Terradue

Advancing Earth Science

Using SNAP in Cloud processing services for GEP

Φ-week event

10 September 2019

ESA-ESRIN

Terradue

- Terradue is an European Space Agency spin-off started in 2006
  - Based in Rome, staff of 16 from 6 nationalities
- Providing support to application builders in Earth sciences
  - To use satellite EO data as information source
  - Cloud PaaS, complemented with APIs for Cloud bursting
- Business model: Platform-centered, a collaborative workplace “**Ellip**” for value adders to interact & co-create

ellip
Collaborative Earth Science Services
Discover the first collaborative platform in Earth Sciences.

Thematic Exploitation Platforms

- R&D activities to create an ecosystem of interconnected Thematic Exploitation Platforms
- Users access a work environment containing the data and resources required, as opposed to downloading and replicating the data “at home”

The fundamental principle is to move the User to the data and tools

esa

coastal tep

forestry tep

geohazards tep

hydrology tep

polar tep

urban tep

food security tep

Geohazards Exploitation Platform | GEP

GEP designed in the context of:

- Geohazards Supersite initiative (GSNL)
- CEOS Disasters Working Group

User-driven model for partnership and community building

Started from Int. Forum on Satellite EO and Geohazards organised by ESA and GEO in Santorini in 2012 (140+ participants)

GEO GROUP ON EARTH OBSERVATIONS

esa

THE INTERNATIONAL FORUM ON SATELLITE EO AND GEOHAZARDS

The Santorini Conference
Santorini, Greece, 21–23 May 2012

Ph. BALLY (Editor)
ESA/ESRIN

Geohazards Exploitation Platform | GEP

Platform based on virtualization & federation of EO data

- Provide services & support to the geohazards community

On-demand & systematic processing services

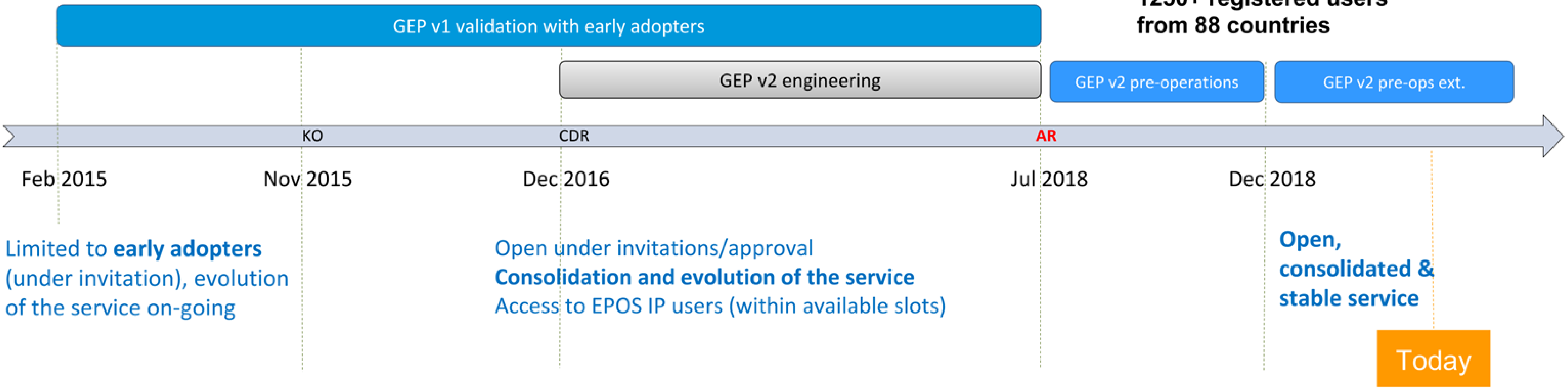
- Cloud Compute power, managing multi-tenant resources

Access to Copernicus Sentinels repositories

- Plus access to hundred TBs of EO data archives (ERS and ENVISAT), and other EO missions (ALOS-2, Cosmo-Skymed and TerraSAR-X) under CEOS WG Disaster and the GSNL agreements

Geohazards Exploitation Platform | GEP

As of March 2019:
1250+ registered users
from 88 countries



GEP v1 validation with early adopters

GEP v2 engineering

GEP v2 pre-operations

GEP v2 pre-ops ext.

KO

CDR

AR

Feb 2015

Nov 2015

Dec 2016

Jul 2018

Dec 2018

Limited to **early adopters** (under invitation), evolution of the service on-going

Open under invitations/approval
Consolidation and evolution of the service
Access to EPOS IP users (within available slots)

Open, consolidated & stable service

Today

22 platform early adopters
integrating application or exploiting on demand processing

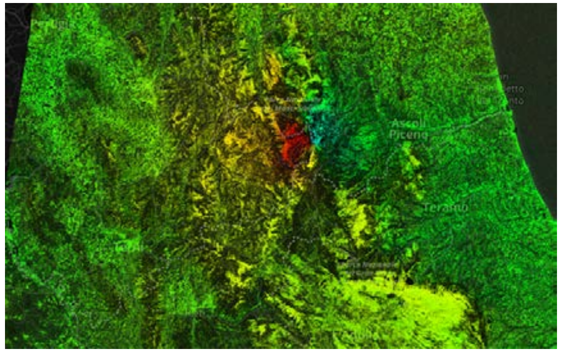
50 platform early adopters
28 new users in the period

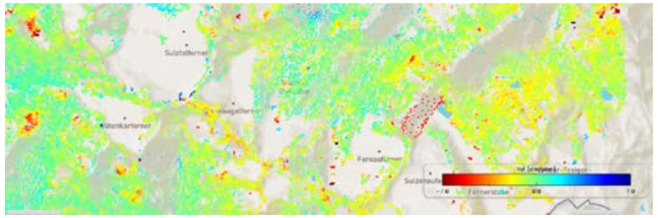
170+ platform early adopters
100+ new users in the period

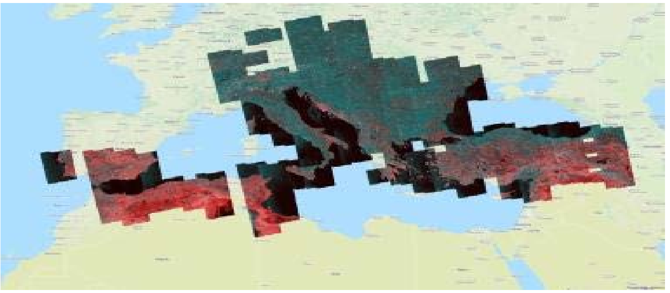
Want to apply as early adopter of the GEP Early Adopters Programme ?
contact@geohazards-tep.eu

GEP | Enriched Services Portfolio

- 25+ on-demand services using Optical & SAR data grouped in Thematic Apps, according to the defined goals of Community Managers
- New basic services providing full resolution and change detection imagery for rapid online visualization
- 8 systematic services delivering continuously updated information layers on GEP, including the large scale production of Sentinel-1 InSAR browse images at both 100m and 50m resolution over tectonic regions and volcanoes





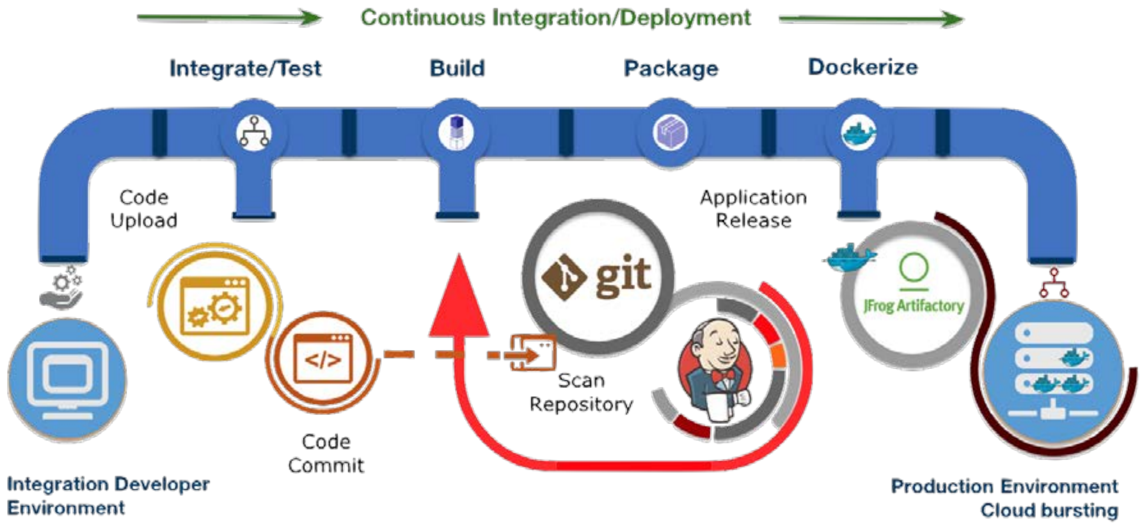


GEP | Application Integration

- Applications developed in any programming language supported
  - C/C++, Java, Python, Matlab and IDL

- Continuous Integration and Deployment Environment with automatic packaging and deployment in production environments

ELLIP
TERRADUE



sobloo

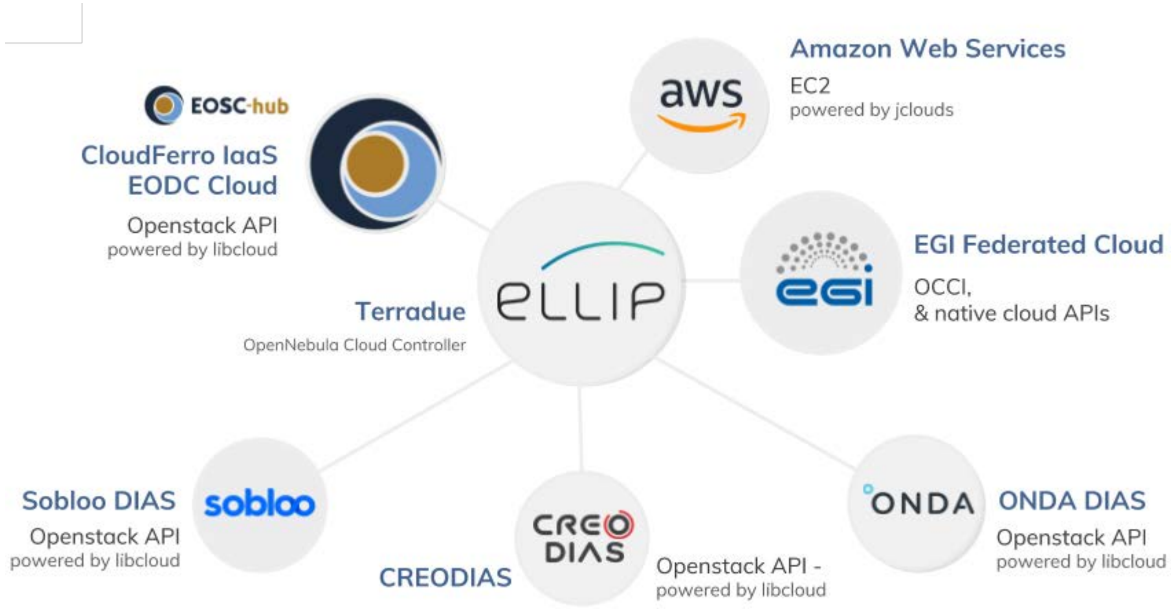
CREODIAS

ONDA

Terradue
Advancing Earth Science

GEP | Cloud Platform - Computing

- Continuous Integration and Deployment Environment with automatic packaging & deployment in production environments
- Improved Production Center, with (auto)scalability allowing cost-effective data processing on Cloud Computing
- Deployment in multiple Cloud-based processing environments with no lock-in on a Cloud provider



