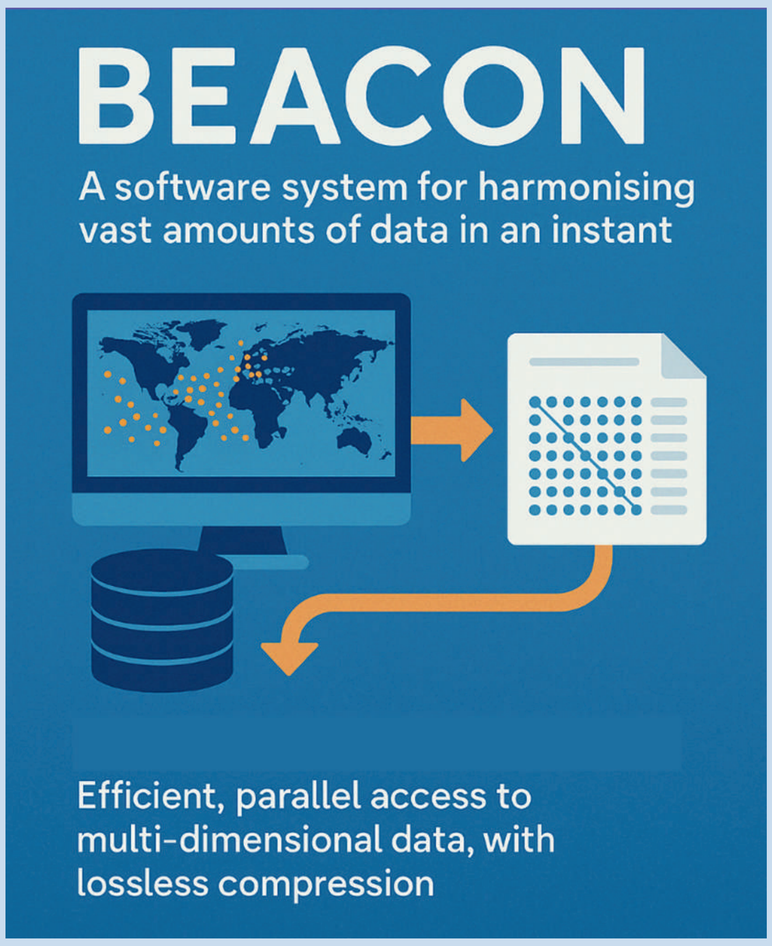
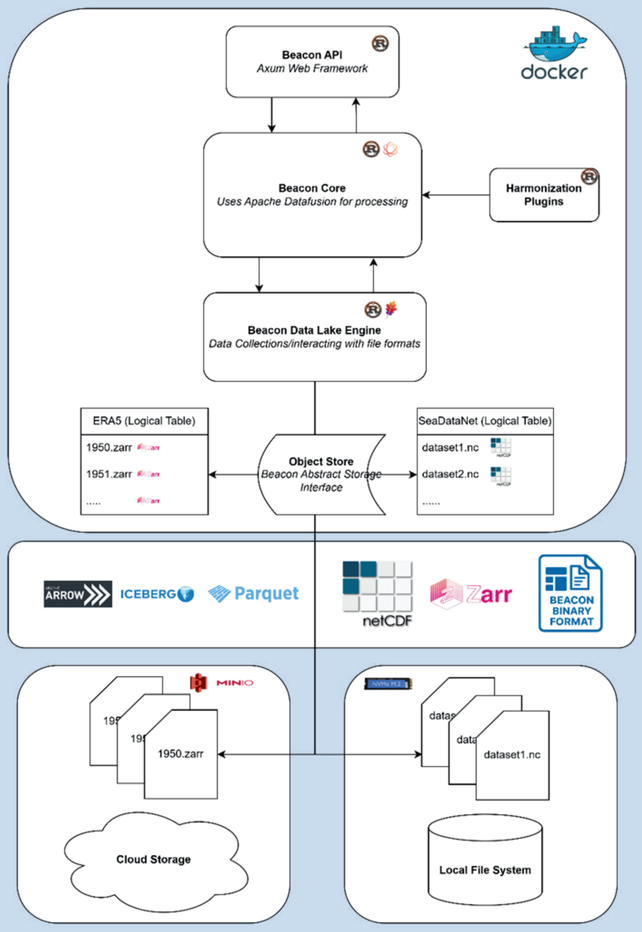
Beacon: A FAIR high-performance, ARCO data lake technology supporting interoperable environmental research

