SINERGISE sentinelhub TOMTOM meteoblue Technical University of Munich SatCen

Global Earth Monitor

Deliverable 3.5

Front Office Services

The project has received funding from European Union's Horizon 2020 Research and Innovation Programme" under the Grant Agreement 101004112

Global Earth Monitor

PREPARATION SLIP			
	Name	**Organisation**	**Date**
Prepared by	Matej Batič	Sinergise	30.8.2022
Approved for submission by	Mariza Pertovt	Sinergise	02.9.2022
Reviewed by the EC		EC	28.01.2022
Comments addressed by	Matej Batič	Sinergise	15.08.2023
Approved for resubmission	Mariza Pertovt	Sinergise	28.08.2022

EXECUTIVE SUMMARY

This deliverable reports on Front Office Services exposing the new capabilities of Sentinel Hub services, and GEM framework building block – eo-learn and eo-grow.

The user-facing services are linked mostly to usage of GEM framework or parts of it with Python, be it via scripts (using eo-learn and eo-grow), or Jupyter notebooks (standalone/local or via cloud deployment on Euro Data Cube - EDC). Additionally, the report outlines the QGIS Sentinel-Hub plugin, as an extension of the popular GIS tool facilitating access to Sentinel-Hub datasets and Adjustable Data Cubes.

Several user-facing services, connected either to Sentinel Hub services, or meteoblue weather API, are not part of this deliverable, but are constantly being improved to facilitate the new/updated functionalities of the services. EO-Browser and Request-Builder should be mentioned explicitly, as they bring most of the Sentinel-Hub functionality to the user as web browser applications.

Contractual Delivery Date	31.08.2022
Actual Delivery Date	02.09.2022
Actual Delivery Date v2	31.08.2023
Type of delivery	Demonstrator
Dissemination level:	Public

Global Earth Monitor

Table of Contents

List of Figures

List of Code blocks

List of Abbreviations

API	Application Programming Interface
ARD	Analysis Ready Data
AWS	Amazon Web Services
BYOC	Bring Your Own Cloud-optimised data
CDSE	Copernicus Data Space Ecosystem
CLI	Command-Line Interface
DIAS	Data and Information Access Services
EDC	Euro Data Cube
EO	Earth Observation
GEM	Global Earth Monitor
GIS	Geographic Information System
ML	Machine learning
NDVI	Normalized Density Vegetation Index

1 Introduction

The main building blocks of the architecture, shown in Figure 1, are presented in detail in the deliverable D3.2 System Requirements and Design.

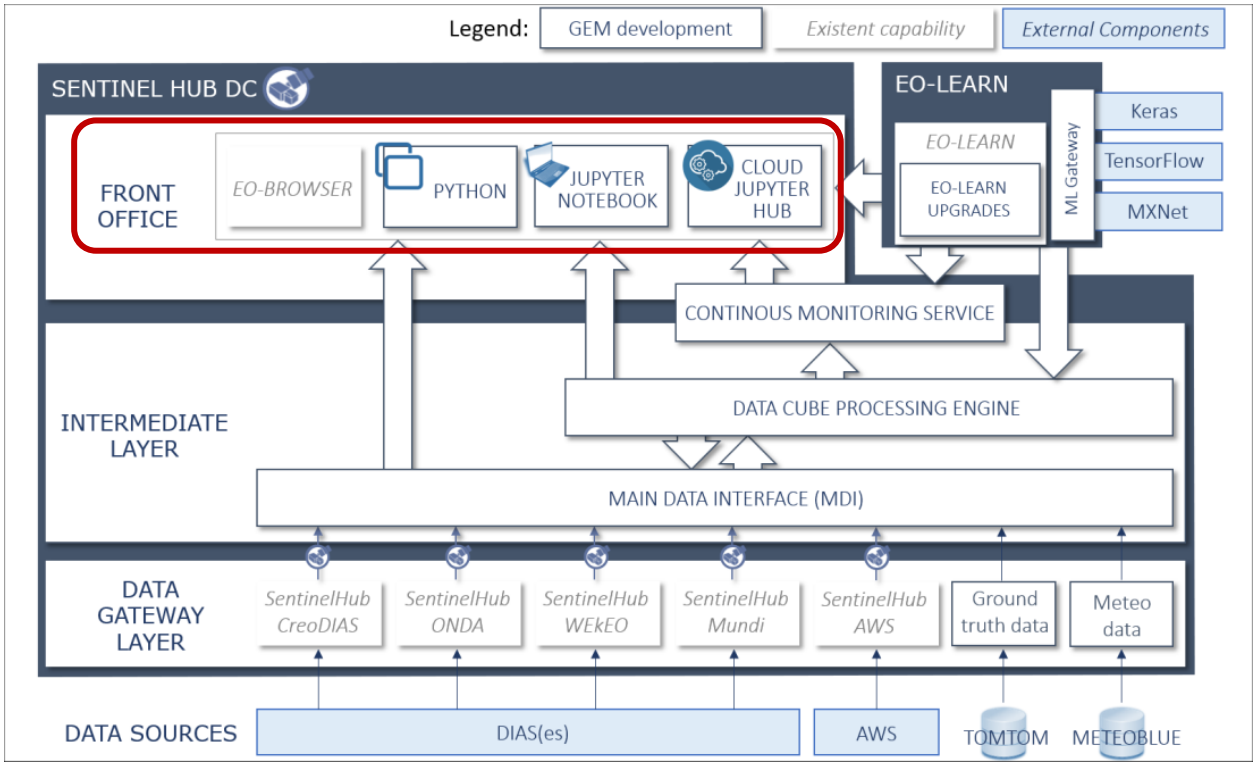

Figure 1: Architecture of Sentinel Hub / eo-learn as proposed in GEM.

This deliverable reports on the Front Office Services, marked with red: Python (scripts), using Jupyter notebook (e.g., locally) and using cloud based Jupyter Hub deployments. It must be noted that several user-facing services, connected either to Sentinel Hub services, or meteoblue weather API, were not included in the architecture design and are not part of this deliverable, but have nevertheless been improved to facilitate the new/updated functionalities of the services. User-facing services (e.g., web applications) of the Sentinel Hub (and meteoblue) are very tightly connected to GEM project in general, and they are constantly being updated to facilitate and support GEM framework, but as not all things were funded by GEM, we did not think appropriate to report about them in this deliverable. EO-Browser[1] and Request-Builder[2] should be mentioned explicitly, as they bring most of the Sentinel-Hub functionality to the user as web browser applications.

[1] http://apps.sentinel-hub.com/eo-browser/

[2] http://apps.sentinel-hub.com/requests-builder/

The user-facing services, pertinent to this deliverable, are linked mostly to Python access, be it via scripts (using eo-learn and eo-grow), or Jupyter notebooks (standalone/local or via cloud deployment on Euro Data Cube - EDC). Additionally, the report outlines the QGIS Sentinel-Hub plugin, as an extension of the popular GIS tool facilitating access to Sentinel-Hub datasets and Adjustable Data Cubes.

In the second version of this deliverable, we touch on the exposed front-office services, made available recently on Copernicus Data Space Ecosystem (CDSE), which is taking on the role of main DIAS platform.

2 Jupyter notebooks

Both Python and Jupyter notebook access to GEM framework is done using sentinelhub-py[3], eo-learn[4] and eo-grow[5] Python libraries, all of them being actively developed within the GEM project. This section focuses on notebooks from eo-learn, eo-learn-workshop and eo-learn-examples.

2.1 eo-learn

At the time of writing this report, eo-learn repository has more than 900 stars, more than 50 followers, and has been forked more than 260 times, as can be seen in Figure 2. Comparing to the statistics reported in deliverable D3.3 – GEM Processing Framework, the numbers have increased significantly for such short time (+60 stars, +10 forks in 6 months). With the large number of external contributors, we believe eo-learn to be a very good example of open-source software in the EO domain.