ellip
Terradue
OpenNebula Cloud Controller

EOSC-hub
CloudFerro IaaS
EODC Cloud
Openstack API
powered by libcloud

Amazon Web Services
EC2
powered by jclouds

EGI Federated Cloud
OCCI,
& native cloud APIs

Sobloo DIAS
Openstack API
powered by libcloud

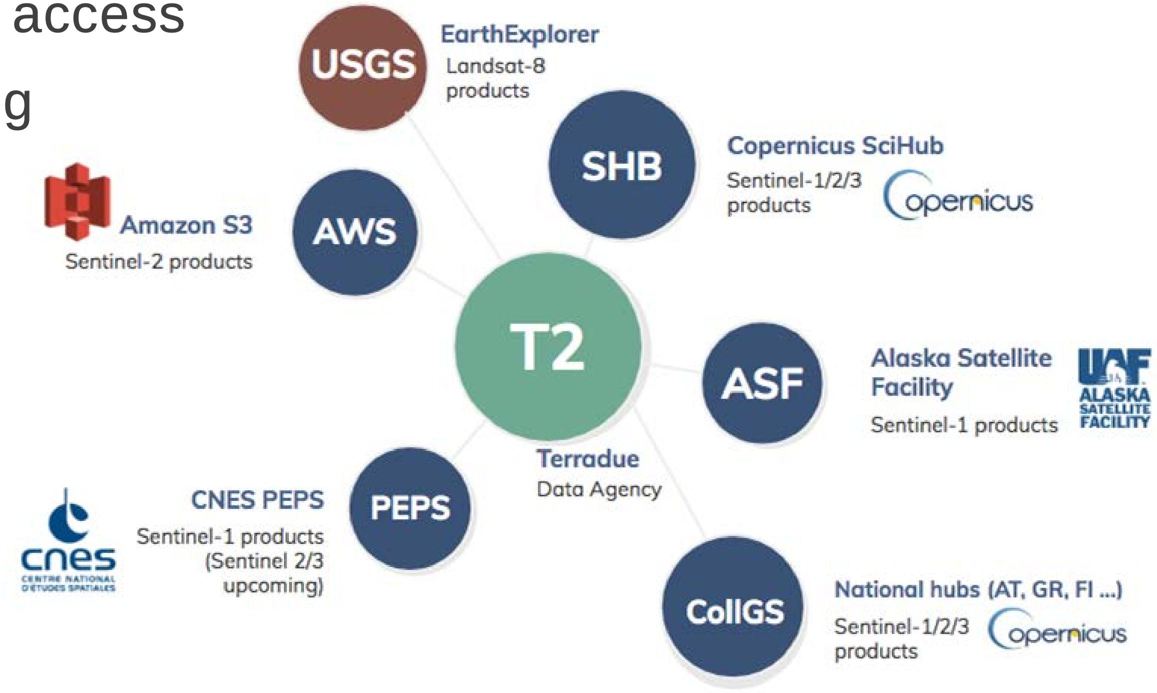
CREODIAS
Openstack API -
powered by libcloud

ONDA DIAS
Openstack API
powered by libcloud

GEP | Cloud Platform - Data

Enhanced Data Gateway using OpenSearch

- Automatic multi-sourcing to optimise data access
- Programmable and systematic data caching
- Data usage accounting
- Personal cloud storage (repository)



USGS
EarthExplorer
Landsat-8 products

SHB
Copernicus SciHub
Sentinel-1/2/3 products

AWS
Amazon S3
Sentinel-2 products

T2
Terradue
Data Agency

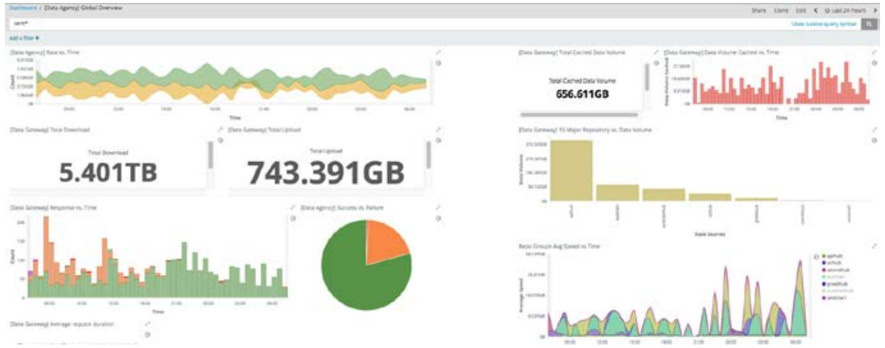
ASF
Alaska Satellite Facility
Sentinel-1 products

PEPS
CNES PEPS
Sentinel-1 products (Sentinel 2/3 upcoming)

CollGS
National hubs (AT, GR, FI ...)
Sentinel-1/2/3 products

Supported Data Providers

Daily figures



Using SNAP in GEP

ESA asked us to provide a feedback:

1. In what context are you using SNAP, which parts and to do what?
2. What other software do you use along with SNAP? Do they have to interact with each other?
3. What works and what doesn't?
4. What performances do you get (time, quality)?
5. What extra SNAP feature would help your work?
6. Did SNAP help you achieve everything that you expect it to?

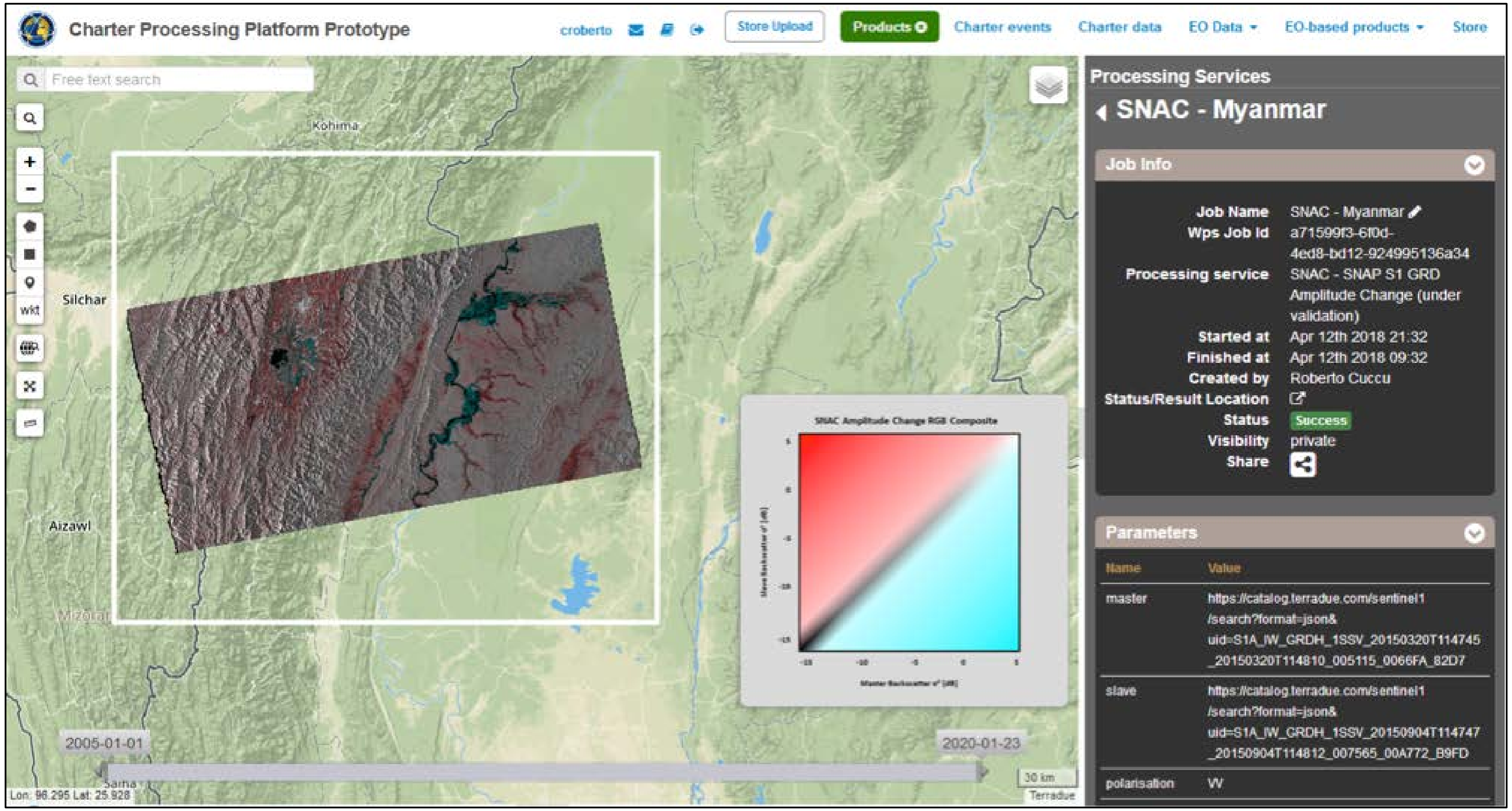
Some examples of SNAP services

- SNAC
- COIN
- COMBI
- Active fire detection with Sentinel-3
- Burned area assessment with Sentinel-2
- CSK interferogram generation

SNAC Sentinel-1 Amplitude Change

SNAC generates RGB composite of backscattering from a pair Sentinel-1 GRD IW and EW products (e.g. pre- and post-event)

https://terradue.github.io/doc-tep-geohazards/tutorials/rss_snap_s1_snac.html



Processing Services

SNAC - Myanmar

Job Info

Job Name SNAC - Myanmar
Wps Job Id a71599f3-6f0d-4ed8-bd12-924995136a34
Processing service SNAC - SNAP S1 GRD Amplitude Change (under validation)
Started at Apr 12th 2018 21:32
Finished at Apr 12th 2018 09:32
Created by Roberto Cuccu
Status/Result Location
Status Success
Visibility private
Share

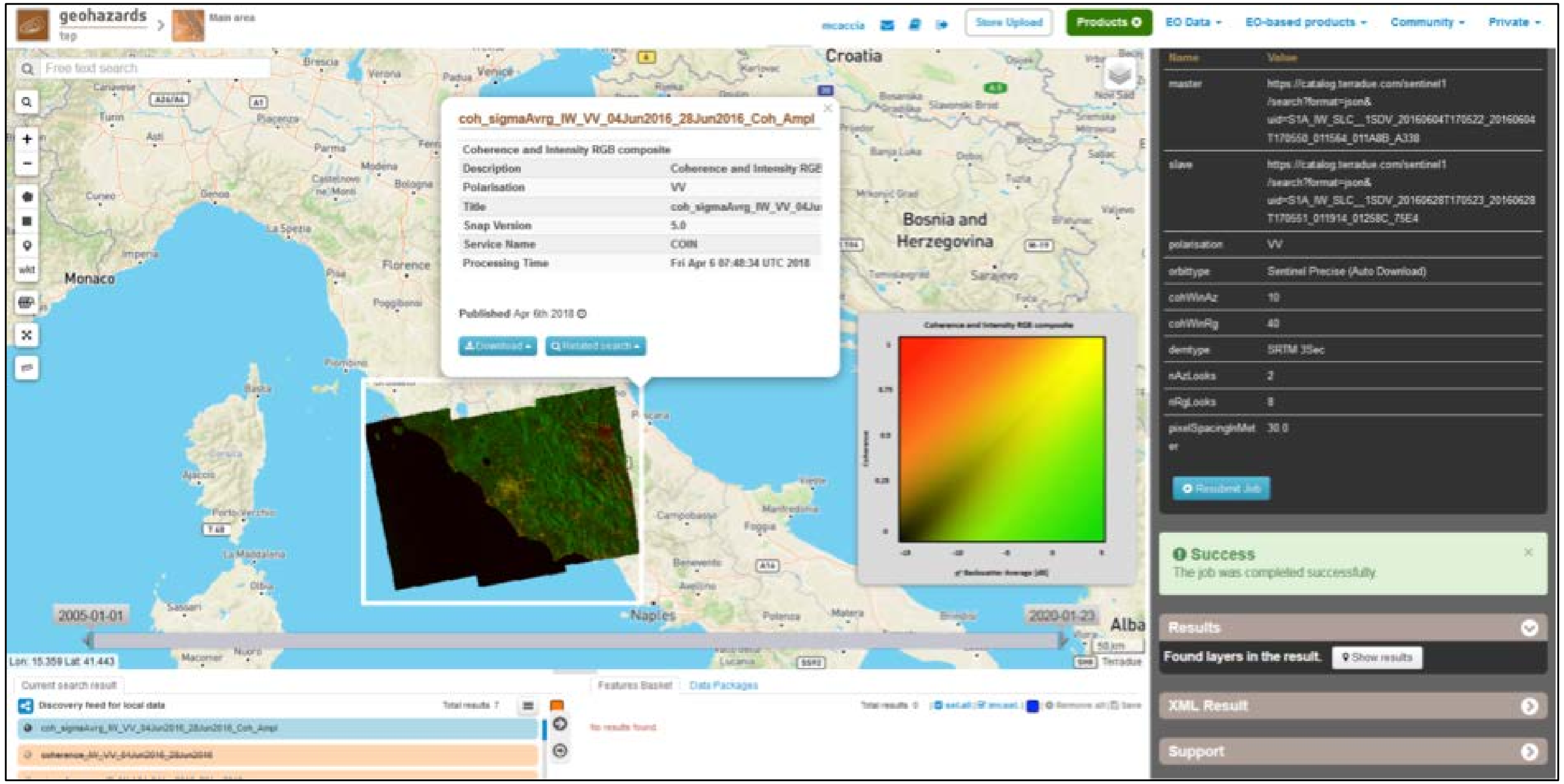
Parameters

Name	Value
master	https://catalog.terradue.com/sentinel1/search?format=json&uid=S1A_IW_GRDH_1SSV_20150320T114745_20150320T114810_005115_0066FA_82D7
slave	https://catalog.terradue.com/sentinel1/search?format=json&uid=S1A_IW_GRDH_1SSV_20150904T114747_20150904T114812_007565_00A772_B9FD
polarisation	VV

