EOPF Sentinel Data Processor Re-engineering status and plans

Vincent Dumoulin

ESA ESRIN

18/02/2025

Agenda

- ➢ Sentinels Data Processors and Data Format Evolution
- ➢ Usage of DGGS
- ➢ Zarr for AI
- ➢ Roadmap
- ➢ Sentinels EOPF Sample Service

Sentinels data processors and data format evolution

Why evolving the Sentinel data format?

Not a 'nice-to-have,' but a fundamental priority to:

- Optimise access to the large data volume generated since the start of the S1 operations in 2014 and prepare for the access to the significnatly larger volumes to be generated by the Coperncius Expansion missions as from 2028

- Facilitate combination of multi-Sentinel data, harmonising representation and access across missions

- Foster the evolution from indivual product download to a streamlined data access, facilitating the access to the target data subset, across missions and along time

- Support the readiness of Sentinel data for AI analysis and applications

- Keep a leading Europe's role in the advanced use of Sentinel data

N.B. Current Sentinels data format is based on the "Standard Archive Format for Europe" (SAFE), originally designed to be used in an archive system for long term preservation of EO data.

Why evolving the Sentinel data processors?

An essential objective, not just a convenience, to:

- Ensure the long-term ability to process the invaluable amounts of data generated by the Sentinel satellites
- Ensure the long-term maintenance and evolution of the Sentinels data processing algorithms and processors
- Mitigate industrial and technical lock-ins, supporting long-term programme affordability
- Support European industrial competitiveness and innovation
- Maximise future data processing and access flexibility, tailoring to specific user and applications needs, further valorising the use of Sentinel data

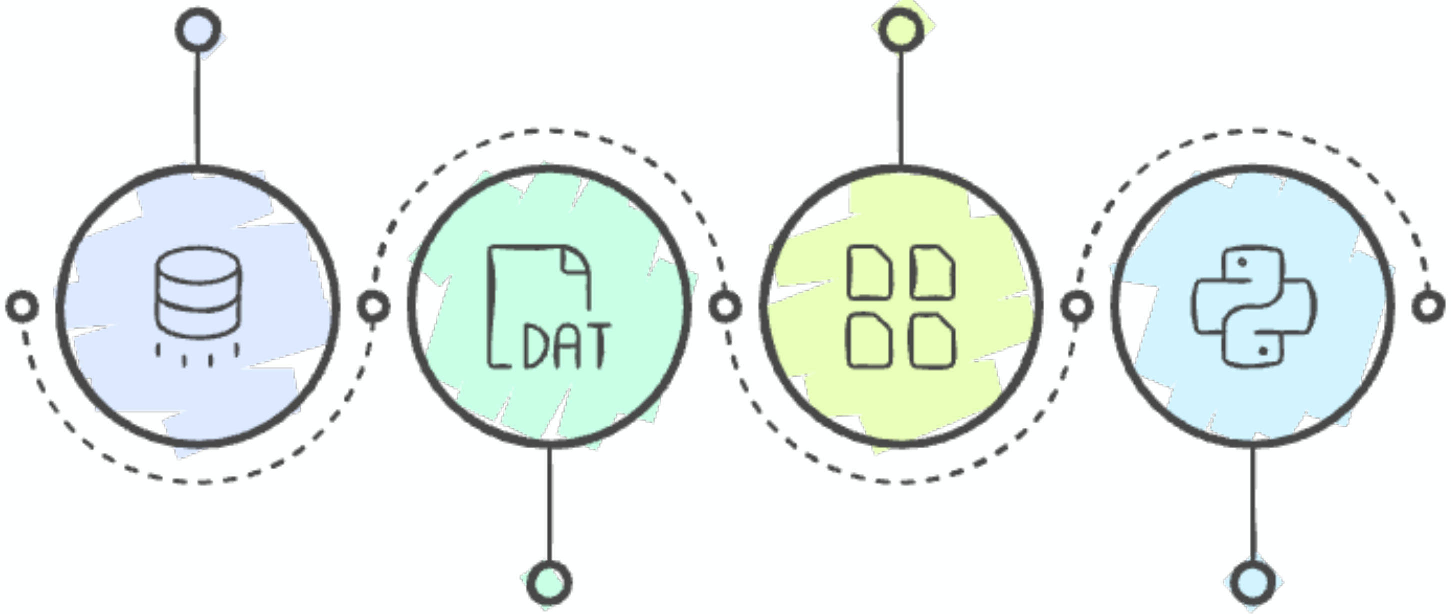
Evolution Overview

Definition of a Meta Model: EOPF

Creating an abstraction framework for Sentinel Products

Data Representation Selection

Choosing Zarr for efficient data access & storage



Definition of STAC Metadata

Establishing metadata for satellite data discovery

Development of Sentinel Processors in Python

Leveraging Zarr and Dask

Data Model and Product Design

Cloud Optimized Format

- Cloud storage for data dissemination and concurrent processing
- Cloud-optimized data formats (Zarr)

Open Source

- Python-based, native support to dask/numpy/xarray
- Large adoption in the data science and EO community

Self-explanatory Product

- Clear documentation of the metadata
- User accessibility

Compliant with CF standard

- Based on Unidata Common Data Model
- Well-established formats and best practices

Interoperability

- Standards and protocols for easy integration of EO datasets and applications

Conversion Capabilities

- Backward compatibility
- New EOPF model to SAFE/COG/NETCDF
- **SAFE to new EOPF model**

Flexible and Scalable

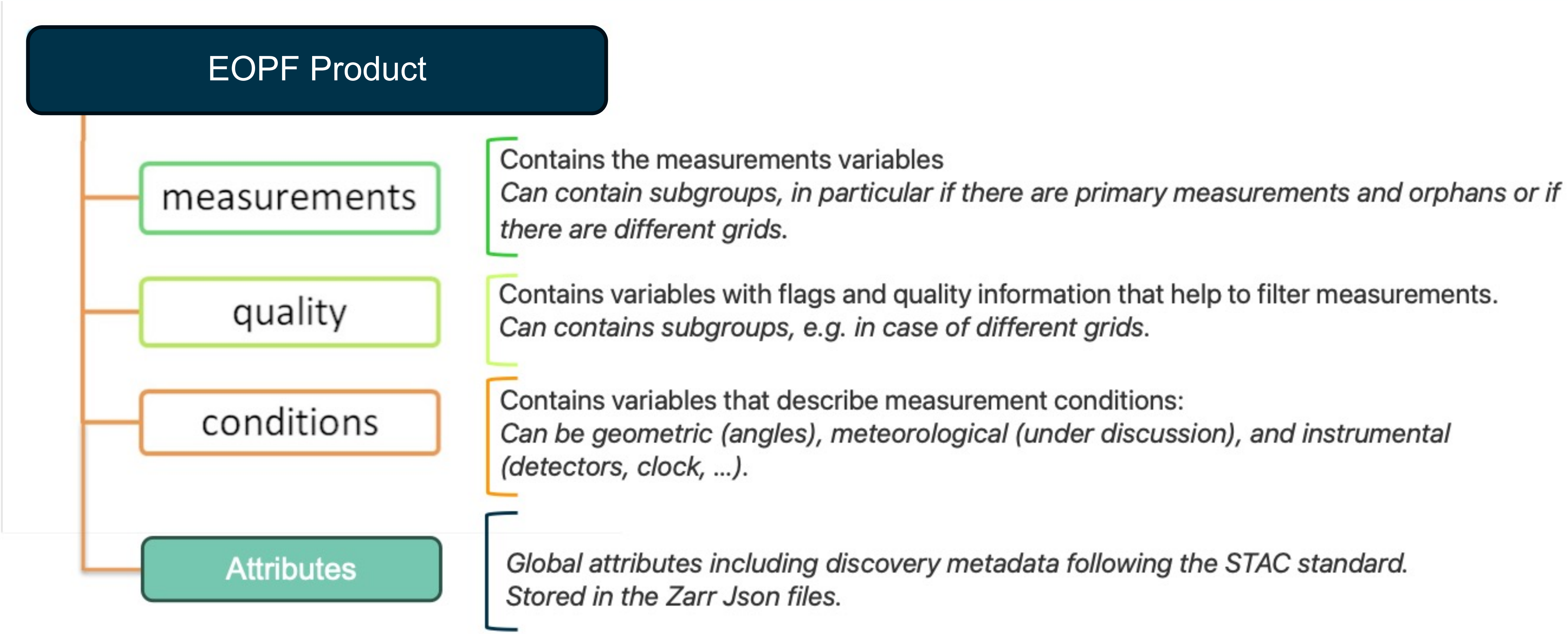
- Accommodate different types of EO data and sensor technologies
- From small scale observation to large datasets
- Time series ready

Harmonized Structure

- Unified structure across Sentinels
- Copernicus Expansion

EOPF Product Structure

The EOPF Product Structure is a standardized meta model applicable across all Sentinel missions and generally any EO mission, offering complete flexibility for defining extensions, which is then tailored in detail for each Sentinel product.



EOPF Product

measurements

Contains the measurements variables
Can contain subgroups, in particular if there are primary measurements and orphans or if there are different grids.

quality

Contains variables with flags and quality information that help to filter measurements.
Can contains subgroups, e.g. in case of different grids.

conditions

Contains variables that describe measurement conditions:
Can be geometric (angles), meteorological (under discussion), and instrumental (detectors, clock, ...).

Attributes

Global attributes including discovery metadata following the STAC standard.
Stored in the Zarr Json files.

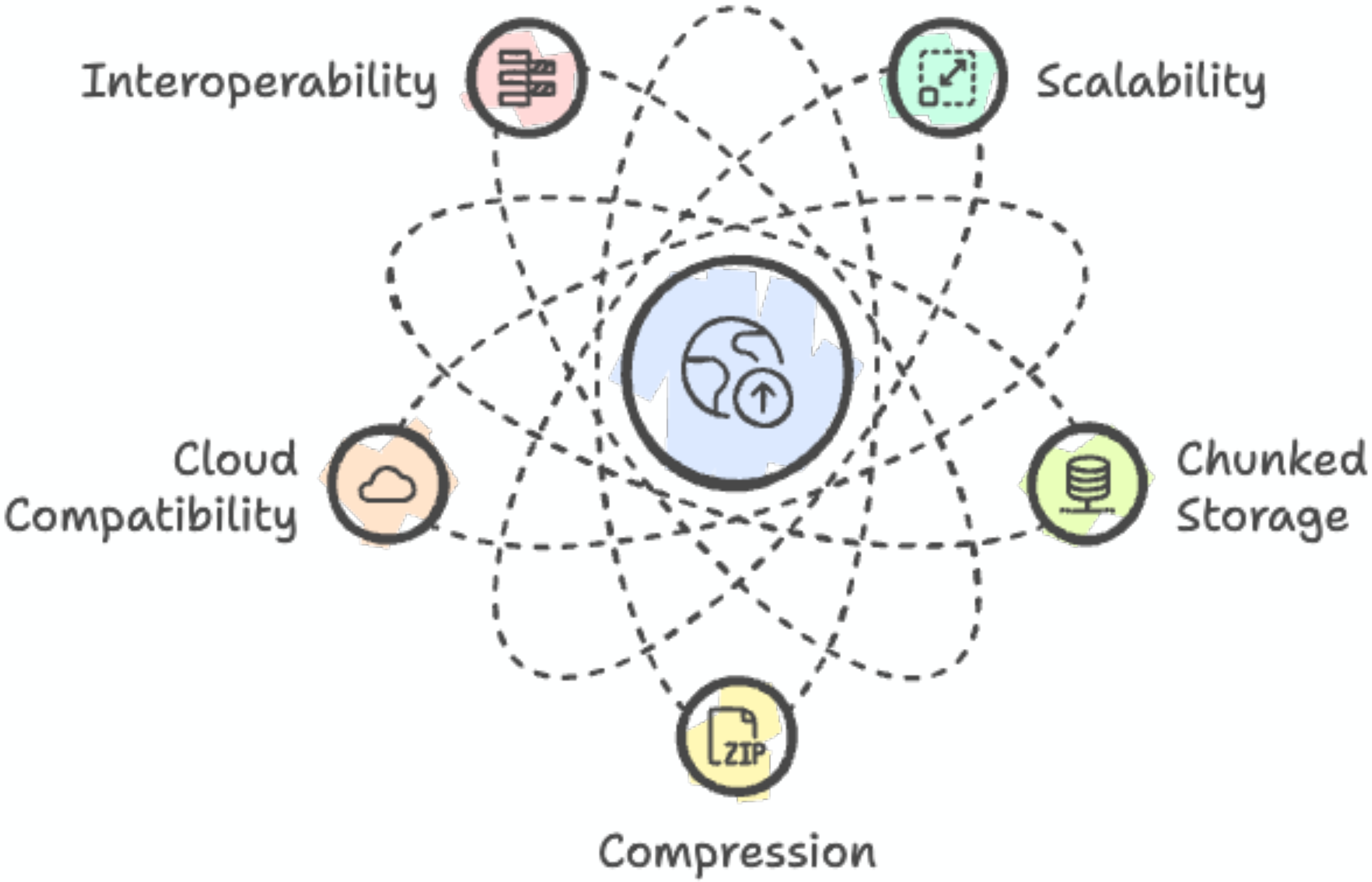
EOPF Metadata

The Product Global Attribute incorporates three primary metadata categories:

