Webinar on DIAS for CbM Outreach - Session 2

Date: Friday, 30th June 2021

Agenda

09:30 - 09:45 Welcome and short introduction into the JRC-CbM backend (Guido Lemoine, JRC)

09:45 - 10:30 DIAS platform: overview of resources and how they fit together (Guido Lemoine, JRC)

10:30 - 11:00 The Spatial Database: basics and support to parcel extraction (Guido Lemoine, JRC)

11:00 - 11:15 Break

11:15 - 11:45 CARD generation and parcel time series extraction (Guido Lemoine, JRC)

11:45 - 12:15 Supporting the Frontend: RESTful and JHub server set up (Konstantinos Anastasakis, JRC)

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

CbM on DIAS: the jrc-cbm backend

On-line training for Outreach, 30 June 2021

JRC D5 – GTCAP Team

Agenda

09:30 - 09:45	Welcome and short introduction into the jrc-cbm backend
09:45 - 10:30	**DIAS platform**: overview of resources and how they fit together
10:30 - 11:00	**The Spatial Database**: basics and support to parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	**CARD generation and parcel time series extraction**
11:45 - 12:15	**Supporting the Frontend**: RESTful and JHub server set up
12:15 - 12:30	Next steps and discussion

Agenda

09:30 - 09:45	Welcome and short introduction into the jrc-cbm backend
09:45 - 10:30	**DIAS platform**: overview of resources and how they fit together
10:30 - 11:00	**The Spatial Database**: basics and support to parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	**CARD generation and parcel time series extraction**
11:45 - 12:15	**Supporting the Frontend**: RESTful and JHub server set up
12:15 - 12:30	Next steps and discussion

Welcome

- A technical introduction to the jrc-cbm **backend** implementation on DIAS.
- Short rehash of the “deep dive” backend essentials
- 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.

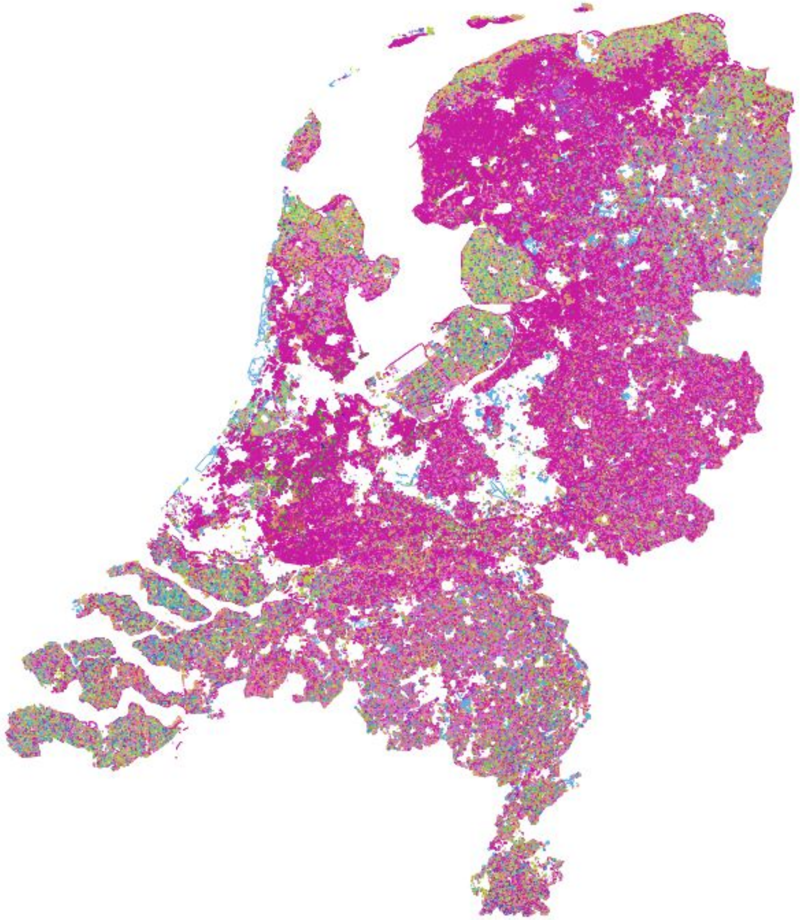
European Commission

Audience

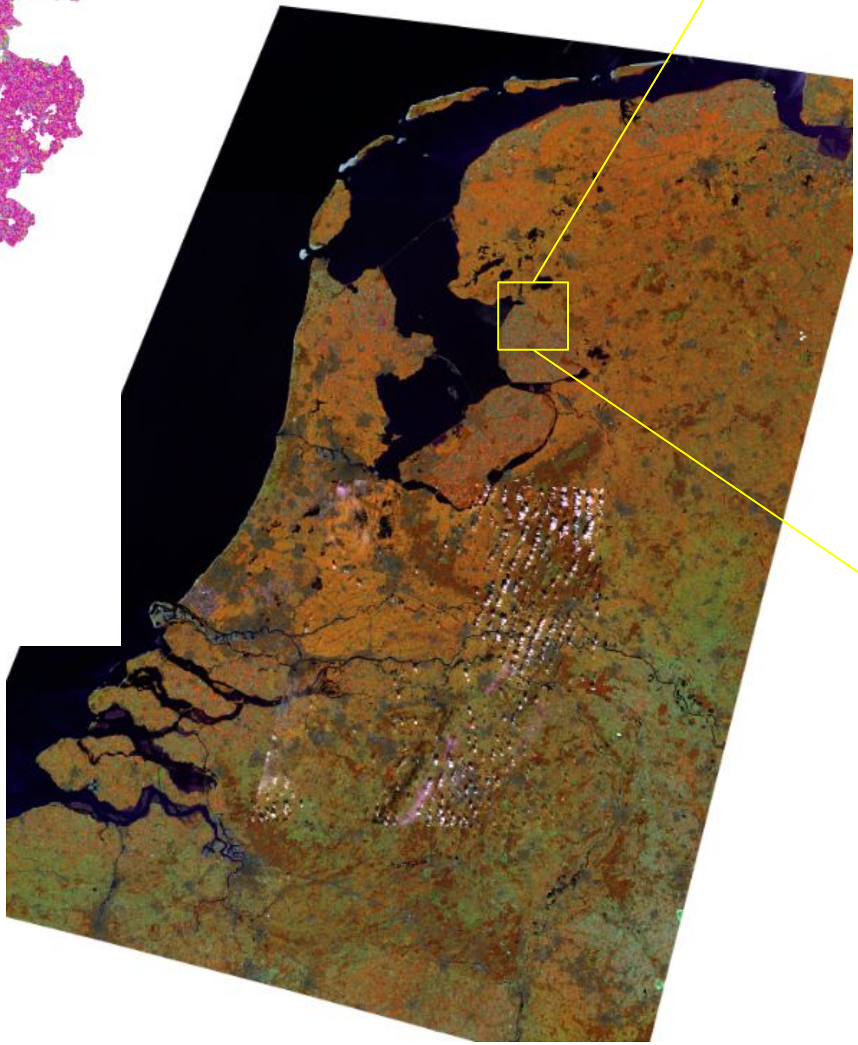
- For IT managers/programmers/developers:
    - Cloud compute on DIAS
    - “Marshalling” resources for CbM backend
    - The Spatial Database components
    - CARD data and extraction
    - Server components to support Frontend
    - Issues, caveats to be aware of

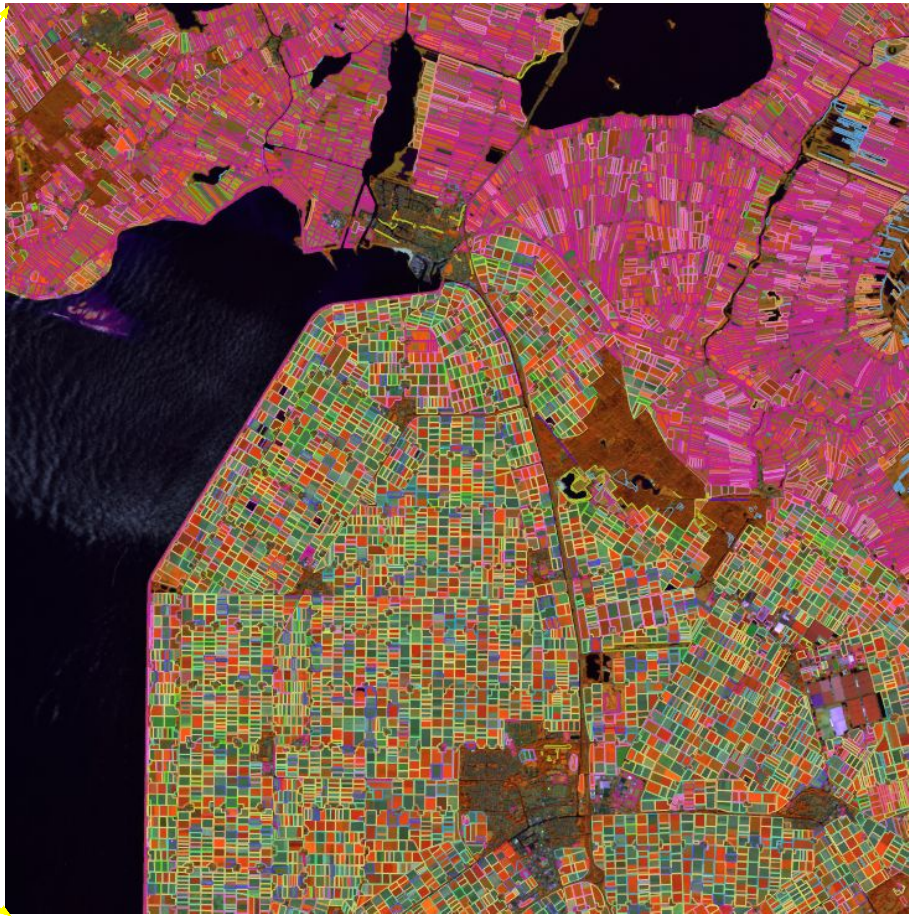
Context

- 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



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





n BRP 2019 @ pdok.nl

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

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

European
Commission

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**!
- Outreach: PA organized in separate database schema
- 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.

European Commission

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
- **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
- Licensed under BSD Clause 3 (facilitates maximum re-use)

Open Source software components used

openstack CLOUD SOFTWARE

ubuntu

SNAP

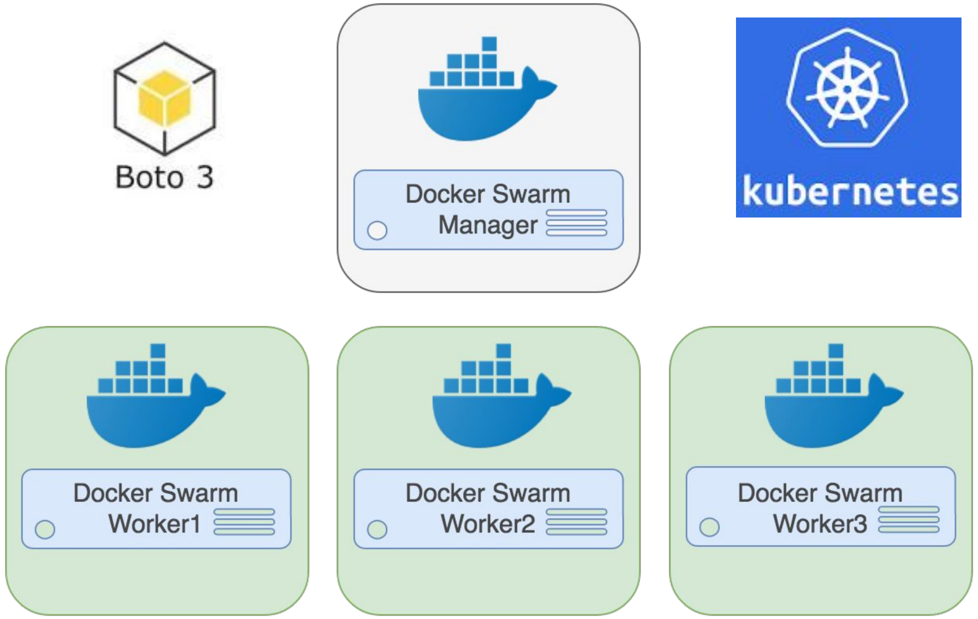
GDAL

python

NumPy

PostGIS Spatial PostgreSQL

Boto 3



kubernetes

Pandas

jupyter

LEAFLET

TensorFlow

PYTORCH

Flask

APACHE HTTP SERVER PROJECT

European Commission

Agenda

09:30 - 09:45	Welcome and short introduction into the jrc-cbm backend
09:45 - 10:30	**DIAS platform**: overview of resources and how they fit together
10:30 - 11:00	**The Spatial Database**: basics and support to parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	**CARD generation and parcel time series extraction**
11:45 - 12:15	**Supporting the Frontend**: RESTful and JHub server set up
12:15 - 12:30	Next steps and discussion

