

agenda	2
Cbm_Outreach_DIAstrainingI_P1	3
Cbm_Outreach_DIAstrainingI_P2	51
Cbm_Outreach_DIAstrainingI_P3	56

## Webinar on DIAS for CbM Outreach - Session 1

Date: Friday, 23rd June 2021

### Agenda

09:30 - 09:45 Welcome and (very) short introduction into CbM+DIAS (Guido Lemoine, JRC)

09:45 - 10:15 The Big Picture: functional modules and why we need them? (Guido Lemoine, JRC)

10:15 - 11:00 The Backend: CARD processing and parcel extraction (Guido Lemoine, JRC)

11:00 - 11:15 Break

11:15 - 11:45 The Frontend: RESTful services and Jupyter Notebooks (Guido Lemoine, JRC)

11:45 - 12:15 The CbM code base structure, documentation and collaboration (Konstantinos Anastasakis, JRC)

12:15 - 12:30 Next steps and discussion (Rafal Zielinski, JRC)



# CbM on DIAS: a technical deep dive

On-line training for Outreach, 23 June 2021

*JRC D5 – GTCAP Team*

# Agenda

09:30 - 09:45	Welcome and (very) short introduction into CbM+DIAS
09:45 - 10:15	<b>The Big Picture:</b> functional modules and why we need them
10:15 - 11:00	<b>The Backend:</b> CARD processing and parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	<b>The Frontend:</b> RESTful services and Jupyter Notebooks
11:45 - 12:15	The <a href="#">cbm code base</a> structure, documentation and collaboration
12:15 - 12:30	Next steps and discussion

# Agenda

09:30 - 09:45	<b>Welcome and (very) short introduction into CbM+DIAS</b>
09:45 - 10:15	<b>The Big Picture:</b> functional modules and why we need them
10:15 - 11:00	<b>The Backend:</b> CARD processing and parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	<b>The Frontend:</b> RESTful services and Jupyter Notebooks
11:45 - 12:15	The <a href="#">cbm code base</a> structure, documentation and collaboration
12:15 - 12:30	Next steps and discussion

# Welcome

- The “deep dive” will provide details about the jrc-cbm implementation on DIAS.
- Short introduction of the CbM context, current status
- Please use the chat for questions during the sessions. Audio & Video during Q&A.
- Remember to switch off video (save bandwidth) and mute audio, when not speaking.

# Audience

- For non-programmers:
  - Cloud principles
  - DIAS and Sentinel data
  - The basis for “Big Data Analytics”, ready for use
- For programmer/developers:
  - A modular system of components
  - python, SQL + linux bash
  - Dockerization + parallel processing
  - How to integrate analytics, image processing
- For all:
  - Fully functional scalable European solution
  - Fit-for-purpose for current and future CbM needs



# Agenda

09:30 - 09:45	Welcome and (very) short introduction into CbM+DIAS
09:45 - 10:15	<b>The Big Picture: functional modules and why we need them</b>
10:15 - 11:00	<b>The Backend:</b> CARD processing and parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	<b>The Frontend:</b> RESTful services and Jupyter Notebooks
11:45 - 12:15	The <a href="#">cbm code base</a> structure, documentation and collaboration
12:15 - 12:30	Next steps and discussion

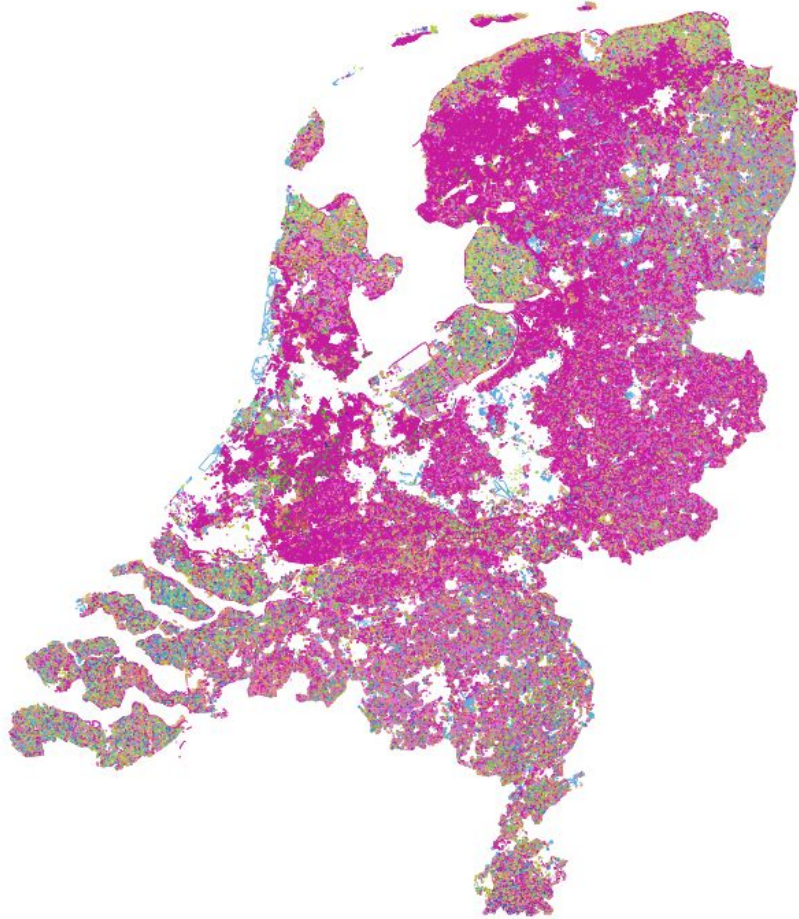


# Context

- Checks by Monitoring introduces **continuous use of Sentinel data streams** for 100% of the Member State territory.
- Copernicus DIAS advantages:
  - Access to a **consistent, complete** Sentinel data archive (push, not pull)
  - Provision of on-demand standard CARD processing
  - Access to compute resources that can (temporarily) scale to needs
  - Based on **open industry standards**, core open source modules
- Facilitates the needs for **TAILORED** automated processing.
- Potential for shared methodology

# Onboarding status

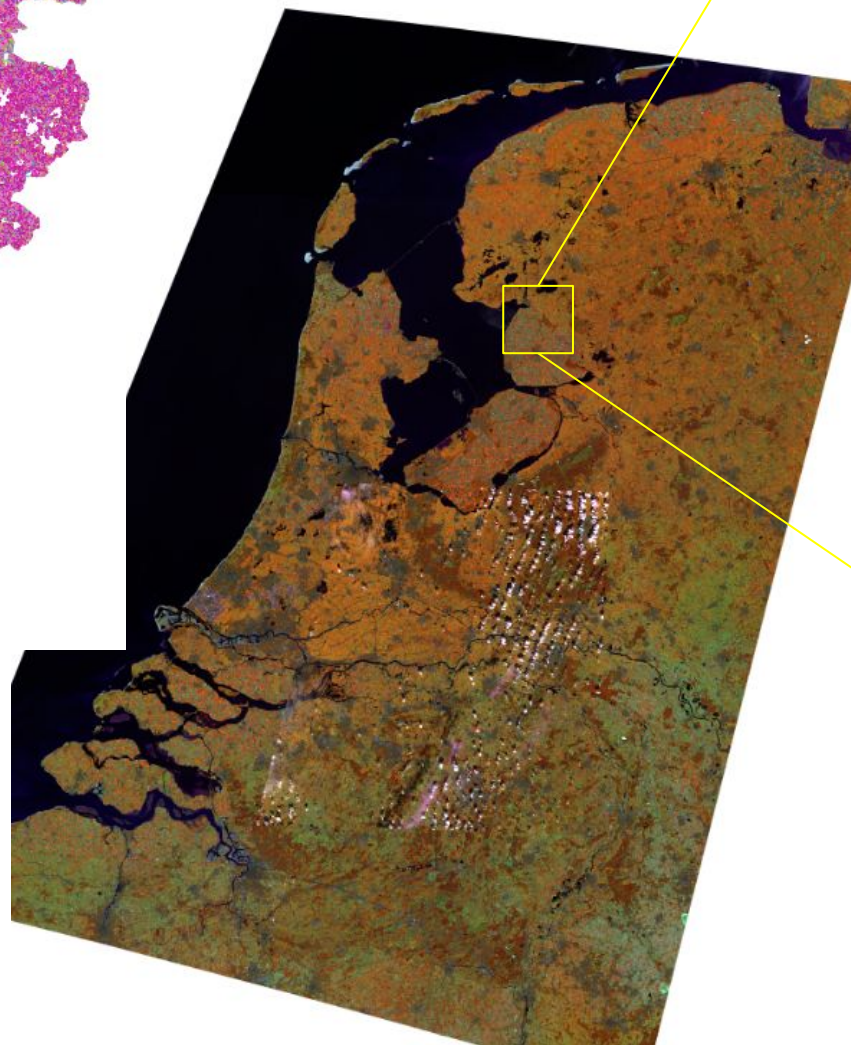
- DIAS for CbM introduced in 2019. DG AGRI + DG DEFIS funded
- 4x ESA managed DIAS (CREODIAS, MUNDI, SOBLOO, ONDA)
- EUMETSAT managed WEkEO (which runs on CloudFerro)
- Categories of Member State Paying Agencies (PA)
  - **Group 1:** Early adopters (2019+): BE-VL, ES, DK, IT
  - **Group 2:** Specific schemes (2020+): MT, BE-WA, FR, 4 (6) DE Länder, FI (IE, PT)
  - **Group 3:** Experimental (2020+, WEkEO): BG, RO, LV
  - **Outreach:** AT, HR, EE, SE, HU, NL, PL, CZ, +DE
- 2021+ arrangement under design, depends on DIAS future, CAP directions.



n BRP 2019 @ pdok.nl

n ~ 770,000 parcel /yr  
m ~ 4000 granules/yr  
p ~ 1200 scenes/yr

m Sentinel-2 @ DIAS  
p Sentinel-1 @ DIAS



n-m, n-p\*2 spatial time series  
for Sentinel-1, -2 CARD  
for b bands (b=14 (S2), 2 (S1))  
x 100 for whole EU

# jrc-cbm versus SEN4CAP

	<b>jrc-cbm</b>	<b>SEN4CAP</b>
<b>API</b>	python, SQL	Java, C++, python, JavaScript, R
<b>CARD processing</b>	DIAS PaaS or internal	Internal
<b>Time series</b>	PostgreSQL/postgis	Arrow (since version 2)
<b>Resource needs</b>	1 medium VM, additional on demand and need basis (cheap)	1 very large VM, continuously (expensive)
<b>Modularity</b>	Role based backend + frontend, scalable	Single monolithic “washing machine”
<b>Graphical interface</b>	Jupyter notebooks	Dashboards, Jupyter notebooks
<b>Open source</b>	Fully, <b>BSD Clause 3</b>	Not all modules, <b>GPL 3</b>
<b>Development choices</b>	PA community of practice. Cloud at the core.	Space data push, academia. Cloud as an afterthought.

- We recommend jrc-cbm as a **TAILORED** solution

# Why you do **NOT** need a crop map

- jrc-cbm versus SEN4CAP, *continued* (Ref. final SEN4CAP meeting 4 Mar 2021).
- The PA has an actual crop map after GSAA update (~ in May!)
  - Which is 95+% accurate, for the full range of crop classes
  - One-to-one object based, if GSAA is **FOI compliant**
  - No satellite data based system will (ever?) achieve that
  - It is not in the PA interest to resolve accuracy issues in crop classifications
  - Depending on schemes, (c)omission errors may not be relevant at all
- in jrc-cbm “crop” is a marker derived from ML applied to time series

# The jrc-cbm “marker” concept

- How to generate the minimum required **INFORMATION** to confirm/reject compliance with the declared **PRACTICE**
- Sentinels produce dense, **hybrid sensor data**
- Time-series of Sentinel sensor data may (not) pick up the signature of the events and patterns that relate to the practice (**markers**)
- Understanding how Sentinel markers relate to practices requires knowledge of the **physics** behind sensor data (a translation step)
- Optimization through automation + reduction to handle the data load
- Feeds into traffic light system for follow up (e.g. the need for complementary information).