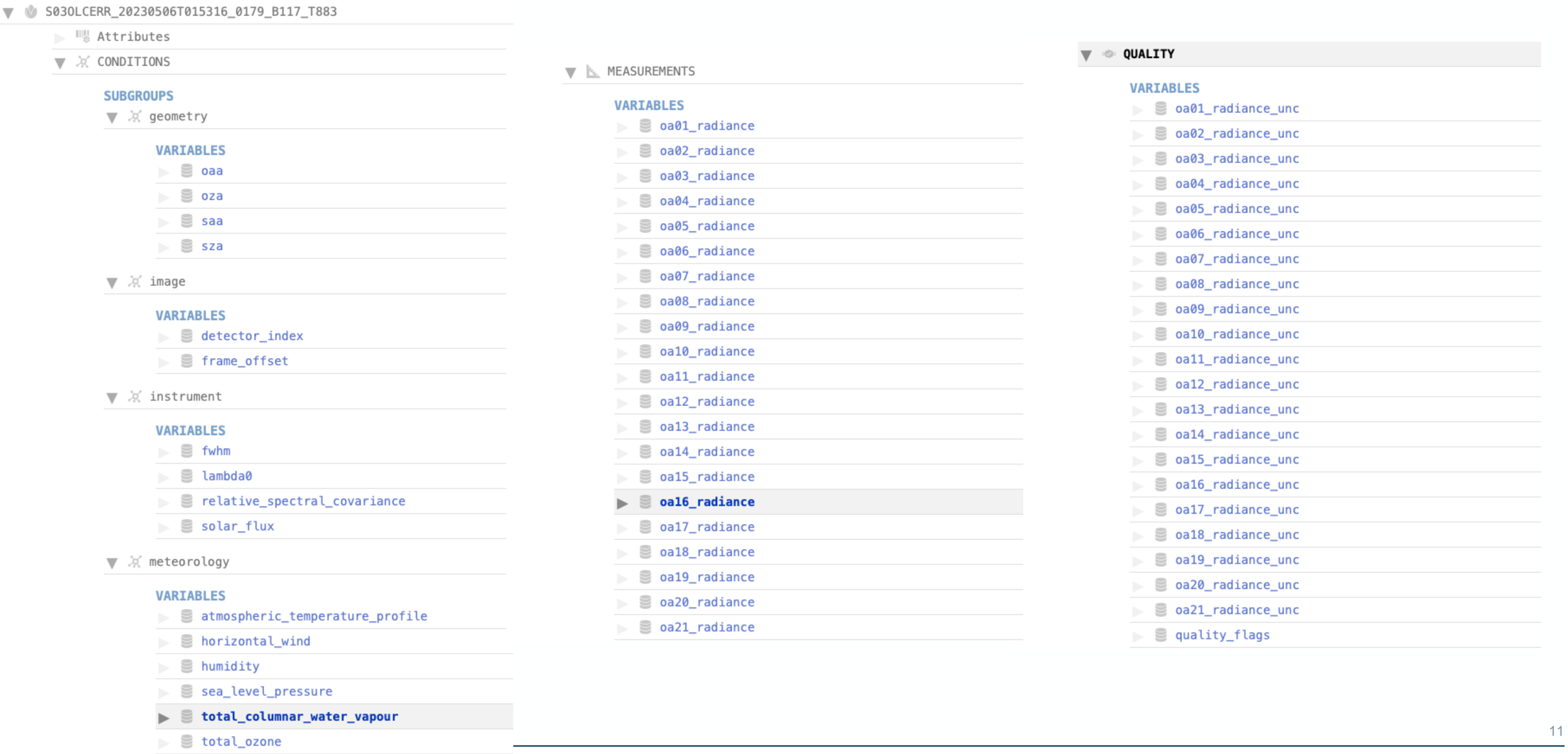
- **Discovery Metadata → STAC**:
    - Can utilise extensions encompassing “processing,” “sat,” "sar," “view,” “eo,” “scientific,” “projection,” and “classification.”
    - Integration of mission-specific extensions is avoided to achieve harmonized metadata across various missions.
    - Additionally, an EOPF STAC extension to accommodate specific fields typical to (ESA) EO catalogues, such as processing baseline, datatake_id, etc.
- **Other Metadata → JSON:** elements beyond product discovery, providing extensive details such as sensor states, orbit parameters, etc.
- **Processing History Metadata → JSON:** a comprehensive record of all processing steps post-acquisition.

EOPF data representation

Zarr is an open-source format for chunked, compressed, N-dimensional arrays



EOPF Product Storage: Example S3 L1 OLCI ERR



- S03OLCERR_20230506T015316_0179_B117_T883
  - Attributes
  - CONDITIONS
    - SUBGROUPS
      - geometry
        - VARIABLES
          - oaa
          - oza
          - saa
          - sza
      - image
        - VARIABLES
          - detector_index
          - frame_offset
      - instrument
        - VARIABLES
          - fwhm
          - lambda0
          - relative_spectral_covariance
          - solar_flux
      - meteorology
        - VARIABLES
          - atmospheric_temperature_profile
          - horizontal_wind
          - humidity
          - sea_level_pressure
          - **total_columnar_water_vapour**
          - total_ozone

- MEASUREMENTS
  - VARIABLES
    - oa01_radiance
    - oa02_radiance
    - oa03_radiance
    - oa04_radiance
    - oa05_radiance
    - oa06_radiance
    - oa07_radiance
    - oa08_radiance
    - oa09_radiance
    - oa10_radiance
    - oa11_radiance
    - oa12_radiance
    - oa13_radiance
    - oa14_radiance
    - oa15_radiance
    - **oa16_radiance**
    - oa17_radiance
    - oa18_radiance
    - oa19_radiance
    - oa20_radiance
    - oa21_radiance

- **QUALITY**
  - VARIABLES
    - oa01_radiance_unc
    - oa02_radiance_unc
    - oa03_radiance_unc
    - oa04_radiance_unc
    - oa05_radiance_unc
    - oa06_radiance_unc
    - oa07_radiance_unc
    - oa08_radiance_unc
    - oa09_radiance_unc
    - oa10_radiance_unc
    - oa11_radiance_unc
    - oa12_radiance_unc
    - oa13_radiance_unc
    - oa14_radiance_unc
    - oa15_radiance_unc
    - oa16_radiance_unc
    - oa17_radiance_unc
    - oa18_radiance_unc
    - oa19_radiance_unc
    - oa20_radiance_unc
    - oa21_radiance_unc
    - quality_flags

Demonstration

Sentinel data processors redesign

- Legacy Sentinel data processors modernisation
- Open-source code
- Preserving the original algorithm and data quality
- Benefit from new technologies
- Generic Framework for future missions (COPEX, NG, …)
- Native Zarr data management

ESA Copernicus Earth Observation Processor Framework (EOPF) facilitates the transition of the Copernicus Sentinel operations towards a more optimized approach which:

- enables customizable processing,
- enhances flexibility and scalability,
- streamlines access to large volumes of data.

Availability of a first version of the Processors is expected for second half of 2025

Available resources

- All the resources can be found here:
  - https://eopf.copernicus.eu/
- Product format currently available:
  - Common Product Spec: https://cpm.pages.eopf.copernicus.eu/eopf-cpm/main/PSFD/
  - Common ADF: https://cpm.pages.eopf.copernicus.eu/adf-auxiliary-data-file/main/adfs/index.html
  - Sentinel 1: https://s1.pages.eopf.copernicus.eu/s1-l12-rp/main/pfs/index.html
  - Sentinel 2: https://s2.pages.eopf.copernicus.eu/pdfs-adfs/
  - Sentinel 3: https://s3.pages.eopf.copernicus.eu/pdfs-adfs/
- Source code for the CPM library:
  - https://gitlab.eopf.copernicus.eu/cpm/eopf-cpm

Discrete Global Grid Systems (DDGS)

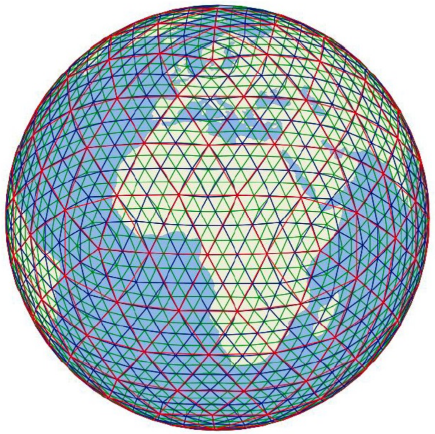
Introduction & Background

- **Context**:
  - The Copernicus Space Component and the Destination Earth (DestinE) initiative generate and make available vast amounts of diverse Earth Observation (EO) and climate data.
  - Different data representation, structures, projections and formats complicate the synergetic use and analysis of large volumes of diverse data.
- **Need**:
  - An **interoperable data representation** that streamlines the combination of large volumes of EO and climate data for easier processing, visualization, and analysis.
- **Solution**:
  - Adoption of **Discrete Global Grid Systems (DGGS)** in combination with the Zarr **cloud-optimized format** (EOPF for the Sentinels) to harmonise the data representation across diverse types of data, including different types of EO sensors and climate data.

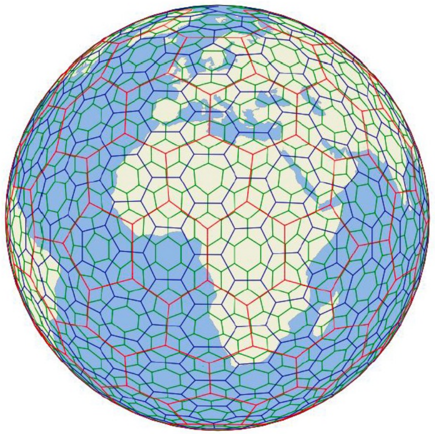
Discrete Global Grid Systems (DDGS)

- **Definition**:
  - A DGGS partitions the Earth's surface into a hierarchy of **cells** with discrete boundaries, enabling systematic storage, indexing, and analysis of geospatial data.
- **Key Features**:
  - **Multi-Resolution**: Cells can be refined at finer levels while preserving relationships to coarser grids.
  - **Spatial Indexing**: Each cell has a unique identifier, supporting efficient queries and data retrieval.
  - **Uniform Coverage**: Ensures consistent global representation with minimal distortion.
  - **Interoperability**: Standardizes data representation across different sources and scales.
- **Why DGGS?**
  - Facilitates **seamless integration** of large volumes of EO multi-mission and multi-domain data.
  - Enables **fast, scalable** analytics in cloud-based or big data environments.
  - Offers a **common reference** framework for diverse geospatial applications.

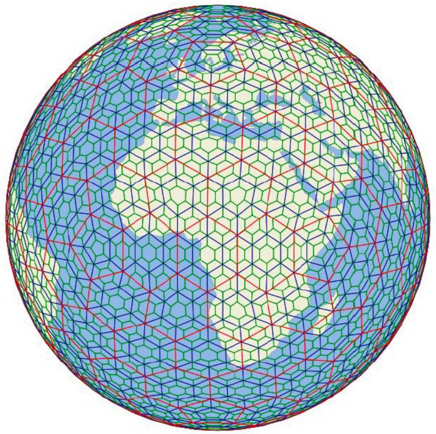
Discrete Global Grid Systems (DDGS)



