Interoperability of EO cloud computing services

openEO

A uniform communication strategy between users and EO service providers

TECHNISCHE UNIVERSITÄT WIEN

vito

WAGENINGEN UNIVERSITY & RESEARCH

SINERGISE

Joint Research Centre JRC

Westfälische Wilhelms-Universität Münster

eurac research

mundialis

SOLENIX

eodc

ESA Φ-week | 11/09/2019 | ESRIN

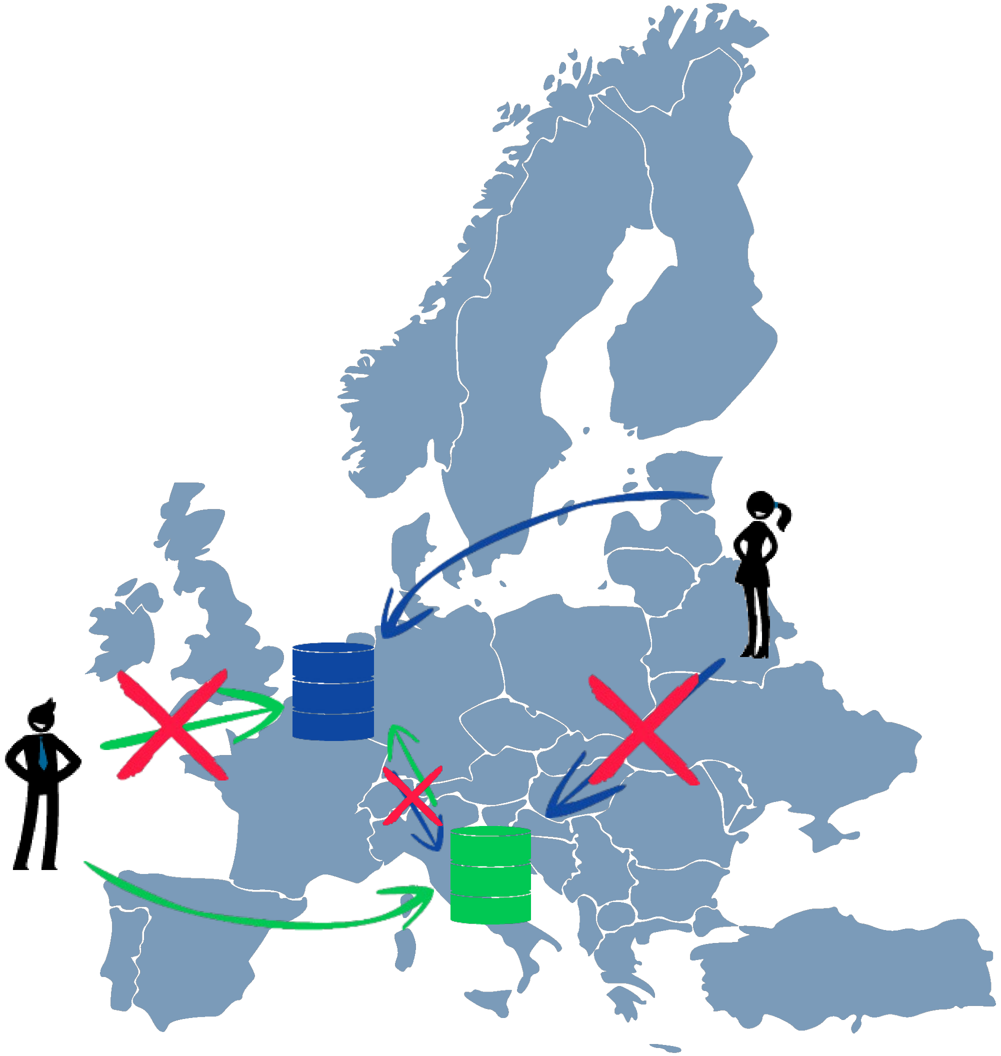
Introduction into openEO

Matthias Schramm

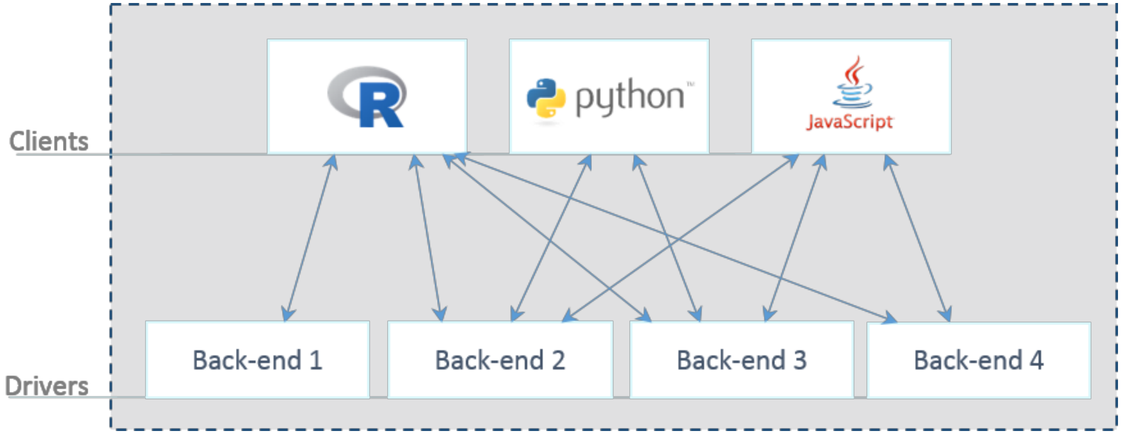
TECHNISCHE UNIVERSITÄT WIEN

ESA Φ-week | 11/09/2019 | ESRIN

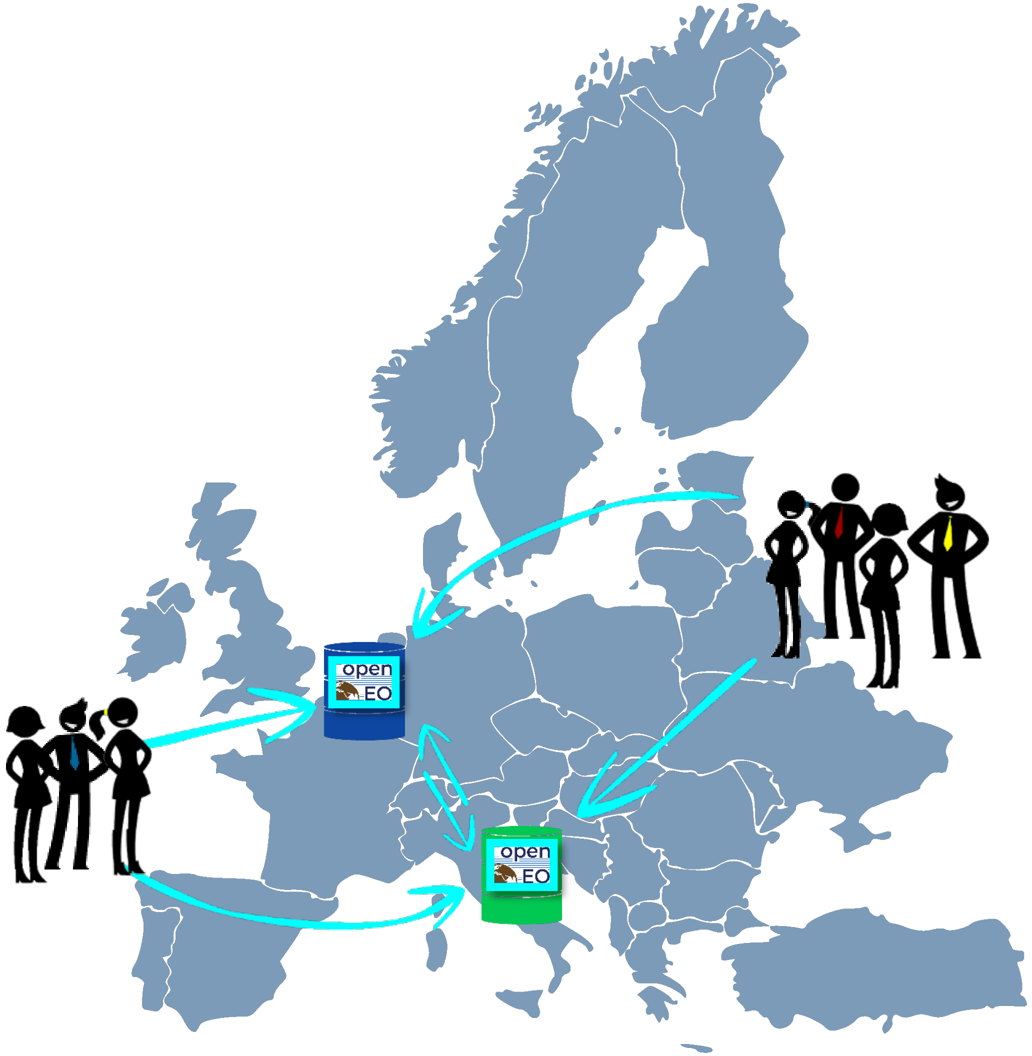
Why openEO?



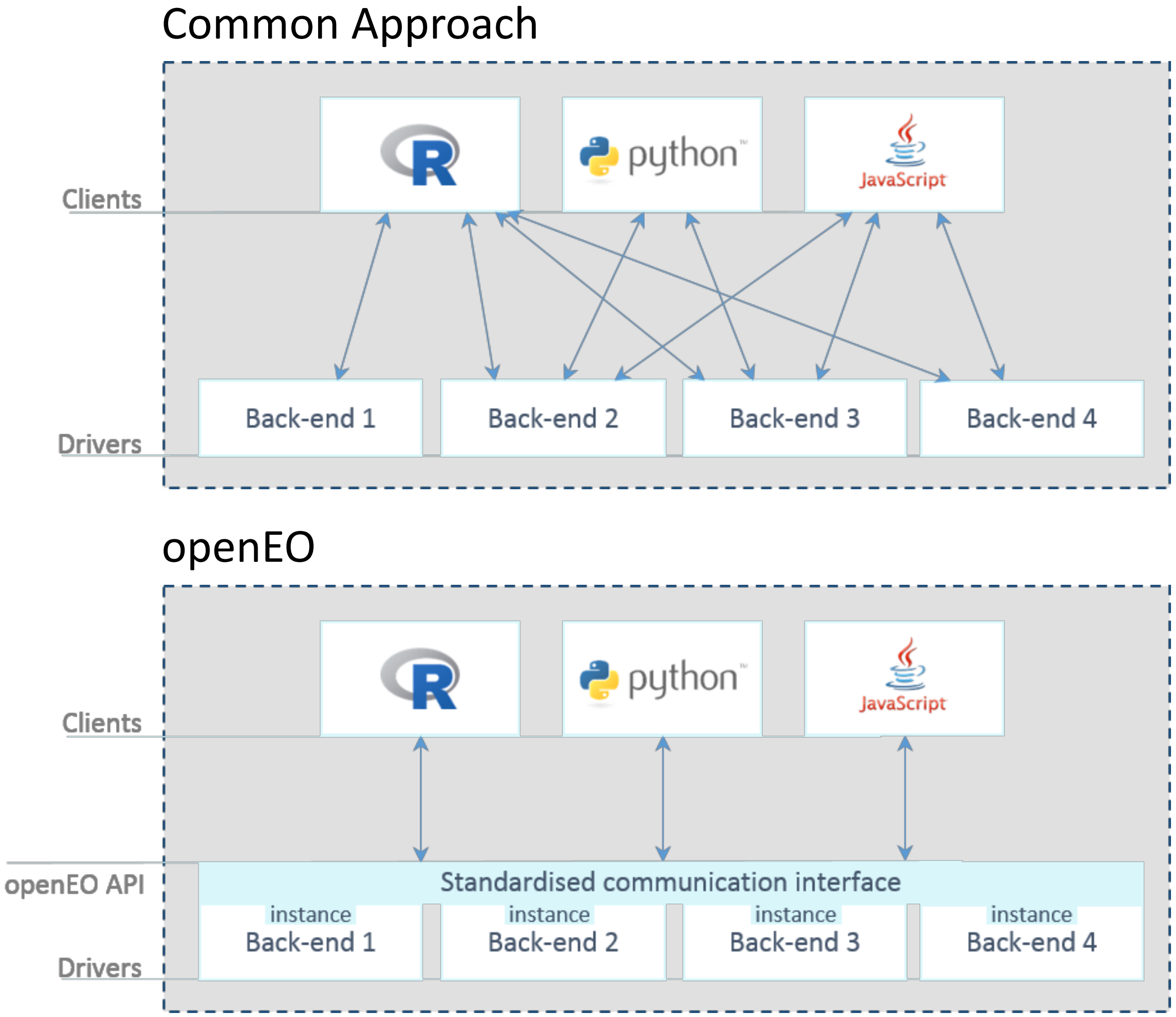
Common Approach



Why openEO?

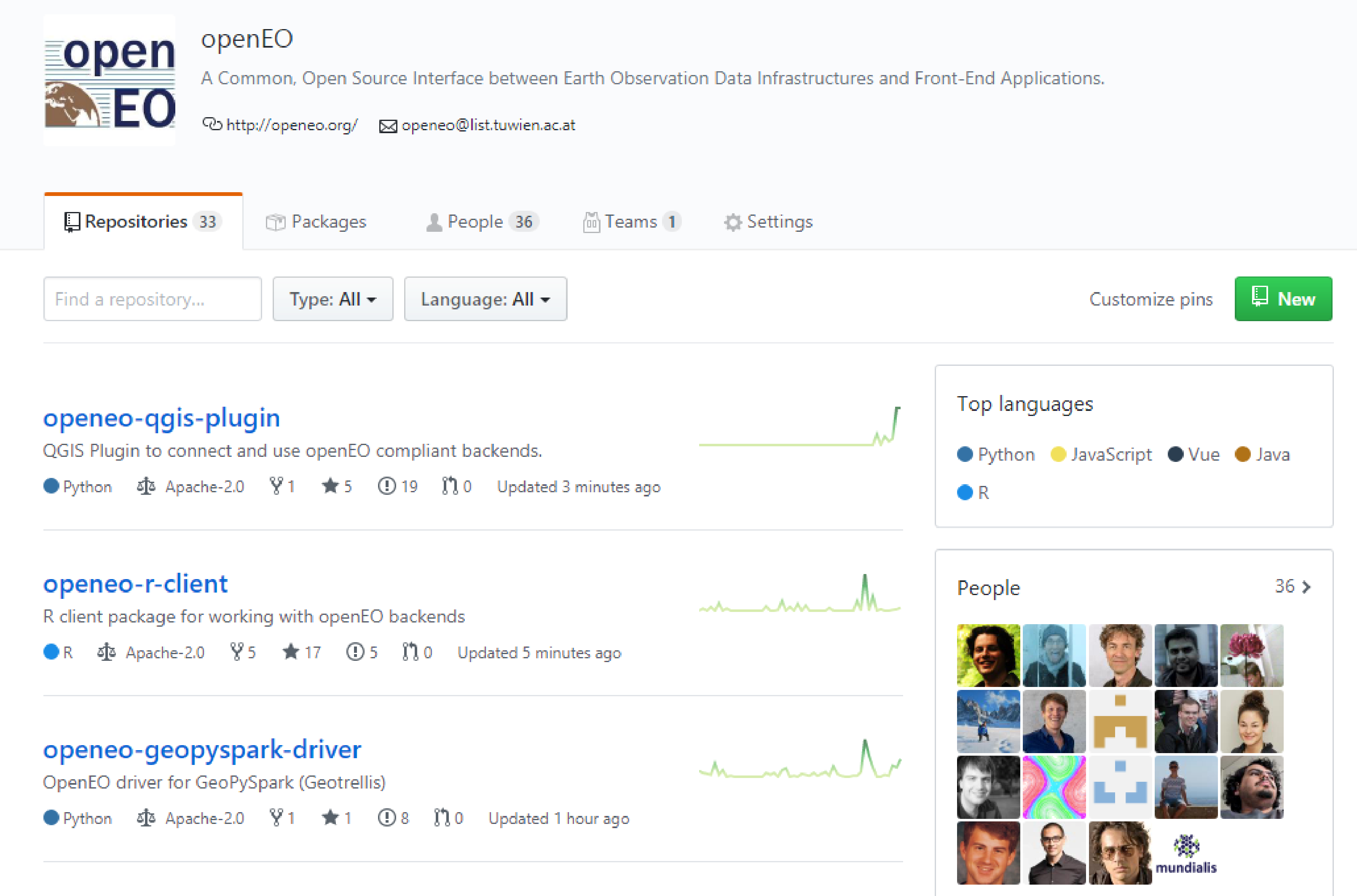


open EO
open EO



Common Approach
Clients
R
python
JavaScript
Drivers
Back-end 1
Back-end 2
Back-end 3
Back-end 4
openEO
Clients
R
python
JavaScript
openEO API
Standardised communication interface
instance
Back-end 1
instance
Back-end 2
instance
Back-end 3
instance
Back-end 4
Drivers

GitHub repository



openEO

A Common, Open Source Interface between Earth Observation Data Infrastructures and Front-End Applications.

http://openeo.org/ openeo@list.tuwien.ac.at

openeo-qgis-plugin

QGIS Plugin to connect and use openEO compliant backends.