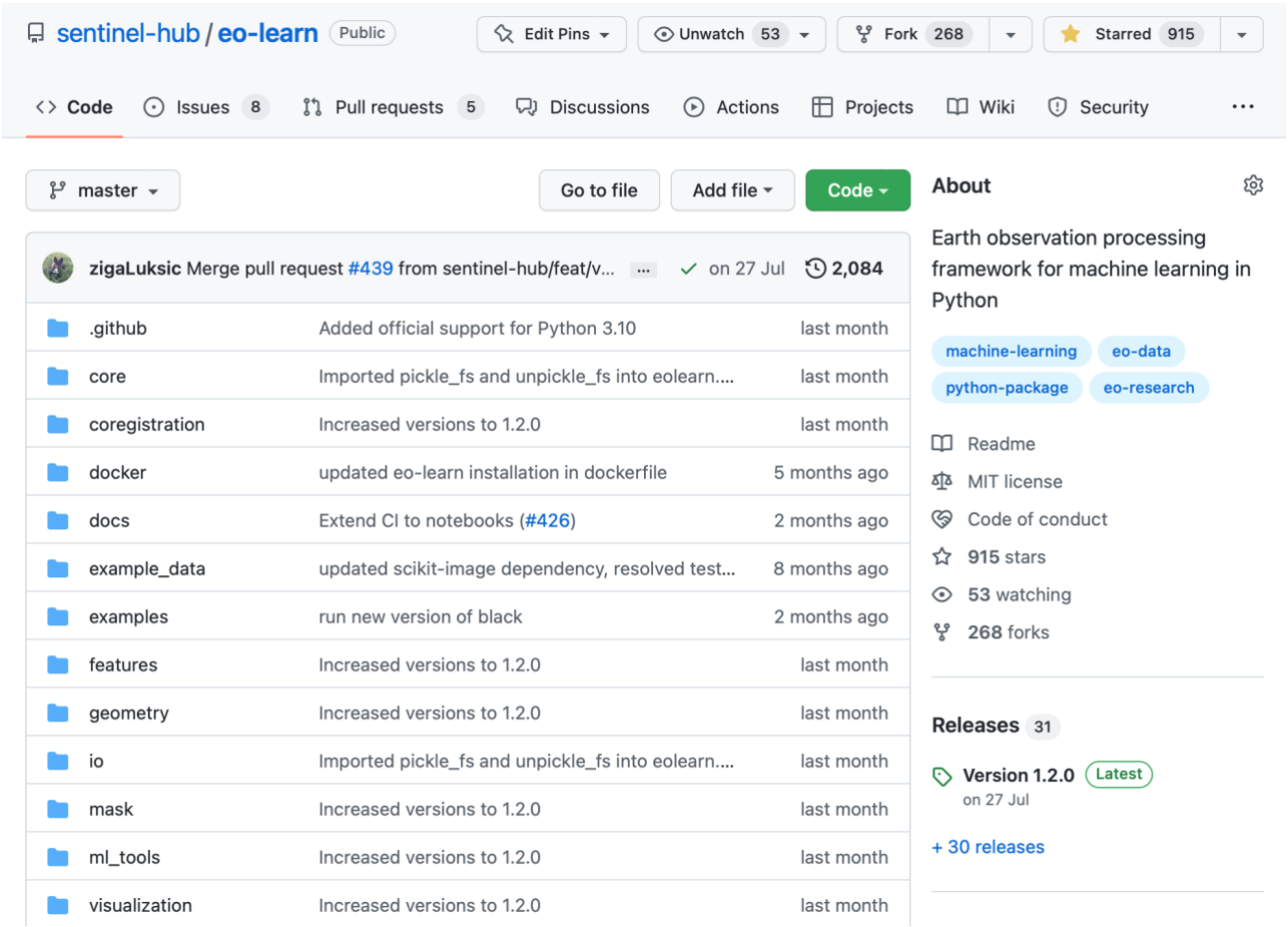

Figure 2: eo-learn home page on https://github.com/sentinel-hub/eo-learn

[3] https://sentinelhub-py.readthedocs.io/en/latest/

[4] https://eo-learn.readthedocs.io/en/latest/

[5] https://eo-grow.readthedocs.io/en/latest/

The main user-facing (and hopefully user-friendly) entry points for eo-learn are surely the Jupyter notebooks with examples. Some of them are included in main eo-learn repository and represent a big part of the eo-learn documentation, as can be seen in Figure 3.

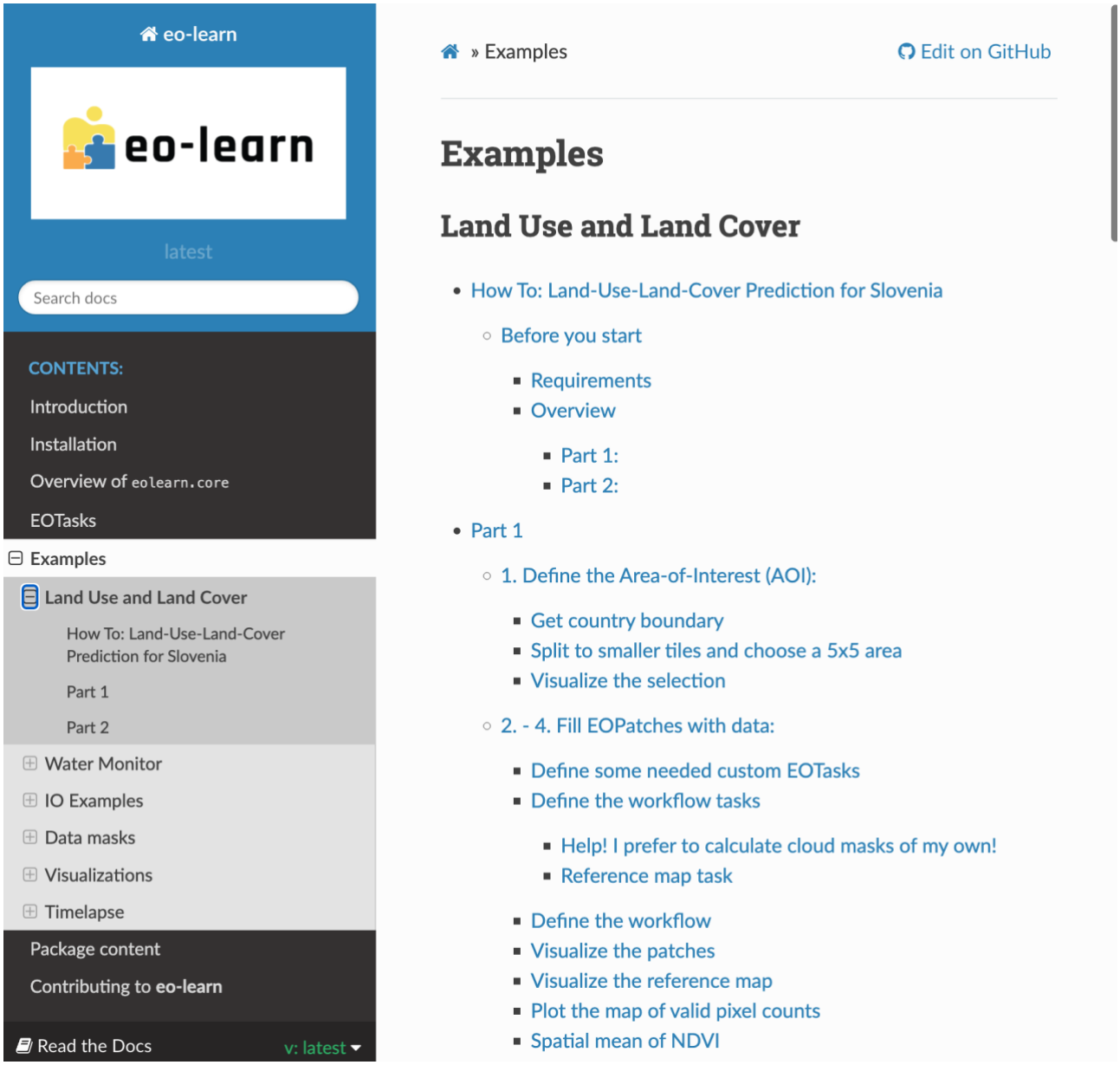

Figure 3: Jupyter notebooks as examples in eo-learn

Both GitHub, where the repository is located, as well as compiled documentation on readthedocs are capable of proper visualization (rendering) of the notebook files (i.e., raw ipynb files are structured text, not suitable for human interaction, only rendered files are). Figure 4 shows how notebook examples are rendered on GitHub and readthedocs platforms.