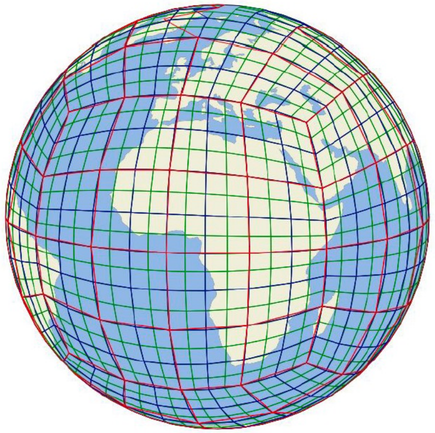
ISEA4T



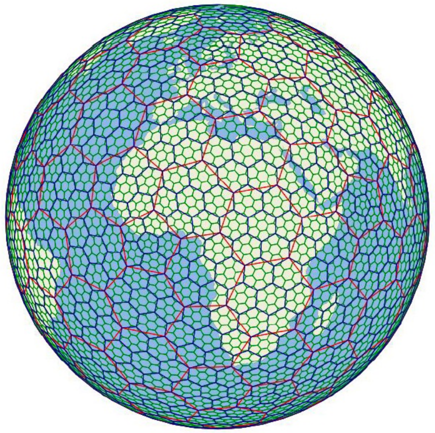
ISEA3H



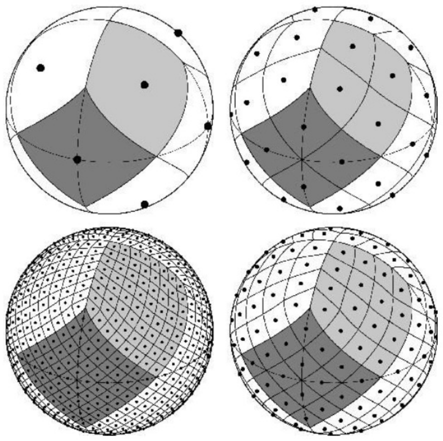
ISEA4H



Google S2



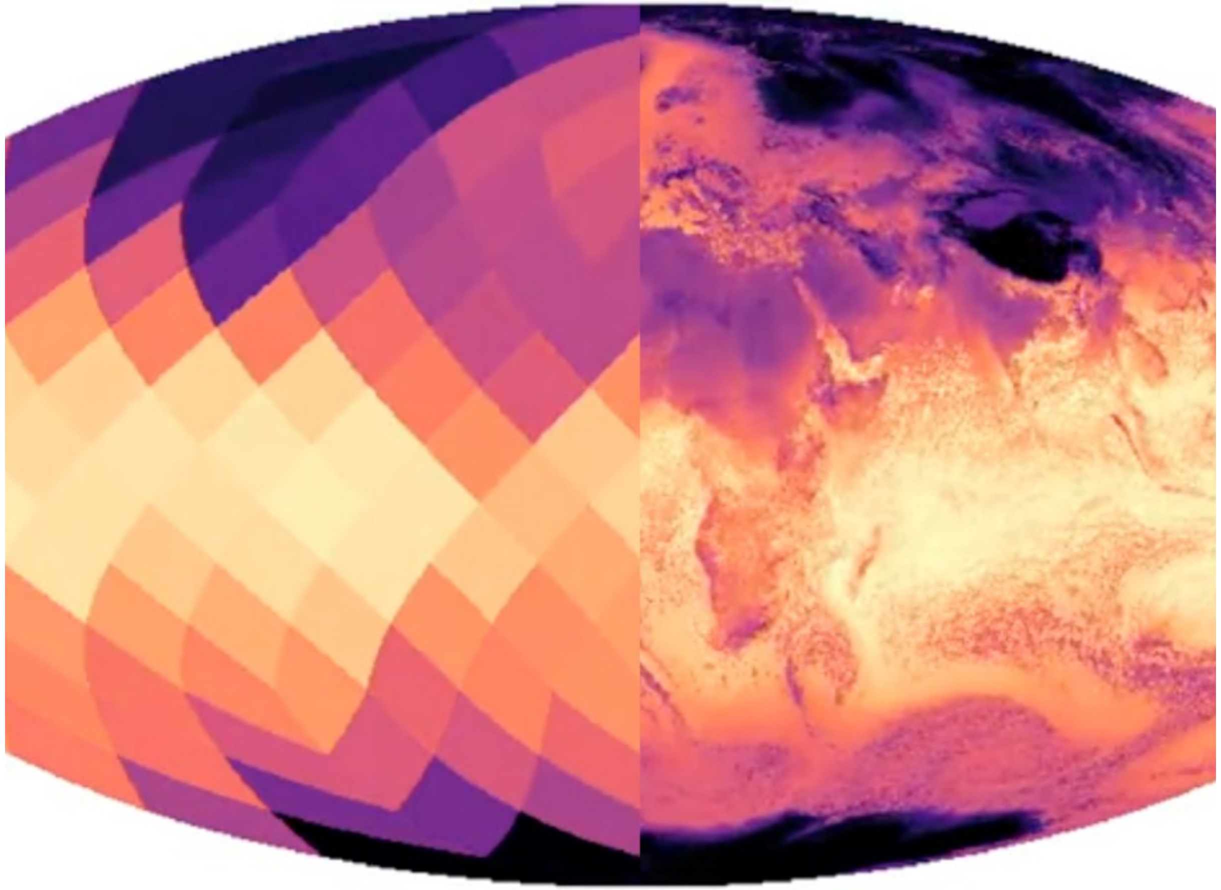
Uber H3



HEALPix

HEALPix (Hierarchical Equal Area isoLatitude Pixelation)

- **Overview**:
  - A specific DGGS originally developed for astronomy (cosmic microwave background mapping).
  - Provides an **equal-area**, **iso-latitude** tiling of the sphere, ensuring uniform coverage.
- **Core Characteristics**:
  - **Equal Area**: Each cell covers the same surface area, avoiding spatial bias.
  - **Hierarchical**: Resolutions can be subdivided while preserving cell boundaries.
  - **isoLatitude**: Rows of pixels lie on lines of constant latitude, simplifying certain spherical computations.
  - **Efficient Indexing**: Unique integer indices enable fast lookups.
- **HEALPix in This Project**:
  - Chosen as the **spatial backbone** for the DGGS-based format
  - Well-supported by **open-source** libraries, making it easier to adopt and extend.
  - Allows **multi-resolution** Earth observation data to be integrated consistently.
  - Currently used in **DestinE** for **DT Climate**
  - Support for efficient **Earth Data Cube**



Objectives

1. **Design a Generic DGGS representation**
   - Based on **HEALPix** and **Zarr** V3 to store and access EO and climate data efficiently.
2. **Instantiate Format for Sentinel & DestinE Products**
   - Sentinel data starting with Sentinel-2 (L1C/L2A) and Sentinel-3 (OLCI/SLSTR/SYN),
   - DestinE atmospheric/climate datasets,
   - Copernicus DEM.
3. **Develop & Demonstrate Processing Workflows**
   - Demonstrate benefits in geometric corrections, DEM integration, sensor calibration
   - Demonstrate benefits in synergetic data analysis and data combination.
4. **Implement Operational Converters**
   - Allowing to convert legacy datasets into the EOPF-DGGS-based representation (e.g. current Sentinel data into the EOPF-DGGS-based representation)
5. **Introduce native DDGS data representation in the data processors**
   - Demonstrate benefits of performing data processing on data natively represented in the DDGS data representation
     - Sentinel-2 processor (e.g. L1B -> L2A)
     - Sentinel-3 SYN processor
6. **Demonstrate the operational readiness processing DDGS-based data**
   - Initial step: deployment of the processors on the DestinE Core Platform to ease the combined use of Sentinel data with atmospheric/climate datasets

ZARR is all you need

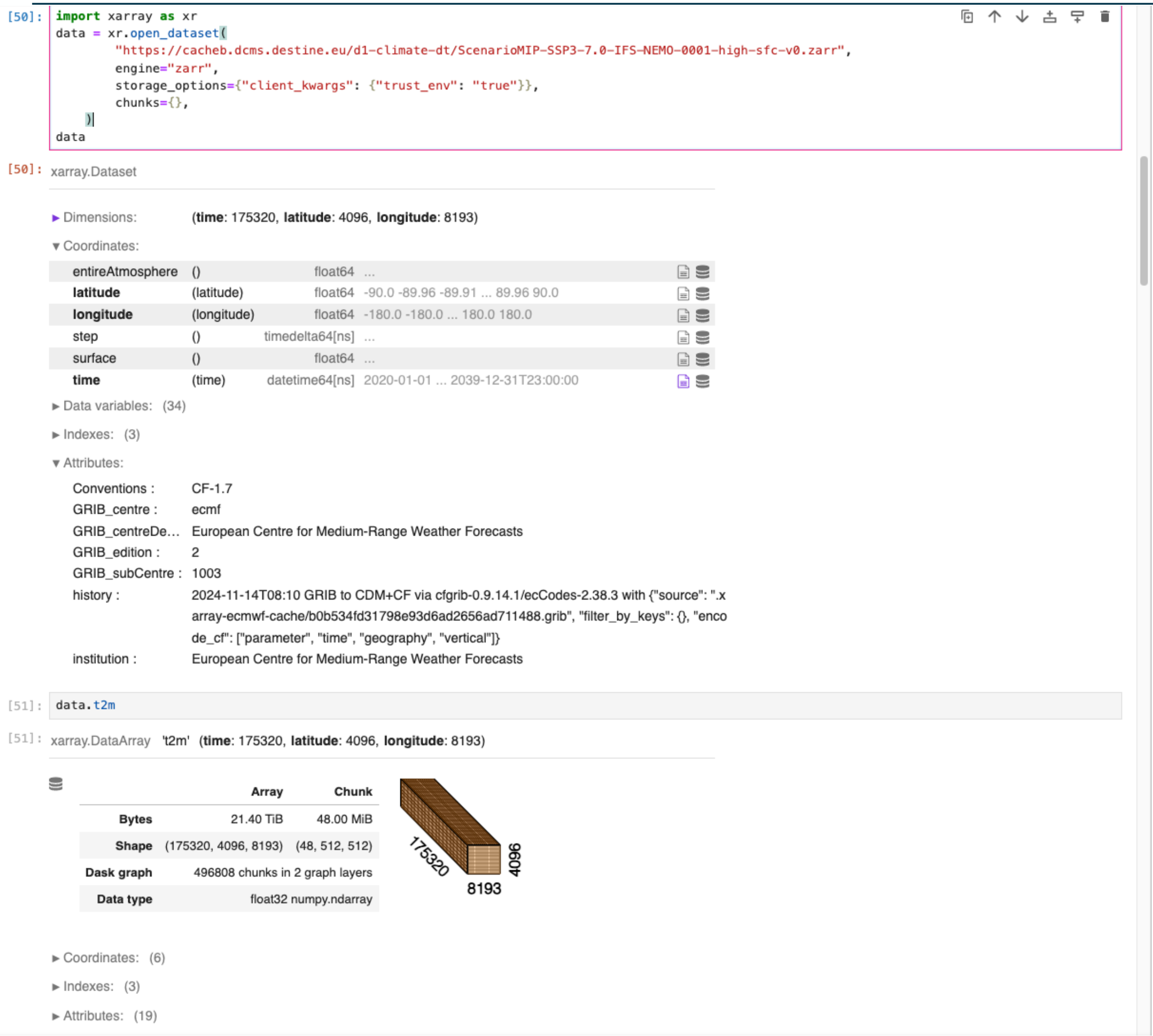
Sébastien Tétaud
sebastien.tetaud@esa.int

20/02/2025

Agenda

- Zarr data access
- Zarr for AI workflow
- Efficient labeling management
- AI reproducibility
- Zarr in action
- Summary

Zarr data access



```
[50]: import xarray as xr
      data = xr.open_dataset(
              "https://cacheb.dcms.destine.eu/d1-climate-dt/ScenarioMIP-SSP3-7.0-IFS-NEMO-0001-high-sfc-v0.zarr",
              engine="zarr",
              storage_options={"client_kwargs": {"trust_env": "true"}},
              chunks={},
          )
      data
```

```
[50]: xarray.Dataset

  ▸ Dimensions:        (time: 175320, latitude: 4096, longitude: 8193)
  ▾ Coordinates:
      entireAtmosphere  ()                float64  ...
      latitude          (latitude)        float64  -90.0 -89.96 -89.91 ... 89.96 90.0
      longitude         (longitude)       float64  -180.0 -180.0 ... 180.0 180.0
      step              ()          timedelta64[ns]  ...
      surface           ()                float64  ...
      time              (time)      datetime64[ns]  2020-01-01 ... 2039-12-31T23:00:00
  ▸ Data variables:  (34)
  ▸ Indexes:  (3)
  ▾ Attributes:
      Conventions :      CF-1.7
      GRIB_centre :      ecmf
      GRIB_centreDe…     European Centre for Medium-Range Weather Forecasts
      GRIB_edition :     2
      GRIB_subCentre :   1003
      history :          2024-11-14T08:10 GRIB to CDM+CF via cfgrib-0.9.14.1/ecCodes-2.38.3 with {"source": ".x
                         array-ecmwf-cache/b0b534fd31798e93d6ad2656ad711488.grib", "filter_by_keys": {}, "enco
                         de_cf": ["parameter", "time", "geography", "vertical"]}
      institution :      European Centre for Medium-Range Weather Forecasts
```

```
[51]: data.t2m
```

```
[51]: xarray.DataArray  't2m'  (time: 175320, latitude: 4096, longitude: 8193)

                            Array             Chunk
            Bytes           21.40 TiB         48.00 MiB
            Shape   (175320, 4096, 8193)  (48, 512, 512)
       Dask graph     496808 chunks in 2 graph layers
        Data type             float32 numpy.ndarray

  ▸ Coordinates:  (6)
  ▸ Indexes:  (3)
  ▸ Attributes:  (19)
```

- Access numpy ndarray data ready to be ingested by Pytorch/TensorFlow to be a tensor
- Standardize data access for data fusion
- Data is stored in compressed chunks, reducing storage costs and improving access speeds.
- Select region of interest and period of interest
- **Current EO data access relies on metadata indexing and HTTP requests, which introduce latency. Zarr eliminates this by storing raw data in an AI-optimized format.**