Sharing is encouraged

MARIS





As part of several major European research initiatives, Blue-Cloud2026, EOSC-FUTURE, and ENVRI-Hub NEXT, MARIS has developed a powerful software system called Beacon. Designed to operate within the European Open Science Cloud (EOSC) ecosystem, Beacon enables fast and dynamic access to large-scale, heterogeneous environmental and oceanographic datasets.

At its core, Beacon uses a unique indexing system that allows it to dynamically extract subsets of data on-the-fly based on user queries. This means that users can request a specific subset from millions of files containing different types and structures of observational data, such as gridded datasets, timeseries, or cruise observations and Beacon will return a single, harmonised output file containing just the requested information.

Because the data sources used by Beacon (such as Euro-Argo, SeaDataNet, ERA5, and World Ocean Database) often consist of millions of individual files, two significant challenges arise:

1. Ensuring fast query performance
2. Managing extremely large storage requirements

To address these, MARIS introduced Beacon ATLAS, a purpose-built, high-performance binary format that significantly improves data storage and retrieval. ATLAS was designed with the goal of:

- Reducing the physical storage footprint of large datasets
- Maximizing read and query performance
- Preserving full data fidelity, ensuring that all original information is retained

ATLAS achieves these goals through several key innovations:

- It stores multi-dimensional data as Apache Arrow arrays, which allows for zero-deserialization access. In practice, this means that the data can be read directly from disk as if it were in computer memory, eliminating the usual performance hit from format translation.
- It is non-blocking, meaning multiple processor cores can access and read from the file simultaneously. This allows for true parallel access to millions of datasets and enables data transfer speeds of multiple gigabytes per second, often shifting the performance bottleneck from software to hardware.
- It uses adaptive, block-level compression, meaning that each dataset is analyzed and compressed using the most efficient method for that specific data. This improves both storage efficiency and decompression speed, in some cases reaching speeds close to a machine's memory bandwidth.
- Despite being highly optimized, ATLAS maintains lossless integrity: all original data values and structures are retained. An ATLAS file created from a NetCDF file, for example, can be used to fully reconstruct the original NetCDF.
- ATLAS uses a pruning block based indexing system that allows you to push filters to the scan level and skip datasets that don't contain any data relevant to your filters.

This new format is benchmarked and compared with traditional data formats like NetCDF, CSV, and ASCII in terms of performance and storage efficiency, and consistently shows substantial advantages similar to parquet (Analysis Ready Cloud Optimized - ARCO). In January 2025, the Beacon software system was officially released as open-source software (version 1.0.0), making it available to the wider scientific and data management community. This allows any organization to deploy their own Beacon instance, improve access to their data, and reduce storage requirements, without compromising on data quality or completeness.

Beacon nodes are easily accessible via a Jupyter Notebook

In order to request data from the WOD Beacon endpoint, the query body needs to be constructed. In the image below you can find an example of how such a query can look. In this case we are querying temperature (parameter), its units, time, depth, longitude and latitude, while filtering on time and depth.

```
query_builder = tables['default'].query()

query_builder.add_select_column("time", alias="TIME")
query_builder.add_select_column("lon", alias="LONGITUDE")
query_builder.add_select_column("lat", alias="LATITUDE")
query_builder.add_select_column("Temperature", alias="TEMPERATURE")
query_builder.add_select_column("Temperature_WODflag", alias="TEMPERATURE_QC")
query_builder.add_select_column("Temperature.units", alias="TEMPERATURE_UNIT")
query_builder.add_select_column("z", alias="DEPTH")
query_builder.add_select_column("z.units", alias="DEPTH_UNIT")
query_builder.add_select_column(".featureType", alias="FEATURE_TYPE")
query_builder.add_range_filter("TIME", "2022-01-01T00:00:00", "2023-01-01T00:00:00")
query_builder.add_is_not_null_filter("TEMPERATURE")
query_builder.add_equals_filter("TEMPERATURE_QC", 0.0)
query_builder.add_range_filter("DEPTH", 0, 10)

df = query_builder.to_pandas_dataframe()
df
```