The DIAS Platform

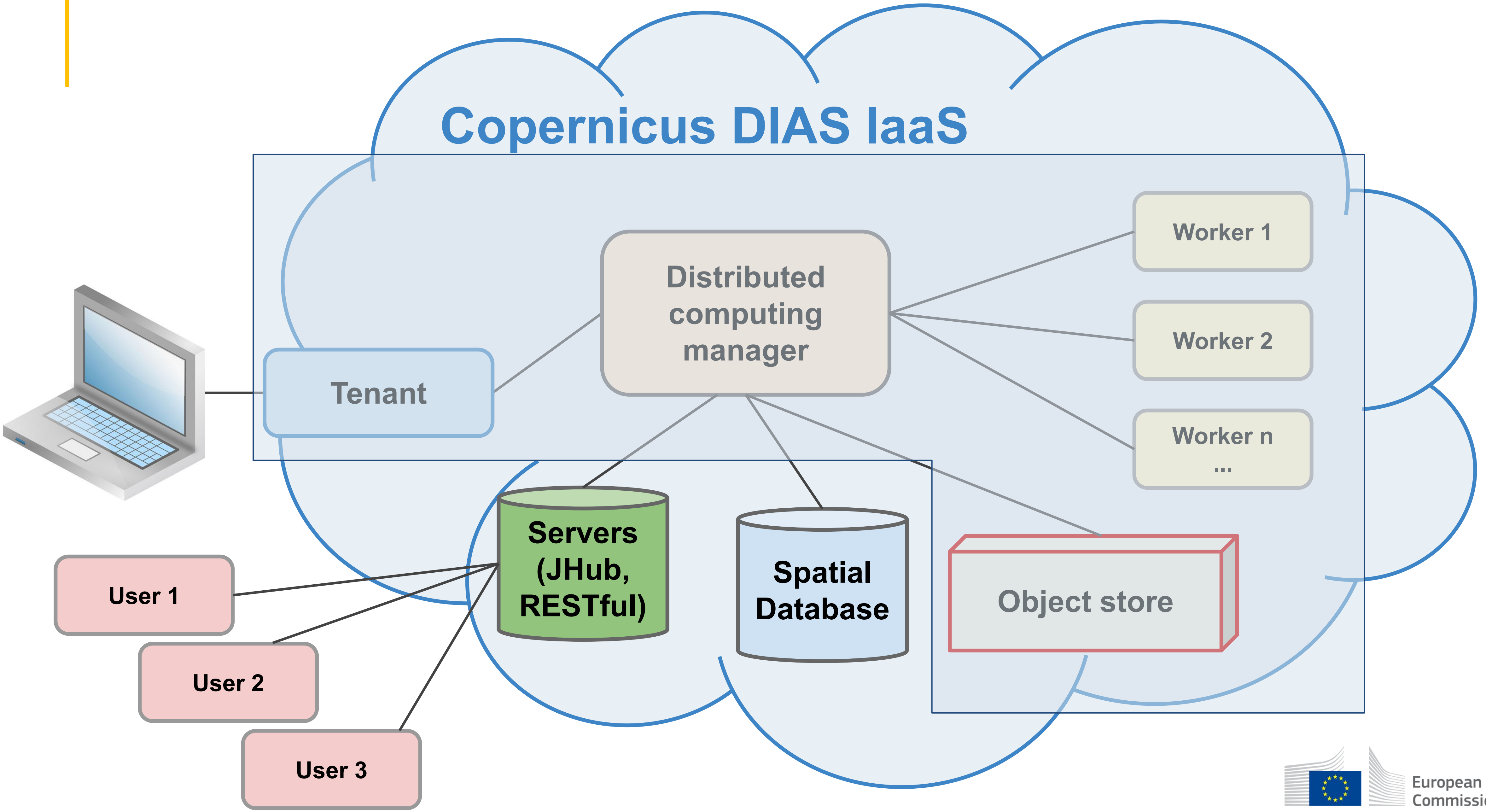
- The “Data and Information Access Services” are cloud compute infrastructures closely coupled to PB-scale Copernicus data archives
- The Copernicus program funds 5 DIAS IaaS providers (until end of 2021)
- We focus on CREODIAS use, but technical parallels with others
- DIAS is built on EU cloud service infrastructure providers
- DIAS manages the dedicated Copernicus object store (size, content, cache)
- A DIAS account provides access to a tenant and web-based GUI
- DG AGRI and DG DEFIS fund DIAS accounts for use in CbM
- 2021+ support is under discussion

DIAS (a technical view)

- DIAS is a bare-bones Infrastructure as a Service (IaaS)
- A **platform** to select and configure compute resources
- **Object storage** giving access to [partial] copies of the Copernicus archive
- An **interface** (catalog) to find what is available in the DIAS store
- **Documentation** on the various components, protocols, FAQ
- **Technical support** for IaaS issues
- Other options beyond standard accounts (not used in CbM)

The “third party” on DIAS

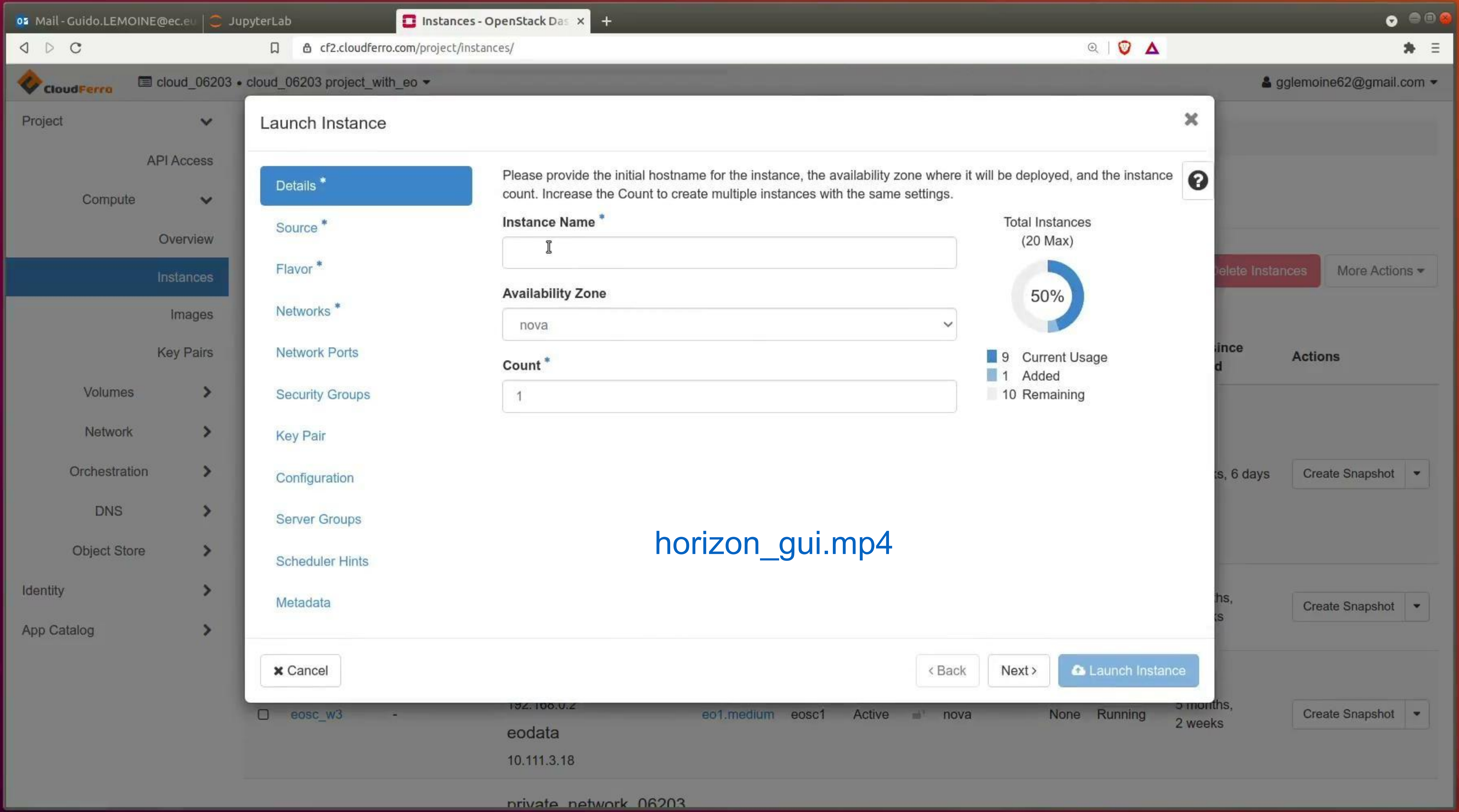
- The “third party” needs to piece together the components, to set up relevant functionalities, using DIAS resources
- This requires expertise in:
    - Linux Virtual Machine (VM) install and configuration
    - Spatial databases
    - (Python) programming for geospatial analysis
    - VM orchestration for parallel computing (openstack, Docker, k8s)
    - Server interfaces for data access and analytics (Jupyter Hub, RESTful)
- **Good news:** all essential components are standards based on open source!



European Commission

DIAS resource marshalling

- The DIAS tenant can select and configure VMs for specific functions
  - permanent VMs (e.g. database server, Jupyter Hub, RESTful)
  - transient VMs (**use on demand**, run large tasks in parallel, **tear down**)
- Via GUI, programmatically (openstack), or with help of DIAS provider
- Fully configurable CPU, RAM, disk size, internal and external network, etc.
- With possibility to choose from pre-configured “flavors”
- And store pre-configured VM images for duplication and later re-use
- All VMs typically accessible via keyed SSH (port 22)
- **WARNING**: you pay for marshalled resources, even if you do not use them!



horizon_gui.mp4

eouser@eosc1: ~/openstack

File Edit View Search Terminal Help

15	eo1.small	2048	16	0	2	True
16	eo1.xmedium	2048	8	0	1	True
17	eo1.medium	4096	16	0	2	True
18	eo1.large	8192	32	0	4	True
19	eo2.medium	4096	16	0	1	True
1d98ebc9-72d4-41ac-8d56-469d723026d5	hmd.medium	16384	50	0	2	True
20	eo2.large	8192	32	0	2	True
21	eo2.xlarge	16384	64	0	4	True
22	eo2.2xlarge	32768	128	0	8	True
23	hm.medium	16384	64	0	2	True
235f4448-c19c-4c58-ac63-0c4eda1fbbeb	hmd.3xlarge	262144	800	0	32	True
24	hm.large	32768	128	0	4	True
25	hm.xlarge	65536	256	0	8	True
26	hm.2xlarge	131072	384	0	16	True
2b78e732-cd8a-469c-8848-8ea291a3f70a	eo2a.large	8192	32	0	2	True
31942087-b93d-40f1-b951-7156a828d509	eo2a.5xlarge	240000	1024	0	64	True
3878c501-022d-4514-b3bb-5884d58d2ffa	eo2a.medium	4096	16	0	1	True
39534135-37cb-47a4-9589-36fe2c7d1339	ds.large	114688	128	0	40	True
410fe469-8a3d-4fba-b078-5b436e585473	eo2a.3xlarge	65536	256	0	16	True
4dbf946e-8bc2-4b83-8d26-3dbbbea949f8	ds.3xlarge.4gpu	497664	128	0	48	True
5ea5fa6a-ae1c-40ab-9c78-529bdea9c656	hmd.2xlarge	131072	400	0	16	True
77e1d9c3-e85a-4cfa-98f1-04529032a084	eo2a.2xlarge	32768	128	0	8	True
7dc7e7d7-87db-4e9a-ad5a-4240a9fe4b6e	hmd.xlarge	65536	200	0	8	True
95a19bd5-f293-46ea-9f26-99e6956ba3d9	hm.4xlarge	496000	384	0	48	True
a7aa0fe1-e7b7-4a8c-9d98-f72826e5d59b	ds.large.gpu	114688	64	0	40	True
ac7be480-670d-49ac-9668-a02cf98c00d6	ds.2xlarge	372736	128	0	40	True
ade62292-ad90-4779-a613-66090803bbe8	eo2a.xlarge	16384	64	0	4	True
c14b747d-f1a7-4db6-af95-566eb114b6fa	ds.3xlarge	497664	128	0	48	True
cc68a87e-1045-4e35-9fee-47a2bfc28cd8	pfe-flv3	12288	40	0	7	True
e03d92be-077f-4551-8605-8403adbc7c75	ds.medium	49152	64	0	40	True
e73edafc-2900-4007-bb63-9dbb695afc16	gpu.medium	120000	64	0	12	True
e9d3ba3e-2814-4f84-8c56-52836331ab97	hmd.large	32768	100	0	4	True
ee35dc90-4729-4a85-85e1-e9e0ab3ba431	re-flv3	2048	40	0	2	True
fede983f-827a-4f40-a70c-243188c65867	hm.3xlarge	262144	384	0	32	True

```
eouser@eosc1:~/openstack$
```

openstack_cli.mp4

DIAS object storage

- Object storage is the preferred storage for immutable “Big Data” blobs
- Write once, read often (e.g. YouTube video, DIAS Sentinel data)
- Simpler to manage and extend than file or block storage, much cheaper
- Multiple 10s of PetaBytes (e.g. CREODIAS ~ 20 PB, GEE ~ 85 PB)
- Requires specific protocol to read, CREODIAS uses S3 (AWS standard)
- Generally slower access, esp. since not optimized for partial reads
- Requires S3 credentials for public and/or private bucket access
- Access via s3fs mount, boto3 (python) and gdal vsis3 drivers