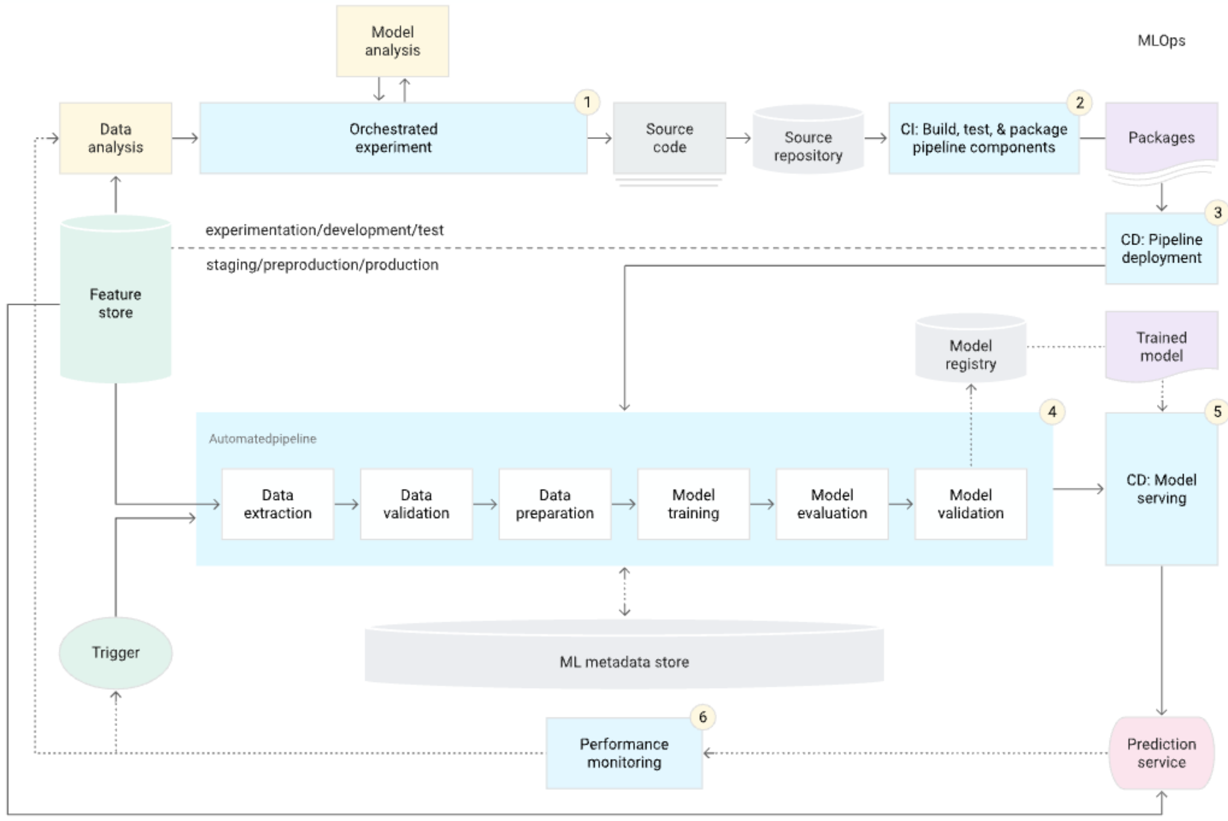
- **AI models require large batches of data, which Zarr provides efficiently by allowing random access to subsets of arrays.**

• **Works Seamlessly with Xarray, Dask, and PyTorch:** Usual format data often needs pre-processing before being fed into AI models, whereas Zarr integrates directly.

• **GPU-Friendly Processing:** Zarr's chunked data storage is optimized for GPU-accelerated workflows.

Zarr for AI workflow

- Training Model needs data to be prepared and analyzed
- **Zarr minimizes the work to deploy the model in production**
- Decrease the ETL coding effort



MLOps

Model analysis

Data analysis

Orchestrated experiment

1

Source code

Source repository

CI: Build, test, & package pipeline components

2

Packages

3

CD: Pipeline deployment

Feature store

experimentation/development/test

staging/preproduction/production

Model registry

Trained model

4

5

Automatedpipeline

Data extraction

Data validation

Data preparation

Model training

Model evaluation

Model validation

CD: Model serving

Trigger

ML metadata store

6

Performance monitoring

Prediction service

[MLOps level 2: CI/CD pipeline automation]()

Model training

Model testing

Dataset Generation

Dataset Analysis

Dataset Cleaning

Data fusion

Data Augmentation

Model deployment

Efficient Label Management supporting AI workflows

Zarr allows preparing and attaching labels to the original data:

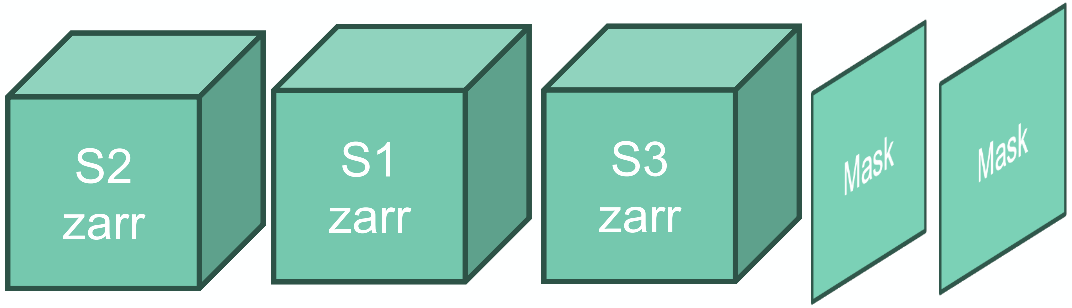
- Support Various Label Types: Handle different label types like segmentation masks and bounding boxes.
- Easily Add or Update Labels: Modify labels without creating new files or buckets.
- Ensure Synchronization: Keep data and labels consistently aligned.
- Consistent Data Versioning: Maintain versions of datasets and labels for reproducibility.

Zarr allows efficient access/parsing to pre-computed labels:

- Streamlined Data Pipeline: Simplify the process of feeding labeled data into machine learning models.
- Reduced Data Preparation Time: Minimize the effort required to prepare data for training.
- Enhanced Collaboration: Facilitate easier sharing and collaboration among team members.

Data segmentation

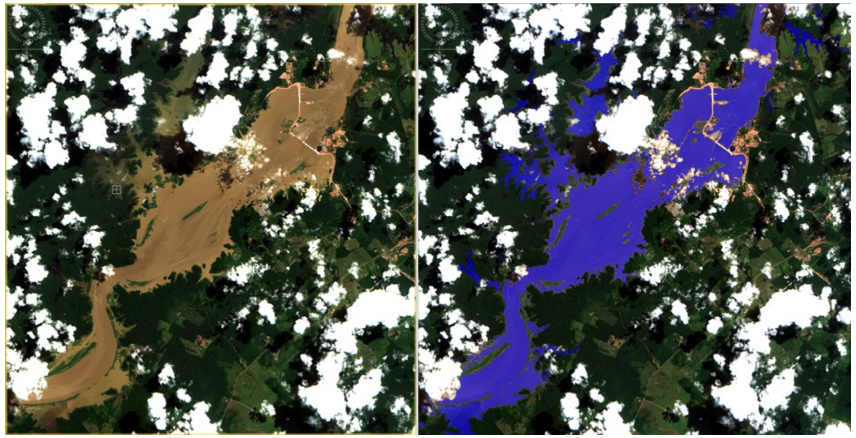
Original image Mask segmentation



+ metadata:

- Image description for LLM
- Object Bounding Box

Data segmentation

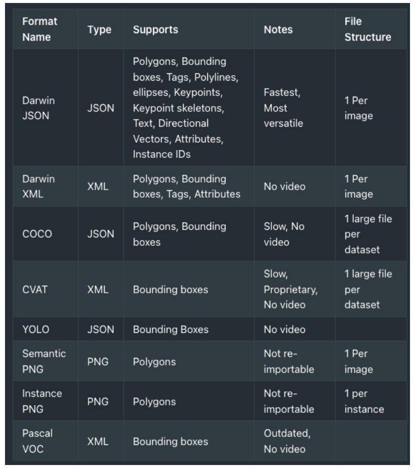


Original image Semantic segmentation

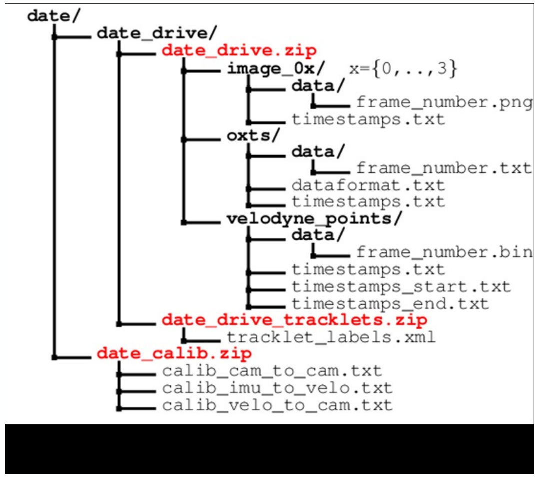
Zarr for AI workflow reproducibility

The complexity of retrieving data for AI workflow reproducibility:

- Public dataset have their own tree architecture
- Need to handle data path -> risk of reproducibility
- File-based formats (GeoTIFF, NetCDF) require precise file path management to be managed
- Some datasets mix file formats (JPEG, TIFF, HDF5), causing preprocessing complexity
- Converting between formats can introduce unintended errors
- Hardcoded paths break across different environments (local/cloud/HPC)



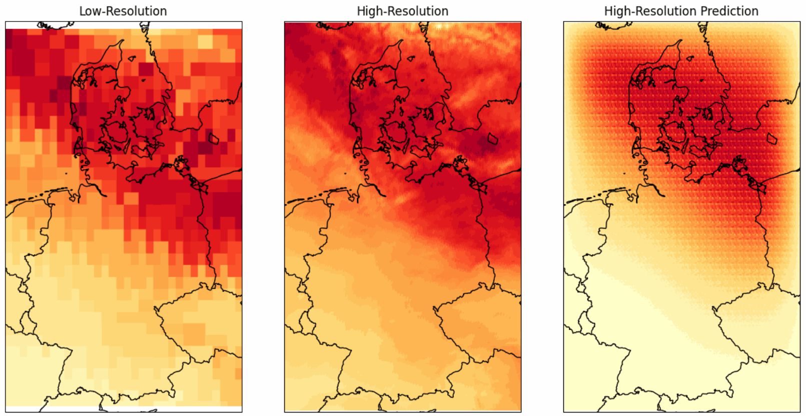
Format Name	Type	Supports	Notes	File Structure
Darwin JSON	JSON	Polygons, Bounding boxes, Tags, Polylines, ellipses, Keypoints, Keypoint skeletons, Text, Directional Vectors, Attributes, Instance IDs	Fastest, Most versatile	1 Per image
Darwin XML	XML	Polygons, Bounding boxes, Tags, Attributes	No video	1 Per image
COCO	JSON	Polygons, Bounding boxes	Slow, No video	1 large file per dataset
CVAT	XML	Bounding boxes	Slow, Proprietary, No video	1 large file per dataset
YOLO	JSON	Bounding Boxes	No video	
Semantic PNG	PNG	Polygons	Not re-importable	1 Per image
Instance PNG	PNG	Polygons	Not re-importable	1 per instance
Pascal VOC	XML	Bounding boxes	Outdated, No video	



```
date/
 └── date_drive/
      └── date_drive.zip
           └── image_0x/   x={0,..,3}
                └── data/
                     └── frame_number.png
                └── timestamps.txt
           └── oxts/
                └── data/
                     └── frame_number.txt
                └── dataformat.txt
                └── timestamps.txt
           └── velodyne_points/
                └── data/
                     └── frame_number.bin
                └── timestamps.txt
                └── timestamps_start.txt
                └── timestamps_end.txt
      └── date_drive_tracklets.zip
           └── tracklet_labels.xml
 └── date_calib.zip
      └── calib_cam_to_cam.txt
      └── calib_imu_to_velo.txt
      └── calib_velo_to_cam.txt
```

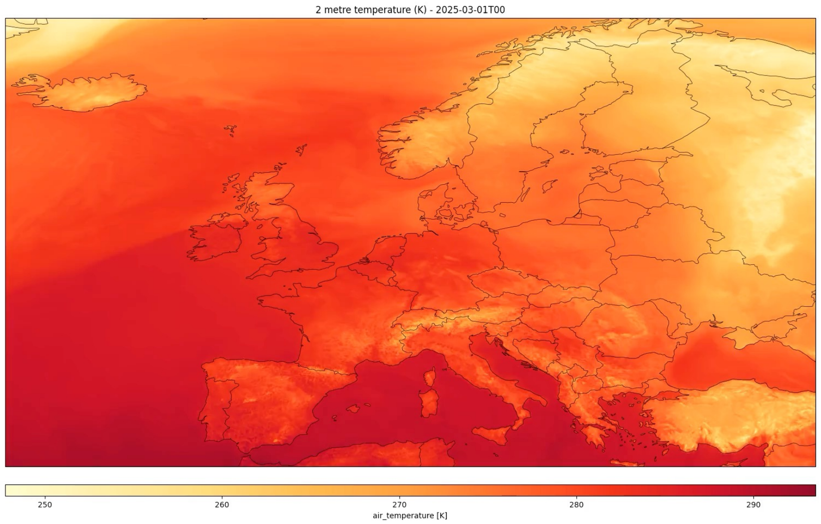
The benefits of coherent Zarr data availability for AI workflow reproducibility:

