Joint Research Centre (JRC)

Advances in interactive processing and visualisation with JupyterLab on the JRC Big Data Platform (JEODPP)

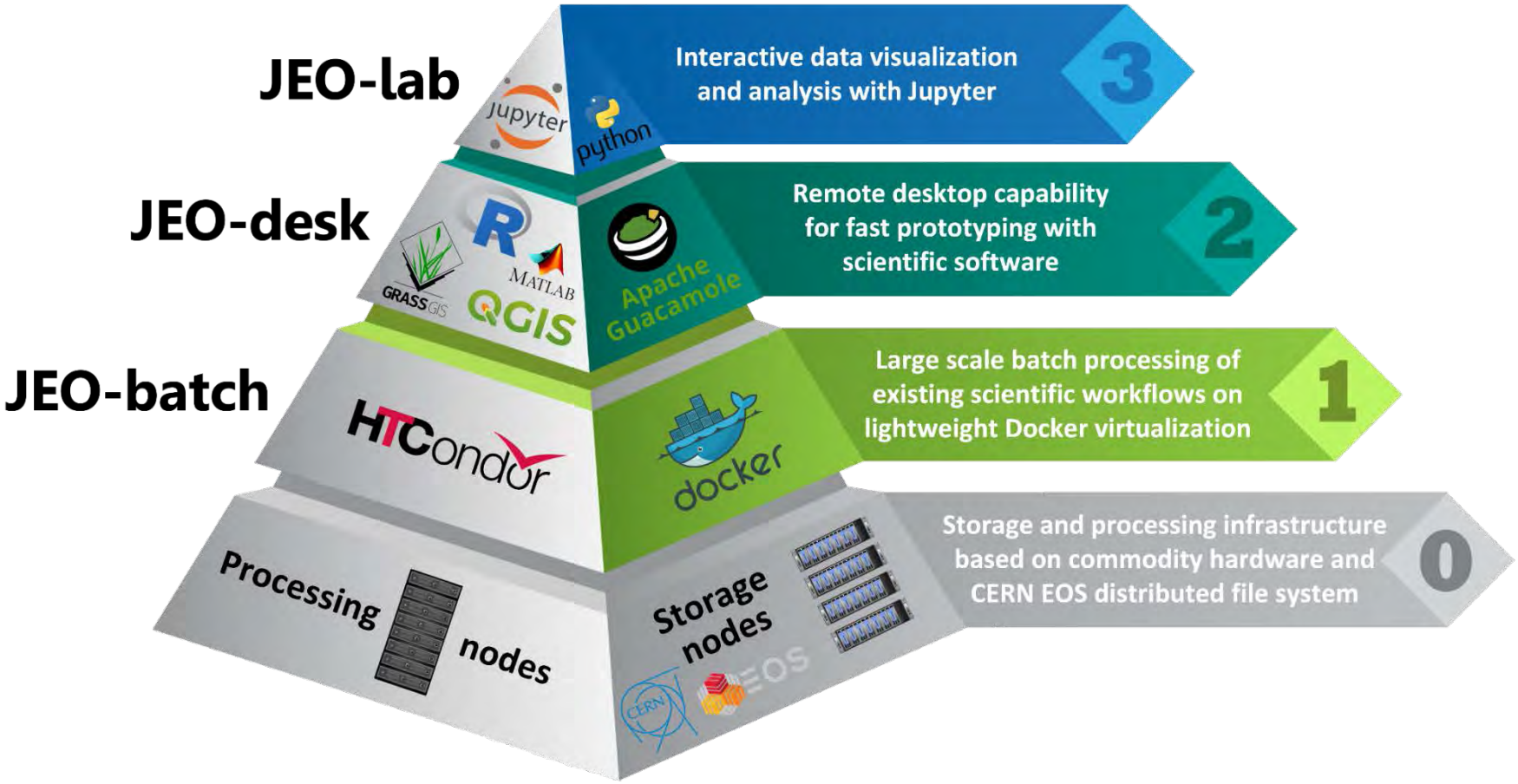
Davide DE MARCHI, Pierre SOILLE

European Commission, Joint Research Centre
Directorate I Competences, Unit I.3 Text and Data Mining
Big Data Analytics Project

Contacts: **davide.de-marchi@ec.europa.eu**
pierre.soille@ec.europa.eu

Big Data From Space 2019
19/02/2019, Munich

JEODPP conceptual representation



Current status of JEODPP platform

Based on:

- **commodity** hardware
- **open-source** software stack

Storage:

- ***CERN EOS*** *distributed file system*
- *Currently 9 PB* ***net*** *capacity*

Processing servers:

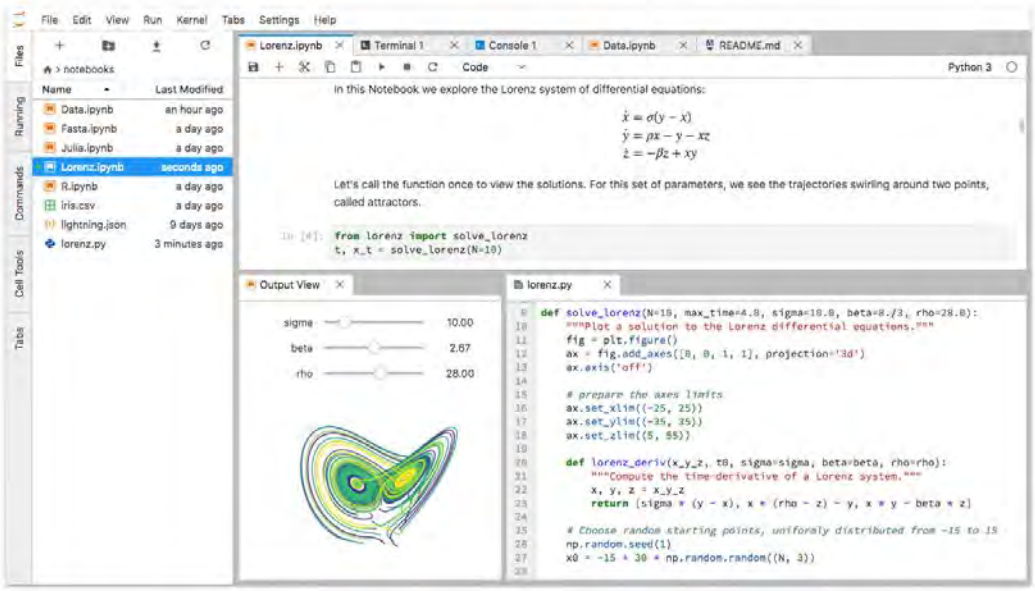
- *1,500 cores over 35 nodes*
- *4 servers equipped with multi-GPUs and dedicated to Machine Learning processing with TensorFlow, Keras, …*