eouser@eosc1: ~

File Edit View Search Terminal Help

```
eouser@eosc1:~$ ls /eodata/Sentinel-1/SAR/CARD-BS/2020/05/10/S1A_IW_GRDH_1SDV_20200510T063052_20200510T063117_032499_03C37F_5E9B_CARD_BS
S1A_IW_GRDH_1SDV_20200510T063052_20200510T063117_032499_03C37F_5E9B_CARD_BS.data
S1A_IW_GRDH_1SDV_20200510T063052_20200510T063117_032499_03C37F_5E9B_CARD_BS.dim
eouser@eosc1:~$ ls /eodata/Sentinel-1/SAR/CARD-BS/2020/05/10/S1A_IW_GRDH_1SDV_20200510T063052_20200510T063117_032499_03C37F_5E9B_CARD_BS/S1A_IW_GRDH
_1SDV_20200510T063052_20200510T063117_032499_03C37F_5E9B_CARD_BS.data/
Gamma0_VH.hdr  Gamma0_VH.img  Gamma0_VV.hdr  Gamma0_VV.img  vector_data
eouser@eosc1:~$ gdalinfo /eodata/Sentinel-1/SAR/CARD-BS/2020/05/10/S1A_IW_GRDH_1SDV_20200510T063052_20200510T063117_032499_03C37F_5E9B_CARD_BS/S1A_I
W_GRDH_1SDV_20200510T063052_20200510T063117_032499_03C37F_5E9B_CARD_BS.data/Gamma0_VV.img
Driver: ENVI/ENVI .hdr Labelled
Files: /eodata/Sentinel-1/SAR/CARD-BS/2020/05/10/S1A_IW_GRDH_1SDV_20200510T063052_20200510T063117_032499_03C37F_5E9B_CARD_BS/S1A_IW_GRDH_1SDV_202005
10T063052_20200510T063117_032499_03C37F_5E9B_CARD_BS.data/Gamma0_VV.img
       /eodata/Sentinel-1/SAR/CARD-BS/2020/05/10/S1A_IW_GRDH_1SDV_20200510T063052_20200510T063117_032499_03C37F_5E9B_CARD_BS/S1A_IW_GRDH_1SDV_202005
10T063052_20200510T063117_032499_03C37F_5E9B_CARD_BS.data/Gamma0_VV.hdr
Size is 31185, 24027
Coordinate System is:
PROJCS["WGS 84 / Auto UTM",
    GEOGCS["WGS84(DD)",
        DATUM["WGS84",
            SPHEROID["WGS84",6378137.0,298.257223563]],
        PRIMEM["Greenwich",0.0],
        UNIT["degree",0.017453292519943295],
        AXIS["Geodetic longitude",EAST],
        AXIS["Geodetic latitude",NORTH]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["central_meridian",3.0],
    PARAMETER["latitude_of_origin",0.0],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000.0],
    PARAMETER["false_northing",0.0],
    UNIT["Meter",1],
    AXIS["Easting",EAST],
    AXIS["Northing",NORTH]]
Origin = (-112107.506448588916101,6089257.482352759689093)
Pixel Size = (10.000000000000000,-10.000000000000000)
Metadata:
  Band 1-Gamma0_VV
```

s3_store.mp4

DIAS catalog

- Metadata of the Sentinel blobs are written to the DIAS catalog
- Searchable via OpenSearch or other standards (not all DIAS instances)
- Interactively in GUI
- Or as parsable XML (or JSON) for scripted queries
- In CbM metadata is parsed into PostgreSQL/PostGIS dias_catalogue table
- **WARNING**: Level 2 may not be in online store (check metadata flags!)

card_catalogue_creodias.mp4

Parallel processing on DIAS

- A key asset of DIAS is the possibility to process across multiple VMs
- jrc-cbm backend tasks are “embarrassingly parallel” (e.g. extraction)
- Marshalled resources need to be orchestrated to run parallel tasks
- Docker containerization to ease cross-VM installation

- Docker containers behave like specialized VMs
- Dockerize the dependencies
- Push stable container images to docker hub
- Pull to VMs that collaborate in the docker swarm
- Set up the swarm and run a service stack

hub.docker.com/r/glemoine62/dias_py

docker hub | dias_py | Explore | Pricing | Sign In | Sign Up

Explore > glemoine62/dias_py

glemoine62/dias_py

Pulls 893

By glemoine62 • Updated 2 years ago

Container for running code on the DIAS

Container

Overview | Tags

Essential libraries for running GDAL based (python) scripts on DIAS

```
FROM thinkwhere/gdal-python:latest

LABEL maintainer="Guido Lemoine"\
      organisation="EC-JRC"\
      version="1.2"\
      release-date="2019-11-12"\
      description="DIAS python3 essentials"

WORKDIR /usr/src/app
```

Docker Pull Command

```
docker pull glemoine62/dias_py
```

Owner

glemoine62

European Commission

```
      description="DIAS python3 essentials"

WORKDIR /usr/src/app

# Update base container install
RUN apt-get update --fix-missing
RUN apt-get upgrade -y

RUN apt-get install -y libxml2-dev libxslt-dev gdal-bin sshfs vim
RUN apt-get -y autoremove

RUN pip install --upgrade pip

COPY requirements.txt /tmp/
RUN pip install --no-cache-dir -r /tmp/requirements.txt
```

glemoine62

with requirements.txt as:

```
lxml
requests
psycopg2-binary
numpy
pandas
gdal>=2.2.4
geojson
rasterio
rasterstats
boto3
ssh2-python
```

eouser@eosc1: ~

File Edit View Search Terminal Help

```
eouser@eosc1:~$ docker swarm init --advertise-addr 192.168.0.4
Swarm initialized: current node (ngzdofpex53fny95iafcasdjd) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-0xmvcx779z51xlhxrw6jeprwk2o8xhvskvbk2vgyeu04dtirjl-2rbsc3i10atpbkuxkua2r43gg 192.168.0.4:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

eouser@eosc1:~$ ssh eosc_w1 'docker swarm join --token SWMTKN-1-0xmvcx779z51xlhxrw6jeprwk2o8xhvskvbk2vgyeu04dtirjl-2rbsc3i10atpbkuxkua2r43gg 192.168
.0.4:2377'
This node joined a swarm as a worker.
eouser@eosc1:~$ ssh eosc_w2 'docker swarm join --token SWMTKN-1-0xmvcx779z51xlhxrw6jeprwk2o8xhvskvbk2vgyeu04dtirjl-2rbsc3i10atpbkuxkua2r43gg 192.168
.0.4:2377'
This node joined a swarm as a worker.
eouser@eosc1:~$ ssh eosc_w3 'docker swarm join --token SWMTKN-1-0xmvcx779z51xlhxrw6jeprwk2o8xhvskvbk2vgyeu04dtirjl-2rbsc3i10atpbkuxkua2r43gg 192.168
.0.4:2377'
This node joined a swarm as a worker.
eouser@eosc1:~$ ssh eosc_w4 'docker swarm join --token SWMTKN-1-0xmvcx779z51xlhxrw6jeprwk2o8xhvskvbk2vgyeu04dtirjl-2rbsc3i10atpbkuxkua2r43gg 192.168
.0.4:2377'
This node joined a swarm as a worker.
eouser@eosc1:~$ docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
ngzdofpex53fny95iafcasdjd *   eosc1      Ready     Active         Leader           20.10.2
cpg32g4lkrtrvahd9cfc9dorn     eosc-w1    Ready     Active                          20.10.2
3biq6nnl1mjx7jp1flabjslqp     eosc-w2    Ready     Active                          20.10.2
uz9je271uu12mnofzb8dypntb     eosc-w3    Ready     Active                          20.10.2
4ju7kda7upp5wd1lts7opcy63     eosc-w4    Ready     Active                          20.10.2
eouser@eosc1:~$
```

Parallel processing on DIAS, *continued*

- Docker Swarm is, by far, the simplest parallelization mechanism
- No specific programming needed, only configuration management
- Some issues with service stack termination
- Fine grained control with kubernetes (k8s), with docker containers
- Programmatic parallelization in python (multiprocessing, dask)
- In CbM, beware of database connections required by parallel tasks
- We use parallel processing in parcel and chip extraction
- We expect further parallel processing needs for more complex analytics
- We will consider use of GPUs as well

European Commission

Agenda

09:30 - 09:45	Welcome and short introduction into the jrc-cbm backend
09:45 - 10:30	**DIAS platform**: overview of resources and how they fit together
10:30 - 11:00	**The Spatial Database**: basics and support to parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	**CARD generation and parcel time series extraction**
11:45 - 12:15	**Supporting the Frontend**: RESTful and JHub server set up
12:15 - 12:30	Next steps and discussion

Agenda

09:30 - 09:45	Welcome and short introduction into the jrc-cbm backend
09:45 - 10:30	**DIAS platform**: overview of resources and how they fit together
10:30 - 11:00	**The Spatial Database**: basics and support to parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	**CARD generation and parcel time series extraction**
11:45 - 12:15	**Supporting the Frontend**: RESTful and JHub server set up
12:15 - 12:30	Next steps and discussion

Agenda

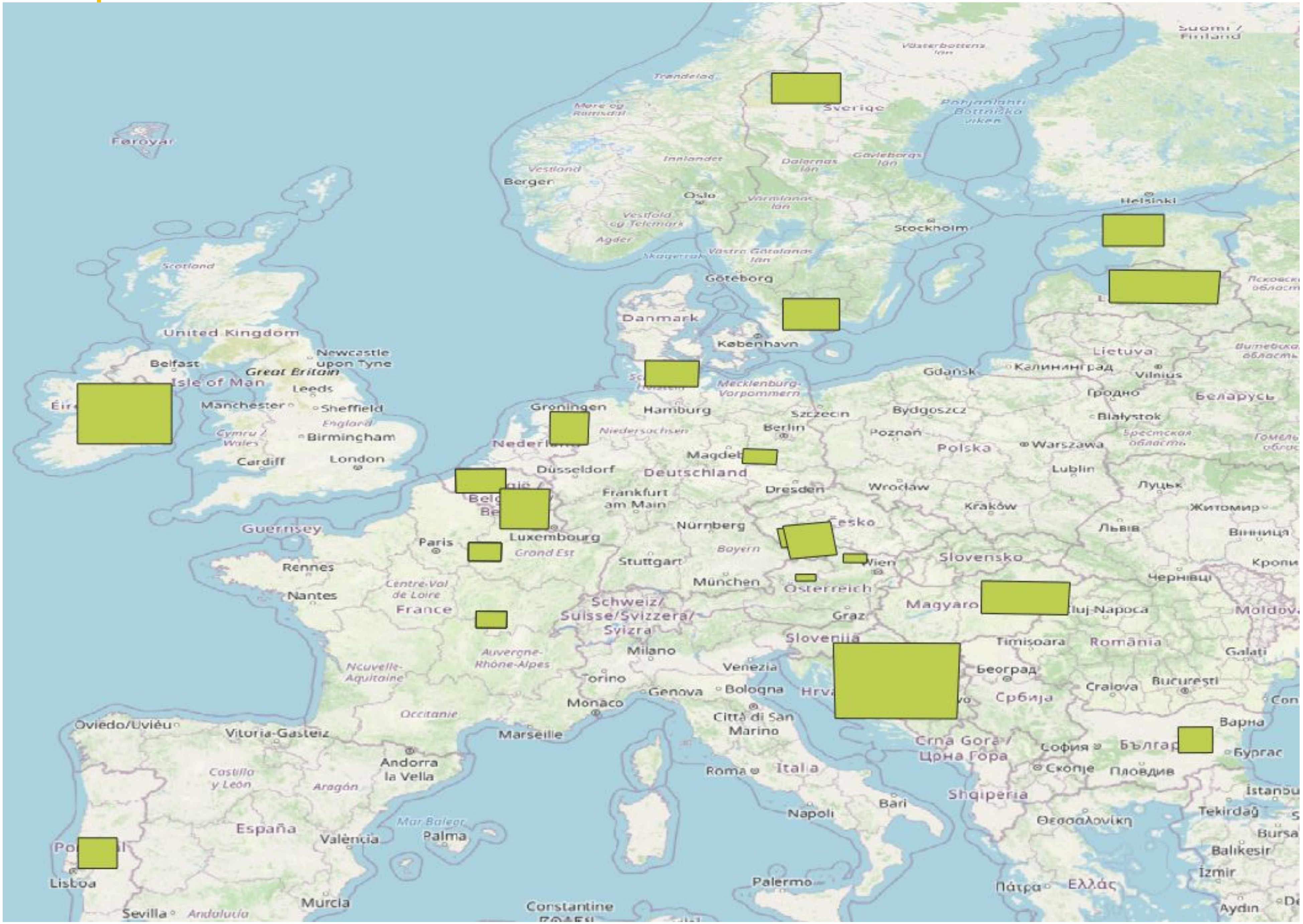
09:30 - 09:45	Welcome and short introduction into the jrc-cbm backend
09:45 - 10:30	**DIAS platform**: overview of resources and how they fit together
10:30 - 11:00	**The Spatial Database**: basics and support to parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	**CARD generation and parcel time series extraction**
11:45 - 12:15	**Supporting the Frontend**: RESTful and JHub server set up
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;
    - to provide server components and APIs for data access.
- DIAS instances offer a Processing as a Service (**PaaS**) solution for ARD
- We now know how to discover ARD and retrieve it from the S3 store
- We also know how to marshall and orchestrate compute resources
- The spatial database is used for storage of control data and reductions
- Extraction combines components to provide a meaningful backend function
- And feeds the frontend with data from the analytical CbM workflow

