



Destination Earth

A Notebook-Based approach to visualise data from DestinE Data portfolio



Authors: Patryk Grzybowski¹; Marcin Ziółkowski¹; Aubin Lambare²; Christoph Reimer³; Michael Schick⁴

Affiliations: ¹CloudFerro S.A., Warsaw, Poland; ²CS Group, Le Plessis Robinson, France; ³EODC, Vienna, Austria; ⁴EUMETSAT, Darmstadt, Germany

Destination Earth

Funded by
the European Union



Implemented by





STACK

- RESTAPI
- EODAG
- PySTAC
- Xcube

Istlet

- Open Data Cube

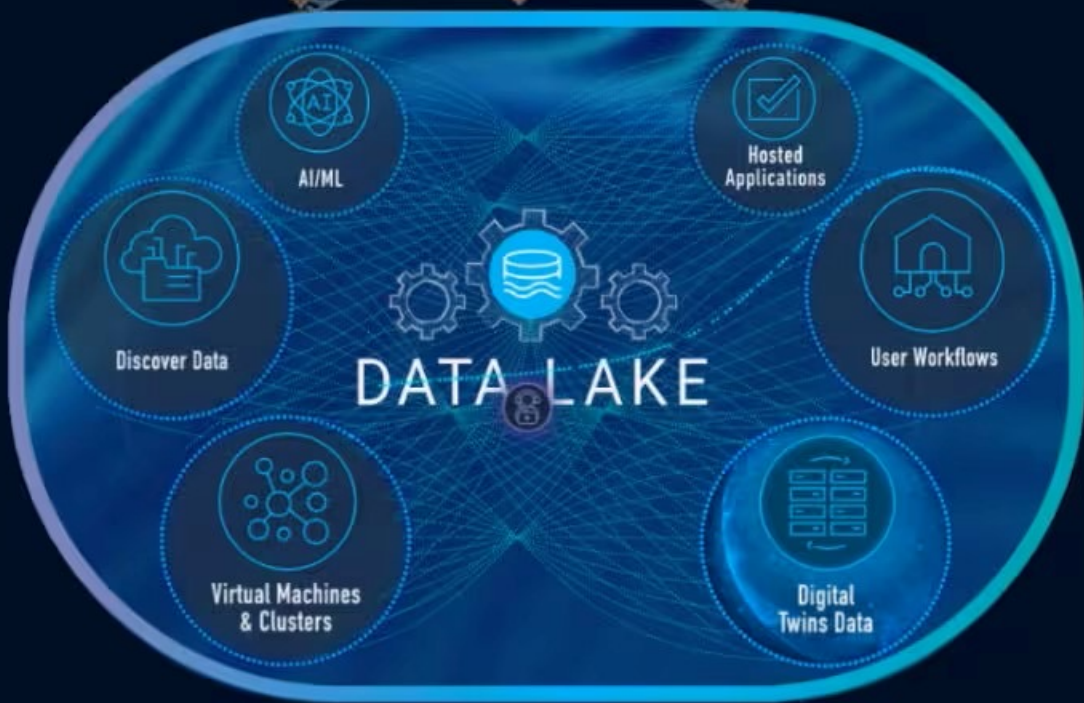


xcube





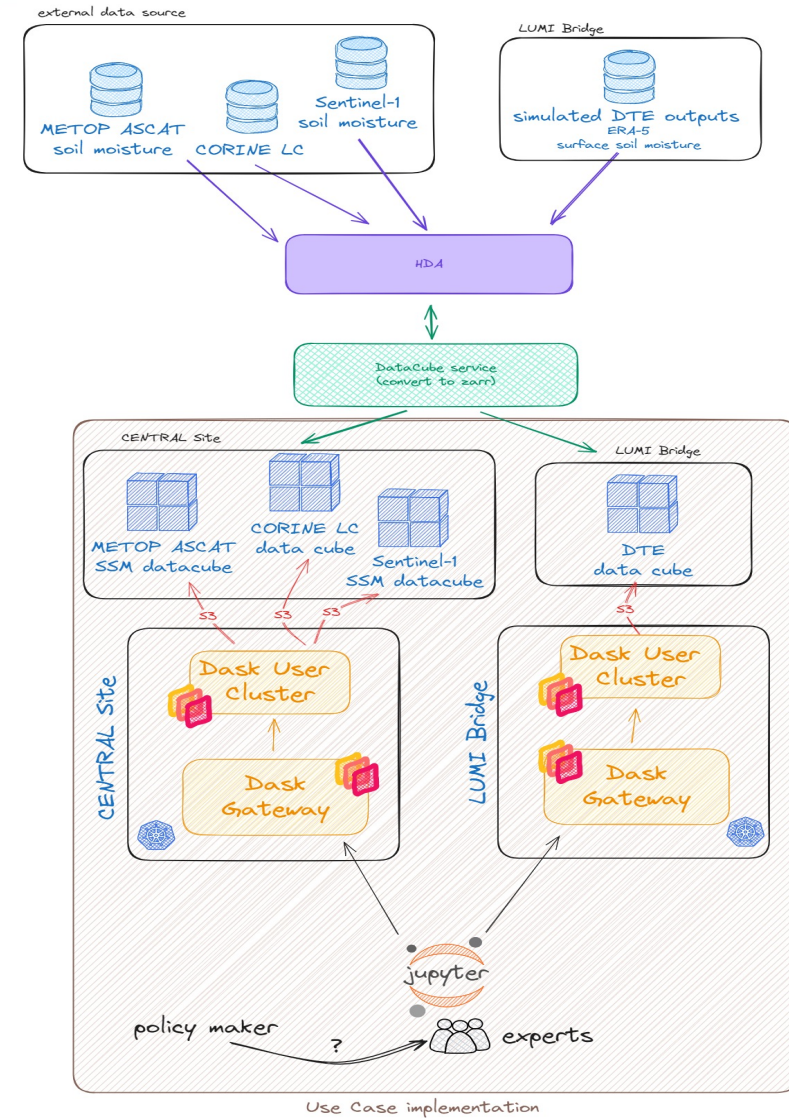
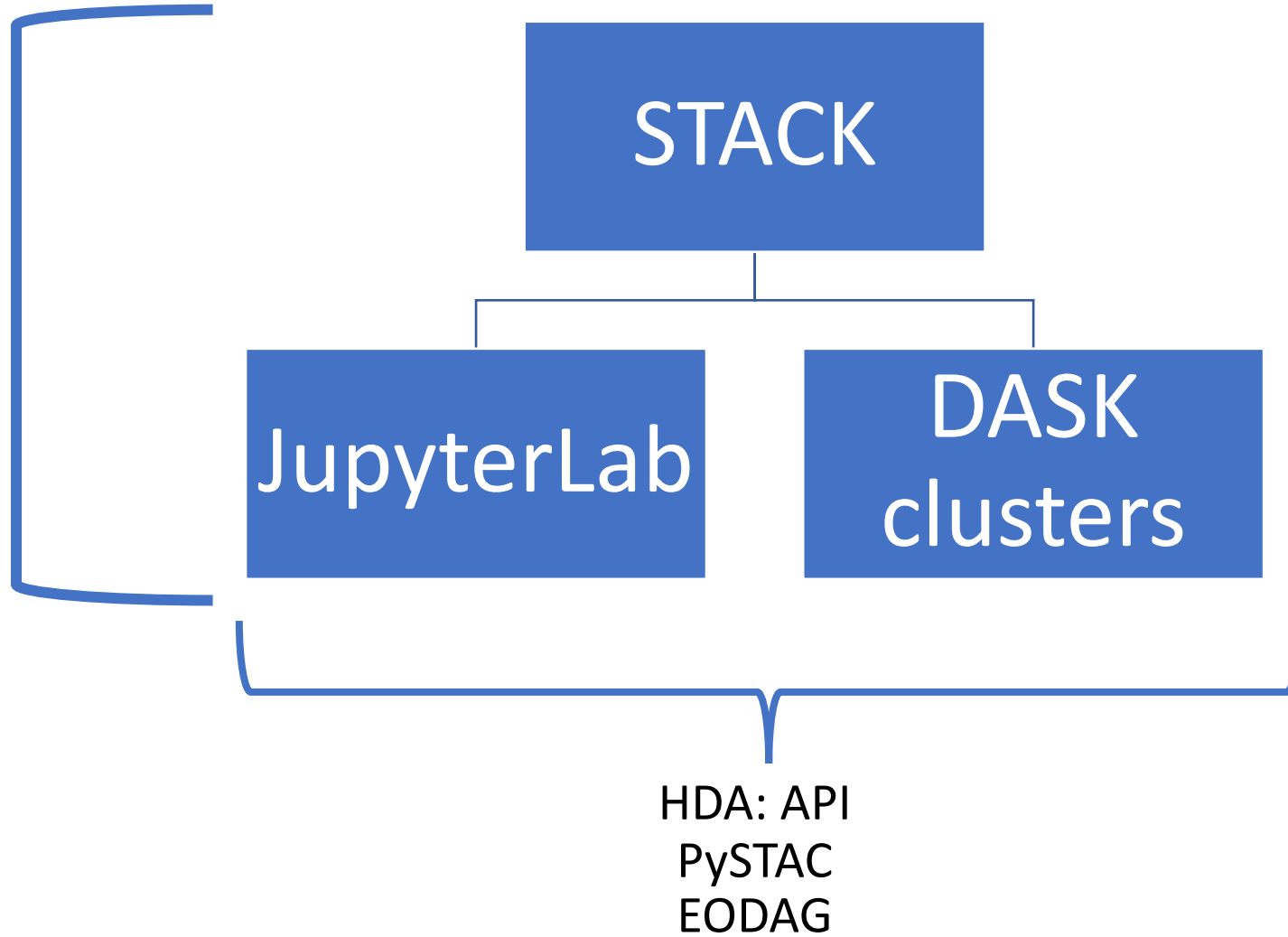
Destination Earth

