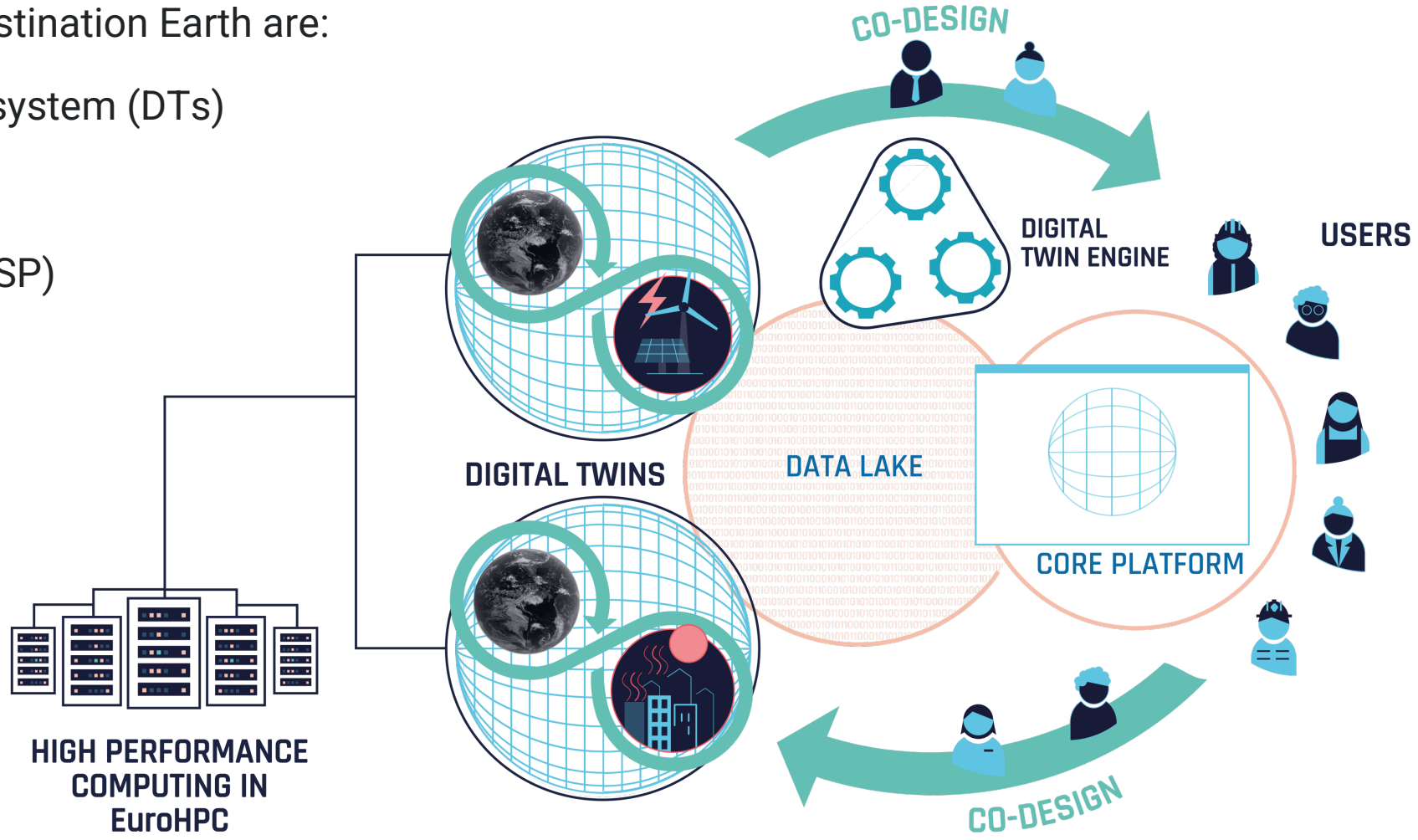
THE DIGITAL TWIN ENGINE

The main components of Destination Earth are:

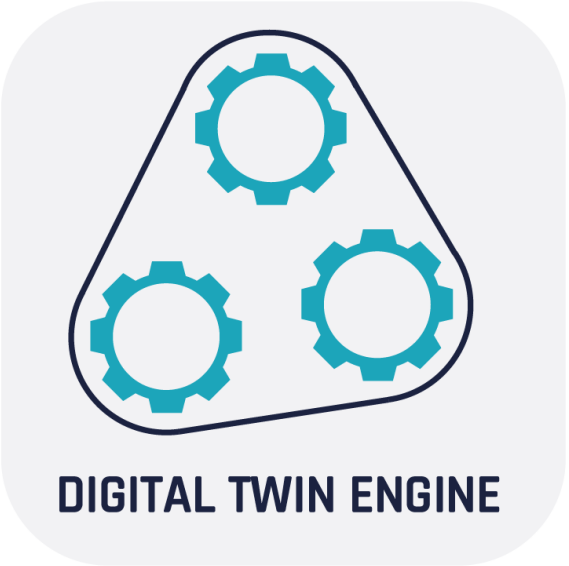
- Digital Twins of the Earth system (DTs)
- Digital Twin Engine (DTE)
- Core Service Platform (DESP)
- Data Lake (DEDL)



THE DIGITAL TWIN ENGINE

The Digital Twin Engine (DTE) is:

- A modular software framework comprising of many different components
- Designed to create a unified ecosystem for Digital Twins
- Designed for **interoperability**

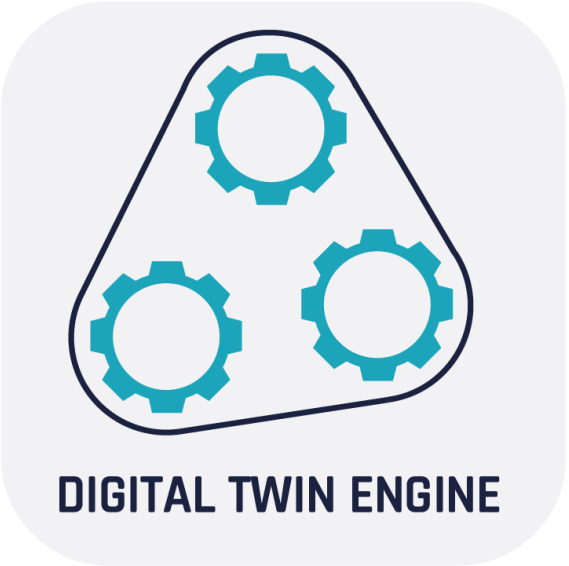


THE DIGITAL TWIN ENGINE

What do we mean by “interoperability”?

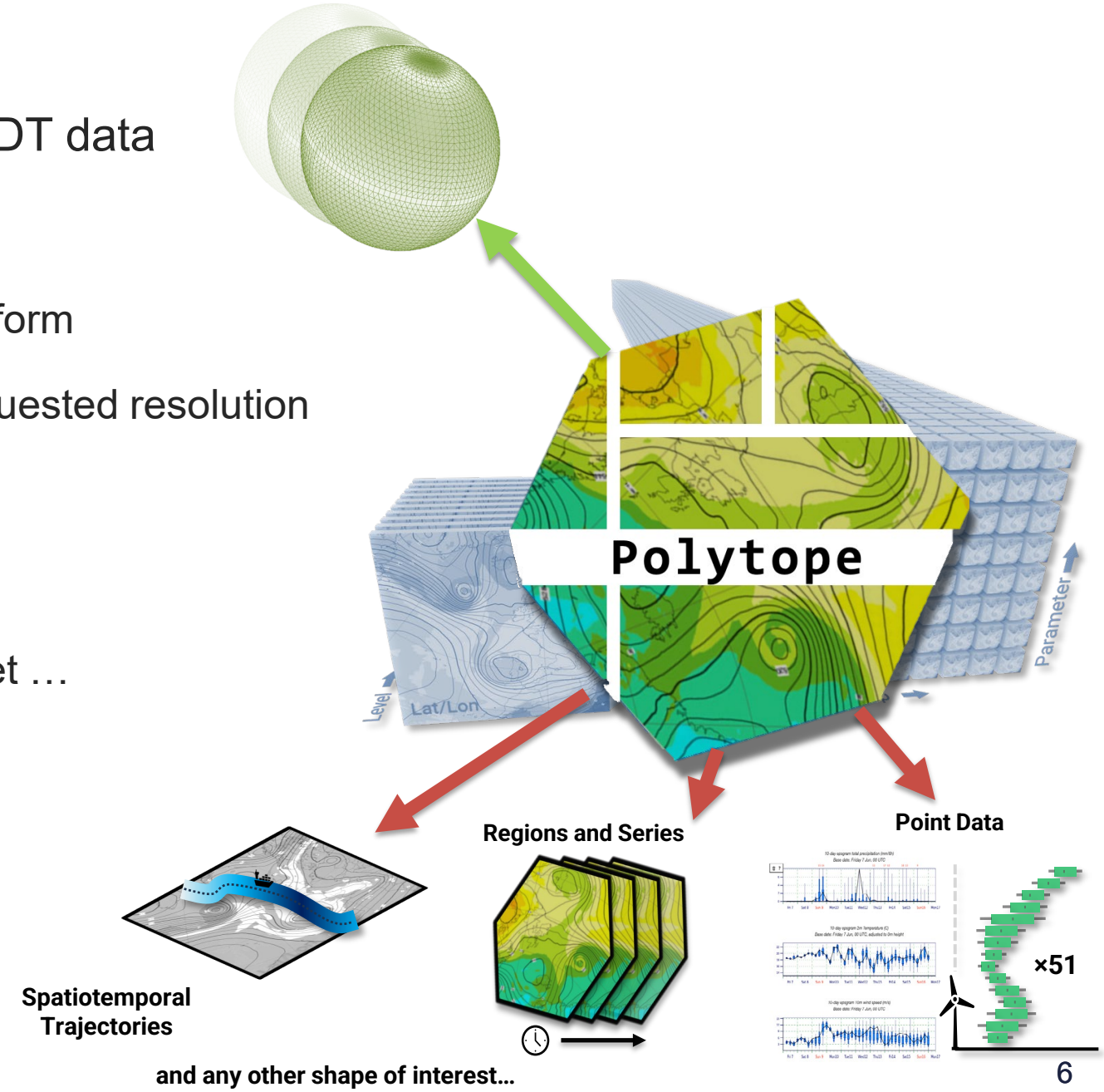
- Make it easy to run DTs on different infrastructure
- Make it easy for DTs to talk to each other
- Make it easy for operators to control their DTs
- **Make it easy for users to interact with DTs and their outputs**