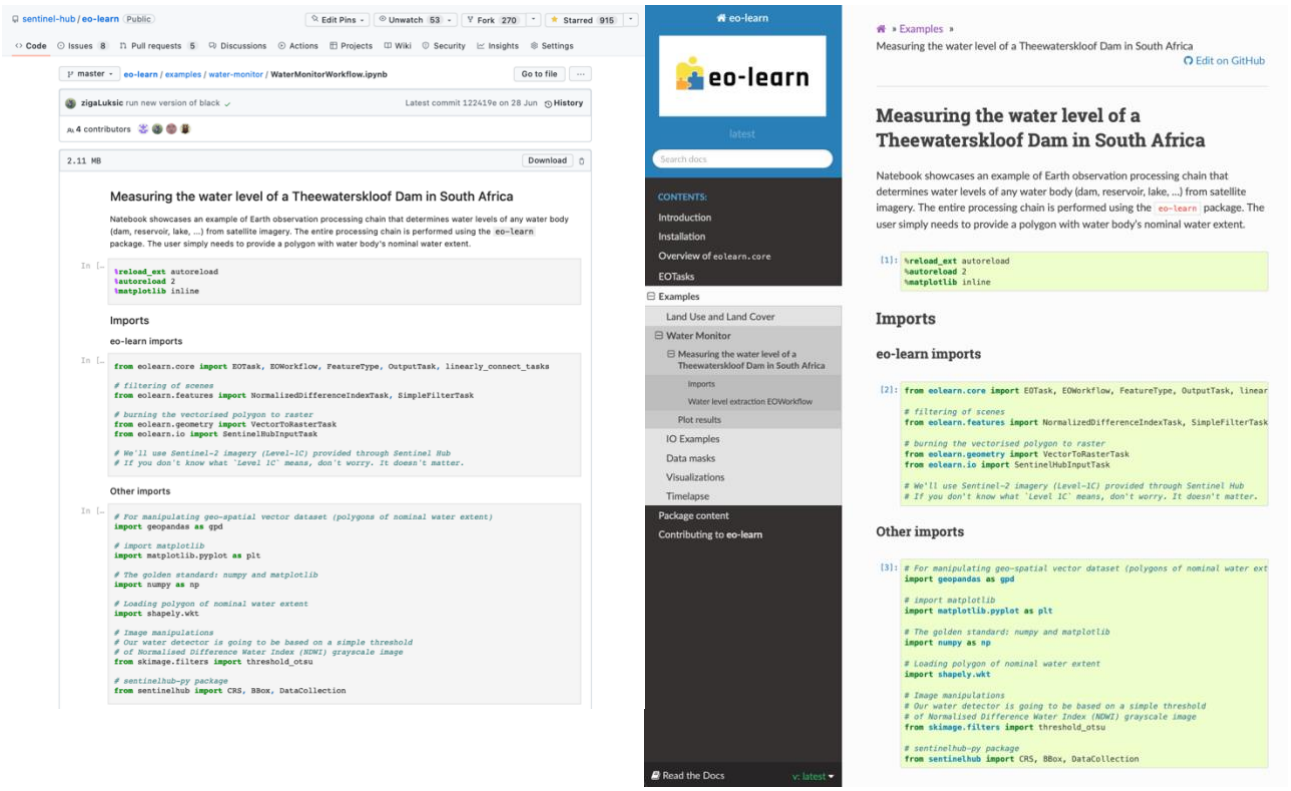

Figure 4: Rendering of Jupyter notebooks on GitHub (left) and readthedocs (right).

To run the examples from eo-learn locally (on user's machine), the installation is rather straightforward:

Code 1: Installation of eo-learn and starting Jupyter with a notebook.

```
# clone the eo-learn repository
git clone --depth 1 https://github.com/sentinel-hub/eo-learn.git

# change the directory to eo-learn
cd eo-learn

# install all eo-learn modules
python install_all.py

# install jupyterlab https://jupyterlab.readthedocs.io/en/stable/
pip install jupyterlab

# start the lab with example notebook
jupyter-lab examples/io/SentinelHubIO.ipynb
```

Additionally, eo-learn comes with docker image, and running examples is as simple as pulling the docker image, running it, and then opening the browser at http://localhost:8888/.

Code 2: Using eo-learn via docker image.

```
# pull the latest eo-learn together with included examples
docker pull sentinelhub/eolearn:latest-examples

# run the docker - it will open the Jupyter with notebooks directly
docker run -p 8888:8888 sentinelhub/eolearn:latest-examples
```

2.2 eo-learn-workshop

eo-learn-workshop[6] repository contains material that has been originally prepared for the Nordic Remote Sensing 2019 conference, where we had a workshop on "Bridging Earth Observation data and Machine Learning in Python". Since then, we've been constantly upgrading the workshop, both to reflect the changes and improvements in eo-learn, as well as to align with several other similar workshops, like the "ESA Land Training 2021" that was held in Ljubljana.

The repository encourages users to run the tutorial by themselves via Binder:

https://mybinder.org/v2/gh/sentinel-hub/eo-learn-workshop/master

With Binder, the user can open and execute the workshop notebooks in an executable environment in the cloud, making the code immediately runnable/reproducible by anyone, anywhere. The JupyterLab Binder instance with Introduction notebook is shown in Figure 5.

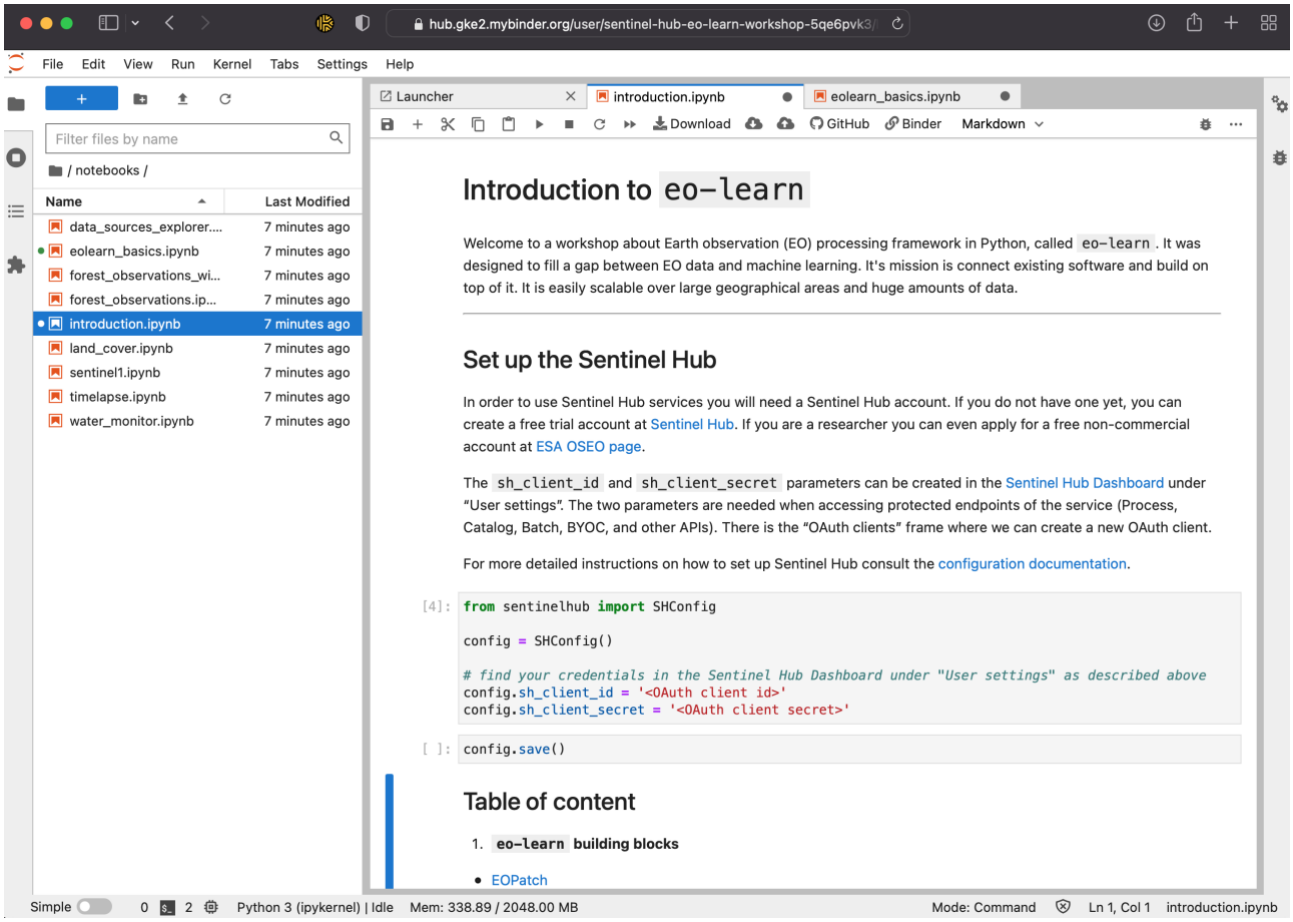

Figure 5: Running eo-learn-workshop on Binder.