# The jrc-cbm “marker” concept, *continued*

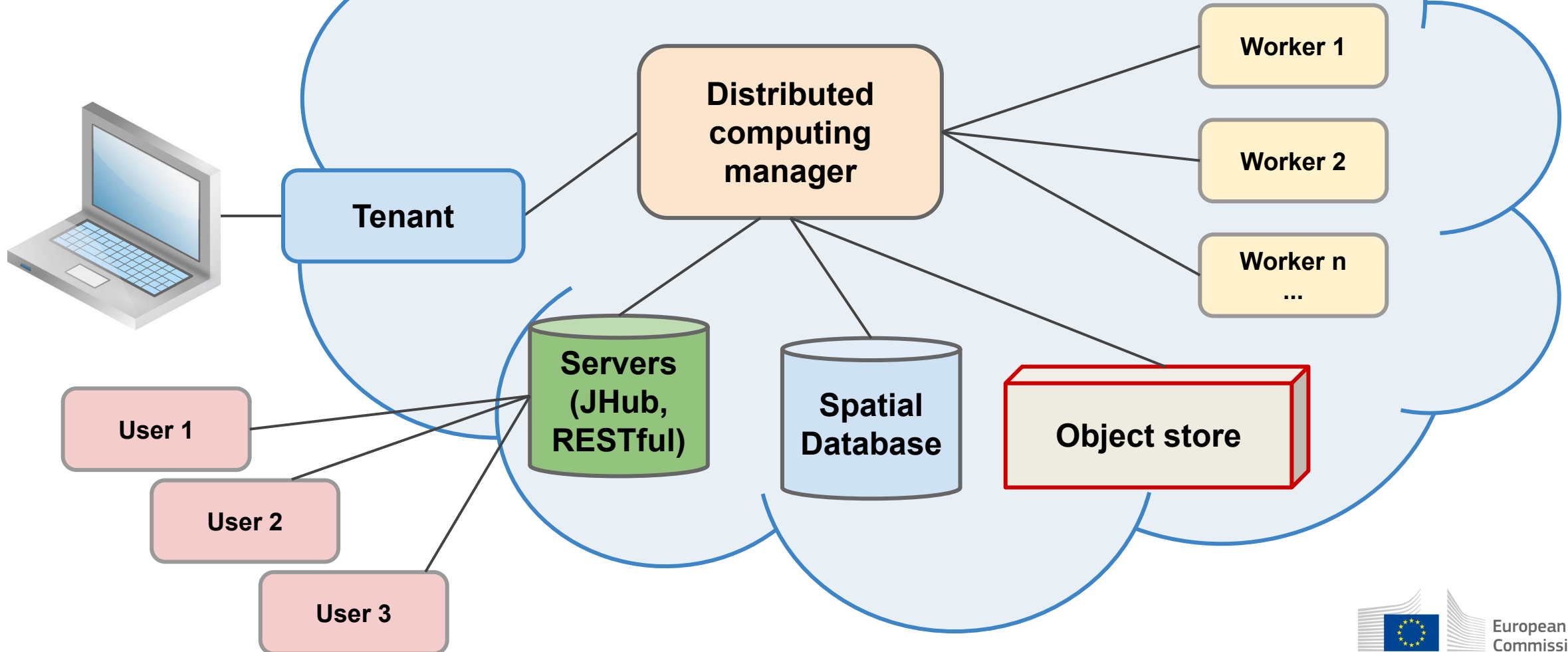
- **INFORMATION** in **hybrid sensor data** is best derived from **signals** that are not (too much) correlated
- i.e. combinations of **DIFFERENT** spectral bands of S2, dual polarized backscattering coefficients and coherence of S1
- **Not** “biophysical parameters” (LAI, fAPAR, fCover), because these are transformations of 2 or 3 S2 bands (therefore redundant, expensive)
- Be careful with “smoothing” or “averaging” (misnamed as “practice markers” in SEN4CAP), because you may just eliminate the event you are looking for.
- Other ancillary data may be required to explain signal artefacts (DEM, weather parameters, etc).

# jrc-cbm roles and use patterns

- jrc-cbm considers roles. Not all roles need to work with all modules
- **ICT expert** maintains IaaS and the required server components, e.g. monitoring CARD production, run extractions
- This BACKEND role can be managed by one person per MS (or even DIAS)
- **Data analytics expert** programs and runs core and markers analytics (e.g. extraction, machine learning). Internal and external to the PA.
- **Data consumers** extract, cross-check, verify, combine, decide and report
- These FRONTEND roles make use of RESTful API, Jupyter Notebooks, etc.
- The combination of BACKEND and FRONTEND ensures full traceability.

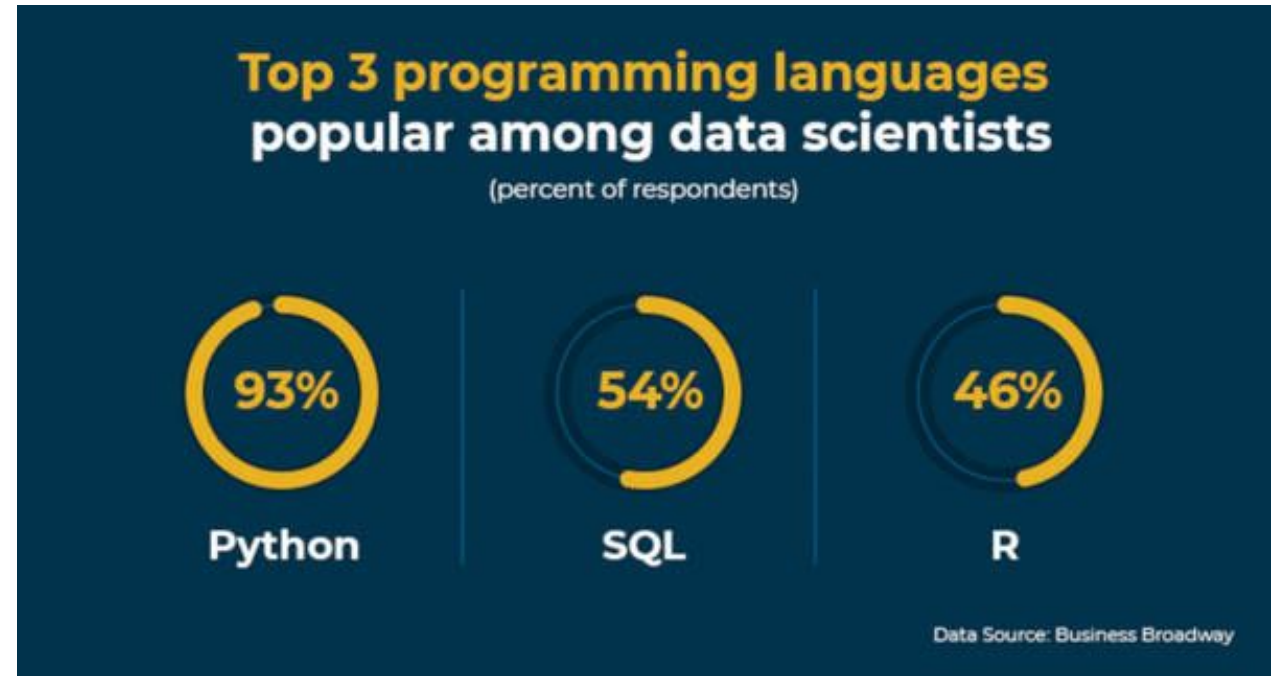
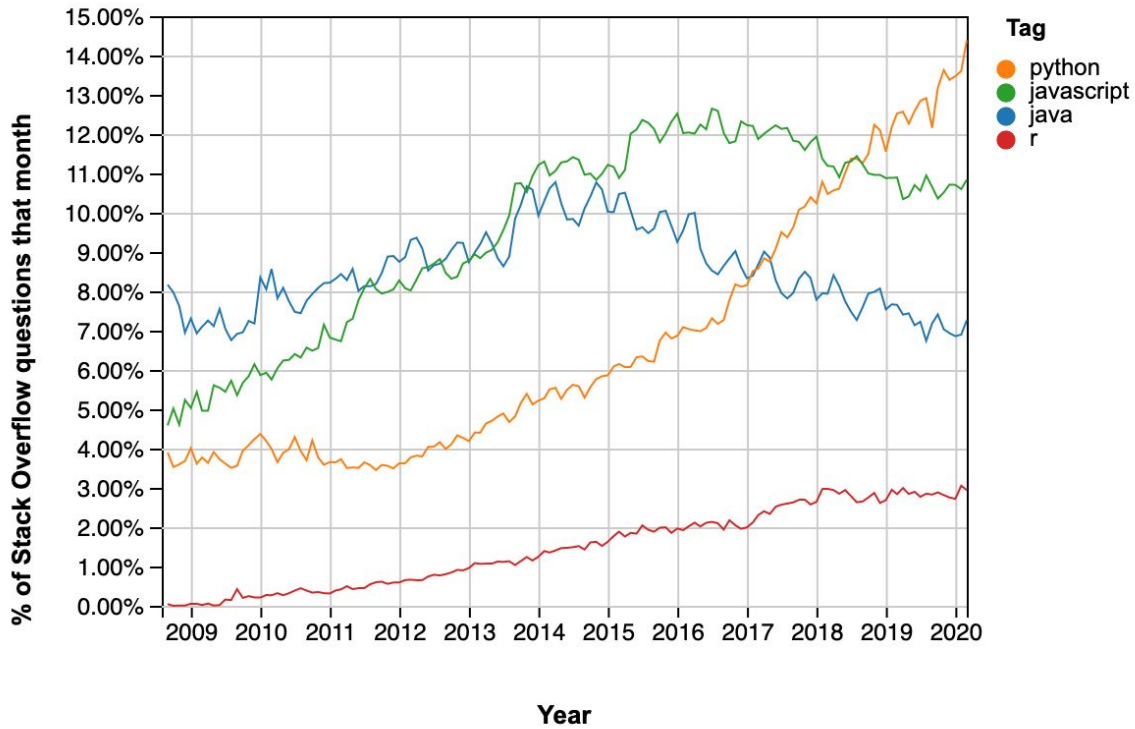


# Copernicus DIAS IaaS



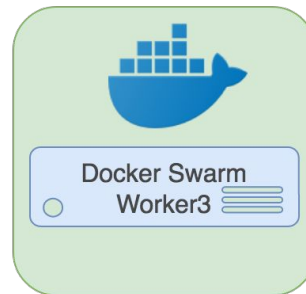
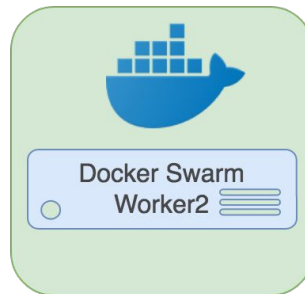
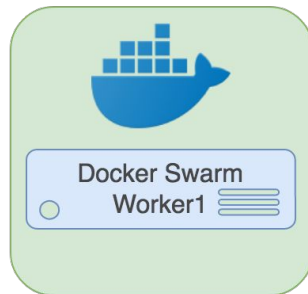
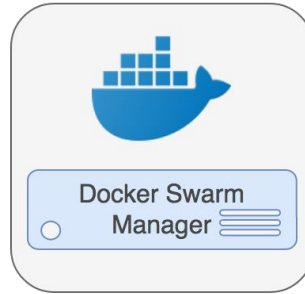
# Technical choices

- jrc-cbm is designed on a cloud centric basis (but can also run stand-alone)
- all programming in **python**, mostly as **syntactic glue**
- using mature modules (see cbm code presentation later)
- **PostgreSQL/Postgis** for (spatial) data management on backend.
- Linux (Ubuntu) bash scripting for orchestration, parsing, conversion (gdal)
- **This combination is sufficient to manage the complete cbm process.**
- And to further expand with emerging solutions in parallel processing, hardware specific processing (GPUs), machine and deep learning, etc.
- All maintained and documented on [github.com/ec-jrc/cbm](https://github.com/ec-jrc/cbm)
- Licensed under BSD Clause 3 (facilitates maximum re-use)



We knew it, but now we know why we knew it...

# Open Source software components used



# Agenda

09:30 - 09:45	Welcome and (very) short introduction into CbM+DIAS
09:45 - 10:15	<b>The Big Picture:</b> functional modules and why we need them
10:15 - 11:00	<b>The Backend:</b> CARD processing and parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	<b>The Frontend:</b> RESTful services and Jupyter Notebooks
11:45 - 12:15	The <a href="#">cbm code base</a> structure, documentation and collaboration
12:15 - 12:30	Next steps and discussion

# The jrc-cbm Backend

- jrc-cbm backend main function is:
  - to generate Application Ready Data (**ARD**), if not already in DIAS archive;
  - to reduce the spatio-temporal image stacks of ARD to parcel time series.
- ARD generation of S-2 is **not** needed, if you can live with sen2cor L2A
- ARD generation of S-1 is needed, because the Copernicus program did not foresee it (an anomaly that may be resolved in the future)
- DIAS instances offer a Processing as a Service (**PaaS**) solution for ARD
- ARD is stored in the S3 object store of the DIAS, in public or private buckets
- There is no real reason why parcel extraction could not be offered as a PaaS
- But currently, it's not, so you need to do this on your DIAS resources

# Sentinel-1 ARD processing

- Sentinel-1 ARD processing is needed to generate:
  - calibrated geocoded backscattering coefficients from Level 1 GRD (CARD-BS);
  - geocoded coherence from Level 1 SLC S1A and S1B pairs (CARD-COH6).
- CARD-BS has 10 m, CARD-COH6 has 20 m pixel spacing. Auto-UTM projected (Cloud Optimized) GeoTIFF or BEAM DIMAP format.
- CREODIAS uses the Copernicus DEM.
- DIAS instances produce CARD with JRC tested recipes for SNAP s1tbx
- Involves the transfer of large files (SLC ~ 7 GB per image) and RAM hungry processing (128 GB of RAM).
- Procedure is not yet part of cbm documentation (available on request)