Joint Research Centre

European Commission

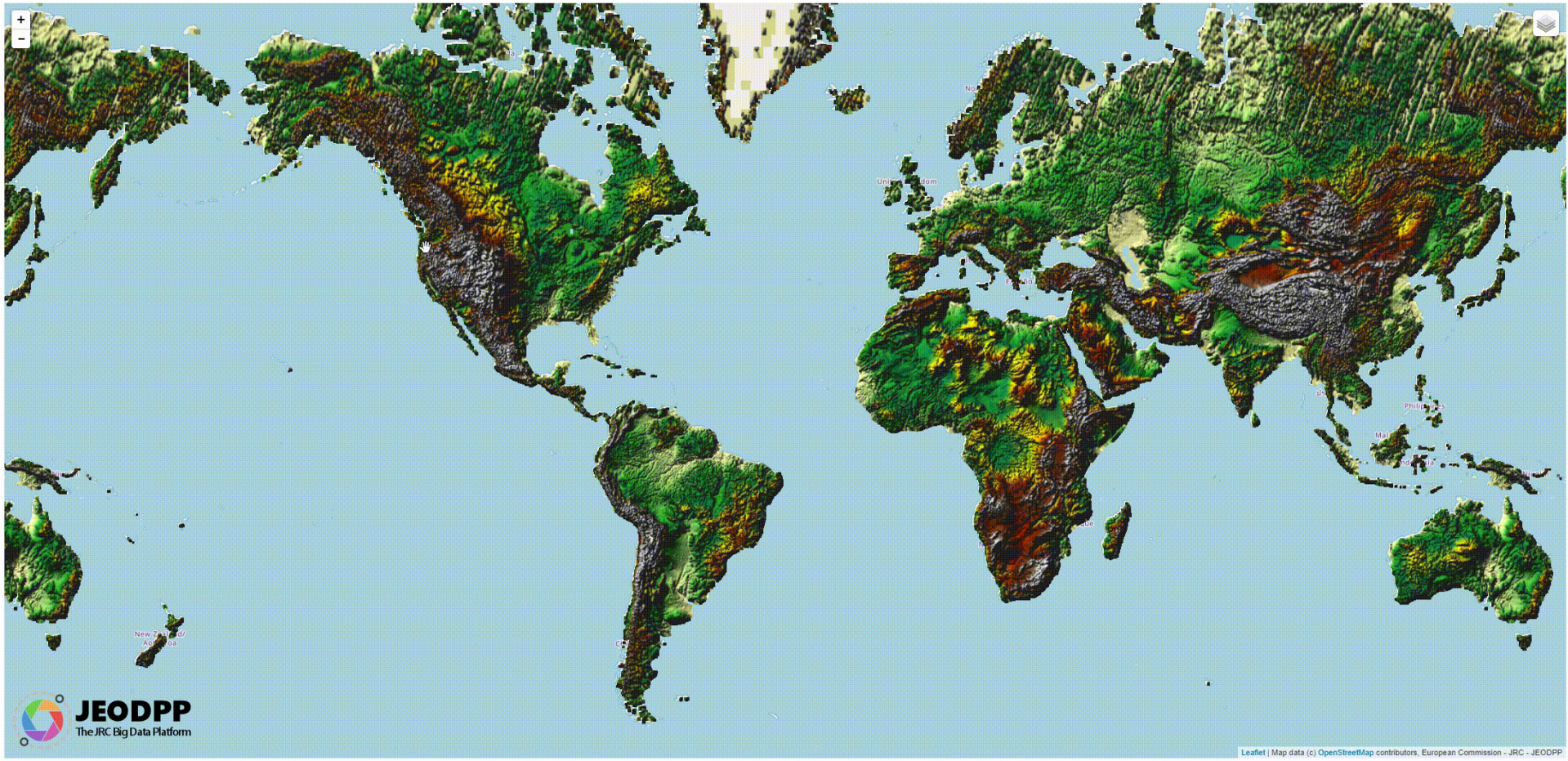
Interactive visualization and analysis with JupyterLab

- Web interface to visualize and analyze big geospatial data
- Allows fast search and display of complex dataset
- Available for geospatial expert with some programming capabilities
- Allows easy creation of GUI applications for non programmers (ipywidgets, ipyleaflet, bqplot, qgrid, …)



New datasets available: ALOS AW3D30

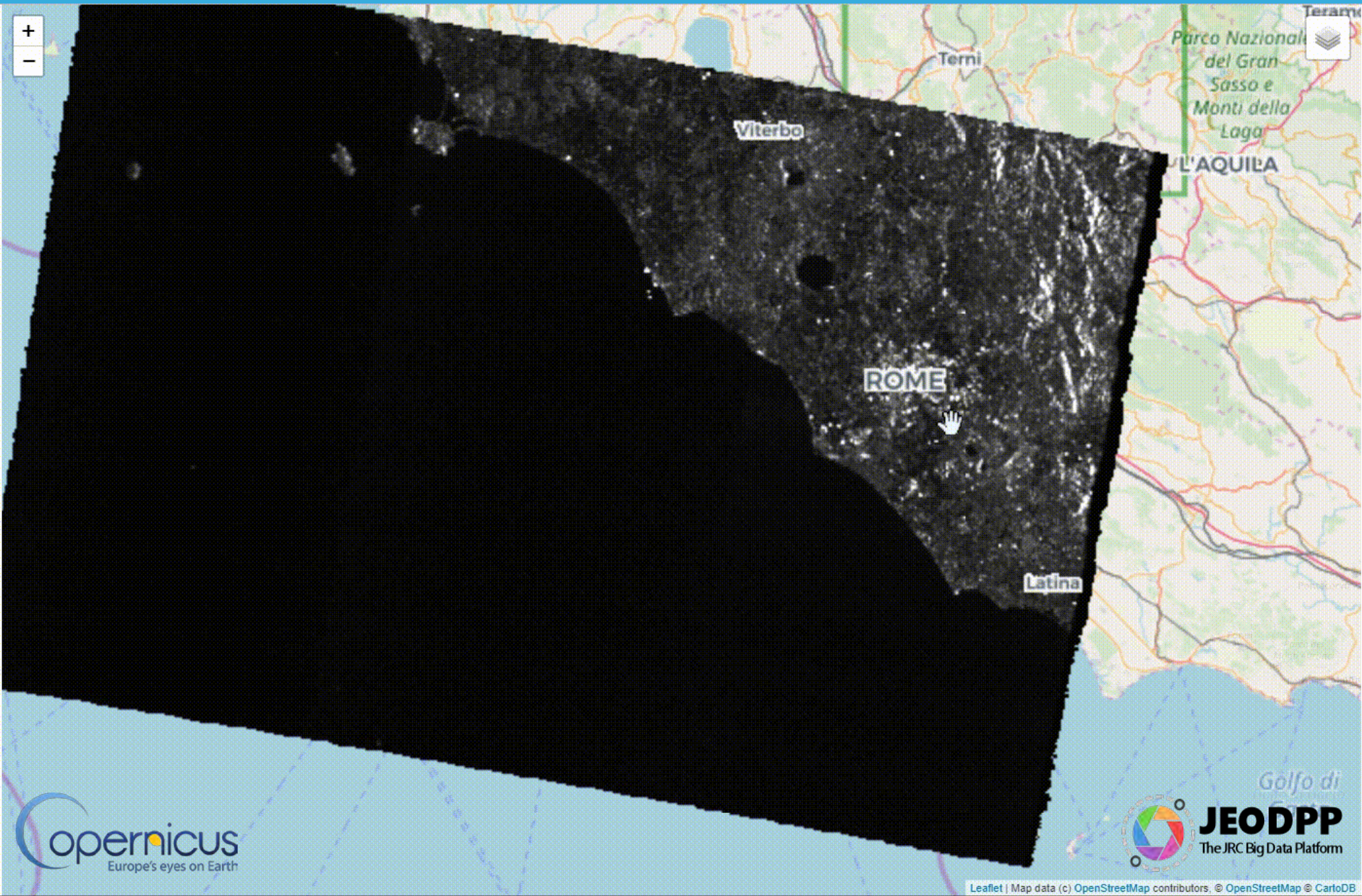
http://www.eorc.jaxa.jp/ALOS/en/aw3d30/index.htm