● Python Apache-2.0 1 ★ 5 19 0 Updated 3 minutes ago

openeo-r-client

R client package for working with openEO backends

● R Apache-2.0 5 ★ 17 5 0 Updated 5 minutes ago

openeo-geopyspark-driver

OpenEO driver for GeoPySpark (Geotrellis)

● Python Apache-2.0 1 ★ 1 8 0 Updated 1 hour ago

Top languages

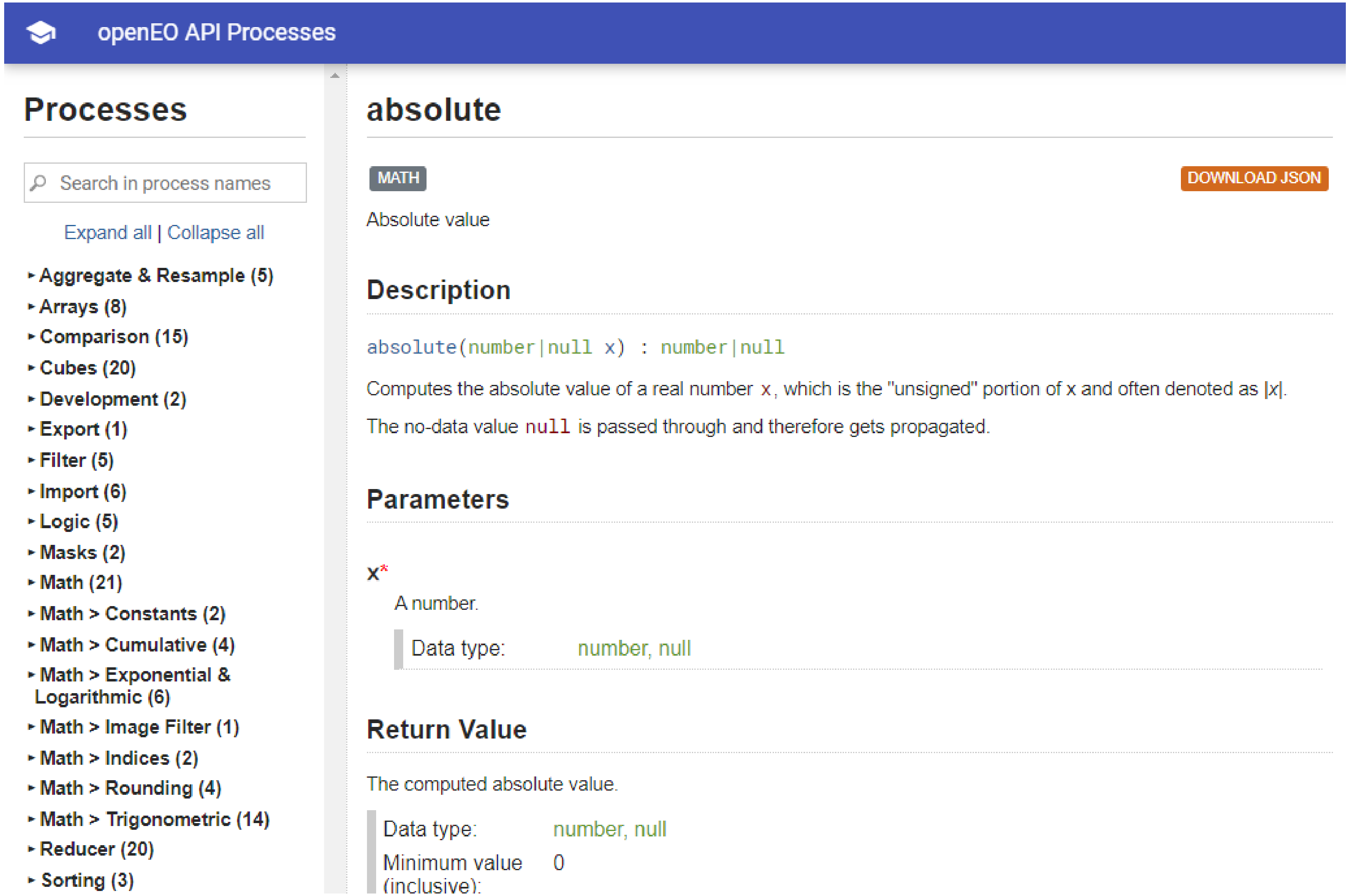
● Python ● JavaScript ● Vue ● Java ● R

People 36

mundialis

https://github.com/Open-EO

Available processes



openEO API Processes

Processes

Search in process names

Expand all | Collapse all

- Aggregate & Resample (5)
- Arrays (8)
- Comparison (15)
- Cubes (20)
- Development (2)
- Export (1)
- Filter (5)
- Import (6)
- Logic (5)
- Masks (2)
- Math (21)
- Math > Constants (2)
- Math > Cumulative (4)
- Math > Exponential & Logarithmic (6)
- Math > Image Filter (1)
- Math > Indices (2)
- Math > Rounding (4)
- Math > Trigonometric (14)
- Reducer (20)
- Sorting (3)

absolute

MATH

DOWNLOAD JSON

Absolute value

Description

```
absolute(number|null x) : number|null
```

Computes the absolute value of a real number `x`, which is the "unsigned" portion of x and often denoted as |*x*|.

The no-data value `null` is passed through and therefore gets propagated.

Parameters

x*

A number.

Data type:	number, null

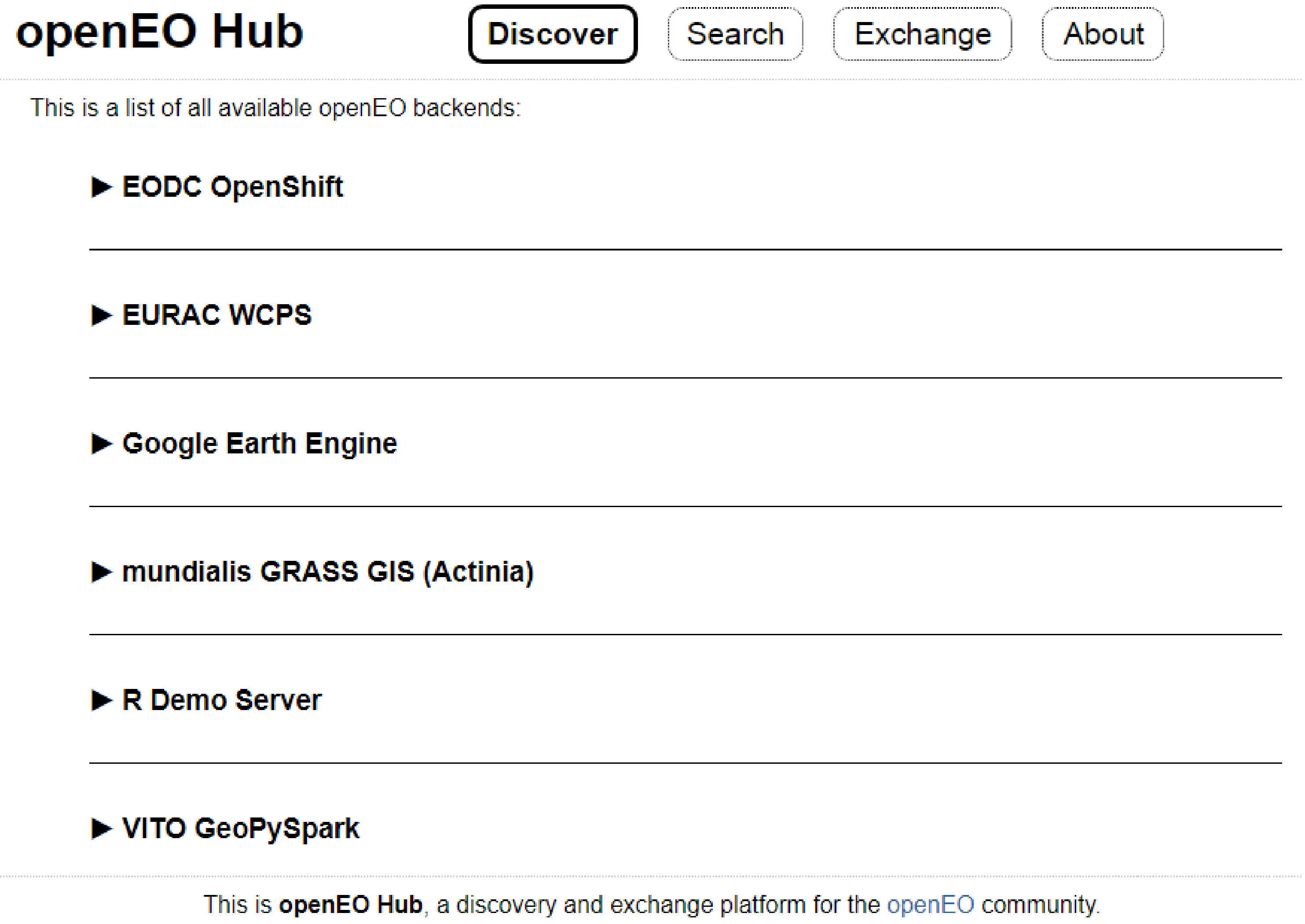
Return Value

The computed absolute value.

Data type:	number, null
Minimum value (inclusive):	0

https://open-eo.github.io/openeo-api/processreference/

Available Backends



openEO Hub

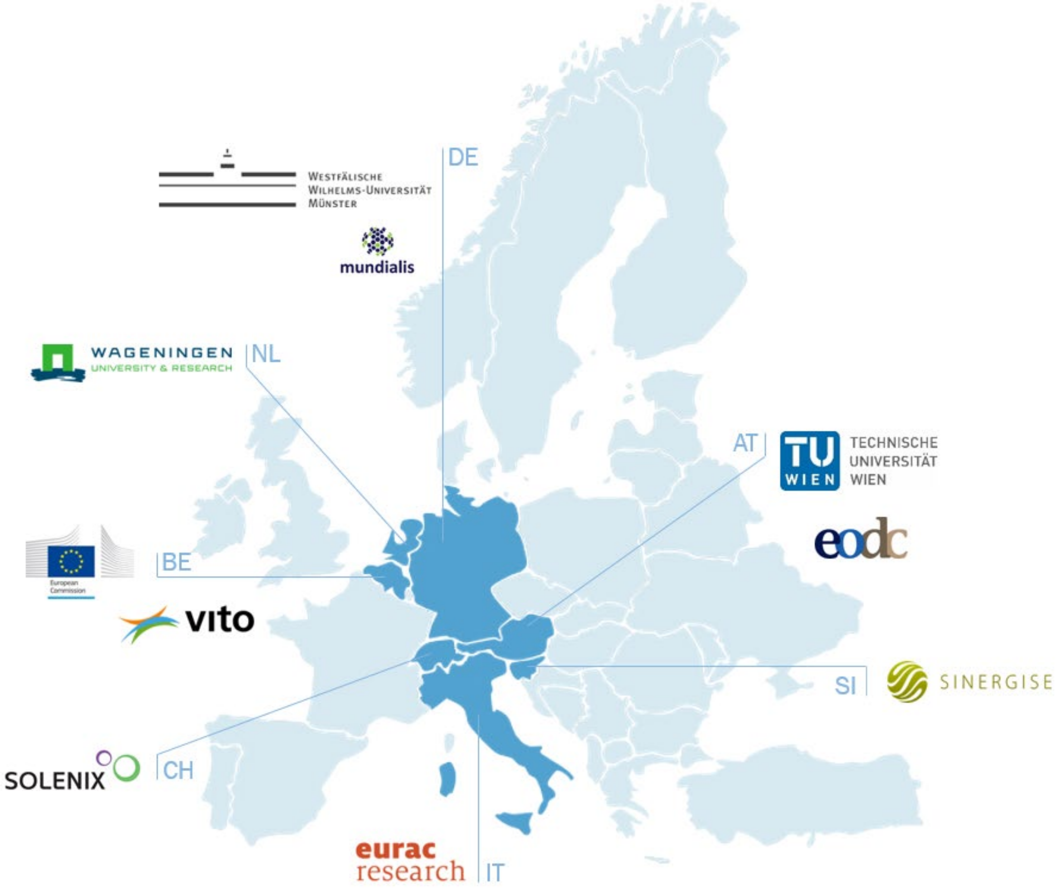
This is a list of all available openEO backends:

- ▶ **EODC OpenShift**
- ▶ **EURAC WCPS**
- ▶ **Google Earth Engine**
- ▶ **mundialis GRASS GIS (Actinia)**
- ▶ **R Demo Server**
- ▶ **VITO GeoPySpark**

This is **openEO Hub**, a discovery and exchange platform for the openEO community.

http://hub.openeo.org/

openEO Consortium / Contact


DE
Westfälische Wilhelms-Universität Münster
mundialis
NL
Wageningen University & Research
AT
TU Wien Technische Universität Wien
eodc
BE
European Commission
vito
SI
Sinergise
SOLENIX
CH
eurac research
IT

- http://openeo.org/
- openEO@list.tuwien.ac.at
- https://github.com/Open-EO
- @open_EO
- https://www.youtube.com/channel/UCMJQil8j9sHBQkcSISaEsvQ
- https://www.researchgate.net/project/openEO
- https://openeo-chat.eodc.eu/channel/public
- zenodo https://zenodo.org/communities/openeo/

Agenda

1st session: user perspective

- Technical overview
- Live demonstrations:
  - Python, R
  - User Defined Functions
  - Web Editor, QGIS
- Hackathon

2nd session: backend perspective

- openEO architecture, standards
- Backend architectures
- Live demonstrations
  - Python client / User Defined Functions on Backend

Technical overview and Processes

Edzer Pebesma, Matthias Mohr

ifgi
Institute for Geoinformatics
University of Münster

ESA Φ-week | 11/09/2019 | ESRIN

Why?

Domain scientists want to get something done, quickly, they

- are not interested in how clouds work, how resources are managed, or how data are stored and accessed
- are interested in *which data* are available, and *what* they can do with it
- want to be able to develop rapidly, and have a system that is responsive

Blueprint for such a system: Google Earth Engine

The openEO API

- Was developed from scratch, as there was no such thing
- Uses OpenAPI (formerly: swagger): developer-friendly!
- Adopts the model of a ***cube view***:
  - Regardless how image collections are stored, they are analysed on a regular grid, in some coordinate reference system, and typically using some regular time intervals
  - Operations can be chained, using a functional programming paradigm
  - Adopt lazy evaluation: only compute pixels when shown, or downloaded
  - Entire dimensions can be *reduced*, e.g. {R,NIR} ⇒ NDVI; {time series} ⇒ trend slope
  - *Aggregation* computes summaries over groups (regions, or time periods)
- Tries to not reinvent anything available (authentication, user management, payment, file formats, CRS, ...)

openEO API endpoints

- `/collections` : get image collections, describe each (STAC/WFS)
- `/processes` : get available processes, describe each
- `/jobs` : manage jobs
- `/subscriptions` : get notified on changed job status
- `/credentials` : manage authentication
- `/files` : manage user files
- `/validation` : validate a process graph
- `/result` : post process graph, get results synchronously
- `/process_graphs` : list graphs, or store a new one
- `/services` : list services available to user, or create a new one

openEO API processes (140+)

- Data cube model: $(\mathrm{dim}_1, \mathrm{dim}_2, \ldots, \mathrm{dim}_n) \Rightarrow \mathrm{value}$
- … means that a pixel is a scalar value
- Math functions: `abs, sqrt, etc; comparison: lt, eq, ...`
- Array functions: `min, max, sort, order, first, last, ...`
- Cube functions: `apply, apply_dimension, reduce, merge_cubes`
- `aggregate_temporal, aggregate_spatial`
- Mask functions
- User-defined functions: `run_udf, run_udf_externally`

openEO API: User-defined functions

What if a back-end:

- doesn’t provide a certain process, or
- it is prohibitively complex or inefficient to translate the model into a process graph (e.g. AI/ML models)?