This is why you’re here



POLYTOPE

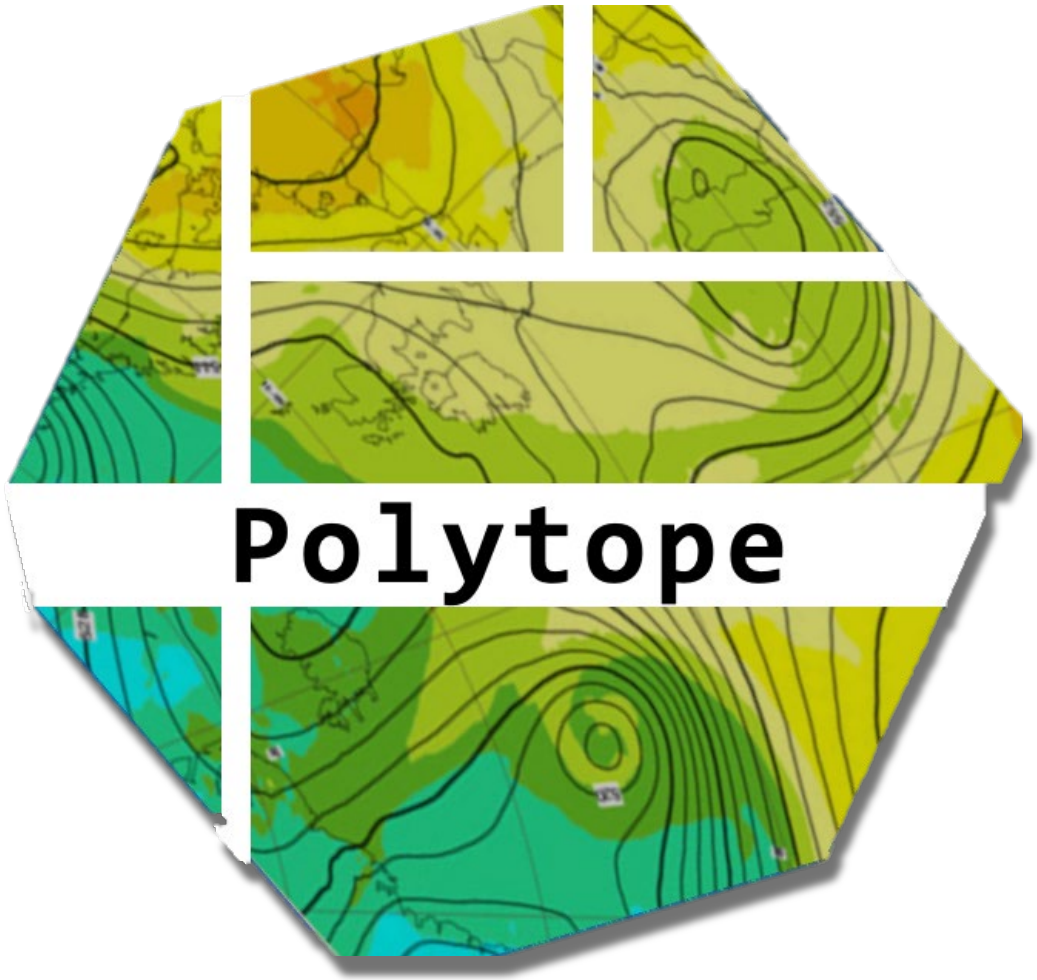
- Data service providing direct access to DT data
- Serves full fields at full resolution
  - The DT data in its original, raw unaltered form
  - Server-side interpolation on-the-fly to requested resolution
- Also offers “feature extraction”
  - Able to jump through the dense data to get …



DATASETS

Polytope provides access to all data on the data bridges including:

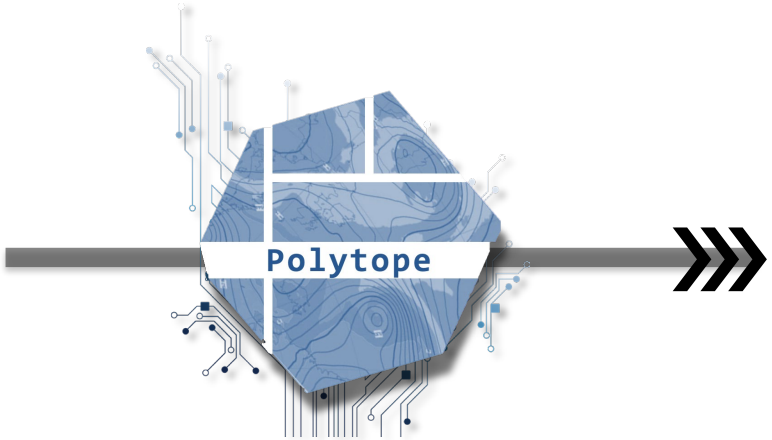
- Climate DT
- Extremes DT
- On-Demand DT
- NextGEMS



POLYTOPE FEATURE EXTRACTION

- Users can submit a data request with a feature specification, on any DT data
- And receive an OGC CoverageJSON with just the few bytes of extracted data


```
{
    "class": "od",
    "stream": "enfo",
    "type": "pf",
    "date": "20231205",
    "time": "0000",
    "levtype": "sfc",
    "expver": "0001",
    "param": "228/49/164/165/166/167",
    "number": "1/to/50",
    "feature": {
        "type": "timeseries",
        "points": [[ 51.5, 2.1 ]],
        "start": 0,
        "end": 9
    }
}
```


```
{
    "type": "CoverageCollection",
    "domainType": "PointSeries",
    "coverages": [
        {
            "mars:metadata": {
                "class": "od",
                "stream": "oper",
                "levtype": "pl",
                "date": "20231205",
                "number": "0",
            },
            "type": "Coverage",
            "domain": {
                "type": "Domain",
                "axes": {
                    "x": {"values": [51.5]},
                    "y": {"values": [2.1]},
                    "t": {
                        "values": [
                            "2017-01-01 00:00:00",
                            "2017-01-01 01:00:00",
                            "2017-01-01 02:00:00",
                            "2017-01-01 03:00:00",
                            "2017-01-01 04:00:00",
                            "2017-01-01 05:00:00",
                            "2017-01-01 06:00:00",
                            "2017-01-01 07:00:00",
                            "2017-01-01 08:00:00",
                            "2017-01-01 09:00:00",
                        ]
                    },
                },
            },
            "ranges": {
                "t2m": {
                    "type": "NdArray",
                    "dataType": "float",
                    "shape": [3],
                    "axisNames": ["t"],
                    "values": [
                        264.93115234375,
                        263.83115234375,
                        265.12313132266,
                        264.93115234375,
                        263.83115234375,
                        265.12313132266,
                        264.93115234375,
                        263.83115234375,
                        265.12313132266,
                    ],
                }
            },
        },
    ],
    "referencing": [...],
    "parameters": {...}
}
```