Joint Research Centre

European Commission

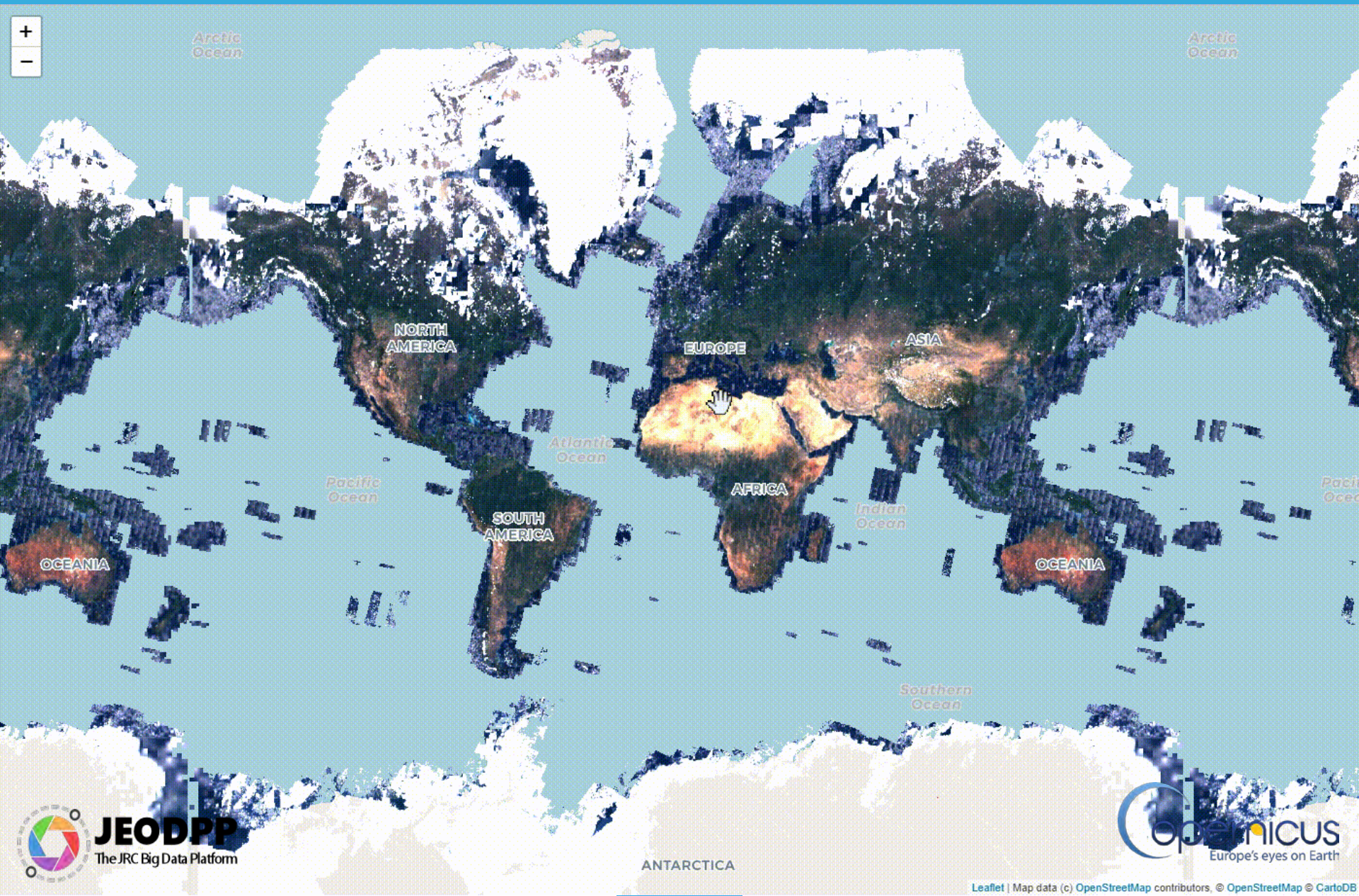
New datasets available: Sentinel-1



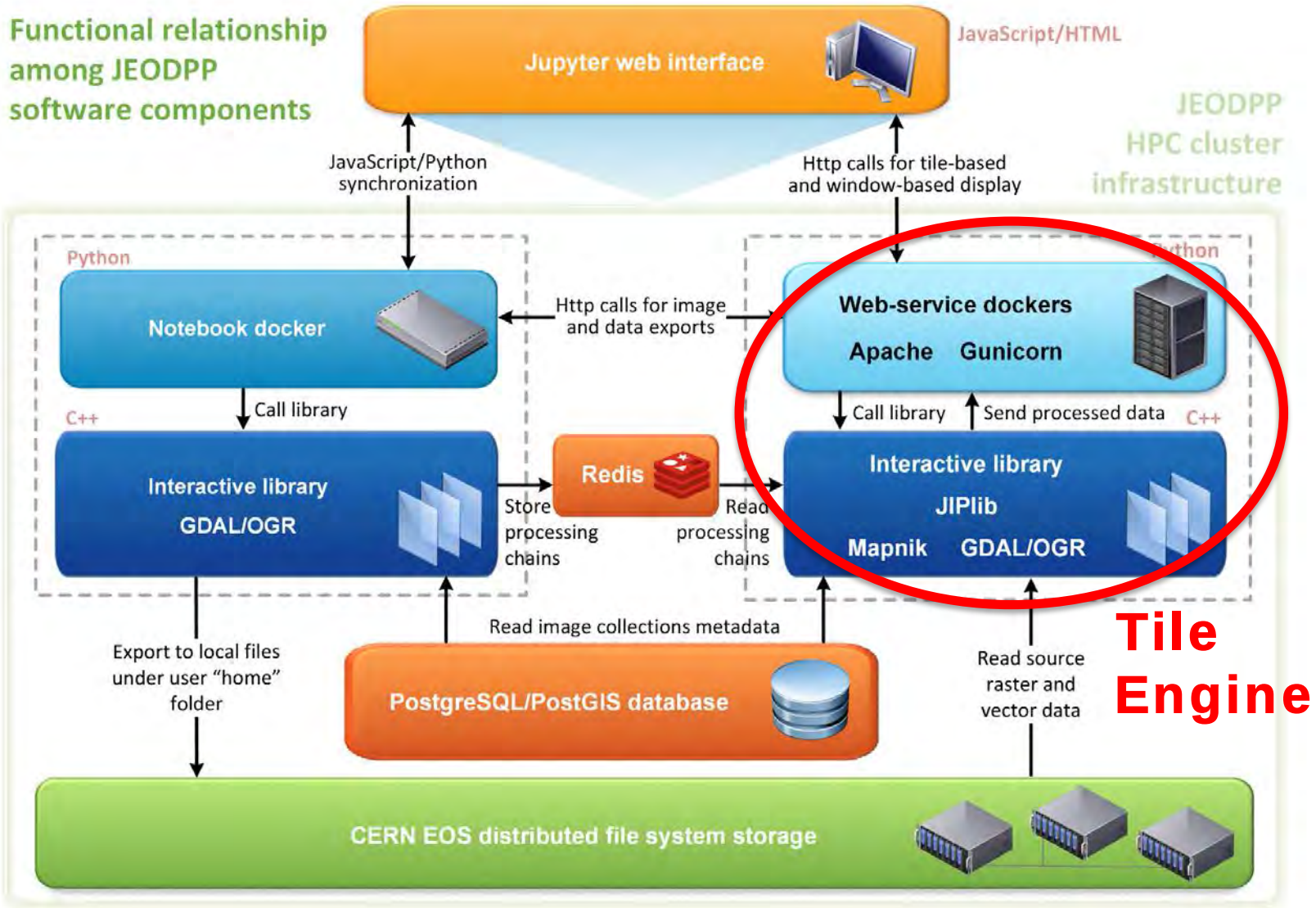
opernicus
Europe's eyes on Earth

JEODPP
The JRC Big Data Platform

New datasets available: Global mosaics



JEO-lab software components



Python code injected server-side

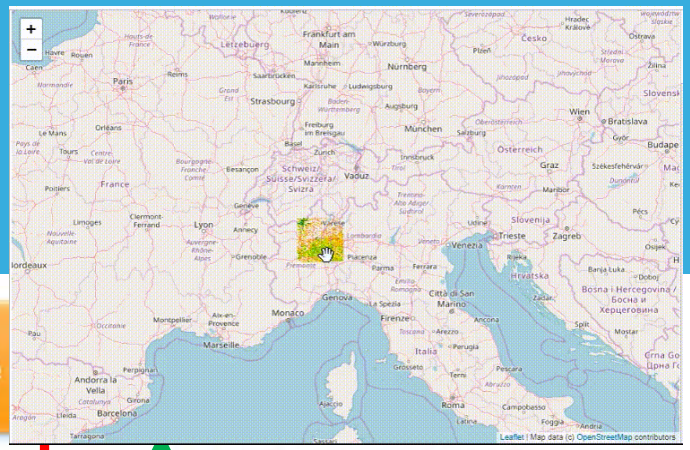
Need to increase user flexibility and use available python libraries