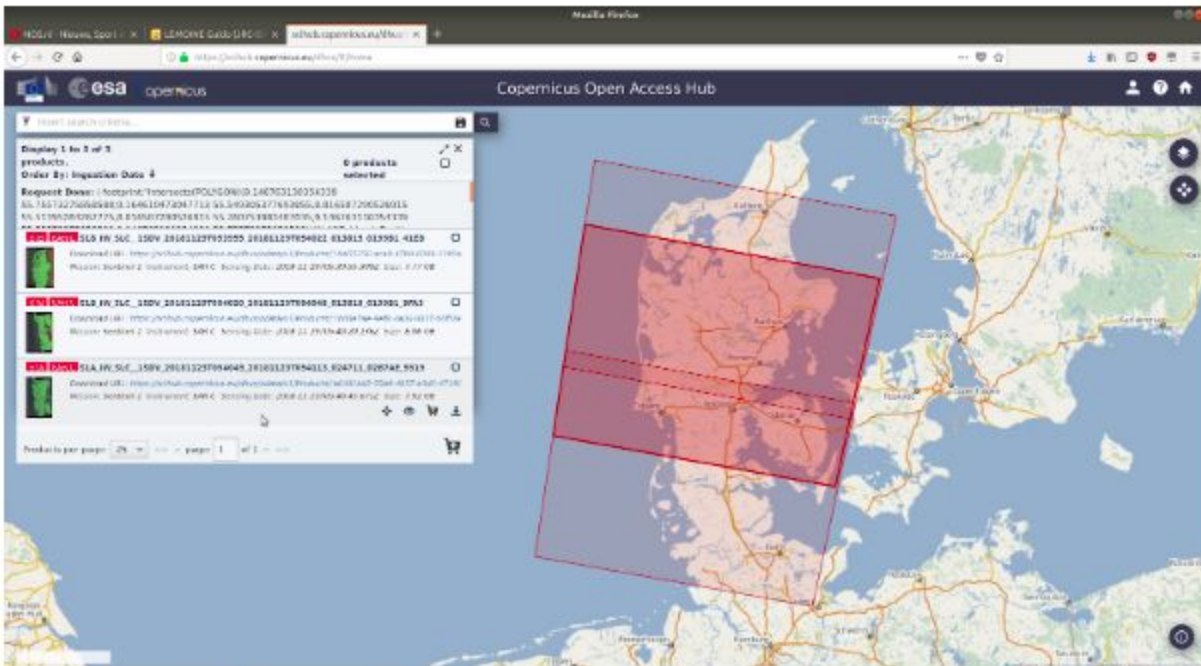


Figure 1. Sentinel-1 coverage for a possible coherence product over central Denmark. The Sentinel-1A scene of 23 November 2018 (dark red) needs to be matched with 2 consecutive Sentinel-1B scenes of 29 November 2018 (light red).

- S1-CARD-COH6 generation requires combining one S1A with 2 consecutive S1B Level 1 SLC frames (or vice versa) because frame boundaries are not synchronized.
- Thus, each coherence scene requires roughly **21 GB** SLC data.
- Each SLC scene consists of 3 subswaths that are processed separately and then merged.
- We use **python** scripts to automatically select the combinations and set up SNAP **gpt** to run the relevant XML graphs.

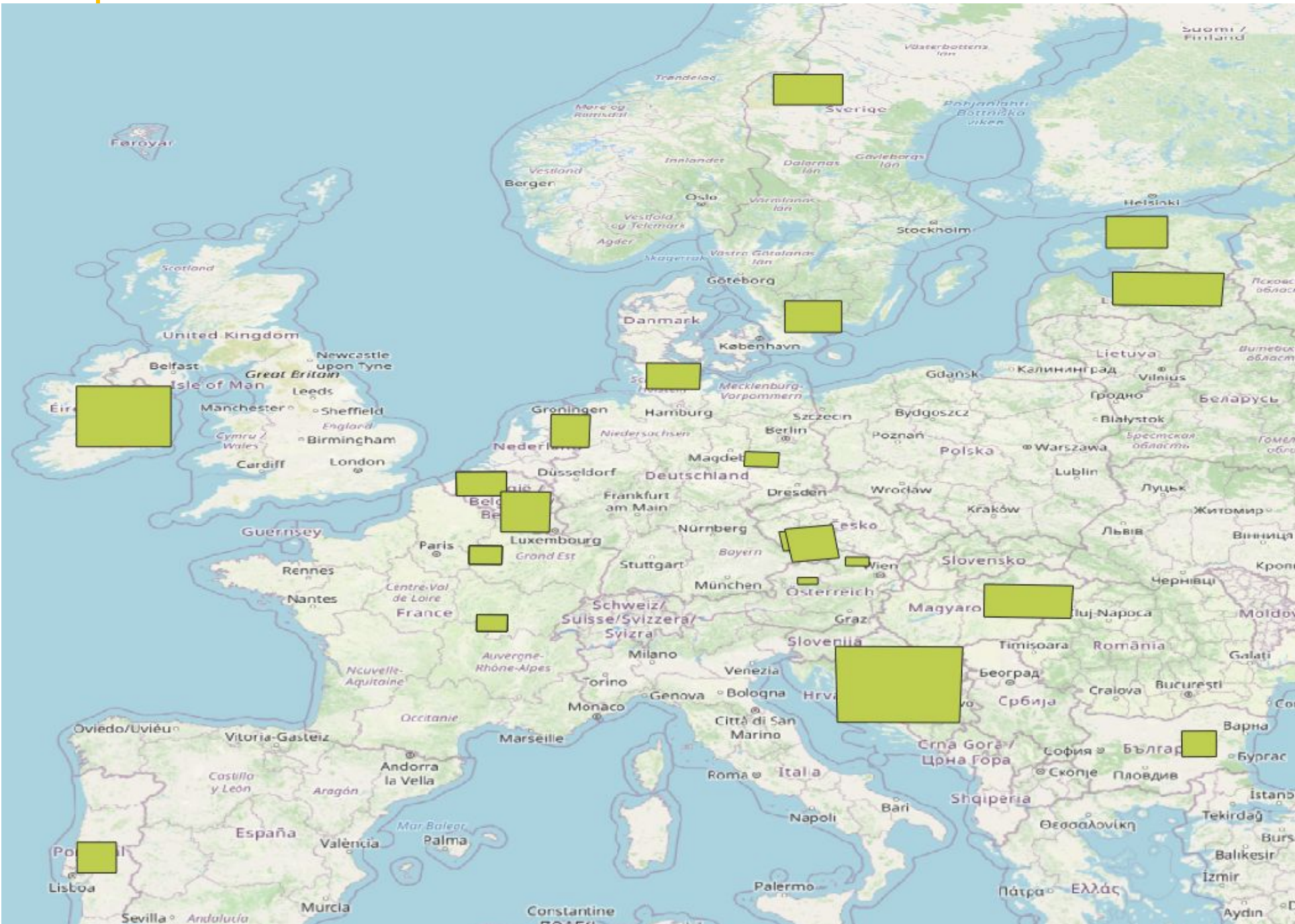


```
# Generate subswath coherence for the 3 subswaths of the first
pair
/home/user/snap6/bin/gpt
/home/user/S1/TOPSAR_Coherence_Single_Swath.xml
-Pinfile_master=/mnt/data/S1/NL/S1A_IW_SLC_1SDV__20181123T054045_2
0181123T054112_024711
_02B7AE_5519.zip
-Pinfile_slave=/mnt/data/S1/NL/S1B_IW_SLC_1SDV__20181129T053955_20
181129T054022_013815_0199B1_41ED.zip -Psub=1
-Poutfile=/mnt/data/scratch/S1A_
S1B_20181123T054045_20181129T053955_coh_IW1.dim
/home/user/snap6/bin/gpt
/home/user/S1/TOPSAR_Coherence_Single_Swath.xml
-Pinfile_master=/mnt/data/S1/NL/S1A_IW_SLC_1SDV__20181123T054045_2
0181123T054112_024711
_02B7AE_5519.zip
-Pinfile_slave=/mnt/data/S1/NL/S1B_IW_SLC_1SDV__20181129T053955_20
181129T054022_013815_0199B1_41ED.zip -Psub=2
-Poutfile=/mnt/data/scratch/S1A_
S1B_20181123T054045_20181129T053955_coh_IW2.dim
```



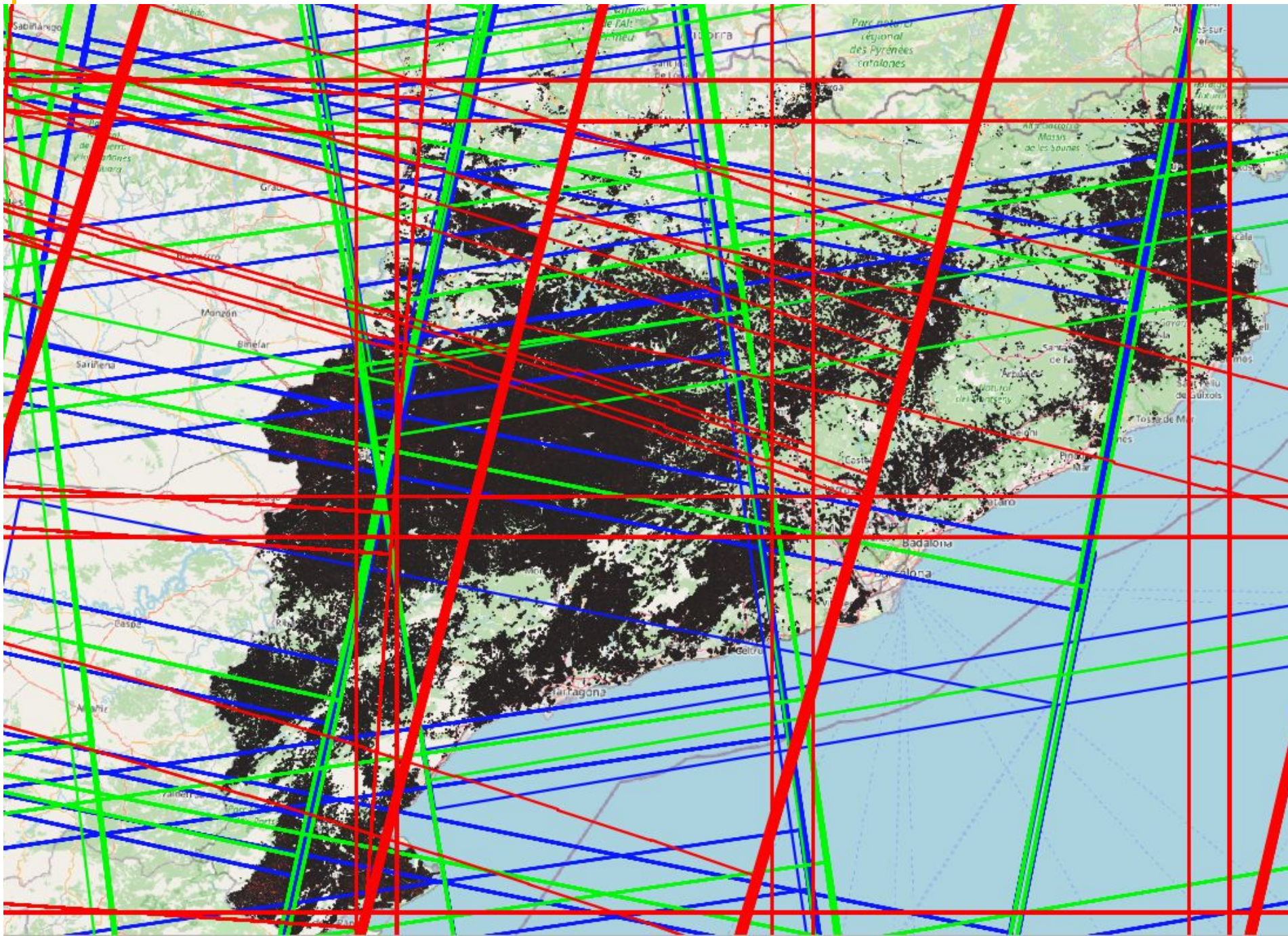
# Sentinel-1 ARD processing, *continued*

- CARD processing can be done for a large seasonal volume or continuously on a daily basis
- CARD processing and storage is charged to your DIAS account (unit prices).
- Public S3 storage makes it available to other (CREO)DIAS users.
- You can decide to run SNAP s1tbx yourself. We do **NOT** recommend changing the recipes.
- SNAP s1tbx is open source, but not particularly efficient code (Java)
- [GPU processing](#) of SLC (COH6) allows for a speed up of a factor 50+
- We are working on a procedure to introduce Radiometric Terrain Correction.