COIN Coherence and Intensity change for S1

COIN produces geocoded composites of coherence and amplitude images from a pair of Sentinel-1 TOPSAR IW data pairs.

https://terradue.github.io/doc-tep-geohazards/tutorials/rss_snap_s1_coin.html



coh_sigmaAvrg_IW_VV_04Jun2016_28Jun2016_Coh_Ampl

Coherence and Intensity RGB composite

Description	Coherence and Intensity RGE
Polarisation	VV
Title	coh_sigmaAvrg_IW_VV_04Ju
Snap Version	5.0
Service Name	COIN
Processing Time	Fri Apr 6 07:48:34 UTC 2018

Published Apr 6th 2018

Name	Value
master	https://catalog.terradue.com/sentinel1/search?format=json&uid=S1A_IW_SLC__1SDV_20160604T170522_20160604T170550_011564_011A8B_A338
slave	https://catalog.terradue.com/sentinel1/search?format=json&uid=S1A_IW_SLC__1SDV_20160628T170523_20160628T170551_011914_01258C_75E4
polarisation	VV
orbittype	Sentinel Precise (Auto Download)
cohWinAz	10
cohWinRg	40
demtype	SRTM 3Sec
nAzLooks	2
nRgLooks	8
pixelSpacingInMeter	30.0

Success
The job was completed successfully

Results

Found layers in the result.

XML Result

Support

Discovery feed for local data Total results 7

coh_sigmaAvrg_IW_VV_04Jun2016_28Jun2016_Coh_Ampl

coherence_IW_VV_04Jun2016_28Jun2016

Total results 0

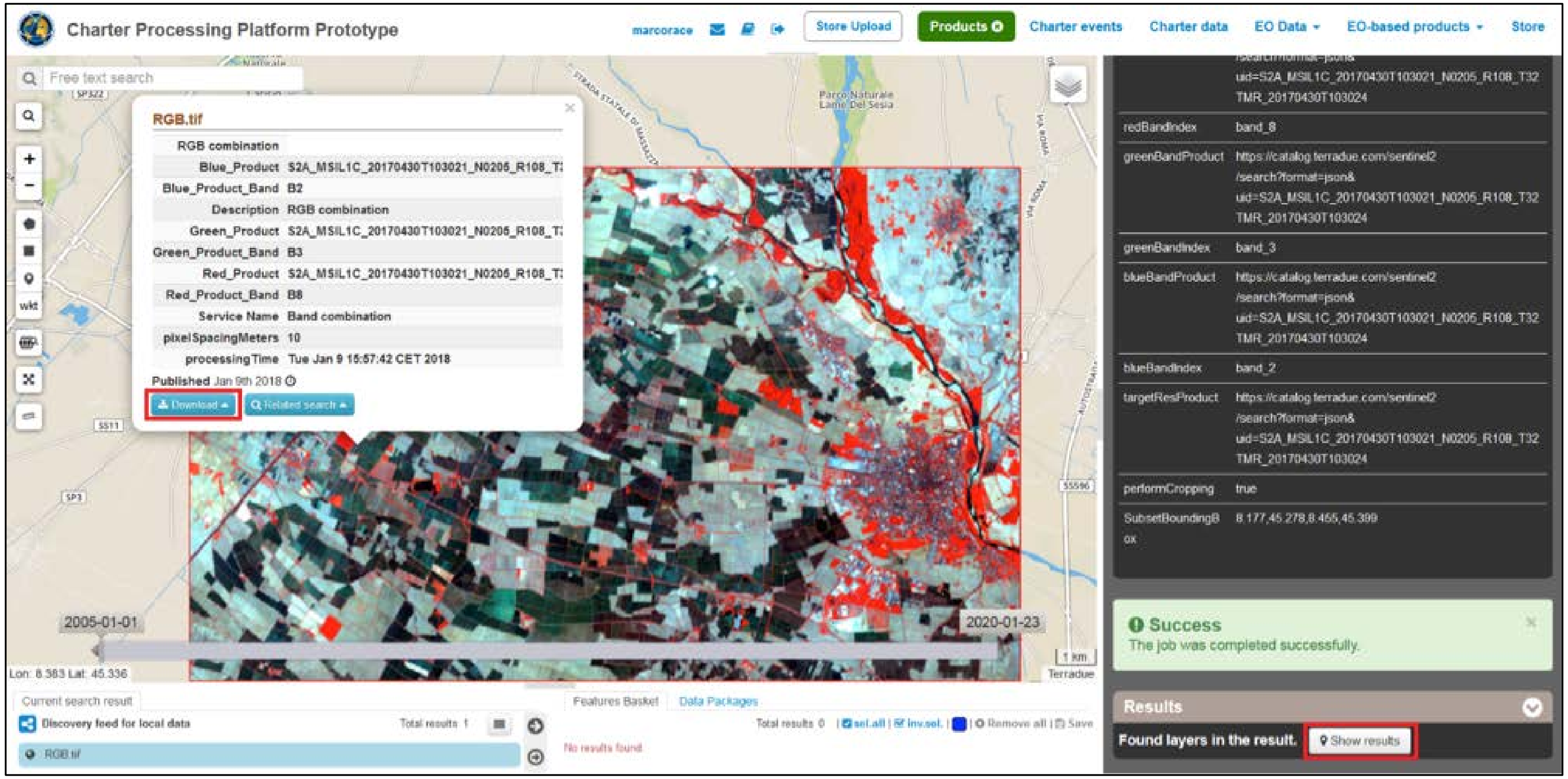
No results found.

COMBI Band Combination

RGB band combination from single or multiple EO data product user-defined band combinations from multi-mission Optical and SAR data.

Missions: ALOS, ALOS-2, Kanopus-V, KOMPSAT-2, KOMPSAT-3, KOMPSAT-5, GF2, Landsat 8, Pleiades 1A/1B, RADARSAT-2, RapidEye, Resurs-P, Sentinel-1, Sentinel-2, SPOT 6, SPOT 7, TerraSAR-X, VRSS1 and UK-DMC 2.

https://terradue.github.io/doc-tep-



RGB.tif

RGB combination	
Blue_Product	S2A_MSIL1C_20170430T103021_N0205_R108_T3
Blue_Product_Band	B2
Description	RGB combination
Green_Product	S2A_MSIL1C_20170430T103021_N0205_R108_T3
Green_Product_Band	B3
Red_Product	S2A_MSIL1C_20170430T103021_N0205_R108_T3
Red_Product_Band	B8
Service Name	Band combination
pixelSpacingMeters	10
processingTime	Tue Jan 9 15:57:42 CET 2018

Published Jan 9th 2018

	/search?format=json& uid=S2A_MSIL1C_20170430T103021_N0205_R108_T32TMR_20170430T103024
redBandIndex	band_8
greenBandProduct	https://catalog.terradue.com/sentinel2/search?format=json&uid=S2A_MSIL1C_20170430T103021_N0205_R108_T32TMR_20170430T103024
greenBandIndex	band_3
blueBandProduct	https://catalog.terradue.com/sentinel2/search?format=json&uid=S2A_MSIL1C_20170430T103021_N0205_R108_T32TMR_20170430T103024
blueBandIndex	band_2
targetResProduct	https://catalog.terradue.com/sentinel2/search?format=json&uid=S2A_MSIL1C_20170430T103021_N0205_R108_T32TMR_20170430T103024
performCropping	true
SubsetBoundingBox	8.177,45.278,8.465,45.399

Success
The job was completed successfully.

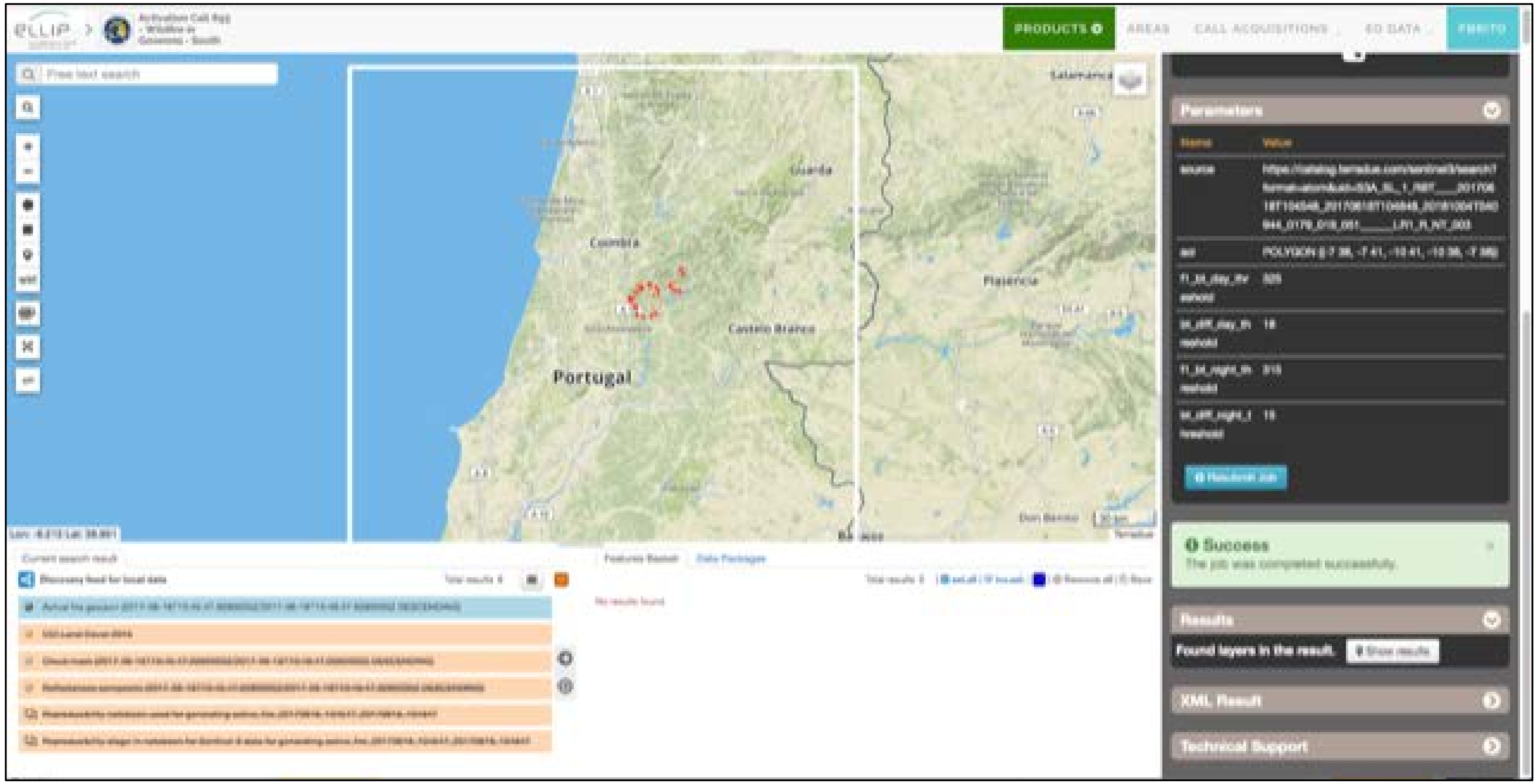
Results

Found layers in the result. Show results

Active fire detection with Sentinel-3

Uses Sentinel-3 SLSTR to detect hot spots to generate a product including:

- A geojson with the hotspots
- A RGB composite for descending acquisitions
- A cloud mask
- CCI Land Cover



Discovery feed for local data Total results 6

- Active fire geojson 2017-08-18T10:45:47.808000Z/2017-08-18T10:48:47.808000Z DESCENDING
- CCI Land Cover 2015
- Cloud mask 2017-08-18T10:45:47.808000Z/2017-08-18T10:48:47.808000Z DESCENDING
- Reflectances composite 2017-08-18T10:45:47.808000Z/2017-08-18T10:48:47.808000Z DESCENDING
- Reproducibility notebook used for generating active_fire 20170818_104547-20170818_104847
- Reproducibility stage-in notebook for Sentinel-3 data for generating active_fire 20170818_104547-20170818_104847

No results found

Parameters

Name	Value
source	https://catalog.terradue.com/sentinel3/search?format=atom&uid=S3A_SL_1_RBT____20170818T104548_20170818T104848_20181004T040944_0179_018_051______LR1_R_NT_003
aoi	POLYGON ((-7 38, -7 41, -10 41, -10 38, -7 38))
f1_bt_day_threshold	325
bt_diff_day_threshold	18
f1_bt_night_threshold	315
bt_diff_night_threshold	15

Resubmit Job

Success
The job was completed successfully.