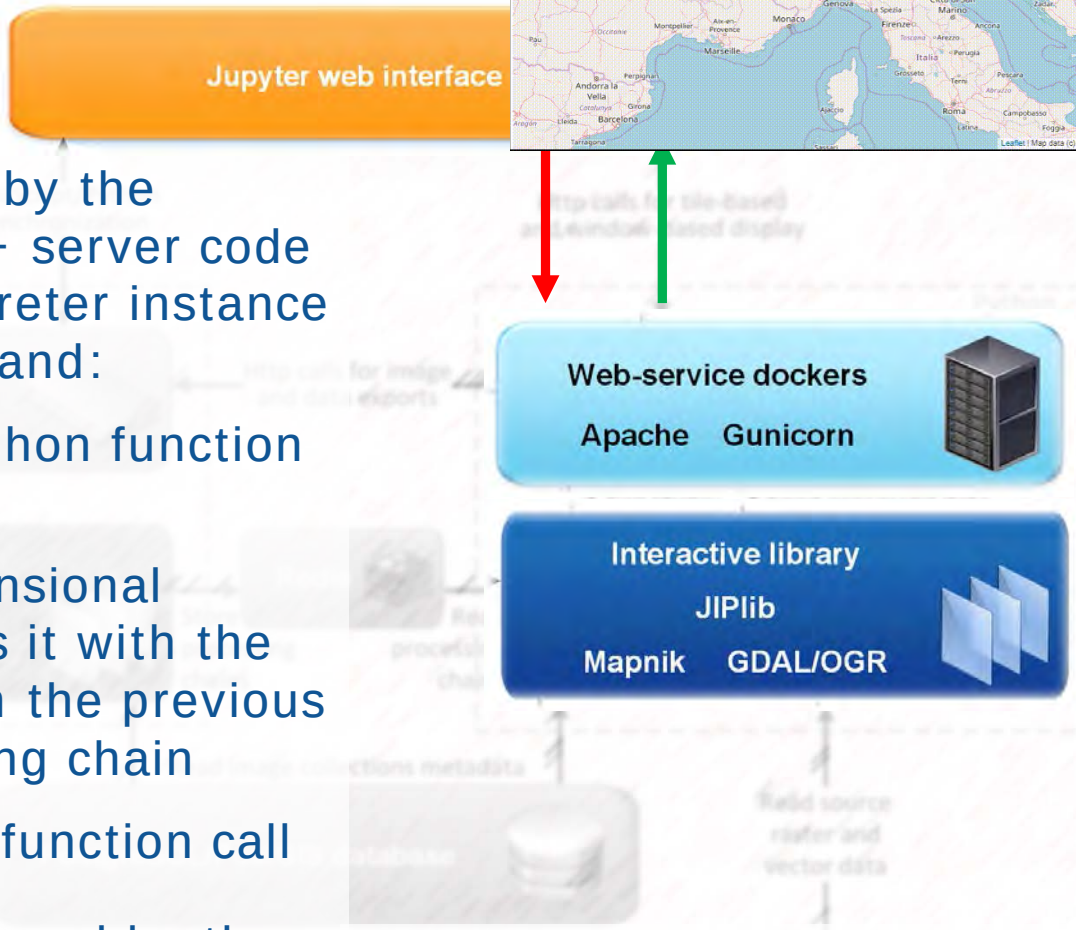
Solution: enable injection of custom python code to the server-side Tile Engine running in the HPC

Function definition and **function call** are converted to strings using python **inspect** module (getsource and getcallargs functions)

For security reasons the list of available libraries is limited but customizable on-demand (numpy, scipy-ndimage, OpenCV, etc.)