## Outreach backend

- Select S2 CARD-2A
- Generate S1 CARD-BS
- Generate S1-CARD-COH6
- Extract parcel statistics



**S2 CARD-2A**  
**S1 CARD-BS**  
**S1-CARD-COH6**  
**Catalunya 2018**

# Reduction to parcel extracts

- The Sentinel ARD collections are a temporal patchwork of image frames
- The PA's parcel set is (usually) for a contiguous geographical area
- Parcel extraction needs to combine
  - P parcel features (P = several 100K per PA)
  - N S2 CARD-L2A, M S1 CARD-BS and ~ M S1 CARD-COH6 image frames
  - N and M in the order of 1000s (N ~ 2.5xM)
  - Each N has 3 or more bands (e.g. B8, B4, SCL), M has 2 (VV and VH)
  - Expandable to Q different indices, SCL as histograms
- Leads to several 100 M records of (parcel\_id, obs\_id, band, statistics).
- Order of 10 GB per (parcel\_id, band) for 1 Million parcels/year in PostgreSQL

# Why do we need PostgreSQL?

- State-of-the-art open source object-relational database system
- [SQL compliant](#) with PostgreSQL specific extensions
- Role based privileges (e.g. db manager, db developer, db viewer)
- postgis extension for spatial objects (**vector** and **raster**), queries, functions
- PostgreSQL server handles network client connections (usually port 5432)



- jrc-cbm uses mostly standard (spatial) SQL
- primary, foreign keys with index/cluster-ing
- logic handled in python (psycopg2)
- parcel data sets, CARD metadata and state management, time series
- **Short psql demo**

# Reduction to parcel extracts, *continued*

- Extraction can be set up as an automated process which:
  - finds the oldest image that is not yet processed (e.g. inserted from the catalogue)
  - transfers the image bands from the S3 store onto local disk (this is fastest)
  - queries the database for all parcels within the image bounds
  - extracts the statistics ( $\mu$ ,  $\sigma$ , min, max, p25, p50, p75) for the bands of the image
  - stores the results in the time series database tables
  - clears the local disk
- Recently refactored to use rasterized parcels and Numba acceleration
- ~ 9 times faster than last year (or, can be run on 1/9th of the resources!)
- Our current World 🌍 record is 3 M parcels x 2 S1 CARD-BS bands for 14 months (~1100 images) in 24 hrs on a single machine (8x CPU/32 GB RAM)

# Parallel processing on DIAS

- A key asset of DIAS is the possibility to process across multiple VMs
- Many jrc-cbm tasks are “embarrassingly parallel” (e.g. extraction)
- As a first step, “marshall the resources” using openstack (Horizon)
- In a second step, “orchestrate the resources” to execute parallel tasks



- **Short openstack cli demo, ssh access**
- [Dockerize](#) the dependencies
- Orchestrate with docker swarm
- Alternative orchestration: k8s, [dask](#) (!)
- Other uses in jrc-cbm: chip extraction
- Future use: complex raster processing, ML

# Other backend modules

- The frontend needs to access the backend data sets, which requires:
  - direct access to database tables for developers, e.g. in Jupyter Notebooks
  - abstracted RESTful access to database tables for users
  - abstracted RESTful access to sub-selections of S3 stored CARD data
  - abstracted RESTful access to advanced server-side processing routines
- RESTful requires a server, JupyterHub is already a DIAS service
- Both may use orchestrated multiple VM for scaling
- jrc-cbm components only address moderate multi-user request loads
- You always have the option to transfer the database tables to a local host!





# Backend take home messages

- The backend is the core jrc-cbm component for server-side requirements
- The backend does the processing heavy-lifting to provide consistent access to CARD data and their parcel reductions
- It makes sense to have **one single maintainer** of the backend per **MS!**
- PA organized in separate databases (configuration management)
- Backend functionalities and performance focuses on common needs
- Backend development may be impacted by Copernicus programme decisions (e.g. ARD production) and adoption of novel approaches (k8s, dask, GPU)
- Frontend developments (e.g. analytics) may be integrated server-side if of generic interest to many users.

# Agenda

09:30 - 09:45	Welcome and (very) short introduction into CbM+DIAS
09:45 - 10:15	<b>The Big Picture:</b> functional modules and why we need them
10:15 - 11:00	<b>The Backend:</b> CARD processing and parcel extraction
11:00 - 11:15	<b>Break</b>
11:15 - 11:45	<b>The Frontend:</b> RESTful services and Jupyter Notebooks
11:45 - 12:15	The <a href="#">cbm code base</a> structure, documentation and collaboration
12:15 - 12:30	Next steps and discussion

# Agenda

09:30 - 09:45	Welcome and (very) short introduction into CbM+DIAS
09:45 - 10:15	<b>The Big Picture:</b> functional modules and why we need them
10:15 - 11:00	<b>The Backend:</b> CARD processing and parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	<b>The Frontend:</b> RESTful services and Jupyter Notebooks
11:45 - 12:15	The <a href="#">cbm code base</a> structure, documentation and collaboration
12:15 - 12:30	Next steps and discussion

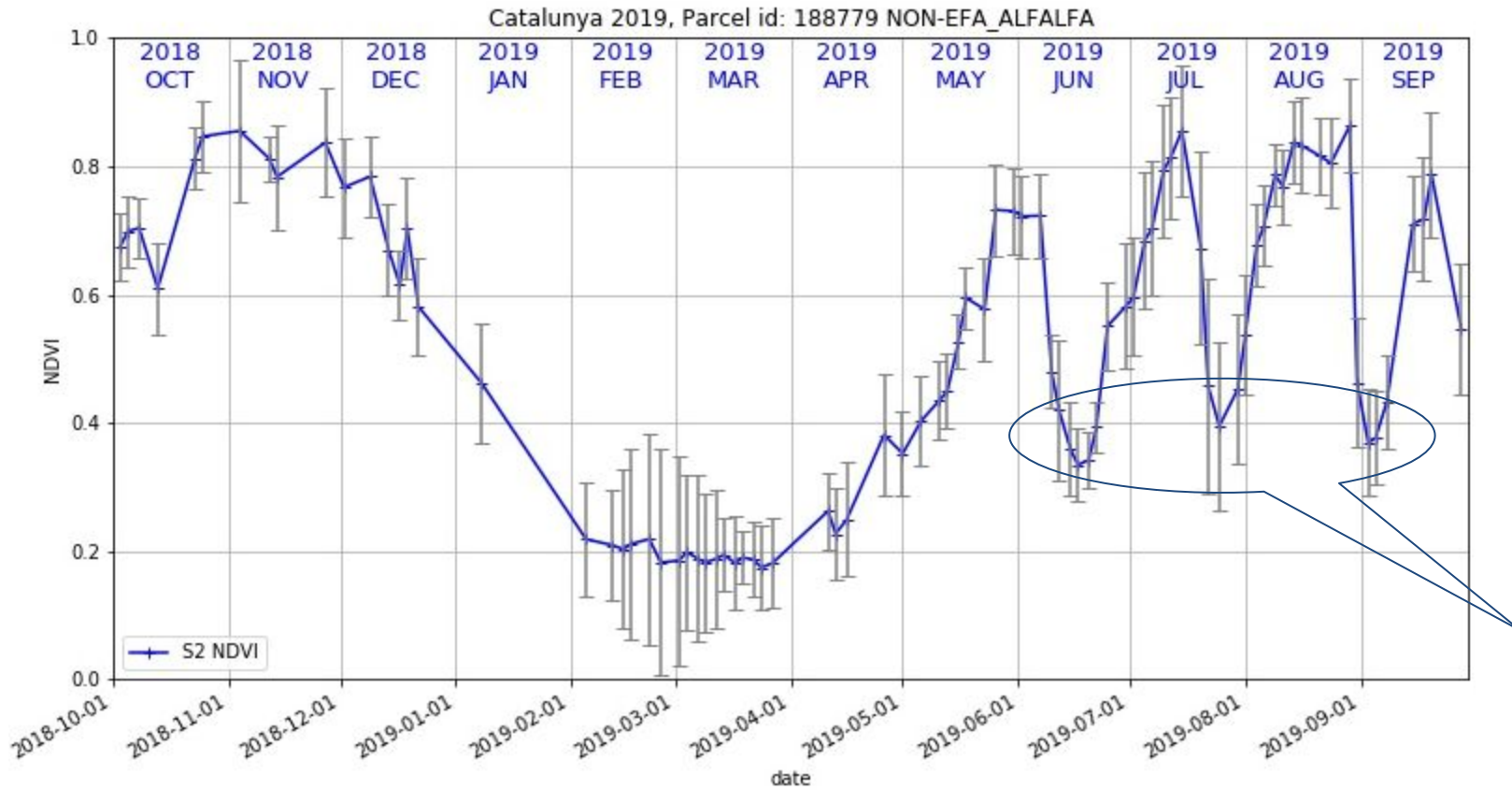
# The jrc-cbm Frontend

- jrc-cbm frontend main function is:
  - to integrate backend results into the PA workflow;
  - to support analytics, development
  - to design and apply marker analysis for decision support
  - to provide access to relevant ancillary reference data
  - to ensure full reproducibility of decisions
  - to support reporting tasks
- The frontend allows the build out of augmented application logic
- This may require multiple backend callbacks for refinement
- Some mature frontend functionality may be integrated as backend logic

# Simple Frontend use cases

- The simplest use case is just visualizing the time series
  - and manually mark the temporal artefact of interest
  - this can be “augmented” with more sophisticated calendar views
  - and include combinations of S1 and S2 time series
- 
- Use visualization to show the time series of an outlier vs it's peers
  - idem, but based on some quantified difference measures (e.g. min distance)
  - this can be augmented by pre-calculating min distance for each parcel and it's N nearest peers

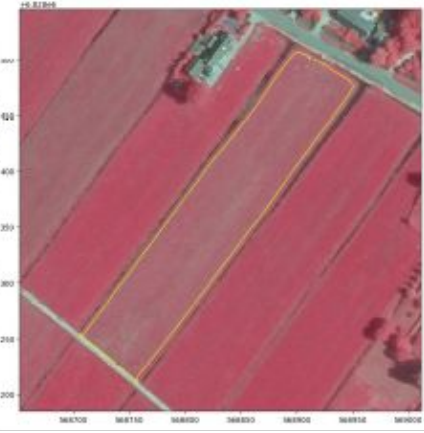
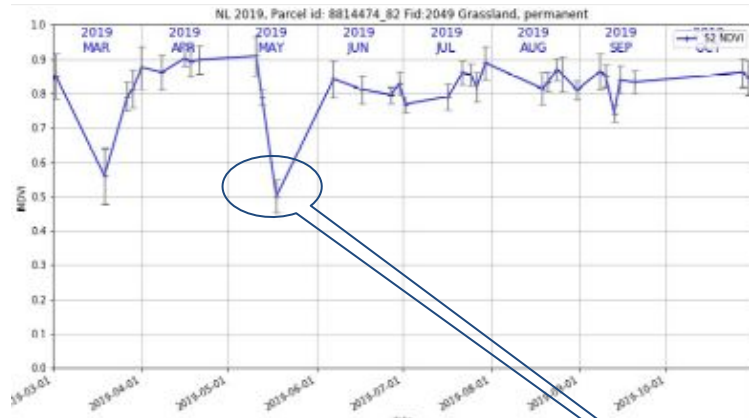
# S2 temporal profile



Pseudo-code:

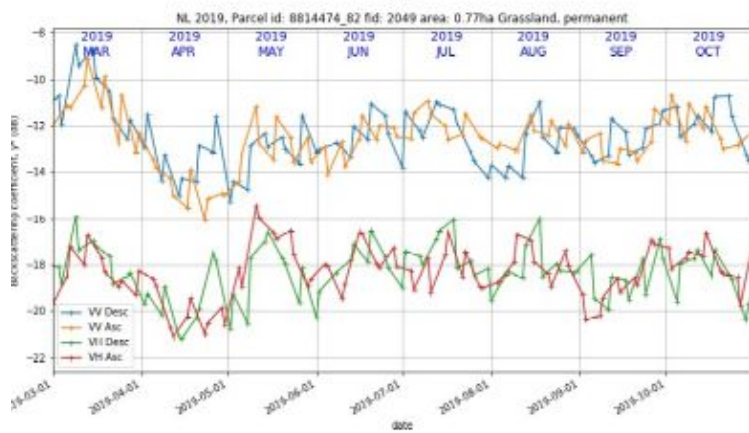
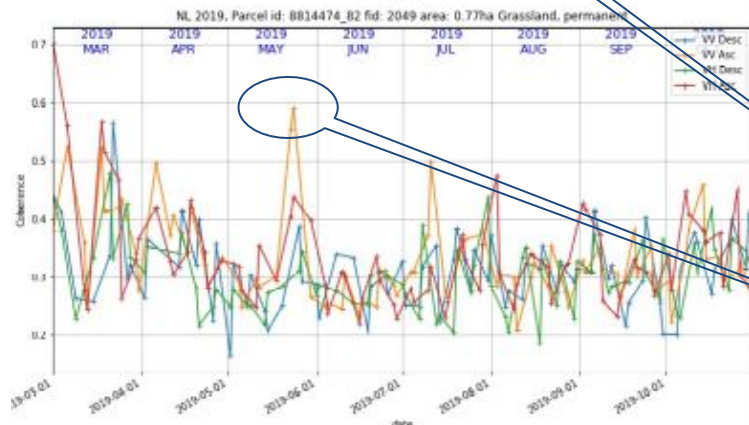
```
get B4, B8 from RESTful
calculate NDVI
plot
```

three mowing events and other telltale signs that this is indeed (irrigated) alfalfa