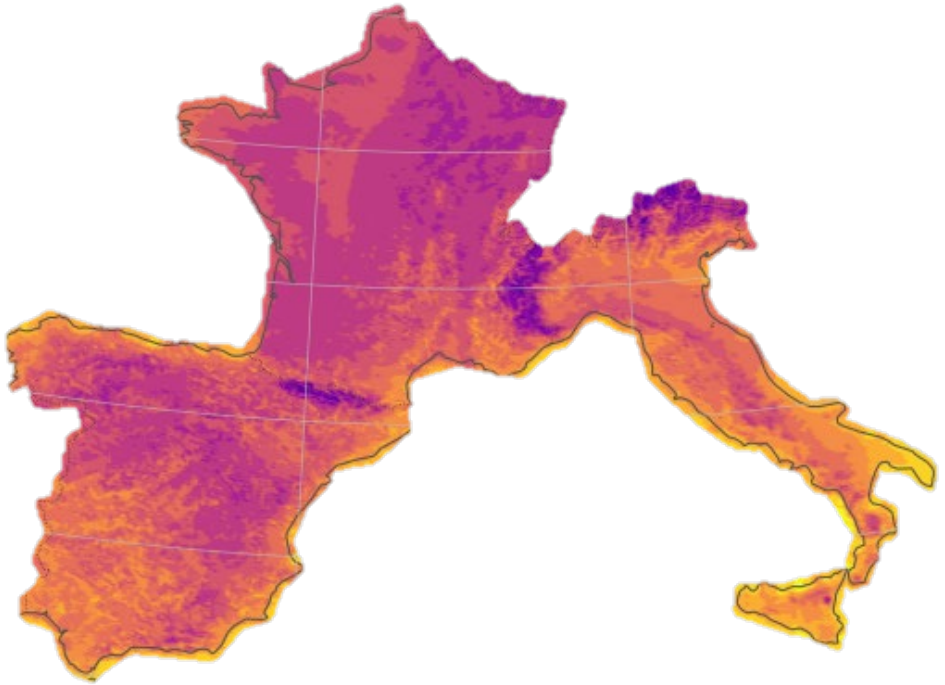
POLYTOPE

Polygon extraction of Spain, France and Italy

vs.

Extracting the entire global field

Percentage Reduction: **~95%**



POLYTOPE

Climate DT 20-year time-series of one parameter

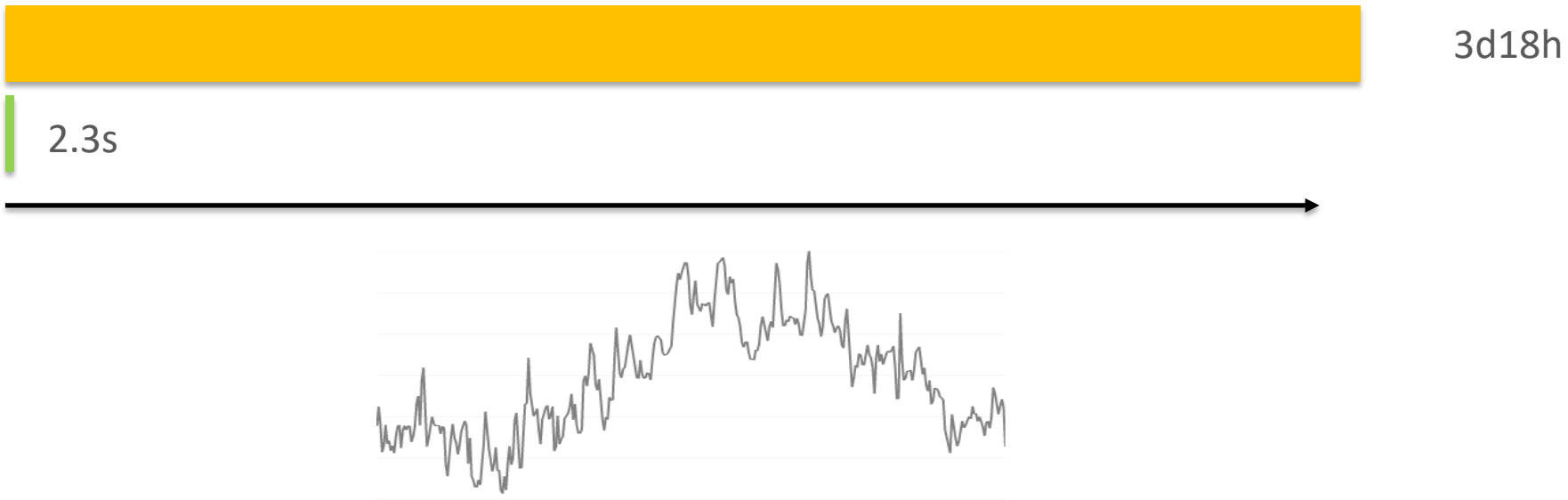
Global fields: **~770 GiB**

vs.

A single point: **5.5 MiB**

We don't even read the excess data from disk!

Percentage Reduction: **~99.999%**
(1:140000)



DOCUMENTATION AND EXAMPLES

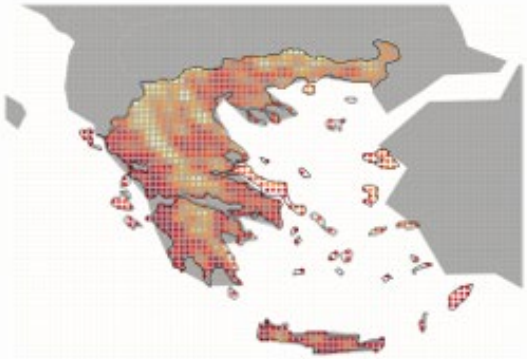
Can be found here https://polytope.readthedocs.io

Welcome to Polytope's documentation!

Polytope is a feature extraction software developed by ECMWF. It uses concepts of computational geometry to extract n-dimensional polygons (also known as *polytopes*) from datacubes.

In particular, it can be used to request:

- 2D cut-outs, such as country cut-outs, from a datacube

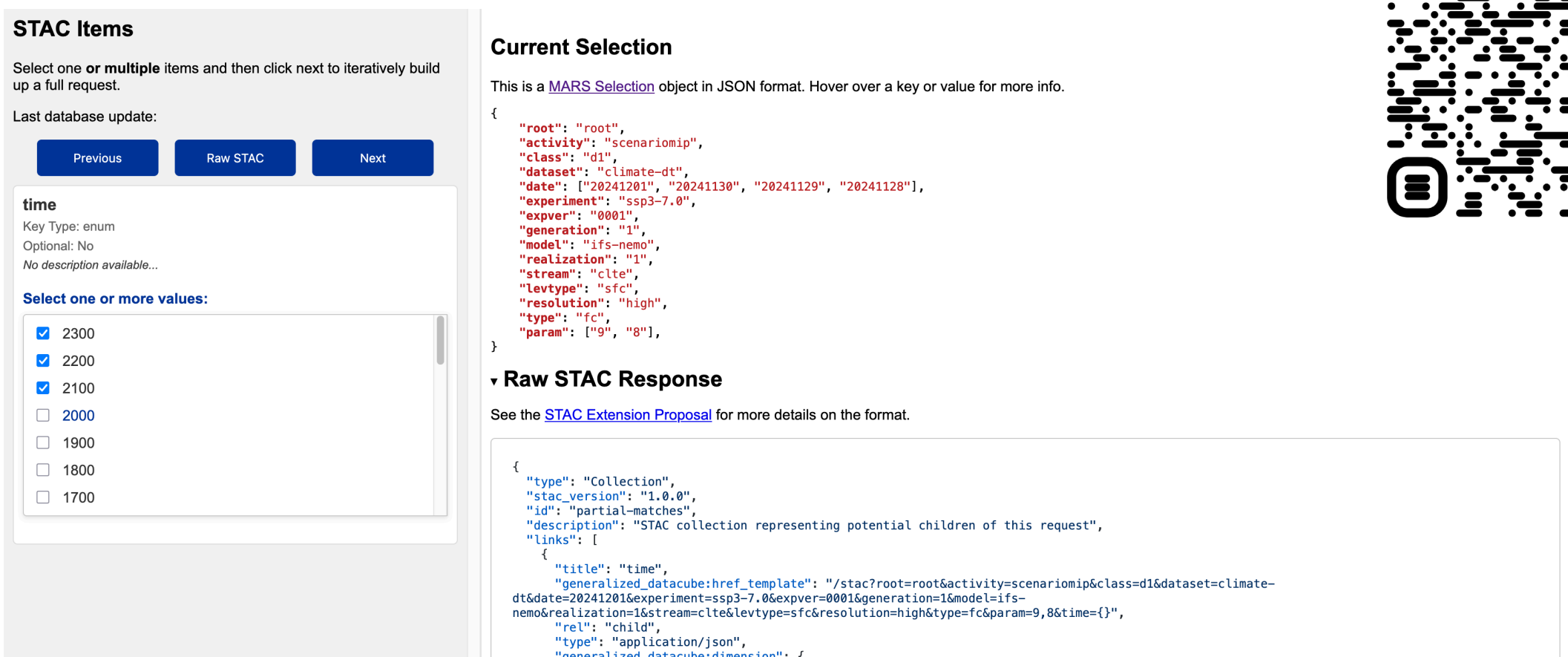


DestinE example notebooks can be found here https://github.com/destination-earth-digital-twins/polytope-examples/tree/main