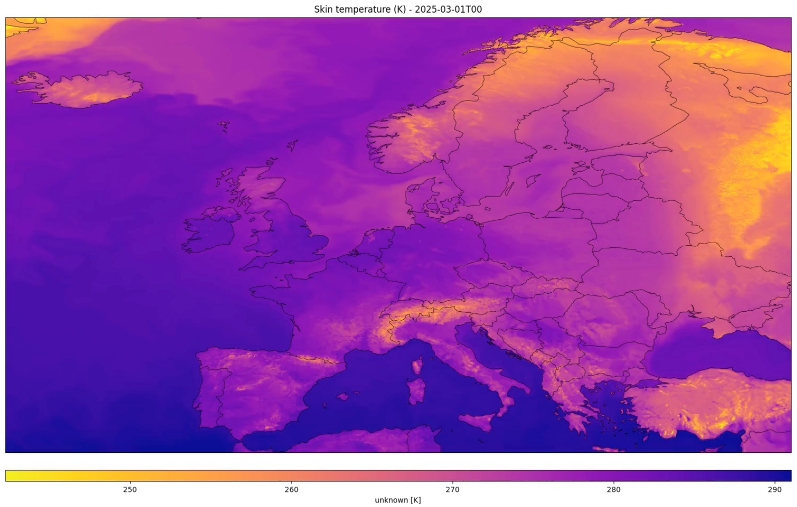
- Zarr data chunks remain unchanged, ensuring consistency across experiments.
  - Pre-processing data Class is the same for the production pipeline
- Zarr provides a **consistent, hierarchical structure**, avoiding file path confusion.
- Easy prototyping on small area and period without download all the data product

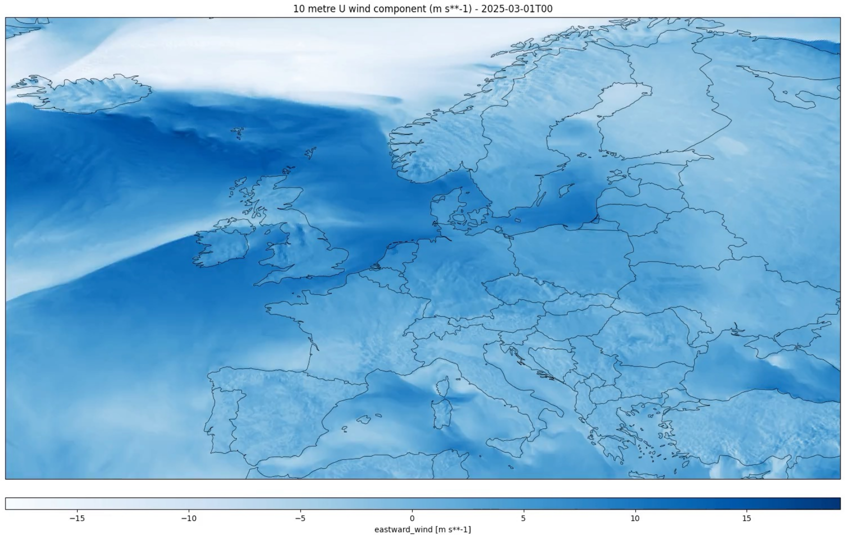


Super Resolution Streaming Training

Zarr in action for DestinE Climate Data Streaming

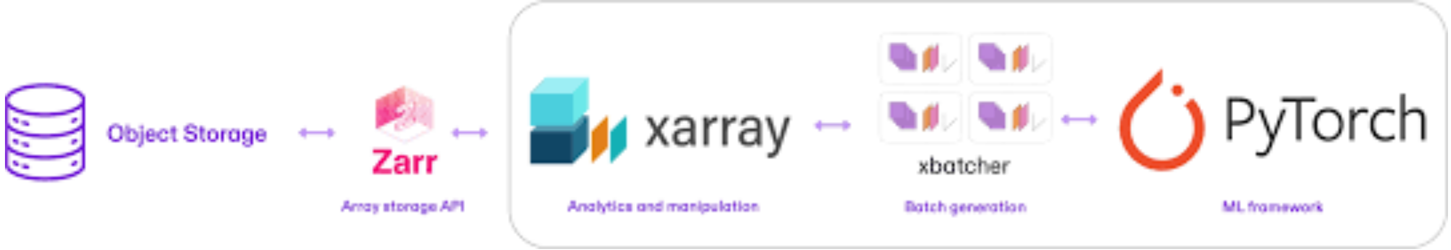






Earth Data Hub Service – Climate DT high resolution

- Representation of data Streaming for AI training inference at 30 FPS (or products per second) like
- DestinE leverage OVH capabilities and can achieve up to 675 products per second.
- Data ready to be ingested by AI pipeline with in few line of code



Summary of Zarr data benefits for AI

Feature	Traditional File Systems (GeoTIFF, NetCDF, JPEG, COG)	Zarr
Cloud-Native Access	No (lacking subset access flexibility)	Yes
Parallel I/O Processing	Limited	High
Random Access to Data Chunks	No (requires full file loads)	Yes
Streaming for AI Training	No (entire dataset must be downloaded)	Yes (on-demand access)
GPU-Friendly Processing	Limited	Optimized for acceleration
Augmentation Performance	Slower due to I/O limitations	Faster (random access to chunks)
Multi-Sensor & Multi-Resolution	Requires multiple file handling	Direct access
Version Control & Reproducibility	Complex, error-prone	Self-contained and structured
Data Fusion	Manual merging of different formats	Standardized access
ETL Complexity	High (file conversions, path handling)	Minimal coding effort + Direct access
Labeling	Separate storage required	Labels can be added directly

Thank you

Sebastien Tetaud

20/02/2025

Supporting the transition
Key principles and action lines

Preparing and supporting the transition from the current CSC operations scenario to the target scenario (Zarr-EOPF based processing and data access)

Ensure backwards compatibility with format conversion tools

Availability of tools, libraries and plug-ins to support adaptation of user applications

Allow for sufficient familiarisation time before start of new operational data flow

Early availability of representative data flow, APIs, notebooks, AI worksflows… to demonstrate benefits and foster early adoption

Early advance notice and information to support familiarisation and planning

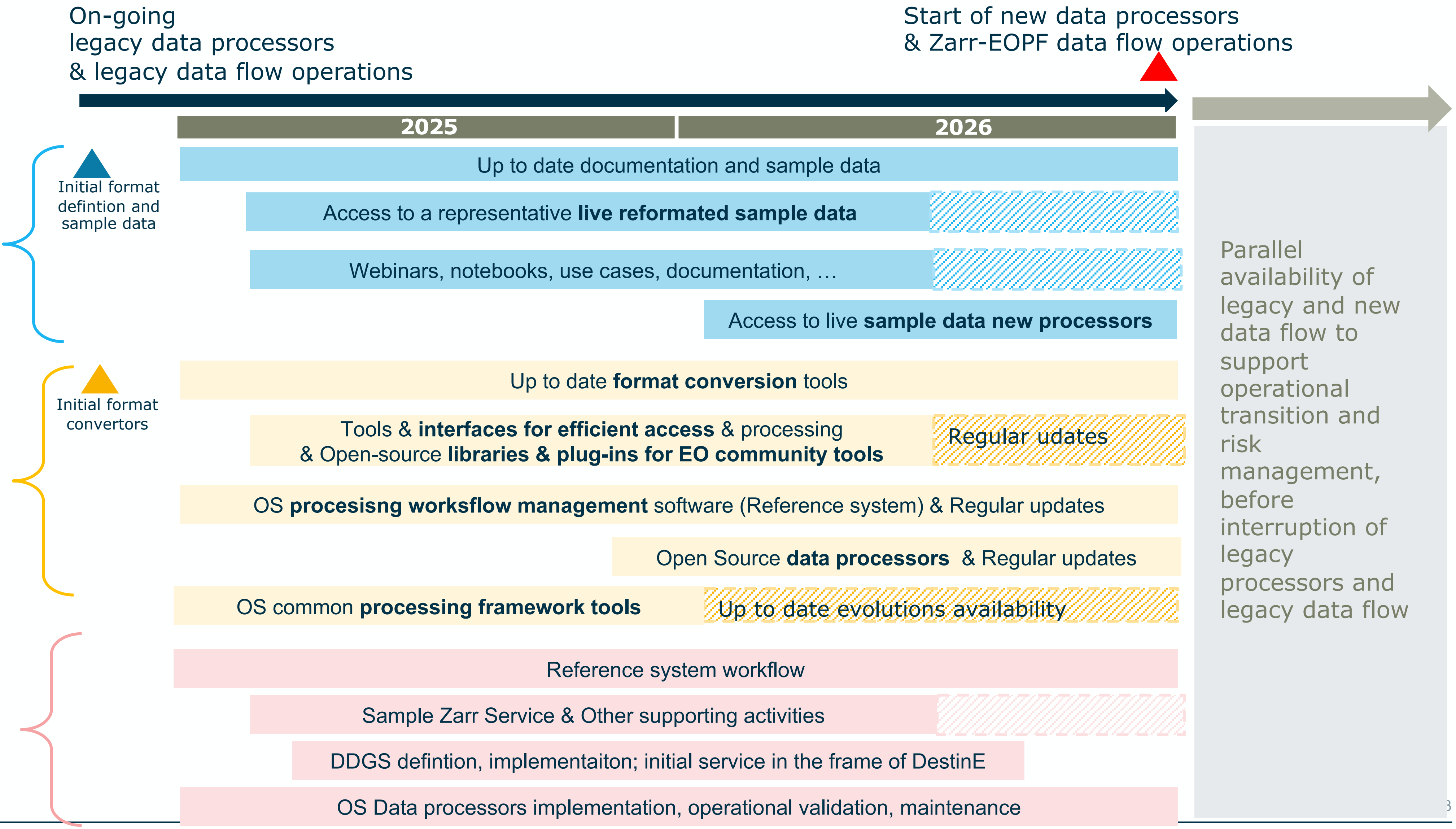
Foster coherence across Copernicus and DestinE/DTE technical data management

Foresee a period of parallel availability of legacy and new data flow to support smooth user adaptation

Dedicated Collaborative support activities

Preparing the future

From legacy data operations to new zarr-based operations



On-going
legacy data processors
& legacy data flow operations

Start of new data processors
& Zarr-EOPF data flow operations

2025

2026

Familiarisation activities:
Documentation, webinars, notebooks, sample data access….

Initial format defintion and sample data

Up to date documentation and sample data

Access to a representative **live reformated sample data**

Webinars, notebooks, use cases, documentation, …

Access to live **sample data new processors**

Software, tools, plug-ins:
Open source software to support access, processing and integration of new data formats in user activities

Initial format convertors

Up to date **format conversion** tools

Tools & **interfaces for efficient access** & processing
& Open-source **libraries & plug-ins for EO community tools**

Regular udates

OS **procesisng worksflow management** software (Reference system) & Regular updates

Open Source **data processors** & Regular updates

OS common **processing framework tools**

Up to date evolutions availability

Key supporting industrial activities and teams

Reference system workflow

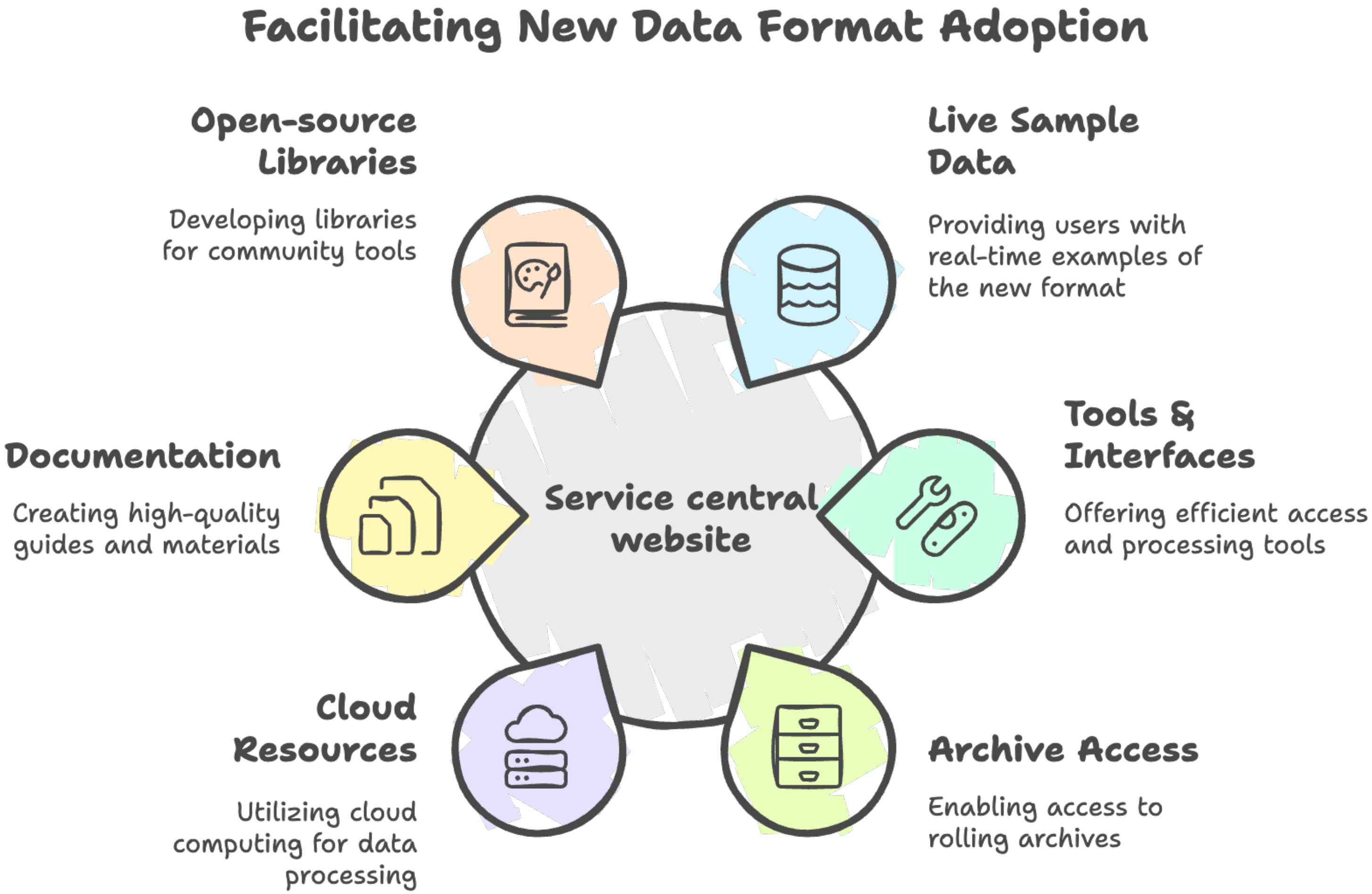
Sample Zarr Service & Other supporting activities

