lake: Using local NetCDF sink path: /beacon/data
beacon-1         | 2025-12-09T13:49:49.062251Z  INFO beacon_data_lake: Initializing tables from object store
beacon-1         | 2025-12-09T13:49:49.068269Z  INFO beacon_data_lake: Found table entry: Ok(ObjectMeta { location: Path { raw: "tables/default/table.json" }, last_modified: 2025-12-09T13:49:03.660954900Z, size: 111, e_tag: So
me("4000000006362-645852cbaa79a-6f"), version: None })
beacon-1         | 2025-12-09T13:49:49.085018Z DEBUG beacon_data_lake: Using local NetCDF sink path: /beacon/data
beacon-1         | 2025-12-09T13:49:49.089707Z  INFO beacon_api: listening on 0.0.0.0:8080
```

Demo

Questions

Beacon User

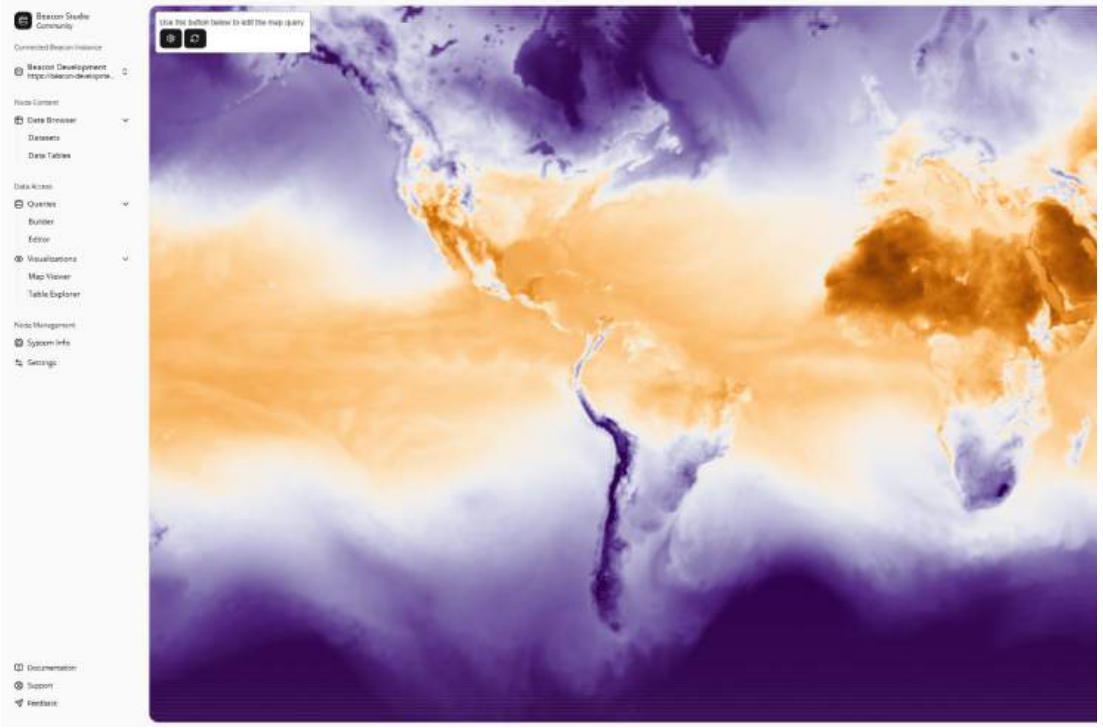
DIGISHAPE DATASCIENCE FOR DELTAS

10 December 2025

MARIS

Beacon studio

GUI with implementation examples



Beacon notebooks

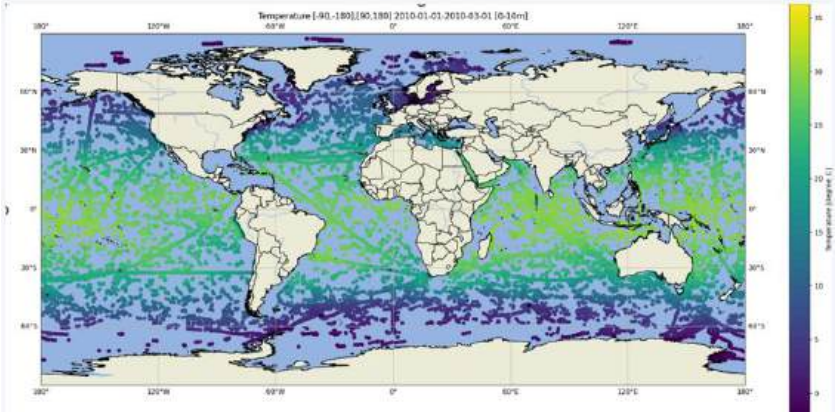
Implementation examples

```
response = requests.post("https://beacon-wod.maris.nl/api/query", json.dumps(query_body), headers={
    "Authorization": f"Bearer {Token}",
    "Content-type": "application/json"
})
```

datetime	Temperature	Unit	DEPTH	LONGITUDE	LATITUDE	DATASET	cruise-identifier	cast	WMO_ID	country	dataset_id
2010-01-01	24.700001	degree_C	1.0	-95.000000	-5.000000	moored buoy	US015955	12327900	2.115095e+09	UNITED STATES	11382402
2010-01-01	26.120001	degree_C	5.0	-110.000000	0.000000	moored buoy	US015969	15272646	2.115095e+09	UNITED STATES	11382406
2010-01-01	26.110001	degree_C	10.0	-110.000000	0.000000	moored buoy	US015969	15272646	2.115095e+09	UNITED STATES	11382406
2010-01-01	28.959999	degree_C	1.5	147.039993	4.960000	moored buoy	JP020811	12327885	2.115114e+09	JAPAN	11382401
2010-01-01	25.440001	degree_C	1.0	-37.880001	20.040001	moored buoy	US026451	12327924	2.115104e+09	UNITED STATES	11382426
...	...	...	...	...	...	...	...	...	...	...	...