User-defined functions let users:

- specify the model in any code (e.g. Python, R)
- submit it as part of the process graph, and have it deployed as a reducer

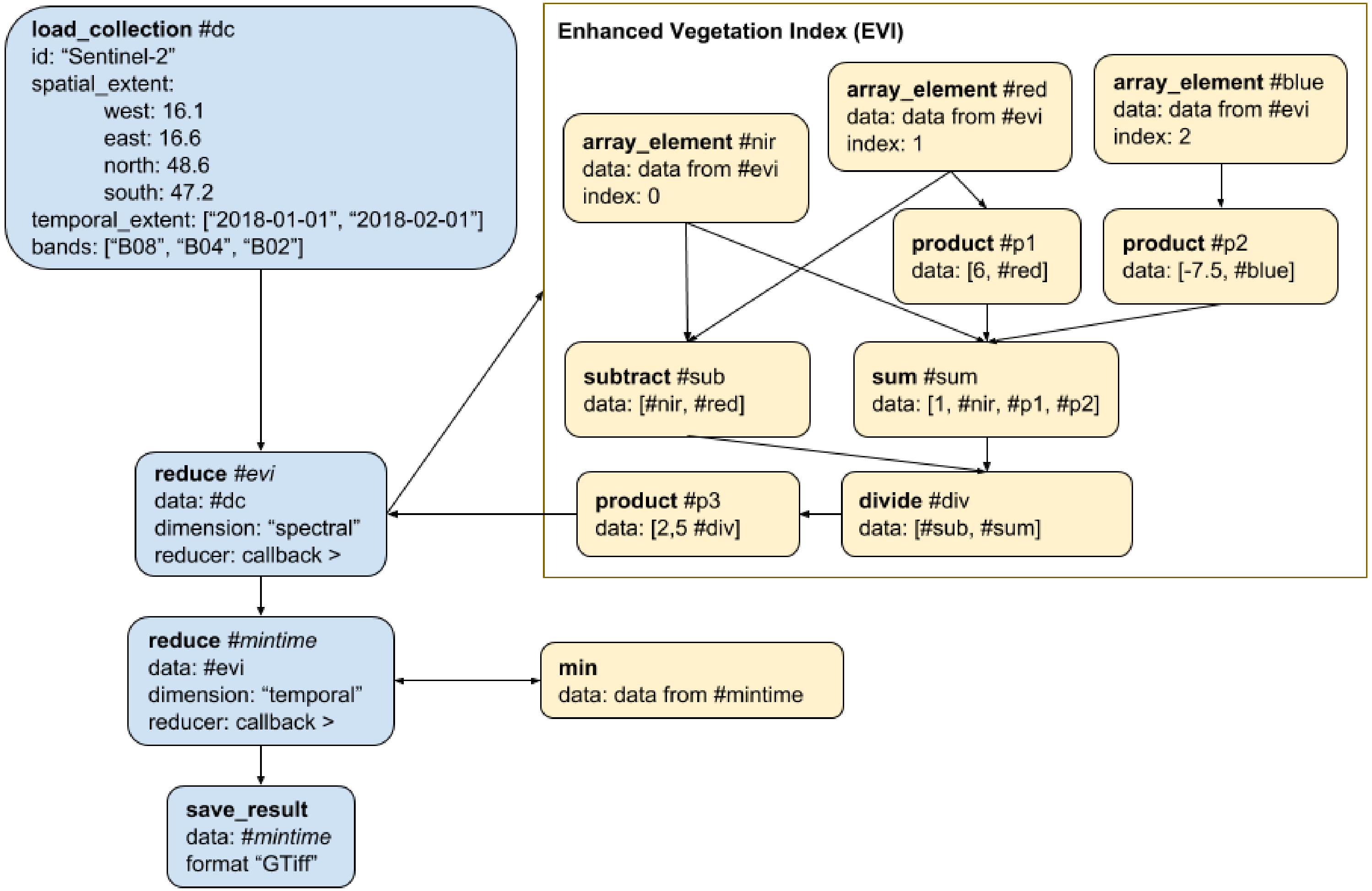
This makes the openEO API extremely flexible and powerful!

(A special openEO API defines communication with externally run UDF engines)

openEO API process graphs

- pièce de résistance
- Expresses what is going to be computed, from what, and how
- Contains nodes (sub-graphs): partial results that are re-used
- May be incomplete, as a computing recipe, and leave e.g. spatial extent and resolution up to the web service generated

openEO API process graphs



load_collection #dc
id: “Sentinel-2”
spatial_extent:
west: 16.1
east: 16.6
north: 48.6
south: 47.2
temporal_extent: [“2018-01-01”, “2018-02-01”]
bands: [“B08”, “B04”, “B02”]

reduce *#evi*
data: #dc
dimension: “spectral”
reducer: callback >

reduce *#mintime*
data: #evi
dimension: “temporal”
reducer: callback >

save_result
data: *#mintime*
format “GTiff”

Enhanced Vegetation Index (EVI)

array_element #nir
data: data from #evi
index: 0

array_element #red
data: data from #evi
index: 1

array_element #blue
data: data from #evi
index: 2

product #p1
data: [6, #red]

product #p2
data: [-7.5, #blue]

subtract #sub
data: [#nir, #red]

sum #sum
data: [1, #nir, #p1, #p2]

product #p3
data: [2,5 #div]

divide #div
data: [#sub, #sum]

min
data: data from #mintime

openEO: Standards and Specifications

Edzer Pebesma, Matthias Mohr

ifgi
Institute for Geoinformatics
University of Münster

ESA Φ-week | 11/09/2019 | ESRIN

openEO: Standards and Specifications

- Used existing standards where possible
- API: REST/JSON, OpenAPI, AsyncAPI
- Authentication: OpenID Connect (extendable)
- Projections: PROJ, WKT2, EPSG codes
- File formats: Aligned with GDAL, not bound to a specific file format
- Well-known discovery, JSON Schema, GeoJSON and more RFC and ISO standards

Relation to OGC Standards

- September 2017: No OGC APIs yet
  - old-fashioned standards, hard to combine
- Now: OGC APIs are evolving
- Contributing to STAC, OGC API - Features (WFS) & Catalogs (CSW), …

- API uses STAC / OGC API - Features (WFS)
- Compliant to OGC API Commons
- Processing: not WPS (doesn’t support chaining)
- Results: Exposing web services possible
  - WMS, WMTS, CSW or corresponding OGC APIs

The backends of openEO – and how to become one

Alexander Jacob, Jeroen Dries, Luca Foresta, Markus Neteler, Matthias Mohr

eurac research

vito

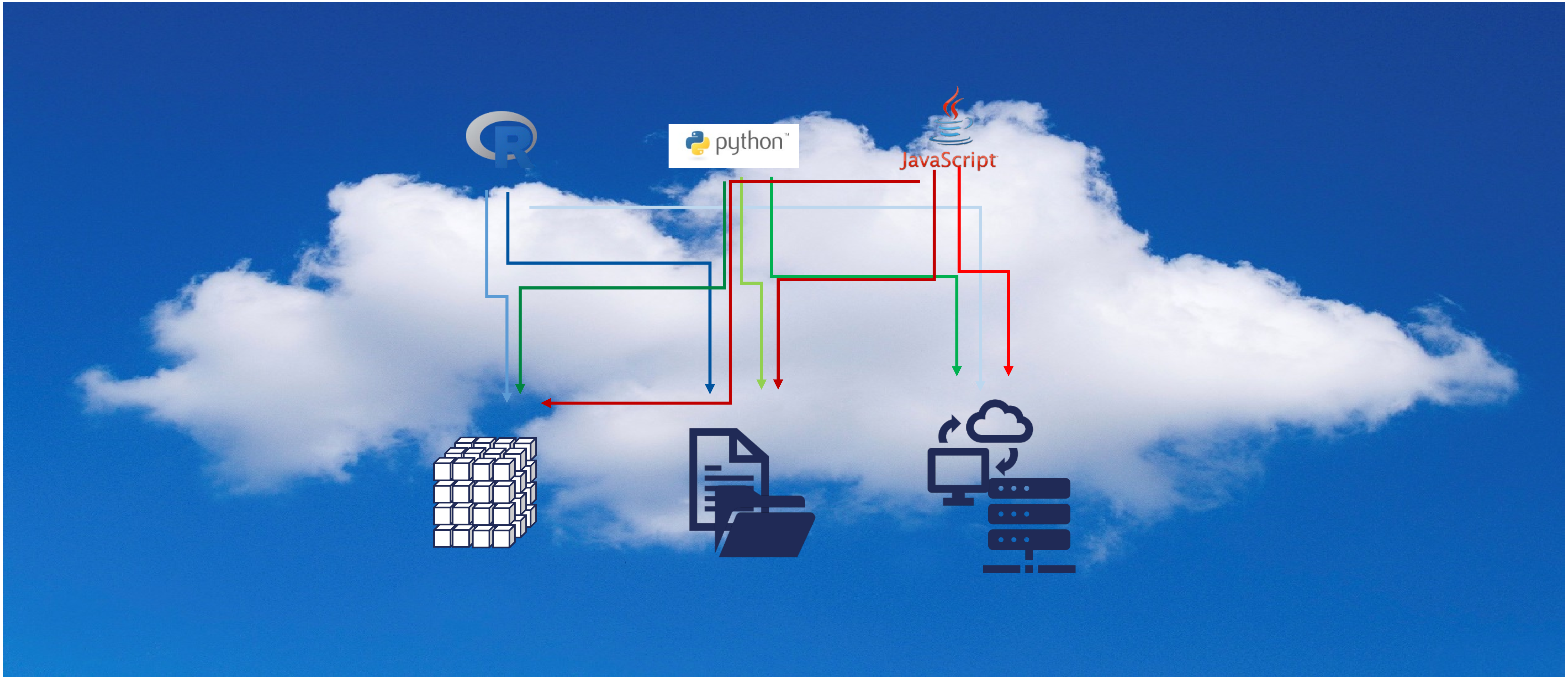
eodc

mundialis

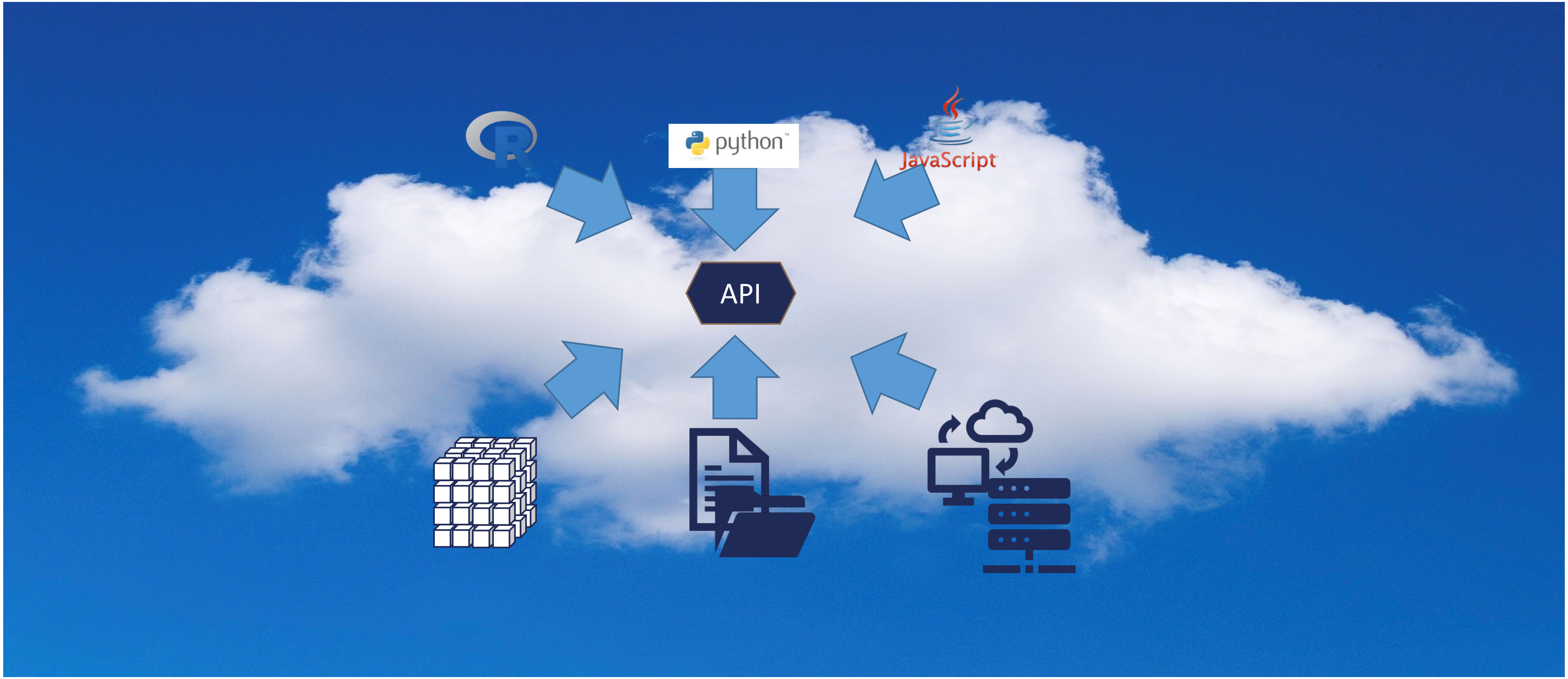
ifgi
Institute for Geoinformatics
University of Münster

ESA Φ-week | 11/09/2019 | ESRIN

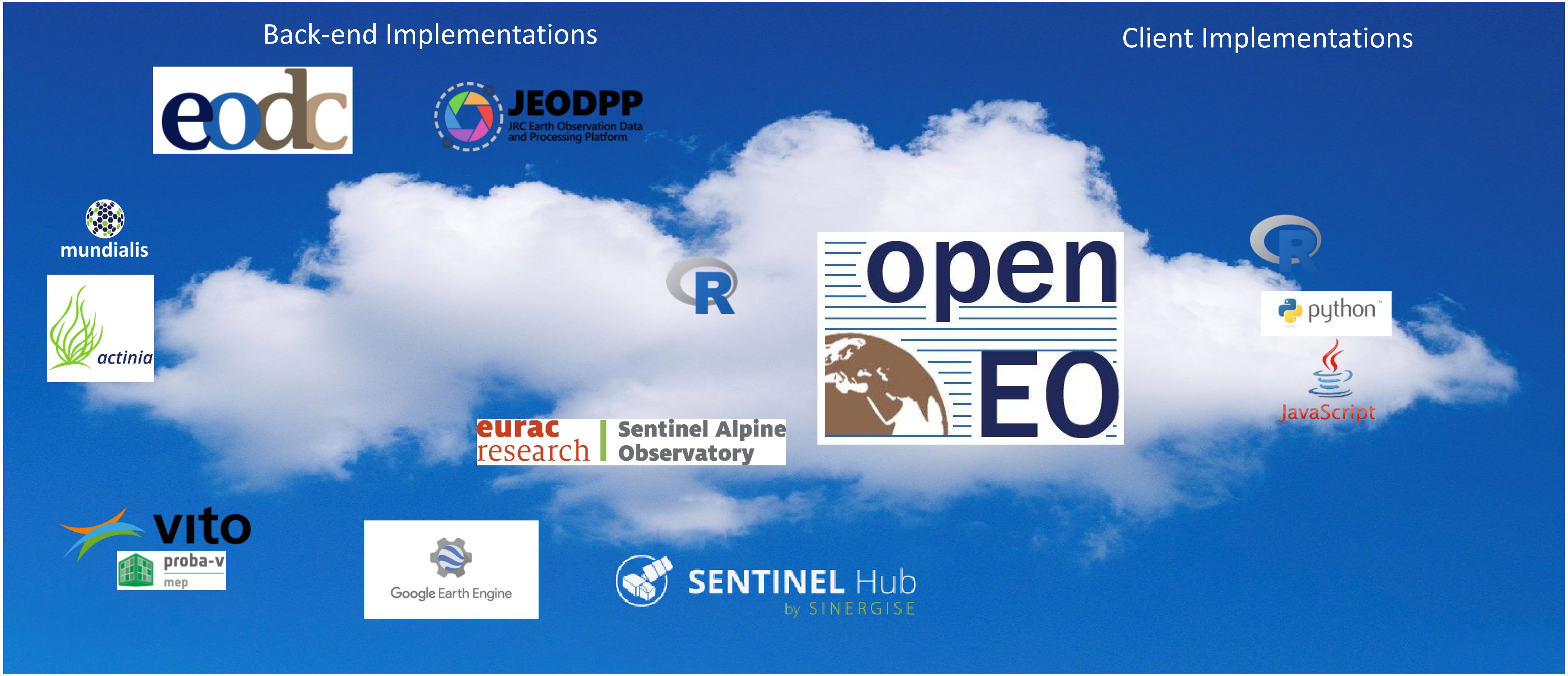
Why do we need openEO?



Why do we need openEO?