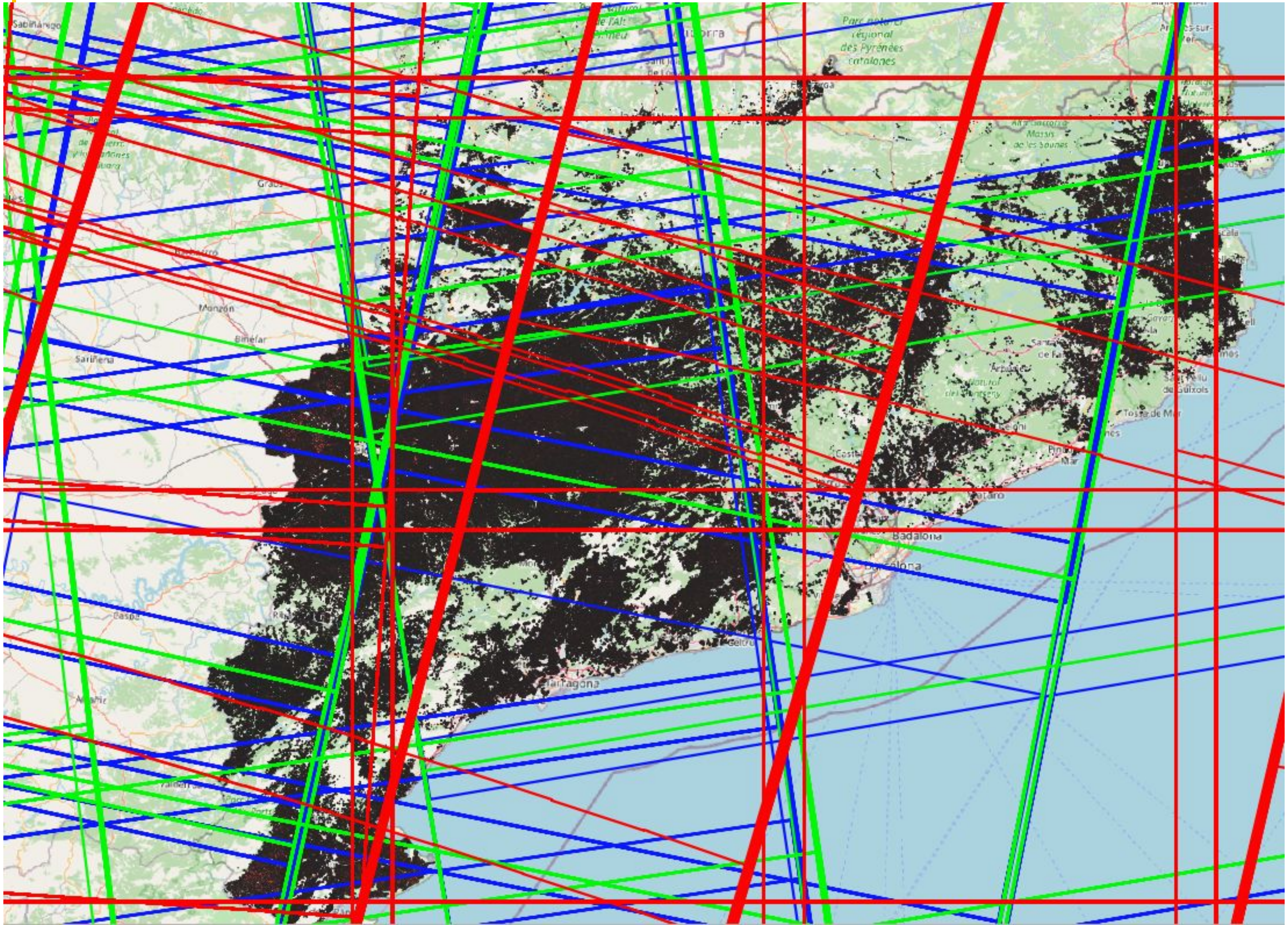
Sentinel ARD issues

- Data formats used: JPEG2000 (S-2), BEAM-DIMAP, (CO)GeoTIFF (S-1)
- For S-2: S3 key (/eodata path) points to (undocumented) sub-directory
- S-2: adjacent granules with 10% overlap, may be projected in straddling UTM
- This leads to data duplication, esp. for S-2 L2A (to be resolved in database)
- S-2A and S-2B till suffer from systematic pixel shift (esp. older data)
- S1 CARD-BS has one or more empty lines between (geocoded) frames
- Parcels with only NODATA are dropped, partial NODATA is not dropped
- S1 CARD-BS is not yet “terrain flattened” (work in progress)



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

- Extraction is 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 (μ, σ, min, max, p25, p50, p75) for the bands of the image
    - stores the results in the time series database tables
    - clears the local disk
- S2 bands: [B02, B03, B04, B08], [B5, B11], S1 bands: [VV, VH]. No indices!
- S2 SCL is extracted as histograms
- python scripts using psycopg2, rasterio, osgeo (gdal), pandas, numpy
- Recently refactored to use rasterized parcels and Numba acceleration

eouser@eosc1: ~/jrc-dias/scripts

File Edit View Search Terminal Help

```
eouser@eosc1:~/jrc-dias/scripts$ more db_config.json
{
    "database": {
        "connection": {
            "host": "IP Address",
            "dbname": "outreach",
            "dbuser": "DBUSER",
            "dbpasswd": "DBPASSWORD",
            "port": 5432
        },
        "tables": {
            "aoi_table": "aois",
            "parcel_table": "ee_2019",
            "catalog_table": "dias_catalogue",
            "results_table": "ee_signatures"
        },
        "args": {
            "aoi_field": "name",
            "name": "ee_2019",
            "startdate": "2018-10-01",
            "enddate": "2020-01-01"
        }
    }
}
eouser@eosc1:~/jrc-dias/scripts$ more docker-compose_s210.yml
version: '3.5'
services:
  vector_extractor:
    image: glemoine/dias_numba_py:latest
    volumes:
            - /home/eouser/jrc-dias/scripts:/usr/src/app
            - /eodata:/eodata
            - /1/DIAS:/1/DIAS
    networks:
      - overnet
    deploy:
      replicas: 4
    command: python factoredMountedExtraction.py s2 10

networks:
  overnet:
eouser@eosc1:~/jrc-dias/scripts$ docker stack deploy -c docker-compose_s210.yml s2stack
```

Agenda

09:30 - 09:45	Welcome and short introduction into the jrc-cbm backend
09:45 - 10:30	**DIAS platform**: overview of resources and how they fit together
10:30 - 11:00	**The Spatial Database**: basics and support to parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	**CARD generation and parcel time series extraction**
11:45 - 12:15	**Supporting the Frontend**: RESTful and JHub server set up
12:15 - 12:30	Next steps and discussion

Agenda

09:30 - 09:45	Welcome and short introduction into the jrc-cbm backend
09:45 - 10:30	**DIAS platform**: overview of resources and how they fit together
10:30 - 11:00	**The Spatial Database**: basics and support to parcel extraction
11:00 - 11:15	Break
11:15 - 11:45	**CARD generation and parcel time series extraction**
11:45 - 12:15	**Supporting the Frontend**: RESTful and JHub server set up
12:15 - 12:30	Next steps and discussion

jrc-cbm backend take home messages

- jrc-cbm provides a **modular** approach to the implementation of CbM workflow
- **cloud-centric** in design, particularly for the **backend**
- the **backend** benefits from DIAS IaaS and S3 Sentinel data store
- programming needs related to component collation
- module choice based on "**best in class**" open source, open standards
- focus on functional needs of CbM data reduction and access
- RESTful and Notebooks provide hooks into the Frontend
- all **code open sourced**, to be maintained on github, as PyPi package
- ready for core tasks, open for collaborative build out

Next steps

- Some Outreach MS are also DIAS onboarders: pip install cbm 😀
- Significant amounts of CARD data already available (CREODIAS, WEkEO)
- The core JRC backend tasks for Outreach are progressing (some delays).
- 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
ferdinando.urbano@ec.europa.eu
konstantinos.anastasakis@ext.ec.europa.eu

CbM on DIAS:
The Database Component

On-line training for Outreach, 30 June 2021

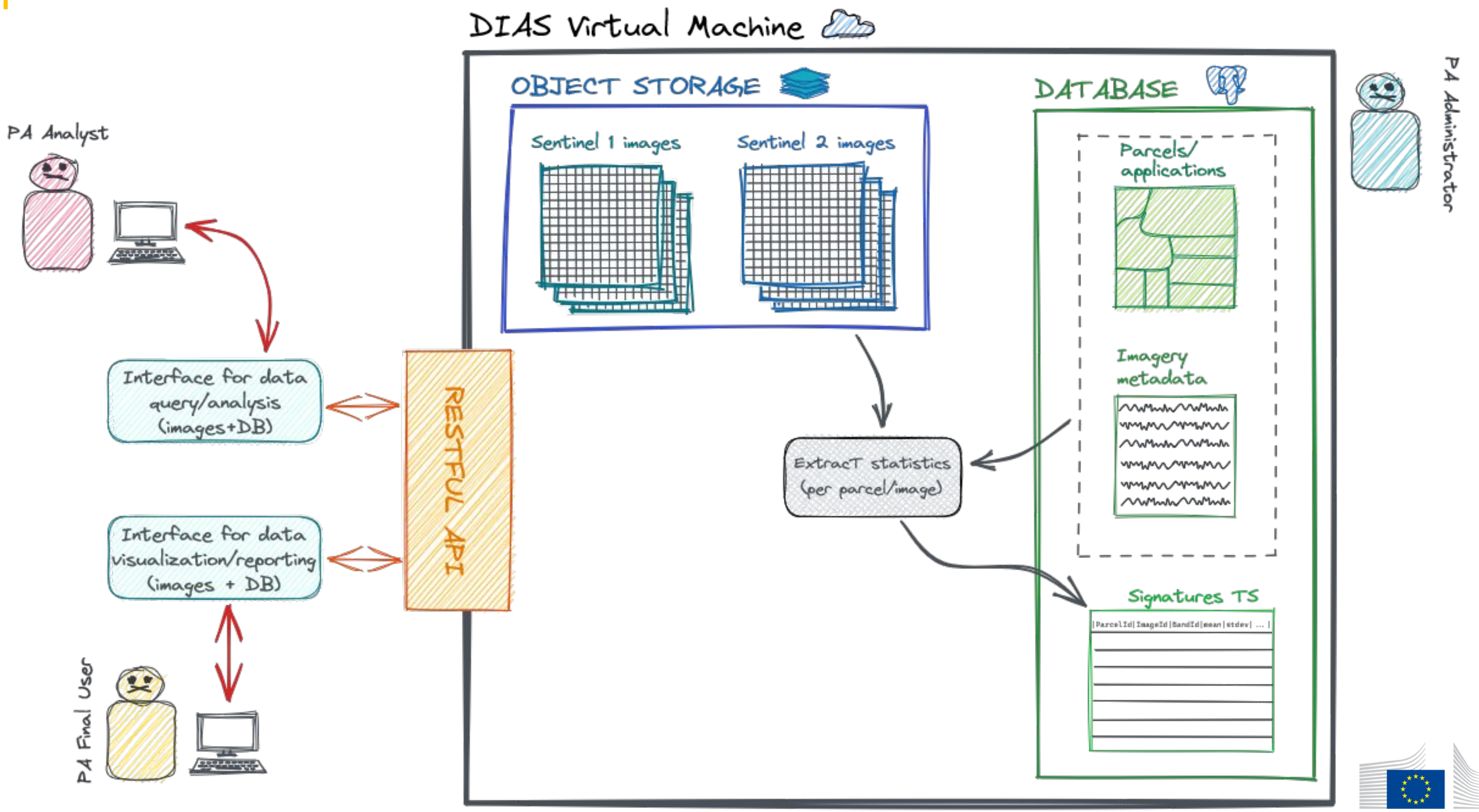
JRC D5 – GTCAP Team

Summary

1. The role of the database in the Outreach CbM system
2. Cut a long story short: introduction to (spatial) relational database
3. The software platform: PostgreSQL and PostGIS
4. What is SQL, the database language in a nutshell
5. Outreach database content
6. Accessing the database
7. Examples of client applications (demos)

European Commission

The database in the JRC CbM architecture



Database for CbM: the context

- Large set of statistics are extracted from Sentinel images for GSAA parcel sets (reduction)
- Statistics are used by analysts to verify compliance
- Critical importance of data security and consistency
- Relevance of performance for continuous monitoring
- Multi-user and distributed environment

- **Challenges in data use and management**

- The DB is the tool for storing and handling the data involved in the process
- The Outreach DB is hosted on the DIAS space close to Sentinel images

What is a (spatial) relational database?

- A **database** (DB) is a set of data organized in such a way as to facilitate its management, use and updating, stored in a computer
- A **relational database** is a DB with a logical model that structures data as relationships (*tables*) that are linked together
  - A table is made of columns and rows and is declaratively created with a structure (column have *data types* with values and properties)
  - The number of columns is fixed, the number of rows is variable
  - Each row is identified by the value of one or more columns (*primary key*)
  - Tables can be formally linked to one another (relation) using *foreign keys* (the value in a specific record column must come from another table)
  - Tables are organized in *schemas*, that are analogous to folders
  - The data is manipulated with SQL language
- A **spatial database** is a DB that can manage the spatial attribute of an object

Main features of a relational database

- Storage capacity
- Retrieval performance
- Concurrency control
- Permission policy
- Data formalization
- Data integrity controls
- Relational environment (data models)
- Data consistency (normalization)
- Industrial standard
- Remote access
- Server/client structure (modularity)
- Prevent data duplication
- Data preservation
- Easy automation of processes
- Backup/recovery functionalities
- Spatio-temporal data types
- Mature technology
- Cost effective

What is SQL?