Python code injected server-side





For each tile requested by the ipyleaflet map, the C++ server code creates a python interpreter instance (**python embedding**) and:

- executes in it the python function definition
- creates a multi-dimensional Numpy array and fills it with the results obtained from the previous steps of the precessing chain
- executes the python function call
- Reads the result returned by the user function and passes it to the next step of the chain

```
def maskpy(img, n):
    return img[img<=n] = 0
```

An example: stubble burning mapping

Courtesy: JRC Directorate D Sustainable Resources, **D.5** Food Security

- Deliberate setting fire of the straw stubble that remains after wheat and other grains have been harvested.
- The practice was widespread until the 1990s, when governments increasingly restricted its use
- Many risks:
  - *Pollution from smoke*
  - *Risk of fires spreading out of control*
  - *…*

Detection of stubble burning from satellite images using python code injected server-side

Stubble burning detection for Sentinel2:

B04 < 1000 AND
B06 < 1200 AND
B08 < 1200 AND
B11 > 500 AND
B11 < 1600

Function definiton that implements the stubble burning algorithm:

```
def stubble(img, v4, v6, v8, v11min, v11max):
    b4,b6,b8,b11 = img[0],img[1],img[2],img[3]
    res = numpy.ones_like(b4)
    res[b4>=v4] = 0
    res[b6>=v6] = 0
    res[b8>=v8] = 0
    res[numpy.logical_or(b11<=v11min,
                         b11>=v11max)] = 0
    return res
```

Multi-band processing chain containing the python function call:

```
p = coll.processMulti(["B04","B06","B08","B11"])
        .execute(stubble,1000,1200,1200,500,1600)
        .band(0).scale(0,1).colorCustom(["Lime"])
```

Joint Research Centre

European Commission

Detection of stubble burning from satellite images using python code injected server-side

Launcher | ExecuteStubbleBurning.ipyn

Code | Python 2

Define a server-side python function to calculate the stubble index

In [26]:
```
def stubble(v4=1000, v6=1200, v8=1200, v11_1=500, v11_2=1600):
    global img
    img = numpy.ones_like(band0)
    img[band0>v4] = 0
    img[band1>v6] = 0
    img[band2>v8] = 0
    img[numpy.logical_or(band3<v11_1, band3>v11_2)] = 0
```

Test the python function for correctness in a simulated client-side environment

In [27]:
```
testExecute(stubble,1000,1200,1200,500,1600)
```

Out[27]: True

Create the processing chain

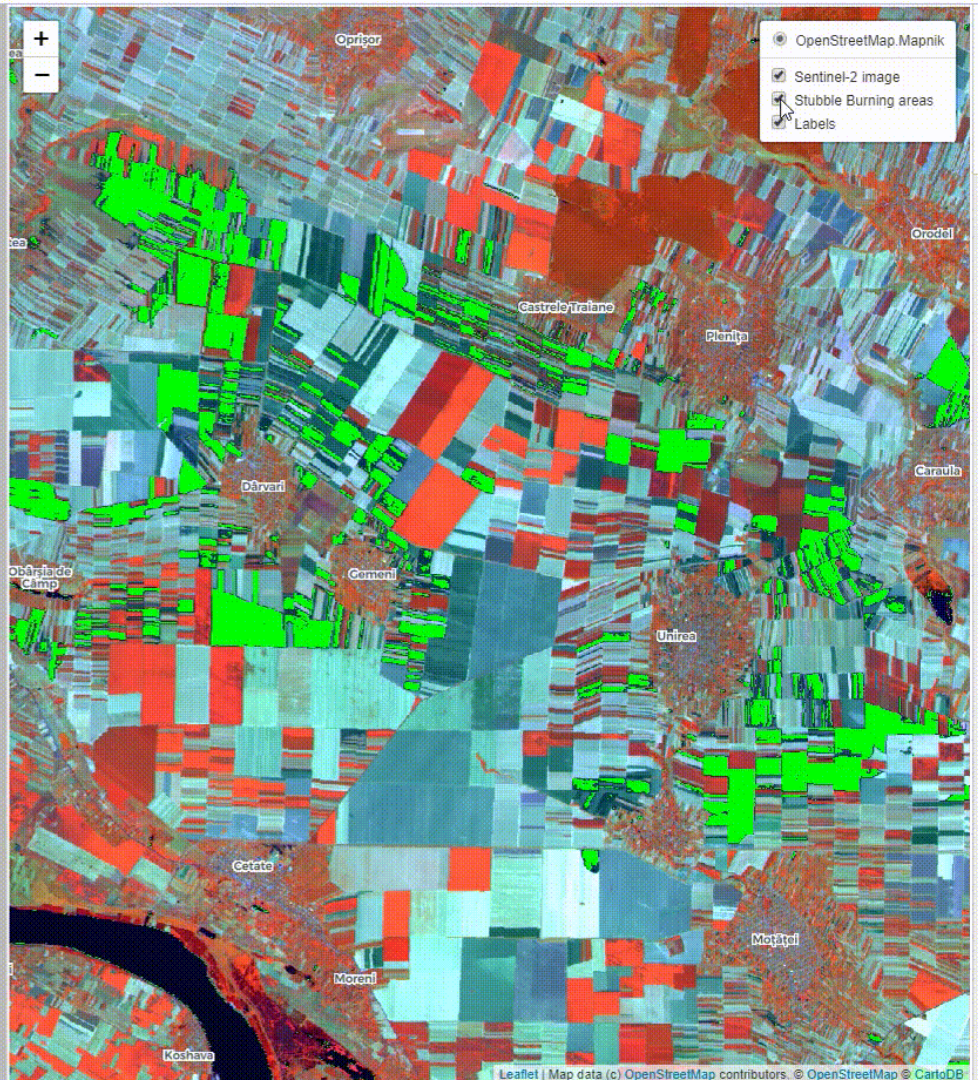
In [28]:
```
bands = ["B04","B06","B08","B11"]
pStubble = coll.processMulti(bands).execute(stubble,1000,1200,1200,500,1600)\
                                   .band(0).scale(0,1).colorCustom(["Lime"])
```