2010-03-01	29.379999	degree_C	1.0	-34.639999	-18.860001	moored buoy	BR001204	12332326	2.115093e+09	BRAZIL	11390007
2010-03-01	27.370001	degree_C	1.0	-155.009995	7.980000	moored buoy	US015934	12332305	2.115114e+09	UNITED STATES	11389980
2010-03-01	29.860001	degree_C	1.0	-170.020004	-2.170000	moored buoy	US015927	12332281	2.115114e+09	UNITED STATES	11389956
2010-03-01	29.540001	degree_C	1.0	89.989998	-1.670000	moored buoy	JP032626	12327389	2.115115e+09	JAPAN	11389989
2010-03-01	28.430000	degree_C	1.0	165.130005	8.050000	moored buoy	US015890	12332307	2.115114e+09	UNITED STATES	11389966

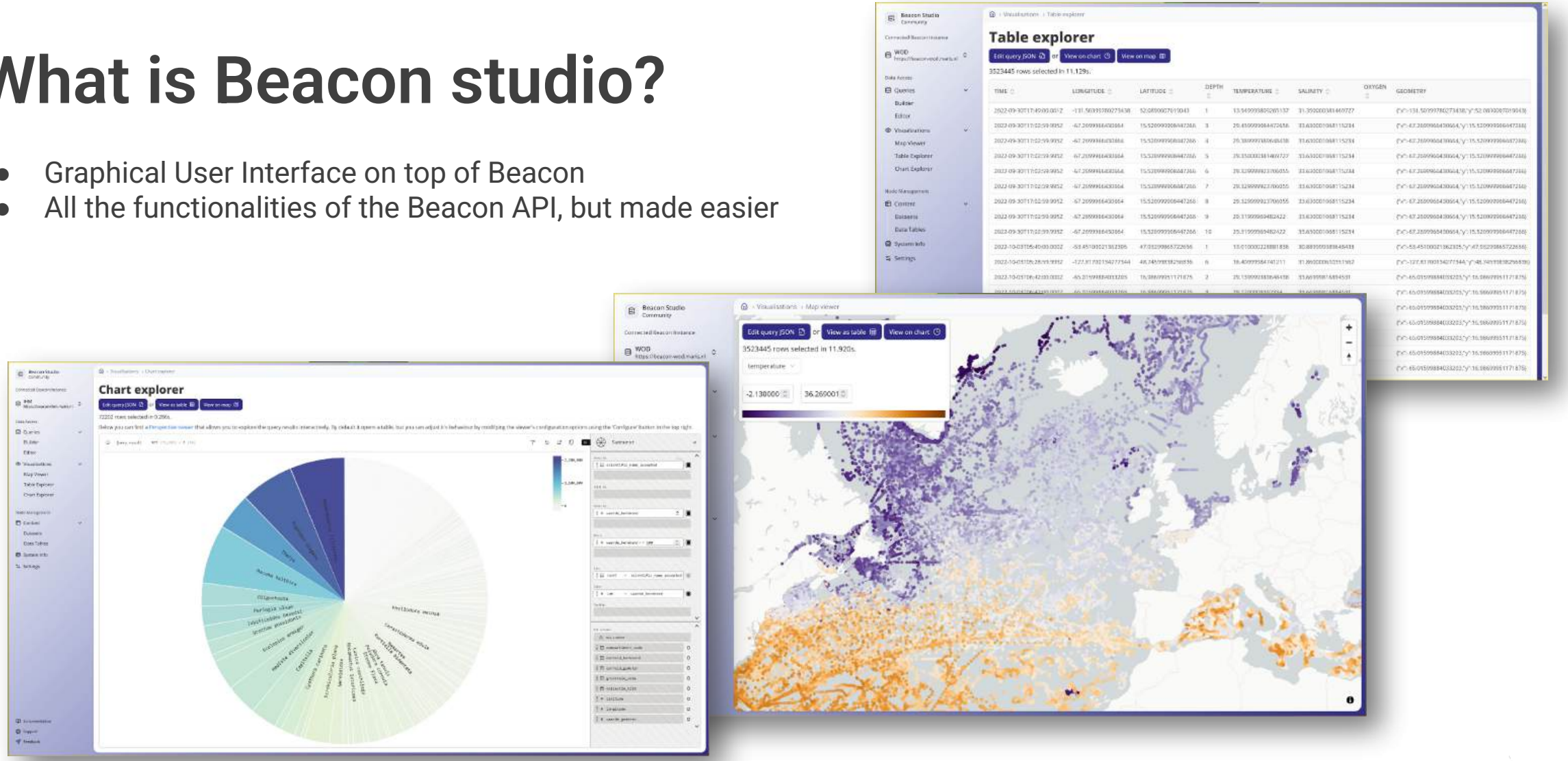
30863 rows × 11 columns

In this example we can then very quickly plot this subset of the temperature data between 0-10 meters depth from the WOD collection of the first months of 2010 produced.



