








NFDI4Earth

Deliverable D2.5.1

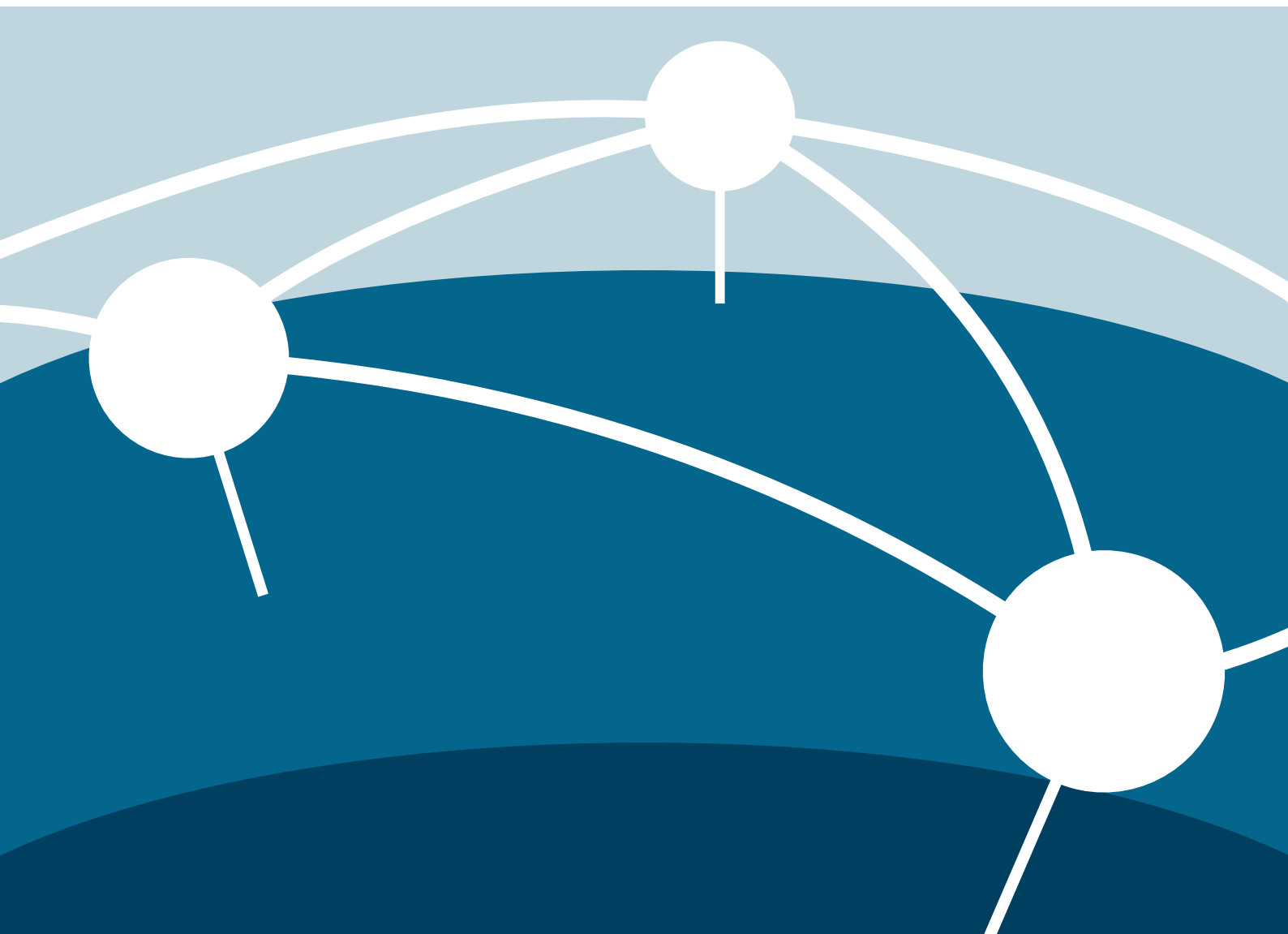
Overview of data cube technologies and review of other emerging technologies

Felix Cremer  (fcremer@bgc-jena.mpg.de), Hannes Dröse , Yomna Eid ,
Fabian Gans , Sibylle Hassler , Arne Osterthun , Edzer Pebesma 

2022-12 | Version: 1

DOI: [10.5281/zenodo.7951050](https://doi.org/10.5281/zenodo.7951050)

nfdi4earth.de



Citation

Felix Cremer, Hannes Dröse, Yomna Eid, Fabian Gans, Sibylle Hassler, Arne Osterthun, Edzer Pebesma. 2023. *Overview of data cube technologies and review of other emerging technologies (NFDI4Earth Deliverable D2.5.1)*. Zenodo. <https://doi.org/10.5281/zenodo.7951050>

License

This work is licensed under a [Creative Commons "Attribution 4.0 International"](#) license.