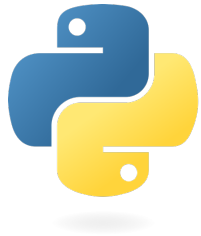


STACK



Data Bridges





```

from IPython.display import JSON
import xarray as xr
import cfgrid
import numpy as np
from tqdm import tqdm
import time
import re
import json
import requests

```

```

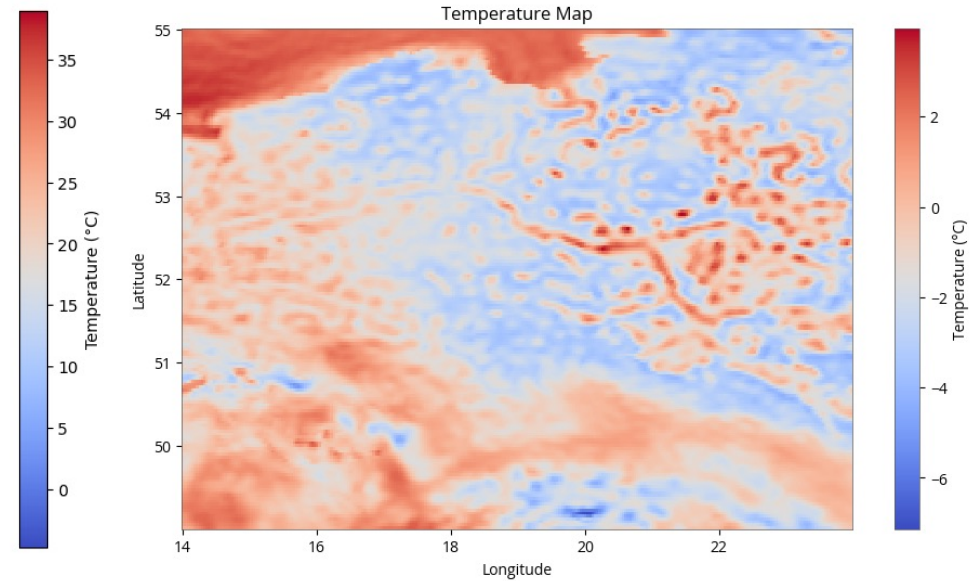
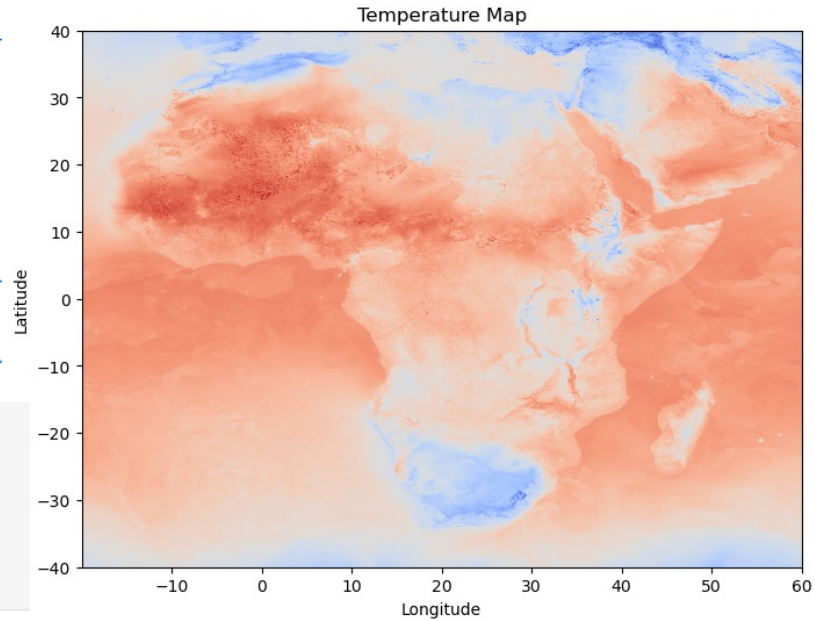
import matplotlib as plt