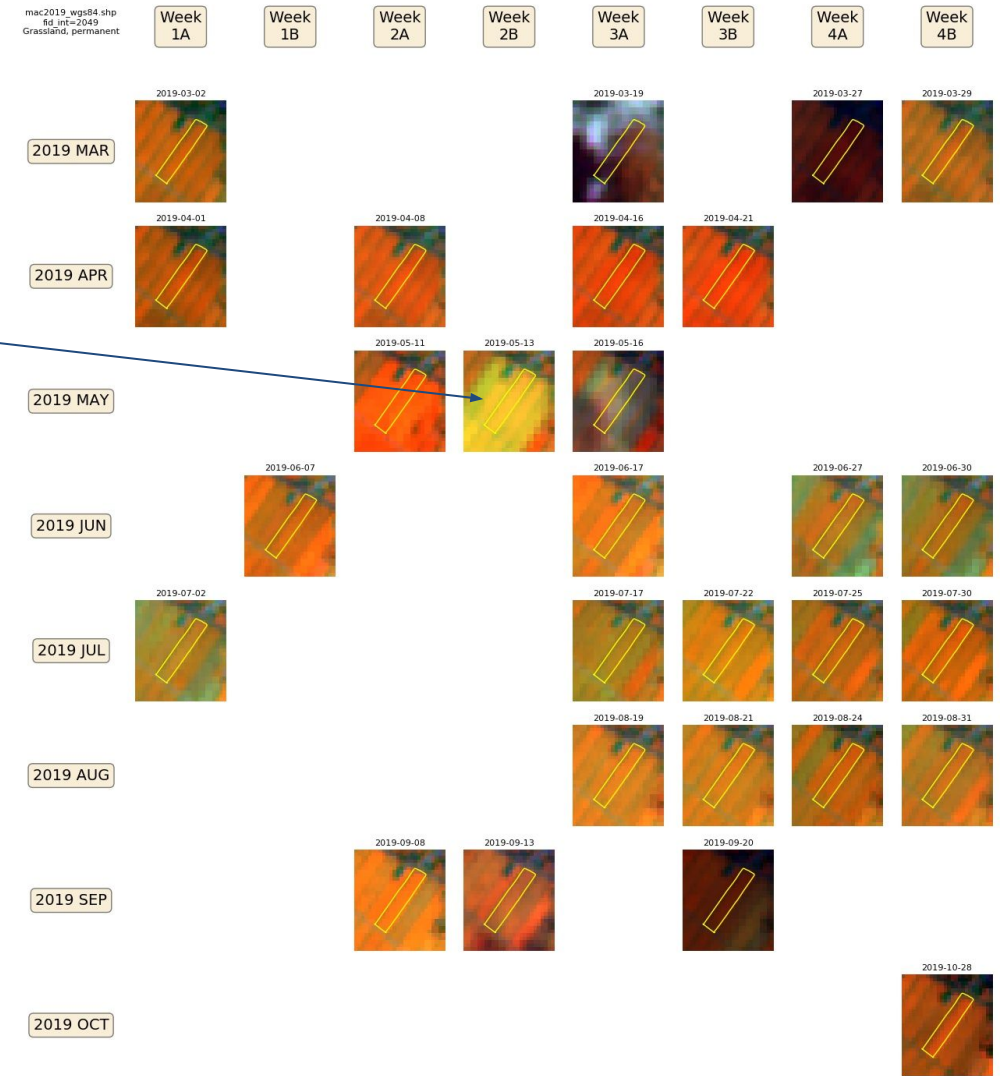
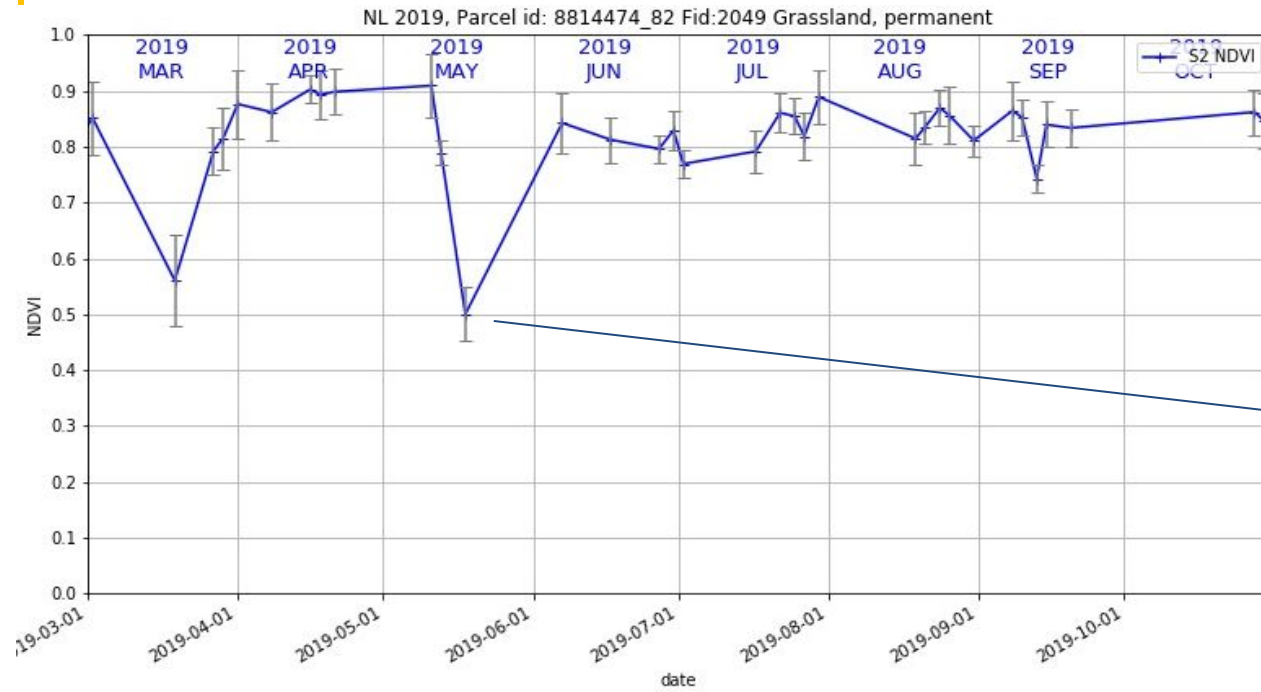


Pseudo-code:

- get B4, B8 from RESTful
- calculate NDVI
- plot
- get S1 C6 VV, VH from RESTful
- plot
- get S1 BS VV, VH from RESTful
- plot
- getBackground from RESTful
- plot
- get parcel from RESTful
- plot



simultaneous drop in NDVI and jump in coherence mark mowing event. Other behavior is typical for permanent grassland.



Spatial expansion of time series view  
(but essentially the same information)

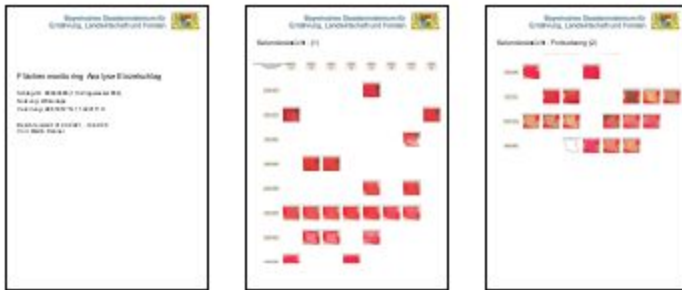
Pseudo-code:

```

get B4, B8 from RESTful
calculate NDVI
plot
get CalendarView (uses RESTful)

```





1

2

3



4

5

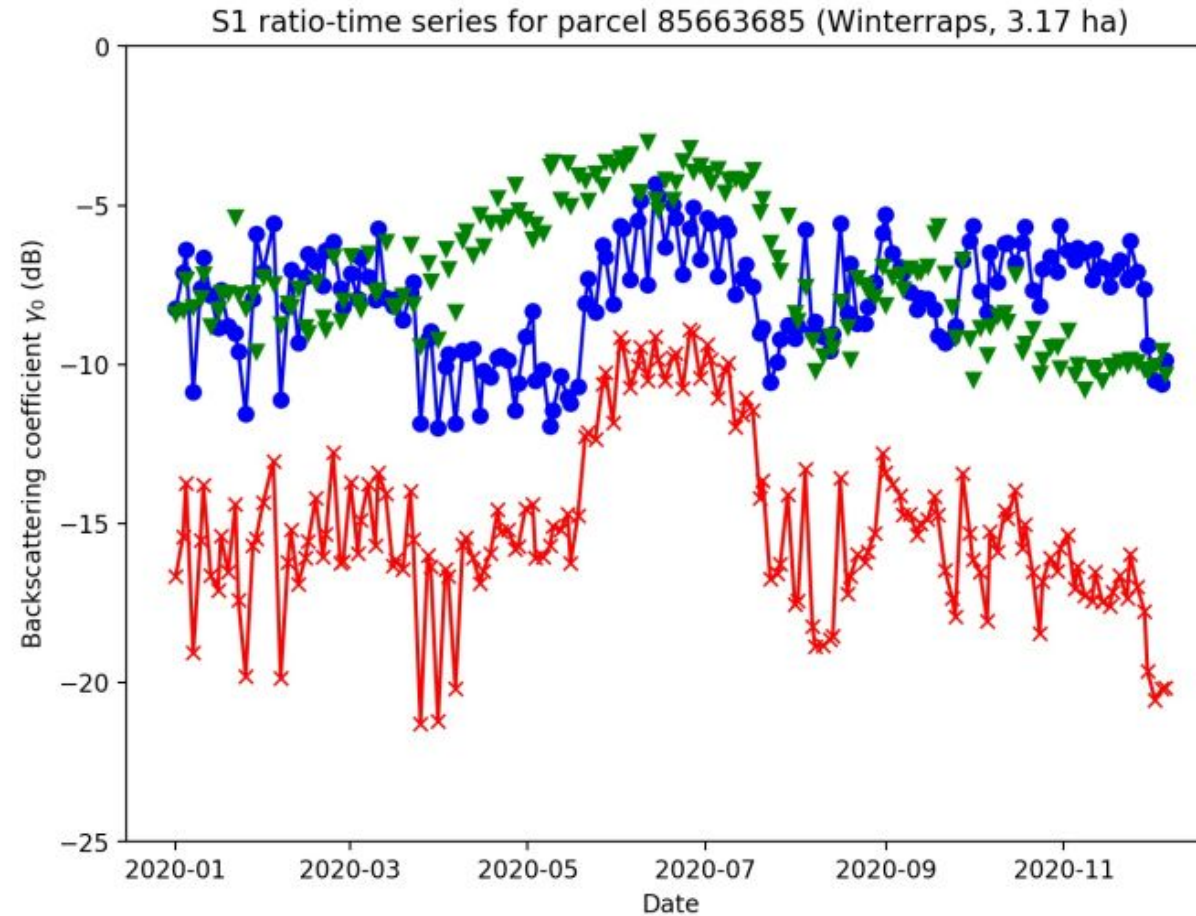
6

Report generation (Bayern PA!)

Pseudo-code:

```
get timeseries from database
get CalendarView (from WMS)
plot to multipage PDF
```

## Zeitreihen Plots - Sentinel-1: Backscatter

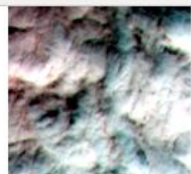


# jrc-cbm RESTful services

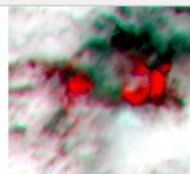
- Run in a Flash server
- Basic information retrieval (**parcelByLocation**)
- Fast parcel time series statistics (**parcelTimeSeries**, **parcelPeers**)
- (Slower) image chip selection, for visualization (**chipByLocation**, **backgroundByLocation** (Google, Bing and orthos via **WMTS**))
- idem, but full resolution GeoTIFFs (**rawChipByLocation**, **rawChipsBatch**, **rawS1ChipsBatch**). Uses multiple VMs on the backend for retrieval.
- Scripts (or Notebooks) “consume” RESTful services via python **requests**

# A slightly more elaborate Frontend use case

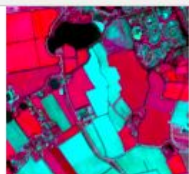
- Check FOI heterogeneity of arable crops:
  - access to S2 time series for arable crop parcels [of class X, of size > 0.3 ha, etc.];
  - use parcels statistics in the Bn band to test if a heterogeneity indicator can be derived;
  - (consider access to current and recent orthophotos, previous GSAA declaration)
  - set thresholds and define constraints (e.g. how often?) and apply markers
  - visualize marker outputs for samples of chips extracted from the backend
  - decide on the attribution to green/yellow/red
  - **Jupyter + direct access + RESTFul demo**
- Augmenting application logic would, for instance, apply segmentation on the full resolution chips to characterize and enumerate subdivision
- If your marker is robust it could move to the backend, for batch processing