- **SQL** (Structured Query Language) is the universally used language in relational databases
- It is a simple declarative language with limited number of commands
- SQL is used to retrieve data from a database (“queries”) and create database objects
- SQL is highly standardized and can be run from any database client

European Commission

Why PostgreSQL and PostGIS

- Full support of spatial data types
- Great spatial and non-spatial tools for data management and analysis
- Stable and secure
- Good documentation
- Many procedural languages
- Consolidated project with long history
- Fast development
- Natively supported by many software
- Collaborative and active community
- Multi-platform

- Possibility of commercial support
- Used by many large companies

In addition, it is **open source** and as such:

- No vendor-lock policies
- No limitations in its use
- No costs for licenses
- Use of standards
- Interoperability with other tools
- Easy to replicate by MS

Performance optimization: basic

- Rows in signature table:
  number of parcels * number of images * number of bands
- 500,000 parcels, 73 Sentinel images, 7 bands: 260,000,000 records (20 GB)
- Issues with data retrieval: optimization
- **Indexes:** better performance (but more disk space and slower upload)

- Extract all sigs for a parcel/band:
  - With indexes: 0.2 seconds
  - Without indexes: 2 seconds
- Extract all parcels for an image/band:
  - With indexes: 2 seconds
  - Without indexes: 2 minutes

European Commission

Performance optimization: advanced

If the performance achieved with indexes are not satisfactory, other actions are possible:

- Tune configuration parameters
- Partitioned tables
- Table clustering
- Increase hardware resources
- Multiple-Server Parallel Query Execution

PostgreSQL scalability

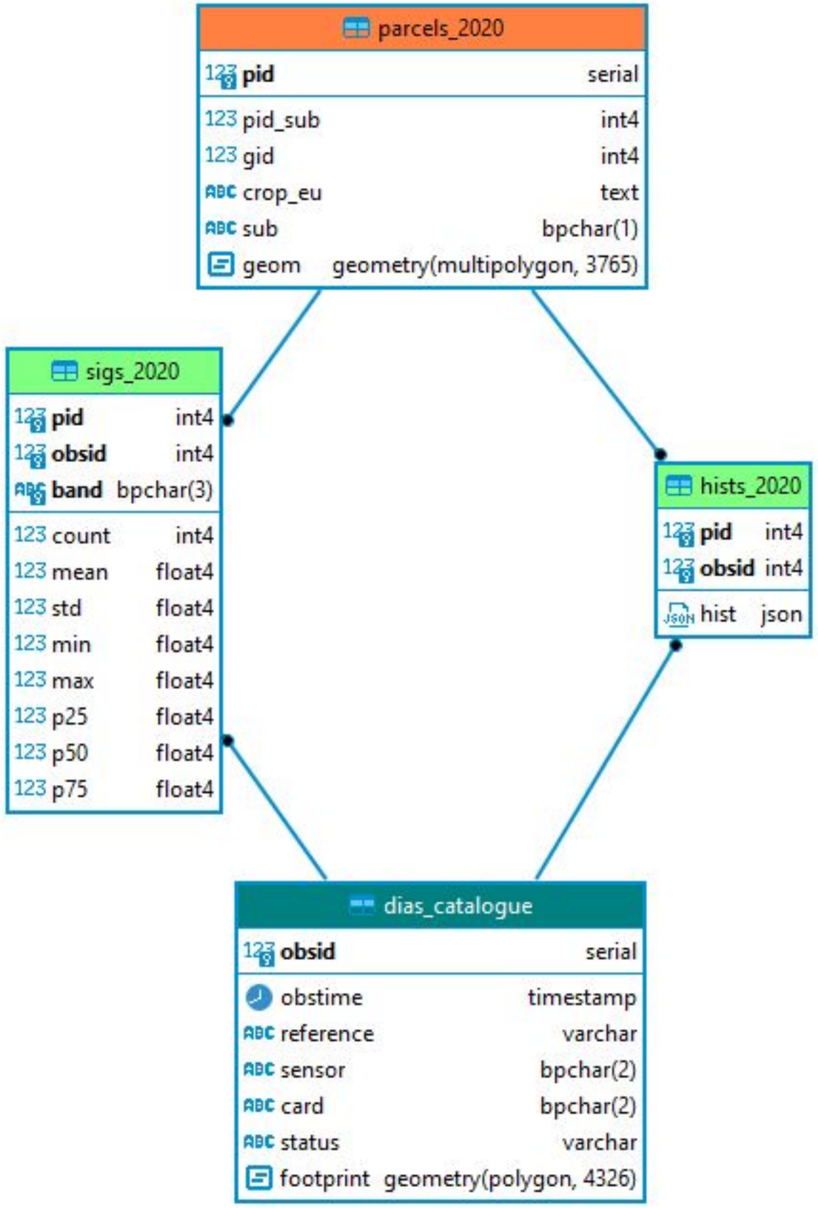
- For some countries, CbM data can be an order of magnitude larger than those tested in the Outreach DB: importance of scalability
- PostgreSQL can scale beyond running on a single server, exploiting cloud based infrastructures (database replication, database clustering, connection pooling)
- Many companies provide commercial support for advanced PostgreSQL high performance, multi-server solutions based on specific requirements
- Database are standard: easy to move to another platform if needed

Outreach DB content

- Outreach DB: store time series of Sentinel bands signature for each parcel

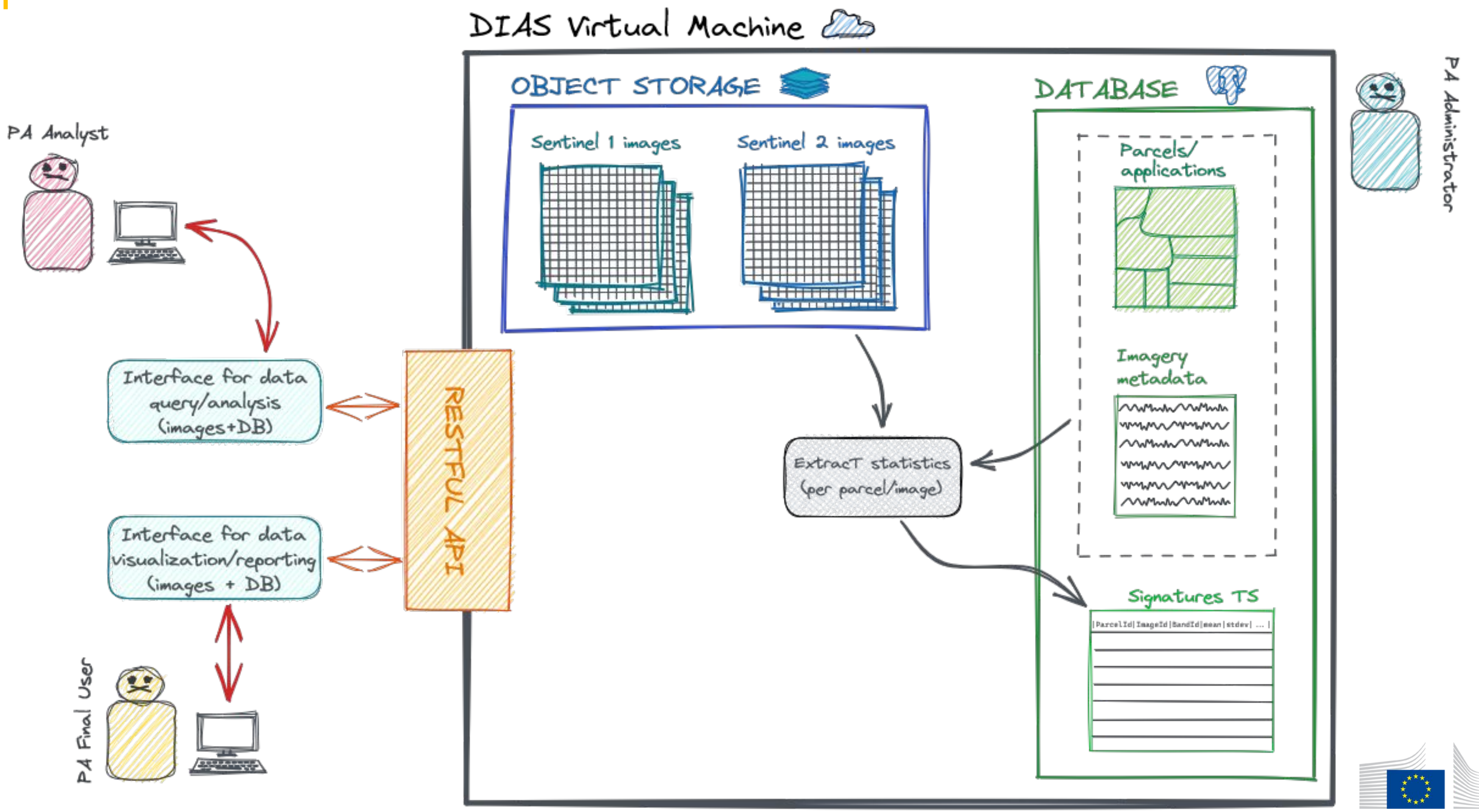
- **Images metadata table** (*public* schema)
- **Parcels table** (country schema)
- **Signatures table** (country schema)
- **Cloud flags histogram table** (country schema)

- Parcels, sigs and hists are **year-specific**
- There is one *dias_catalogue* table for all countries

