

# Welcome!

## Contributing Data to EDITO

Learn how to contribute your marine datasets to the EDITO Data Lake using STAC standards.

Presented by **Samuel Fooks**  
*Flanders Marine Institute (VLIZ)*



# What We'll Cover

- What is STAC?** - Standard metadata format
- EDITO Data Lake** - How it works
- Creating STAC Items** - From your data files
- Supported Formats** - NetCDF, Zarr, Parquet
- Posting to EDITO** - Official API documentation



# What is STAC?

**STAC = SpatioTemporal Asset Catalog**

**A standardized way to describe geospatial data:**

- JSON-based metadata format
- Describes when, where, and what your data contains
- Links to actual data files
- Searchable and discoverable

**STAC Specification:** [stacspec.org](https://stacspec.org)

- Open standard (v1.0.0)
- Defines structure for Catalogs, Collections, and Items
- Ensures interoperability across tools and platforms



# STAC Specification

## Key Concepts:

- 📦 **Catalog** - Top-level container, links to Collections
- 📚 **Collection** - Groups related Items (e.g., climate forecasts)
- 📄 **Item** - Individual dataset with geometry, properties, assets
- 🔗 **Asset** - Link to actual data file (NetCDF, Parquet, etc.)

## STAC Spec Benefits:

- Standardized structure across all geospatial data
- Machine-readable metadata
- Enables search and discovery
- Tool interoperability



# STAC Item Structure

## Required Fields (STAC Spec):

- `id` - Unique identifier
- `type` - Must be "Feature"
- `stac_version` - STAC version (e.g., "1.0.0")
- `geometry` - GeoJSON geometry (Polygon, Point, etc.)
- `properties` - Must include `datetime` OR `start_datetime` / `end_datetime`
- `assets` - Links to actual data files

**Recommended:** Title, description, providers, bbox



# Reading STAC Catalogs

## readstac.py - Explore Existing Data

```
import pystac

stac_url = "https://api.dive.edito.eu/data/catalogs/Galicia_CCMM_catalog"
stac = pystac.Catalog.from_file(stac_url)

# Save locally for offline exploration
stac.normalize_and_save("data/mystac/", catalog_type="SELF_CONTAINED")
```

### What it does:

- Connects to EDITO STAC catalogs
- Downloads metadata for offline exploration
- Preserves catalog structure locally



# Creating STAC Items

## makestac.py - Learn STAC Structure

```
from pystac.validation import validate_dict
import pystac

metadata = {
    "type": "Feature",
    "stac_version": "1.0.0",
    "id": "example-item-001",
    "properties": {
        "datetime": "2020-01-01T12:00:00Z",
        "start_datetime": "2020-01-01T12:00:00Z",
        "end_datetime": "2020-02-01T12:00:00Z"
    },
    "geometry": {
        "type": "Polygon",
        "coordinates": [[[5.0, 51.0], [5.1, 51.0],
                         [5.1, 51.1], [5.0, 51.1],
                         [5.0, 51.0]]]
    },
    "bbox": [5.0, 51.0, 5.1, 51.1],
    "assets": {
        "data": {
            "href": "https://example.org/data/example-item-001.tif",
            "type": "image/tiff; application=geotiff",
            "roles": ["data"]
        }
    }
}
```



# EDITO Data Lake

**Three main components:**

-  **STAC Catalog** - Metadata and discovery
-  **Object Storage** - Actual data files (S3-compatible)
-  **API Access** - [api.dive.edito.eu/data](http://api.dive.edito.eu/data)

**Your workflow:**

1. Create STAC item from your data
2. Upload data to accessible storage
3. Post STAC item to EDITO API



# Creating STAC Items from Data

# Example: make\_stac\_from\_data.py

## Shows one approach to creating STAC items using metadata from a dataset

# Example usage:

```
python make_stac_from_data.py netcdf my_data.nc <data_url>
python make_stac_from_data.py zarr my_data.zarr <data_url>
python make_stac_from_data.py parquet my_data.parquet <data_url>
```

## What it demonstrates:

- Extracting metadata from data files
  - Building STAC item structure
  - Validation process



# Parquet Example

## What the Script Extracts

From Parquet files:

- Temporal range from datetime-typed column (any name)
- Spatial bounds from geometry column OR lat/lon columns
- Provider metadata from Parquet file metadata

## Example snippet:

```
import duckdb

conn = duckdb.connect()
s3_url = 's3://my-bucket/data/my_marine_data.parquet'

# Find datetime column (any name)
columns_info = conn.execute(
    f"DESCRIBE SELECT * FROM '{s3_url}'").fetchall()

# Extract temporal range
datetime_query = f"""
    SELECT MIN({datetime_col}), MAX({datetime_col})
    FROM '{s3_url}'
"""

# Extract spatial bounds
bounds_query = f"""
    SELECT MIN(lon), MIN(lat), MAX(lon), MAX(lat)
    FROM '{s3_url}'
"""


```

# Datetime Handling

**If temporal info is missing:**

The script will prompt for start and end datetime.

**Format:** 2023-01-01T00:00:00Z or 2023-01-01

**Requirement:** Must be UTC (ends with z)



# Best Practices

- ✓ Unique IDs** - Use timestamps or meaningful names
  - ✓ Accurate Geometry** - Verify bounding boxes match your data
  - ✓ Complete Assets** - Include proper URLs and MIME types
  - ✓ Provider Info** - Credit data creators
  - ✓ Data URL** - Must be accessible (S3, MinIO, cloud storage)



# Posting to EDITO

## UNDER CONSTRUCTION



EU  
MISSIONS  
RESTORE OUR OCEAN & WATERS

EDITO European Digital Twin Ocean  
EMODnet Copernicus Geodetic Information Service



# Summary

- ✓ **STAC** - Standard metadata format ([stacspec.org](https://stacspec.org))
- ✓ **Example Scripts** - Demo STAC item creation
- ✓ **Parquet** - Example with DuckDB
- ✓ **Validation** - Automatic validation per STAC spec

## Next Steps:

- Create STAC items from your data
- Upload data to accessible storage
- Post to EDITO API (see official docs)

# 🎉 Thank You!

## Resources:

-  **GitHub:** [edito-workshops-presentations](#)
-  **EDITO:** [dive.edito.eu](#)
-  **Contact:** [samuel.fooks@vliz.be](mailto:samuel.fooks@vliz.be)

**Example Scripts:** `readstac.py` , `makestac.py` , `make_stac_from_data.py`

**Happy STAC item creation!** 