The Implementation of openEO

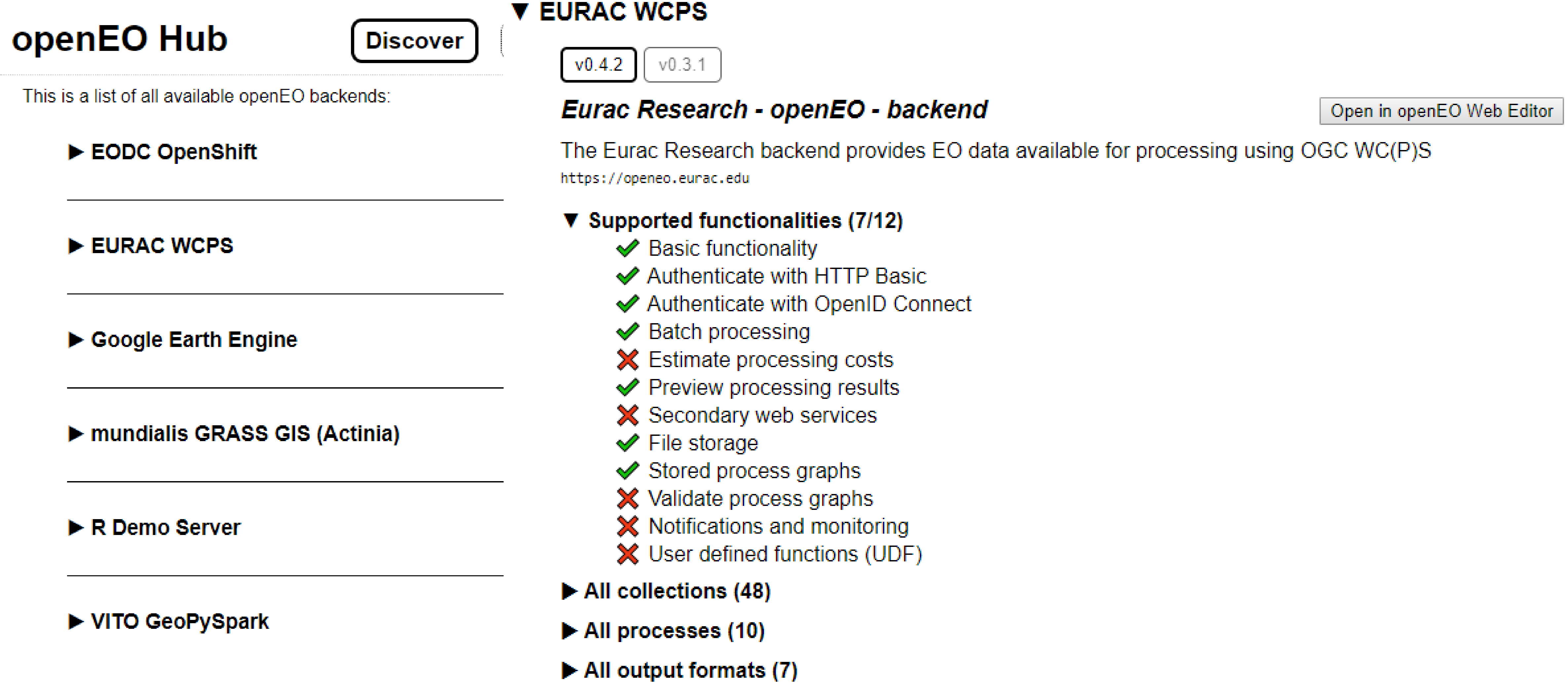


Back-end Implementations

Client Implementations

eodc
JEODPP
JRC Earth Observation Data and Processing Platform
mundialis
actinia
R
openEO
eurac research | Sentinel Alpine Observatory
vito
proba-v mep
Google Earth Engine
SENTINEL Hub by SINERGISE
R
python
JavaScript

The Implementations of openEO



openEO Hub

Discover

This is a list of all available openEO backends:

- ▶ EODC OpenShift
- ▶ EURAC WCPS
- ▶ Google Earth Engine
- ▶ mundialis GRASS GIS (Actinia)
- ▶ R Demo Server
- ▶ VITO GeoPySpark

▼ EURAC WCPS

v0.4.2 v0.3.1

Eurac Research - openEO - backend

Open in openEO Web Editor

The Eurac Research backend provides EO data available for processing using OGC WC(P)S

`https://openeo.eurac.edu`

▼ Supported functionalities (7/12)

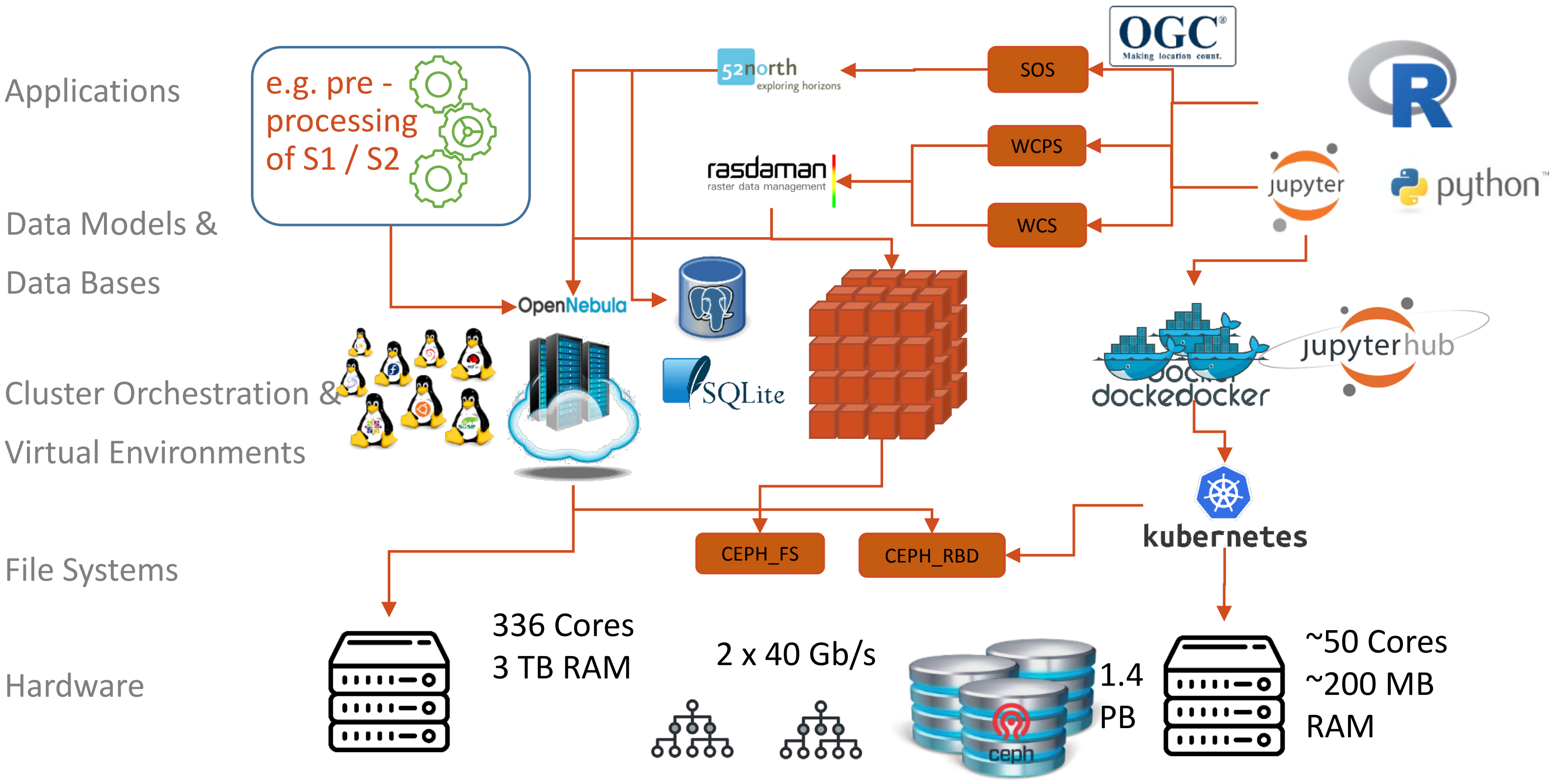
- ✓ Basic functionality
- ✓ Authenticate with HTTP Basic
- ✓ Authenticate with OpenID Connect
- ✓ Batch processing
- ✗ Estimate processing costs
- ✓ Preview processing results
- ✗ Secondary web services
- ✓ File storage
- ✓ Stored process graphs
- ✗ Validate process graphs
- ✗ Notifications and monitoring
- ✗ User defined functions (UDF)

▶ All collections (48)

▶ All processes (10)

▶ All output formats (7)

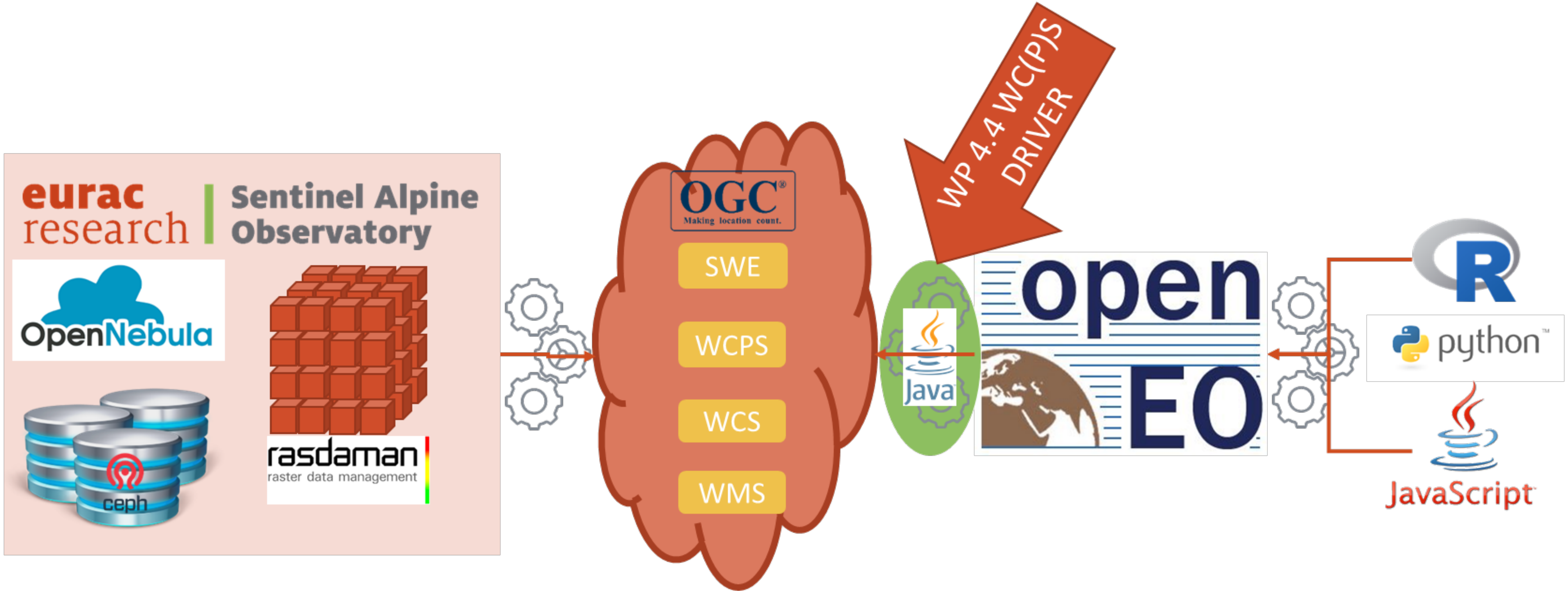
WCPS backend (EURAC)



Grant Agreement 776242

WCPS backend (EURAC)

https://github.com/Open-EO/openeo-wcps-driver



WCPS backend (EURAC)

Based on swagger-jersey2-jaxrs for rest API implementation.

Sqlite for openEO related DB

→ Batch job management, storing of process graphs

GDAL for image operations and coordinate transformations

JJWT for openID connect implementation

→ Linked to Microsoft azure for authentication

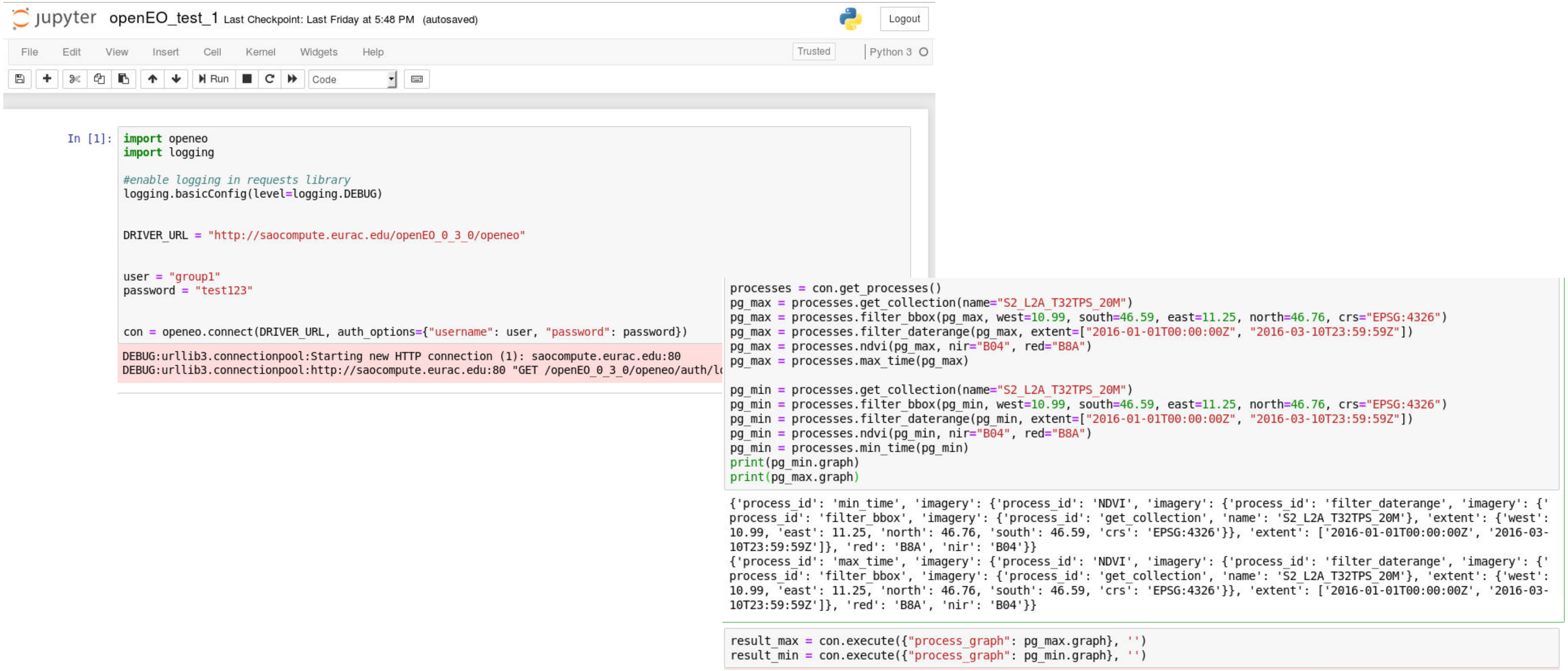
Packaged as web archive using maven

→ Deployable on any java capable web container (e.g. tomcat or jetty)

Configuration:

- Properties File
  - WCPS endpoint
  - openEO endpoint
  - Authentication endpoint (for oidc)
  - DB location
  - TMP location
  - Session timings (auth expiry, tmp duration, etc.)
- Setup of Host Environment
  - Centos 7 or Ubuntu 18.04
  - Install Tomcat 7 or later
  - Configure for https
  - Install sqlite (v3) & GDAL (v2.4)
  - Deploy openEO.war
- Setup of proxy server for public access

WCPS backend (EURAC)



In [1]:

```
import openeo
import logging

#enable logging in requests library
logging.basicConfig(level=logging.DEBUG)


DRIVER_URL = "http://saocompute.eurac.edu/openEO_0_3_0/openeo"


user = "group1"
password = "test123"


con = openeo.connect(DRIVER_URL, auth_options={"username": user, "password": password})
```