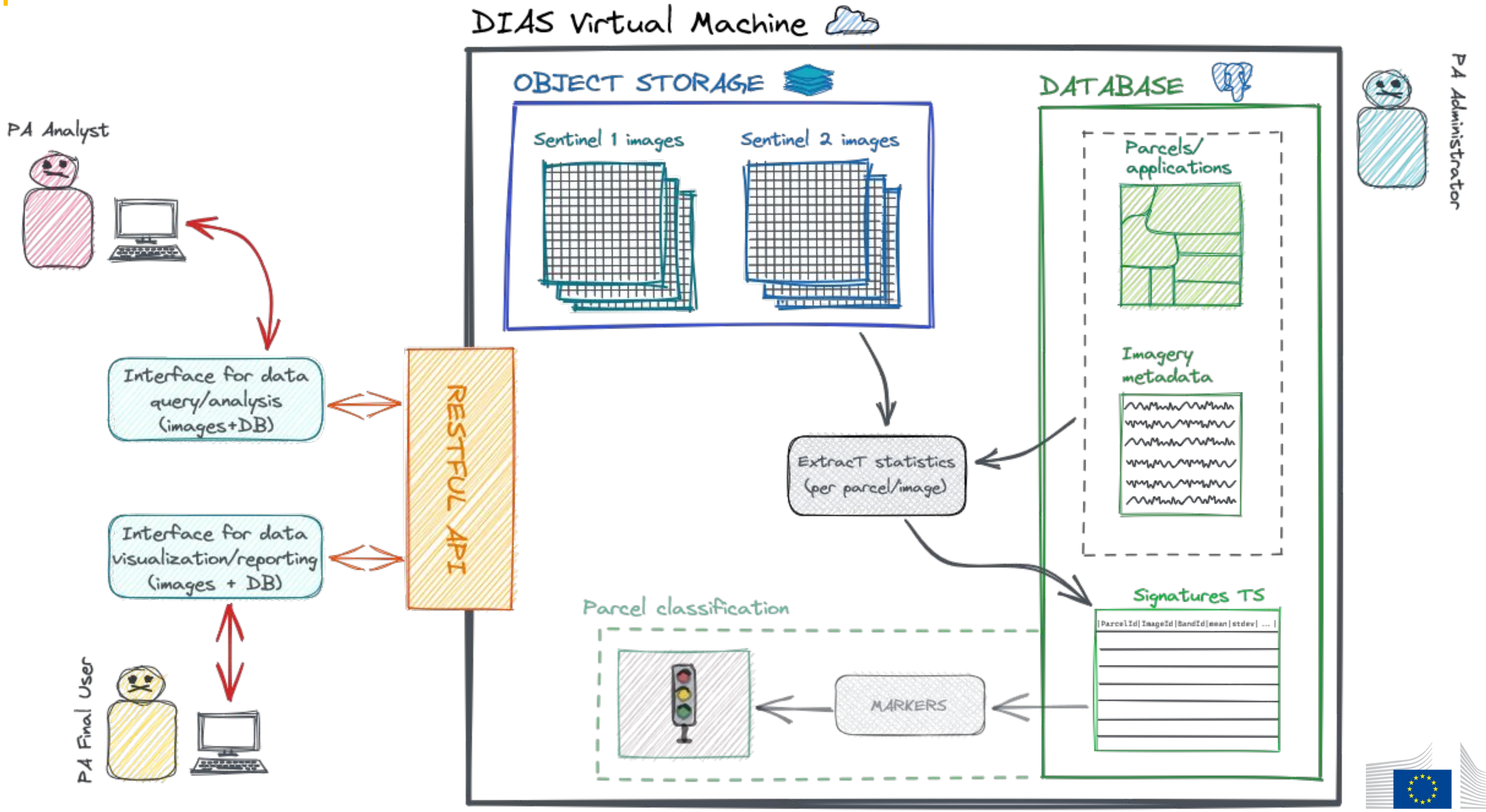


European Commission

Extending the DB in the Cbm system - 1



Extending the DB in the Cbm system - 2



Client/server architecture

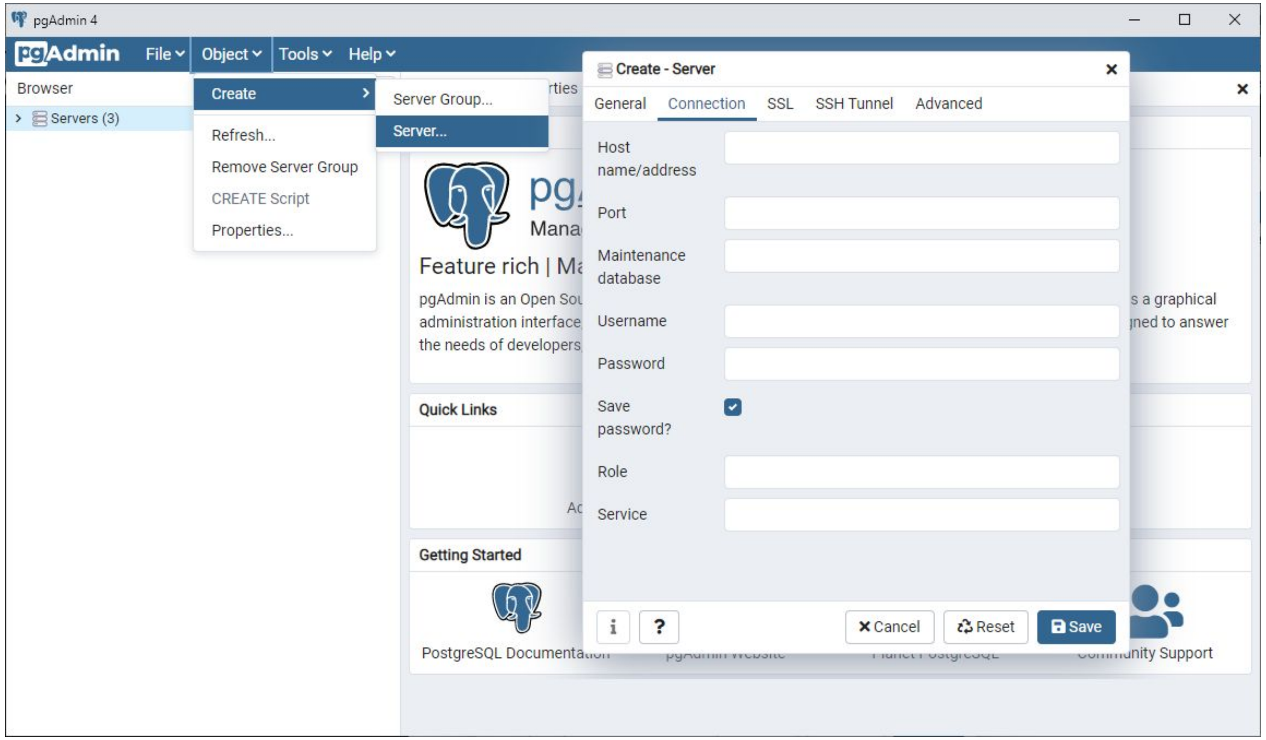
- The database architecture is based on a client-server structure
- The database server (PostgreSQL) is the back-end system of the database application and provides database functionality to client applications
- A database server can host many databases and a server can host many database servers
- The client is an interface through which a user makes a request to the server through SQL commands and converts the server's response into the form requested by the user
- Data management and storage layer is physically separated from data use

European Commission

Connection parameters

In order to remotely access a DB from a client, 5 parameters are required:

- Server IP address
- Port
- (Database name)
- User name
- User password



The parameters to connect to the Outreach DB are provided on requests, but in general Outreach DB access is granted through an intermediate layer.

Access though an intermediate layer

- It ensures performance and security by preventing poorly designed resource-intensive queries
- It facilitates access to basic users with no knowledge of SQL who can be guided by predefined queries offered as a simple graphical interface where only defined parameters need to be defined
- This intermediate layer is implemented in the Outreach project using a RESTful API

Permission policy

- Restrict access and possible operations on the data according to the different types of users
- PostgreSQL manages access permissions to the database through ROLES
- A role is an entity that can own objects and have privileges on the database
- Users can be grouped to facilitate privilege management
- Each user is assigned a password together with the role
- Access to the server can be restricted to certain IP addresses

Data export/import from/to the DB

- Through a GUI (e.g. PgAdmin for tables, QGIS or OGR2OGR for shapefile)
- Using COPY - /COPY commands from command line (as CSV)
- With a backup and restore of the DB (pg_dump, pg_restore)

Data import/export is easy, but table size can be an issue! (millions of rows)

Database clients

- **PgAdmin**: the native and most common GUI for querying data and managing PostgreSQL
- **Psql**: the interactive terminal for working with PostgreSQL
- **QGIS**: desktop GIS tool perfectly integrated with R
- **PhpPgAdmin**: a simplified version of PgAdmin available as web tool (no installation is required by the user)
- **R**: a language and environment for statistical computing and graphics
- **Python**: a complete and powerful programming language for data processing
- **RESTful API**: an architectural style for an application program interface (API) that uses HTTP requests to access and use data

LIVE DEMO

PgAdmin DEMO

European Commission

QGIS DEMO

Q&A

guido.lemoine@ec.europa.eu
ferdinando.urbano@ec.europa.eu
konstantinos.anastasakis@ext.ec.europa.eu

Documentation on database available in the JRC CbM GitHUB repository (soon)

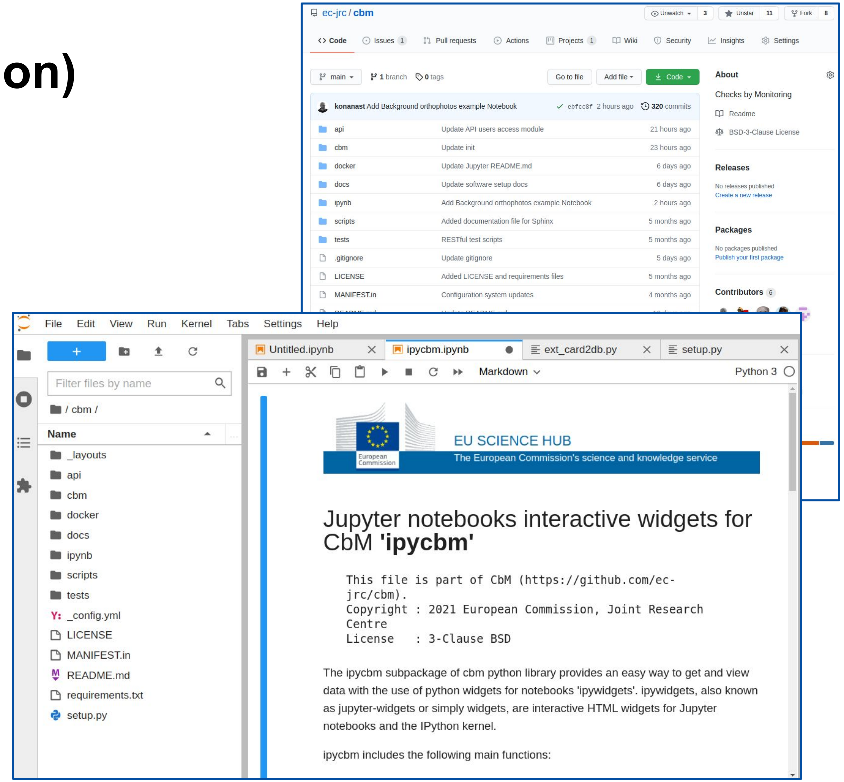
CbM on DIAS: Server components

On-line training for Outreach, 30 June 2021

JRC D5 – GTCAP Team

CbM Server components

- CbM git repository (quick introduction)
- Database deployment
- Jupyter Server
  - Single user
  - Multi user (Hub)
- RESTful API server setup
- CbM RESTful API use
  - Python scripts
  - Notebook widgets
- Jupyter Notebooks examples



CbM git repository

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

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
```

Technical issues page on github
https://github.com/ec-jrc/cbm/issues

Docker images available on Dockerhub:
https://hub.docker.com/u/gtcap

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.

Server deployment

PostgreSQL

PostGIS

Spatial PostgreSQL

jupyter

jupyterhub

voilà

Flask

Meinheld

gunicorn

Database Server

- PostgreSQL is a powerful, open source object-relational database system

Deploy a Postgres database server wit PostGIS: extension

```
docker run --name cbm_db -d --restart always -v database:/var/lib/postgresql --shm-size=2gb -p 5432:5432 -e POSTGRES_USER=postgres -e POSTGRES_PASS=mydiaspassword kartoza/postgis
```

-> list docker containers: docker ps -a:

```
docker ps -a
CONTAINER ID        IMAGE              COMMAND                  CREATED           STATUS            PORTS                    NAMES
  75fc1f296c79        kartoza/postgis  "docker-entrypoint.s…"  9 seconds ago     Up 7 seconds      0.0.0.0:5432->5432/tcp   cbm_db
```

-> Install postgresql client tools

```
sudo apt-get install postgresql-client-common postgresql-client-10
```

```
psql -h localhost -d postgres -U postgres
```

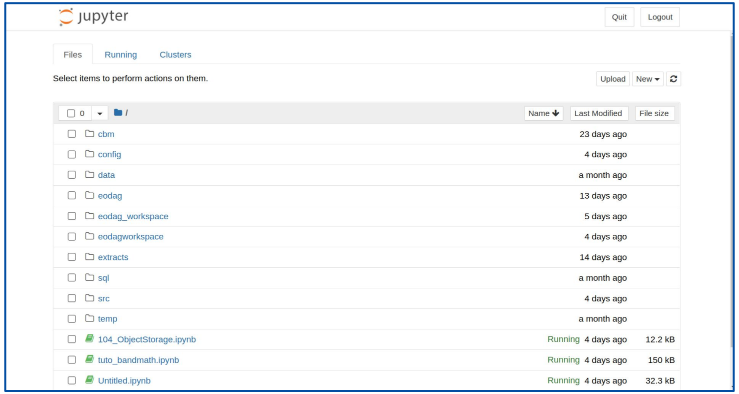
```
psql (10.12 (Ubuntu 10.12-0ubuntu0.18.04.1), server 10.7 (Debian 10.7-1.pgdg90+1))
Type "help" for help.

  postgres=#
```

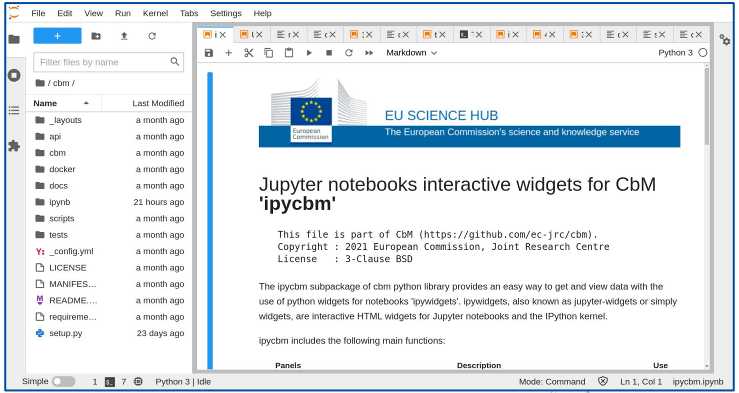
Jupyter Server

What is Jupyter;

Jupyter is a web-based interactive development environment that allows users to create and share codes, equations, visualisations, as well as text. There are two main interfaces:

- Jupyter Tree view: the first generation of Jupyter interface, a simplified interface with the basic functionalities for running Jupyter Notebooks ([http://hostname/**tree**](http://hostname/tree))
- JupyterLab the next generation of the Jupyter server interface, with the ability to configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular with plugins that add new features. ([http://hostname/**lab**](http://hostname/lab))



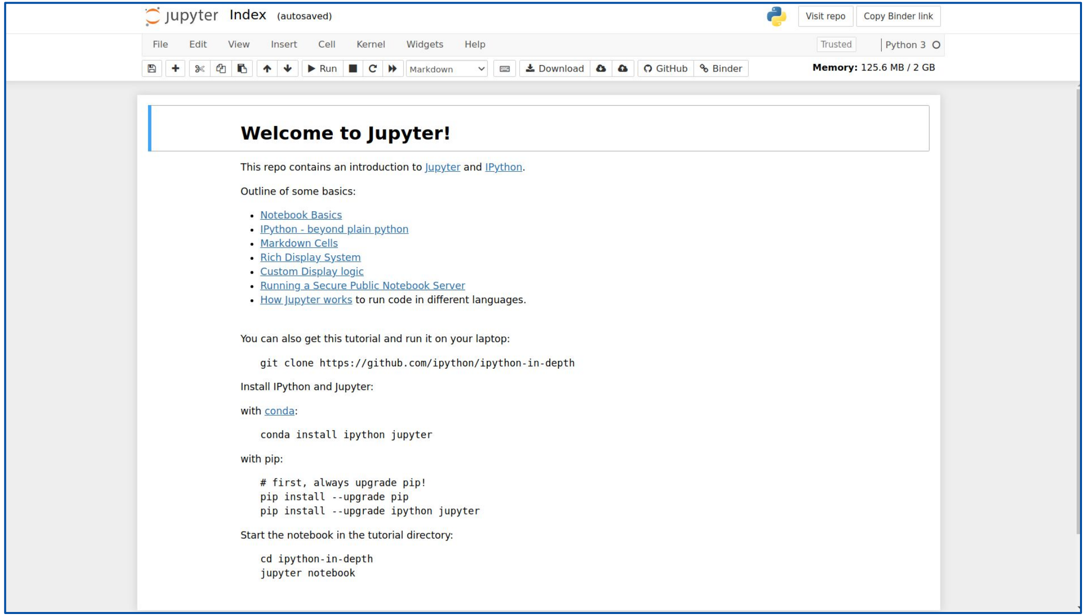


European Commission

Jupyter Notebooks

The Jupyter Notebook

- The Jupyter Notebooks are documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.



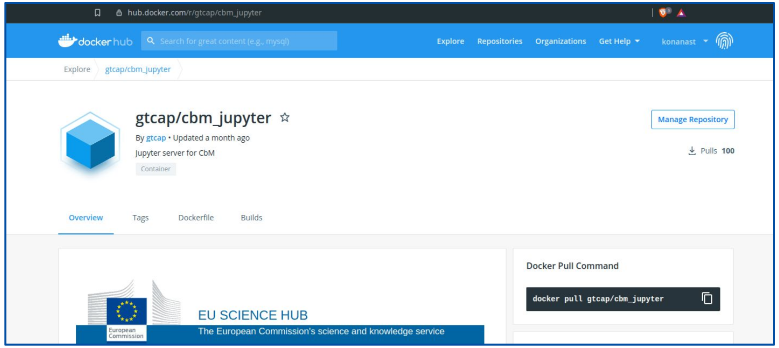
Get started at https://jupyter.org/try

Deploy a Jupyter server

Pre-requisites:

- A server running Ubuntu >18.04 where you have root access.
- At least 1.5GB of RAM on your server.
- Ability to ssh into the server & run commands from the prompt.
- A IP address where the server can be reached from the browsers of your target audience.
- Docker (for CbM Jupyter server)

CbM Jupyter docker image:
https://hub.docker.com/r/gtcap/cbm_jupyter



More info at https://jrc-cbm.readthedocs.io/en/latest/setup_software.html#jupyter-server

European Commission

Deploy a Jupyter server

With **no** shared folder "bindmount", (your files within the container will be deleted if you stop the container):

docker run -p 8888:8888 gtcap/cbm_jupyter

With a shared folder "bindmount", (your files will not be deleted if you stop the container):

- Navigate to the folder you want to bindmount to the container, e.g. the home directory (cd ~/):

- docker run -it --privileged=true --user root -e NB_USER="$USER" -e NB_UID="$UID" -e NB_GID="$UID" /
-p 8888:8888 -v "$PWD":/home/"$USER" --name=jupyter4cbm gtcap/cbm_jupyter

Terminal output :