```

```

plt.figure(figsize=(10, 6))
plt.pcolormesh(lon, lat, temperature, cmap='coolwarm')
plt.colorbar(label='Temperature (°C)')
plt.title('Temperature Map')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

```





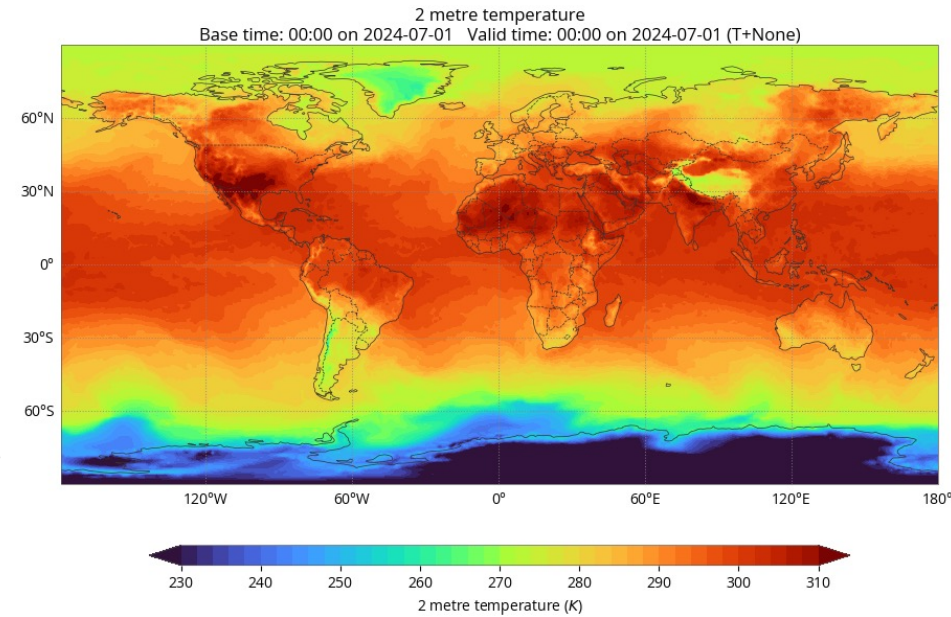
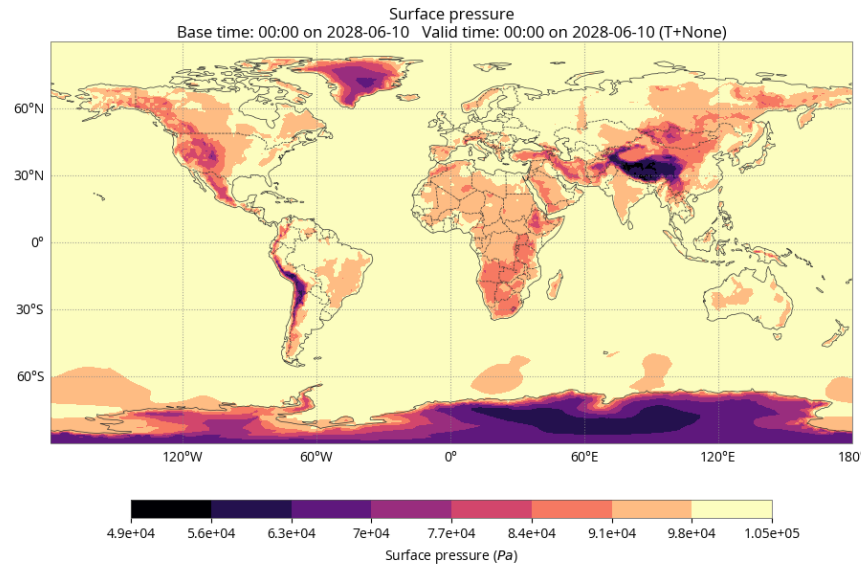
earthkit

```
import requests
import json
import os
from getpass import getpass
import destinelab as death

import importlib.metadata
```

```
import earthkit.data
import earthkit.maps
import earthkit.regrid
```

```
data = earthkit.data.from_source("file", filename)
data.ls
earthkit.maps.quickplot(data, #style=style
)
```



Extremely useful with Digital Twin outputs!





```
import rich.table

selected_item = coll_items[3]

table = rich.table.Table(title="Assets in STAC Item")
table.add_column("Asset Key", style="cyan", no_wrap=True)
table.add_column("Description")
for asset_key, asset in selected_item.assets.items():
    table.add_row(asset_key, asset.title)

console.print(table)
```