DDGS defintion, implementaiton; initial service in the frame of DestinE

OS Data processors implementation, operational validation, maintenance

Parallel availability of legacy and new data flow to support operational transition and risk management, before interruption of legacy processors and legacy data flow

Sentinels EOPF Sample Service



Facilitating New Data Format Adoption

Service central website

Open-source Libraries

Developing libraries for community tools

Live Sample Data

Providing users with real-time examples of the new format

Documentation

Creating high-quality guides and materials

Tools & Interfaces

Offering efficient access and processing tools

Cloud Resources

Utilizing cloud computing for data processing

Archive Access

Enabling access to rolling archives

EOPF

SENTINEL ZARR SAMPLES

Collaborative Ground Segment Workshop
Lisbon – 05.03.2025

Christoph Reimer - christoph.reimer@eodc.eu

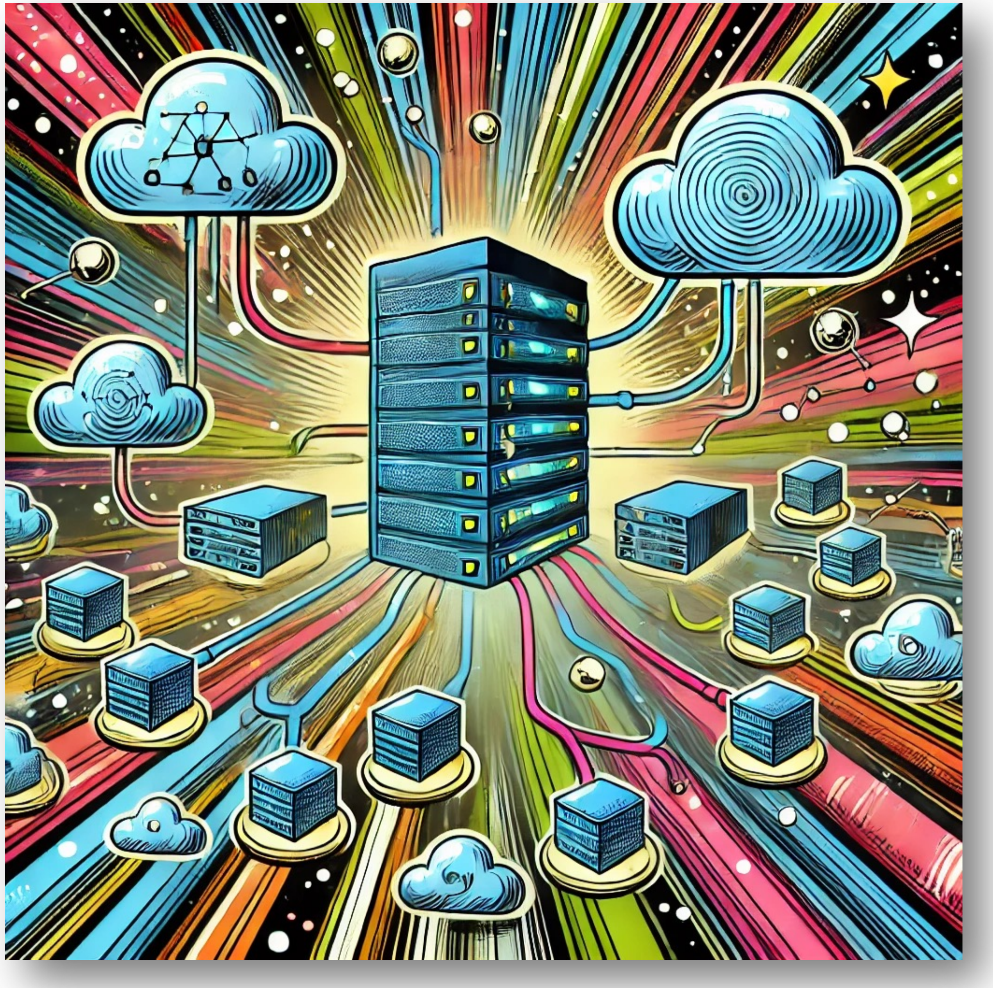
Setting the scene

- SAFE – the Standard Archive Format for Europe
  - Each mission data provided in different data formats: GeoTIFF, NetCDF or JPEG2000
- CSC Data Processors Re-engineering initiative by ESA
  - EO Processing Framework (EOPF) to support re-engineering of operational L0-L2 data processors of the Copernicus Sentinel missions (Sentinel-1, Sentinel-2 and Sentinel-3 [Land])
- EOPF **Core Python Modules (CPM)** library provides all tools to build processors based on a **harmonised data model** for all missions

https://cpm.pages.eopf.copernicus.eu/eopf-cpm/main/index.html#

Paradigm shift – data proximate computing

API first approach for
data discovery and access



Selective data transfer:
transfer only relevant data

A new storage format

- Shift towards **cloud-native** and interoperable solutions to enhance data **accessibility, scalability and integration** with modern processing frameworks
- EOPF CPM addresses this by utilising
  - Dask for parallel and distributed computing, and
  - Zarr as storage format of choice for large multi-dimensional arrays
- Zarr in combination with cloud object storage provides
  - Efficient storage and access (chunking, compression)
  - Parallel access (read and write)
  - Lazy loading and partial reads of data (get only data you need)
  - Stores metadata alongside data in a structured format (catalogue)

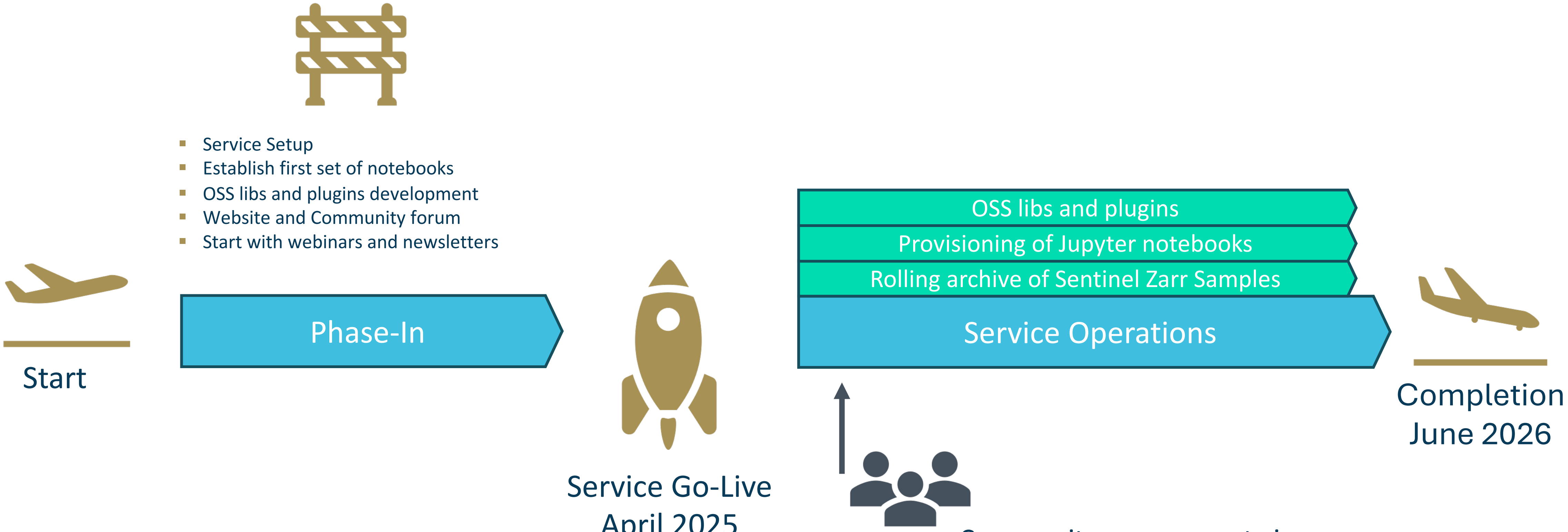
Sample Service: High-level Objective

The early and smooth adoption of the new data format by providing access to "live" sample data in the new format, including tools and interfaces to showcase the efficient access and processing of data subsets and time series.

- Three workstreams to address this:
  - Establish a rolling archive of EOPF Zarr sample data
  - Development of various OSS libraries and plug-ins
  - User community engagement

esa EOPF SENTINEL ZARR SAMPLES

Timeline



- Service Setup
- Establish first set of notebooks
- OSS libs and plugins development
- Website and Community forum
- Start with webinars and newsletters

Start

Phase-In

Service Go-Live
April 2025

OSS libs and plugins

Provisioning of Jupyter notebooks

Rolling archive of Sentinel Zarr Samples

Service Operations

Completion
June 2026

Community engagement via
- Webinars
- Newsletters
- User Support Forum
- Community building