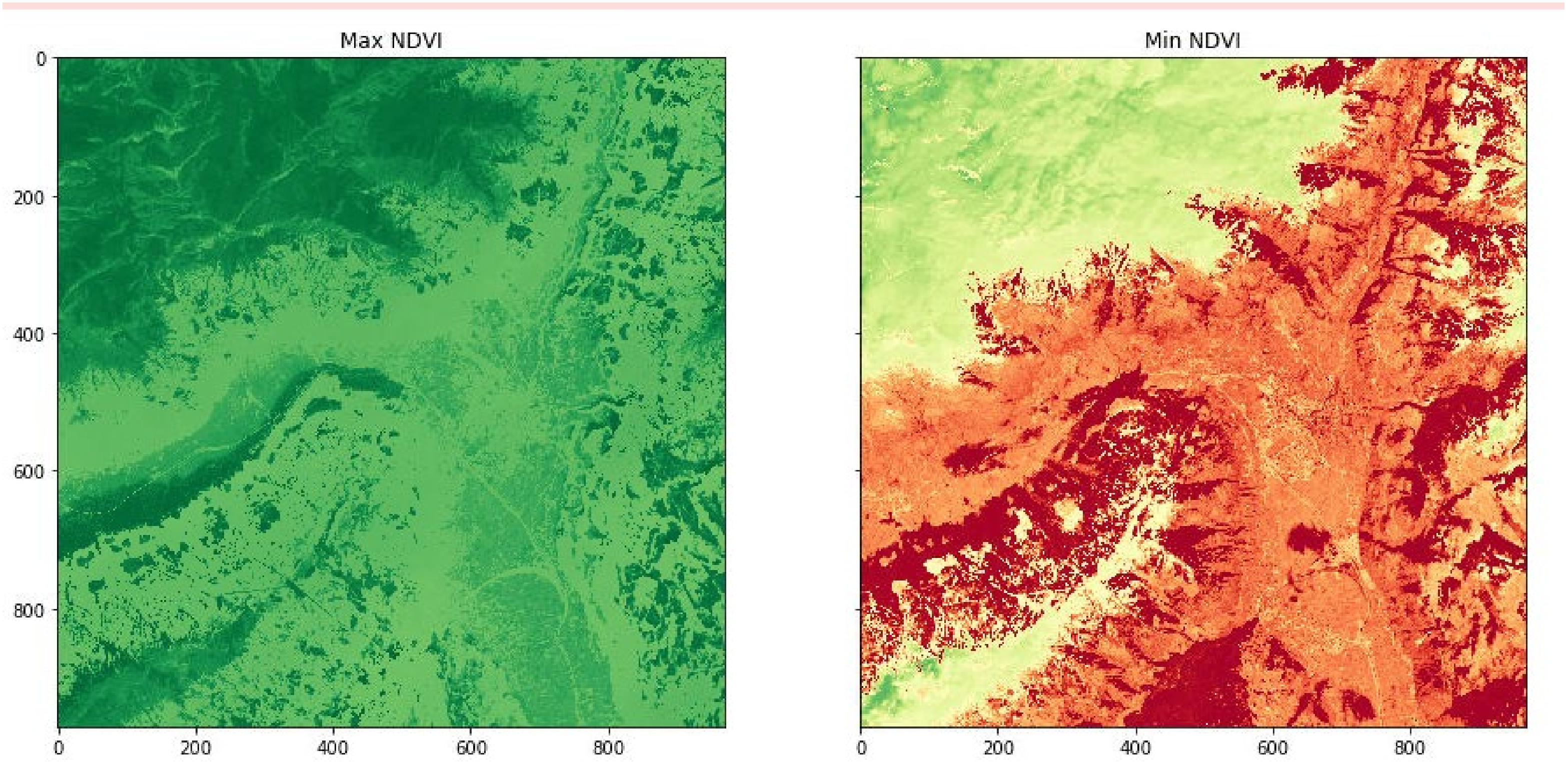
```
DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): saocompute.eurac.edu:80
DEBUG:urllib3.connectionpool:http://saocompute.eurac.edu:80 "GET /openEO_0_3_0/openeo/auth/l
```

```
processes = con.get_processes()
pg_max = processes.get_collection(name="S2_L2A_T32TPS_20M")
pg_max = processes.filter_bbox(pg_max, west=10.99, south=46.59, east=11.25, north=46.76, crs="EPSG:4326")
pg_max = processes.filter_daterange(pg_max, extent=["2016-01-01T00:00:00Z", "2016-03-10T23:59:59Z"])
pg_max = processes.ndvi(pg_max, nir="B04", red="B8A")
pg_max = processes.max_time(pg_max)

pg_min = processes.get_collection(name="S2_L2A_T32TPS_20M")
pg_min = processes.filter_bbox(pg_min, west=10.99, south=46.59, east=11.25, north=46.76, crs="EPSG:4326")
pg_min = processes.filter_daterange(pg_min, extent=["2016-01-01T00:00:00Z", "2016-03-10T23:59:59Z"])
pg_min = processes.ndvi(pg_min, nir="B04", red="B8A")
pg_min = processes.min_time(pg_min)
print(pg_min.graph)
print(pg_max.graph)
```

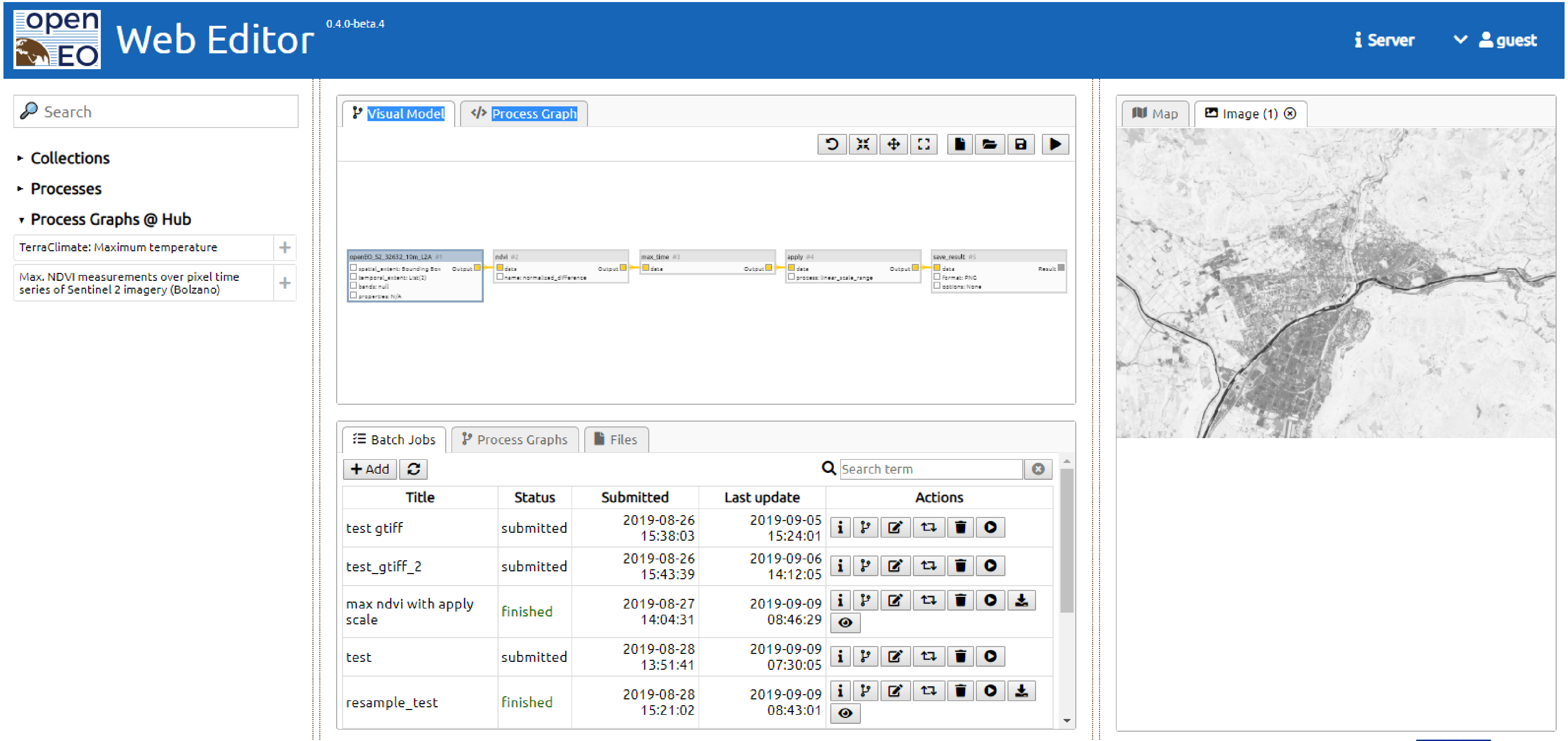
```
{'process_id': 'min_time', 'imagery': {'process_id': 'NDVI', 'imagery': {'process_id': 'filter_daterange', 'imagery': {'
process_id': 'filter_bbox', 'imagery': {'process_id': 'get_collection', 'name': 'S2_L2A_T32TPS_20M'}, 'extent': {'west':
10.99, 'east': 11.25, 'north': 46.76, 'south': 46.59, 'crs': 'EPSG:4326'}}, 'extent': ['2016-01-01T00:00:00Z', '2016-03-
10T23:59:59Z']}, 'red': 'B8A', 'nir': 'B04'}}
{'process_id': 'max_time', 'imagery': {'process_id': 'NDVI', 'imagery': {'process_id': 'filter_daterange', 'imagery': {'
process_id': 'filter_bbox', 'imagery': {'process_id': 'get_collection', 'name': 'S2_L2A_T32TPS_20M'}, 'extent': {'west':
10.99, 'east': 11.25, 'north': 46.76, 'south': 46.59, 'crs': 'EPSG:4326'}}, 'extent': ['2016-01-01T00:00:00Z', '2016-03-
10T23:59:59Z']}, 'red': 'B8A', 'nir': 'B04'}}
```

```
result_max = con.execute({"process_graph": pg_max.graph}, '')
result_min = con.execute({"process_graph": pg_min.graph}, '')
```

WCPS backend (EURAC)

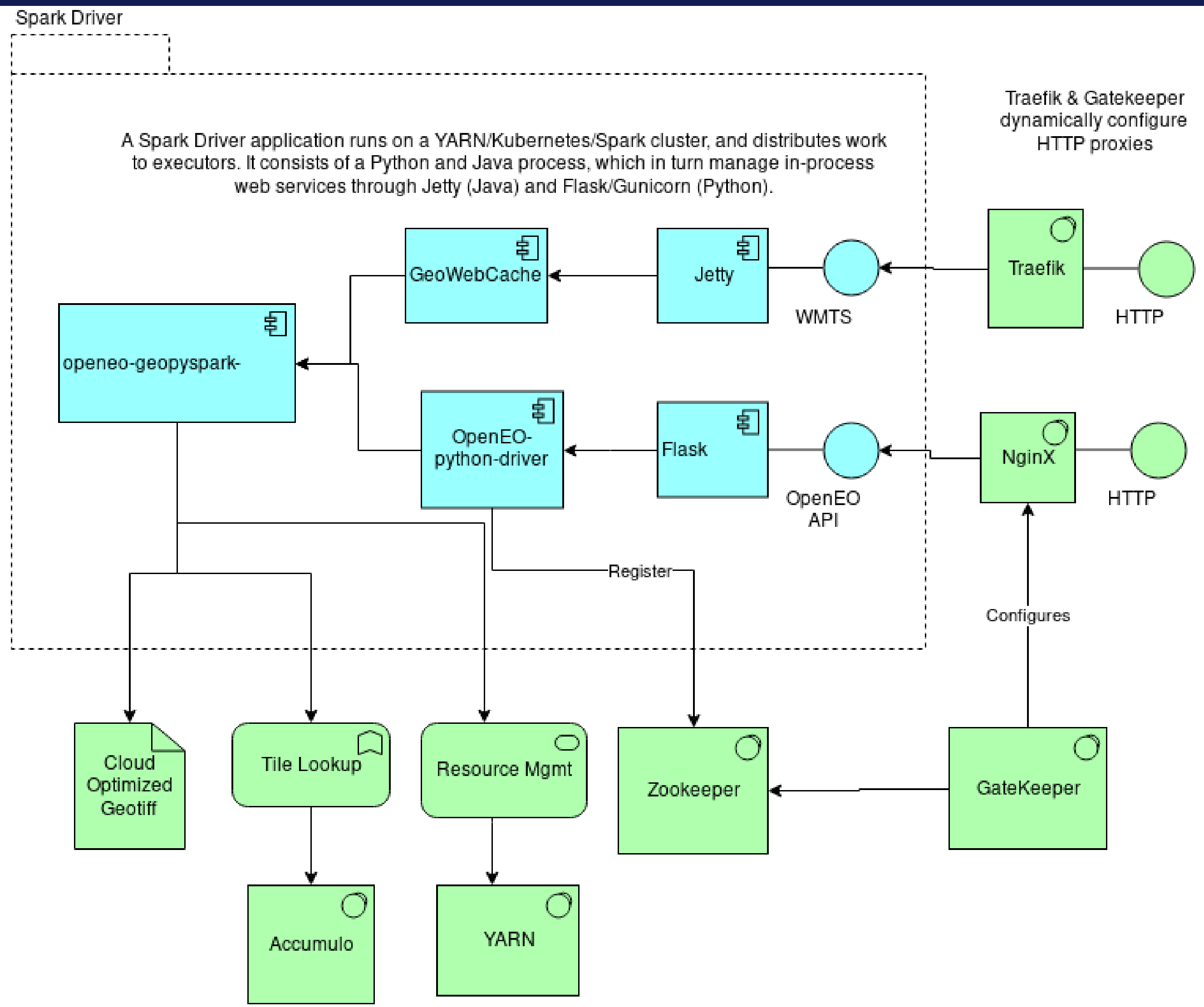


WCPS backend (EURAC)

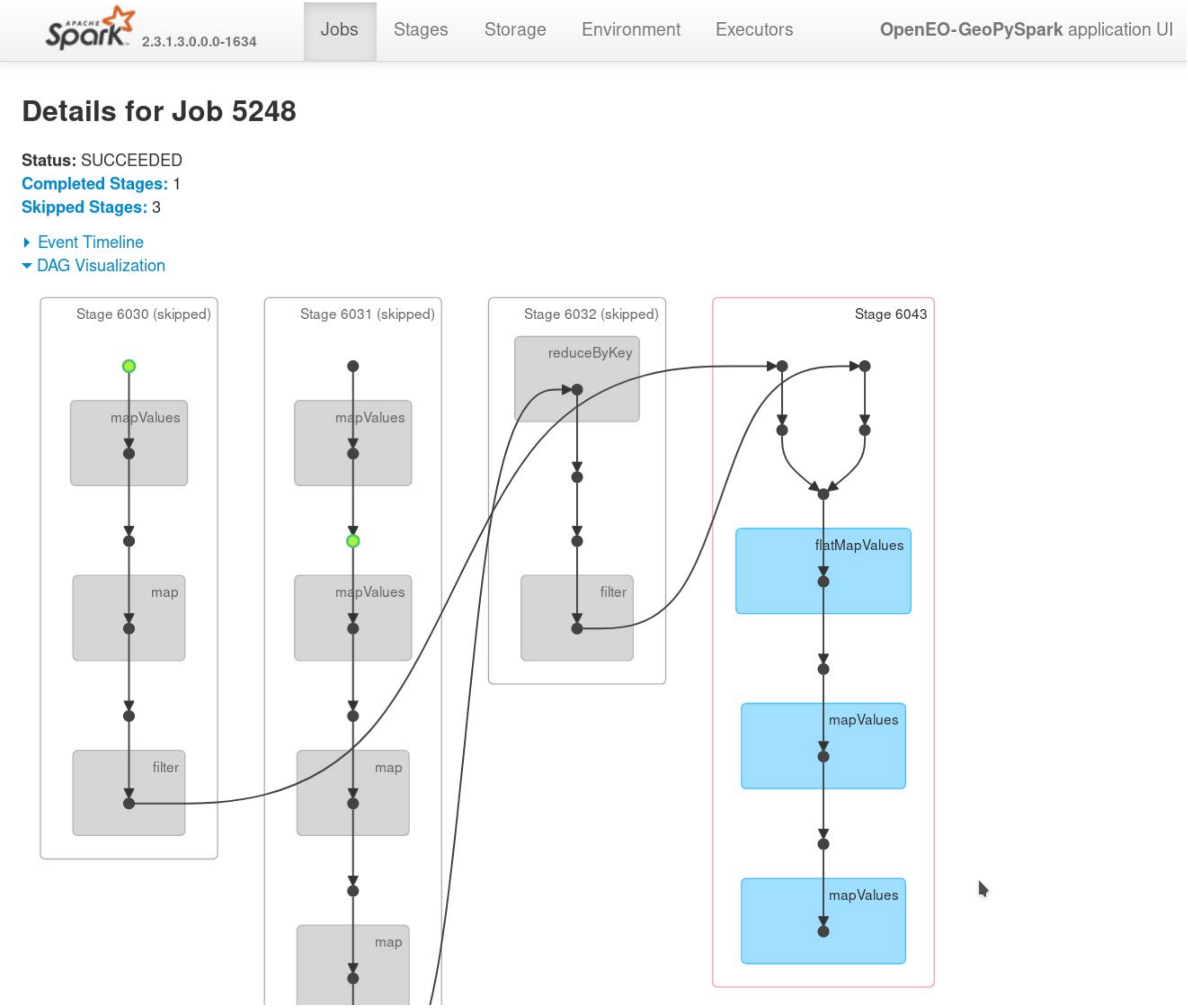


Title	Status	Submitted	Last update	Actions
test gtiff	submitted	2019-08-26 15:38:03	2019-09-05 15:24:01	
test_gtiff_2	submitted	2019-08-26 15:43:39	2019-09-06 14:12:05	
max ndvi with apply scale	finished	2019-08-27 14:04:31	2019-09-09 08:46:29	
test	submitted	2019-08-28 13:51:41	2019-09-09 07:30:05	
resample_test	finished	2019-08-28 15:21:02	2019-09-09 08:43:01	

Geotrellis/Spark backend (VITO)


Spark Driver
A Spark Driver application runs on a YARN/Kubernetes/Spark cluster, and distributes work to executors. It consists of a Python and Java process, which in turn manage in-process web services through Jetty (Java) and Flask/Gunicorn (Python).
Traefik & Gatekeeper dynamically configure HTTP proxies
GeoWebCache
Jetty
WMTS
Traefik
HTTP
openeo-geopyspark-
OpenEO-python-driver
Flask
OpenEO API
NginX
HTTP
Register
Configures
Cloud Optimized Geotiff
Tile Lookup
Resource Mgmt
Zookeeper
GateKeeper
Accumulo
YARN

Geotrellis/Spark backend (VITO)