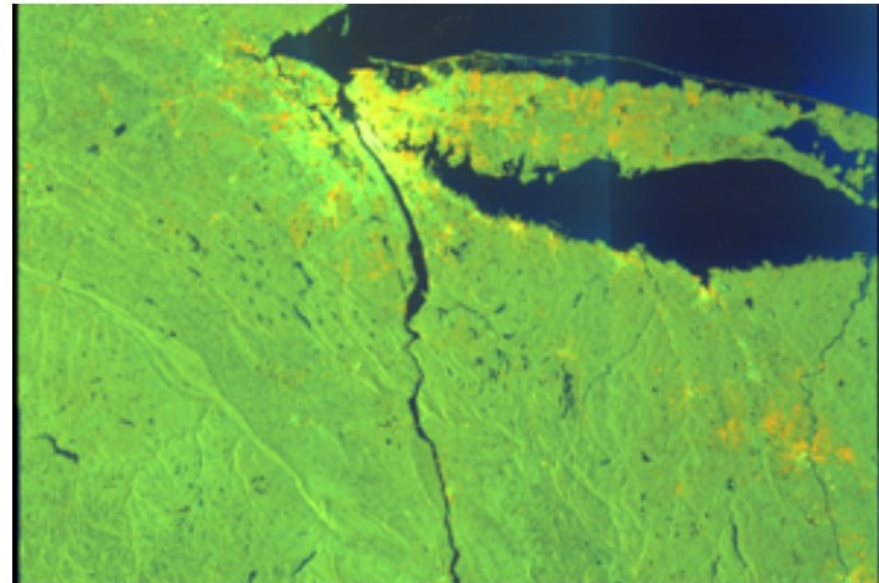
Assets in STAC Item

Asset Key	Descript...
downloadLink	Download link
S1A_IW_GRDH_1SDV_20230914T225135_20230914T225200_050330_060F40_9F54.SAFE-report-20230915T011838.pdf	S1A IW G...
calibration-s1a-iw-grd-vh-20230914t225135-20230914t225200-050330-060f40-002.xml	calibrat...
calibration-s1a-iw-grd-vv-20230914t225135-20230914t225200-050330-060f40-001.xml	calibrat...
noise-s1a-iw-grd-vh-20230914t225135-20230914t225200-050330-060f40-002.xml	noise-s1...
noise-s1a-iw-grd-vv-20230914t225135-20230914t225200-050330-060f40-001.xml	noise-s1...
rfi-s1a-iw-grd-vh-20230914t225135-20230914t225200-050330-060f40-002.xml	rfi-s1a--
rfi-s1a-iw-grd-vv-20230914t225135-20230914t225200-050330-060f40-001.xml	rfi-s1a--
s1a-iw-grd-vh-20230914t225135-20230914t225200-050330-060f40-002.xml	s1a-iw-g...
s1a-iw-grd-vv-20230914t225135-20230914t225200-050330-060f40-001.xml	s1a-iw-g...
manifest.safe	manifest...
s1a-iw-grd-vh-20230914t225135-20230914t225200-050330-060f40-002.tiff	s1a-iw-g...
s1a-iw-grd-vv-20230914t225135-20230914t225200-050330-060f40-001.tiff	s1a-iw-g...
logo.png	logo.png
map-overlay.kml	map-over...
product-preview.html	product-...
quick-look.png	quick-lo...
thumbnail.png	thumbnai...
s1-level-1-calibration.xsd	s1-level...
s1-level-1-measurement.xsd	s1-level...
s1-level-1-noise.xsd	s1-level...
s1-level-1-product.xsd	s1-level...
s1-level-1-quicklook.xsd	s1-level...
s1-level-1-rfi.xsd	s1-level...
s1-map-overlay.xsd	s1-map-o...
s1-object-types.xsd	s1-objec...
s1-product-preview.xsd	s1-produ...
thumbnail	Thumbnail

```
[16]: from IPython.display import Image
```

```
Image(url=selected_item.assets["thumbnail"].href, width=500)
```

[16]:





Destination Earth



PRODUCTS SEARCH

DestinE-DataLake-Lab/HDA/EODAG

Notebook

- Python 3 (pykernel)
- Python DEFL

Console

- Python 3 (pykernel)
- Python DEFL

Other

- Terminal
- Text File
- Markdown File
- Python File
- Show Contextual Help

Provider: Any

Product Type: S2_...

Date range: Start, End

Max cloud cover 100%

Additional Parameters

PRODUCTS SEARCH

Provider: eumetsat_ds

Product Type: METOP_AVHRR_L1

Date range: 01/10/2023, 30/09/2024

Max cloud cover 100%

Additional Parameters

Preview Results Generate Code

Search results

Start time	End time	Cloud cover	Actions
2023-10-011708:40:01Z	2023-10-011708:40:01Z		
2023-10-01109:28:03Z	2023-10-01109:28:03Z		
2023-10-01110:22:03Z	2023-10-01110:22:03Z		
2023-10-01117:49:01Z	2023-10-01117:49:01Z		
2023-10-01118:43:03Z	2023-10-01118:43:03Z		
2023-10-01119:31:03Z	2023-10-01119:31:03Z		
2023-10-02108:19:03Z	2023-10-02108:19:03Z		

Quicklook

Metadata

Platform serial identifier: METOP

Instrument: AVHRR

Product type: AVHRR_L1

Processing level: L1

Start time from ascending node: 2023-10-01108:40:03Z

Completion time from ascending node: 2023-10-01110:22:03Z

Orbit number: 25421

ID: METOP_AVHRR_L1

Abstract: The Advanced Very High Resolution Radiometer (AVHRR) operates at 5 different channels simultaneously in the visible and infrared bands, with wavelengths specified in the instrument channels description. Channel 3 switches between 3a and 3b for daytime and nighttime. As a high-resolution imager (about 1.1 km near nadir) its main purpose is to provide cloud and surface information such as cloud coverage, cloud top temperature, surface temperature over land and sea, and vegetation or snow/ice. In addition, AVHRR products serve as input for the level 2 processing of IAS and ATOVS.

Default geometry: POLYGON((-100 -90, 180 90, -180 90, -180 -90))

Download link: https://api.eumetsat.int/data/download/1.0/collections/EODAG/EUM3ADAT%3AMETOP3AAVHRR_L1/products/AVHRR_wc_10_M03_2023100104003Z_20231001102203Z_N_0_202310