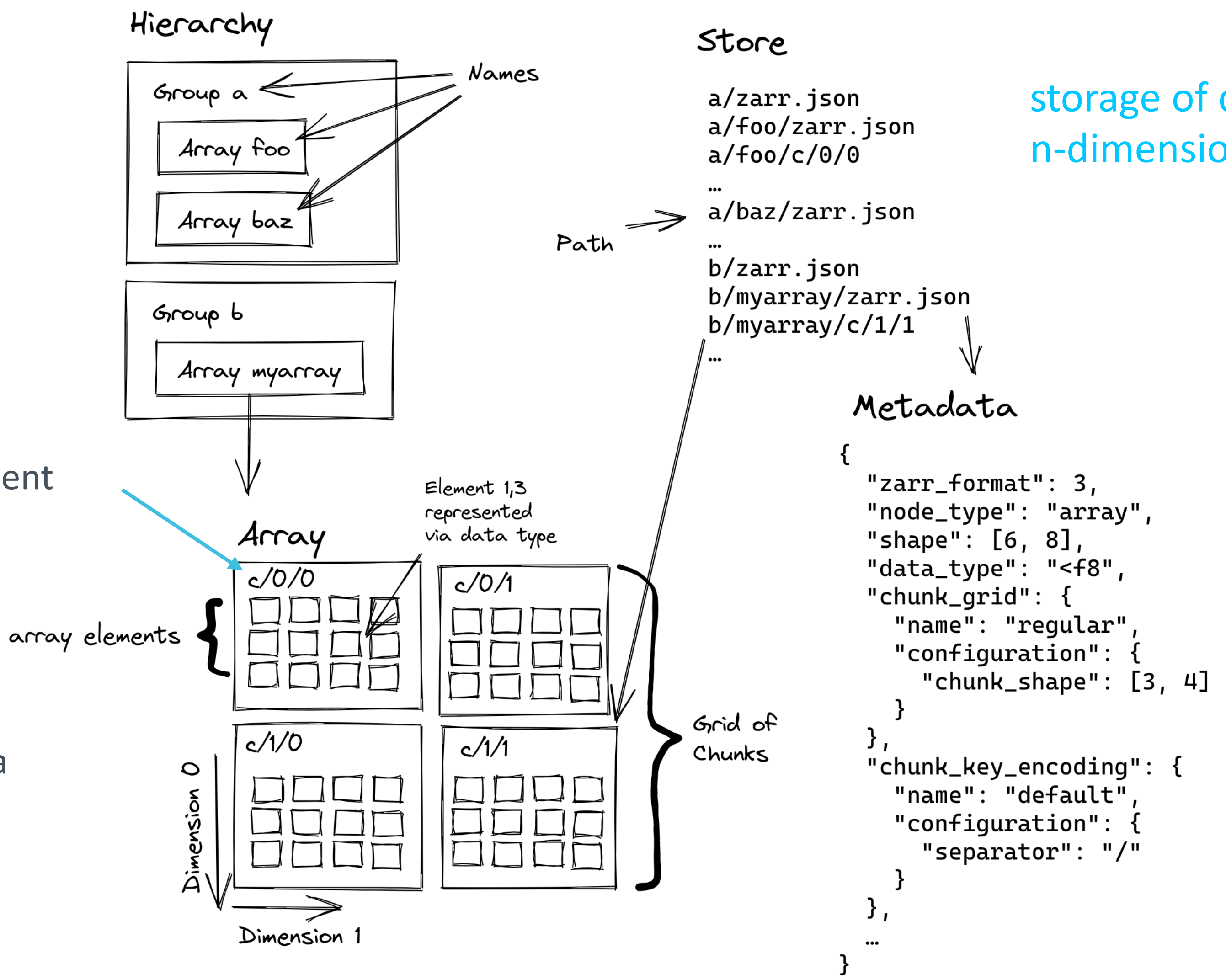
Zarr Spec.– Concept and Terminology

storage of chunked, compressed, n-dimensional typed arrays

Key to array element

binary data

JSON Metadata

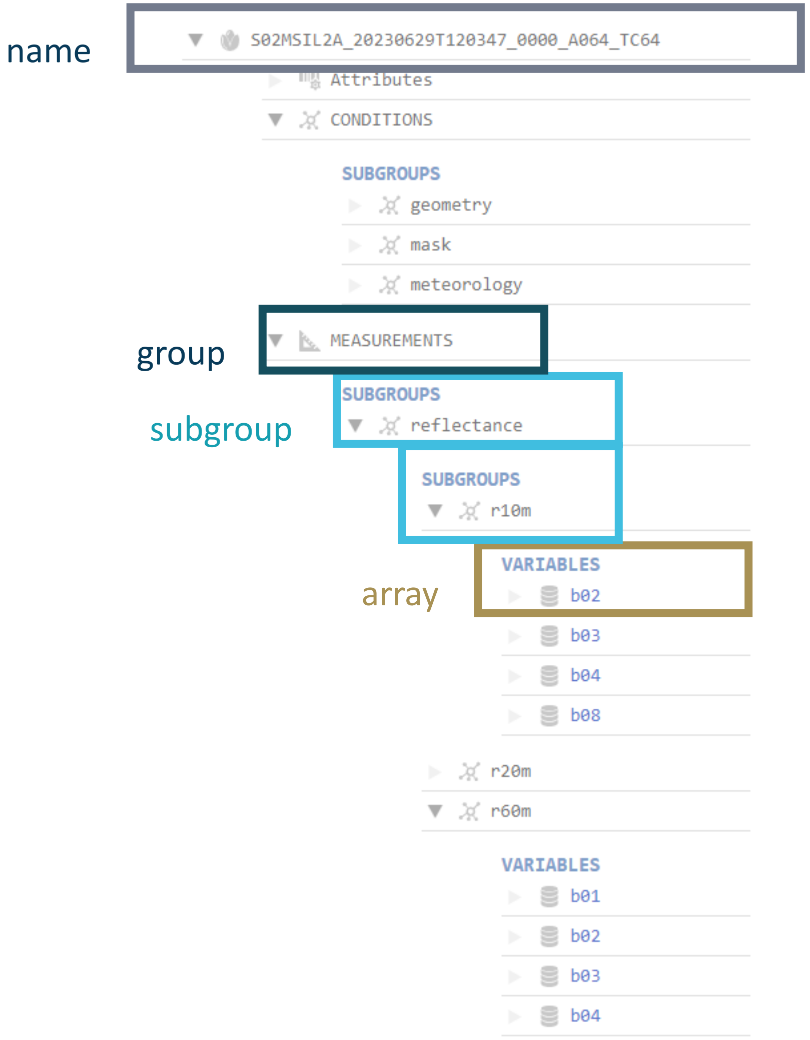


```
a/zarr.json
a/foo/zarr.json
a/foo/c/0/0
…
a/baz/zarr.json
…
b/zarr.json
b/myarray/zarr.json
b/myarray/c/1/1
…
```

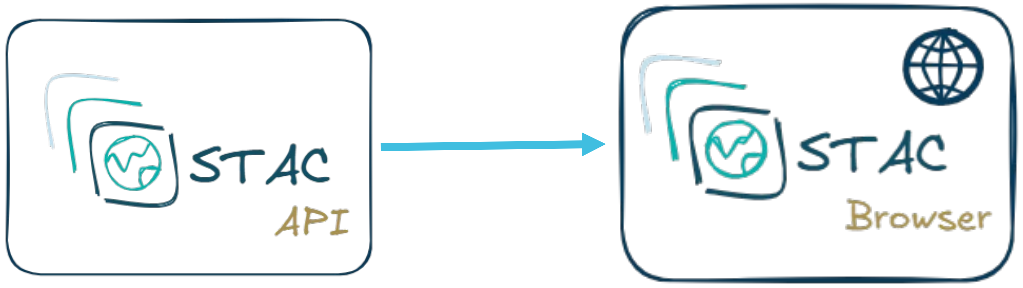
```
{
  "zarr_format": 3,
  "node_type": "array",
  "shape": [6, 8],
  "data_type": "<f8",
  "chunk_grid": {
    "name": "regular",
    "configuration": {
      "chunk_shape": [3, 4]
    }
  },
  "chunk_key_encoding": {
    "name": "default",
    "configuration": {
      "separator": "/"
    }
  },
  …
}
```

EOPF Zarr – How does it look like?

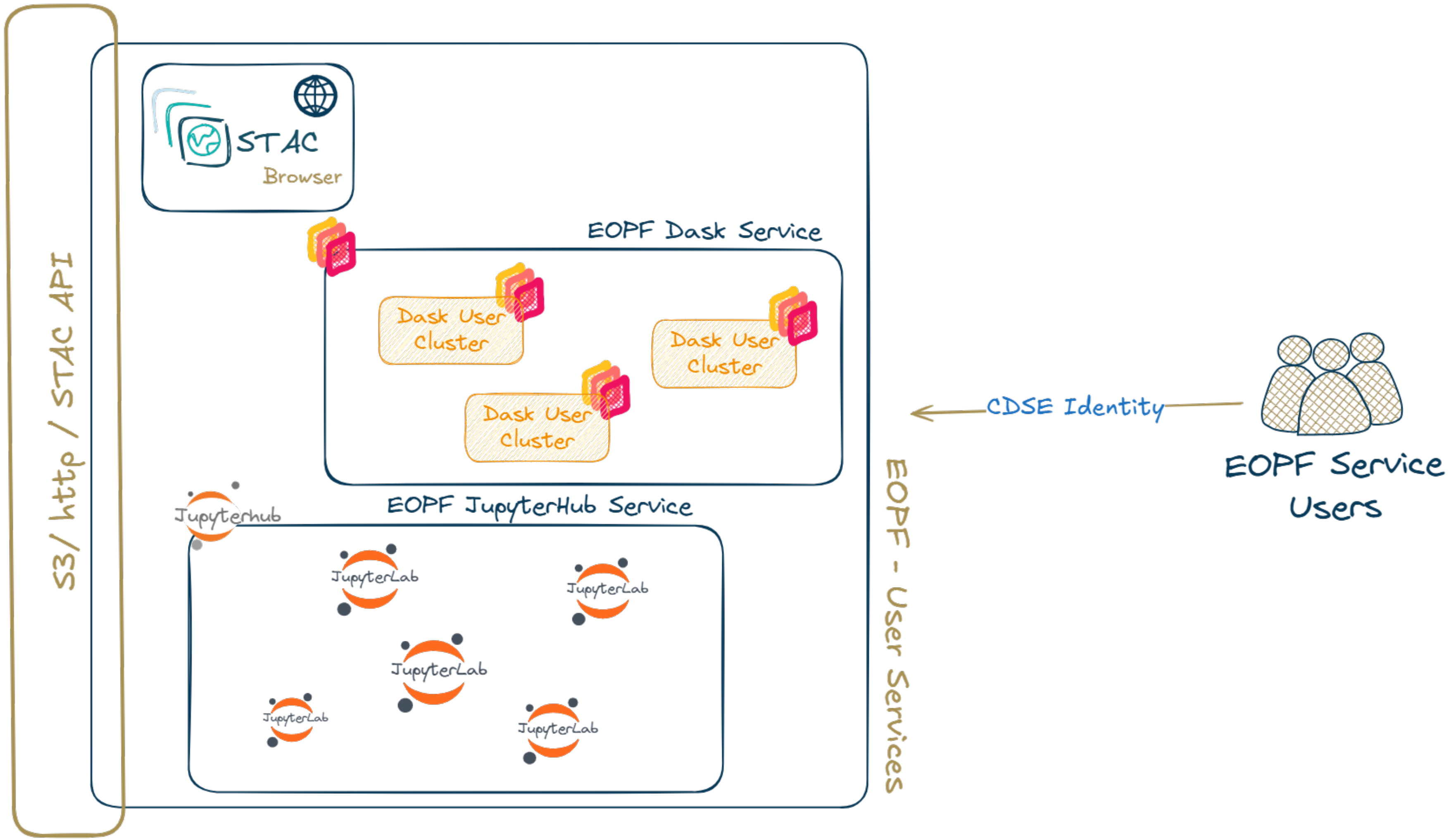
- Sentinel-2 L2A Example
  - Three main groups
    - CONDITIONS
    - MEASUREMENTS
    - QUALITY
  - Subgroups hold the actual data as arrays
- EOPF Zarr catalogued via STAC



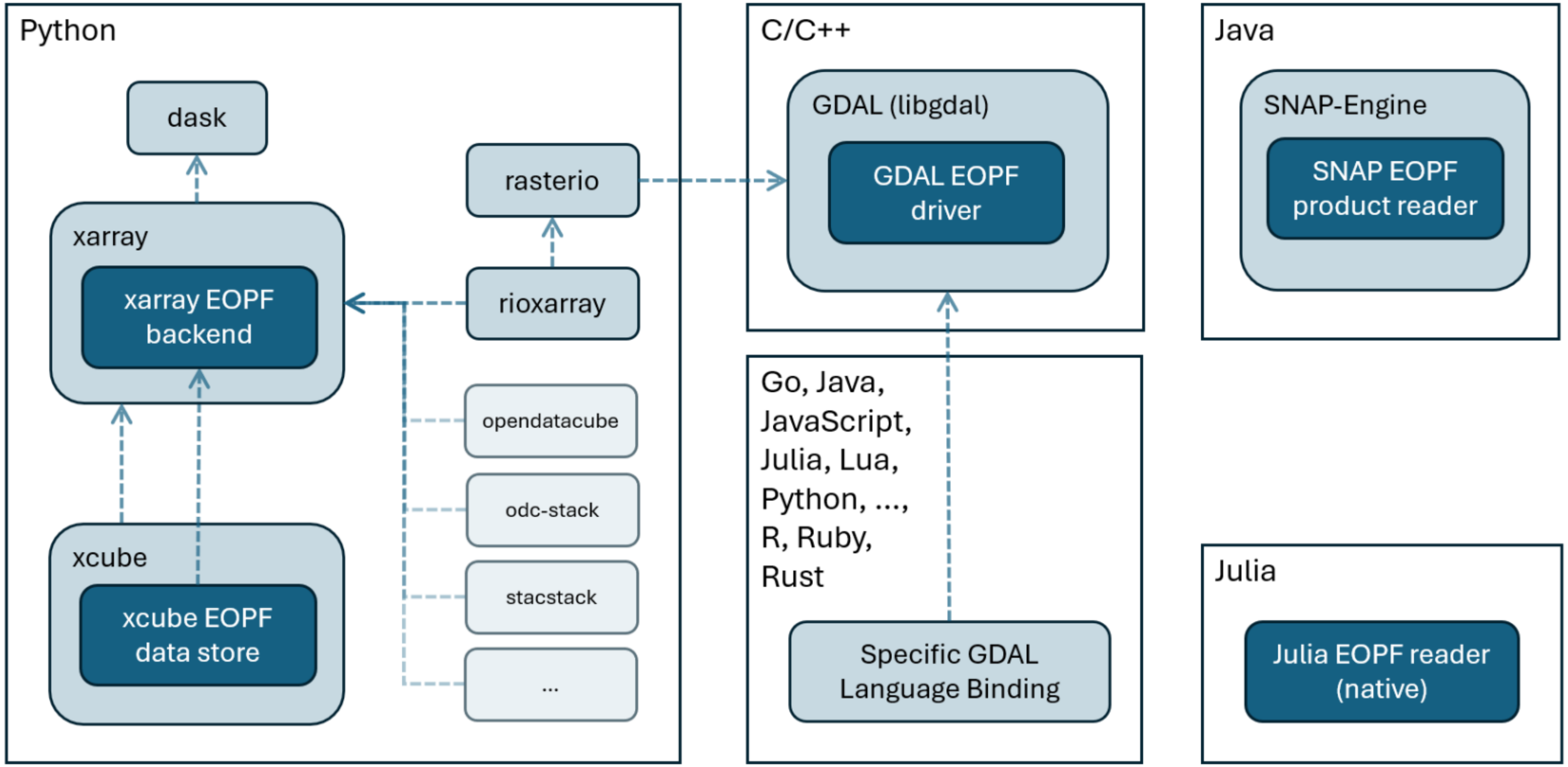
```
▼ S02MSIL2A_20230629T120347_0000_A064_TC64
  ▶ Attributes
  ▼ CONDITIONS
      SUBGROUPS
      ▶ geometry
      ▶ mask
      ▶ meteorology
  ▼ MEASUREMENTS
      SUBGROUPS
      ▼ reflectance
          SUBGROUPS
          ▼ r10m
              VARIABLES
              ▶ b02
              ▶ b03
              ▶ b04
              ▶ b08
          ▶ r20m
          ▼ r60m
              VARIABLES
              ▶ b01
              ▶ b02
              ▶ b03
              ▶ b04
```



Access to EOPF Zarr Samples



OSS libs and plug-ins Landscape



Python

- dask
- xarray
  - xarray EOPF backend
- xcube
  - xcube EOPF data store
- rasterio
- rioxarray
- opendatacube
- odc-stack
- stacstack
- ...