What is Beacon studio?

- Graphical User Interface on top of Beacon
- All the functionalities of the Beacon API, but made easier

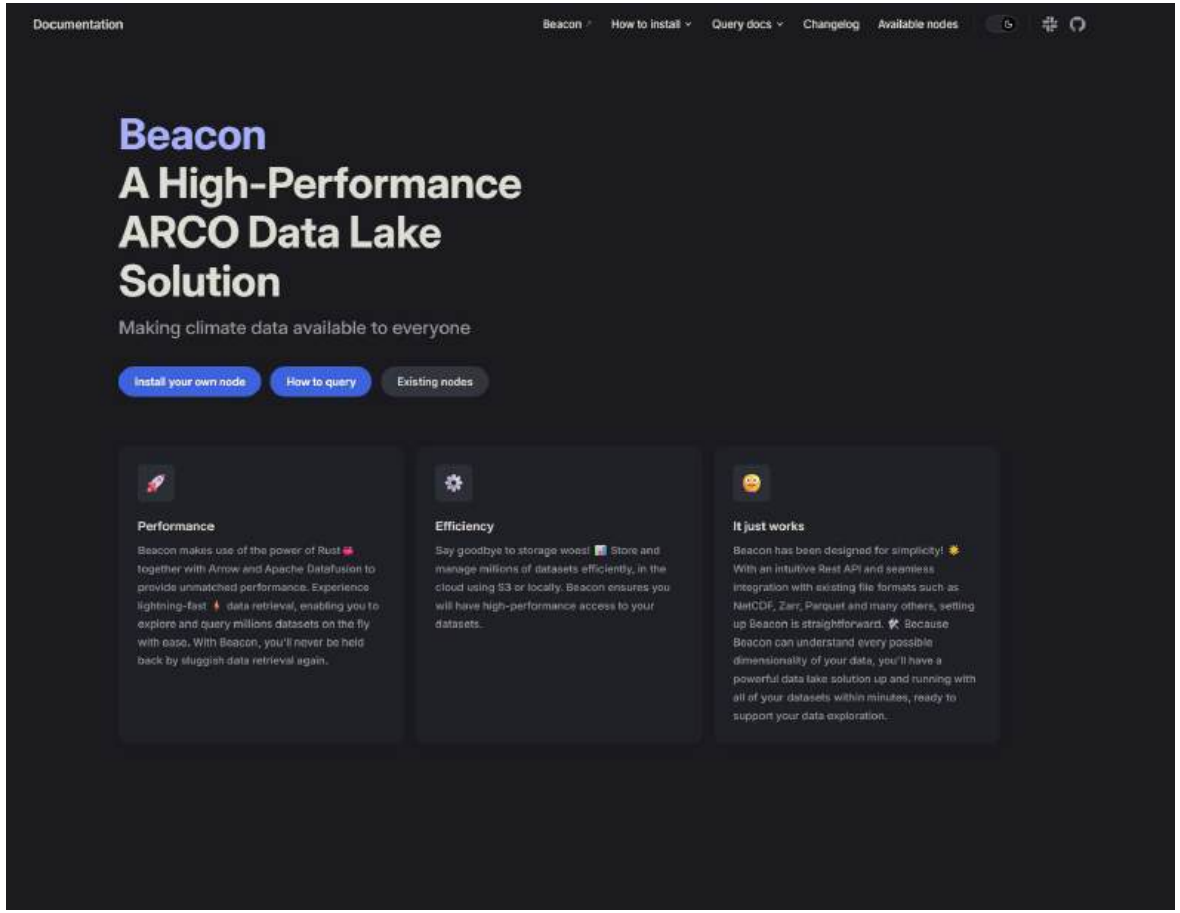


Demo

Closure

STATUS

- Beacon freely available under Open Source license at:
  - https://beacon.maris.nl/
- Docs:
  - https://maris-development.github.io/beacon/
- GitHub:
  - https://github.com/maris-development/beacon
- Studio (user interface for IHM POC):
  - https://beacon-ihm.maris.nl/studio
- Python library:
  - https://pypi.org/project/beacon-api/



Jullie ontvangen na afloop een feedback formulier, inclusief een vraag of verder contact gewenst is.

Contact:

peter@maris.nl
robin@maris.nl
tjerk@maris.nl

Questions
<15 mins>