Results

Found layers in the result. Show results

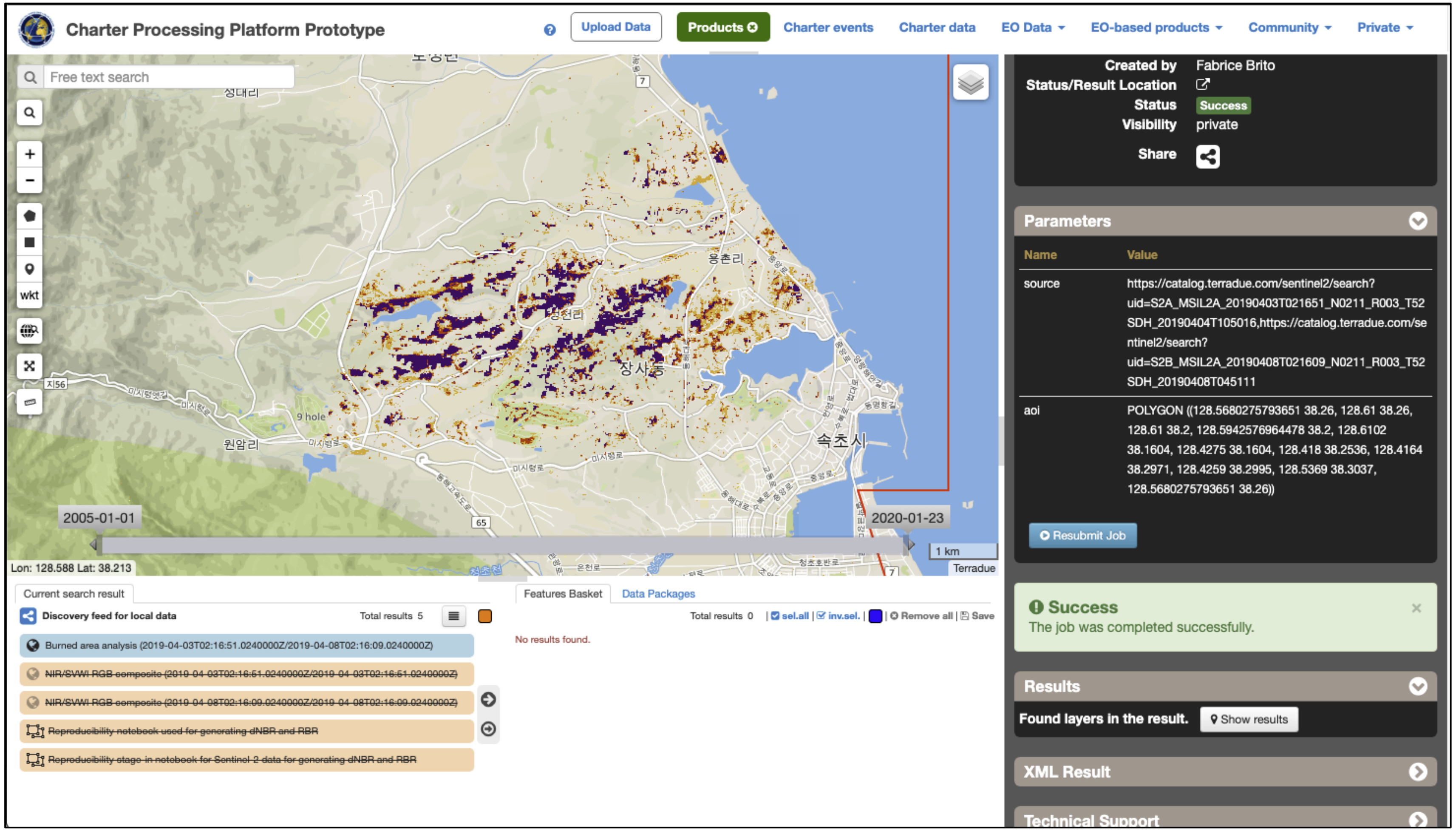
XML Result

Technical Support

Burned area assessment with Sentinel-2

Sentinel-2 burned area assessment including:

- RGB composite B12, B11, B8A
- dNBR delta normalized burn ratio
- RBR Relativized Burn Ratio



Created by	Fabrice Brito
Status/Result Location	
Status	Success
Visibility	private
Share	

Parameters

Name	Value
source	https://catalog.terradue.com/sentinel2/search?uid=S2A_MSIL2A_20190403T021651_N0211_R003_T52SDH_20190404T105016,https://catalog.terradue.com/sentinel2/search?uid=S2B_MSIL2A_20190408T021609_N0211_R003_T52SDH_20190408T045111
aoi	POLYGON ((128.5680275793651 38.26, 128.61 38.26, 128.61 38.2, 128.5942576964478 38.2, 128.6102 38.1604, 128.4275 38.1604, 128.418 38.2536, 128.4164 38.2971, 128.4259 38.2995, 128.5369 38.3037, 128.5680275793651 38.26))

Resubmit Job

Success
The job was completed successfully.

Results
Found layers in the result. Show results

XML Result

Technical Support

Current search result

Discovery feed for local data Total results 5

- Burned area analysis (2019-04-03T02:16:51.0240000Z/2019-04-08T02:16:09.0240000Z)
- ~~NIR/SWIR RGB composite (2019-04-03T02:16:51.0240000Z/2019-04-03T02:16:51.0240000Z)~~
- ~~NIR/SWIR RGB composite (2019-04-08T02:16:09.0240000Z/2019-04-08T02:16:09.0240000Z)~~
- ~~Reproducibility notebook used for generating dNBR and RBR~~
- ~~Reproducibility stage-in notebook for Sentinel-2 data for generating dNBR and RBR~~

Features Basket | Data Packages

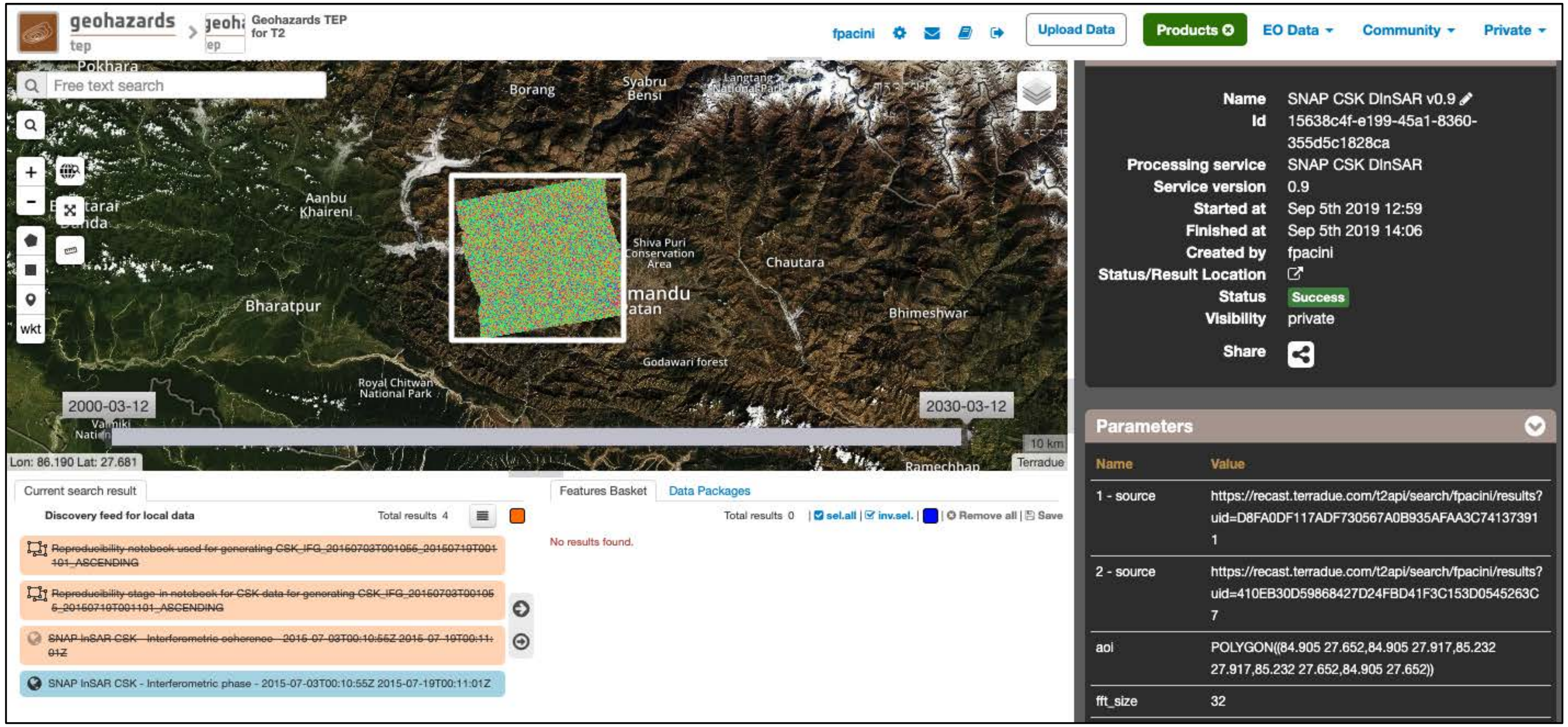
Total results 0 | ☑ sel.all | ☑ inv.sel. | ■ | Remove all | Save

No results found.

CSK interferogram generation

COSMO-SkyMed Differential SAR Interferometry using SNAP.

This service performs an InSAR workflow on a pair (master, slave) of COSMO-SkyMed single look complex (L1A SCS) acquisitions producing interferograms and coherence map



Current search result

Discovery feed for local data Total results 4

~~Reproducibility notebook used for generating CSK_IFG_20150703T001055_20150719T001101_ASCENDING~~

~~Reproducibility stage-in notebook for CSK data for generating CSK_IFG_20150703T001055_20150719T001101_ASCENDING~~

~~SNAP InSAR CSK - Interferometric coherence - 2015-07-03T00:10:55Z 2015-07-19T00:11:01Z~~

SNAP InSAR CSK - Interferometric phase - 2015-07-03T00:10:55Z 2015-07-19T00:11:01Z

Features Basket | Data Packages

Total results 0 | ☑ sel.all | ☑ inv.sel. | | Remove all | Save

No results found.