Acknowledgement

This work has been funded by the German Research Foundation (DFG) through the project NFDI4Earth (DFG project no. 460036893, <https://www.nfdi4earth.de/>) within the German National Research Data Infrastructure (NFDI, <https://www.nfdi.de/>).

Executive summary

In Earth System Science, large gridded data is becoming more important with large model data collections or remote sensing data. The handling of large gridded data is important in multiple steps of the research life cycle. This deliverable provides a state of the art overview of solutions for local and cloud based data storage and analysis for large gridded data. This document shall not just give a raw description of available tools but should rather help categorizing tools and show how they relate to each other.

Contents

1	Introduction	1
2	Data formats and databases	2
2.1	CF Conventions	3
2.2	Cloud Optimized Geotiffs	3
2.3	HDF5	3
2.4	NetCDF	3
2.5	TileDB	4
2.6	Zarr	4
3	Analysis software	4
3.1	EarthDataLab.jl	4
3.2	Rasters.jl	5
3.3	Rasdaman	5
3.4	OpenEO	5
3.5	xarray (Python package)	6
3.6	xcube (Python package)	6
3.7	Iris (Python package)	6
3.8	Open Data Cube	7
3.9	stars (R package)	7
3.10	terra (R package)	7
3.11	gdalcubes (R package)	7
3.12	rstac (R package)	7
4	Cloud solutions	8
4.1	Google Earth Engine	8
4.2	DIAS	8
4.3	EOSC	8
4.4	Open Data Cube Instances	9
4.5	openEO Platform	9
4.6	Pangeo	9
5	Miscellaneous Technologies	9
5.1	Spatio Temporal Asset Catalog (STAC)	9
5.2	Interplanetary File System	10
6	Summary	10

1 Introduction

The purpose of this document is to provide the state of the art solutions for local and cloud based data storage and analysis for large gridded data. In this document we are going to describe software that is currently used by scientists for the analysis of geospatial raster data. This is going to be a living document that is updated at least on an annual basis and other emerging technologies are going to be added in the future. When looking at the diversity of available tools, data formats and services, an end user might quickly be overwhelmed and confused. This document shall not just give a raw description of available tools but should rather help categorizing tools and show how they relate to each other and we will try to use a variety of “angles” to look at the tools:

Geospatial raster data is used in very different user scenarios which can be interpreted as points of view of the comparison of the different raster analysis tools. The size of the data and available computing resources play a big role in the choice of the correct tool. The computing resources can go from a single laptop, to a local cluster where the processing needs to be multi threaded or distributed up to cloud computing environments which are composed of multiple clusters. On their own laptop, scientists have full control over their computing environment but they would have to install software themselves. The software for laptops is mainly doing computations single threaded and there is a higher level of interactivity. In High Performance Computing (HPC) environments the software runs mainly multi-threaded or multi-core and the software is usually installed by a local system administrator. The computation environment is suited for larger processing and is not as interactive as on a local computer, because computation resources need to be mitigated by a job scheduler. In a cloud computing environment like the Google Earth Engine (GEE) or the European Open Science Cloud (EOSC) users are bound to what the cloud provider allows on the platform and this can be quite closed like the GEE where only a subset of Python or Javascript can be used or open as on the EOSC where the user can use containers to run any software on the provided computing resources.

The tools that are described in this document can be used for different tasks. These tasks range from extraction of a subset of data for a given spatial and temporal extent to the visualisation of the data, to the processing of the data in batches which also includes machine learning training. These different tasks have all different software needs.

Another angle to look at is the size of the dataset. The size of the data set can be described spatially, going from a local dataset like a single satellite acquisition to a regional dataset to a global harmonised dataset. The size can also be in terms of the occupied data storage from a few megabytes to multiple petabytes.

A classification of the type of the tool and the cloud readiness, describing, how well one can work on larger scale computations with this tool is shown in Figure 1. We acknowledge that “Cloud readiness” is not a well-defined term and Cloud readiness can have different meanings

for tools belonging to different categories. For data formats a higher cloud readiness indicates, that cloud datasets data can be accessed partially, i.e. a dataset which is provided in an object store in the cloud does not need to be downloaded in full to access only a small subset of the data. For software the cloud readiness entails, that the software can semi automatically use the different computing resources in a scalable way, which are provided in the cloud as a multi-computer cluster.