[6] https://github.com/sentinel-hub/eo-learn-workshop

2.3 eo-learn-examples

A big change in eo-learn repository that was done within GEM is the creation of a new repository eo-learn-examples[7], where we have extracted many examples from eo-learn. This removes a big burden of maintainability of eo-learn repository, reduces the size of the repository, and allows users to better browse through examples while keeping the responsibility of the code limited to eo-learn.

eo-learn-examples repository contains numerous examples of EO processing workflows that extract valuable information from satellite imagery, giving you hints and ideas how to use the EO data.

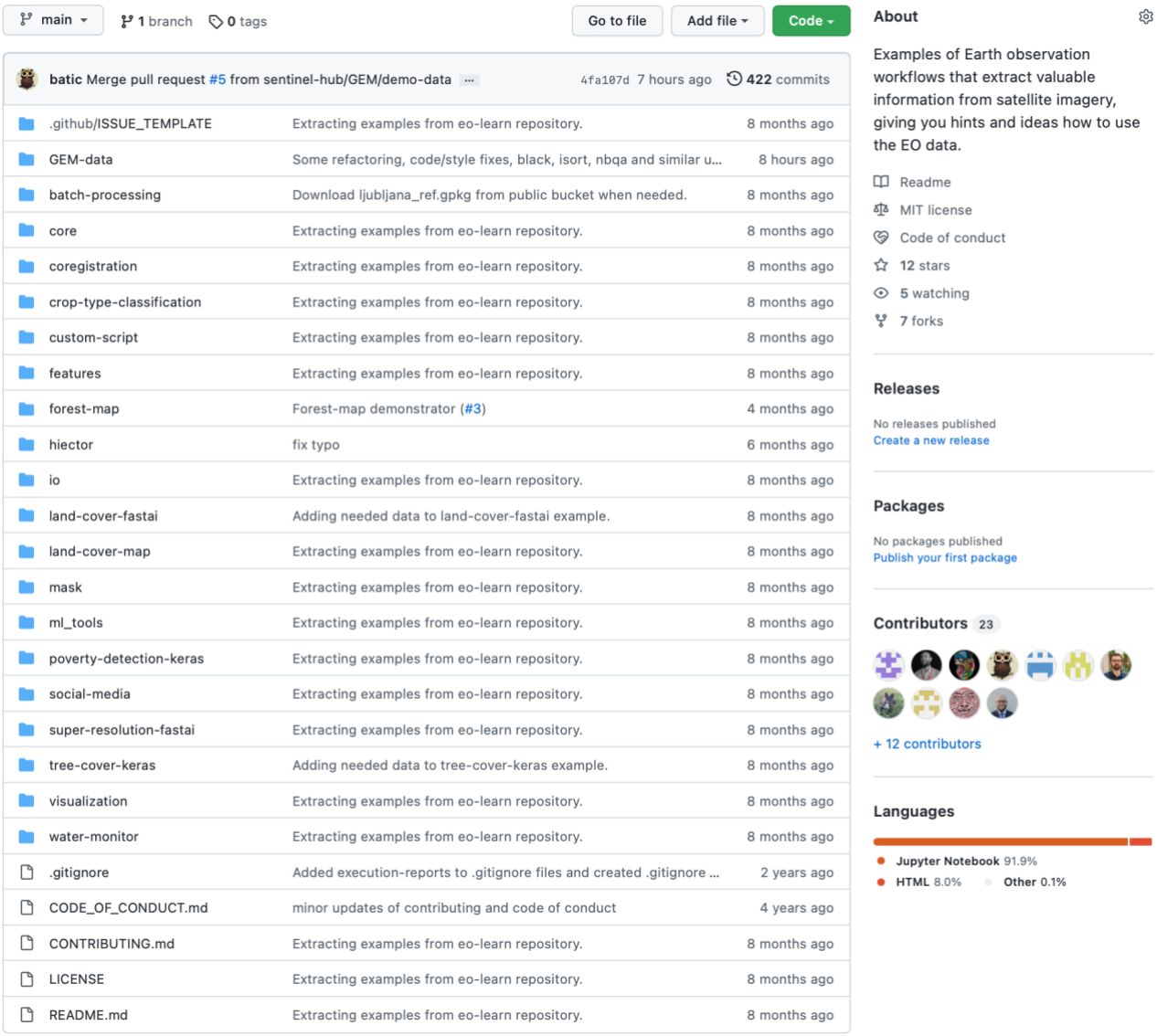

Figure 6: eo-learn-examples repository

[7] https://github.com/sentinel-hub/eo-learn-examples

Perhaps of particular importance for GEM project are the GEM-data examples, showcasing the the data, available within the project for the use cases and demonstration activities. The notebooks are in the `GEM-data` folder of the eo-learn-examples repository. The code snippet to retrieve Theia Land Cover data[8] (available from CNES for France areas from 2016-2021 on yearly basis) is shown in Figure 7.

8. Theia (CNES) Land Cover Map (TLCM)

```
tlcm_data_collection = DataCollection.define_byoc(
    "9baa2732-6597-49d2-ae3b-68ba0a5386b2",
    service_url="https://services.sentinel-hub.com",
    bands=[Band(name="OCS", units=(Unit.DN,), output_types=(np.uint8,))],
)
```

•••

```
tlcm_download_task = SentinelHubEvalscriptTask(
    features=(FeatureType.DATA, "TLCM_data"),
    evalscript=evalscript,
    data_collection=tlcm_data_collection,
    resolution=30,
    max_threads=3,
)
```

```
eopatch_tlcm = tlcm_download_task.execute(bbox=roi1_bbox, time_interval=["2020-01-01", "2020-12-01"])
```

```
tlcm_cmap, tlcm_norm, tlcm_ticks, tlcm_labels = construct_cmap(TLCM, eopatch_tlcm.data["TLCM_data"], name="tlcm_cmap")
```

•••

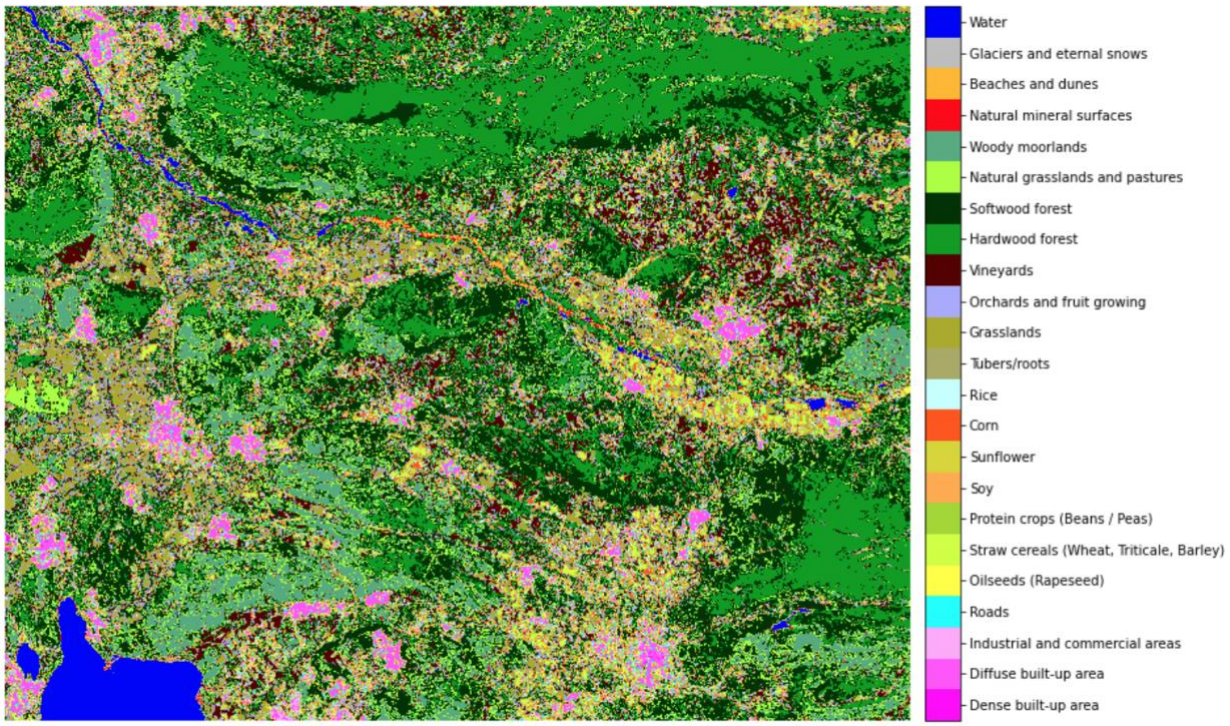

Figure 7: eo-learn-examples code snippet for retrieving Theia Land Cover Map product.