Name	SNAP CSK DInSAR v0.9
Id	15638c4f-e199-45a1-8360-355d5c1828ca
Processing service	SNAP CSK DInSAR
Service version	0.9
Started at	Sep 5th 2019 12:59
Finished at	Sep 5th 2019 14:06
Created by	fpacini
Status/Result Location	
Status	Success
Visibility	private
Share	

Parameters

Name	Value
1 - source	https://recast.terradue.com/t2api/search/fpacini/results?uid=D8FA0DF117ADF730567A0B935AFAA3C741373911
2 - source	https://recast.terradue.com/t2api/search/fpacini/results?uid=410EB30D59868427D24FBD41F3C153D0545263C7
aoi	POLYGON((84.905 27.652,84.905 27.917,85.232 27.917,85.232 27.652,84.905 27.652))
fft_size	32

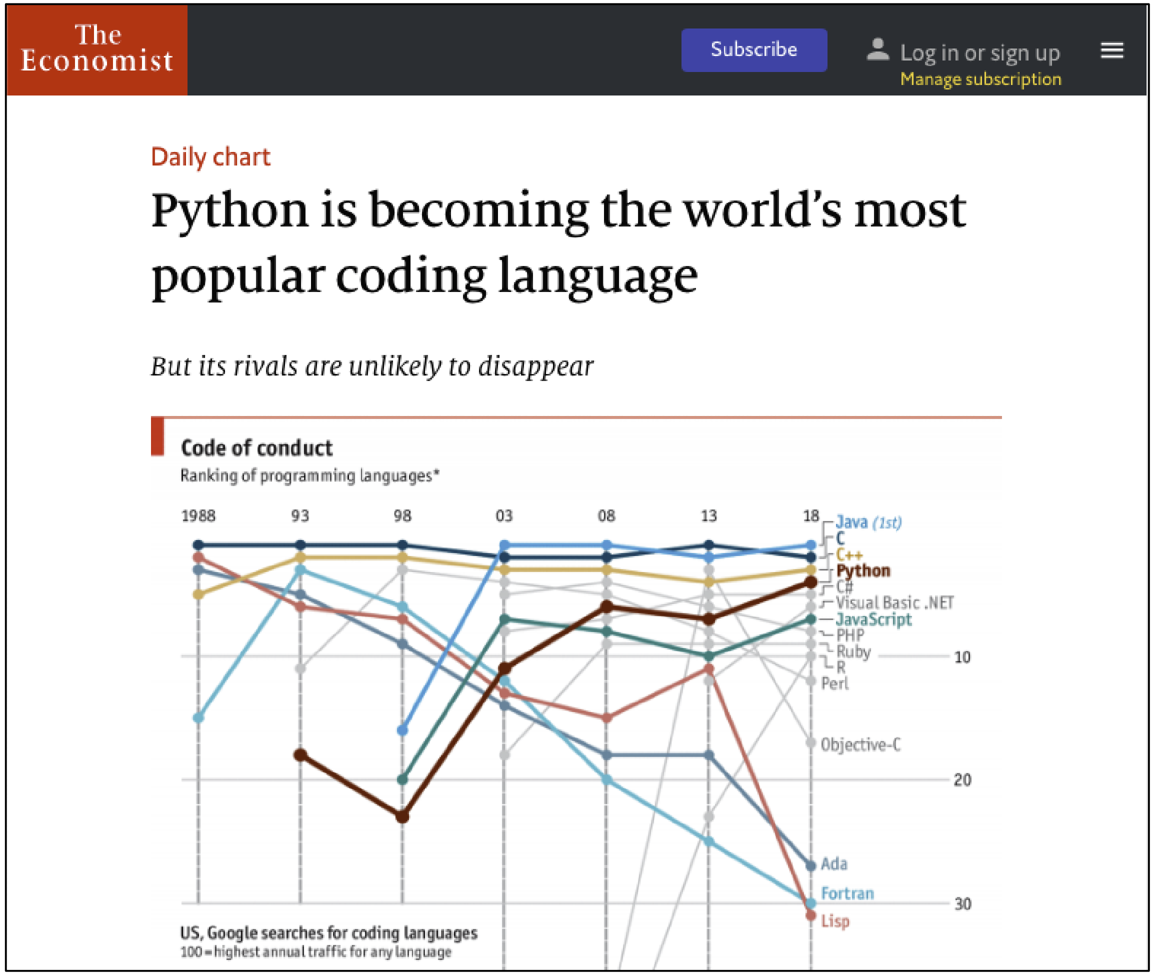
Other software

- **Libraries and toolboxes:**
  - Orfeo Toolbox
  - Gdal
  - pandas/geopandas
  - NumPy
- **Environments**
  - Jupyter
  - Xarray
  - Dask
- **Processing**
  - YARN, OOZIE
  - Kubernetes

The importance of Snappy

“I CERTAINLY didn’t set out to create a language that was intended for mass consumption,” says **Guido van Rossum**, a Dutch computer scientist who devised **Python**, a programming language, in 1989.

But nearly three decades on, his invention has overtaken almost all of its rivals and brought coding to the fingertips of people who were once baffled by it.



Daily chart

Python is becoming the world’s most popular coding language

But its rivals are unlikely to disappear

Code of conduct

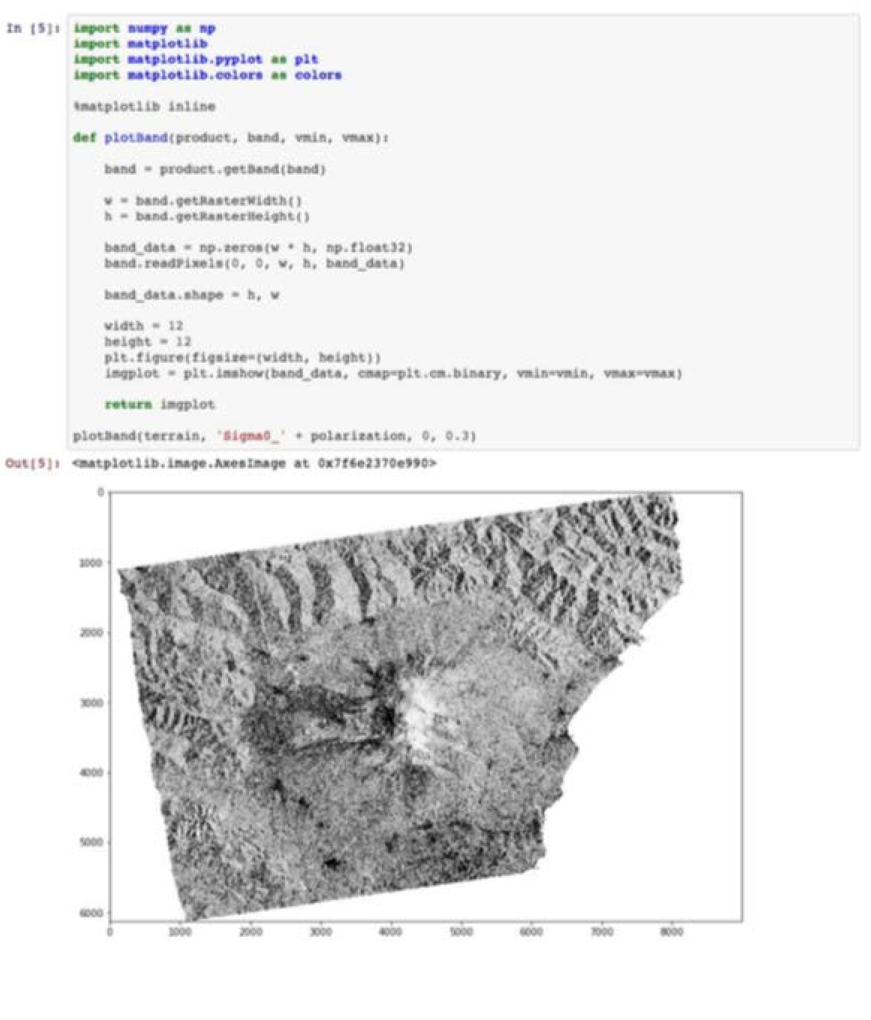
Ranking of programming languages*

US, Google searches for coding languages
100=highest annual traffic for any language

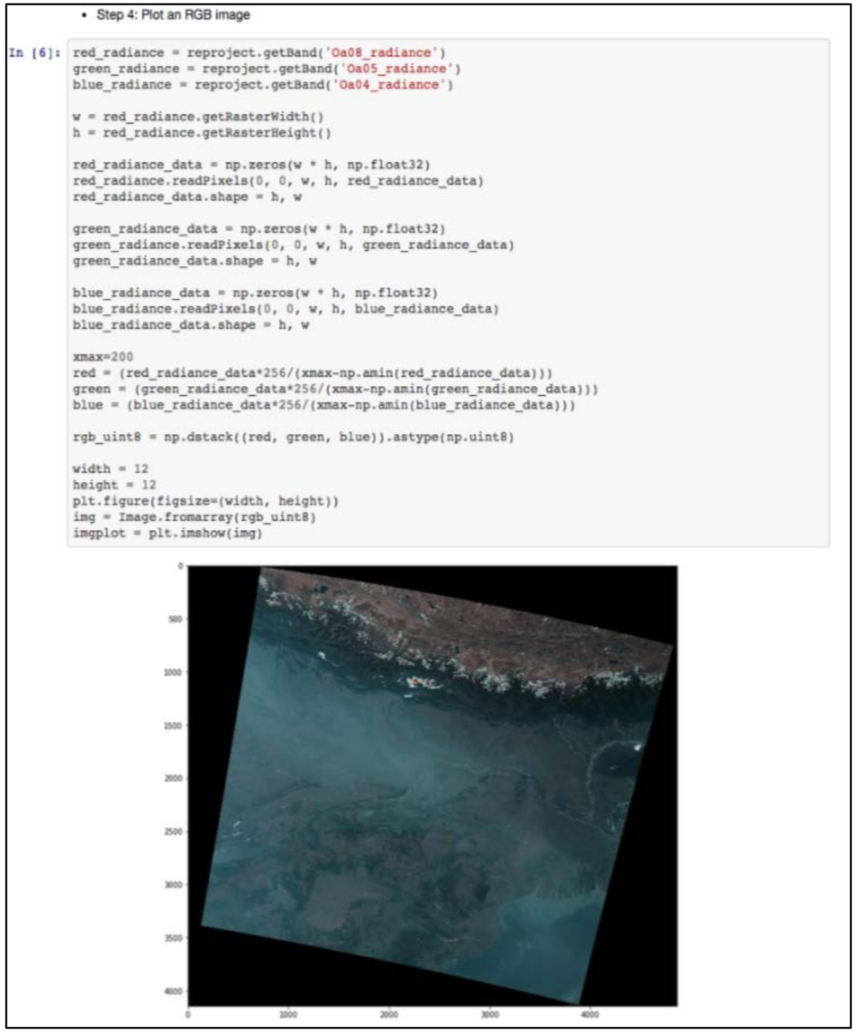
https://www.economist.com/graphic-detail/2018/07/26/python-is-becoming-the-worlds-most-popular-coding-language

Data Exploitation: Access to EO Products

Using Snappy in notebooks to plot and analyse
Sentinel data



```
In [5]: import numpy as np
        import matplotlib
        import matplotlib.pyplot as plt
        import matplotlib.colors as colors

        %matplotlib inline

        def plotBand(product, band, vmin, vmax):

            band = product.getBand(band)

            w = band.getRasterWidth()
            h = band.getRasterHeight()

            band_data = np.zeros(w * h, np.float32)
            band.readPixels(0, 0, w, h, band_data)

            band_data.shape = h, w

            width = 12
            height = 12
            plt.figure(figsize=(width, height))
            imgplot = plt.imshow(band_data, cmap=plt.cm.binary, vmin=vmin, vmax=vmax)

            return imgplot

        plotBand(terrain, 'Sigma0_' + polarization, 0, 0.3)

Out[5]: <matplotlib.image.AxesImage at 0x7f6e2370e990>
```



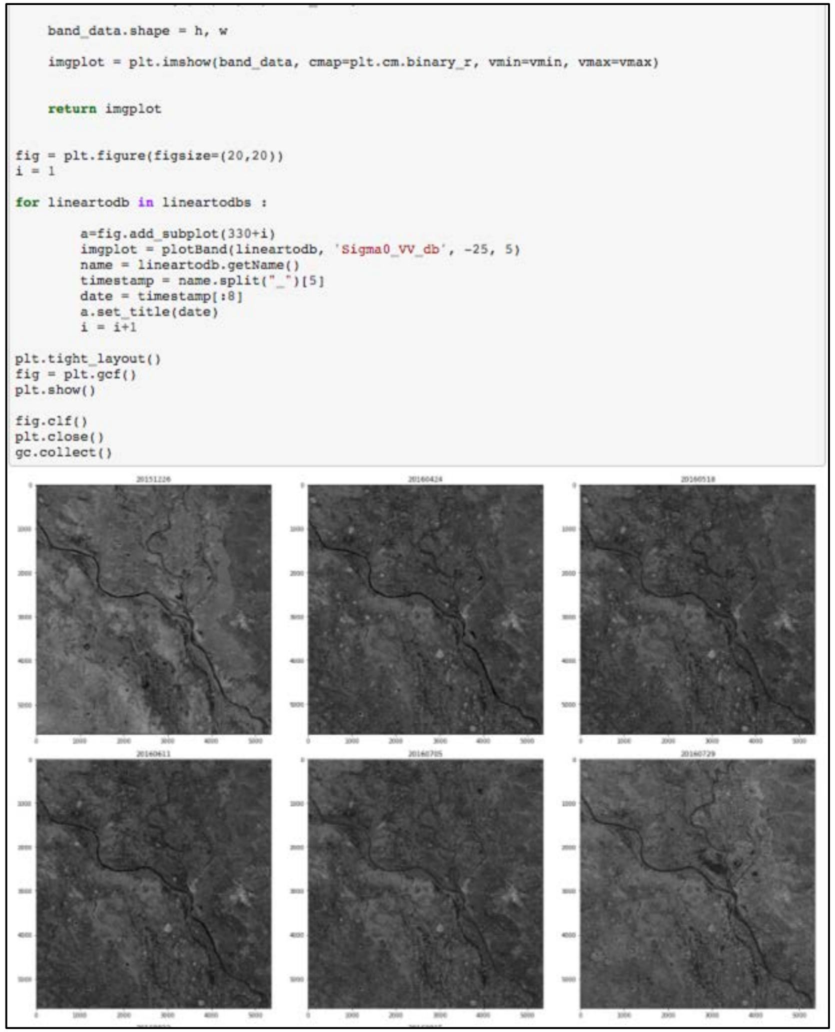
- Step 4: Plot an RGB image

```
In [6]: red_radiance = reproject.getBand('Oa08_radiance')
        green_radiance = reproject.getBand('Oa05_radiance')
        blue_radiance = reproject.getBand('Oa04_radiance')

        w = red_radiance.getRasterWidth()
        h = red_radiance.getRasterHeight()

        red_radiance_data = np.zeros(w * h, np.float32)
        red_radiance.readPixels(0, 0, w, h, red_radiance_data)
        red_radiance_data.shape = h, w

        green_radiance_data = np.zeros(w * h, np.float32)
        green_radiance.readPixels(0, 0, w, h, green_radiance_data)
        green_radiance_data.shape = h, w

        blue_radiance_data = np.zeros(w * h, np.float32)
        blue_radiance.readPixels(0, 0, w, h, blue_radiance_data)
        blue_radiance_data.shape = h, w

        xmax=200
        red = (red_radiance_data*256/(xmax-np.amin(red_radiance_data)))
        green = (green_radiance_data*256/(xmax-np.amin(green_radiance_data)))
        blue = (blue_radiance_data*256/(xmax-np.amin(blue_radiance_data)))

        rgb_uint8 = np.dstack((red, green, blue)).astype(np.uint8)

        width = 12
        height = 12
        plt.figure(figsize=(width, height))
        img = Image.fromarray(rgb_uint8)
        imgplot = plt.imshow(img)
```