```
[I 08:51:48.705 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 08:51:48.708 NotebookApp]

   To access the notebook, open this file in a browser:
       file:///home/jovyan/.local/share/jupyter/runtime/nbserver-8-open.html
   Or copy and paste one of these URLs:
       http://abcd12345678:8888/?token=abcd12345678
    or http://127.0.0.1:8888/?token=abcd12345678
```

- Access the Jupyter server on port 8888 (or any other port you set) on your VM's public ip (EIP) or your local ip if you have set port forwarding e.g.: **0.0.0.0:8888**
- Copy the token from the command line and add it to the web interface.

JupyterHub

What is JupyterHub;

- JupyterHub brings the power of notebooks to groups of users. It makes it possible to serve a pre-configured data science environment multiple users. It is customizable and scalable, and is suitable for small and large teams, academic courses, and large-scale infrastructure.

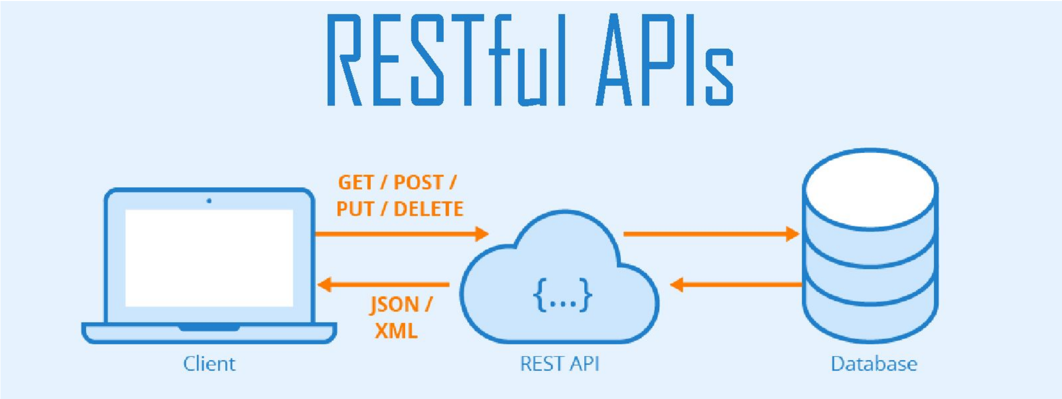
Key features

- Customizable **- Flexible** **- Scalable** **- Portable**

Experience required to Deploy a JupyterHub server:

- Cloud infrastructure management
- Docker and/or Kubernetes
  - Helm if using Kubernetes to configure and control the packaged JupyterHub installation
- Understanding Jupyter Server structure and operation
- Linux terminal interface usage
- Linux user management

RESTful APIs



API can be described as a mediator between the users or clients and the resources or web services they want to get. A RESTful API is an architectural style for an application program interface (API) that uses HTTP requests to access and use data. That data can be used to GET, PUT, POST and DELETE data types, which refers to the reading, updating, creating and deleting of operations concerning resources.

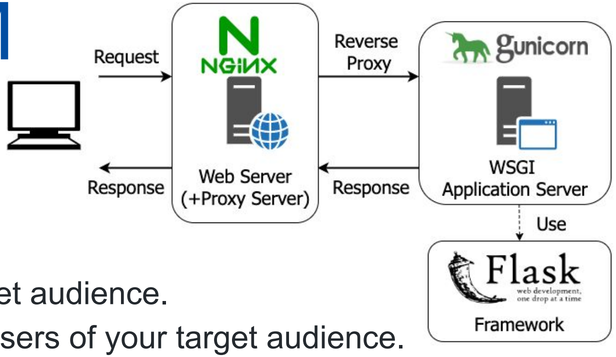
Disadvantages:

- **Takes time to develop***
- **Server side processing***

Advantages:

- **Portability**
- **Security**
- **Maintainability**
- **Performance**

Deploy a RESTful API for CbM



Pre-requisites:

- A server running Ubuntu >18.04 where you have root access.
- Ability to ssh into the server & run commands from the prompt.
- A IP address where the server can be reached from the browsers of your target audience.
- Docker installed.A IP address where the server can be reached from the browsers of your target audience.

JRC provides the code to deploy a RESTful API for CbM (https://github.com/ec-jrc/cbm)

Documentation at: https://jrc-cbm.readthedocs.io/en/latest/setup_build_api.html

CbM RESTful API use:

- **Flask**: a micro web framework written in Python.
- **Gunicorn**: a Python Web Server Gateway Interface (WSGI) HTTP server.
- **Meinheld**: a high-performance WSGI-compliant web server

Future implementations: FastAPI, Supervisord, nginx

Deploy a RESTful API for CbM

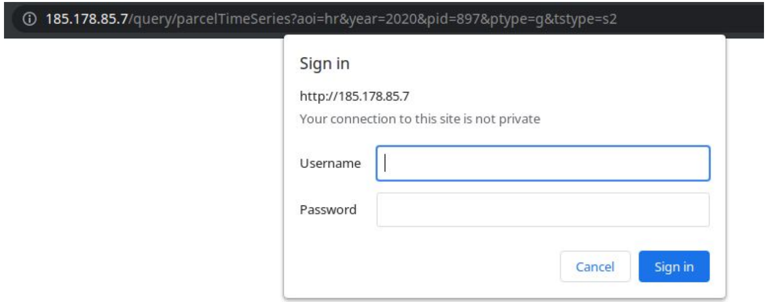
1. Clone the cbm repository: **git clone https://github.com/ec-jrc/cbm.git**
2. Navigate to the api folder: **cd cbm/api**
3. Add an API user:
    a. python3 scripts/users.py add username password dataset
    b.
    ```
    from scripts import users # Import the users module
    # Create a new user with:
    users.add('username', 'password', 'dataset')
    ```
4. Set database connection settings - config/main.json
5. Add available option (optional) - options.json
6. Deploy the RESTful API docker container

docker run -it --name api -v "$PWD":/app -p 80:80 gtcap/cbm_api

Documentation at https://jrc-cbm.readthedocs.io/en/latest/setup_build_api.html

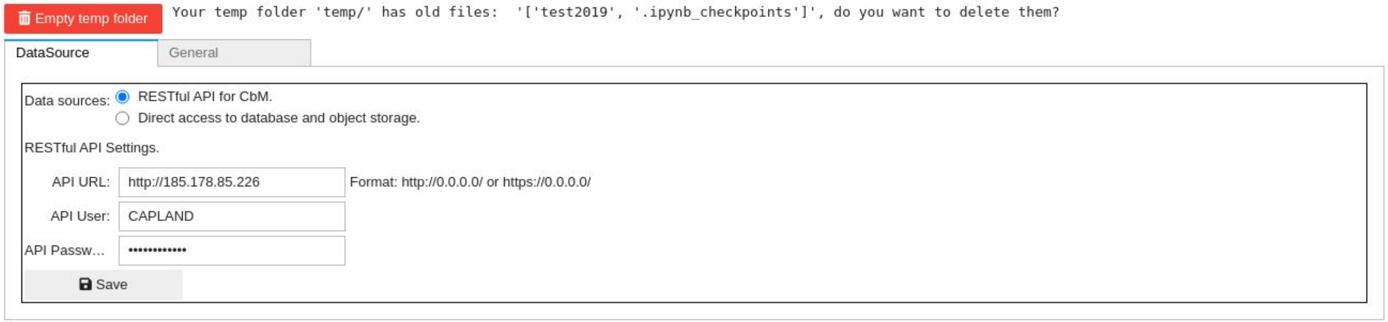
Access the RESTful API

- From the web browser



- Interactively in a jupyter notebook with the cbm python package:

```
[ ]: from cbm import ipycbm
     ipycbm.config()
```



Access the RESTful API

```
[10]: import sys
      import json

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

      # Define your credentials here
      host = 'http://185.178.85.7'
      username = ''
      password = ''

      # Get the parcel information
      locurl = """{}/query/parcelById?aoi={}&year={}&pid={}&ptype={}&withGeometry=True"""

      # set the query parameters
      aoi = 'hr'
      year = '2020'
      ptype = 'g'
      pid = '897'
      tstype = 's2'

      # Parse the response with the standard json module
      response = requests.get(locurl.format(host, aoi, year, pid, ptype), auth = (username, password))

      parcel = json.loads(response.content)
      parcel
```


```
[10]: {'ogc_fid': [897],
       'cropname': ['karst pasture'],
       'cropcode': [1702.0],
       'srid': [3765],
       'geom': ['{"type":"MultiPolygon","crs":{"type":"name","properties":{"name":"EPSG:3765"}},"coordinates":
      [[[[417380.266,4877830.79],[417337.745,4877865.978],[417344.647,4877871.123],[417456.891,4878001.521],[417
```

RESTful USE

Get parcel information:

http://185.178.85.7/query/parcelById?aoi=ms&year=2020&pid=1234&withGeometry=True

Not secure | 185.178.85.7/query/parcelById?aoi=hr&year=2020&pid=324&ptype=g&withGeometry=True

{"ogc_fid": [324], "cropname": ["karst pasture"], "cropcode": [1702.0], "srid": [3765], "geom": ["{\"type\":\"MultiPolygon\",\"crs\":{\"type\":\"name\",\"properties\":{\"name\":\"EPSG:3765\"}},\"coordinates\":[[[[421224.59,4882801.222],[421202.728,4882773.123],[421167.328,4882813.117],[421130.577,4882854.192],[421129.045,4882873.401],[421139.485,4882880.788],[421146.95,4882879.834],[421150.539,4882878.055],[421159.507,4882869.963],[421185.558,4882839.787],[421224.59,4882801.222]]]]}"], "area": [3770.727024015728], "clon": [15.515777062644197], "clat": [44.08121398033482]}


```
[2]: import cbm
     cbm.get.parcel_info.by_pid('nld', 2019, 575541, True)
```