CATALOGUE

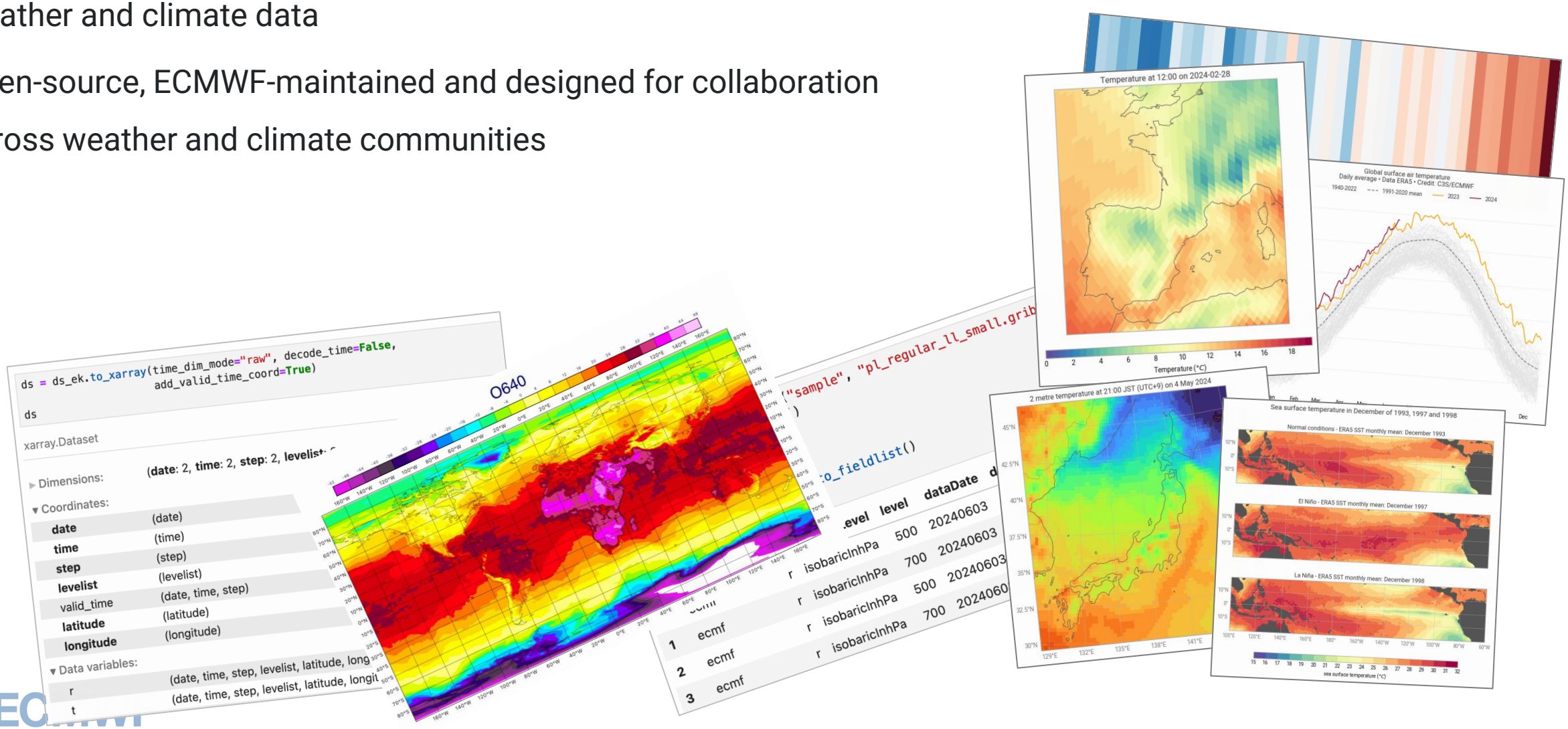
catalogue.lumi.apps.dte.destination-earth.eu

- Prototype of this catalogue is up and running
- Climate DT data on LUMI is approximately 8.6 million fields
- JSON representation of the entire tree ~145KiB
- Our STAC extension makes it easy to build a dynamic request for 1000's of fields at once



EARTHKIT

- **Earthkit** is a set of powerful Python components for interacting with weather and climate data
- Open-source, ECMWF-maintained and designed for collaboration across weather and climate communities



EARTHKIT

earthkit data

earthkit geo

earthkit hydro

earthkit workflows

earthkit meteo

earthkit utils

earthkit plots

earthkit regrid

earthkit time

earthkit transforms

POLYTOPE SUPPORT

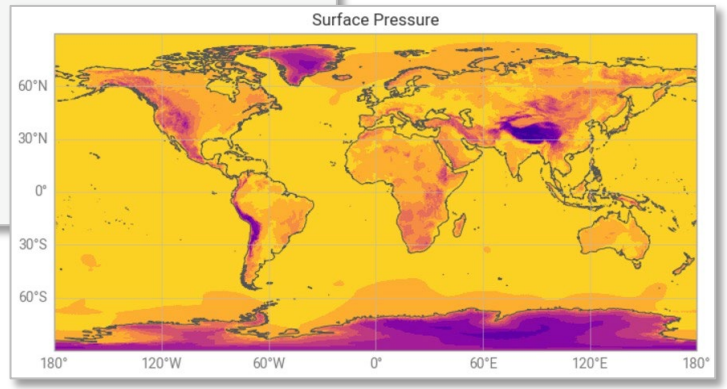
HEALPix GRIB DATA from Polytope

```python
import earthkit.data
import earthkit.plots
import earthkit.regrid


request = {
    'activity': 'ScenarioMIP',
    'class': 'd1',
    'dataset': 'climate-dt',
    'date': '20200102',
    'experiment': 'SSP3-7.0',
    'expver': '0001',
    'generation': '1',
    'levtype': 'sfc',
    'model': 'IFS-NEMO',
    'param': '134/165/166',
    'realization': '1',
    'resolution': 'standard',
    'stream': 'clte',
    'time': '0100', # '0100/0200/0300/0400/0500/0600'
    'type': 'fc'
}

# data is an earthkit streaming object but with stream=False will
data = earthkit.data.from_source("polytope", "destination-earth",
```

earthkit data

earthkit plots

```python
chart = earthkit.plots.Map(extent=[-180, 180, -90, 90])
chart.plot(
    data[0]
)

chart.title("Surface Pressure")
chart.coastlines()
chart.gridlines()
chart.show()
```

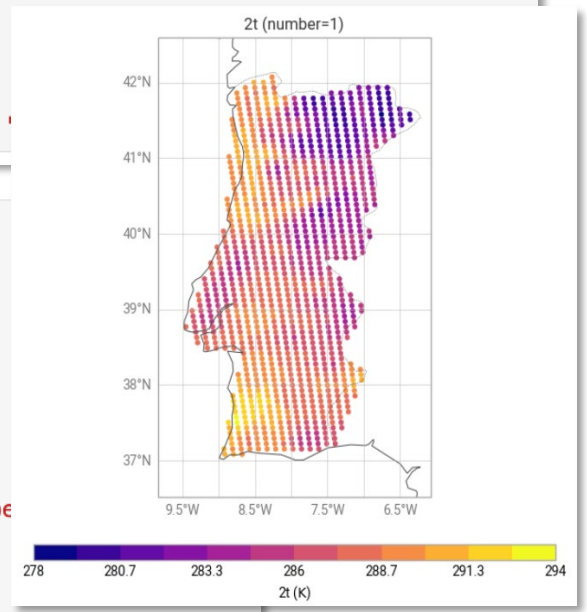


ECMWF

Polygons from Polytope (CovJSON)

```python
import earthkit.data as ekd

request = {
    "class": "od",
    "stream" : "enfo",
    "type" : "pf",
    "date" : -1,
    "time" : "1200",
    "levtype" : "sfc",
    "expver" : 1,
    "domain" : "g",
    "param" : "167/169",
    "number" : "1",
    "step": "0",
    "feature": {
        "type": "polygon",
        "shape": coords
    },
}

ds = ekd.from_source("polytope", '
```

```python
da = ds.to_xarray()

import earthkit.plots as ekp

chart = ekp.Map(domain="Portugal")
chart.point_cloud(da['2t'], x="y",
chart.coastlines()
chart.borders()
chart.gridlines()
chart.title("{variable_name} (numbe
chart.legend()

chart.show()
```



XARRAY CONVERSION

earthkit data

HEALPix GRIB DATA from Polytope

```python
import earthkit.data as ekd

request = {
    'activity': 'ScenarioMIP',
    'class': 'd1',
    'dataset': 'climate-dt',
    'date': '20200102',
    'experiment': 'SSP3-7.0',
    'expver': '0001',
    'generation': '1',
    'levtype': 'sfc',
    'model': 'IFS-NEMO',
    'param': '134/166/167',
    'realization': '1',
    'resolution': 'standard',
    'stream': 'clte',
    'time': '1/2',
    'type': 'fc'
}

ds = ekd.from_source("polytope", "destination-earth", request,
                     address="polytope.lumi.apps.dte.destinati
                     stream=False)
```

```python
ds.to_xarray(time_dim_mode="valid_time")
```

xarray.Dataset

▸ Dimensions: (**valid_time**: 2, values: 196608)

▾ Coordinates:

valid_time	(valid_time)	datetime64[ns]	2020-01-02T01:00:00 2
latitude	(values)	float64	...
longitude	(values)	float64	...