✓ 8.4s

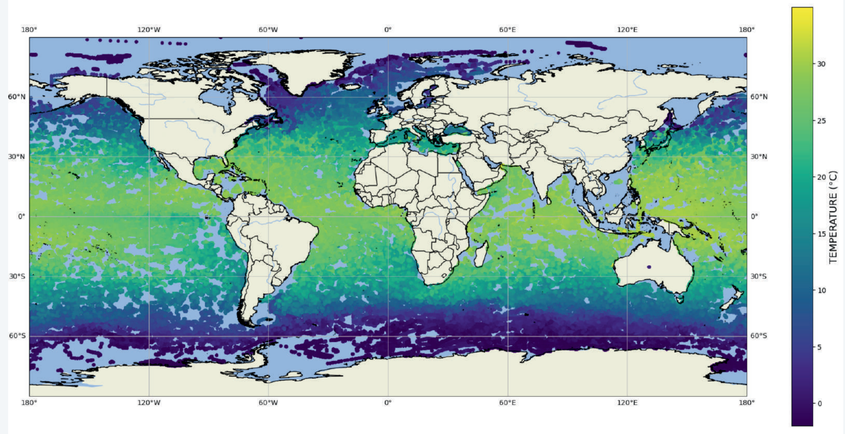
A user can then send this query body to the Beacon endpoint, as seen in the image below. This provides us with the requested data in a dataframe containing all the data fitting the provided filters above in only 8 seconds! Extra metadata columns can be added to the dataframe by extending the query body above with extra elements.

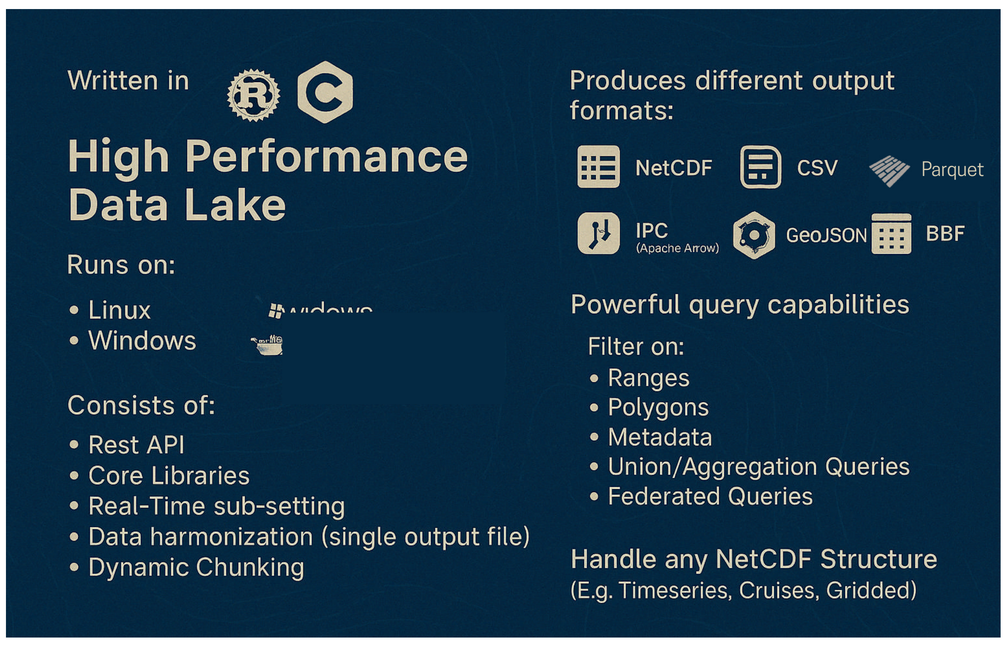
```
Creating JSONQuery with from: FromTable(table='default')
Running query: {"output": {"format": "parquet"}, "select": [{"column": "time", "alias": "TIME"}, {"column": "lon", "alias": "LONGITUDE"}, {"column
```