Keywords: METOP_AVHRR_RADIOMETER_L1_ATMOSPHERE_OCEAN_AVHRR_L1_AV

License: proprietary

Generate code



Data Cubes

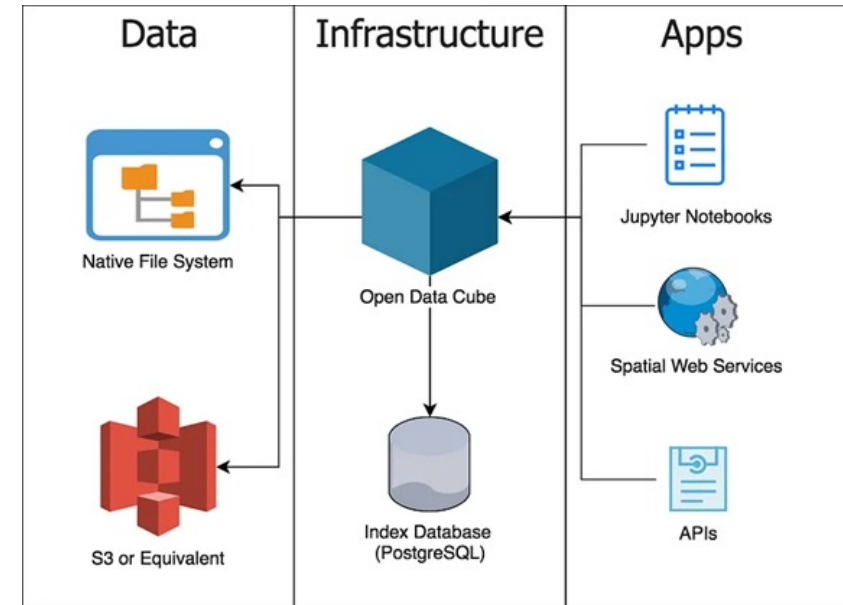
STACK/Islet/Locally

- Xarray and Xcube

```
<xarray.DataArray 'time' (time: 25)>
array(['2024-04-10T00:00:00.000000000', '2024-04-10T01:00:00.000000000',
      '2024-04-10T02:00:00.000000000', '2024-04-10T03:00:00.000000000',
      '2024-04-10T04:00:00.000000000', '2024-04-10T05:00:00.000000000',
      '2024-04-10T06:00:00.000000000', '2024-04-10T07:00:00.000000000',
      '2024-04-10T08:00:00.000000000', '2024-04-10T09:00:00.000000000',
      '2024-04-10T10:00:00.000000000', '2024-04-10T11:00:00.000000000',
      '2024-04-10T12:00:00.000000000', '2024-04-10T13:00:00.000000000',
      '2024-04-10T14:00:00.000000000', '2024-04-10T15:00:00.000000000',
      '2024-04-10T16:00:00.000000000', '2024-04-10T17:00:00.000000000',
      '2024-04-10T18:00:00.000000000', '2024-04-10T19:00:00.000000000',
      '2024-04-10T20:00:00.000000000', '2024-04-10T21:00:00.000000000',
      '2024-04-10T22:00:00.000000000', '2024-04-10T23:00:00.000000000',
      '2024-04-11T00:00:00.000000000'], dtype='datetime64[ns]')
Coordinates:
  * time      (time) datetime64[ns] 2024-04-10 2024-04-10T01:00:00 ... 2024-04-11
Attributes:
  axis:      T
  standard_name: time
```

ISLET

- Open Data Cube





```

africa_bbox = [-20, # West
               -40, # South
               60,  # East
               40] #North

africa_dt = xr.open_zarr(url)['sp'].sel(lon=slice(africa_bbox[0],
                                                africa_bbox[2]),
                                       lat=slice(africa_bbox[3],
                                                africa_bbox[1]),
                                       time=slice('20240410T000000', '20240411T000000'))

print(africa_dt.var)

```

```

<bound method DataArrayAggregations.var of <xarray.DataArray 'sp' (time: 25, lat: 2276, lon: 2279)>
dask.array<getitem, shape=(25, 2276, 2279), dtype=float32, chunks=(25, 417, 837), chunktype=numpy.ndarray>
Coordinates:
  * lat      (lat) float64 39.99 39.95 39.92 39.88 ... -39.92 -39.95 -39.99
  * lon      (lon) float64 -19.97 -19.94 -19.9 -19.87 ... 59.92 59.95 59.99
  * time     (time) datetime64[ns] 2024-04-10 2024-04-10T01:00:00 ... 2024-04-11
Attributes:
  CDI_grid_num_LPE: 2560
  CDI_grid_type:    gaussian
  long_name:        Surface pressure
  param:            0.3.0
  standard_name:    surface_air_pressure
  units:            Pa>

```

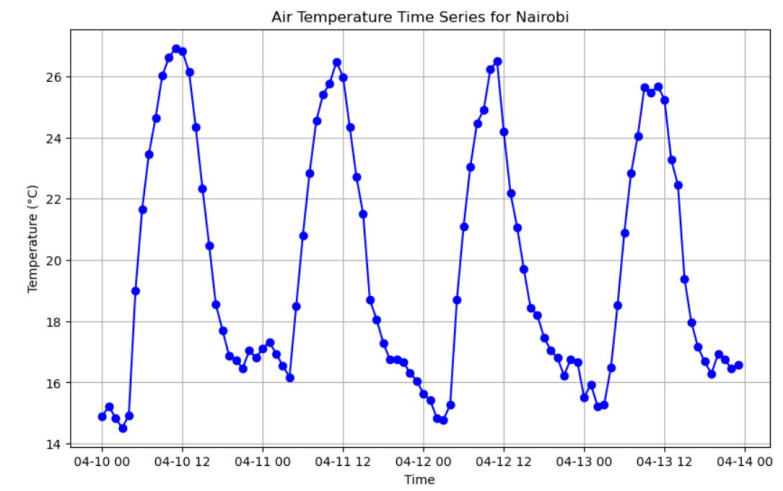
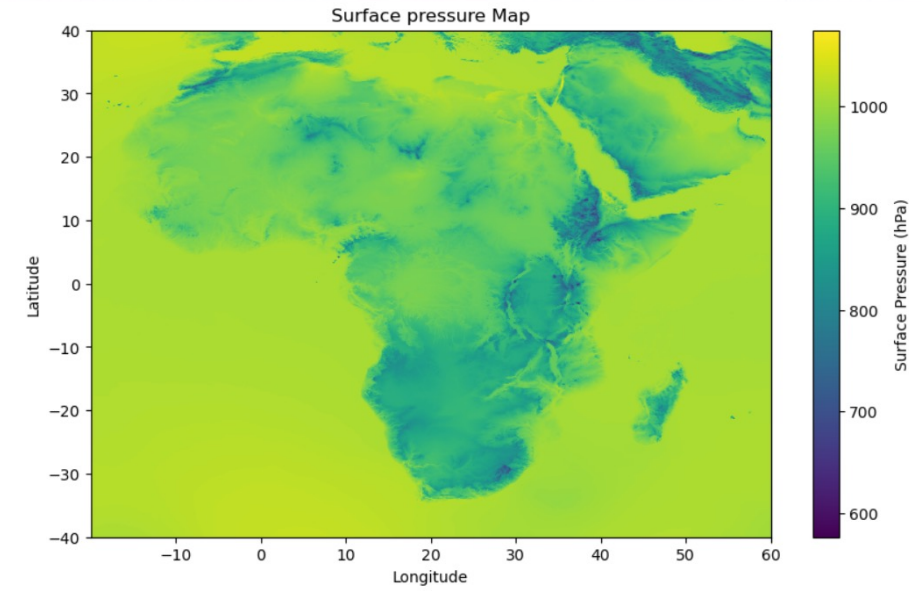
Plot map of surface pressure over Africa.