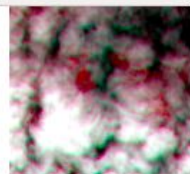
20190501T105039



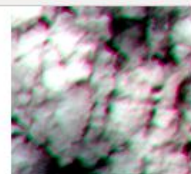
20190506T105031



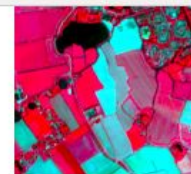
20190511T105039



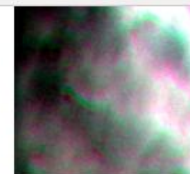
20190516T105031



20190526T105031



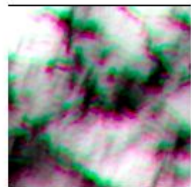
20190531T105039



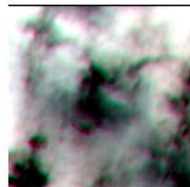
20190605T105031



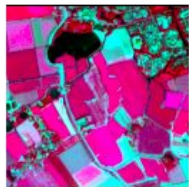
20190610T105039



20190615T105031



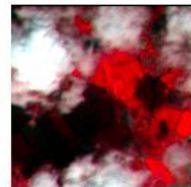
20190620T105039



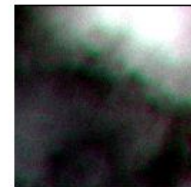
20190625T105031



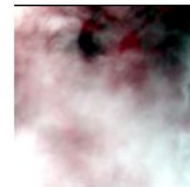
20190630T105039



20190705T105031



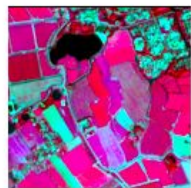
20190710T105039



20190715T105031



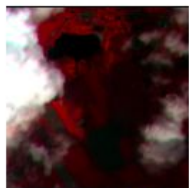
20190720T105039



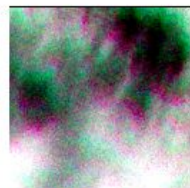
20190725T105031



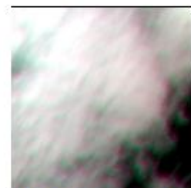
20190730T105039



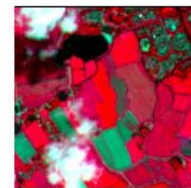
20190804T105031



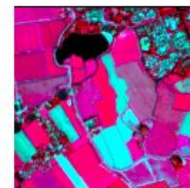
20190809T105039



20190814T105031



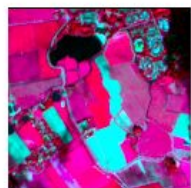
20190819T105029



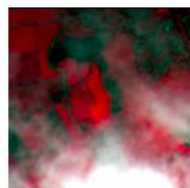
20190824T105031



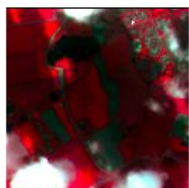
20190829T105029



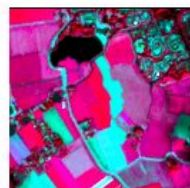
20190903T105031



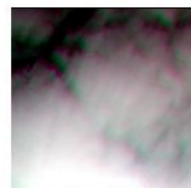
20190908T105029



20190913T105031



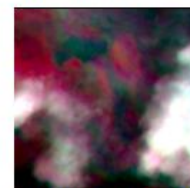
20190918T105029



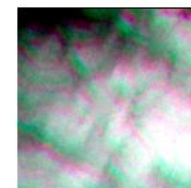
20190928T105029



20191003T105031



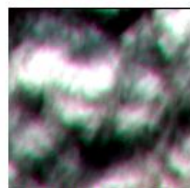
20191008T105029



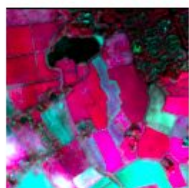
20191013T105031



20191018T105039



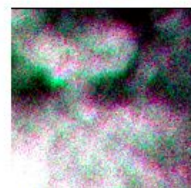
20191023T105111



20191028T105049



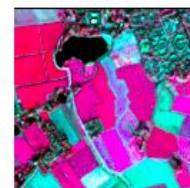
20191102T105211



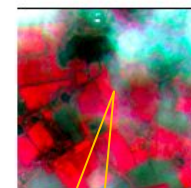
20191107T105139



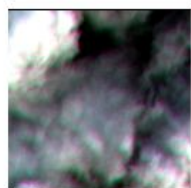
20191112T105301



20191117T105229



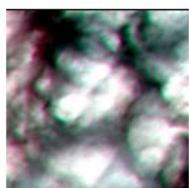
20191122T105351



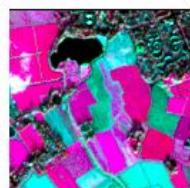
20191127T105309



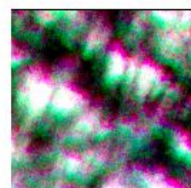
20191202T105421



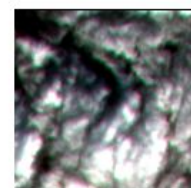
20191207T105329



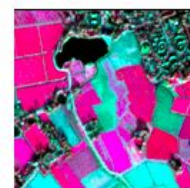
20191212T105441



20191217T105349



20191222T105441



20191227T105349

Sugarbeet,  
not yet  
harvested.

# Frontend take home messages

- The frontend “consumes” data from the backend to feed into post-processing
- Direct connections are possible for developer/analytics (e.g. in Notebooks)
- But most end-users will access RESTful API requests
- Access to time series (and parcel features) is generally very fast
- Access to S3 image extracts is slower, best after (significant) reduction
- Markers are designed and tested in the frontend
- If sufficiently robust, markers can be implemented on the back-end, with dedicated RESTful endpoints
- Frontend analytics is **boundless**, allowing full integration of the python software stack and adoption of new approaches “on the fly”

# Agenda

09:30 - 09:45	Welcome and (very) short introduction into CbM+DIAS
09:45 - 10:15	<b>The Big Picture:</b> functional modules and why we need them
10:15 - 11:00	<b>The Backend:</b> CARD processing and parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	<b>The Frontend:</b> RESTful services and Jupyter Notebooks
11:45 - 12:15	The <u>cbm code base</u> structure, documentation and collaboration
12:15 - 12:30	Next steps and discussion

# jrc-cbm take home messages

- jrc-cbm provides a **modular** approach to the implementation of CbM workflow
- **cloud-centric** in design, but not prescriptive in what should run where
- close to community of practice concerns
- complexity mostly set by required components and frameworks
- module choice based on “**best in class**” open source
- portability amongst DIAS instances proven
- separation of concerns with backend and frontend, but fluid boundary
- all **code open sourced**, to be maintained on github, as PyPi package
- ready for core tasks, open for collaborative build out



# Agenda

09:30 - 09:45	Welcome and (very) short introduction into CbM+DIAS
09:45 - 10:15	<b>The Big Picture:</b> functional modules and why we need them
10:15 - 11:00	<b>The Backend:</b> CARD processing and parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	<b>The Frontend:</b> RESTful services and Jupyter Notebooks
11:45 - 12:15	The <a href="#">cbm code base</a> structure, documentation and collaboration
12:15 - 12:30	<b>Next steps and discussion</b>



# Next steps

- Some Outreach MS are also DIAS onboarders: pip install cbm 😊
- A dedicated technical backend webinar is planned for June 30.
- The core JRC backend tasks for Outreach will be finishing soon.
- This will allow us to show core front-end tasks.
- And tailor to the thematic domains (mowing, grazing, catch crops, etc.)
- A dedicated technical frontend seminar is planned for July and September
- Outreach is an excellent platform to benchmark cross-MS robustness
- Decisions on CAP 2022+ and Copernicus DIAS are key drivers for future
- We will continue to add to jrc-cbm components

# Q&A

[guido.lemoine@ec.europa.eu](mailto:guido.lemoine@ec.europa.eu)

[konstantinos.anastasakis@ext.ec.europa.eu](mailto:konstantinos.anastasakis@ext.ec.europa.eu)



© European Union 2021

Unless otherwise noted the reuse of this presentation is authorised under the [CC BY 4.0](https://creativecommons.org/licenses/by/4.0/) license. For any use or reproduction of elements that are not owned by the EU, permission may need to be sought directly from the respective right holders.



In [ ]:

```
!pip install cbm --upgrade
```

In [3]:

```
from cbm import ipycbm
ipycbm.config()
```

In [4]:

```
import requests
import json

from cbm.utils import config
parcels = 'nld2019'
lon = 4.7785
lat = 52.7784

url = config.get_value(['api', 'url'])
user = config.get_value(['api', 'user'])
pwd = config.get_value(['api', 'pass'])

# Get the parcel id for this location
locurl = f"{url}/query/parcelByLocation?parcels={parcels}&lon={lon}&lat={lat}"

# Parse the response with the standard json module
response = requests.get(locurl.format(parcels, lon, lat), auth = (user, pwd))

parcel = json.loads(response.content)

# Check response
if not parcel:
    print("Parcel query returned empty result")
    sys.exit()
elif not parcel.get(list(parcel.keys())[0]):
    print(f"No parcel found in {parcels} at location ({lon}, {lat})")
    sys.exit()

print(parcel)
```

```
{'ogc_fid': [165238], 'cropname': ['Bieten, suiker-'], 'cropcode': [256], 'srid': [28992], 'area': [44708.6747390683], 'clon': [4.77838203674297], 'clat': [52.7785134313635]}
```

In [5]:

```

import pandas as pd
from datetime import datetime
from matplotlib import pyplot as plt

# Use pid for next request
pid = parcel['ogc_fid'][0]
cropname = parcel['cropname'][0]
area = parcel['area'][0]/10000.0

# query parameter values
aoi = 'nld'
year = '2019'
tstype = 's2'

# Set up the timeseries request
tsurl = f"{url}/query/parcelTimeSeries?aoi={aoi}&year={year}&pid={pid}&tstype={tstype}"

response = requests.get(tsurl.format(aoi, year, pid, tstype), auth = (user, pwd))

# Directly create a pandas DataFrame from the json response
# This should work even if the response is an empty dictionary
df = pd.read_json(response.content)

# Check for an empty dataframe
if df.empty:
    print(f"Timeseries query returned empty result for parcel {pid} and {aoi}, {year} and {tstype}")
    sys.exit()

# Convert the epoch timestamp to a datetime
df['date_part'] = df['date_part'].map(lambda e: datetime.fromtimestamp(e))

# Treat each band separately. Drop duplicate timestamps and rename the 'mean'
df4 = df[df['band']=='B4'][['date_part', 'mean']]
df4.drop_duplicates(['date_part'], inplace=True)
df4.rename(columns={'mean': 'B4'}, inplace=True)

df8 = df[df['band']=='B8'][['date_part', 'mean']]
df8.drop_duplicates(['date_part'], inplace=True)
df8.rename(columns={'mean': 'B8'}, inplace=True)

dfQA = df[df['band']=='SC'][['date_part', 'mean']]
dfQA.drop_duplicates(['date_part'], inplace=True)
dfQA.rename(columns={'mean': 'SC'}, inplace=True)

# Merge back into one DataFrame
dff = pd.merge(df4, df8, on = 'date_part')
dff = pd.merge(dff, dfQA, on = 'date_part')
# Create a NDVI
dff['ndvi'] = (dff['B8'] - dff['B4']) / (dff['B8'] + dff['B4'])

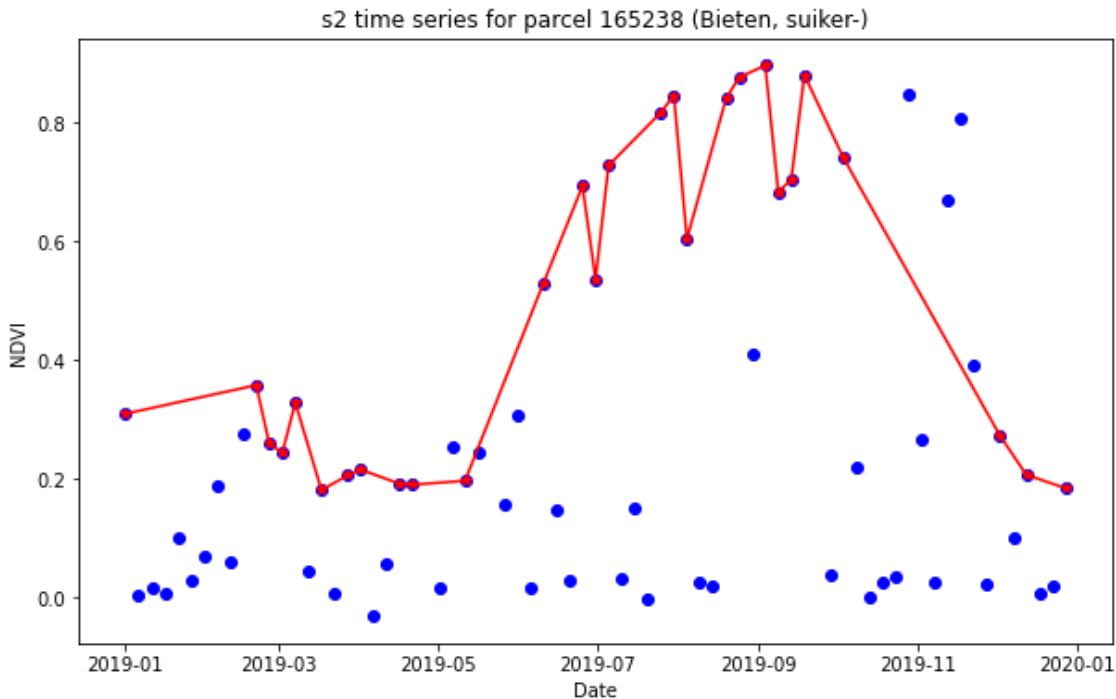
#print(dff)

# Define the criteria for having a cloud free observation
cloudfree = ((dff['SC'] >= 4) & (dff['SC'] < 7))

plt.figure(figsize = (10, 6))

```

```
plt.plot(dff['date_part'], dff['ndvi'], linestyle = ' ', marker = 'o', color = 'blue')
plt.plot(dff[cloudfree]['date_part'], dff[cloudfree]['ndvi'], linestyle = '-', marker = '*', color = 'red')
plt.title(f"{tstype} time series for parcel {pid} ({cropname})")
plt.xlabel('Date')
plt.ylabel('NDVI')
plt.show()
```



In [6]:

```
tstype = 'bs'

# Set up the timeseries request
tsurl = f"{url}/query/parcelTimeSeries?aoi={aoi}&year={year}&pid={pid}&tstype={tstype}"

response = requests.get(tsurl.format(aoi, year, pid, tstype), auth = (user, pwd))

# Directly create a pandas DataFrame from the json response
# This should work even if the response is an empty dictionary
df = pd.read_json(response.content)

# Check for an empty dataframe
if df.empty:
    print(f"Timeseries query returned empty result for parcel {pid} and {aoi}, {year} and {tstype}")
    sys.exit()
```