Add layers to the map

In [29]:
```
map.clear()

pImage = coll.process().bands(S2_LandWater)\
                       .scale(1006, 4318, 1027, 4242, 283, 2109)
map.addLayer(pImage, name='Sentinel-2 image')

map.addLayer(pStubble, name='Stubble Burning areas')

pLabels = Collection(collections.Basemaps.Labels.Light).process()
map.addLayer(pLabels.toLayer(),name='Labels')

map.layersControl(True)
```

In [ ]:

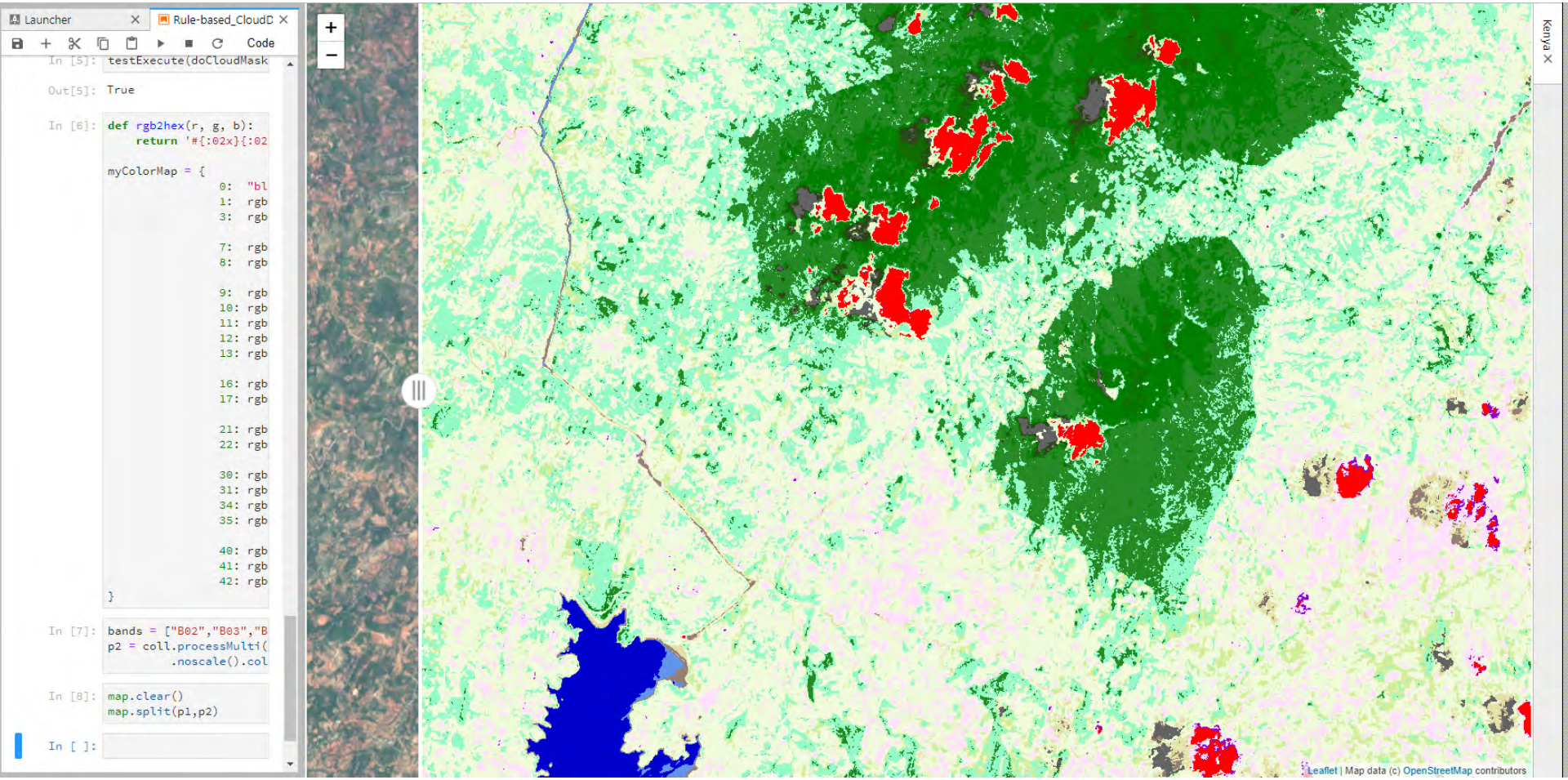
JEODPP
The JRC Big Data Platform



Rule based cloud detector implemented in numpy

Credits:
Dario Simonetti,
JRC, doi:10.2760/790249

http://forobs.jrc.ec.europa.eu/recaredd/S2_composite.php



Joint Research Centre

European Commission

Multi-Temporal Maximum-NDI composition

Launcher | MaxNDVI_v3.ipynb

Code | Python 2

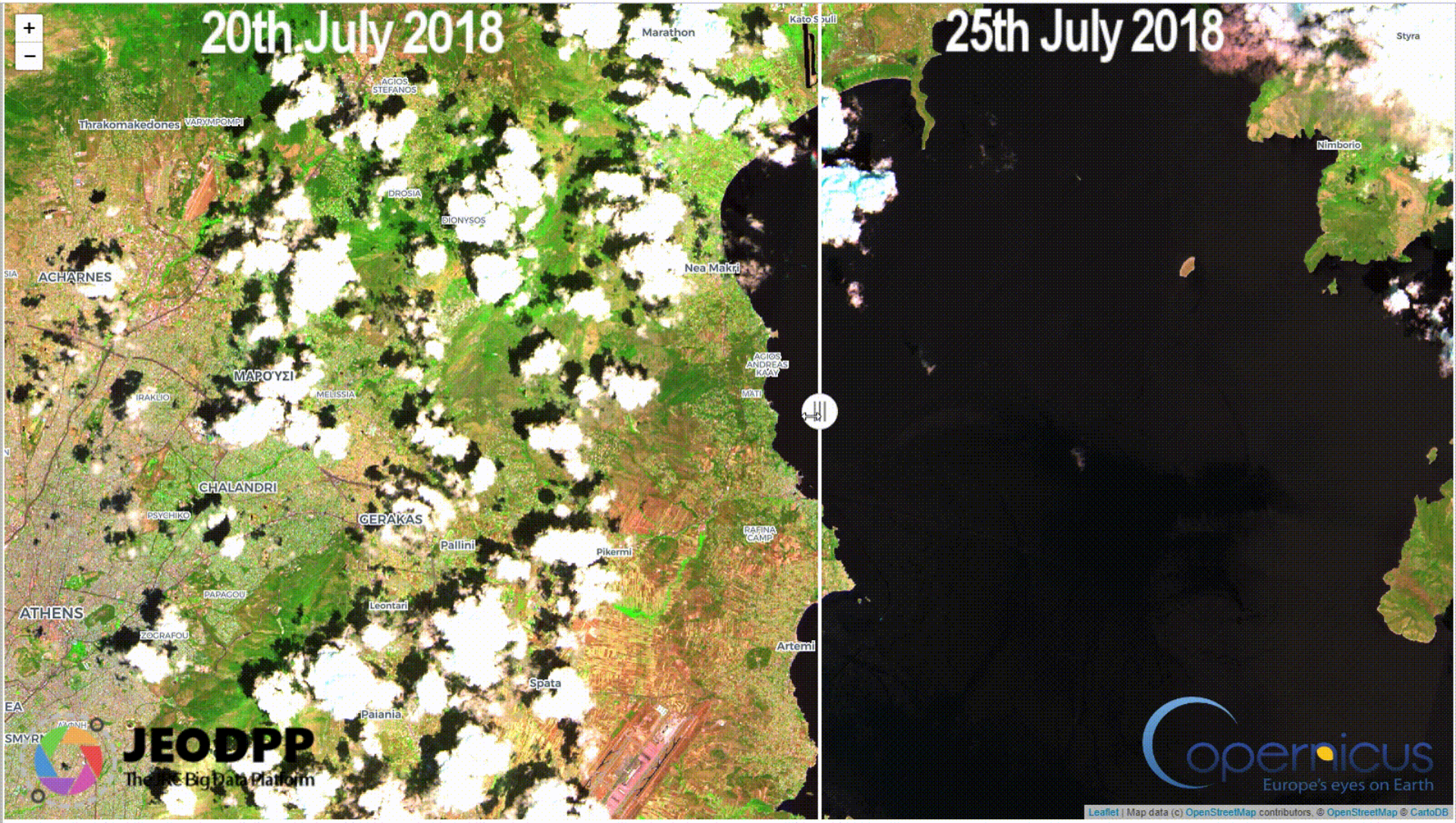
MaxNDI RGB composition using multi-band multi-temporal execute function and simple numpy programming (argmax and choose functions)

```
In [1]: map = Map(side="Maximum NDVI composition")
        z = map.zoomToExtent(inter.search("Munich"))

In [2]: def maxNDI(inputimage, iband11, iband8, iband4, ibandR=0, iban

            B11 = extractBand(inputimage, iband11)
            B8  = extractBand(inputimage, iband8)
            B4  = extractBand(inputimage, iband4)

            r = extractBand(inputimage, ibandR)
            g = extractBand(inputimage, ibandG)
            b = extractBand(inputimage, ibandB)

            def calculate_ndi(bx, by):
                den = by + bx
                ndi = (by - bx)/den
                ndi[by==0] = -1.0
                ndi[bx==0] = -1.0
                return ndi

            NDVI = calculate_ndi(B4,B8)
            NDWI = calculate_ndi(B11,B8)

            index_NDVI = np.argmax(NDVI,axis=0)
            index_NDWI = np.argmax(NDWI,axis=0)

            for (x,y), value in numpy.ndenumerate(index_NDVI):
                if  2 * NDVI[index_NDVI[x][y]][x][y] <= NDWI[index_NDW
                    index_NDVI[x][y] = index_NDWI[x][y]

            outr = index_NDVI.choose(r)
            outg = index_NDVI.choose(g)
            outb = index_NDVI.choose(b)

            return np.array([outr,outg,outb])