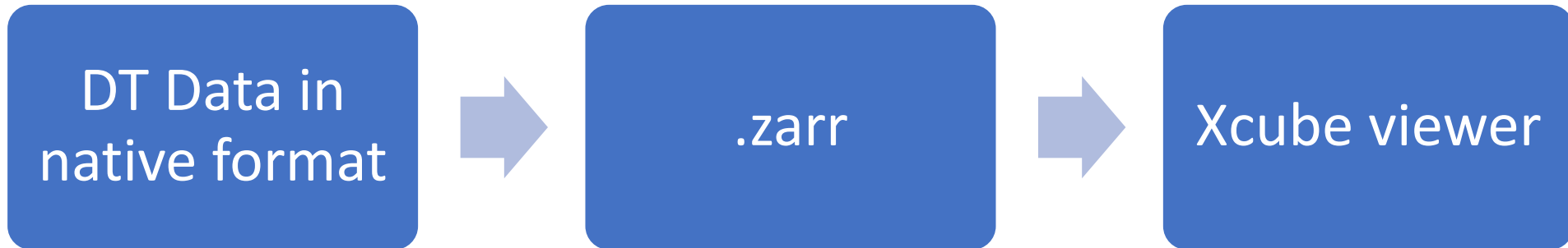
```

lon = africa_dt['lon']
lat = africa_dt['lat']
temperature = africa_dt[0] / 100 # Conversion to hectoPascals

plt.figure(figsize=(10, 6))
plt.pcolormesh(lon, lat, temperature, cmap='viridis')
plt.colorbar(label='Surface Pressure (hPa)')
plt.title('Surface pressure Map')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

```







```

africa_bbox = [-20,
              -40,
              60,
              40]

africa_aq = xr.open_zarr("data_zar/global_aq_2023.zarr").sel(longitude=slice(africa_bbox[0],
                                                                              africa_bbox[2]),
                                                             latitude=slice(africa_bbox[3],
                                                                              africa_bbox[1]),
                                                             time=slice('2023-07-01', '2023-11-30')
                )

```

```

from xcube.webapi.viewer import Viewer

viewer = Viewer(server_config={
    "styles": [
        {
            "Identifier": "no2_legend",
            "ColorMappings": {
                "no2": {
                    "ValueRange": [0, 0.000000008],
                    "ColorBar": "viridis_r"
                },
                'go3': {
                    "ValueRange": [0, 0.00000007],
                    "ColorBar": "viridis"
                },
            },
        },
    ],
})

```

```

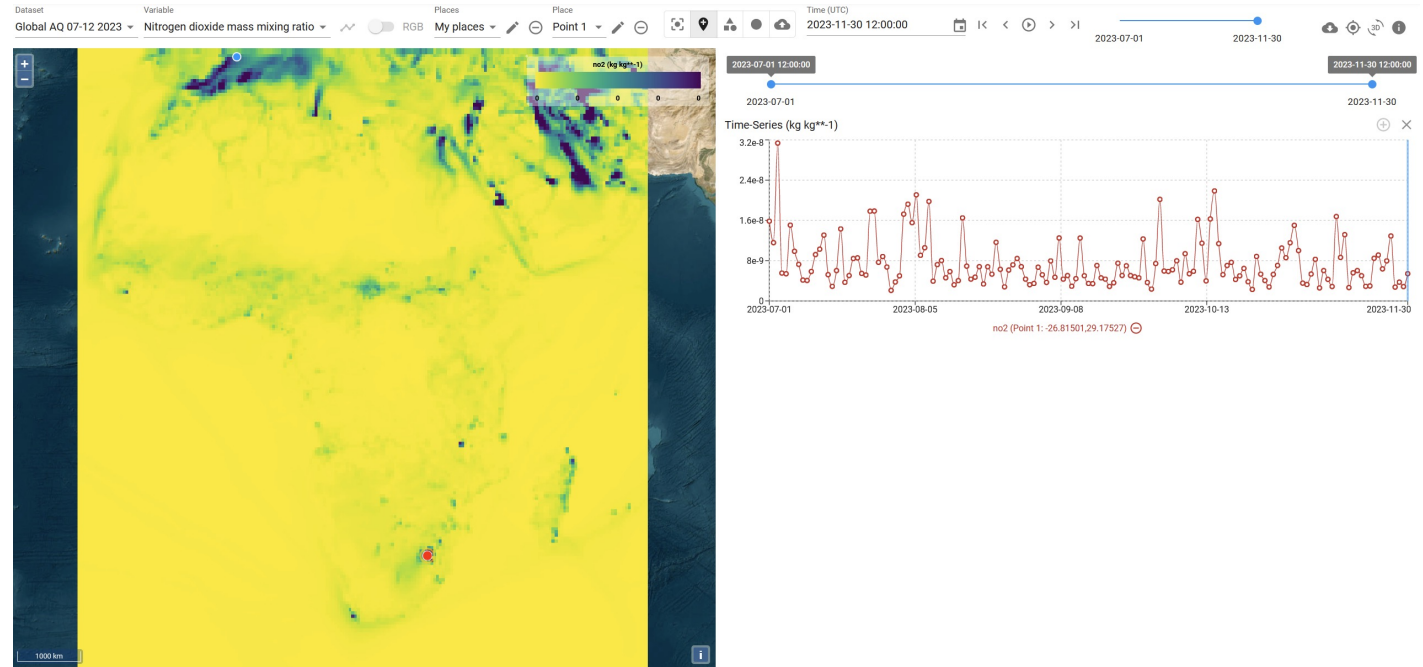
viewer.add_dataset(africa_aq,
                  style="no2_legend")

```

```

viewer.info()

```





```

from xcube.webapi.viewer import Viewer

germany_bbox = [5.866, # West
                47.270, # South
                15.042, # East
                55.099] # North

germany_dt = xr.open_zarr(url).sel(lon=slice(germany_bbox[0],
                                           germany_bbox[2]),
                                  lat=slice(germany_bbox[3],
                                           germany_bbox[1]),
                                  )

# Convert temperature from Kelvin to Celsius
germany_dt['T2M'] -= 273.15 # Conversion to Celsius degrees
germany_dt['D2M'] -= 273.15 # Conversion to Celsius degrees
# Convert pressure from Pascals to hectoPascals
germany_dt['SP'] /= 100 # Conversion to hectoPascals

# Set attribute units
germany_dt['T2M'].attrs['units'] = '°C'
germany_dt['D2M'].attrs['units'] = '°C'
germany_dt['SP'].attrs['units'] = 'hPa'

```

```

viewer = Viewer(server_config={
    "styles": [
        {
            "Identifier": "dt_legend",
            "ColorMappings": {
                "T2M": {
                    "ValueRange": [10, 20],
                    "ColorBar": "coolwarm",
                },
                "D2M": {
                    "ValueRange": [0, 10],
                    "ColorBar": "coolwarm",
                },
                'SP': {
                    "ValueRange": [0, 1000],
                    "ColorBar": "viridis"
                },
            },
        },
    ],
})

```

```

viewer.show()
viewer.info()