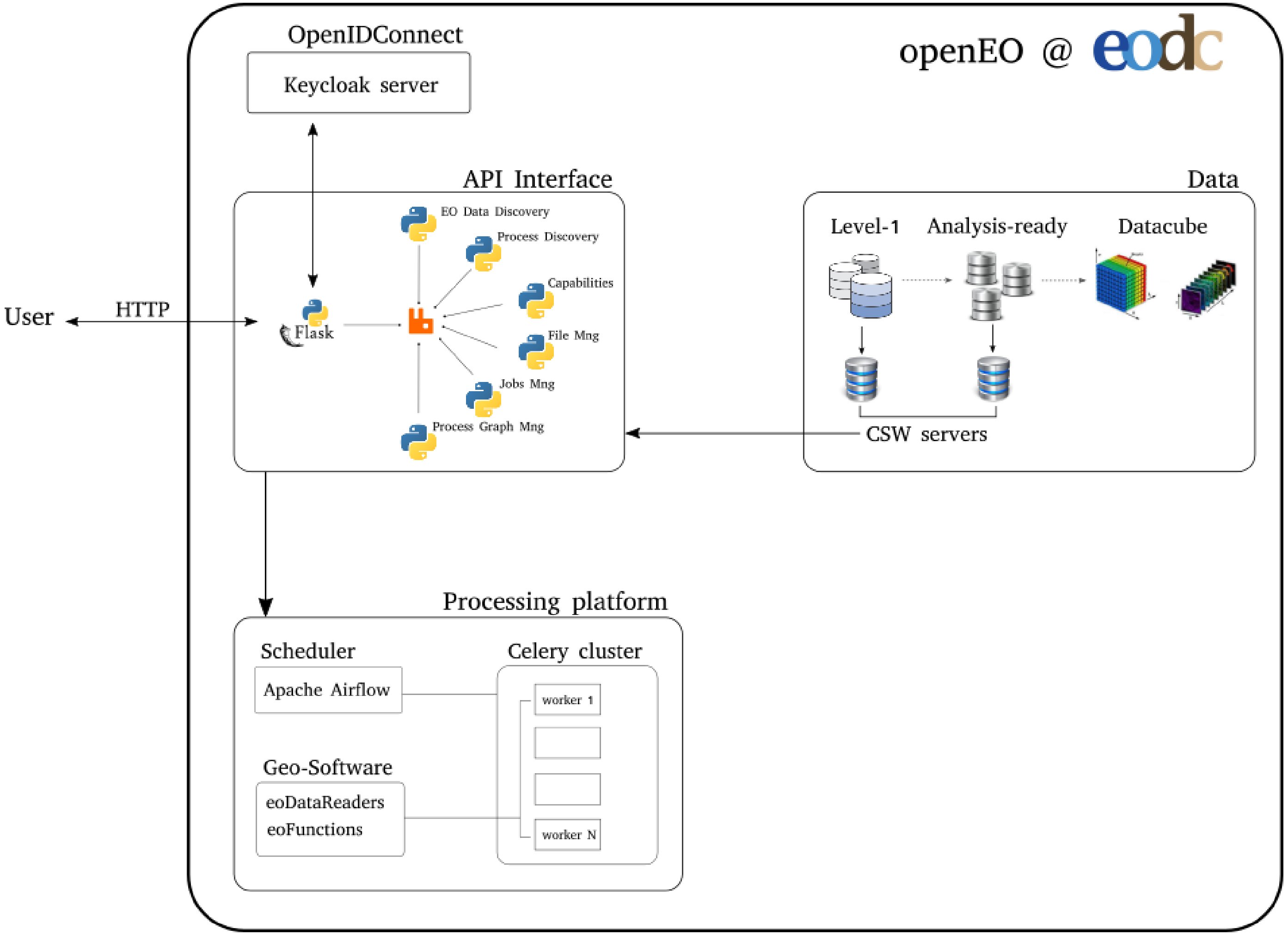


Details for Job 5248

Status: SUCCEEDED
Completed Stages: 1
Skipped Stages: 3

▸ Event Timeline
▾ DAG Visualization

Implementation @ EODC, Overview



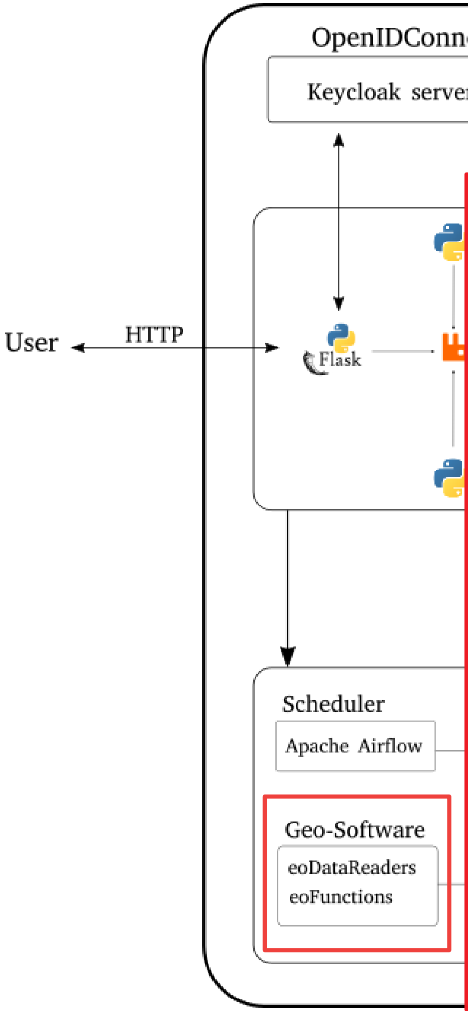
https://github.com/Open-EO/openeo-openshift-driver

https://openeo.eodc.eu

Technologies:

- Flask, Nameko
- Nginx, Gunicorn
- Keycloak
- Apache Airflow
- Docker
- Celery
- CSW
- OSGEO GDAL

Implementation @ EODC, Overview

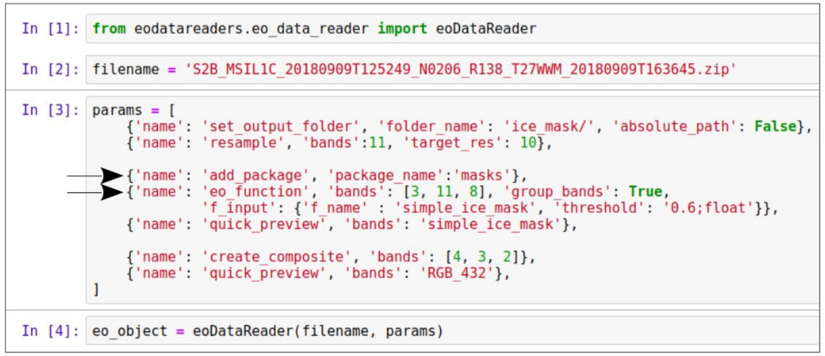


https://github.com/Open-EO/openeo-openshift-driver

https://openeo.eodc.eu

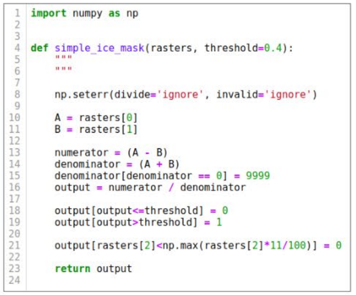
eoDataReaders

in collaboration with TU Wien



```
In [1]: from eodatareaders.eo_data_reader import eoDataReader

In [2]: filename = 'S2B_MSIL1C_20180909T125249_N0206_R138_T27WWM_20180909T163645.zip'

In [3]: params = [
            {'name': 'set_output_folder', 'folder_name': 'ice_mask/', 'absolute_path': False},
            {'name': 'resample', 'bands':11, 'target_res': 10},

            {'name': 'add_package', 'package_name':'masks'},
            {'name': 'eo_function', 'bands': [3, 11, 8], 'group_bands': True,
                  'f_input': {'f_name' : 'simple_ice_mask', 'threshold': '0.6;float'}},
            {'name': 'quick_preview', 'bands': 'simple_ice_mask'},

            {'name': 'create_composite', 'bands': [4, 3, 2]},
            {'name': 'quick_preview', 'bands': 'RGB_432'},
        ]

In [4]: eo_object = eoDataReader(filename, params)
```

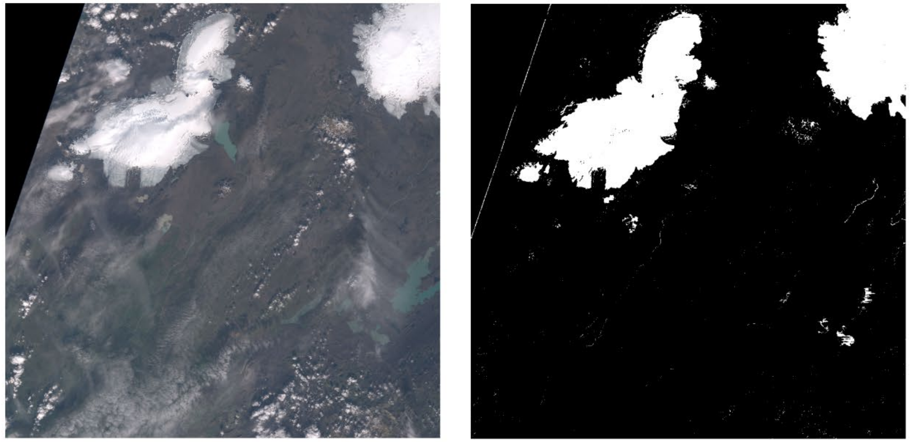
Processing chains with custom scripts



```python
import numpy as np


def simple_ice_mask(rasters, threshold=0.4):
    """
    """

    np.seterr(divide='ignore', invalid='ignore')

    A = rasters[0]
    B = rasters[1]

    numerator = (A - B)
    denominator = (A + B)
    denominator[denominator == 0] = 9999
    output = numerator / denominator

    output[output<=threshold] = 0
    output[output>threshold] = 1

    output[rasters[2]<np.max(rasters[2]*11/100)] = 0

    return output

```

masks/simple_ice_mask.py

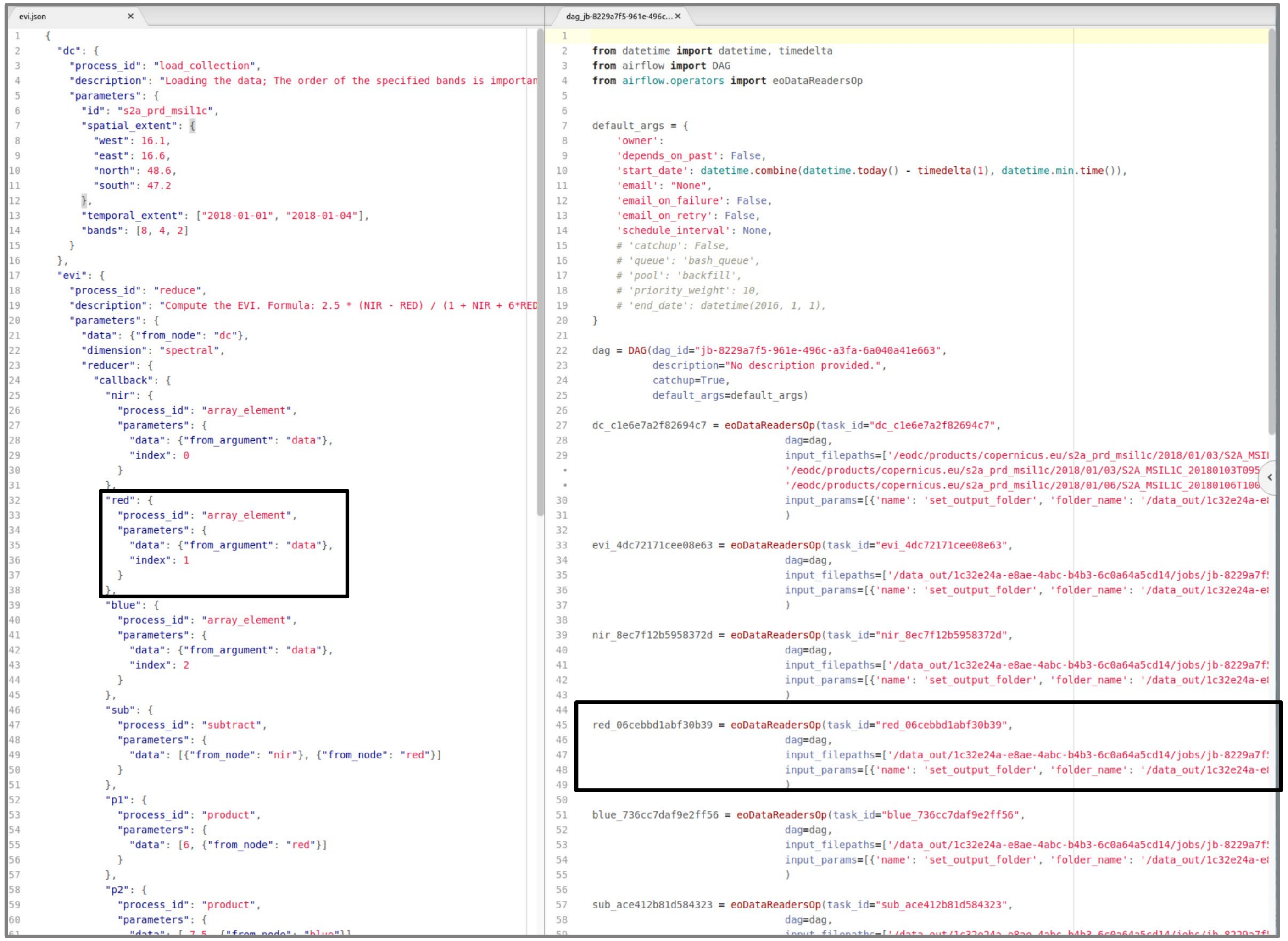
use your own code



Grant Agreement 776242

Implementation @ EODC, process graph parsing

OpenEO
process graph

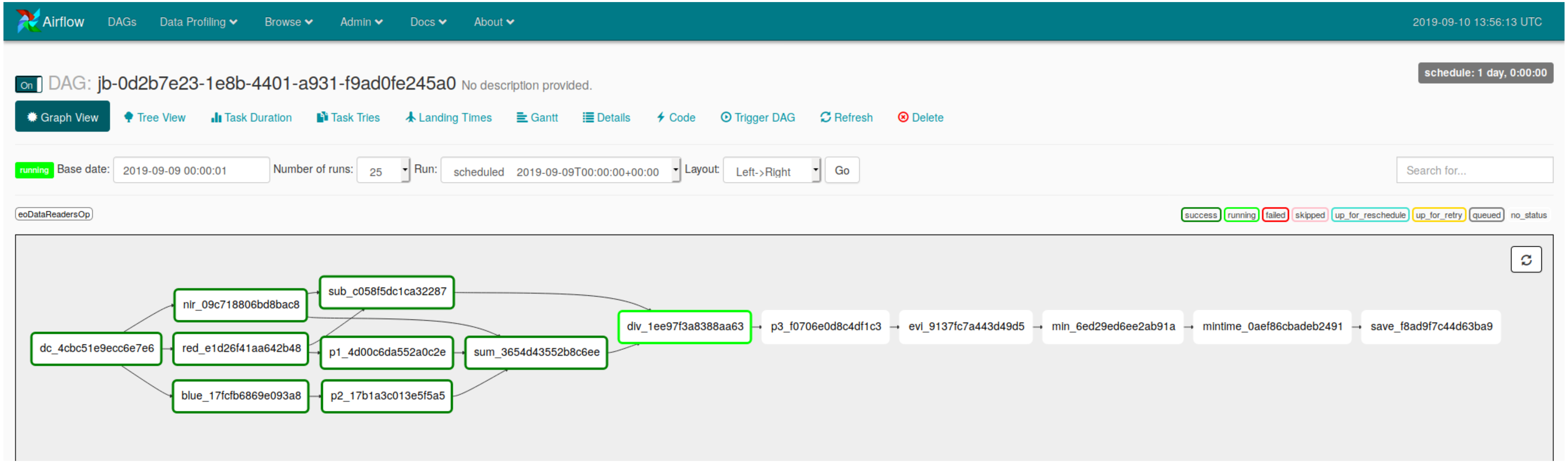


```json
{
  "dc": {
    "process_id": "load_collection",
    "description": "Loading the data; The order of the specified bands is importan
    "parameters": {
      "id": "s2a_prd_msil1c",
      "spatial_extent": {
        "west": 16.1,
        "east": 16.6,
        "north": 48.6,
        "south": 47.2
      },
      "temporal_extent": ["2018-01-01", "2018-01-04"],
      "bands": [8, 4, 2]
    }
  },
  "evi": {
    "process_id": "reduce",
    "description": "Compute the EVI. Formula: 2.5 * (NIR - RED) / (1 + NIR + 6*RED
    "parameters": {
      "data": {"from_node": "dc"},
      "dimension": "spectral",
      "reducer": {
        "callback": {
          "nir": {
            "process_id": "array_element",
            "parameters": {
              "data": {"from_argument": "data"},
              "index": 0
            }
          },
          "red": {
            "process_id": "array_element",
            "parameters": {
              "data": {"from_argument": "data"},
              "index": 1
            }
          },
          "blue": {
            "process_id": "array_element",
            "parameters": {
              "data": {"from_argument": "data"},
              "index": 2
            }
          },
          "sub": {
            "process_id": "subtract",
            "parameters": {
              "data": [{"from_node": "nir"}, {"from_node": "red"}]
            }
          },
          "p1": {
            "process_id": "product",
            "parameters": {
              "data": [6, {"from_node": "red"}]
            }
          },
          "p2": {
            "process_id": "product",
            "parameters": {
              "data": [ 7.5, {"from_node": "blue"}]
```

```python

from datetime import datetime, timedelta
from airflow import DAG
from airflow.operators import eoDataReadersOp


default_args = {
    'owner':
    'depends_on_past': False,
    'start_date': datetime.combine(datetime.today() - timedelta(1), datetime.min.time()),
    'email': "None",
    'email_on_failure': False,
    'email_on_retry': False,
    'schedule_interval': None,
    # 'catchup': False,
    # 'queue': 'bash_queue',
    # 'pool': 'backfill',
    # 'priority_weight': 10,
    # 'end_date': datetime(2016, 1, 1),
}

dag = DAG(dag_id="jb-8229a7f5-961e-496c-a3fa-6a040a41e663",
          description="No description provided.",
          catchup=True,
          default_args=default_args)

dc_c1e6e7a2f82694c7 = eoDataReadersOp(task_id="dc_c1e6e7a2f82694c7",
                                dag=dag,
                                input_filepaths=['/eodc/products/copernicus.eu/s2a_prd_msil1c/2018/01/03/S2A_MSI
                                '/eodc/products/copernicus.eu/s2a_prd_msil1c/2018/01/03/S2A_MSIL1C_20180103T095
                                '/eodc/products/copernicus.eu/s2a_prd_msil1c/2018/01/06/S2A_MSIL1C_20180106T100
                                input_params=[{'name': 'set_output_folder', 'folder_name': '/data_out/1c32e24a-e
                                )

evi_4dc72171cee08e63 = eoDataReadersOp(task_id="evi_4dc72171cee08e63",
                                dag=dag,
                                input_filepaths=['/data_out/1c32e24a-e8ae-4abc-b4b3-6c0a64a5cd14/jobs/jb-8229a7f
                                input_params=[{'name': 'set_output_folder', 'folder_name': '/data_out/1c32e24a-e
                                )

nir_8ec7f12b5958372d = eoDataReadersOp(task_id="nir_8ec7f12b5958372d",
                                dag=dag,
                                input_filepaths=['/data_out/1c32e24a-e8ae-4abc-b4b3-6c0a64a5cd14/jobs/jb-8229a7f
                                input_params=[{'name': 'set_output_folder', 'folder_name': '/data_out/1c32e24a-e
                                )

red_06cebbd1abf30b39 = eoDataReadersOp(task_id="red_06cebbd1abf30b39",
                                dag=dag,
                                input_filepaths=['/data_out/1c32e24a-e8ae-4abc-b4b3-6c0a64a5cd14/jobs/jb-8229a7f
                                input_params=[{'name': 'set_output_folder', 'folder_name': '/data_out/1c32e24a-e
                                )

blue_736cc7daf9e2ff56 = eoDataReadersOp(task_id="blue_736cc7daf9e2ff56",
                                dag=dag,
                                input_filepaths=['/data_out/1c32e24a-e8ae-4abc-b4b3-6c0a64a5cd14/jobs/jb-8229a7f
                                input_params=[{'name': 'set_output_folder', 'folder_name': '/data_out/1c32e24a-e
                                )

sub_ace412b81d584323 = eoDataReadersOp(task_id="sub_ace412b81d584323",
                                dag=dag,
                                input_filepaths=['/data_out/1c32e24a-e8ae-4abc-b4b3-6c0a64a5cd14/jobs/jb-8229a7f
```