In [3]: testExecuteTimes(2, 8, maxNDI, 0, 1, 2, 2, 3, 4)

Out[3]: True
```

Joint Research Centre

European Commission

Easy comparison with the split map control



Georeferenced temporal videos

```
coll = coll.filterOn("jrc_filepath", "<>", "")
coll = coll.filterOnPoint(8.6214,45.8198)
coll = coll.limit(10000).sort("cloudcover",True)

def getprocessing(collection):
    return collection.process().bands("B04","B03","B02",1.1).opacity(255)

#exportTemporalVideo(coll,map,getprocessing,"Baveno.mp4",1,2)
```

Add the video as an overlay to the map

By putting the generated video on a cloud service and getting the direct link, it can e displayed on the map

```
In [9]: map = Map(side="Baveno Video", basemap=basemaps.OpenStreetMap.DE)
        z = map.zoomToExtent((45.9153150000001, 8.495186000000001, 10.0))
```

```
In [10]: video = map.addVideo(
             'https://dl.dropboxusercontent.com/s/lyhoz2uvt3dt5zv/Baveno.mp4?dl=0',
             [(45.80223851020229, 8.23802947998047),
              (45.99338392574006, 8.750610351562502)],
             'Video layer')
         video.interact(opacity=(0.0,1.0,0.01))
```

opacity 1.00

```
In [ ]: map.layersControl()
```

Exporting a processing chain to a plain HTML page

To be viewed also outside of the JEODPP platform

```
In [ ]: coll = Collection(collections.EarthObservation.Copernicus.Sentinel2.Level1C)
        coll = coll.filterOnPoint(8.6214,45.8198)
        coll = coll.filterOnDay(2017,10,14)
        coll = coll.filterOn("cloudcover", "<=", 50)
        coll = coll.filterOn("jrc_filepath", "<>", "")

        p = coll.process().bands(S2_FalseColorInfrared)
```

```
In [ ]: p.exportHtml("Baveno.html")
```

MODIS temporal slider

Example on how temporal sliders can be used to display MODIS images for specific dates:

Joint Research Centre

European Commission

Widgets enabled applications: s2explorer

Launcher | S2explorerVideo.ipynb

Code | Python 2

Interactively explore the Sentinel-2 data catalog

```
In [8]: map = Map(basemap=35, side='s2explorer')
In [9]: s2explorer(map)
```

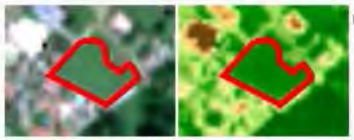
Search | Display | Stretch | Zoom | Exports | Measure | Draw | Overlay | Split | Extract