```

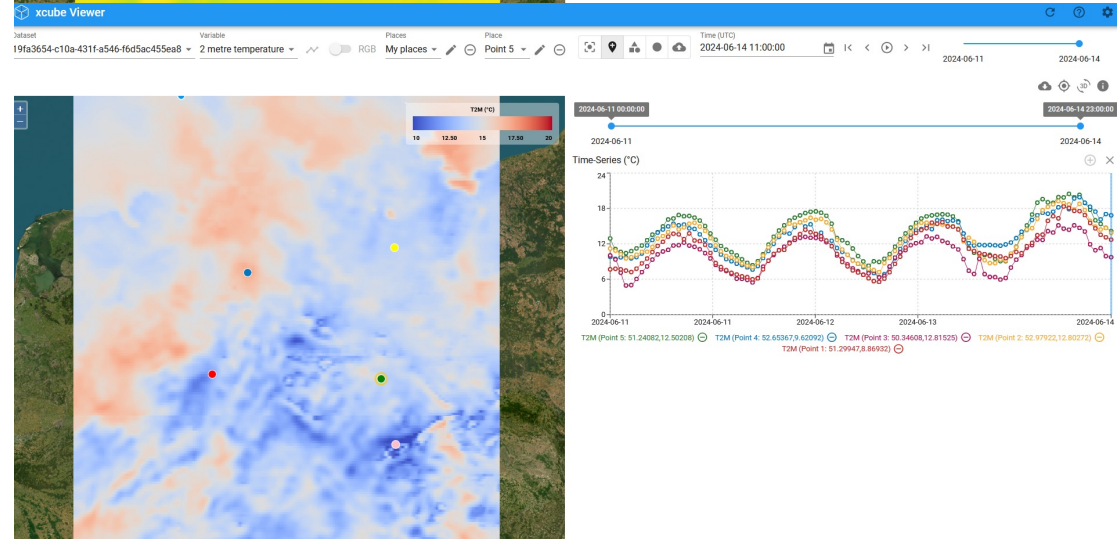
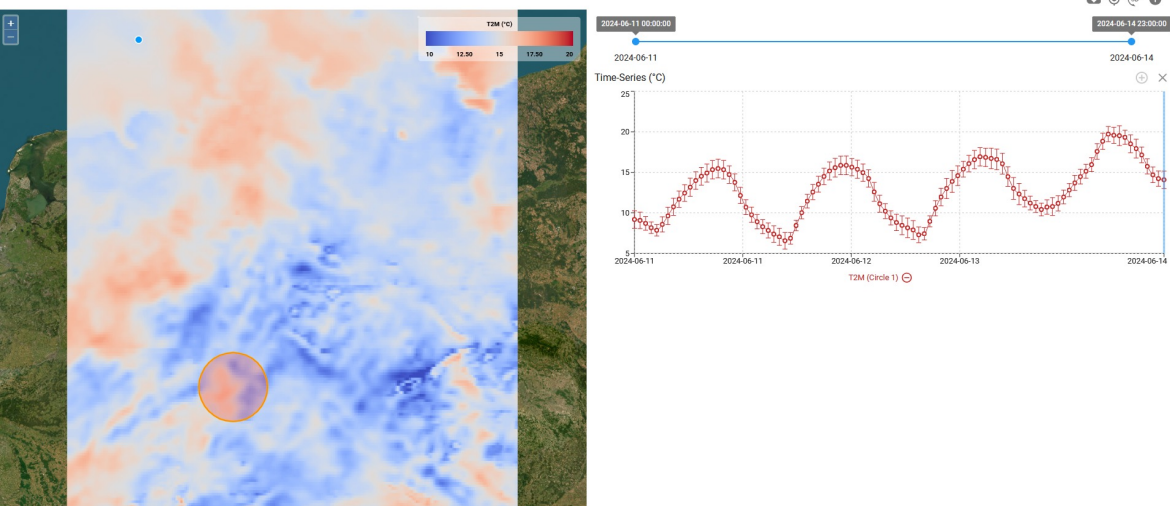
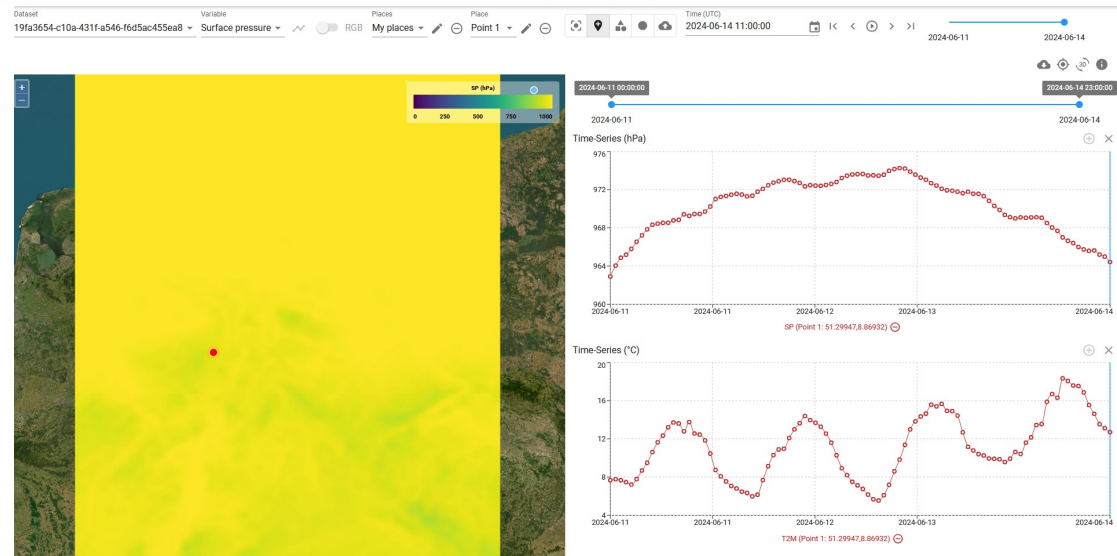
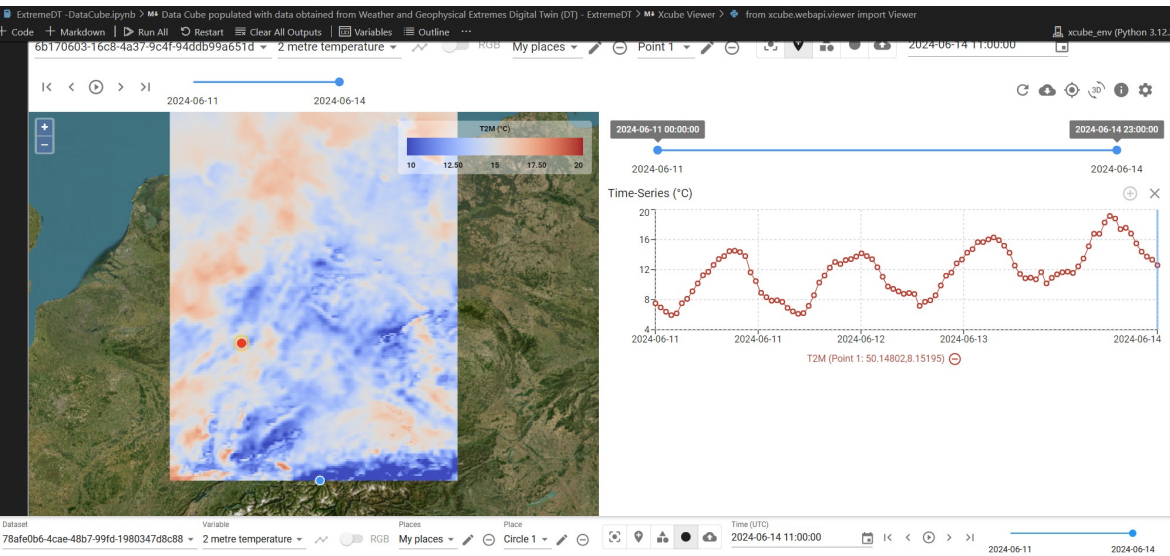
```

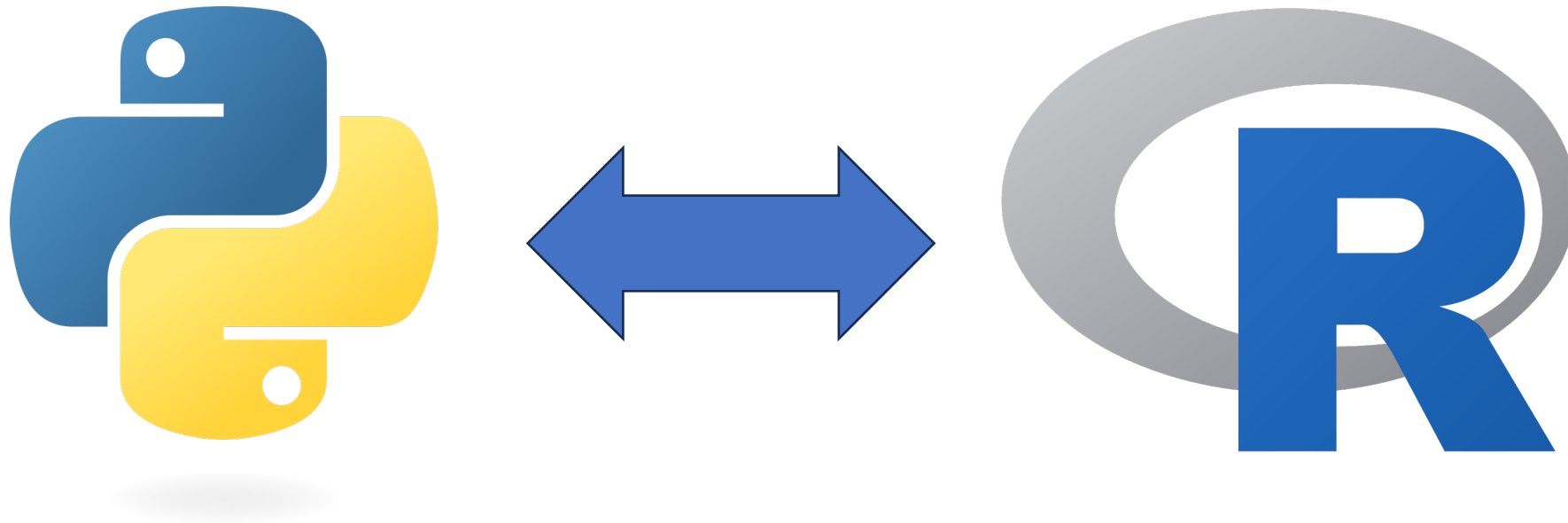
Server: http://localhost:8009
Viewer: http://localhost:8009/viewer/?serverUrl=http://localhost:8009