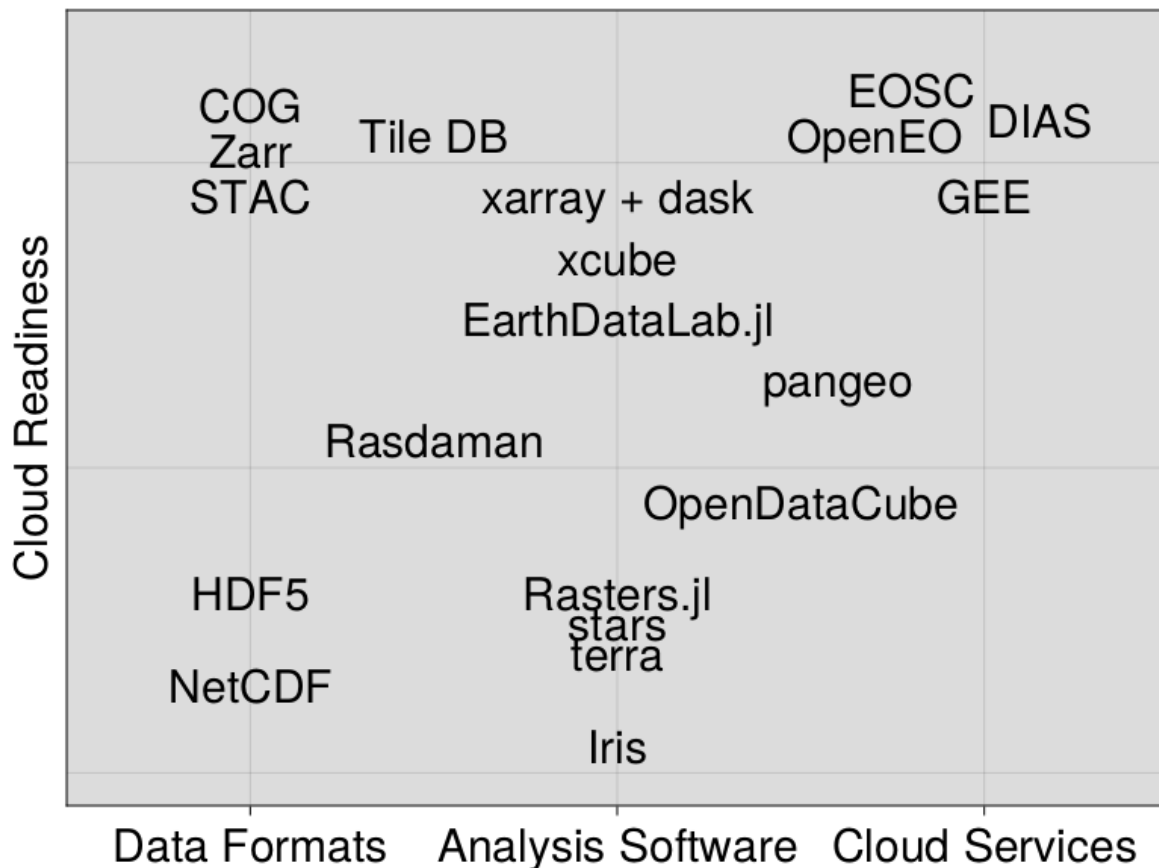


Figure 1: Overview of tools, their type and cloud-readiness. Be aware that there are some grey zones in this classification and many of the formats/tools at the lower end might be used in a cloud-context as well. However, we want to show the distinction between still widely used legacy tools that started before the advent of cloud computing and newer tools that were designed specifically with cloud computing in mind.

2 Data formats and databases

In this section we discuss data formats and data bases that are often used for the storage of large gridded geospatial data. There is in some cases an overlap between the data formats and

the accompanying software which is also doing data analysis, i.e Rasdaman and TileDB.

2.1 CF Conventions

The CF (Climate and Forecast) Metadata conventions are a set of rules to describe which metadata are necessary to describe a data set. The CF conventions have been developed on top of the NetCDF format. The aim of the CF conventions is to set the metadata to provide a definitive description of various datasets, including, what the variable represents, the spatial and temporal extent. The CF conventions include self-describing data, so that no external table for the data description is needed. CF conventions constrain the [Common Data Language \(CDL\)](#), and CF conventions and CDL are also the basis for Zarr files.