Figure 8 showcases how eo-learn task for meteoblue Weather API services can be used to access weather/climate data provided by meteoblue.

[8] https://collections.sentinel-hub.com/cnes-land-cover-map/

2. Retrieving NEMS4 data in raster format

```
eopatch = EOPatch(bbox=to_utm_bbox(coast_bbox), timestamp=[datetime.datetime(2022, 8, 1, 11, 0, 0)])
```

```
query = {
    "domain": "NEMS4",
    "gapFillDomain": None,
    "timeResolution": "hourly",
    "codes": [{"code": 11, "level": "2 m above gnd"}],
    "transformations": [
        {"type": "aggregateTimeInterval", "aggregation": "mean"},
        {
            "type": "resampleToUtm", "zone": 31, "hemisphere": "northern",
            "xResolutionMeters": 100, "yResolutionMeters": 100,
            "interpolationMethod": "linear",
            "spatialAggregation": "mean",
            "disjointArea": "keep",
            "elevationDownscale": "enabled",
        },
    ],
}
units = { "temperature": "C", "velocity": "km/h", "length": "metric", "energy": "watts" }

weather_feature = FeatureType.DATA, "WEATHER-DATA"
meteoblue_task = MeteoblueRasterTask(weather_feature, METEOBLUE_API_KEY, query=query, units=units)

eopatch = meteoblue_task.execute(eopatch)
```

```
fig, ax = plt.subplots(figsize=(10, 10))
ax.imshow(eopatch.data["WEATHER-DATA"][0].squeeze());
```

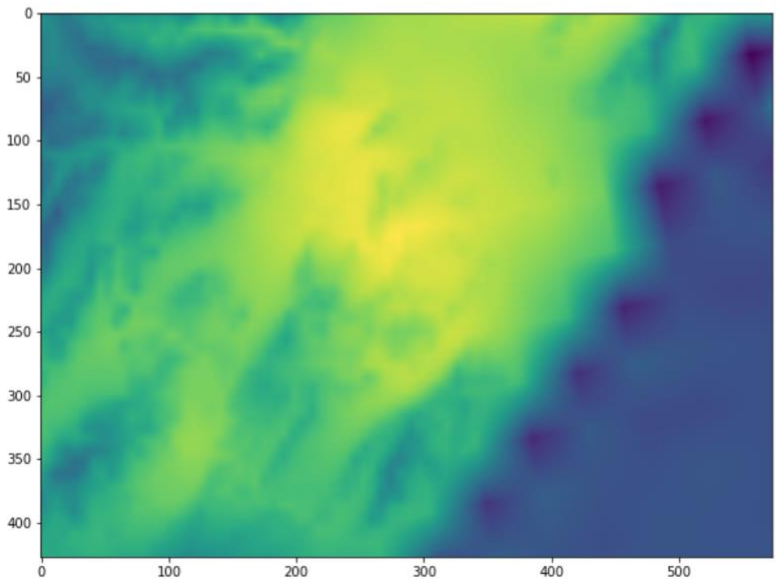

Figure 8: eo-learn-examples code snippet for retrieving NEMS4 data from meteoblue services using eo-learn task.

3 Python (scripts)

Jupyter notebooks are really good way for running small scale experiments, data exploration, and one-off tasks. Unfortunately, they are not suitable for large, asynchronous tasks like most of the complex EO workflows are. Thanks to EOWorkflow and EOExecutor, eo-learn has the functionality to scale up to larger areas. Nevertheless, issues like reproducibility and traceability of the experiments are not sufficiently addressed with just eo-learn. On top of that, we wanted the capability of coordinating several machines to do the work over large areas (spanning regions, continents, or whole world).

The tagline of eo-grow library is "Earth observation framework for scaled-up processing in Python". Working with EO data is facilitated by the eo-learn package, while the eo-grow package takes care of running the solutions at a large scale, providing reproducibility of the experiments at very low (code) development cost.

eo-grow library has been publicly released on GitHub: https://github.com/sentinel-hub/eo-grow

Features of eo-grow include:

- Direct use of EOWorkflow procedures.
- Parametrizing workflows by using validated configuration files, making executions easy to reproduce and adjust.
- Easy use of both local and AWS S3 storage with no required code adaptation.
- Splitting large areas of interest into grids and defining collections of EOPatches.
- Workflows can be run either single-process, multi-process, or even on multiple machines (by using ray clusters).
- Execution reports and customizable logging.
- Options for skipping already processed data when re-running a pipeline.
- Offers a CLI interface for running pipelines, validating configuration files, and generating templates.
- A collection of basic pipelines, with methods that can be overridden to tailor to many use-cases.

Although, or possibly because, eo-grow is facilitating large scale experimentation, spawning instances on cloud infrastructure (if/when desired), the learning curve is rather steep. That is why we are working on end-to-end examples with the extensive documentation, that presents an example of a "generic" EO workflow with pipelines as shown in Figure 9.

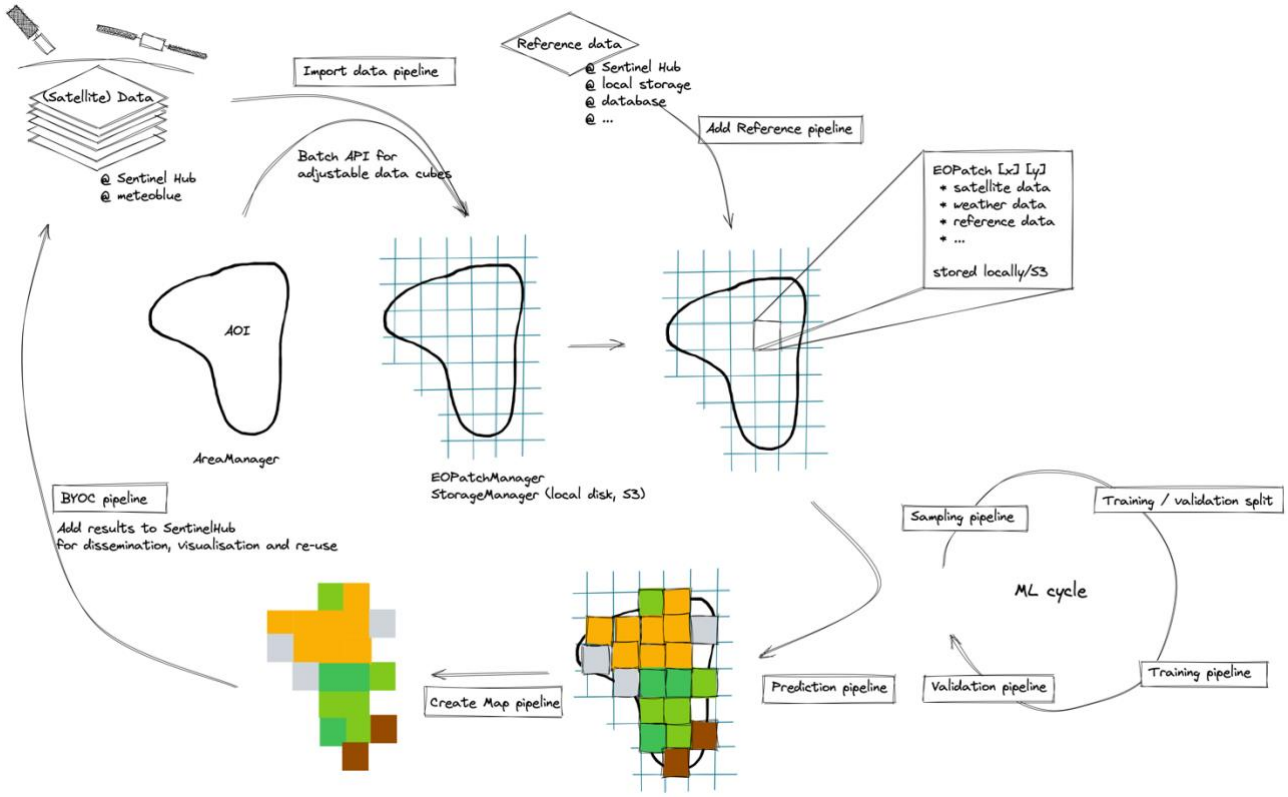

Figure 9: Schematic diagram of "generic" EO workflow.