	TIME	LONGITUDE	LATITUDE	TEMPERATURE	TEMPERATURE_QC	TEMPERATURE_UNIT	DEPTH	DEPTH_UNIT	FEATURE_TYPE
0	2022-01-01 00:02:01.287	-145.503998	80.081703	-1.5947	0	degree_C	3.957628	m	Profile
1	2022-01-01 00:02:01.287	-145.503998	80.081703	-1.5948	0	degree_C	5.936413	m	Profile
2	2022-01-01 00:02:01.287	-145.503998	80.081703	-1.5947	0	degree_C	7.915180	m	Profile
3	2022-01-01 00:02:01.287	-145.503998	80.081703	-1.5947	0	degree_C	9.893927	m	Profile
4	2022-01-04 00:02:01.824	-156.945602	73.264198	-1.5380	0	degree_C	3.958745	m	Profile
...	...	...	...	...	...	...	...	...	...
3540982	2022-12-30 15:45:00.000	-52.298000	-47.342499	14.6800	0	degree_C	4.958282	m	Profile
3540983	2022-12-30 15:45:00.000	-52.298000	-47.342499	14.6800	0	degree_C	5.850760	m	Profile
3540984	2022-12-30 15:45:00.000	-52.298000	-47.342499	14.6790	0	degree_C	6.842398	m	Profile
3540985	2022-12-30 15:45:00.000	-52.298000	-47.342499	14.6780	0	degree_C	7.834031	m	Profile
3540986	2022-12-30 15:45:00.000	-52.298000	-47.342499	14.6770	0	degree_C	8.924822	m	Profile

3540987 rows × 9 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 year 2022.