Airflow DAG

Implementation @ EODC, Example



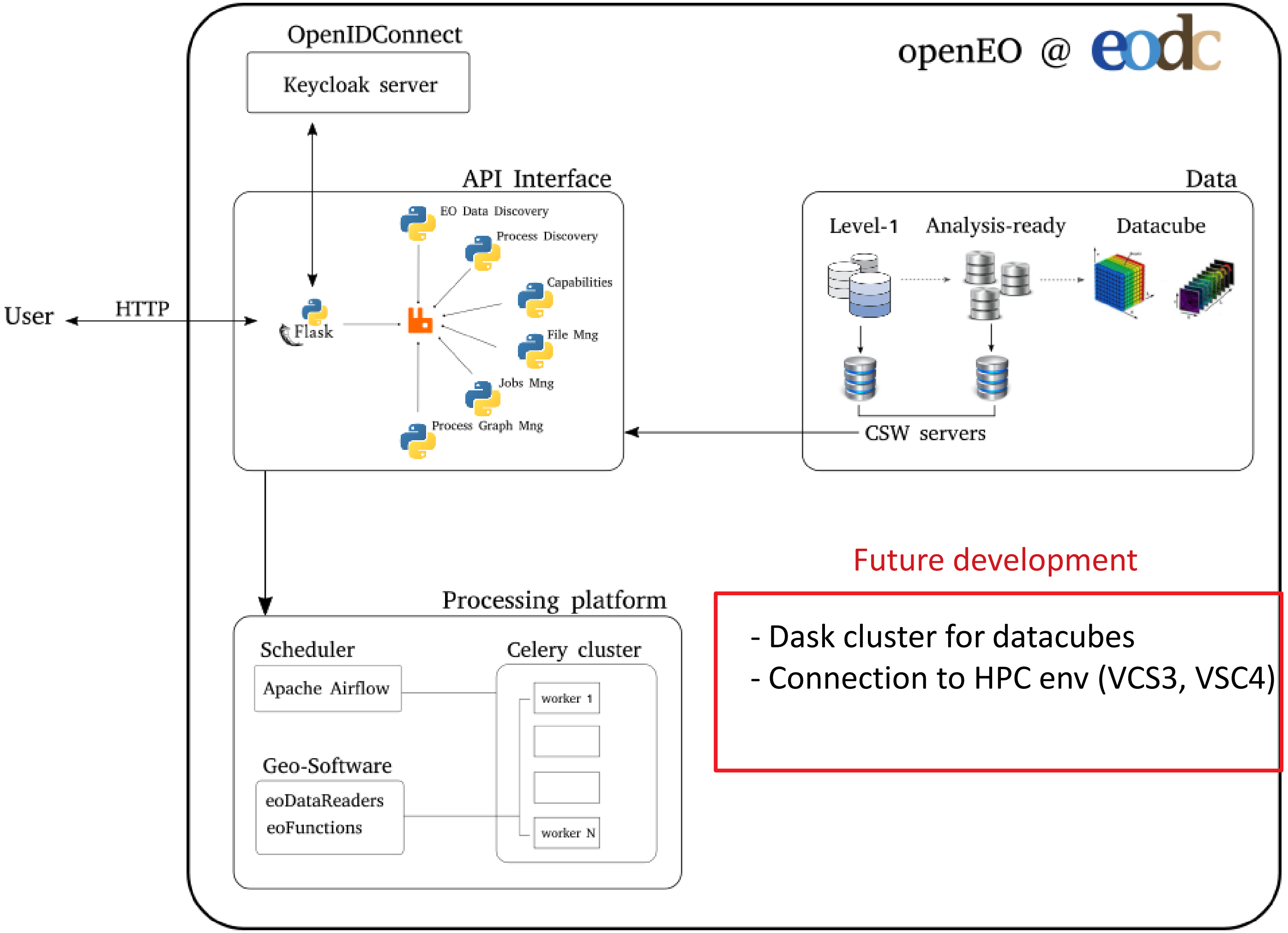
2019-09-10 13:56:13 UTC

DAG: jb-0d2b7e23-1e8b-4401-a931-f9ad0fe245a0 No description provided.

schedule: 1 day, 0:00:00

Base date: 2019-09-09 00:00:01 Number of runs: 25 Run: scheduled 2019-09-09T00:00:00+00:00 Layout: Left->Right

Implementation @ EODC, Future Development



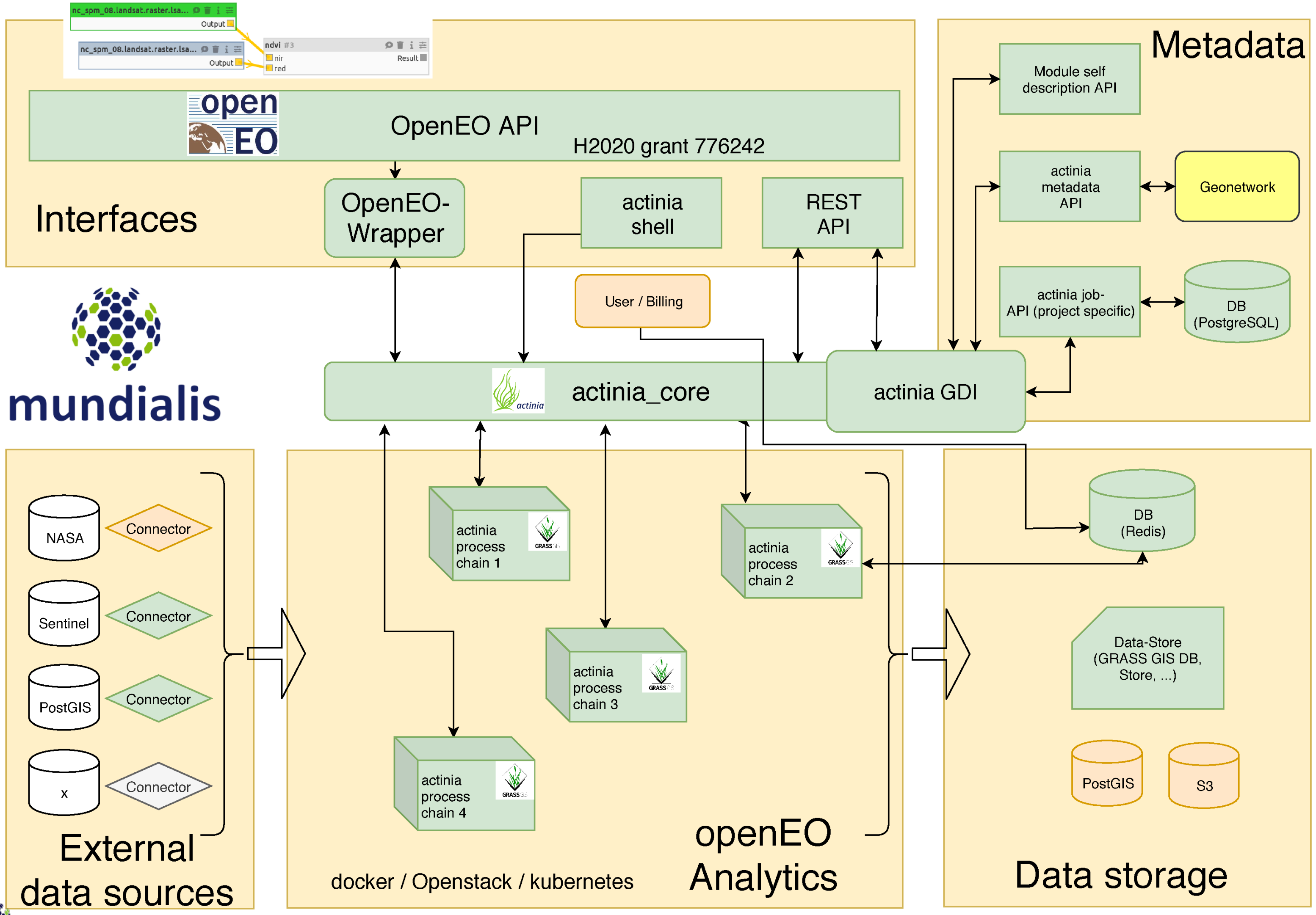
Future development

- Dask cluster for datacubes
- Connection to HPC env (VCS3, VSC4)

https://github.com/Open-EO/openeo-openshift-driver

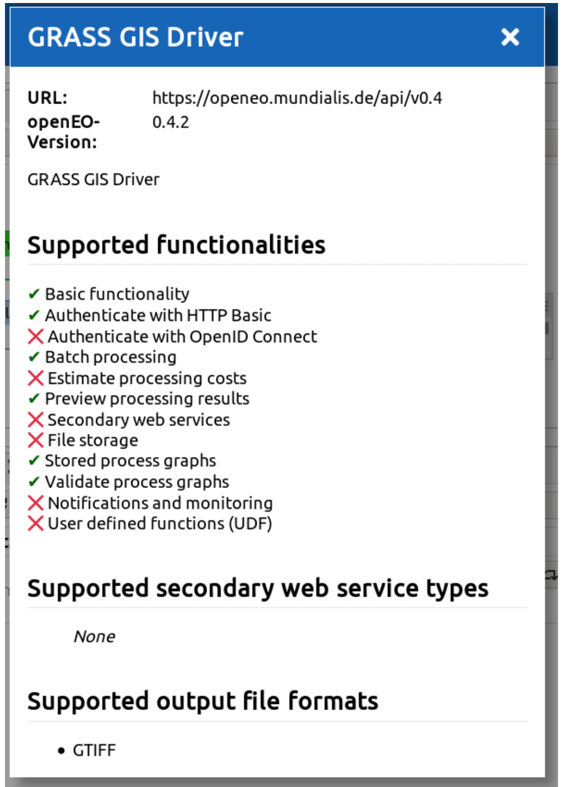
https://openeo.eodc.eu

GRASS GIS/Actinia backend (Mundialis)



openEO-grassgis-driver

https://github.com/Open-EO/openeo-grassgis-driver



GRASS GIS Driver ✕

URL: https://openeo.mundialis.de/api/v0.4
openEO-Version: 0.4.2

GRASS GIS Driver

Supported functionalities

- ✓ Basic functionality
- ✓ Authenticate with HTTP Basic
- ✗ Authenticate with OpenID Connect
- ✓ Batch processing
- ✗ Estimate processing costs
- ✓ Preview processing results
- ✗ Secondary web services
- ✗ File storage
- ✓ Stored process graphs
- ✓ Validate process graphs
- ✗ Notifications and monitoring
- ✗ User defined functions (UDF)

Supported secondary web service types

None

Supported output file formats

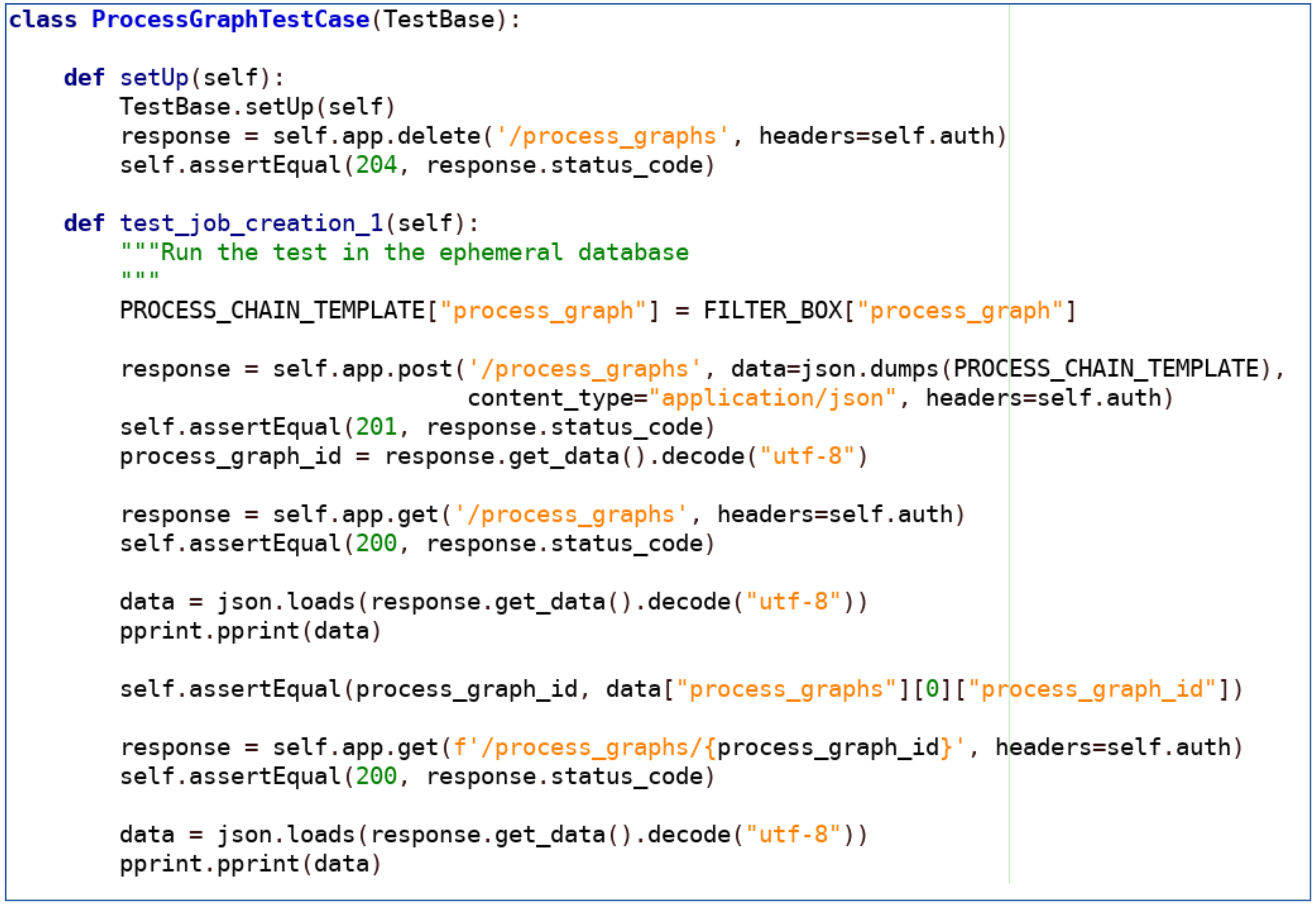
- GTIFF

GRASS GIS/Actinia backend (Mundialis)

Implementing unit tests

- no deployment of the system without unit tests passing
- feel free to use the existing unit tests to see how an openEO backend is to be programmed (“shortcut” - please also see the API definitions!)