In [7]:

```

import numpy as np

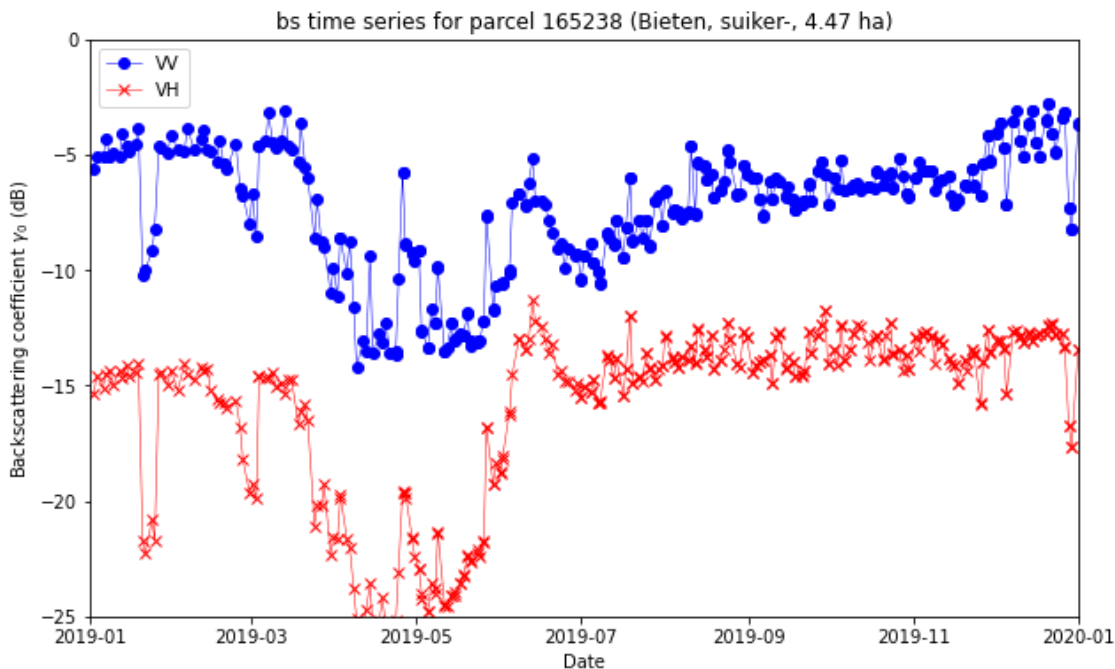
df['obstime']=df['date_part'].map(lambda e: datetime.fromtimestamp(e))
dfVV = df[df['band']=='VV'][['obstime', 'mean']]
dfVV.drop_duplicates(['obstime'], inplace=True)
dfVV.rename(columns={'mean': 'VV'}, inplace=True)

dfVH = df[df['band']=='VH'][['obstime', 'mean']]
dfVH.drop_duplicates(['obstime'], inplace=True)
dfVH.rename(columns={'mean': 'VH'}, inplace=True)

# Merge back into one DataFrame
dff = pd.merge(dfVV, dfVH, on = 'obstime')
# Create a ratio
dff['ratio'] = dff['VH']/dff['VV']

plt.figure(figsize = (10, 6))
plt.plot(dff['obstime'], 10.0*np.log10(dff['VV']), linestyle = '-', linewidth =
0.5, marker = 'o', color = 'blue', label = 'VV')
plt.plot(dff['obstime'], 10.0*np.log10(dff['VH']), linestyle = '-', linewidth =
0.5, marker = 'x', color = 'red', label = 'VH')
#plt.plot(dff['obstime'], 10.0*np.log10(dff['ratio']), linestyle = ' ', marker =
'v', color = 'green')
plt.title(f"{tstype} time series for parcel {pid} ({cropname}, {area:.2f} ha)")
plt.xlabel('Date')
plt.ylabel('Backscattering coefficient  $\gamma_0$  (dB)')
plt.xlim(pd.to_datetime('2019-01-01'), pd.to_datetime('2020-01-01'))
plt.legend()
plt.ylim(-25, 0)
plt.show()

```



In [8]:

```
from cbm import background

aoi = 'nld'      # area of interest (str)
year = 2019     # the year of the parcels dataset (int)
pid = 165238    # latitude in decimal degrees (float)
chipsize = 750 # size of the chip in pixels (int)
extend = 1500   # size of the chip in meters (float)

# images from tile map servers: Google Bing (list)
# tms=['Google', 'Bing', 'nl2016', 'nl2017', 'nl2018', 'nl2019', 'nl2020']
tms=['nl2017', 'nl2018', 'nl2019', 'nl2019ir']

background(aoi, year, pid, chipsize, extend, tms)
```





# CbM git repository

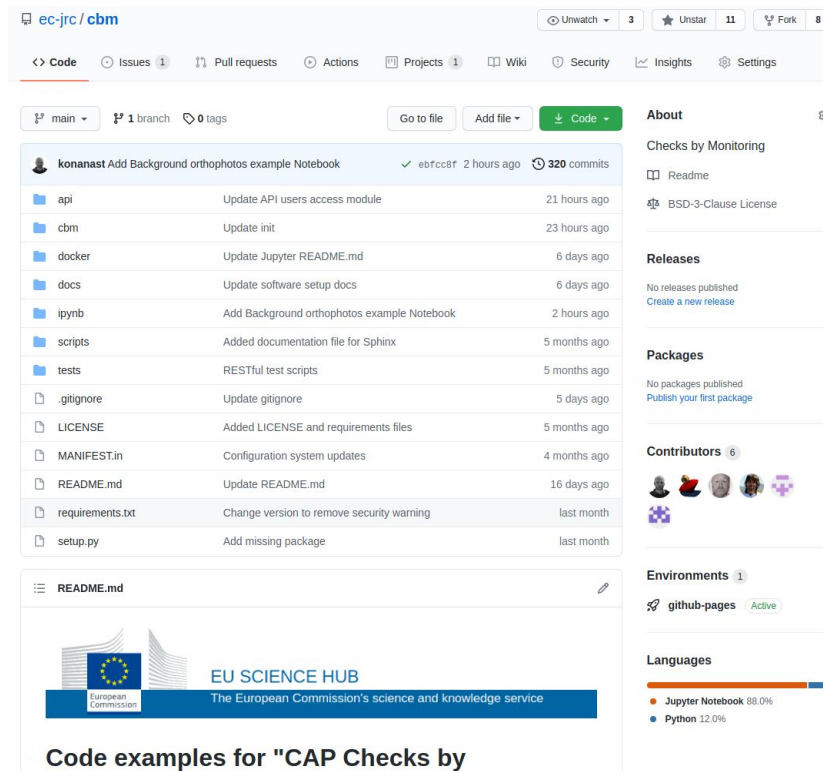
Introduction of CbM git repository

*GTCAP Team*



# The cbm code base structure, documentation and collaboration

- git repository
- Documentation
- Jupyter Notebooks
- CbM python requirements
- The jrc 'cbm' Python library
- How to contribute
- Links to get started



ec-jrc / cbm


<> Code Issues 1 Pull requests Actions Projects 1 Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file Code

File	Commit	Time
konanast Add Background orthophotos example Notebook	✓ ebfcc8f	2 hours ago 320 commits
api	Update API users access module	21 hours ago
cbm	Update init	23 hours ago
docker	Update Jupyter README.md	6 days ago
docs	Update software setup docs	6 days ago
ipynb	Add Background orthophotos example Notebook	2 hours ago
scripts	Added documentation file for Sphinx	5 months ago
tests	RESTful test scripts	5 months ago
.gitignore	Update gitignore	5 days ago
LICENSE	Added LICENSE and requirements files	5 months ago
MANIFEST.in	Configuration system updates	4 months ago
README.md	Update README.md	16 days ago
requirements.txt	Change version to remove security warning	last month
setup.py	Add missing package	last month

README.md

 **EU SCIENCE HUB**  
The European Commission's science and knowledge service

Code examples for "CAP Checks by

**About**

Checks by Monitoring

Readme

BSD-3-Clause License


**Releases**

No releases published  
[Create a new release](#)

**Packages**

No packages published  
[Publish your first package](#)

**Contributors** 6



**Environments** 1

[github-pages](#) Active

**Languages**

- Jupyter Notebook 88.0%
- Python 12.0%

# CbM git repository

**JRC cbm  
git repository**  
<https://github.com/ec-jrc/cbm>



**Technical issues  
page on github**  
[https://github.com/  
ec-jrc/cbm/issues](https://github.com/ec-jrc/cbm/issues)

**Docker images available  
on Dockerhub:**

<https://hub.docker.com/u/gtcap>



**Documentation published with Sphinx**, a full featured intelligent python documentation generator. Can be viewed at:

- <https://jrc-cbm.readthedocs.io> or
- <https://ec-jrc.github.io/cbm/> (under development)



**cbm python library available on  
Python Package Index (PyPI)**

<https://pypi.org/project/cbm/>

users can install cbm with:

```
pip install cbm
```



# CbM Repository structure

<https://github.com/ec-jrc/cbm>

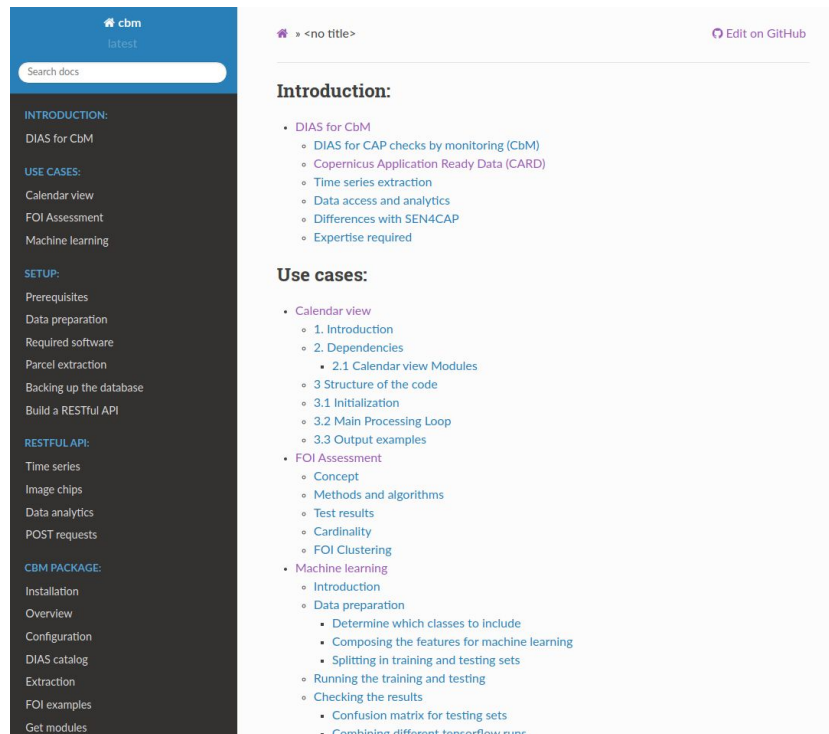
This repository contains example scripts and documentation to get started with CbM, includes:

- **api/**: Files to create a RESTful API for cbm
- **cbm/**: Python library for Checks by Monitoring (available at pypi.org)
- **docker/**: Docker image files
- **docs/**: Sphinx documentation files
- **ipynb/**: Jupyter Notebook examples
- **scripts/**: Command line scripts
  - **extraction/**: Extraction example scripts
  - **calendar\_view/**: Time series calendar (Requires RESTful API server)
- **tests/**: Test scripts for testing a variety of functionalities.