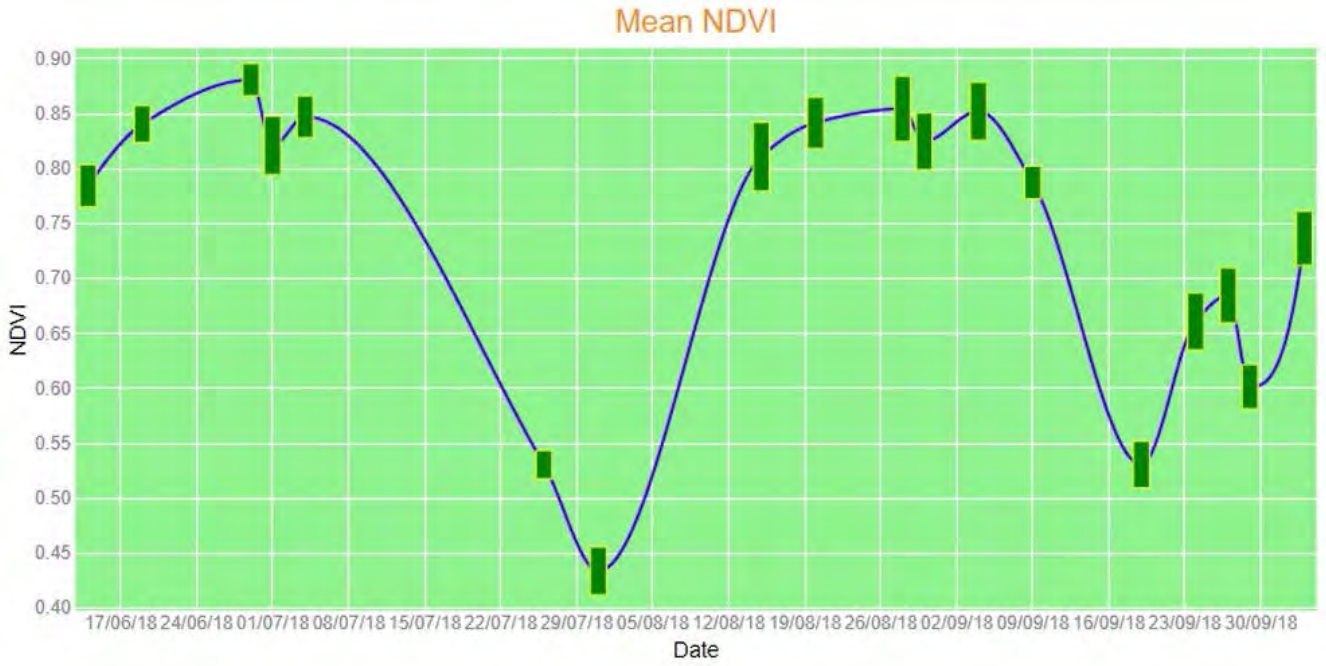
Deactivate measure control | Color while measuring: Orch | Color after measuring: Medi

	Date	Orbit	Type	Sat	Full	SizeMB	Cloud	UTM	MGRS
10	2018-12-12	51	L2A	B	✔	1001	8	31N	31UET
12	2018-11-17	51	L2A	A	✔	968	0	31N	31UET
29	2018-09-13	51	L2A	B	✔	963	7	31N	31UET
44	2018-06-30	51	L2A	A	✔	1018	1	31N	31UET
54	2018-05-11	51	L2A	A	✔	953	1	31N	31UET
56	2018-05-06	51	L2A	B	✔	921	1	31N	31UET
61	2018-04-06	51	L2A	B	✔	950	6	31N	31UET

Joint Research Centre

European Commission

16	2018-09-07	108	L2A	B	17		0.84064	0.01677	113	✔
17	2018-09-04	65	L2A	B	3		0.78443	0.01926	113	✔



In [ ]:

JEODPP
The JRC Big Data Platform

Extraction of NDVI temporal profile

Takeaway message

- Versatile Big Data platform serving wide variety of projects
- Importance of Copernicus temporal resolution for many different applications (agriculture, forest, disasters, etc.)
- Suitable for experienced scientists and also for final users
- With the server-side injection of python code, the interactive visualization is even more flexible and open and allows fast prototyping of batch mode processing

Thank you for your attention!

ELSEVIER

Future Generation Computer Systems

Volume 81, April 2018, Pages 30-40

A versatile data-intensive computing platform for information retrieval from big geospatial data

P. Soille, A. Burger, D. De Marchi, P. Kempeneers, D. Rodriguez, V. Syrris, V. Vasilev

Show more

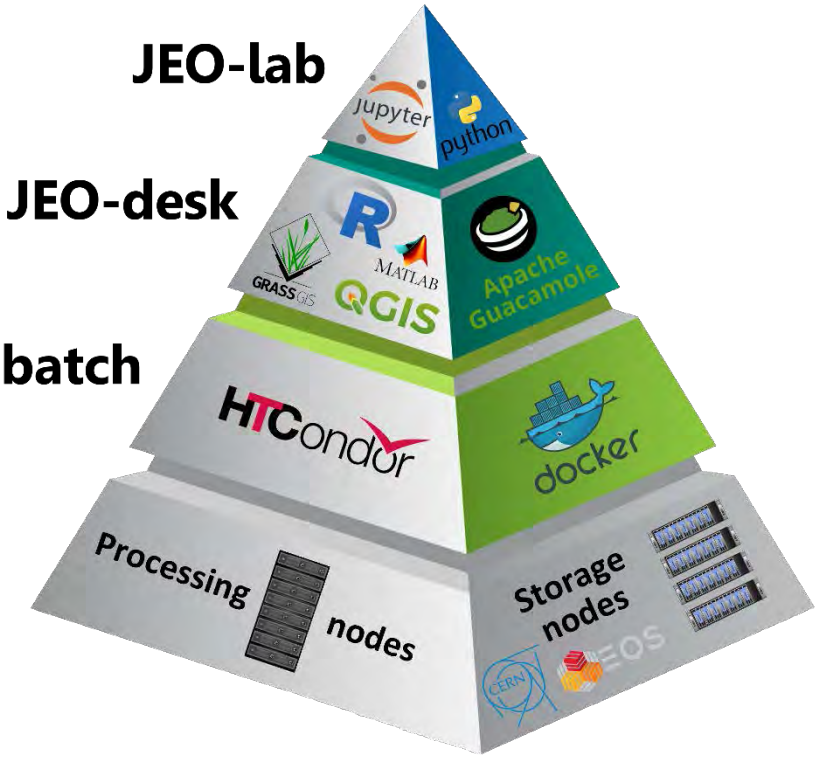
https://doi.org/10.1016/j.future.2017.11.007

Open Access funded by Joint Research Centre

https://doi.org/10.1016/j.future.2017.11.007

Publication list:
https://cidportal.jrc.ec.europa.eu/home/publications

JEODPP
The JRC Big Data Platform



Big Data Analytics project

Unit I.3 Text and Data Mining Unit
Directorate I Competences

Joint Research Centre

European Commission