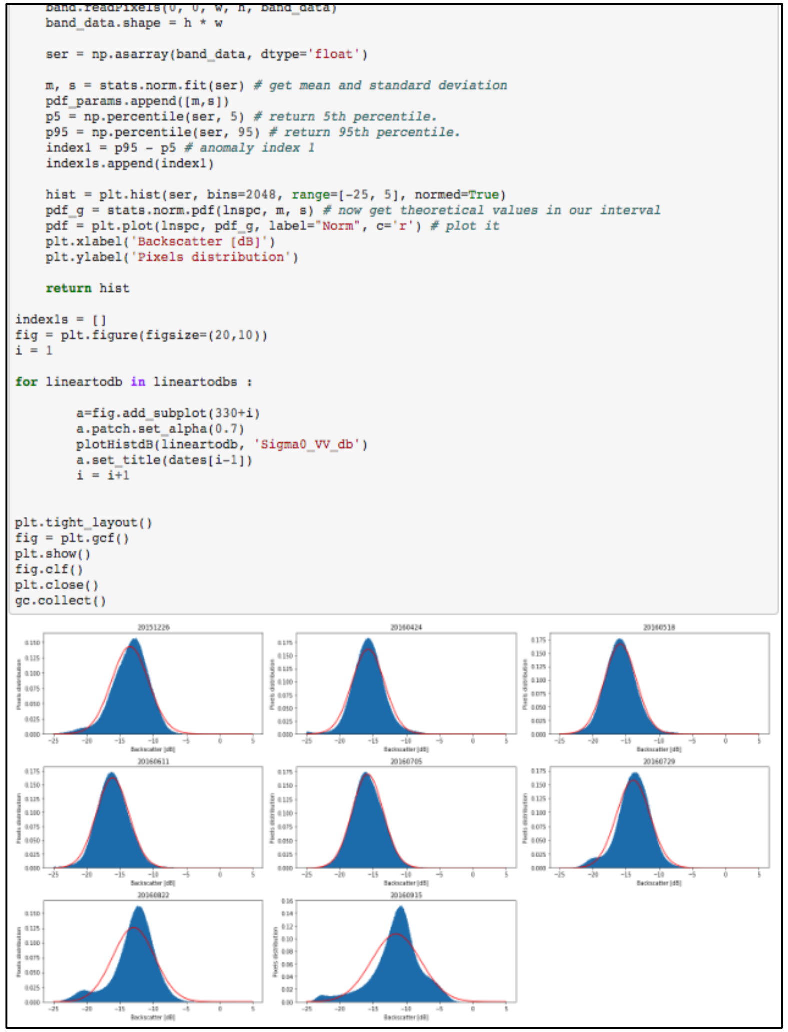
Data Exploitation: Time Series Analyses

Using Snappy to analyse a stack of GRD Sentinel-1 data

Backscatter profiles for reference image used in flood change detection analysis



```
    band_data.shape = h, w

    imgplot = plt.imshow(band_data, cmap=plt.cm.binary_r, vmin=vmin, vmax=vmax)


    return imgplot


fig = plt.figure(figsize=(20,20))
i = 1

for lineartodb in lineartodbs :

        a=fig.add_subplot(330+i)
        imgplot = plotBand(lineartodb, 'Sigma0_VV_db', -25, 5)
        name = lineartodb.getName()
        timestamp = name.split("_")[5]
        date = timestamp[:8]
        a.set_title(date)
        i = i+1

plt.tight_layout()
fig = plt.gcf()
plt.show()

fig.clf()
plt.close()
gc.collect()
```



```
    band.readPixels(0, 0, w, h, band_data)
    band_data.shape = h * w

    ser = np.asarray(band_data, dtype='float')

    m, s = stats.norm.fit(ser) # get mean and standard deviation
    pdf_params.append([m,s])
    p5 = np.percentile(ser, 5) # return 5th percentile.
    p95 = np.percentile(ser, 95) # return 95th percentile.
    index1 = p95 - p5 # anomaly index 1
    index1s.append(index1)

    hist = plt.hist(ser, bins=2048, range=[-25, 5], normed=True)
    pdf_g = stats.norm.pdf(lnspc, m, s) # now get theoretical values in our interval
    pdf = plt.plot(lnspc, pdf_g, label="Norm", c='r') # plot it
    plt.xlabel('Backscatter [dB]')
    plt.ylabel('Pixels distribution')

    return hist

index1s = []
fig = plt.figure(figsize=(20,10))
i = 1

for lineartodb in lineartodbs :

        a=fig.add_subplot(330+i)
        a.patch.set_alpha(0.7)
        plotHistdB(lineartodb, 'Sigma0_VV_db')
        a.set_title(dates[i-1])
        i = i+1


plt.tight_layout()
fig = plt.gcf()
plt.show()
fig.clf()
plt.close()
gc.collect()
```

TERRADUE

Advancing Earth Science

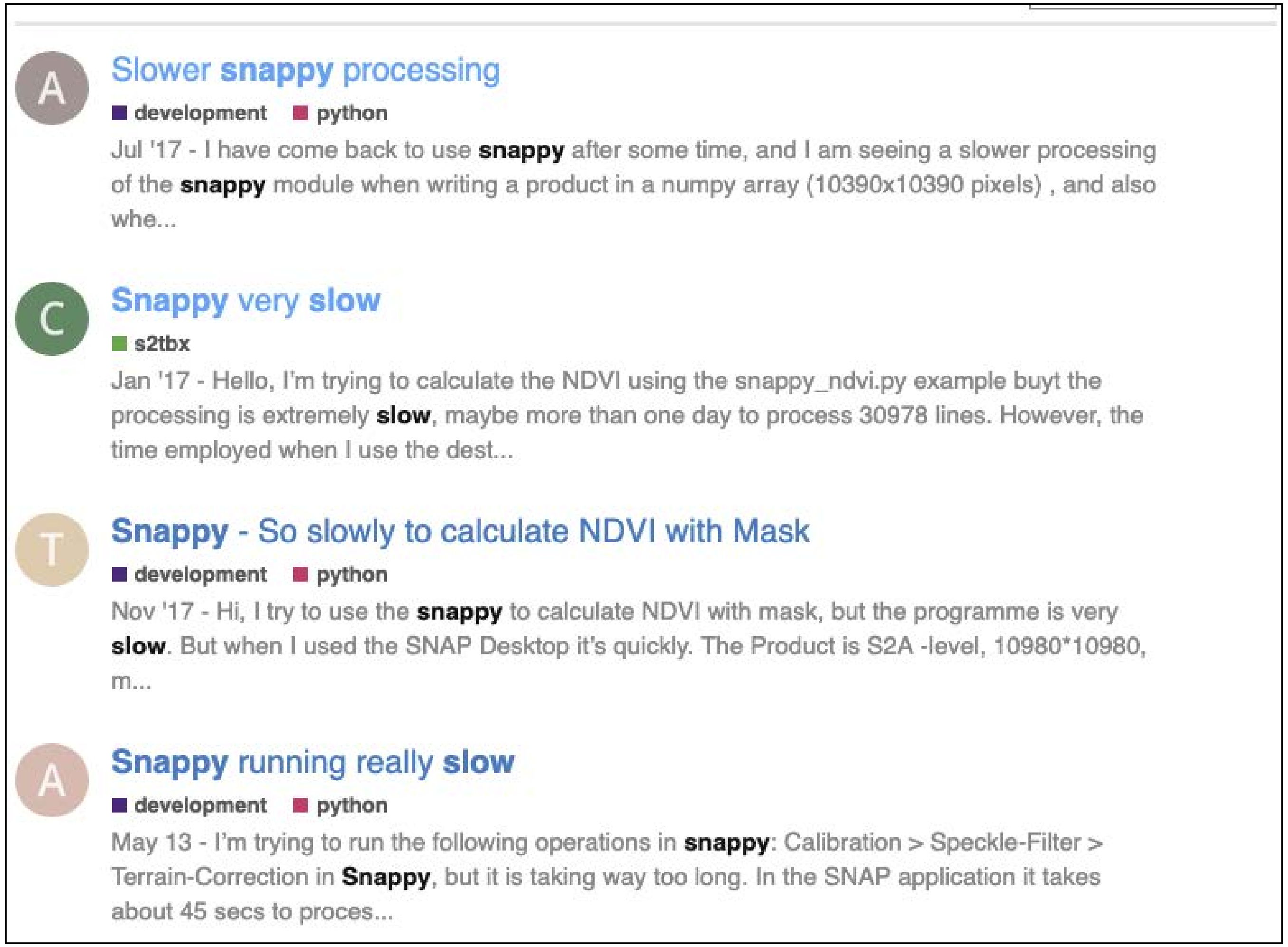
What doesn’t work: Snappy

Snappy is slow compared to GPT

Snappy has memory issues, results with Snappy are not the same as with GPT

Conclusion:

Snappy is barely used in our services, maybe just for inspecting band names for example. Sad.



Slower **snappy** processing

■ development ■ python

Jul '17 - I have come back to use **snappy** after some time, and I am seeing a slower processing of the **snappy** module when writing a product in a numpy array (10390x10390 pixels) , and also whe...

Snappy very **slow**

■ s2tbx

Jan '17 - Hello, I'm trying to calculate the NDVI using the snappy_ndvi.py example buyt the processing is extremely **slow**, maybe more than one day to process 30978 lines. However, the time employed when I use the dest...

Snappy - So slowly to calculate NDVI with Mask

■ development ■ python

Nov '17 - Hi, I try to use the **snappy** to calculate NDVI with mask, but the programme is very **slow**. But when I used the SNAP Desktop it's quickly. The Product is S2A -level, 10980*10980, m...