2.2 Cloud Optimized Geotiffs

Cloud optimized geotiffs (COG) are geotiff files which are organized, so that they can be hosted on a HTTP file server. This allows more efficient workflows on the cloud. COGs in contrast to plain geotiffs can be opened only partially and so it is not needed to download the whole file but you can access only the parts of the file, that are actually needed. The cloud optimized geotiff files can be used like normal tiff files and therefore this format is supported by many libraries and software solutions.

Example:

- [Sentinel-2 L2A COGS on AWS](#)

2.3 HDF5

HDF5 (Hierarchical Data Format) is a widely used data format for storing n-dimensional array in all kinds of scientific and non-scientific applications. It supports storing multiple arrays in a single file, named dimensions, several compression codecs and array chunking. HDF5 and variants like NetCDF are widely used in HPC applications because of their support for MPI (Message Passing Interface) based distributed read and write operations.

2.4 NetCDF

NetCDF stands for Network Common Data Form and is one of the standard data formats in the geosciences. It is a binary format. The development for NetCDF started in 1988 by the University Corporation for Atmospheric Research (UCAR). A NetCDF file is self describing, which means, that it includes a header that describes the data that is included in the netcdf file. The

most common used NetCDF version is NetCDF-4 which is based on the HDF 5 format. It allows for efficient subsetting but it can not be accessed multi threaded, but supports multi-process read/write operations through mpi.

2.5 TileDB

TileDB is an open source universal storage engine for dense and sparse multidimensional arrays. In May 2017 TileDB spun out of Intel Labs and the MIT. TileDB allows parallel I/O and supports both local as well as remote storage systems (i.e. object stores, HDFS and Lustre). Data is stored in tiles/chunks. TileDB provides filters like compression and byte shuffling that can be applied on the chunk/tile level. Instead of writing the data in place, TileDB creates a new fragment with each write operation. While this idea of fragments improves latency for write operations this introduces overhead for read operations.

2.6 Zarr

Zarr is a data format for handling of large N-dimensional typed arrays. It focuses on support for distributed storage systems (i.e. object stores). It aims to provide efficient I/O for parallel computing. It can handle different chunks and there are multiple extensions of the data format for different use cases. It can handle compression. It is suitable for data sets that are larger than RAM. The computations on these datasets should be parallelizable. The conceptual data structure uses CDL and CF conventions, which it shares with NetCDF.

Examples:

- static STAC to [CMIP6 on GCS](#)
- [reading Zarr files with R](#) blog post

3 Analysis software

In this section we describe software which is used to analyse the data. These software solutions are mostly installed on a local computer by single persons. There is an overlap in different tools, which provide the same syntax to allow the analysis on a local computer as in a cloud computing environment, i.e. OpenEO, OpenDataCube.

3.1 EarthDataLab.jl

Open source Julia library for with an interface for large-scale computations on Earth Data Cubes. Provides access to a variety of file formats like Zarr, NetCDF, GeoTiff and more.

Enables large distributed computations by applying arbitrary user-defined functions to huge datasets in an efficient way by respecting the underlying chunking structure of the input datasets. Automatically supports multi-threaded and distributed computations or a mix of both, depending on the cluster architecture. It tries to mimic the data model of the xarray python library, so that dimension names form the basis of merging broadcast operations

3.2 Rasters.jl

Rasters.jl is a Julia package to work with rasterized spatial data. It can open data from GeoTiff, NetCDF and R grd files. It provides a standardised interface so that many source data types can be used with identical syntax. This way, the user does not have to deal with the specifics of spatial file types. It can be used for visualisation of the data and for the analysis of the data along one given dimension.

3.3 Rasdaman

Rasdaman is an Array / Datacube Database System, which pioneered array databases. Rasdaman offers an SQL-like query language called rasql for data definition, retrieval and manipulation. Rasql embeds into standard SQL, sets and implements the ISO 9075 SQL Part 15 for Multi-Dimensional Arrays. It is set up as a client server system. Datasets have to be imported to the server with rasql. There are different client implementations from Rasdaman: a command-line utility, a Python and web client. All of them offer access to the server through rasql. Other third-party clients for different languages and use cases are available as well. Rasdaman has a commercial version, as well as an open source version with [limited capability](#).