The full example, including the instructions for setting up cloud environment and infrastructure, is given in eo-grow-examples repository:

https://github.com/sentinel-hub/eo-grow-examples/tree/main/GEM

The main benefit of eo-grow is the “no development” approach to EO experiments: once an eo-learn workflow is mature enough to be made into an eo-grow pipeline, the further use of such pipeline is facilitated simply by writing appropriate config file, like shown in Code 3.

The config files are very lightweight, can be versioned very simply, parameters changed without introducing code-changes. Additionally, the config files can simply be sent to worker machines when scaling up the infrastructure, bringing very low overhead to the network load.

Code 3: Configuration file to retrieve Sentinel-2 L2A data cube using Sentinel Hub Batch processing.

```
{
  "pipeline": "eogrow.pipelines.download_batch.BatchDownloadPipeline",
  "**global_config": "${config_path}/../config_2017.json",
  "output_folder_key": "data",
  "evalscript_path": "${import_path:gem}/../config_files/FRCD/data/evalscript_${var:year}.js",
  "data_collection": "SENTINEL2_L2A",
  "time_period": ["${var:start_time}", "${var:end_time}"],
  "tiff_outputs": ["B01", "B02", "B03", "B04", "B05", "B06", "B07", "B08", "B8A", "B09", "B11", "B12", "QM"],
  "save_userdata": true,
  "resampling_type": "BILINEAR",
  "mosaicking_order": "leastRecent",
  "monitoring_sleep_time": 60,
  "                    "
```

4 EOxHub Workspace

Sentinel Hub (with all its components - process API, Batch API) are being also used within the scope of ESA-funded Euro Data Cube (EDC) project. In EDC, EOX (company from Austria) is developing the "front-office" functionality, named EOxHub Workspace[9].

EOxHub Workspace provides, among other things, hosted JupyterLab environment with a set of pre-configured notebooks, well integrated with Sentinel Hub (and other) services. It provides to the user a seamless way to start prototyping the algorithm development and then scale it up. JupyterLab environment on EOxHub comes with GEM processing framework libraries sentinelhub-py, eo-learn and eo-grow pre-installed.

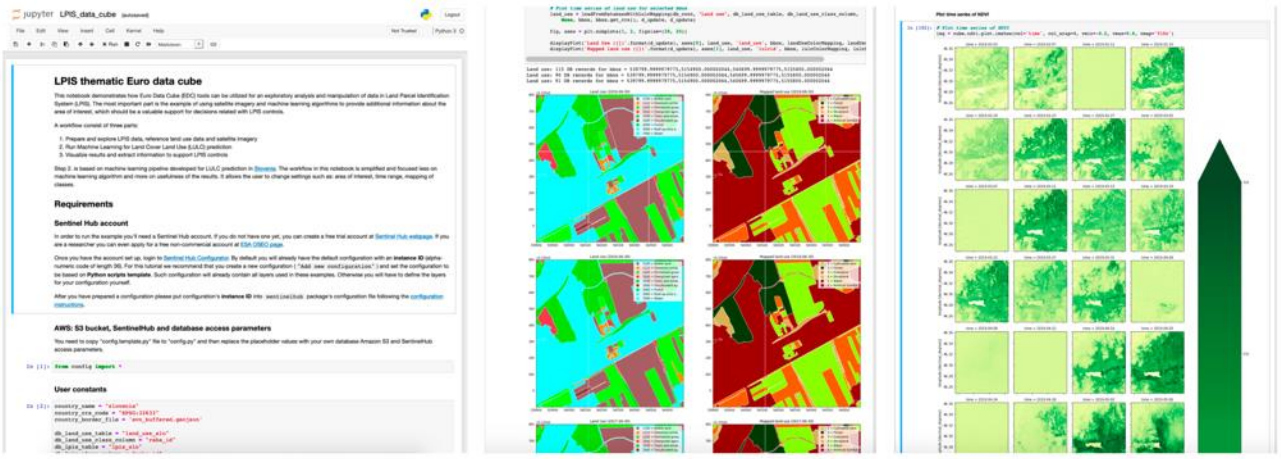

Figure 10: EOxHub Workspace in operations.

Disclosure: This platform (EOxHUB) is done out of the scope of GEM - it is mentioned in this document for consistency and transparency purpose.

To facilitate uptake of eo-grow, the eo-grow-example for GEM notebook is included on EDC Marketplace. It contains a stripped-down version of the example shown in Figure 9, where the machine learning part (training the model) has been removed. The model included in the example comes from the GEM use-case, presented in deliverable D5.2 – Built-up area use-case.

The notebook with the simplified workflow that can be run on Euro Data Cube (or locally, if copied), is shown in Figure 11. Because the procedure to publish (openly available) notebooks on EDC takes some time, the link to the notebook is not yet available. The notebook, once published, will be available through EDC Marketplace, under Notebooks, as shown in Figure 12.

[9] https://eurodatacube.com/marketplace/infra/eoxhub

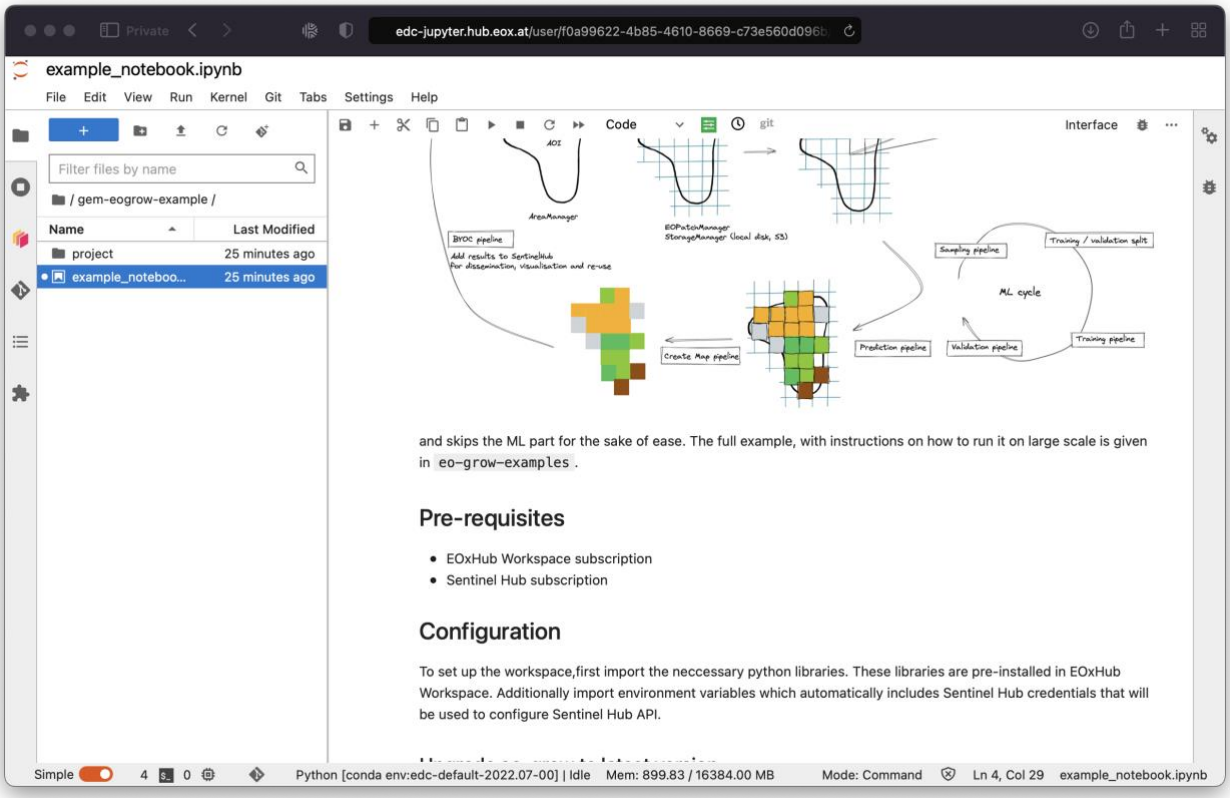

Figure 11: Notebook running example workflow with eo-grow on EDC.