Snappy running really **slow**

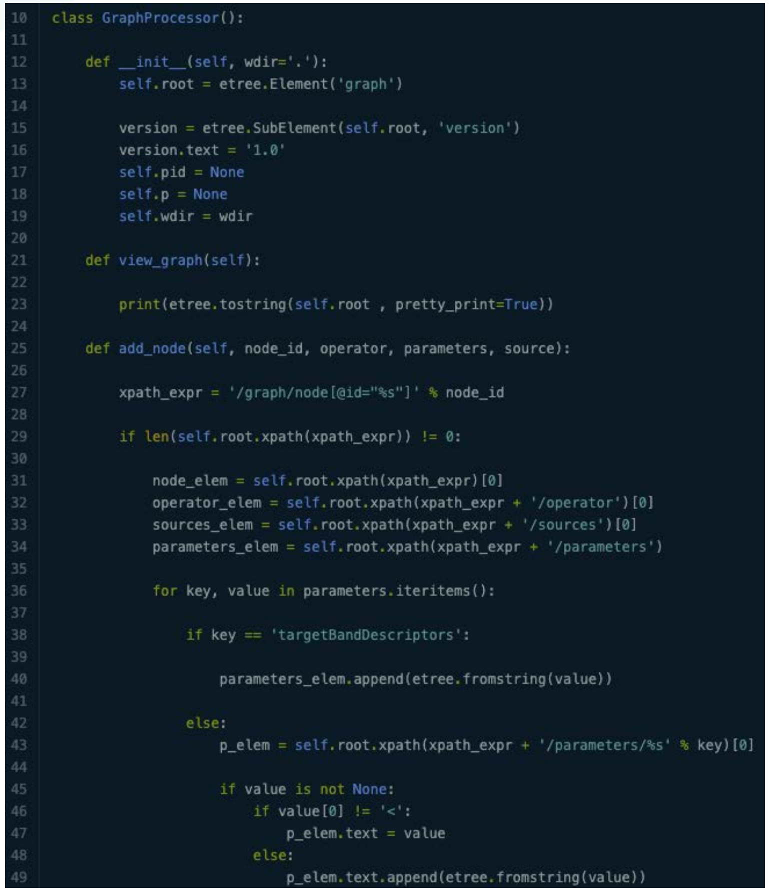
■ development ■ python

May 13 - I'm trying to run the following operations in **snappy**: Calibration > Speckle-Filter > Terrain-Correction in **Snappy**, but it is taking way too long. In the SNAP application it takes about 45 secs to proces...

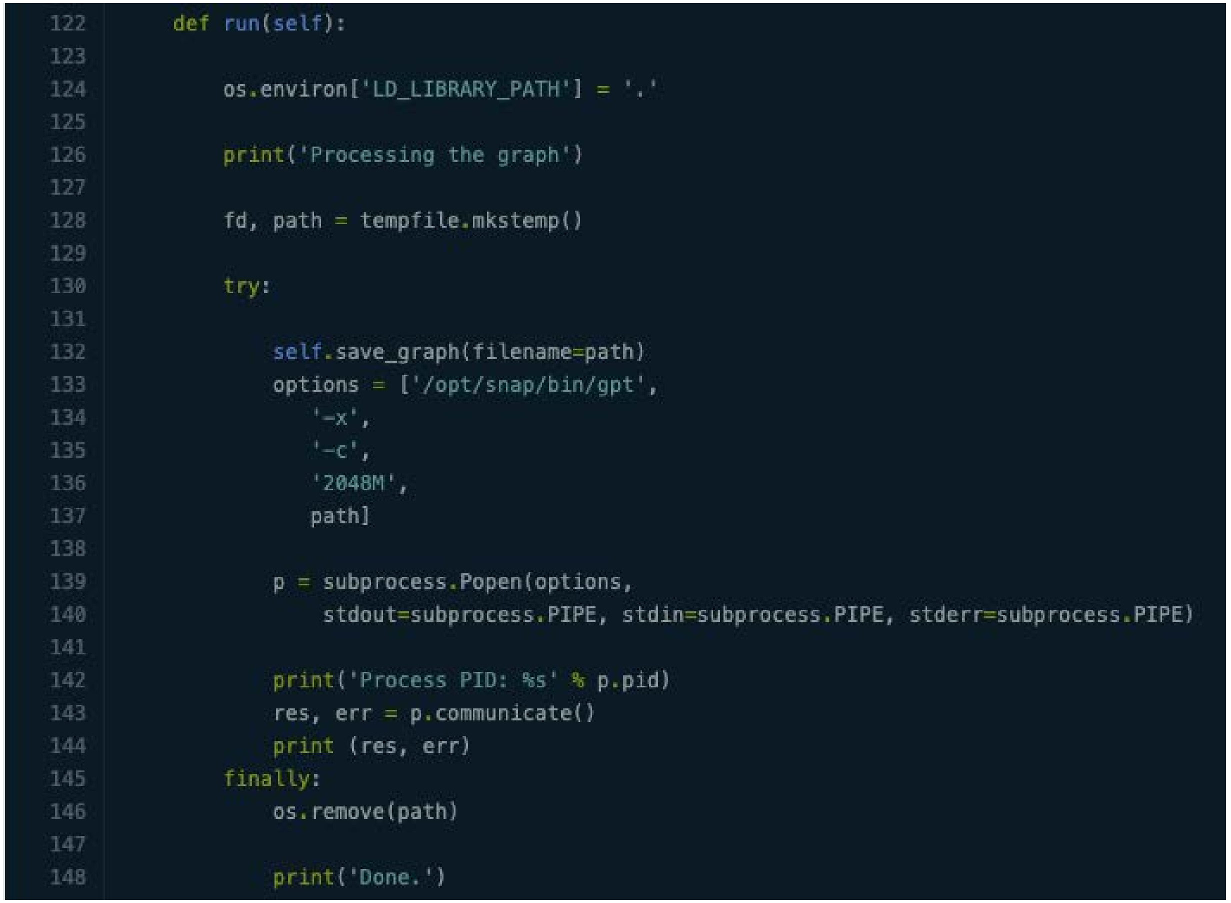
Terradue

Advancing Earth Science

What doesn’t work: Snappy



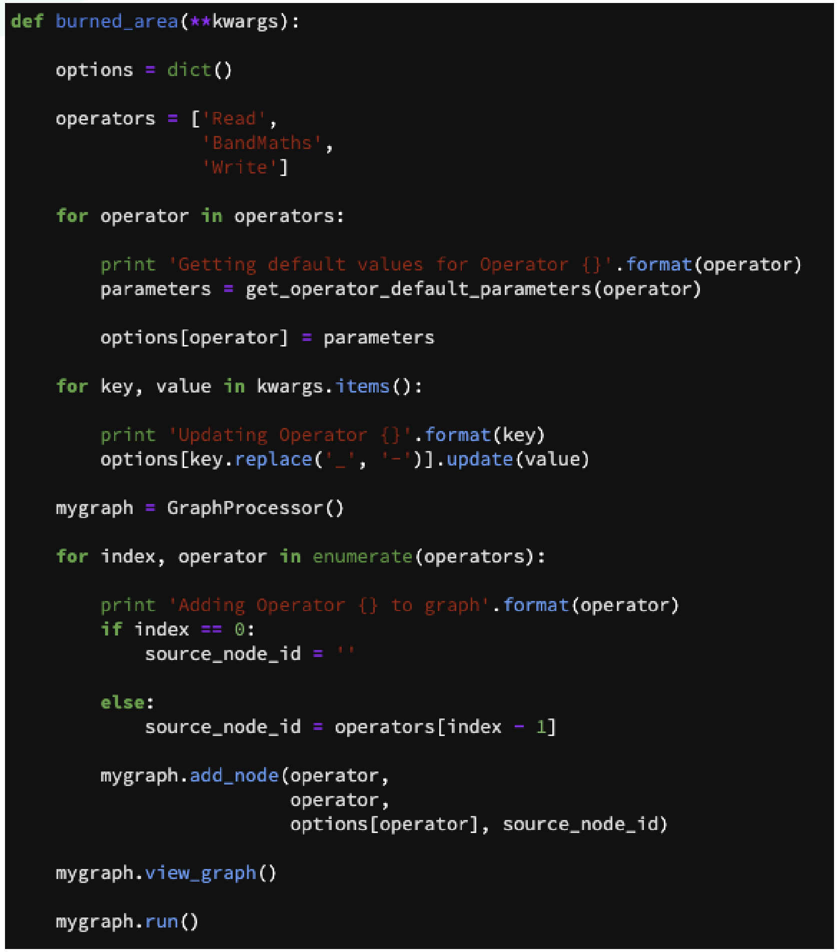
```python
class GraphProcessor():

    def __init__(self, wdir='.'):
        self.root = etree.Element('graph')

        version = etree.SubElement(self.root, 'version')
        version.text = '1.0'
        self.pid = None
        self.p = None
        self.wdir = wdir

    def view_graph(self):

        print(etree.tostring(self.root , pretty_print=True))

    def add_node(self, node_id, operator, parameters, source):

        xpath_expr = '/graph/node[@id="%s"]' % node_id

        if len(self.root.xpath(xpath_expr)) != 0:

            node_elem = self.root.xpath(xpath_expr)[0]
            operator_elem = self.root.xpath(xpath_expr + '/operator')[0]
            sources_elem = self.root.xpath(xpath_expr + '/sources')[0]
            parameters_elem = self.root.xpath(xpath_expr + '/parameters')

            for key, value in parameters.iteritems():

                if key == 'targetBandDescriptors':

                    parameters_elem.append(etree.fromstring(value))

                else:
                    p_elem = self.root.xpath(xpath_expr + '/parameters/%s' % key)[0]

                    if value is not None:
                        if value[0] != '<':
                            p_elem.text = value
                        else:
                            p_elem.text.append(etree.fromstring(value))
```



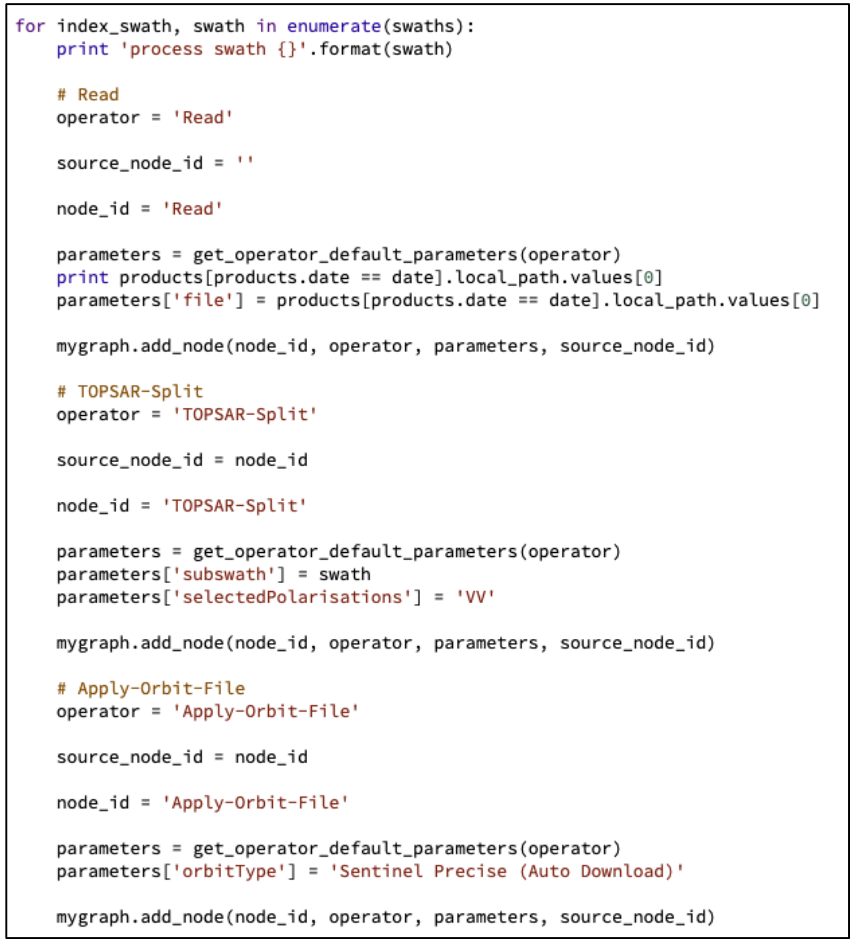
```python
    def run(self):

        os.environ['LD_LIBRARY_PATH'] = '.'

        print('Processing the graph')

        fd, path = tempfile.mkstemp()

        try:

            self.save_graph(filename=path)
            options = ['/opt/snap/bin/gpt',
               '-x',
               '-c',
               '2048M',
               path]

            p = subprocess.Popen(options,
                stdout=subprocess.PIPE, stdin=subprocess.PIPE, stderr=subprocess.PIPE)

            print('Process PID: %s' % p.pid)
            res, err = p.communicate()
            print (res, err)
        finally:
            os.remove(path)

            print('Done.')
```

We all had to write code to generate the graph XML and do a system call to /opt/snap/bin/gpt

What doesn’t work: Snappy



```
def burned_area(**kwargs):

    options = dict()

    operators = ['Read',
                 'BandMaths',
                 'Write']

    for operator in operators:

        print 'Getting default values for Operator {}'.format(operator)
        parameters = get_operator_default_parameters(operator)

        options[operator] = parameters

    for key, value in kwargs.items():

        print 'Updating Operator {}'.format(key)
        options[key.replace('_', '-')].update(value)

    mygraph = GraphProcessor()

    for index, operator in enumerate(operators):

        print 'Adding Operator {} to graph'.format(operator)
        if index == 0:
            source_node_id = ''

        else:
            source_node_id = operators[index - 1]

        mygraph.add_node(operator,
                         operator,
                         options[operator], source_node_id)

    mygraph.view_graph()

    mygraph.run()
```