3.4 OpenEO

[OpenEO](#) is an application programming interface (API) developed by its namesake Horizon2020 project, providing a language neutral connection between clients and back-ends. The three official client libraries currently available are in R, Python, and JavaScript (for web developers, and used by the [graphical web editor](#)). It allows users to by-pass the issue of using language-specific and data-provider-specific (cloud back-end) APIs, instead using a unified and simple API schema to connect to back-end data providers and their services. This allows easy comparison between back-ends, in terms of compatibility, capability and cost, as well as combining the multiple sources in a joint analysis.

It is also available as a downloadable [QGIS-plugin](#), so that users can explore the available openEO back-ends, as well as make further analyses and visualisation steps in the QGIS environment.

OpenEO was started as an open science alternative to closed source platforms, is led by a project steering committee, and has support from several larger European institutes who use it in production.

3.5 xarray (Python package)

Xarray is a Python package that provides an array type with labels and dimension names on top of a NumPy array. Dimensions, coordinates and attributes gives a more intuitive, more concise and less error-prone developer experience. Xarray allows to apply operations along dimension names, select values of the array based on the label and not only on the integer positions. The mathematical operations are broadcasted across multiple dimensions not based on the array shape, but based on the array labels. You can keep track of metadata as a python dictionary. You do not need to keep track of the order of the arrays and you do not need to align dimensions with added dimensions of length one.

3.6 xcube (Python package)

xcube is a Python toolkit which provides Earth Observation data in an analysis-ready form. xcube converts EO data sources into self-contained data cubes which can then be published on the cloud. The main data model in the xcube package is an xcube dataset. A dataset can contain multiple data variables whose values are stored in a common multi dimensional spatio temporal grid. These datasets follow the CF conventions. A xcube dataset is represented in memory as an xarray Dataset instance. xcube uses chunking of the dataset to allow of out-of-core computations. The chunking of the dataset has a substantial impact on the processing performance of a workflow and xcube provides capabilities to rechunk the xcube dataset. xcube uses the zarr data format for the on disk representation of the data.

3.7 Iris (Python package)

Iris is a python package for the analysis and visualisation of Earth science data. Its data model is based on the [CF Conventions]. The visualisation is based on matplotlib and cartopy. The data formats that can be used with Iris are: NetCDF, GRIB and PP. It also has a plugin system available to include other formats. The Iris package builds upon numpy and dask. It interoperates with the wider python ecosystem by using the standard numpy dask array types as underlying data storage.

3.8 Open Data Cube

The Open Data Cube (ODC) is a Python based geospatial data management and analysis software. ODC can be used to catalogue large amounts of data and to provide them as a Python API for analysis. It is mainly used for the analysis of Earth observation data in the framework of regional or national data cube platforms.

3.9 stars (R package)

R package `stars` is a package for handling and analysing raster and vector data cubes, with arbitrary amount of dimensions. It uses the GDAL C++ Multidimensional Array interface (through R package `sf`) to read and write raster or vector data cubes, or sections of them (slices, limiting dimension ranges, or strided reads: reading dimensions at lower resolution)

3.10 terra (R package)

`terra` is an R package for raster and vector data. It has its own classes for raster layers, raster stacks, and vector geometries with attributes. It is built for efficiency, scales to large rasters, and has a wide range of raster and vector algorithms available. It interfaces with R packages `sf` and `stars` (conversion of objects, both ways). It does not handle raster datacubes with more than three dimensions, or vector data cubes.

3.11 gdalcubes (R package)

R package `gdalcubes` is an R package that creates a (regular) datacube from an image collection, by specifying the cube dimensions (extent, resolution, coordinate reference system) and spatial and/or temporal aggregation method(s). It can use arbitrary R function for this aggregation, and write the datacube as a NetCDF file or return it as a `stars` object. For finding individual images contributing to an image collection it can use a STAC (through `rstac`, see below), as illustrated in [this blog post](#).

3.12 rstac (R package)

R package `rstac` provides a programmatic (R) interface to a STAC API catalog. It was developed as part of the [Brazil Data Cube](#) project.

4 Cloud solutions

In this section we describe Cloud computing environments and tools for cloud computing environments which are used to scale the analysis which might be prepared and tested on a smaller setup on a local computer to a larger computational platform to enable the analysis of very large data sets which can not be downloaded locally.

4.1 Google Earth Engine

Google Earth Engine (GEE) is a cloud platform for the analysis of large geospatial datasets. It provides easy access to the Landsat data, the Sentinel data and other different datasets. It is currently free, because Google sees it more as a marketing tool. The algorithms that you want to compute on google earth engine have to be written in a strange sub language of java script and you cannot bring your own already working script. We see a large problem with vendor lock in, because as soon as Google decides that the computation is not free anymore, you can't easily switch to other systems. The data that might be available on other cloud providers, but the algorithms have to be recoded, because GEE uses a special subset of python or java script. The Google Earth Engine is used for example to provide the global forest cover change map.