# CbM documentation

- INTRODUCTION:
  - DIAS for CbM
- USE CASES:
  - Calendar view
  - FOI Assessment
  - Machine learning
- SETUP:
  - Prerequisites
  - Data preparation
  - Required software
  - Parcel extraction
  - Build a RESTful API
- RESTFUL API USE:
  - Time series
  - Image chips
  - Data analytics
  - POST requests

<https://jrc-cbm.readthedocs.io> or  
<https://ec-jrc.github.io/cbm/> (under development)



The screenshot shows the CbM documentation website. The sidebar on the left contains a search bar and a table of contents with the following sections:

- INTRODUCTION:
  - DIAS for CbM
- USE CASES:
  - Calendar view
  - FOI Assessment
  - Machine learning
- SETUP:
  - Prerequisites
  - Data preparation
  - Required software
  - Parcel extraction
  - Backing up the database
  - Build a RESTful API
- RESTFUL API:
  - Time series
  - Image chips
  - Data analytics
  - POST requests
- CBM PACKAGE:
  - Installation
  - Overview
  - Configuration
  - DIAS catalog
  - Extraction
  - FOI examples
  - Get modules

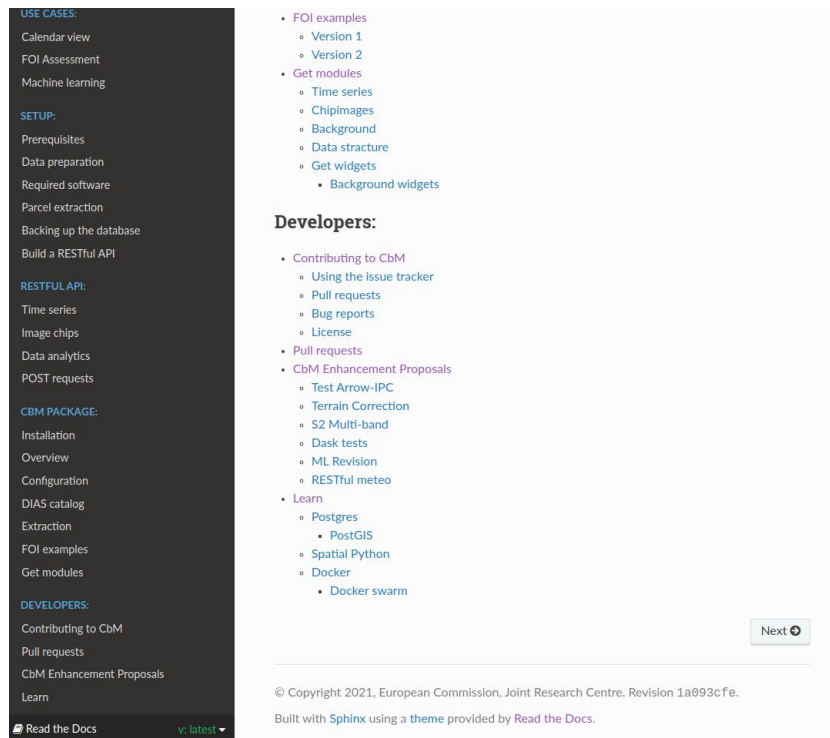
The main content area on the right shows the 'Introduction' section with a table of contents:

- **DIAS for CbM**
  - DIAS for CAP checks by monitoring (CbM)
  - Copernicus Application Ready Data (CARD)
  - Time series extraction
  - Data access and analytics
  - Differences with SEN4CAP
  - Expertise required
- **Use cases:**
  - **Calendar view**
    - 1. Introduction
    - 2. Dependencies
      - 2.1 Calendar view Modules
    - 3 Structure of the code
      - 3.1 Initialization
      - 3.2 Main Processing Loop
      - 3.3 Output examples
  - **FOI Assessment**
    - Concept
    - Methods and algorithms
    - Test results
    - Cardinality
    - FOI Clustering
  - **Machine learning**
    - Introduction
    - Data preparation
      - Determine which classes to include
      - Composing the features for machine learning
      - Splitting in training and testing sets
    - Running the training and testing
      - Checking the results
        - Confusion matrix for testing sets
        - Combine different tensorflow runs

# CbM documentation

- CBM PACKAGE:
  - Installation
  - Configuration
  - Functions
    - DIAS catalog
    - Extraction
    - FOI examples
    - Get modules
- DEVELOPERS:
  - Contributing to CbM
  - Pull requests
  - CbM Enhancement Proposals
  - Learn

<https://ec-jrc.github.io/cbm/> (under development) or  
<https://jrc-cbm.readthedocs.io>



The screenshot shows the CbM documentation website. On the left is a dark navigation menu with the following sections:

- USE CASES:
  - Calendar view
  - FOI Assessment
  - Machine learning
- SETUP:
  - Prerequisites
  - Data preparation
  - Required software
  - Parcel extraction
  - Backing up the database
  - Build a RESTful API
- RESTFUL API:
  - Time series
  - Image chips
  - Data analytics
  - POST requests
- CBM PACKAGE:
  - Installation
  - Overview
  - Configuration
  - DIAS catalog
  - Extraction
  - FOI examples
  - Get modules
- DEVELOPERS:
  - Contributing to CbM
  - Pull requests
  - CbM Enhancement Proposals
  - Learn

At the bottom of the menu is a "Read the Docs" button and a version selector set to "v. latest".

The main content area on the right displays the "DEVELOPERS" section:

- **FOI examples**
  - Version 1
  - Version 2
- **Get modules**
  - Time series
  - Chipimages
  - Background
  - Data structure
  - Get widgets
    - Background widgets

Below this is the "Developers:" section:

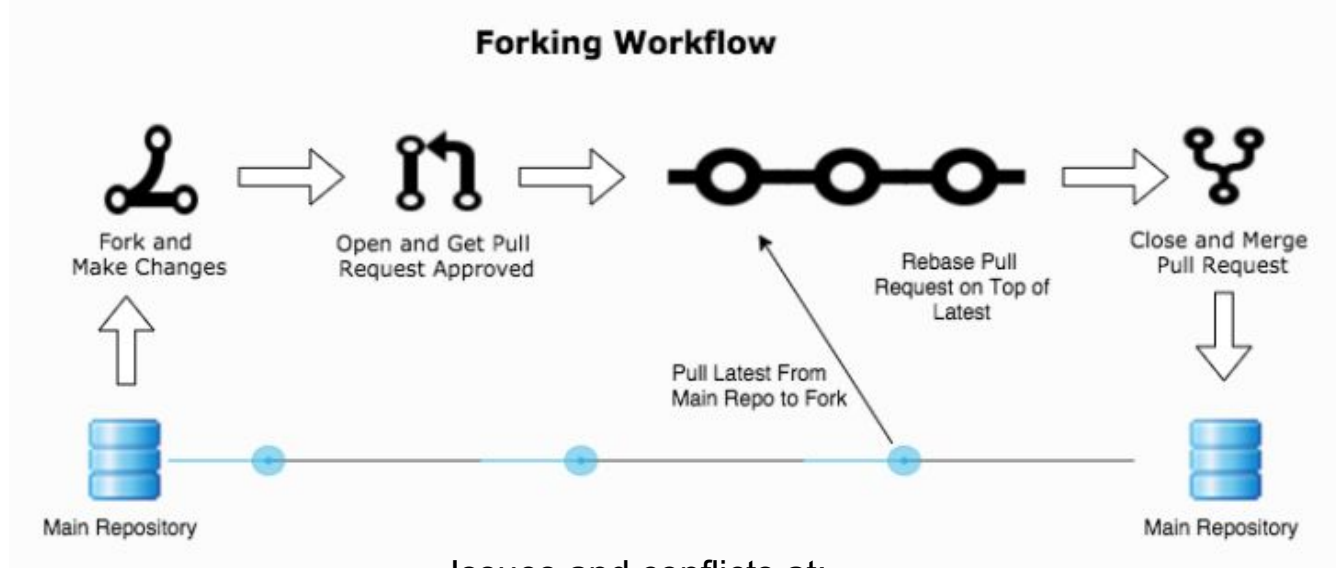
- **Contributing to CbM**
  - Using the issue tracker
  - Pull requests
  - Bug reports
  - License
- **Pull requests**
- **CbM Enhancement Proposals**
  - Test Arrow-IPC
  - Terrain Correction
  - S2 Multi-band
  - Dask tests
  - ML Revision
  - RESTful meteo
- **Learn**
  - Postgres
    - PostGIS
  - Spatial Python
  - Docker
    - Docker swarm

At the bottom right of the main content area is a "Next" button with a right-pointing arrow.

At the bottom of the page, there is a copyright notice: "© Copyright 2021, European Commission, Joint Research Centre. Revision 1a093cfe." and a note: "Built with Sphinx using a theme provided by Read the Docs."

# Contribution

We use the standard github method: create a fork, make changes and open a pull request.



Issues and conflicts at:

<https://github.com/ec-jrc/cbm/issues>

\*Cbm contribution suggestions: use of common naming standards, no binary files, no hard-coded configurations, code length 80 characters, help docstring in the code

# Python requirements for CbM

Most important python libraries used in cbm:

- **\*boto3**: transfer data from S3 store to local store (or directly into memory)
- **\*psycopg2**: PostgreSQL database adapter for Python
- **\*requests**: for making HTTP requests in Python (relevant for RESTful use)
- **\*rasterio**: gives access to a geospatial raster file
- **\*\*gdal**: translator library for raster and vector geospatial data formats
  
- **numpy**: for multi-dimensional arrays and matrices
- **pandas**: for manipulating numerical tables and time series
- **scipy**: for scientific computing and technical computing
- **tflearn**: deep learning library featuring a higher-level API for TensorFlow
- **matplotlib**: plotting library for Python
- **lxml**: XML parsing (to be migrated to **json**)
- **ipyleaflet**: Interactive maps in the Jupyter notebook
- **ipywidgets**: interactive HTML widgets for Jupyter notebooks
- **voila**: turn the jupyter notebook into a standalone web application

# The 'cbm' Python package

cbm: Python library for Checks by Monitoring

Install with “`pip install cbm`”

- `card2db`: Transfer metadata from the DIAS catalog
- `extraction`: Parcel extraction routines
- `foi`: FOI module
- `get`: Download parcels data (time series, sentinel chip images and orthophotos)
- `report`: Generate reports for selected parcels (under development)
- `ipycbm`: Interactive notebook widgets with the subpackages:
  - `ipycbm.config`: Interactive configuration module
  - `ipycbm.foi`: Interactive FOI module
  - `ipycbm.qa`: Interactive QA module
  - `ipycbm.extract`: Interactive extraction modules
  - `ipycbm.get`: Interactive module for downloading data to local storage
  - `ipycbm.view`: Interactive module to view data (downloaded or directly from remote location)




# cbm - Configuration file

- The first time the cbm library is imported it will create the config/main.json configuration file, if it does not exist. To edit the main configuration file manually, run in the terminal:

```
python3 -c "import cbm"
nano config/main.json
```

- To edit the configuration file interactively in a jupyter notebook:

```
from cbm import ipycbm
ipycbm.config()
```

 Empty temp folder Your temp folder 'temp/' has old files: '['test2019', '.ipynb\_checkpoints']', do you want to delete them?

DataSource General


Data sources:  RESTful API for CbM.  
 Direct access to database and object storage.

RESTful API Settings.

API URL:  Format: http://0.0.0.0/ or https://0.0.0.0/

API User:

API Passw...

 Save

# cbm - FOI example

- FOI non interactive workflow (For version 1 you will need to set manually the configuration file for database connection):

```
import cbm
vector_file = "data/parcels2020.shp"
raster_file = "data/raster.tif"
yaml_file = "pixelvalues_classes.yml"
pre_min_het = 30
pre_max_het = 70
area_threshold = 2000
```

```
cbm.foi(vector_file, raster_file, yaml_file,pre_min_het, pre_max_het, area_threshold)
```


- FOI with interactive python widgets in notebooks:

```
from cbm import ipycbm
ipycbm.foi()
```




FOI Assessment V1 | FOI Assessment V2 | Help | Settings

FOI procedures need direct access to the database.  
In case there no image is provided, access to object storage will be needed to generate the base image from sentinel images.



1. Connect to database and object storage.

Configure: 1 

a. Upload .shp to the server.

Folder:   Select files: (0)  

b. Import uploaded .shp to the database. Add a short name, max 15 characters for the parcels table e.g.:escat2020.

Table name:  Select .shp:    Import .shp file  Remove old entries

# ipycbm.get()

Get Data Help Settings

- Select the region and the year to get parcel information.  
AOI: Spain (NOUR Subset) Year: 2019
- Select a method to download parcel data.  
Parcel ID Coordinates Map marker Polygon
- Select datasets to download.  
Time series Chip images
- Download the selected data.  
By default data will be stored in the temp folder (temp/), you will be asked to empty the temp folder each time you start the notebook. In your personal data folder (data/) you can permanently store the data.  
Select folder:  Temporary folder: 'temp/'.  
 Personal data folder: 'data'.  
Download

# ipycbm.view()



# Links to get started

- **CbM repository:** <https://github.com/ec-jrc/cbm>
- CbM Documentation: <https://jrc-cbm.readthedocs.io> or <https://ec-jrc.github.io/cbm/> (under development)
- CbM Python library: <https://pypi.org/project/cbm>
- CbM docker images: <https://hub.docker.com/u/gtcap>

## Other technical information:

- Creating pull requests with an interactive way:
  - [docs.github.com/en/github/collaborating-with-issues-and-pull-requests/creating-a-pull-request](https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/creating-a-pull-request)
- Using git guide non interactively:
  - <http://rogerdudler.github.io/git-guide>
- Google Python Style Guide:
  - <https://google.github.io/styleguide/pyguide.html>
- Markdown (.md) and reStructuredText (.rst) guides:
  - <https://www.markdownguide.org>, <https://docutils.sourceforge.io/rst.html>
- Jupyter Notebooks:
  - <https://jupyter-notebook.readthedocs.io/>
  - Jupyter Notebook CheatSheet: [Jupyter\\_Notebook\\_CheatSheet\\_Edureka.pdf](#)
- Get started with python:
  - <https://python101.pythonlibrary.org/>
  - <https://www.programiz.com/python-programming/first-program>
  - <https://realpython.com/tutorials/data-viz> <https://python-graph-gallery.com>
  - <https://realpython.com/tutorials/machine-learning>

# Q&A

[guido.lemoine@ec.europa.eu](mailto:guido.lemoine@ec.europa.eu)

[konstantinos.anastasakis@ext.ec.europa.eu](mailto:konstantinos.anastasakis@ext.ec.europa.eu)



© European Union 2020

Unless otherwise noted the reuse of this presentation is authorised under the [CC BY 4.0](https://creativecommons.org/licenses/by/4.0/) license. For any use or reproduction of elements that are not owned by the EU, permission may need to be sought directly from the respective right holders.