Examples: https://github.com/Open-EO/openeo-grassgis-driver/tree/openeo-api-0.4.0/tests



```python
class ProcessGraphTestCase(TestBase):

    def setUp(self):
        TestBase.setUp(self)
        response = self.app.delete('/process_graphs', headers=self.auth)
        self.assertEqual(204, response.status_code)

    def test_job_creation_1(self):
        """Run the test in the ephemeral database
        """
        PROCESS_CHAIN_TEMPLATE["process_graph"] = FILTER_BOX["process_graph"]

        response = self.app.post('/process_graphs', data=json.dumps(PROCESS_CHAIN_TEMPLATE),
                                 content_type="application/json", headers=self.auth)
        self.assertEqual(201, response.status_code)
        process_graph_id = response.get_data().decode("utf-8")

        response = self.app.get('/process_graphs', headers=self.auth)
        self.assertEqual(200, response.status_code)

        data = json.loads(response.get_data().decode("utf-8"))
        pprint.pprint(data)

        self.assertEqual(process_graph_id, data["process_graphs"][0]["process_graph_id"])

        response = self.app.get(f'/process_graphs/{process_graph_id}', headers=self.auth)
        self.assertEqual(200, response.status_code)

        data = json.loads(response.get_data().decode("utf-8"))
        pprint.pprint(data)
```

Google Earth Engine backend (WWU)

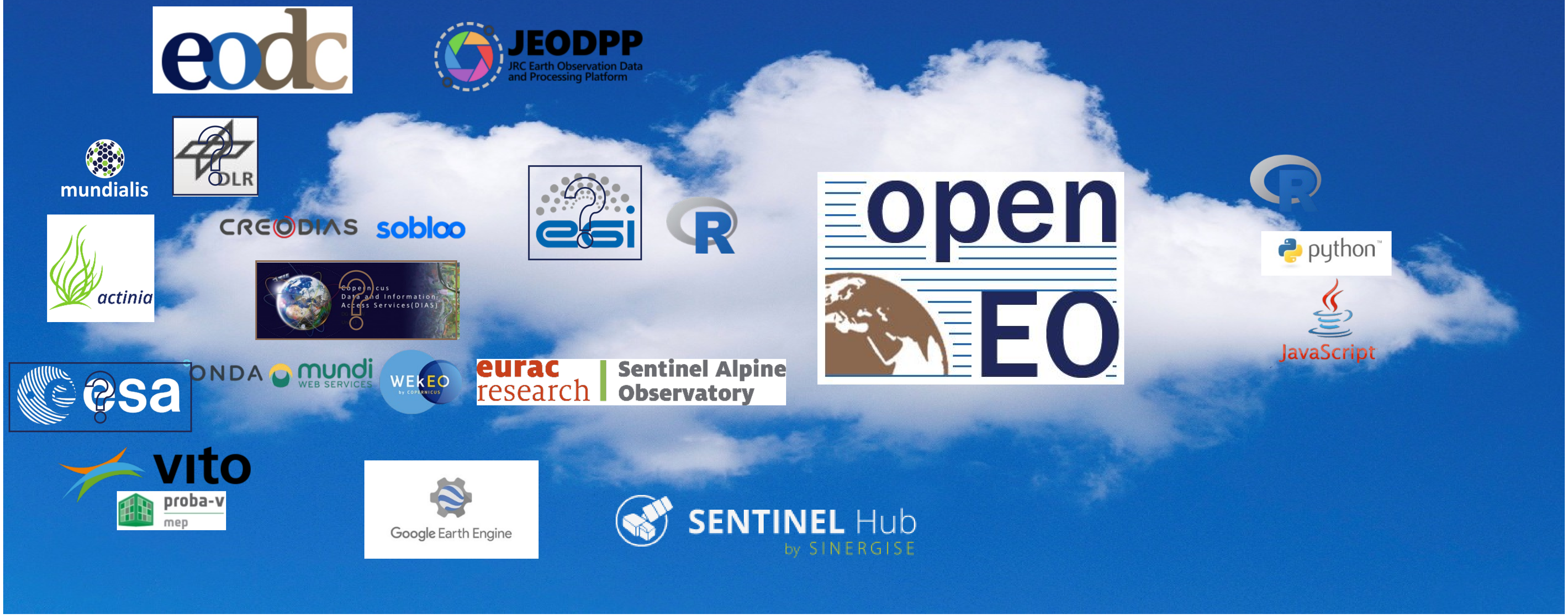
- Implemented and hosted by University of Münster
- Wrapper around the Earth Engine JavaScript API
- Translating process graphs into Earth Engine JavaScript
- Data Cube model managed in the background

- Easy implementation in less than a month
- Changes on Google‘s side: None, except providing STAC catalog
- Challenges:
  - Authentication / Data storage

So, who wants to be on board?

Back-end Implementations

Client Implementations

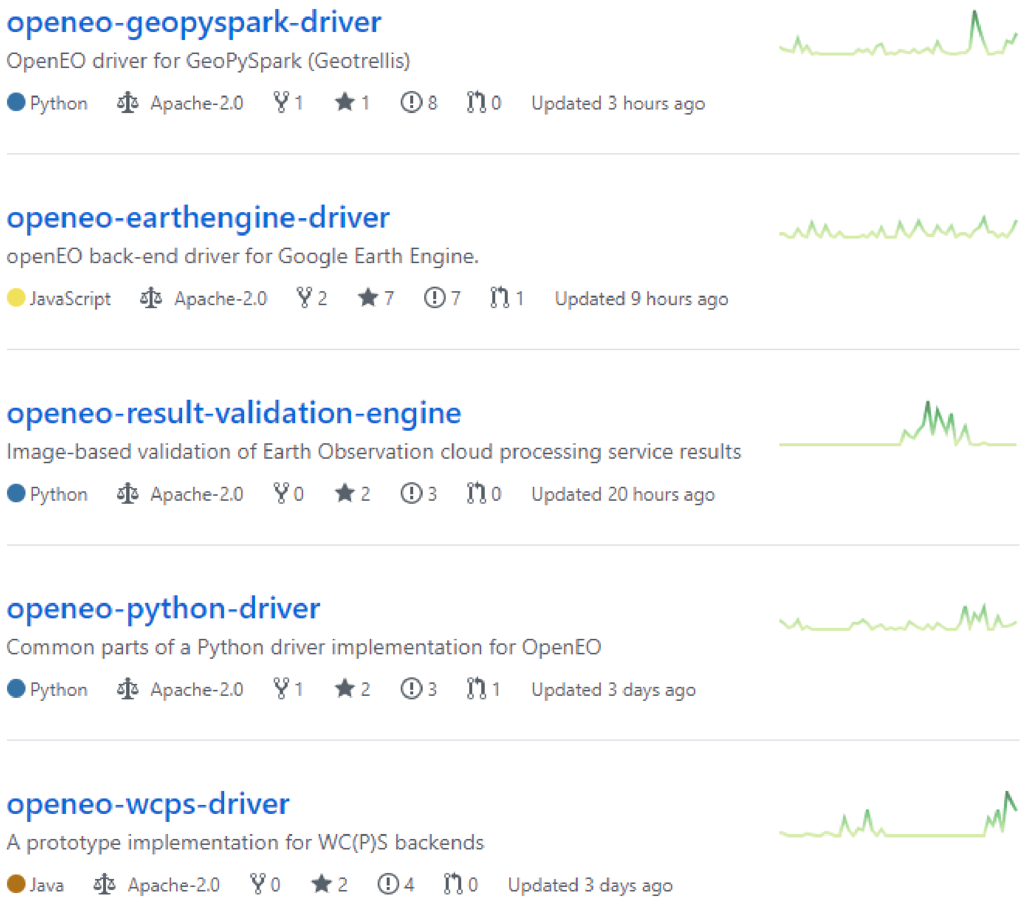


Getting started...

https://open-eo.github.io/openeo-api/gettingstarted-backends/

- First check for existing drivers @ https://github.com/Open-EO
- If an own implementation is needed:
  - You can still rely on some base functionality in the existing implementations
  - Or start from with openAPI code generator @ https://github.com/OpenAPITools/openapi-generator
  - Start with implementing the essential endpoints

RESTful implementation using OpenAPI Specification Version 3.0.1



openeo-geopyspark-driver
OpenEO driver for GeoPySpark (Geotrellis)
● Python ⚖ Apache-2.0 ⑂ 1 ★ 1 ⓘ 8 ⇅ 0 Updated 3 hours ago

openeo-earthengine-driver
openEO back-end driver for Google Earth Engine.
● JavaScript ⚖ Apache-2.0 ⑂ 2 ★ 7 ⓘ 7 ⇅ 1 Updated 9 hours ago

openeo-result-validation-engine
Image-based validation of Earth Observation cloud processing service results
● Python ⚖ Apache-2.0 ⑂ 0 ★ 2 ⓘ 3 ⇅ 0 Updated 20 hours ago

openeo-python-driver
Common parts of a Python driver implementation for OpenEO
● Python ⚖ Apache-2.0 ⑂ 1 ★ 2 ⓘ 3 ⇅ 1 Updated 3 days ago

openeo-wcps-driver
A prototype implementation for WC(P)S backends
● Java ⚖ Apache-2.0 ⑂ 0 ★ 2 ⓘ 4 ⇅ 0 Updated 3 days ago

The Endpoints of openEO

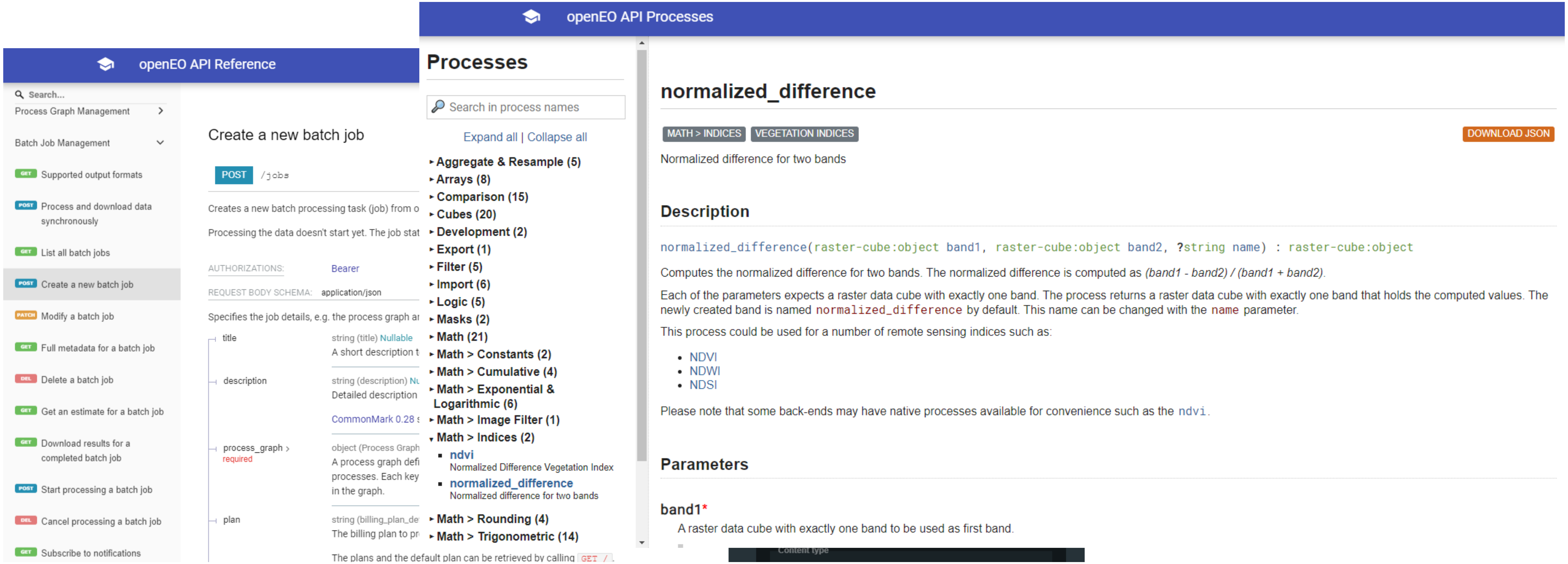
/ /.well-known/openeo /output_formats	root slash for capabilities and well-known for versioning
/collections /collections/{collectionid}	openEO strives for compatibility with STAC and OGC API as far as possible for data discovery.
/processes	The basis for all computation are processes
/process_graphs /process_graphs/{graphID}	Processes can be chained into process graphs
/results	Results can be processed and downloaded synchronously
/jobs /jobs/{jobid} /jobs/{jobid}/results	Process graphs can be submitted as batch-jobs, queued for processing and results can be download upon completion

Handling of user authentication and billing.	/credentials/basic /credentials/oidc
User can upload own files	/files/{userid}
Consume results as secondary web services (e.g. WMS, XYZ, WCS)	/service_types/{serviceid}
User can create and integrate user defined functions into process graphs	/udf_runtimes

Useful resources http://docs.openeo.org

https://open-eo.github.io/openeo-api/apireference/

http://processes.openeo.org



openEO API Reference

Search...

Process Graph Management

Batch Job Management

GET Supported output formats

POST Process and download data synchronously

GET List all batch jobs

POST Create a new batch job

PATCH Modify a batch job

GET Full metadata for a batch job

DEL Delete a batch job

GET Get an estimate for a batch job

GET Download results for a completed batch job

POST Start processing a batch job

DEL Cancel processing a batch job

GET Subscribe to notifications

Create a new batch job

POST /jobs

Creates a new batch processing task (job) from o

Processing the data doesn't start yet. The job stat

AUTHORIZATIONS: Bearer

REQUEST BODY SCHEMA: application/json

Specifies the job details, e.g. the process graph ar

title — string (title) Nullable
A short description t

description — string (description) Nu
Detailed description
CommonMark 0.28 s

process_graph › required — object (Process Graph
A process graph defi
processes. Each key
in the graph.

plan — string (billing_plan_de
The billing plan to pr

The plans and the default plan can be retrieved by calling GET /.

openEO API Processes

Processes

Search in process names

Expand all | Collapse all

- Aggregate & Resample (5)
- Arrays (8)
- Comparison (15)
- Cubes (20)
- Development (2)
- Export (1)
- Filter (5)
- Import (6)
- Logic (5)
- Masks (2)
- Math (21)
- Math > Constants (2)
- Math > Cumulative (4)
- Math > Exponential & Logarithmic (6)
- Math > Image Filter (1)
- Math > Indices (2)
  - ndvi
    Normalized Difference Vegetation Index
  - normalized_difference
    Normalized difference for two bands
- Math > Rounding (4)
- Math > Trigonometric (14)

normalized_difference

MATH > INDICES VEGETATION INDICES DOWNLOAD JSON

Normalized difference for two bands

Description

```
normalized_difference(raster-cube:object band1, raster-cube:object band2, ?string name) : raster-cube:object
```

Computes the normalized difference for two bands. The normalized difference is computed as *(band1 - band2) / (band1 + band2)*.

Each of the parameters expects a raster data cube with exactly one band. The process returns a raster data cube with exactly one band that holds the computed values. The newly created band is named `normalized_difference` by default. This name can be changed with the `name` parameter.

This process could be used for a number of remote sensing indices such as:

- NDVI
- NDWI
- NDSI

Please note that some back-ends may have native processes available for convenience such as the `ndvi`.

Parameters

band1*

A raster data cube with exactly one band to be used as first band.

Conclusions

- A number of reference backend implementations are currently in development

- Based on different programming languages
  - Python
  - Java
  - R
  - Java-script

- Based on different existing hard and software infrastructure

- Can be used as starting point for own backend implementation
  - Together with extensive documentation
  - And guides
  - All of this is open source

Thank you for your attention!

Westfälische Wilhelms-Universität Münster
mundialis
DE
WAGENINGEN UNIVERSITY & RESEARCH
NL
European Commission
BE
vito
AT
TU WIEN TECHNISCHE UNIVERSITÄT WIEN
eodc
SI
SINERGISE
SOLENIX
CH
eurac research
IT

- http://openeo.org/
- openEO@list.tuwien.ac.at
- https://github.com/Open-EO
- @open_EO
- https://www.youtube.com/channel/UCMJQil8j9sHBQkcSISaEsvQ
- https://www.researchgate.net/project/openEO
- https://openeo-chat.eodc.eu/channel/public
- zenodo https://zenodo.org/communities/openeo/