▾ Data variables:

10v	(valid_time, values)	float64	...
2t	(valid_time, values)	float64	...
sp	(valid_time, values)	float64	...

ATHER FORECASTS

Polygons from Polytope (CovJSON)

```python
import earthkit.data as ekd

request = {
    "class": "od",
    "stream" : "enfo",
    "type" : "pf",
    "date" : -1,
    "time" : "1200",
    "levtype" : "sfc",
    "expver" : 1,
    "domain" : "g",
    "param" : "167/169",
    "number" : "1",
    "step": "0",
    "feature": {
        "type": "polygon",
        "shape": coords
    },
}

ds = ekd.from_source("polytope", "ecmwf-mars", request, stream=False, a
```

```python
ds.to_xarray()
```

xarray.Dataset

▸ Dimensions: (**datetimes**: 1, **number**: 1, **steps**: 1, **points**: 1093)

▾ Coordinates:

datetimes	(datetimes)	<U20	'2025-11-02T12:00:00Z'
number	(number)	int64	1
steps	(steps)	int64	0
points	(points)	int64	0 1 2 3 4 ... 1089 1090 1091 10...
latitude	(points)	float64	37.08 37.08 37.08 ... 42.0 42.07
longitude	(points)	float64	351.1 351.2 351.3 ... 351.7 351.7
levelist	(points)	float64	0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.0

▾ Data variables:

2t	(datetimes, number, steps, points)	float64	293.9 295.0 294.0 ... 285.2 28...
ssrd	(datetimes, number, steps, points)	float64	0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.0

REGRIDDING

Getting data on HEALPix grid from Polytope

```
import earthkit.data as ekd

request = {
    'activity': 'ScenarioMIP',
    'class': 'd1',
    'dataset': 'climate-dt',
    'date': '20200102',
    'experiment': 'SSP3-7.0',
    'expver': '0001',
    'generation': '1',
    'levtype': 'sfc',
    'model': 'IFS-NEMO',
    'param': '134',
    'realization': '1',
    'resolution': 'standard',
    'stream': 'clte',
    'time': '1/2',
    'type': 'fc'
}

ds = ekd.from_source("polytope", "destination-earth", request,
                     address="polytope.lumi.apps.dte.dest
                     ream=False)
```

```
ds.ls(keys=["param", "gridType"])
```

	param	gridType
0	2t	healpix
1	2t	healpix

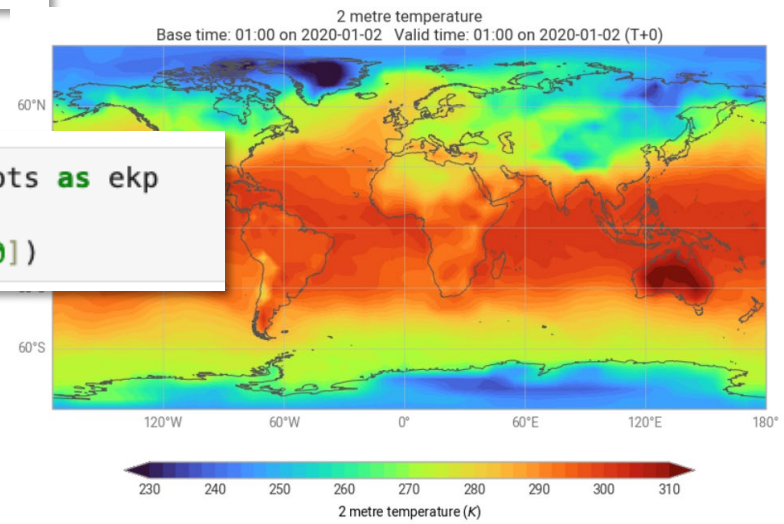
Interpolate to regular-latlon grid

earthkit regrid

```
from earthkit.regrid import interpolate


ds1 = interpolate(ds, out_grid={"grid": [5,5]})
ds1.ls(keys=["param", "gridType"])
```

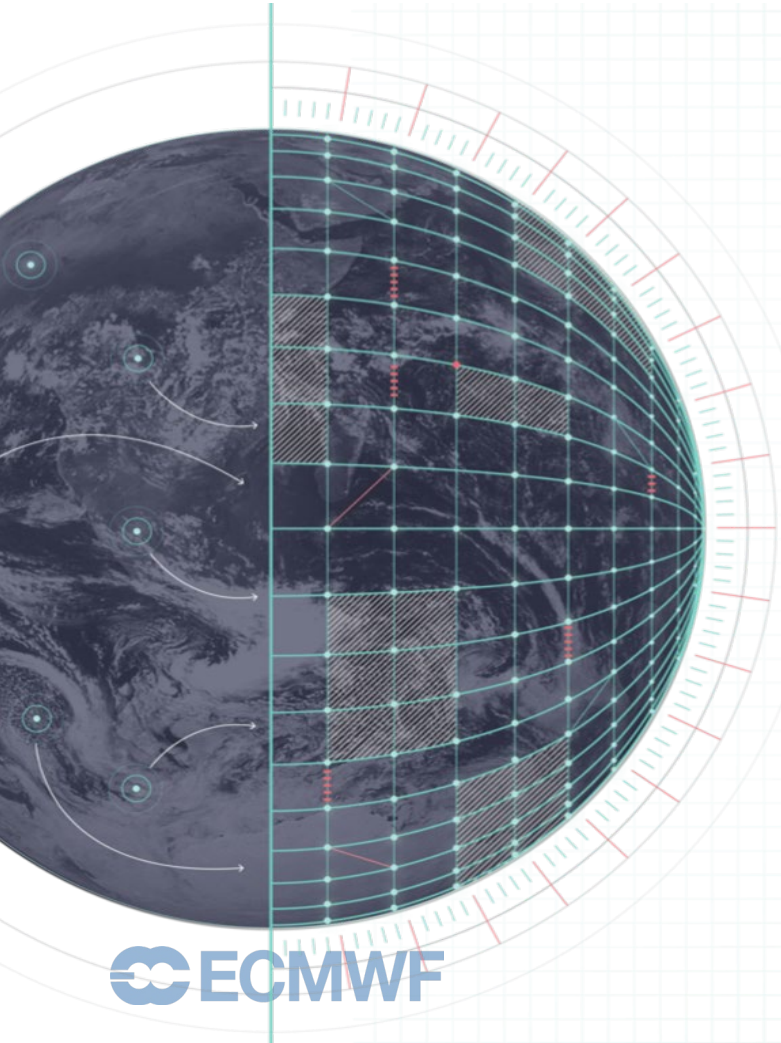
	param	gridType
0	2t	regular_ll
1	2t	regular_ll

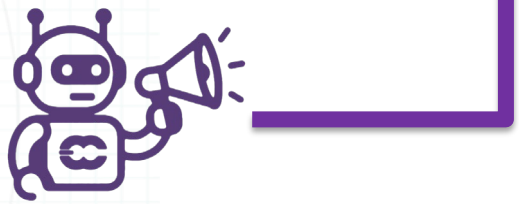
```
import earthkit.plots as ekp

ekp.quickplot(ds1[0])
```



AVISO

- Notifies of data availability from the ECMWF and DestinE
- Designed to integrate with and automate downstream workflows
- When «**this**» then «**that**»





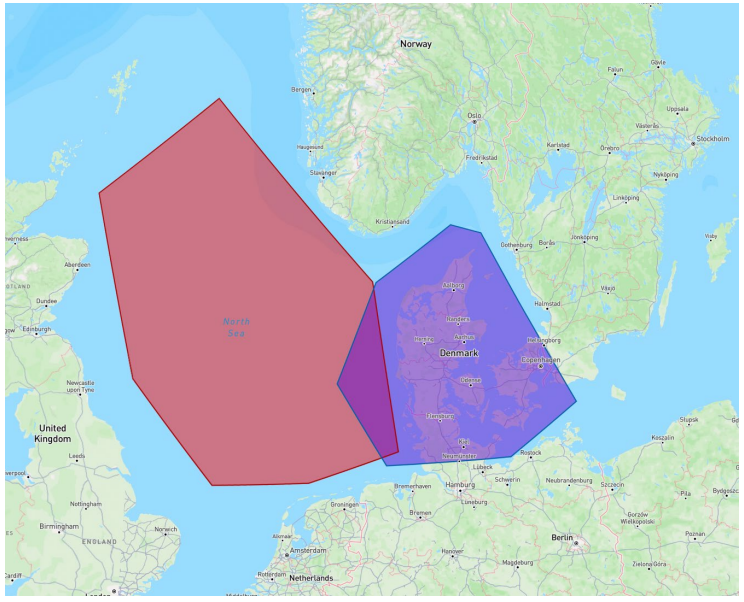
New ECMWF data available
today's Extremes DT available
new climate DT available
on-demand DT available
ECPDS product available