```
[2]: {'ogc_fid': [575541],
      'cropname': ['Grasland, blijvend'],
      'cropcode': [265],
      'srid': [28992],
      'geom': ['{"type":"MultiPolygon","crs":{"type":"name","properties":{"name":"EPSG:28992"}},"coordinates":[[[[96576.009,417328.430199999],[96574.206300002,417366.527],[96572.206999998,417375.69],[96571.040800002,417391.0174],[96571.374000002,417396.6818],[96575.705699999,417414.175],[96578.3713,417432.334600002],[96688.661499999,417434.667],[96689.6611,417427.5031],[96690.993900001,417375.523400001],[96692.9932,417375.523400001],[96691.827,417434.167199999],[96695.159000002,417435.5],[96758.301,417435.9998],[96795.286499999,417436.832800001],[96800.2846,417436.499600001],[96803.616599999,417433.834],[96803.9498,417421.6721],[96804.9494,417376.189800002],[96806.282200001,417369.525800001],[96805.4492,417363.694699999],[96805.949,417336.372099999],[96804.283,417333.04],[96803.350099999,417333.450100001],[96782.7914,417332.207],[96738.1422,417331.374],[96589.866799999,417328.5418],[96576.009,417328.430199999]]]]}'],
      'area': [23898.22330816267],
      'clon': [4.542926127180021],
      'clat': [51.74211190705306]}
```

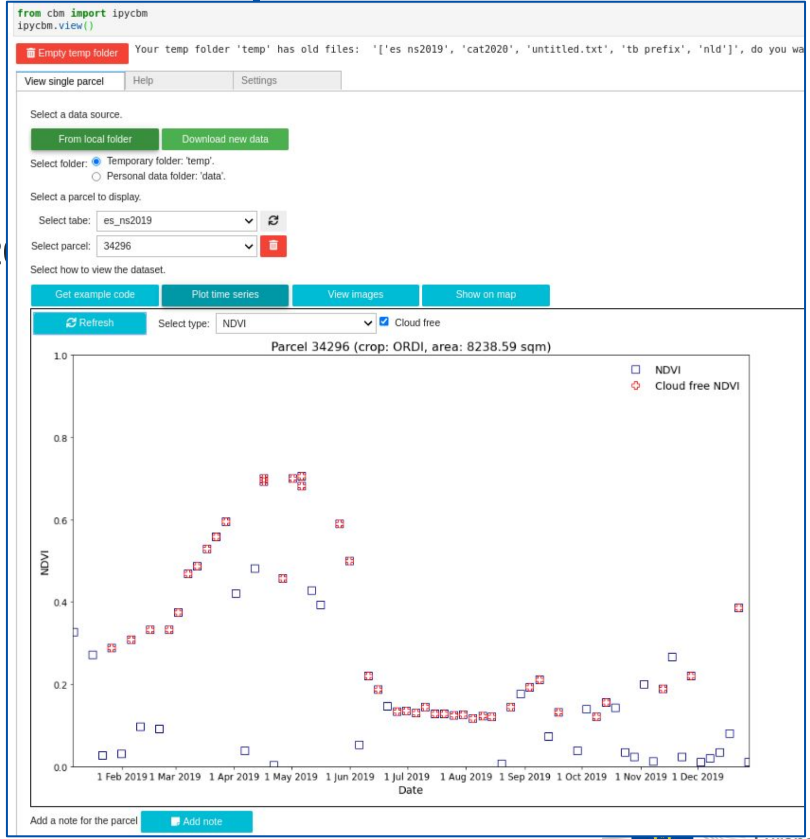
RESTful USE Notebook examples

Get parcel Time series:

http://185.178.85.7/query/parcelTimeSeries?aoi=ms&year=2(

Not secure | 185.178.85.7/query/parcelTimeSeries?aoi=hr&year=2020&pid=897&ptype=g&tstype=s2

{"date_part": [1570269031.024, 1570269031.024, 1570269031.024, 1570269031.024, 1570269031.024, 1570269031.024, 1570701029.025, 1570701029.025, 1570701029.025,
1570701029.025, 1570701029.025, 1570701029.025, 1571133031.024, 1571133031.024, 1571133031.024, 1571133031.024, 1571133031.024, 1571133031.024, 1571565029.024,
1571565029.024, 1571565029.024, 1571565029.024, 1571565029.024, 1571565029.024, 1571997061.024, 1571997061.024, 1571997061.024, 1571997061.024, 1571997061.024,
1571997061.024, 1572429039.024, 1572429039.024, 1572429039.024, 1572429039.024, 1572429039.024, 1572429039.024, 1572861121.024, 1572861121.024, 1572861121.024,
1572861121.024, 1572861121.024, 1572861121.024, 1573293089.024, 1573293089.024, 1573293089.024, 1573293089.024, 1573293089.024, 1573293089.024, 1573725171.024,
1573725171.024, 1573725171.024, 1573725171.024, 1573725171.024, 1573725171.024, 1574157129.024, 1574157129.024, 1574157129.024, 1574157129.024, 1574157129.024,
1574157129.024, 1574589211.024, 1574589211.024, 1574589211.024, 1574589211.024, 1574589211.024, 1574589211.024, 1575021169.024, 1575021169.024, 1575021169.024,
1575021169.024, 1575021169.024, 1575021169.024, 1575453241.024, 1575453241.024, 1575453241.024, 1575453241.024, 1575453241.024, 1575453241.024, 1575885189.024,
1575885189.024, 1575885189.024, 1575885189.024, 1575885189.024, 1575885189.024, 1576317241.024, 1576317241.024, 1576317241.024, 1576317241.024, 1576317241.024,
1576317241.024, 1576749199.024, 1576749199.024, 1576749199.024, 1576749199.024, 1576749199.024, 1576749199.024, 1577181261.024, 1577181261.024, 1577181261.024,
1577181261.024, 1577181261.024, 1577181261.024, 1577613199.024, 1577613199.024, 1577613199.024, 1577613199.024, 1577613199.024, 1577613199.024, 1578045241.024,
1578045241.024, 1578045241.024, 1578045241.024, 1578045241.024, 1578045241.024, 1578477189.024, 1578477189.024, 1578477189.024, 1578477189.024, 1578477189.024,
1578477189.024, 1578909231.024, 1578909231.024, 1578909231.024, 1578909231.024, 1578909231.024, 1578909231.024, 1579341149.024, 1579341149.024, 1579341149.024,
1579341149.024, 1579341149.024, 1579341149.024, 1579773191.024, 1579773191.024, 1579773191.024, 1579773191.024, 1579773191.024, 1579773191.024, 1580205109.024,
1580205109.024, 1580205109.024, 1580205109.024, 1580205109.024, 1580205109.024, 1580637141.024, 1580637141.024, 1580637141.024, 1580637141.024, 1580637141.024,
1580637141.024, 1581069059.024, 1581069059.024, 1581069059.024, 1581069059.024, 1581069059.024, 1581069059.024, 1581501081.024, 1581501081.024, 1581501081.024,
1581501081.024, 1581501081.024, 1581501081.024, 1581933029.024, 1581933029.024, 1581933029.024, 1581933029.024, 1581933029.024, 1581933029.024, 1582365021.024,
1582365021.024, 1582365021.024, 1582365021.024, 1582365021.024, 1582365021.024, 1582797029.024, 1582797029.024, 1582797029.024, 1582797029.024, 1582797029.024,
1582797029.024, 1583229031.024, 1583229031.024, 1583229031.024, 1583229031.024, 1583229031.024, 1583229031.024, 1583661029.024, 1583661029.024, 1583661029.024,
1583661029.024, 1583661029.024, 1583661029.024, 1584093031.024, 1584093031.024, 1584093031.024, 1584093031.024, 1584093031.024, 1584093031.024, 1584525029.024,
1584525029.024, 1584525029.024, 1584525029.024, 1584525029.024, 1584525029.024, 1584957031.024, 1584957031.024, 1584957031.024, 1584957031.024, 1584957031.024,



RESTful USE Notebook examples

[17]:

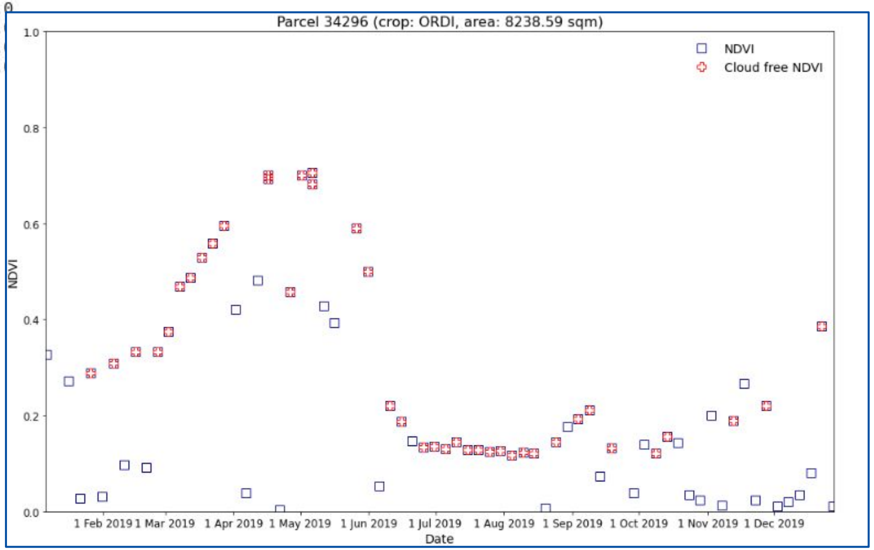
```
# Check response
if not parcel:
    print("Parcel query returned empty result")
    sys.exit()
elif not parcel.get(list(parcel.keys())[0]):
    print(f"No parcel found in {parcels} at location ({lon}, {lat})")
    sys.exit()

print(parcel)

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

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

print(tsurl.format(host, aoi, year, pid, tstype))
response = requests.get(tsurl.format(host, aoi, year, pid, tstype), auth = (username, password))

# Directly create a pandas DataFrame from the json response
# This should work even if the response is and empty dictionary
df = pd.read_json(response.content)
print(df)
# 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))
print(df['date_part'])

# Treat each band separately. Drop duplicate timestamps and rename the 'mean'
df4 = df[df['band']=='B04'][['date_part', 'mean']]
df4.drop_duplicates(['date_part'], inplace=True)
df4.rename(columns={'mean': 'B04'}, inplace=True)

df8 = df[df['band']=='B08'][['date_part', 'mean']]
df8.drop_duplicates(['date_part'], inplace=True)
df8.rename(columns={'mean': 'B08'}, 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['B08']-dff['B04'])/(dff['B08']+dff['B04'])

print(dff)

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

plt.figure()
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()
```

European Commission

RESTful USE Notebook examples

```
       date_part band  count       mean         std   min    p25    p50  \
0    1.570269e+09  B02     46  6522.5220  155.878130  6312  6404.0  6494.0
1    1.570269e+09  B03     46  6192.8696  125.061134  5900  6109.0  6188.0
2    1.570269e+09  B04     46  5841.6523  144.777070  5392  5760.0  5838.0
3    1.570269e+09  B05     13  6151.7690  203.443860  5672  6094.0  6154.0
4    1.570269e+09  B08     46  6366.1740  108.372000  6180  6292.0  6334.0
..            ...  ...    ...        ...         ...   ...     ...     ...
541  1.609149e+09  B03     46  9346.4350  196.348360  8760  9322.0  9420.0
542  1.609149e+09  B04     46  8971.6520   88.342520  8768  8908.0  8968.0
543  1.609149e+09  B05     13  8743.1540  112.023250  8612  8661.0  8691.
544  1.609149e+09  B08     46  9013.2180  203.285830  8592  8820.0  9080.
545  1.609149e+09  B11     13  3412.0000   67.654100  3298  3369.0  3402.

       p75   max        hist
0    6584.0  6908  {'9': 13}
1    6247.0  6528  {'9': 13}
2    5951.0  6132  {'9': 13}
3    6295.0  6424  {'9': 13}
4    6436.0  6660  {'9': 13}
..      ...   ...         ...
541  9456.0  9536  {'9': 13}
542  9032.0  9176  {'9': 13}
543  8829.0  8978  {'9': 13}
544  9160.0  9320  {'9': 13}
545  3473.0  3513  {'9': 13}

[546 rows x 11 columns]
0     2019-10-05 09:50:31.024
1     2019-10-05 09:50:31.024
2     2019-10-05 09:50:31.024
3     2019-10-05 09:50:31.024
4     2019-10-05 09:50:31.024
                ...
541   2020-12-28 09:54:21.024
542   2020-12-28 09:54:21.024
543   2020-12-28 09:54:21.024
544   2020-12-28 09:54:21.024
545   2020-12-28 09:54:21.024
Name: date_part, Length: 546, dtype: datetime64[ns]
Empty DataFrame
Columns: [date_part, B04, B08, SC, ndvi]
Index: []
```

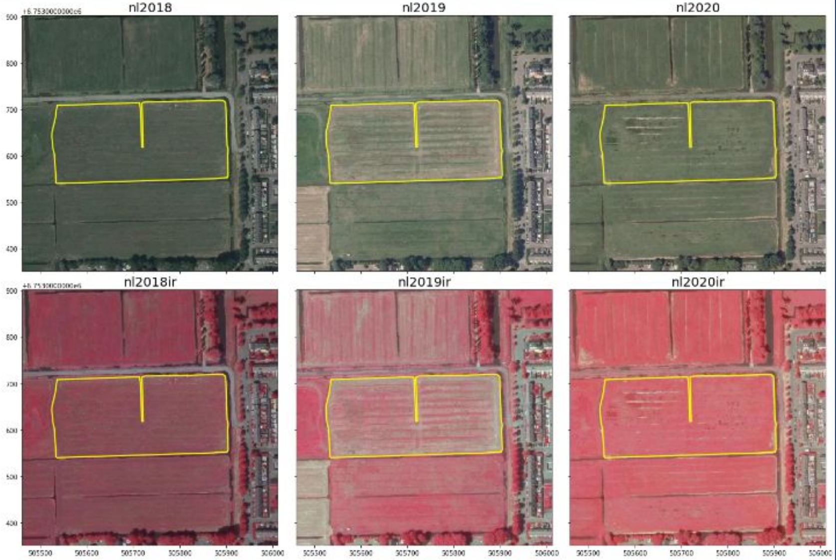


RESTful USE

Get parcel’s orthophotos

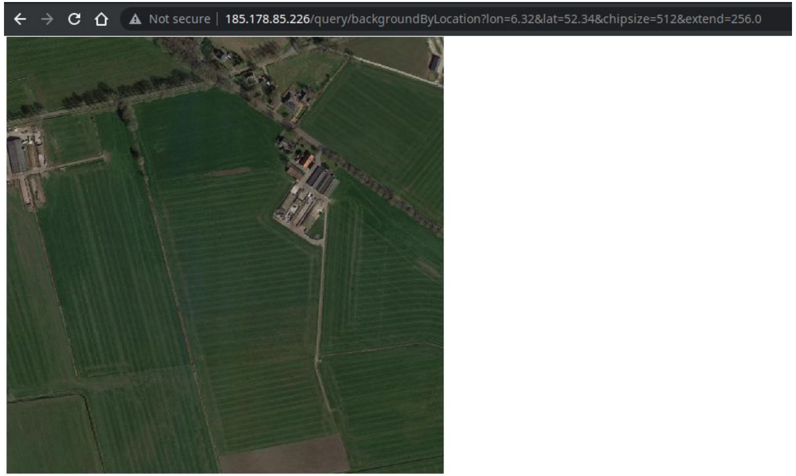
With cbm package:

```
[10]: import cbm
      cbm.show.background.by_pid('nld', 2019, 575541, 550, 550,
                                 ['nl2018', 'nl2019', 'nl2020',
                                  'nl2018ir', 'nl2019ir', 'nl2020ir'])
```



Within the browser:

http://hostname/query/backgroundByLocation?lon=6.32&lat=52.34



dump/62_74_7_192E6_32N52_34_512_256_0_Google/google.tif

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
- 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

European Commission

Q&A

guido.lemoine@ec.europa.eu
ferdinando.urbano@ec.europa.eu
konstantinos.anastasakis@ext.ec.europa.eu

European Commission