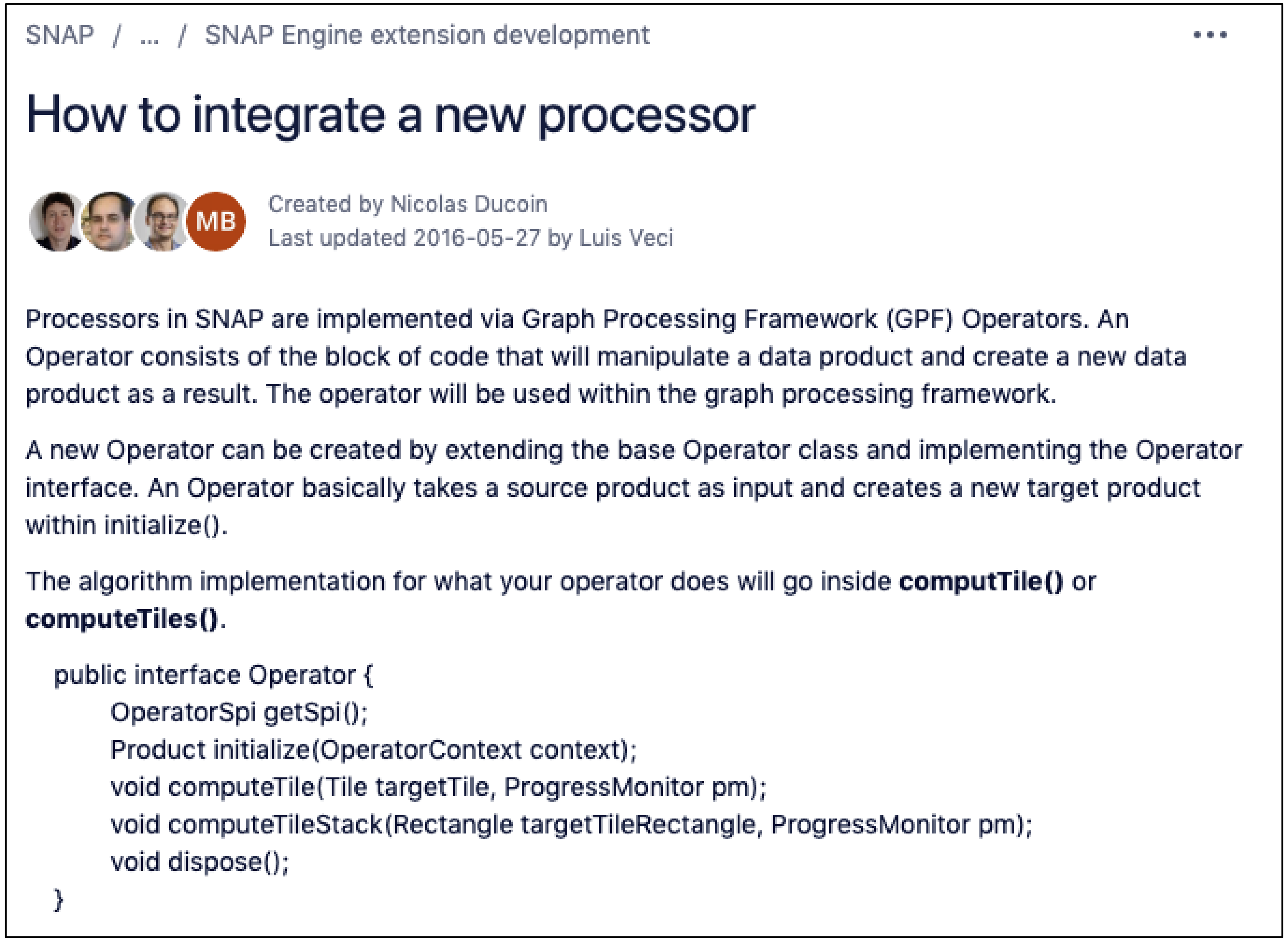


```
for index_swath, swath in enumerate(swaths):
    print 'process swath {}'.format(swath)

    # Read
    operator = 'Read'

    source_node_id = ''

    node_id = 'Read'

    parameters = get_operator_default_parameters(operator)
    print products[products.date == date].local_path.values[0]
    parameters['file'] = products[products.date == date].local_path.values[0]

    mygraph.add_node(node_id, operator, parameters, source_node_id)

    # TOPSAR-Split
    operator = 'TOPSAR-Split'

    source_node_id = node_id

    node_id = 'TOPSAR-Split'

    parameters = get_operator_default_parameters(operator)
    parameters['subswath'] = swath
    parameters['selectedPolarisations'] = 'VV'

    mygraph.add_node(node_id, operator, parameters, source_node_id)

    # Apply-Orbit-File
    operator = 'Apply-Orbit-File'

    source_node_id = node_id

    node_id = 'Apply-Orbit-File'

    parameters = get_operator_default_parameters(operator)
    parameters['orbitType'] = 'Sentinel Precise (Auto Download)'

    mygraph.add_node(node_id, operator, parameters, source_node_id)
```

Examples of such “wrapping code”

What is great: gpt processor

SNAP provides an excellent framework to develop additional processors



SNAP / ... / SNAP Engine extension development

How to integrate a new processor

Created by Nicolas Ducoin
Last updated 2016-05-27 by Luis Veci

Processors in SNAP are implemented via Graph Processing Framework (GPF) Operators. An Operator consists of the block of code that will manipulate a data product and create a new data product as a result. The operator will be used within the graph processing framework.

A new Operator can be created by extending the base Operator class and implementing the Operator interface. An Operator basically takes a source product as input and creates a new target product within initialize().

The algorithm implementation for what your operator does will go inside **computTile()** or **computeTiles()**.

```
public interface Operator {
      OperatorSpi getSpi();
      Product initialize(OperatorContext context);
      void computeTile(Tile targetTile, ProgressMonitor pm);
      void computeTileStack(Rectangle targetTileRectangle, ProgressMonitor pm);
      void dispose();
}
```

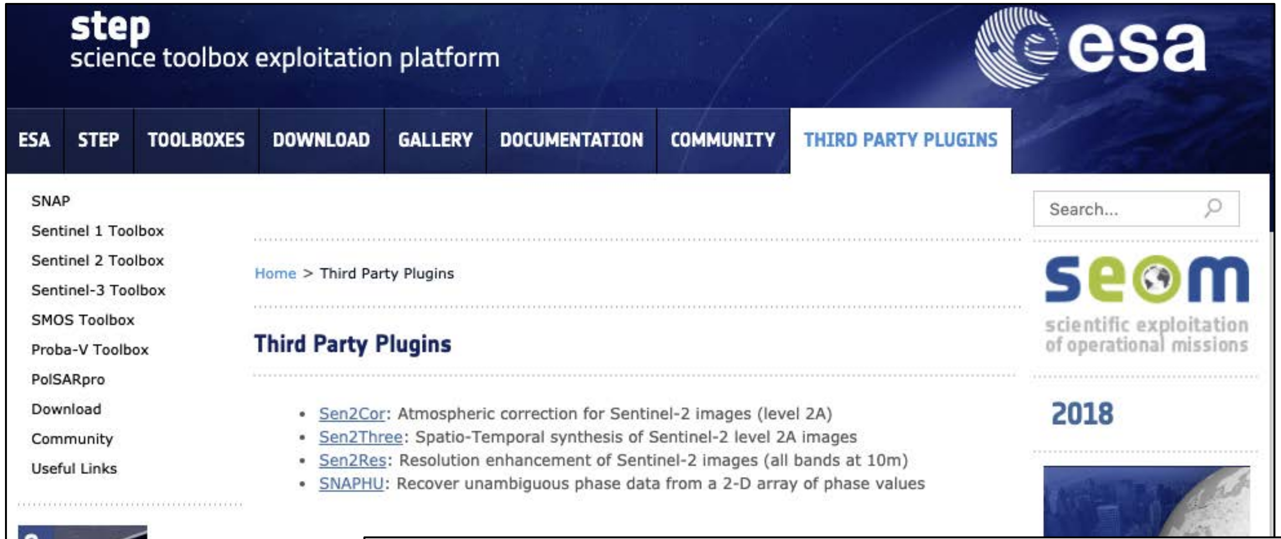
TERRADUE
Advancing Earth Science

What doesn't work: plugins

On the other hand, we see no value in providing plugins.

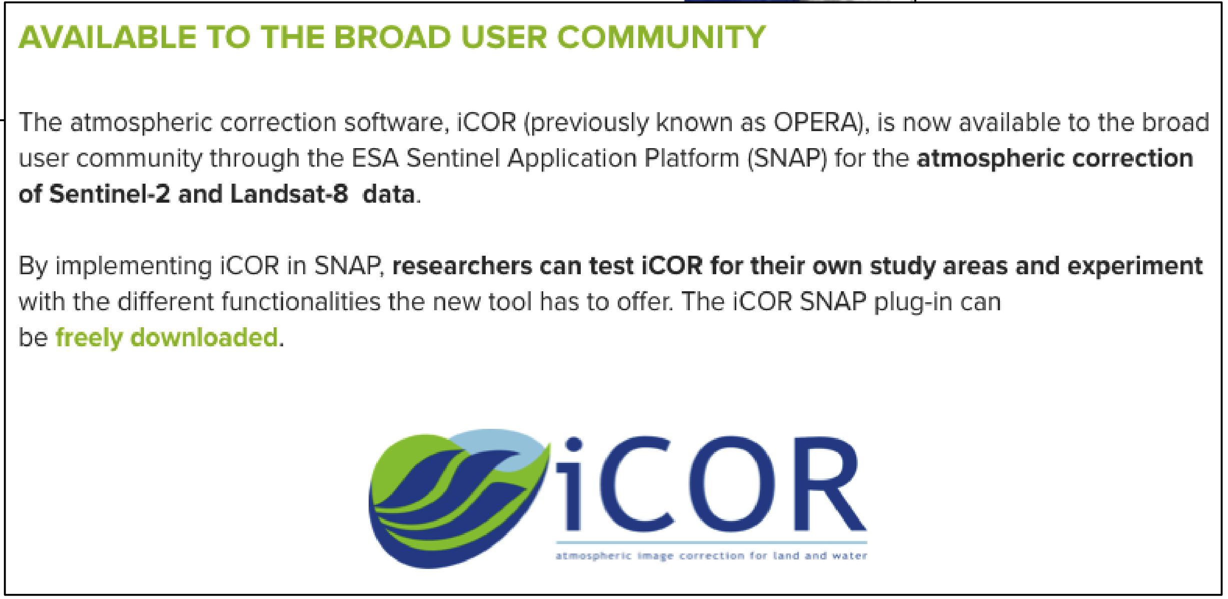
In this case, SNAP acts as a wrapper on top of CLI applications.

We go straight to the CLI.



Third Party Plugins

- Sen2Cor: Atmospheric correction for Sentinel-2 images (level 2A)
- Sen2Three: Spatio-Temporal synthesis of Sentinel-2 level 2A images
- Sen2Res: Resolution enhancement of Sentinel-2 images (all bands at 10m)
- SNAPHU: Recover unambiguous phase data from a 2-D array of phase values



AVAILABLE TO THE BROAD USER COMMUNITY

The atmospheric correction software, iCOR (previously known as OPERA), is now available to the broad user community through the ESA Sentinel Application Platform (SNAP) for the **atmospheric correction of Sentinel-2 and Landsat-8 data**.

By implementing iCOR in SNAP, **researchers can test iCOR for their own study areas and experiment** with the different functionalities the new tool has to offer. The iCOR SNAP plug-in can be **freely downloaded**.

Conclusion

Performances

- Processing performance are OK for us
- SNAP Graph developer learning curve could be better: several GOTCHAS here and there (e.g. Thermal Noise Removal)
- Split Graphs is a mandatory strategy

Extra SNAP feature would help

- Cloud Optimized Geotiff as an output format
- BEAM-DIMAP as an internal format
- ‘no data’ support

Does SNAP help you achieve everything that you expect it to?

- A single toolbox will never be able to gather all features and functions we need to build services
- We are used to cherry picking what works from toolboxes and libraries and avoid what doesn’t

Terradue

Advancing Earth Science

Thank you !