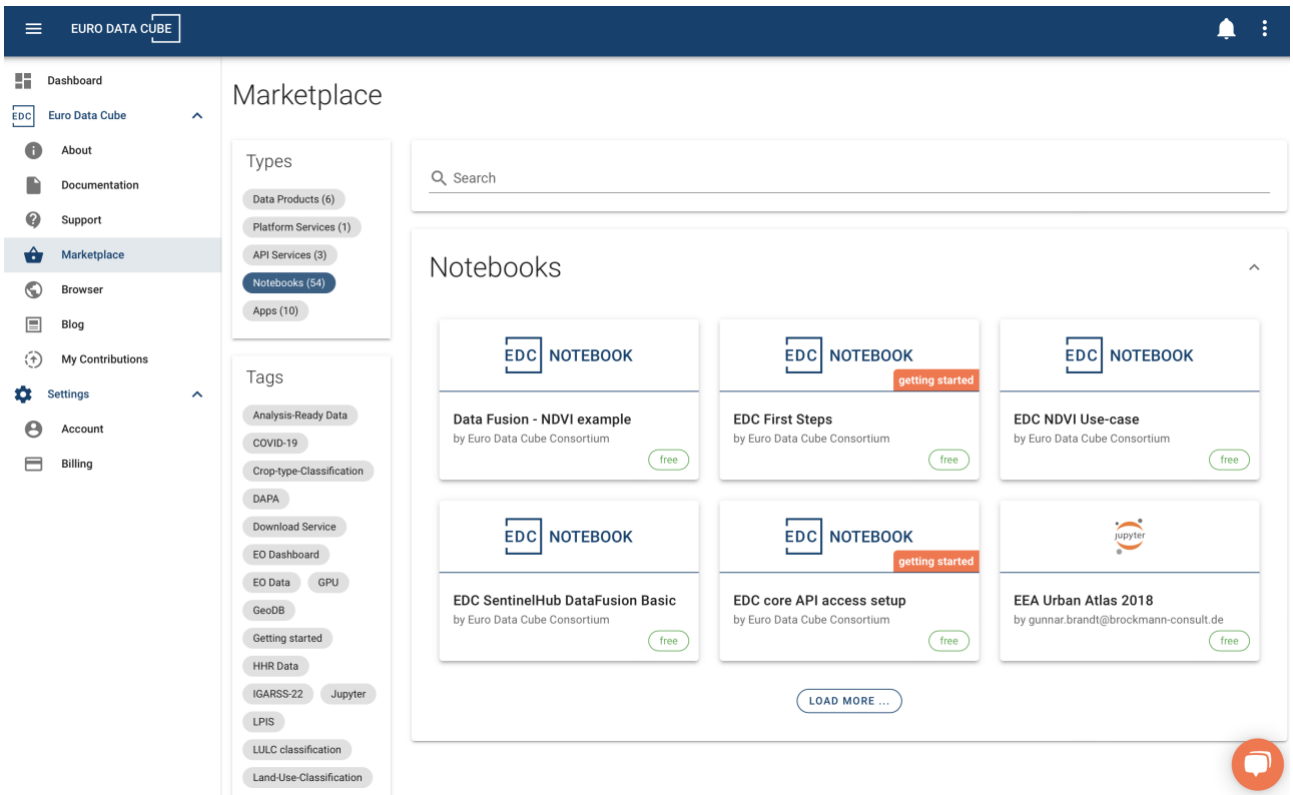

Figure 12: EDC Marketplace.

5 Sentinel-Hub QGIS plugin

SentinelHub QGIS Plugin enables users to harness the power of Sentinel Hub services directly from QGIS, a free and open source Geographic Information System (GIS) tool. The plugin is released as open source as well, the code available on GitHub:

https://github.com/sentinel-hub/sentinelhub-qgis-plugin

The plugin is available also on QGIS Official Plugin Repository, so the installation is straight-forward: open QGIS, select Plugins -> Manage and Install Plugins and search for the plugin, as can be seen in Figure 13.

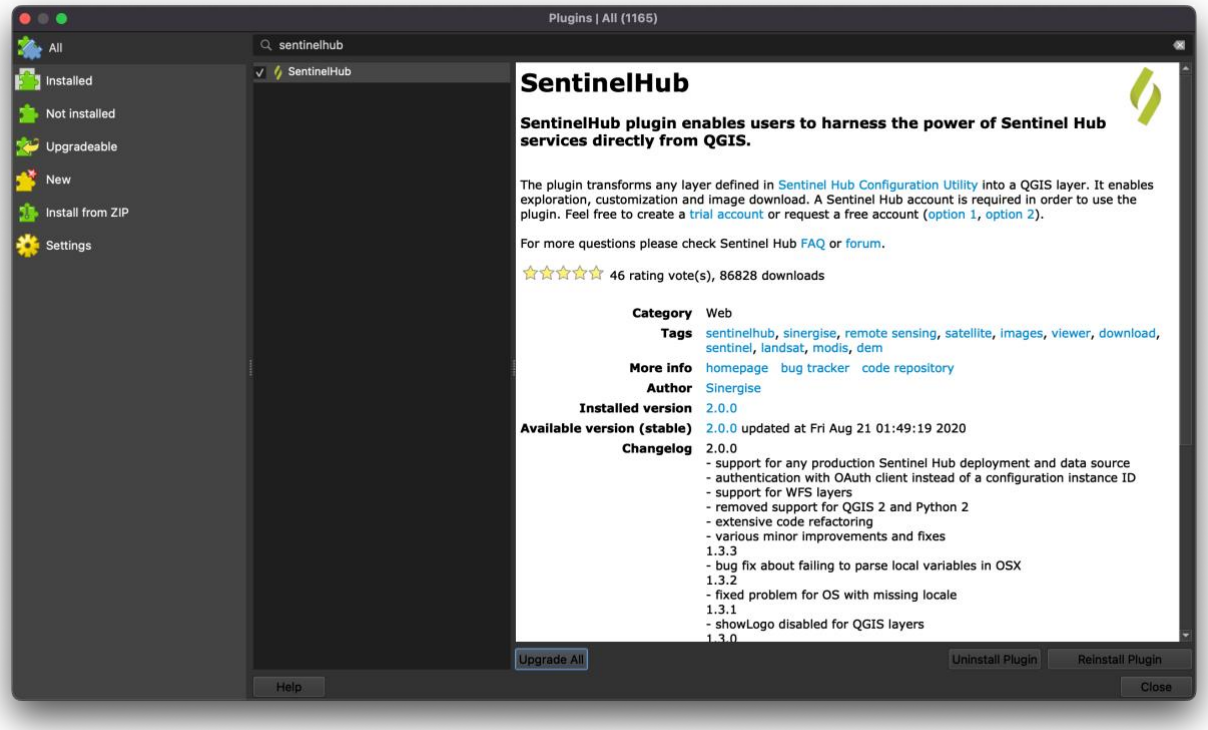

Figure 13: Installation of Sentinel-Hub QGIS plugin.

After the installation, the plugin is available by a button  in the toolbar. Clicking on it will open a docked window below the map area, where the users can select any of the publicly available Sentinel-Hub datasets,

or their own commercial and other Bring Your Own Data imagery. The example in Figure 14 shows the results of built-up area predictions that were produced using eo-grow-example presented in Section 3