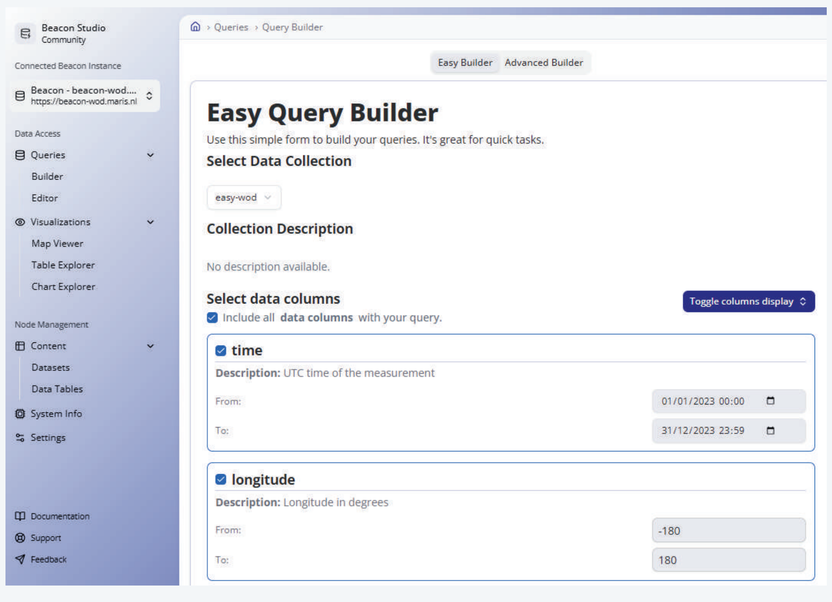


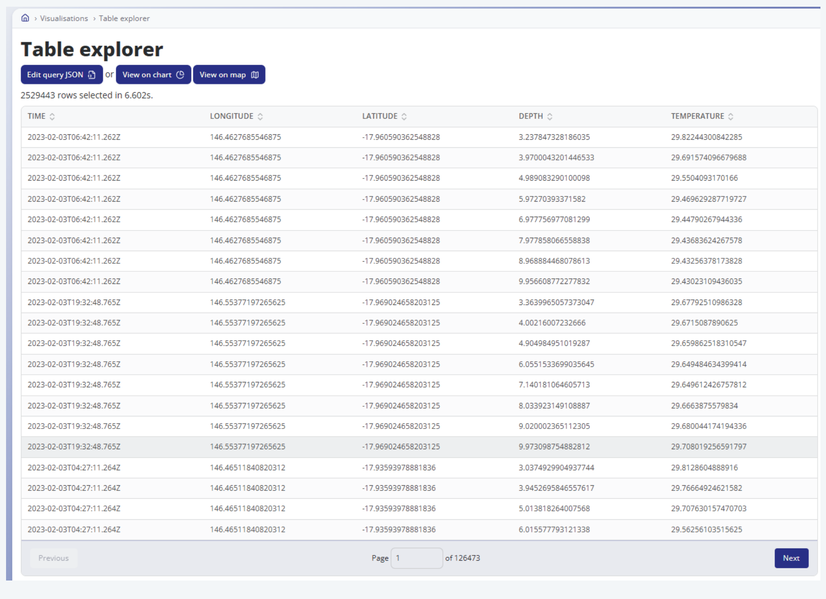


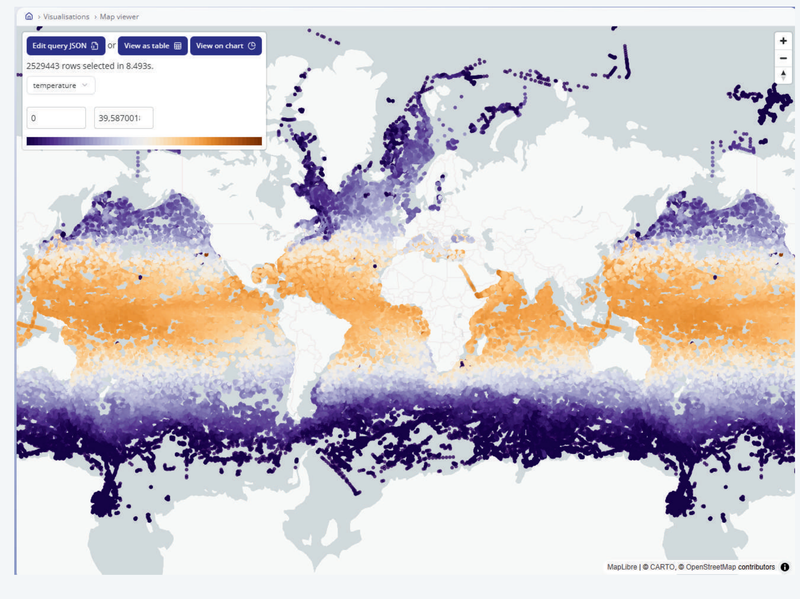
Beacon Studio

Apart from Jupyter Notebooks, there is a also a User Interface available on top of Beacon instances for the non-technical user. This allows via pre-set tables defined by the Beacon instance manager to filter on the data collection.

There are different outputs available, on of them is the table view that can be seen here, while another option is a map viewer that plots the data points on the map. Next to this, it is possible to download the dataset as well.







eosc | Blue-Cloud2026
A federated European FAIR and Open Research Ecosystem for oceans, seas, coastal and inland waters

eosc | FAIR-EASE
Building Interoperable Earth Science & Environmental Services

SeaDataNet

EOSC Future

Contact MARIS
Download now!
AGPLv3 license
beacon.maris.nl
github.com/maris-development/beacon