C/C++

- GDAL (libgdal)
  - GDAL EOPF driver

Go, Java, JavaScript, Julia, Lua, Python, ..., R, Ruby, Rust

- Specific GDAL Language Binding

Java

- SNAP-Engine
  - SNAP EOPF product reader

Julia

- Julia EOPF reader (native)

esa SNAP

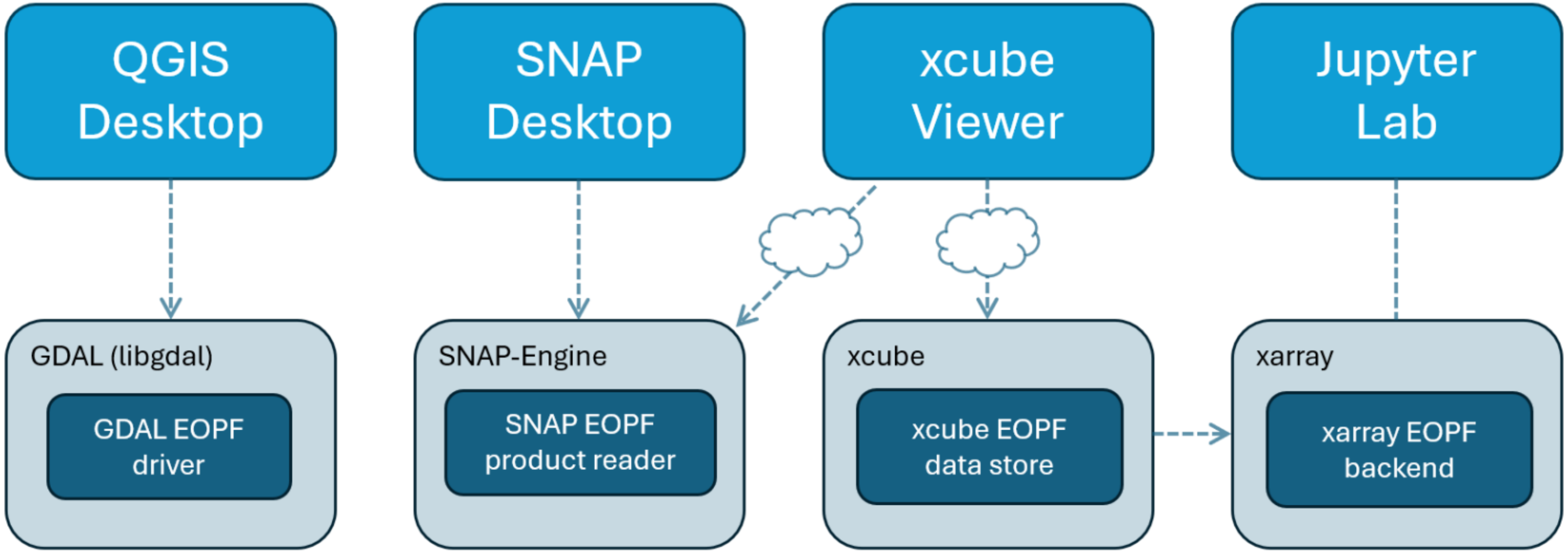
Communities

PANGEO

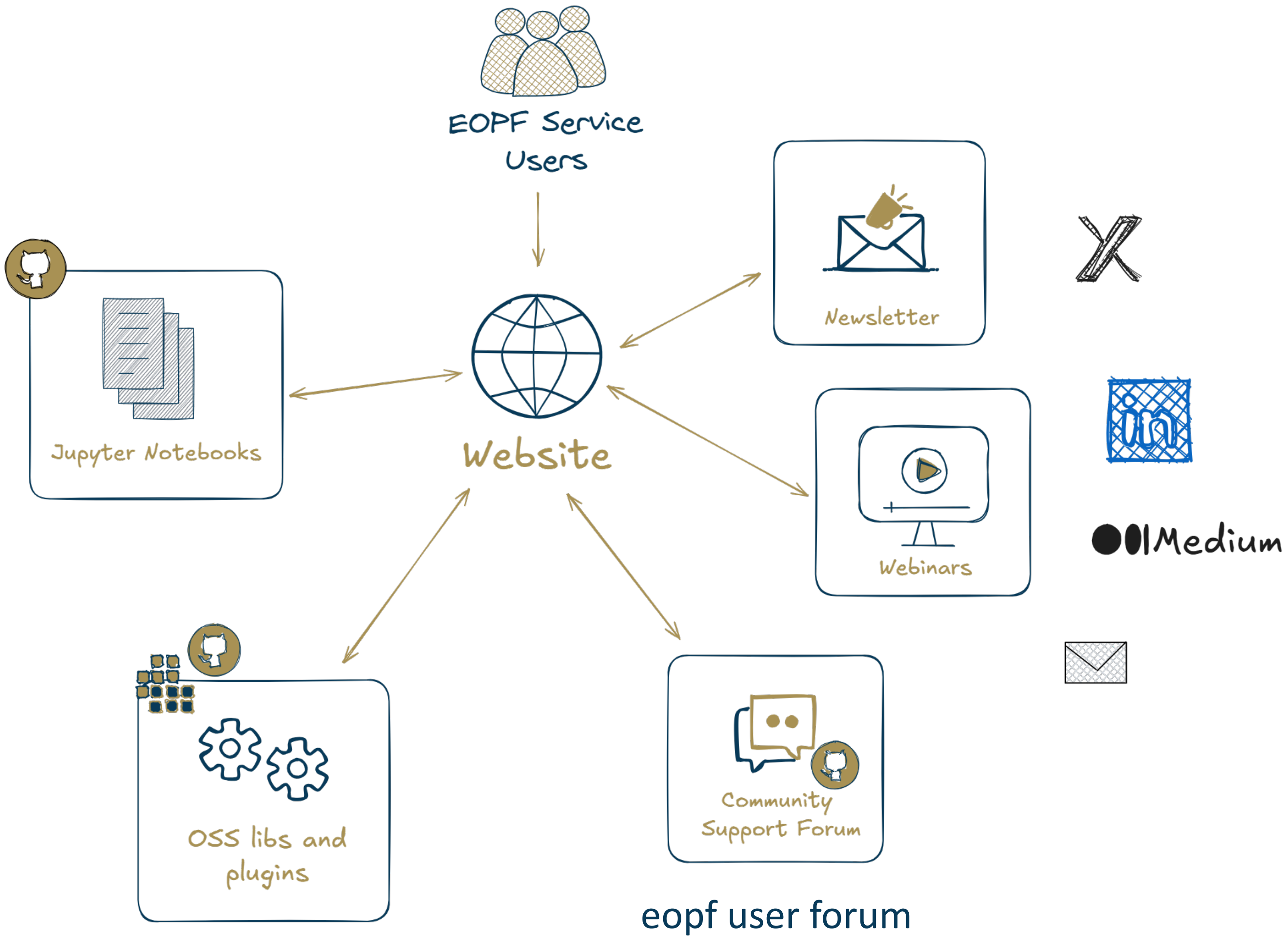
GDAL

julia

OSS plug-ins visualisation capabilities



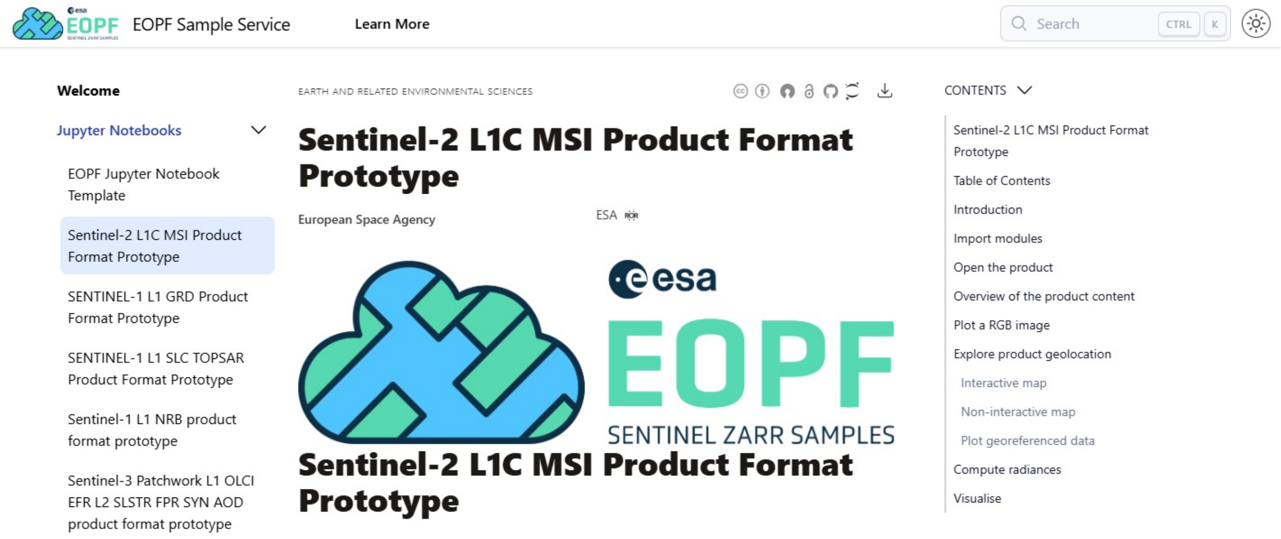
Community engagement



eopf user forum

Jupyter Notebooks

- Jupyter Notebooks will be provided showcasing the use of the new EOPF Zarr data format
  - https://eopf-sample-service.github.io/eopf-sample-notebooks/
- Roadmap to continuously extend notebooks for various user communities and thematic topics



EOPF Sample Service   Learn More   Search CTRL K

Welcome

Jupyter Notebooks

- EOPF Jupyter Notebook Template
- Sentinel-2 L1C MSI Product Format Prototype
- SENTINEL-1 L1 GRD Product Format Prototype
- SENTINEL-1 L1 SLC TOPSAR Product Format Prototype
- Sentinel-1 L1 NRB product format prototype
- Sentinel-3 Patchwork L1 OLCI EFR L2 SLSTR FPR SYN AOD product format prototype

EARTH AND RELATED ENVIRONMENTAL SCIENCES

Sentinel-2 L1C MSI Product Format Prototype

European Space Agency   ESA

Sentinel-2 L1C MSI Product Format Prototype

CONTENTS

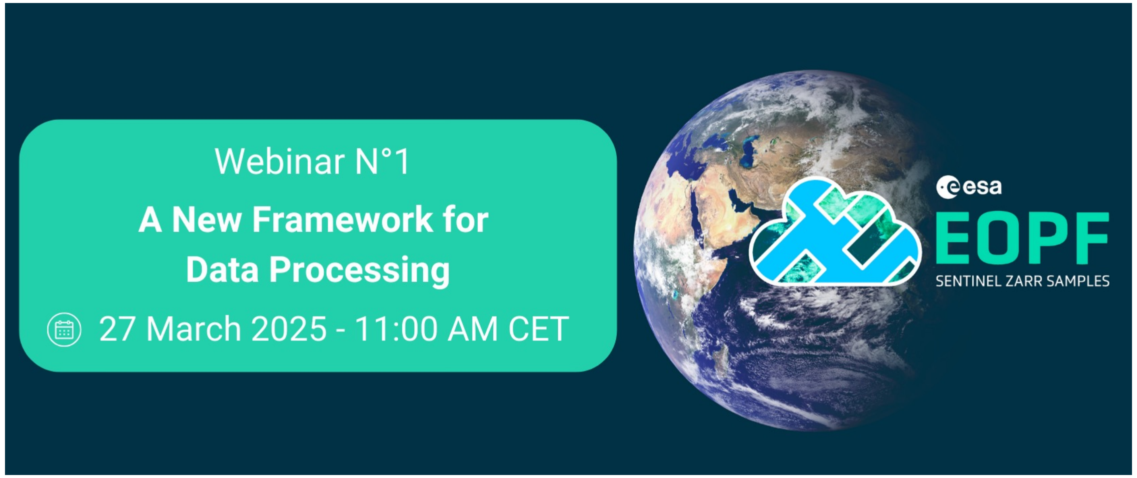
- Sentinel-2 L1C MSI Product Format Prototype
- Table of Contents
- Introduction
- Import modules
- Open the product
- Overview of the product content
- Plot a RGB image
- Explore product geolocation
  - Interactive map
  - Non-interactive map
  - Plot georeferenced data
- Compute radiances
- Visualise

GitHub Organisation

https://github.com/EOPF-Sample-Service

Webinars and Newsletter

- Currently 9 webinars are planned, focusing on the EOPF Zarr and the use of it in various thematic areas
- Newsletter service to keep users informed about and any updates



Webinar N°1

A New Framework for Data Processing

27 March 2025 - 11:00 AM CET

Newsletter

Webinars

eodc DLR evenflow eurac research Brockmann Consult simula Ifremer

https://zarr.eopf.copernicus.eu/