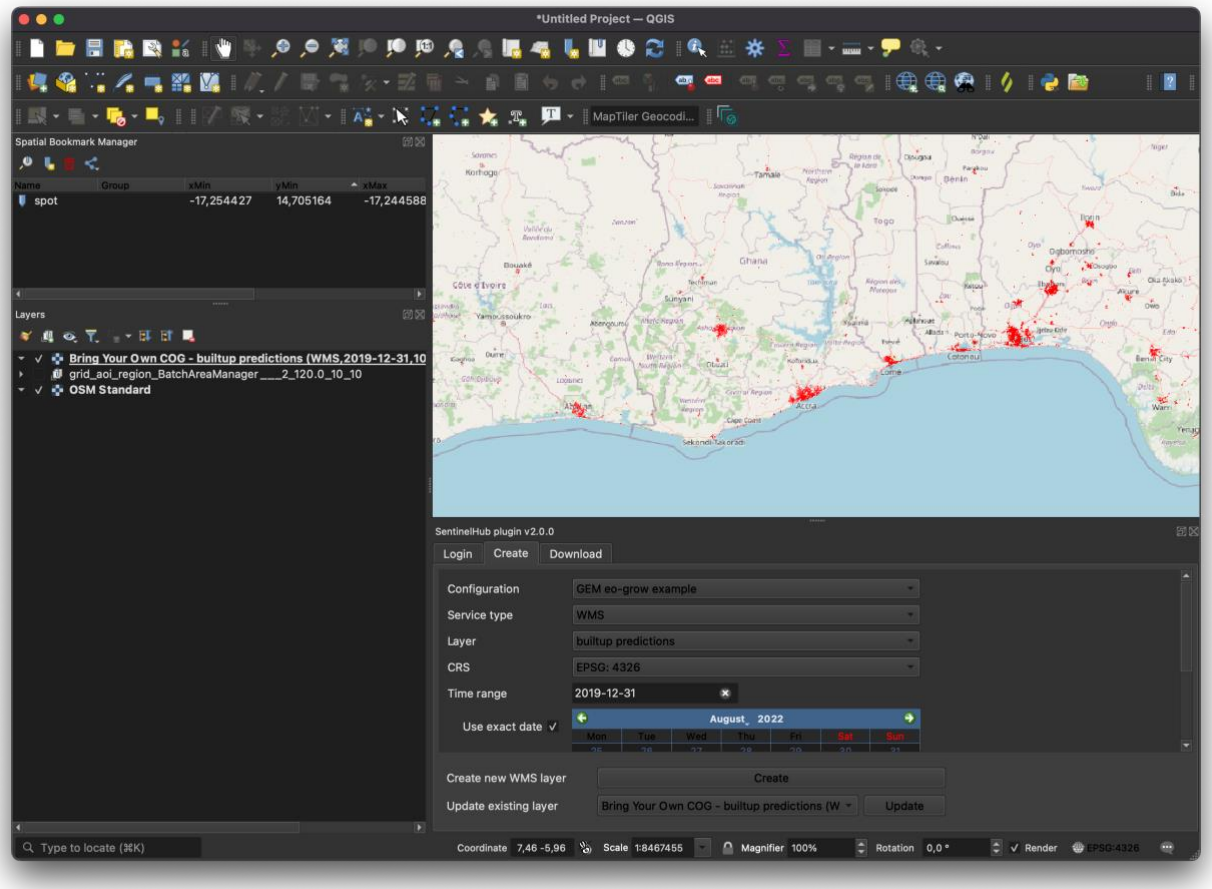

Figure 14: Sentinel-Hub QGIS plugin in action.

The plugin is focused on retrieving the data from Sentinel Hub services mostly for visualisation purposes.

Since the first version of the deliverable, QGIS plugin has been updated, and can now also serve data available through Sentinel Hub services deployed on Copernicus Data Space Ecosystem:

Figure 15: Sentinel Hub QGIS plugin supports Sentinel Hub CDSE deployment.

6 GEM front-office services on CDSE

The Copernicus Data Space Ecosystem (CDSE) was formed by combining the existing EO data centre of Copernicus with onboard virtual machine cloud computing capacity and a range of code repositories. In addition to the full archive of Sentinel data, the ecosystem hosts all data from Copernicus Contributing missions and is also offered as a platform for direct access to data products of commercial imagery providers. Cloud computing and virtual machine capacity is offered within CREODIAS 2.0 and the Open Telekom Cloud, providing the necessary capacity for regular national scale processing of imagery time series. This processing capacity comes with many popular EO data analysis packages already on board, and capacity for hosting additional code repositories.

The Copernicus Data Space Ecosystem has been designed to allow optimum collaboration between public sector and private industry actors. In addition to the code repositories and labs allowing efficient sharing of open code, the necessary pathways for commercializing datasets, data products and analysis solutions are also in place. The repository structure and the available access opportunities support streamlined access in code for building new commercial applications. Therefore, the Ecosystem is an ideal platform for both established market players and growing new companies for developing and commercializing new solutions. The seamless integration of these capacities provides a transformative platform for earth observation data analysis, with opportunities that would not have been possible with incremental improvements of the individual sub-systems as separate units. All in all, the Copernicus Data Space Ecosystem lays down the foundations for a transformation in the Earth Observation industry, and specifically for a transformation in CAP monitoring.

We have built GEM processing framework (both eo-learn and eo-grow) to be cluster (e.g., cloud infrastructure) as agnostic as possible. Everything implemented in eo-learn is platform independent, meaning that the workflows can run on any infrastructure.

Since July 2023 CDSE is offering Jupyter Lab environment, where users can exploit computational capabilities of the CDSE:

https://dataspace.copernicus.eu/analyse/jupyter-notebooks

with more documentation at:

https://documentation.dataspace.copernicus.eu/Applications/JupyterHub.html

As soon as this offering was available, we have made sure that the eo-grow-examples notebook, openly available at https://github.com/sentinel-hub/eo-grow-examples/blob/main/GEM/example_notebook.ipynb can be run on the CDSE environment as well, thus validating the CDSE capabilities as well as the usability and the platform independence of the GEM framework software stack.

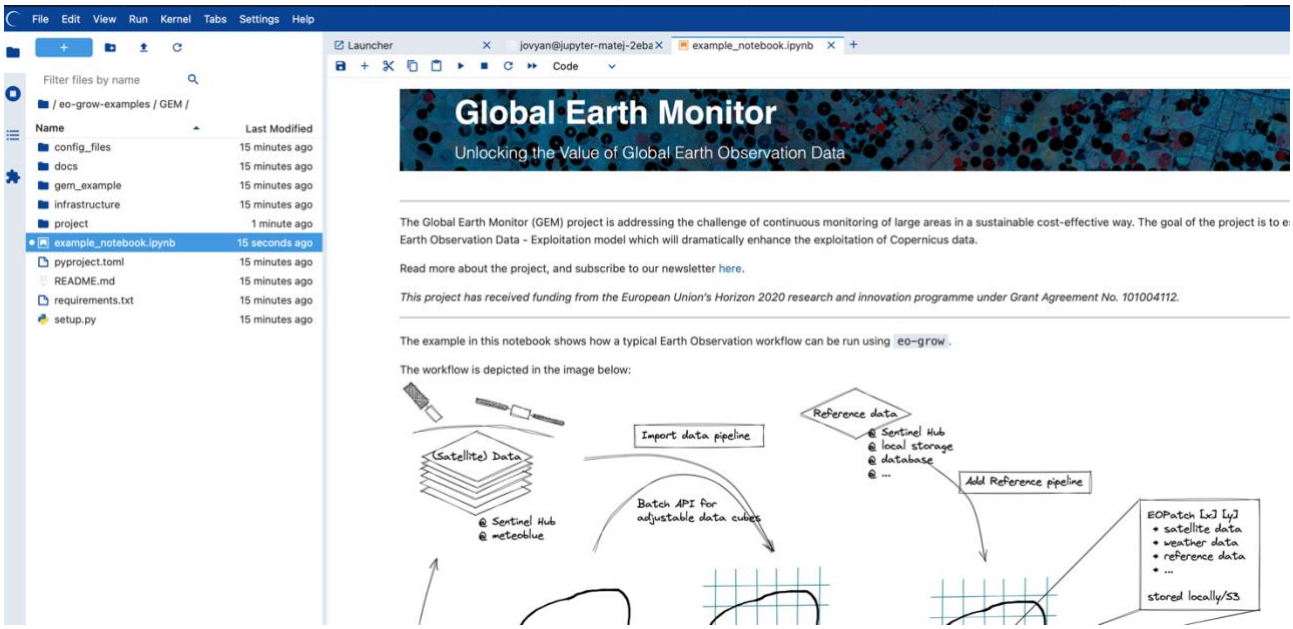

Figure 16: GEM example notebook being run on CDSE Jupyter Lab Environment.

Conclusion

This deliverable reports the Front Office Services – the tools where users can interact with the GEM framework and services in a user-friendly way.

So far, the presented services revolve around using the framework with Python language, either through interactive Jupyter notebooks, or in a more rigorous way using configuration files and scripts for repeatable work. eo-learn comes with "batteries included": the workshop repository is a hands-on walk-through the library and can be run without any installation when using Binder. The eo-learn-examples repository contains many use-cases facilitating the uptake of the library. The examples repository for the eo-grow is on the other hand limited to one sample, but with detailed instructions on how to make an EO workflow scalable on AWS infrastructure using cost-optimised spot computing instances.

Furthermore, making use of Euro Data Cube offering of EOxHUB, a demonstration Jupyter notebook shows the building blocks of eo-grow project, from start – getting the data through Data Cube processing of Sentinel Hub, to end – making the results available through Bring Your Own Data capabilities of Sentinel Hub.

The deliverable also presents the Sentinel Hub QGIS plugin, focused on retrieving the data for visualisation purposes within widely used open source GIS tool.

With the incremental development approach in GEM, we will continue to update examples in both eo-learn and eo-grow, particularly for the purposes of GEM use-cases, their demonstration and validation of the results.

Since the first version of the deliverable, the Copernicus Data Space Ecosystem data centre with onboard virtual machine cloud computing capacity has been maturing as well. We are happy to show that users can make use of GEM framework on CDSE as well, rendering the GEM framework for scalable and cost-effective workflows a first-class citizen of the CDSE.