do

run my analysis
plot the new data
update my website
schedule another DT

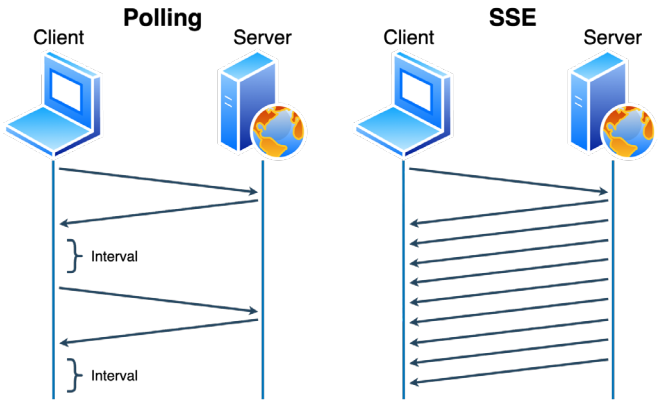
AVISO

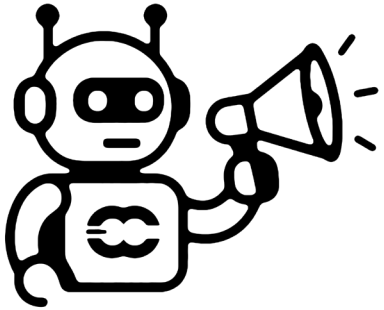
- New in phase 2: Geofenced Notifications
  - Allows us to send notifications of extreme events with an area of impact
  - And allows users to listen to areas of interest
  - Used to trigger downstream modelling
  - Strong collaboration with on-demand DT and ECMWF member states
  - Prototype is working, aiming for production in Q1 2026



AVISO

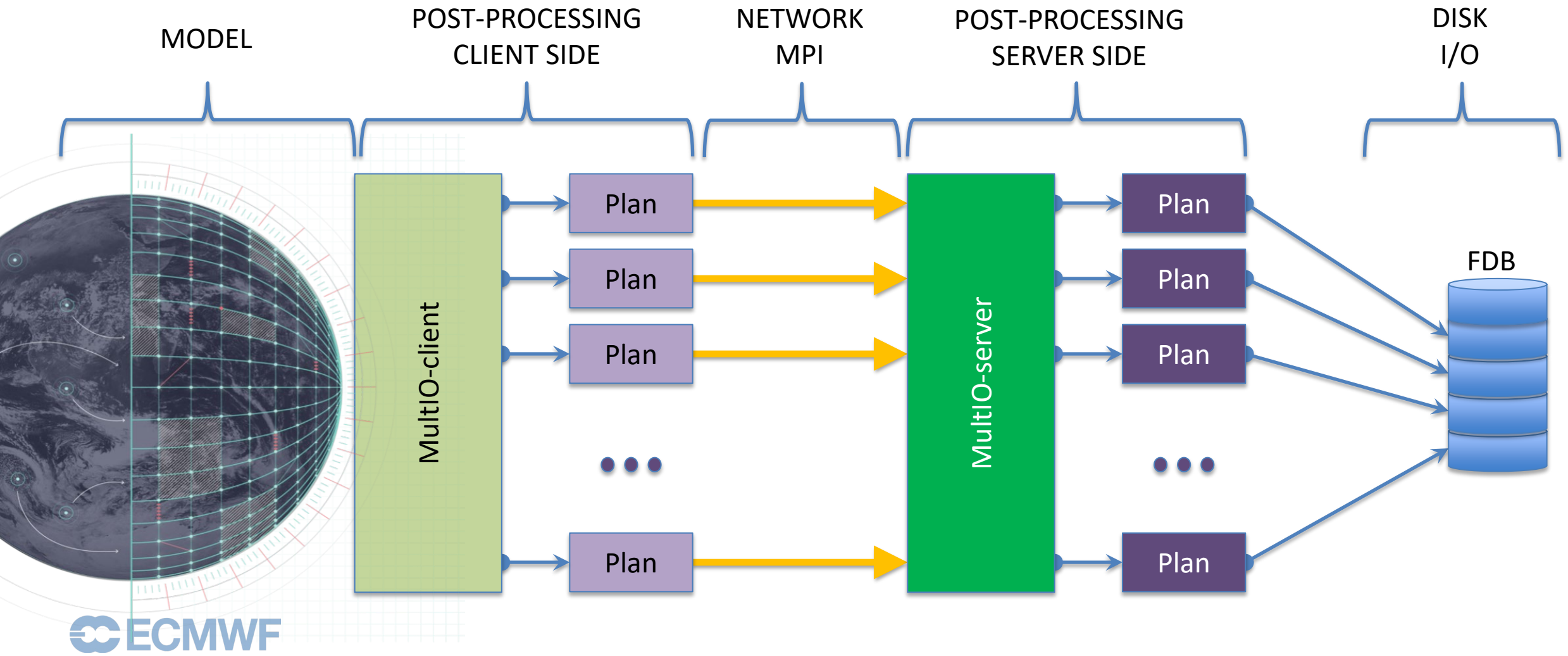
- Many other improvements
  - Redesigned client-server protocol to be much more lightweight
  - Replaced backend storage to be much more maintainable
  - Rebuilt server backend in Rust, more streamlined