4.2 DIAS

The DIAS are the Data and Information Access Services in the european Copernicus programme. They provide centralized access to Copernicus data, information and processing tools. There are five DIAS platforms online, which allow the users to discover, manipulate, process and download Copernicus data and information. All DIAS platforms provide access to the Sentinel and other Copernicus datasets as well as to other additional satellite or non-space datasets. The aim of the DIAS platforms is to allow the users to build data processing on the web without having to download the data to a local computer first.

4.3 EOSC

The EOSC aims to develop a Web of FAIR Data and services for Europe, while covering the scientific cloud infrastructure available. Although at the moment of writing it is not entirely clear what this entails or will entail in the near future, a service provided by EOSC (EGI to be precise) and reused by openEO Platform is the EGI login for authentication: this can reuse all the authentication systems of European universities, as well as openID systems like GitHub or Google.

4.4 Open Data Cube Instances

Open Data Cube is deployed for several larger (national/continental) Earth observation data cubes, notably the [Australian Geoscience Data Cube](#), the [African Regional Data Cube](#) and the [Swiss Data Cube](#).

It is also used as back-end infrastructure for openEO instances, for instances operated by EURAC Research (internal) and EODC (public, part of the openEO Platform federated cube)

4.5 openEO Platform

To provide broad accessibility to the openEO API for users, OpenEO Platform, found at [openEO.cloud](#) provides compute access on large scale Earth observation data collections, including Copernicus and Landsat data. Users can create a 30-day trial account to explore or process available data collections using code-scripts or in a user-friendly graphical process workflow with pre-defined, as well as user-defined, functions. User-defined functions can be written in Python or R, and are executed on the pixels at the back-end. The openEO.cloud aggregator passes requests on to one of four back-ends: VITO, EODC, Sentinel-Hub, and CreoDIAS, depending on which collections and processes are requested. Cloud resources are sponsored by the ESA NoR (network of resources).

4.6 Pangeo

Pangeo is a community of like-minded people which collaboratively develop software and infrastructure for Big Data geoscience research. The Pangeo community also promotes a software stack of interconnected python packages including xarray and Iris. They also deploy this software stack as a Pangeo environment in different cloud computing facilities.

5 Miscellaneous Technologies

5.1 Spatio Temporal Asset Catalog (STAC)

[STAC](#) stands for *Spatiotemporal Asset Catalog* and is an (emerging) standard for describing metadata of spatiotemporal assets. Initially designed for describing large amounts of satellite imagery, where an asset is a file, it is now more and more adapted to other use cases such as SAR, point clouds, data cubes, full motion videos, and in situ (sensor) data.

STAC consists of four semi-independent specifications. From the specification website:

- **STAC Item** is the core atomic unit, representing a single spatiotemporal asset as a GeoJSON feature for the spatial extent plus datetime and links.
- **STAC Catalog** is a simple, flexible JSON file of links that provides a structure to organize and browse STAC Items. A series of best practices helps make recommendations for creating real world STAC Catalogs.
- **STAC Collection** is an extension of the STAC Catalog with additional information such as the extents, license, keywords, providers, etc that describe STAC Items that fall within the Collection.
- **STAC API** provides a RESTful endpoint that enables search of STAC Items, specified in OpenAPI, following OGC's WFS 3.

Modern software layers to interact with large amounts of satellite imagery such as Open Data Cube and openEO allow users to interact with image collections defined as STAC collections.

Example STAC:

- [Sentinel-2 L2A on AWS](#)
- static STAC to [CMIP6 on GCS](#)
- [stacindex](#) a collection of STAC catalogs

5.2 Interplanetary File System

The interplanetary file system <https://ipfs.io/> is a further development of the torrent system. It would allow to have one version of a file in a larger network and if a file is requested for download it can be retrieved from the nearest available position. IPFS uses content addressable storage to distribute the file across the network and this makes it more fault tolerant and independent of single infrastructure components.

6 Summary

As this is the first revision of this living document, the authors do admit that the list of tools presented here is far from complete. The list will be extended in the future and more data formats, tools and processing services will be added. NFDI4Earth is an open community. In case you want to suggest additional tools to be added in the future, feel free to reach out to the NFDI4Earth Measure 2.5 team (fcremer@bgc-jena.mpg.de as representative) or start a discussion directly on [Github](#).