```



Destination Earth







https://github.com/destination-earth/DestinE-DataLake-Lab

DestinE-DataLake-Lab Public

main 6 Branches 0 Tags

serenaateum different variable in plotter 76b4240 · 12 hours ago 245 Commits

- HDA different variable in plotter 12 hours ago
- HOOK Update Hook Tutorial Desp Only Credentials and Readme 4 days ago
- STACK Data Cube in xviewer - update 3 months ago
- img style 6 months ago
- LICENSE.txt Create LICENSE.txt 4 months ago
- README.md README instruction updated and different management of er... 12 hours ago

README MIT license

Funded by the European Union **Destination Earth** IMPLEMENTED BY EUMETSAT esa ECMWF

DestinE-DataLake-Lab

Author: EUMETSAT

Destination Earth Data Lake Laboratory, which contains additional information for working with DestinE Data Lake services:

- Harmonised Data Access (Jupyter notebooks examples + Python Tools)
- STACK service (Jupyter Notebook examples on how to use DASK for near data processing)
- HOOK service (Jupyter Notebook examples on how to use HOOK for workflows)

Further information available in DestinE Data Lake documentation: <https://destine-data-lake-docs.data.destination-earth.eu/en/latest/index.html>

Additional resources:

DestinE-DataLake-Lab / HDA / DestinE Digital Twins / DEDL-HDA-EO-ECMWF.DAT.DT_CLIMATE.ipynb

Preview Code Blame 484 lines (484 loc) · 250 KB

EarthKit

Lets plot the result file [EarthKit Documentation] <https://earthkit-data.readthedocs.io/en/latest/index.html>

This section requires that you have 'ecCodes >= 2.35' installed on your system. You can follow the installation procedure at <https://confluence.ecmwf.int/display/ECC/ecCodes+installation>

```
In [23]: import earthkit.data
import earthkit.maps
import earthkit.regrid
```

```
In [24]: data = earthkit.data.from_source("file", filename)
data.ls
earthkit.maps.quickplot(data, #style=style
)
```

Surface pressure
Base time: 00:00 on 2028-06-10 Valid time: 00:00 on 2028-06-10 (T+None)

4.9e+04 5.6e+04 6.3e+04 7e+04 7.7e+04 8.4e+04 9.1e+04 9.8e+04 1.05e+05

Surface pressure (Pa)





Destination Earth

Thank you!

pgrzybowski@cloudferro.com

<https://github.com/destination-earth/DestinE-DataLake-Lab>



Destination Earth

Funded by
the European Union



Implemented by