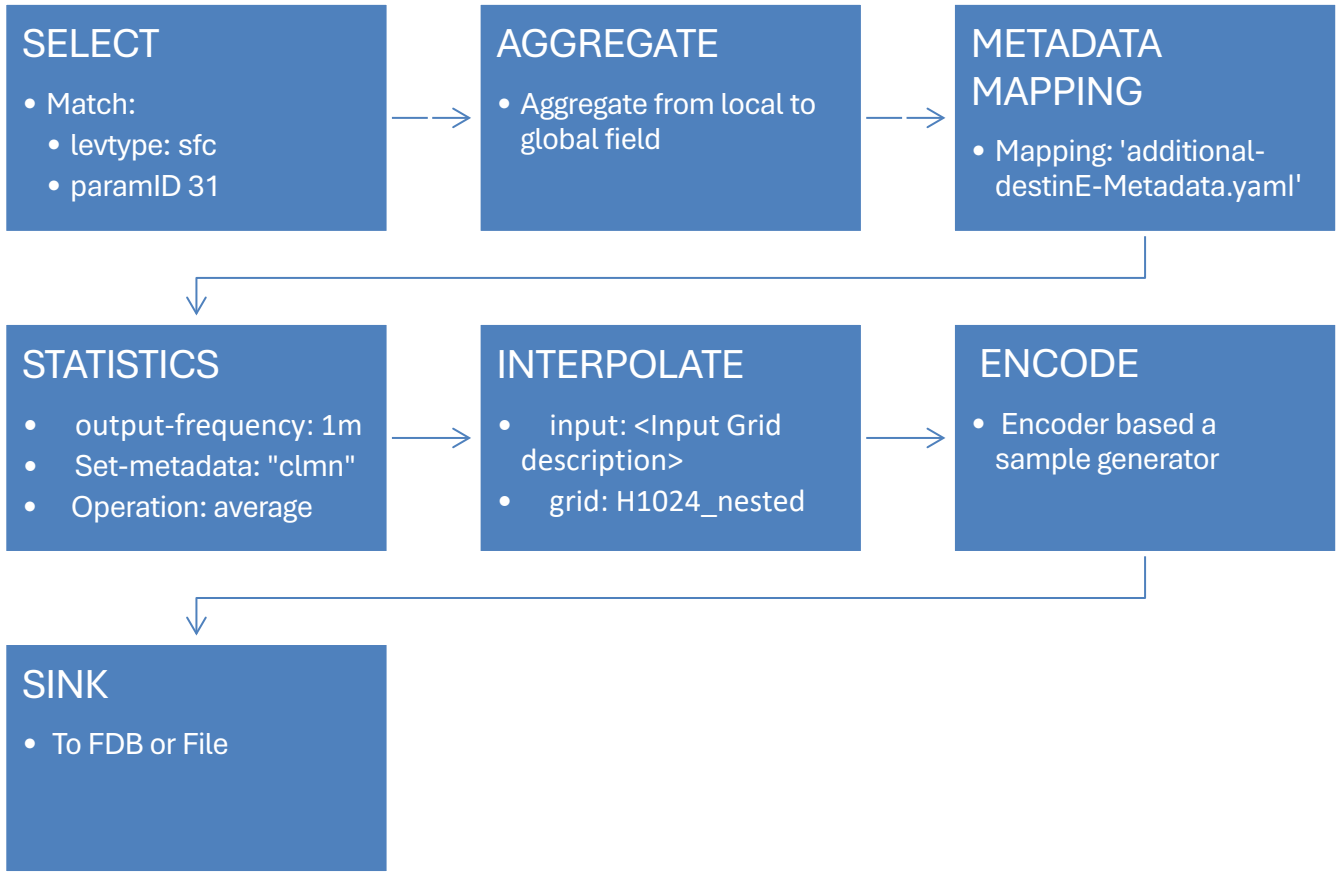


AVISO
v2.0

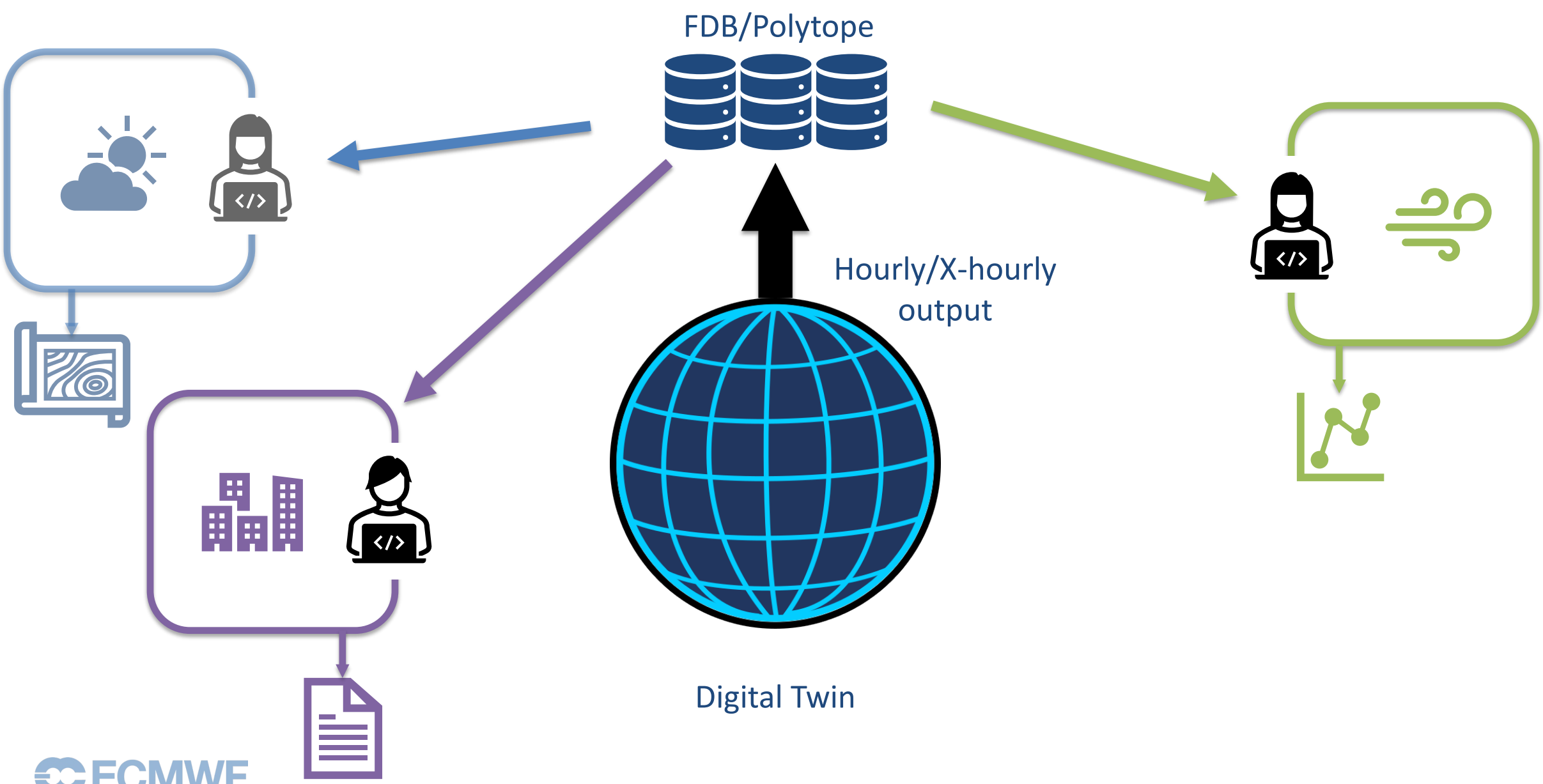
MULTIO: Message based postprocessing framework



EXAMPLE POST PROCESSING PIPELINE



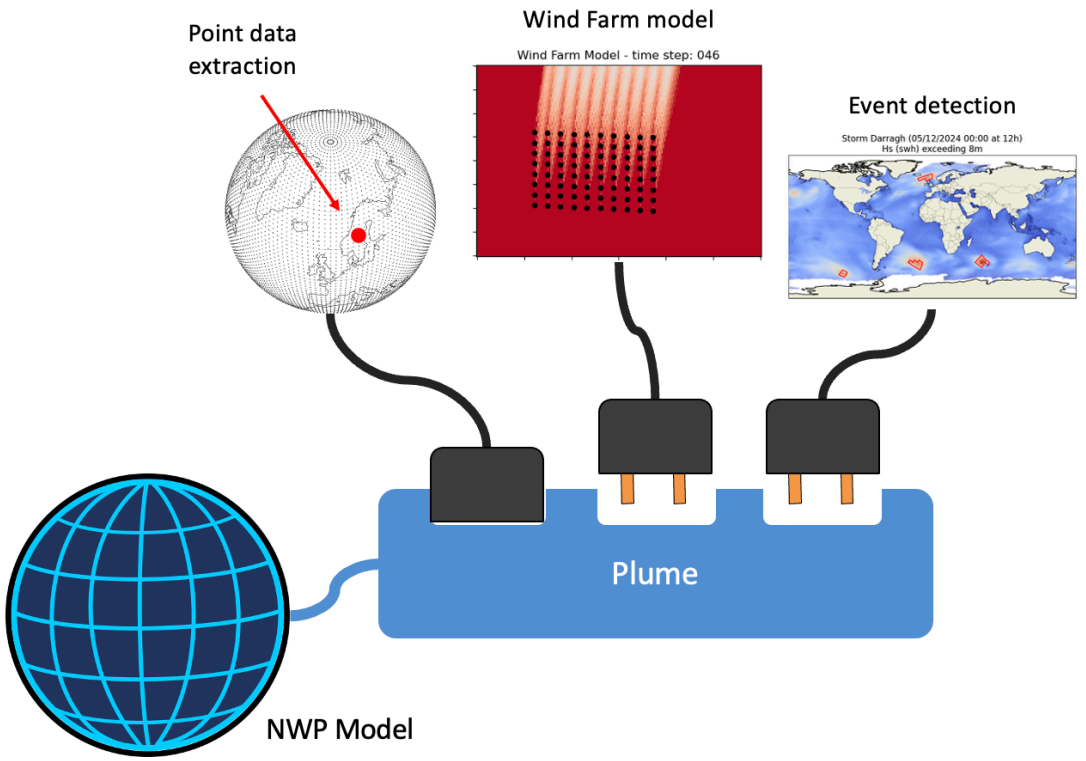
PLUME



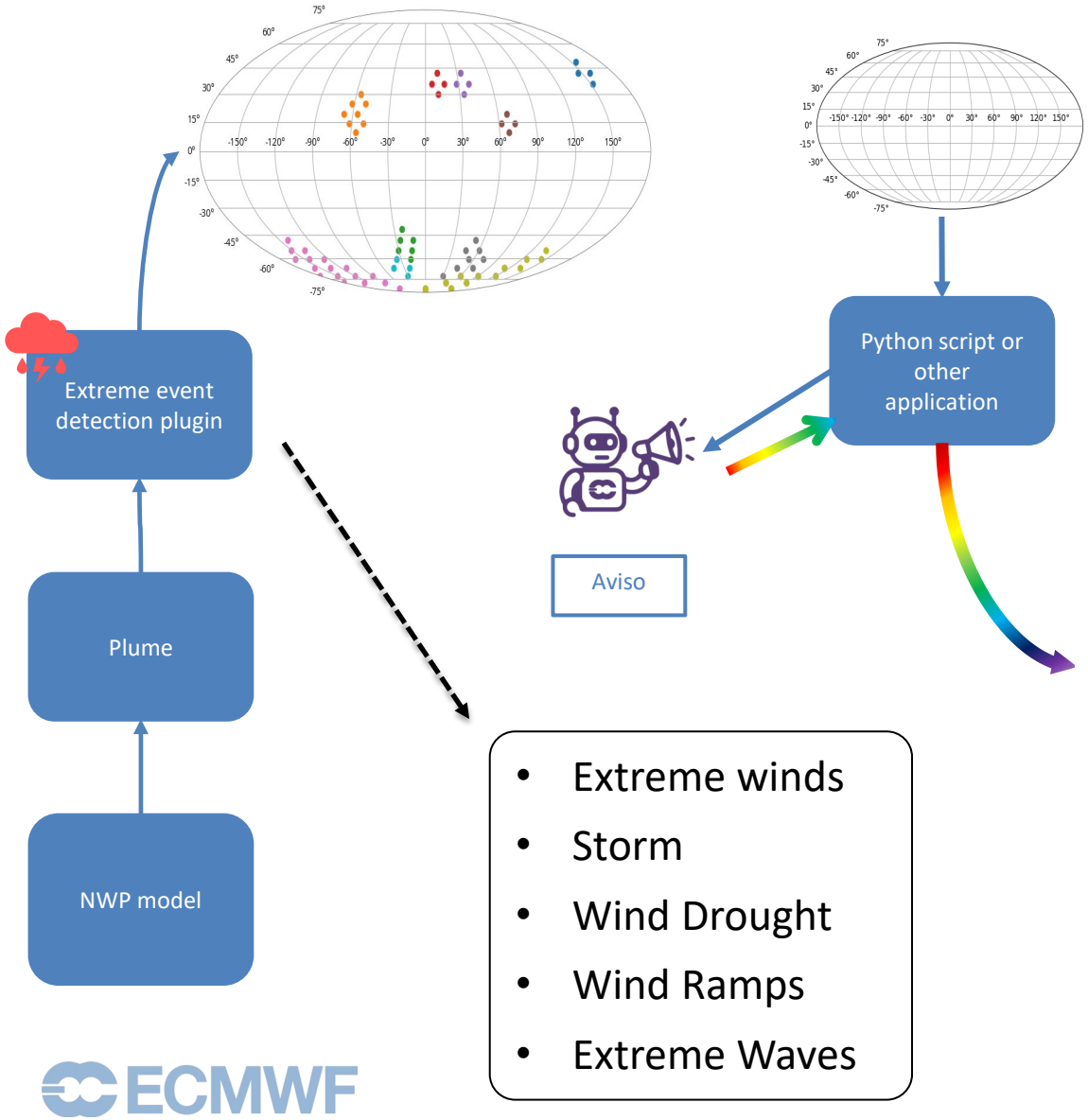
PLUME

- Some **use-cases would benefit from closer coupling**:
  - Run as part of the HPC job which executes the model
  - Read model data in-memory
  - Access data at every (internal) timestep

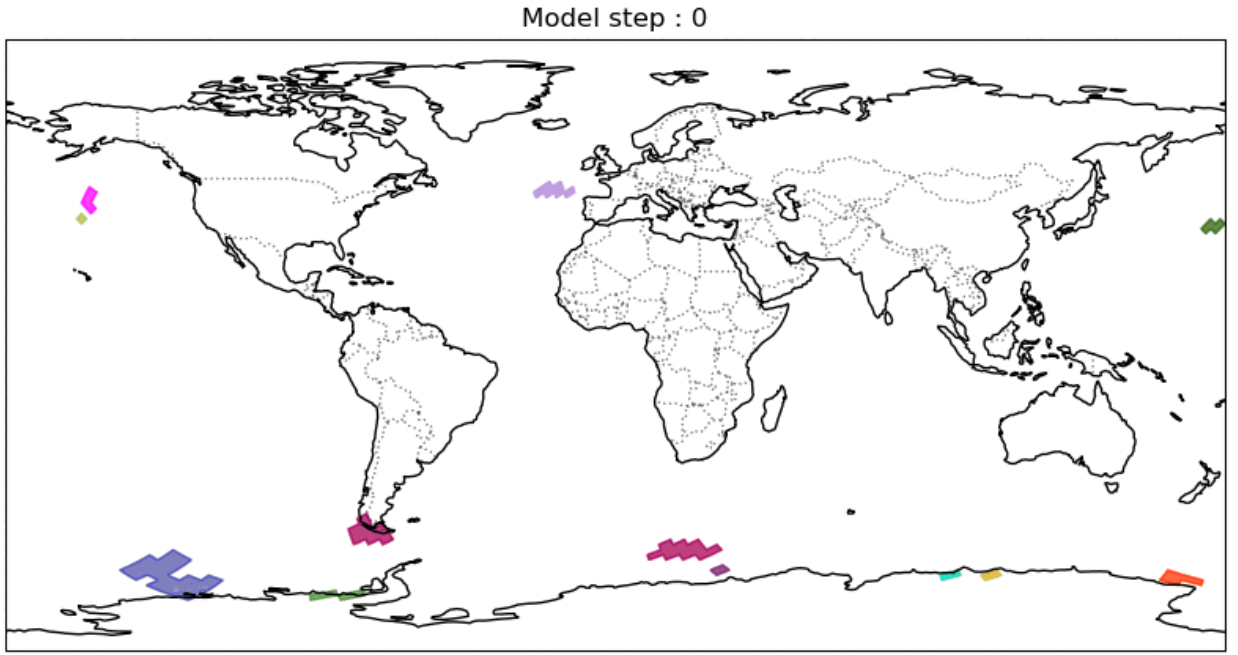
- Plume is a **plug-in mechanism** which allows this interactivity with Digital Twins
  - One-way plugins: generate specific outputs from the Digital Twin
  - Two-way plugins (later): influence the physics of the Digital Twin
  - Provides a standardized API for accessing internal parts of the model, without DT-specific knowledge



PLUME



Extreme Event Detection Plugin

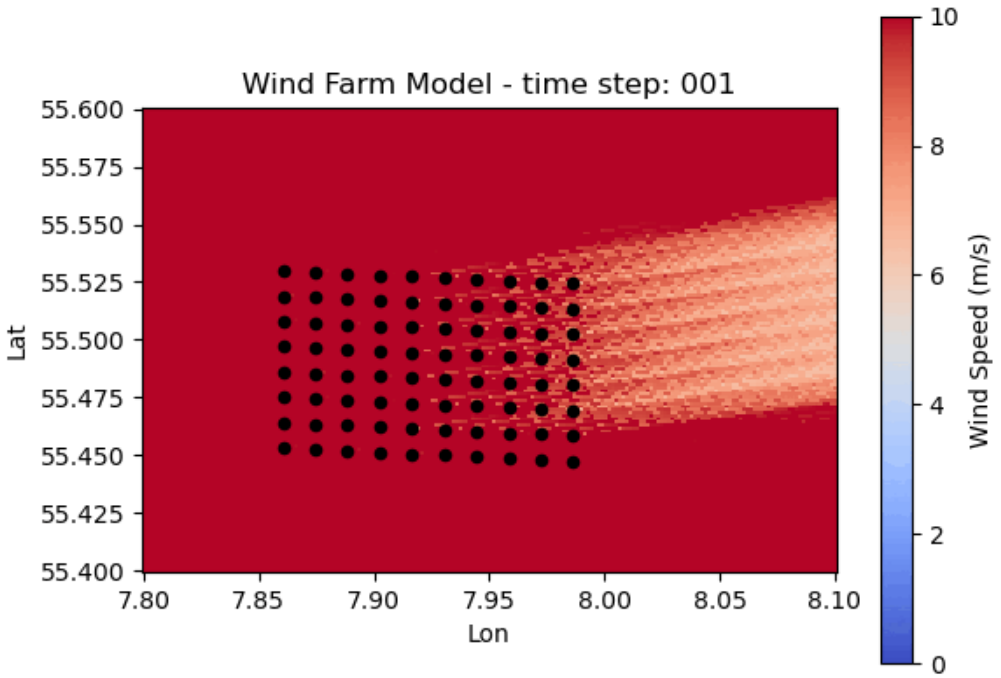


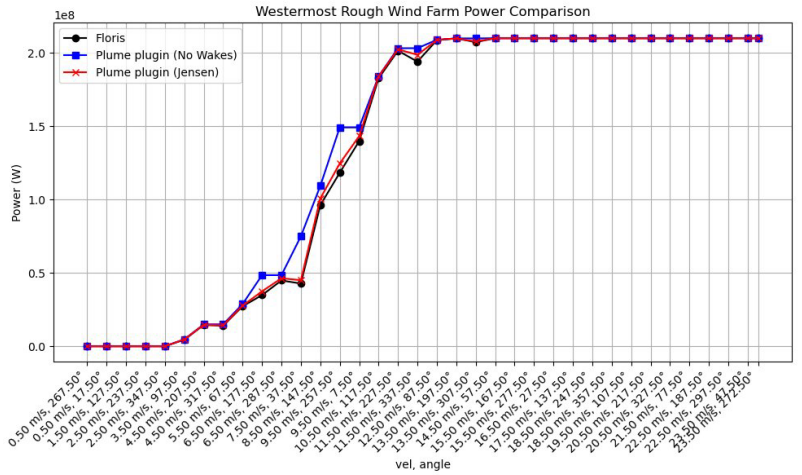
Cells highlight regions of extreme winds (>25m/s) on 27/10/2013 (ERA5 data)

PLUME

Wind Farm Model Plugin

- Simulates Wind Turbines and wake interactions
- Wake model currently implemented: “Jensen”





Validation: “Floris” model @Westermost Rough wind farm

PLUME

Plume is highly collaborative

Got a use-case?

Speak to us!