This project has received funding from the European Union's Horizon 2022 Research & Innovation Actions - Project. 101086355 – HORIZON-CL6-2022-GOVERNANCE-01

D4.1 The RI ENVIRONMENT

31 January 2024

Author		
Name	**Organization**	**Draft release date**
Nuno Grosso	DES	04/12/2023

Approval on behalf of the Executive Board		
Name	**Organization**	**Date of approval**
Dainis Jakovels	VRI IES	27/12/2023
Giorgos Charvalis	NP	25/01/2024

Revision records			
Version	**Date**	**Changes**	**Authors**
1.0	30/01/2024	Original document (version 1)	DES

Acronyms and Abbreviations

Acronyms and Abbreviations	
AGINS	AgroInsurance International
AKIS	Agricultural Knowledge and Innovation System
ATB	Institut für angewandte Systemtechnik Bremen GmbH
AUTh	Aristotle University of Thessaloniki
CA	Consortium Agreement
CAP	European Union's Common Agricultural Policy
CORDIS	Community Research and Development Information Service
DES	Deimos Spain
DL	Deep Learning
DME	DEIMOS ENGENHARIA SA
DMK	DMK Deutsches Milchkontor GmbH
EC	European Commission
EEA	European Environment Agency
EEAB	External Expert Advisory Board
EGM	Easy Global Market SAS
EO	Earth Observation
EOSC	European Open Science Cloud
EU	European Union
EURAC	Accademia Europea di Bolzano (Eurac Research)
EV ILVO	Eigen Vermogen van het Instituut voor Landbouw en Visserij Onderzoek
ExBo	Executive Board
F2F	Farm to Fork
FE	Farm Europe
GEOGLAM	Group on Earth Observations Global Agricultural Monitoring Initiative
ICCS	Institute of Communication and Computer Systems
GA	Grant Agreement
IFAPA	Instituto Andaluz de Investigación y Formación Agraria, Pesquera y Alimentaria
IGAD	Improving Global Agricultural Data
IPR	Intellectual Property Rights
JRC	Joint Research Centre
JRC-MARS	JRC Monitoring Agricultural ResourceS
KPI	Key Performance Indicator
KUVA	Kuva Space Oy

LUKE	Natural Resources Institute Finland
MIGAL	MIGAL Galilee Research Institute
NP	Neuropublic SA
OHB DS	OHB Digital Services GmbH, Bremen, Germany
PSNC	Instytut Chemii Bioorganicznej Polskiej Akademii Nauk
R&D	Research and Development
RDA	Research Data Alliance
RIL	Research and Innovation Lab
RSE	Remote Sensing of Environmnent
SDG	Sustainable Development Goal
SF	Smart Farming
SoM	Social Media
SWIR	Short Wave Infra Red
UGent	Universiteit Gent
UN	United Nations
VITO	Vlaamse Instelling voor Technologische Onderzoek
VNIR	Visible and Near Infra-Red
VRI IES	Foundation "Institute for Environmental Solutions"
VTT	Technical Research Centre of Finland Ltd.
WODR	Wielkopolski Osrodek Doradztwa Rolniczego w Poznaniu
WP	Work Package

Table of Contents

List of Figures

1. Introduction

1.1. Project overview

ScaleAgData is a response to the call HORIZON-CL6-2022-GOVERNANCE-01-11 Upscaling (real-time) sensor data for EU-wide monitoring of production and agri-environmental conditions. The ScaleAgData project will run from January 2023 till December 2026 and consists of a consortium of twenty-six partners from fourteen countries. The vision of ScaleAgData is two-fold. On one hand, it wants to obtain insights in how the complex data streams should be governed and organized (governance call). On the other hand, it aims to develop the data technology needed to scale data collected at the farm level to regional datasets and demonstrate the benefit of these datasets for agri-environmental monitoring, and the management of agricultural production.

To do so, ScaleAgData has five objectives:

- To develop innovative approaches for collecting in-situ data and applying data technologies.
- To enable and promote data sharing along the entire data value chain.
- To demonstrate how the sensor data can be scaled to agri-environmental data products at the national, regional, or European level.
- To demonstrate the benefit of improved monitoring capacities in a precision farming context.
- To demonstrate the benefit of upscaled regional datasets for the agricultural sector in general.

During its lifecycle, the project will explore seven innovation areas: innovative sensor technology, edge processing, data sharing architecture and data governance, satellite data augmentation, from data assimilation to service development, privacy-preserving technology, and data integration methodologies.

Six Research and Innovation Labs (RIL) have been identified within the project, across various bio-geographical regions of Europe, where different data upscaling and integration models or approaches will be evaluated and demonstrated. The six RILs are: water productivity, crop management, yield monitoring, soil health, grasslands, and sustain dairy. Recommendations will be formulated on how such integrated datasets can be capitalized to help national and regional policy making to strengthen both the competitiveness and sustainability of European agriculture.

1.2. Scope of the document

This document provides an overview of the R&I Environment that will support throughout the project the algorithm/application development activities of the different ScaleAgData partners. The R&I environment related activities are the scope of task 4.3 of WP4 - Product and service development.

1.3. Document structure

This document is structured as follows:

- Overview of R&I Environment
- Architecture of the R&I Environment
- User Manual

2. Overview of the R&I Environment

The R&I environment is built over the services4EO component ecosystem. services4EO is a collaborative EO system, managed by Deimos, relying on a set of core IT and Earth observation tools, which can be chained together to build services easily tailored by the users. The aim is to empower service providers with easy-to-use, robust tools for leveraging their investment and quickly setup and develop bespoke solutions, without the need to develop many tools they need from scratch.

This system is composed of different components:

- Processing and Chaining – management of EO processing service's executions. Supports both on-demand and recurrent executions. Also supports execution of processing services composed of chains of services.
- Resource Management – management of the resources available in the platform. Supports discovery and exploitation of EO data and services. Also, responsible for the appropriate persistence of the resources into the storage.
- Common Services - transversal functionality. Encompasses system functionality that is required by all domains. It includes identity and authorization management, and system monitoring.
- Exploitation User Interfaces – external interfaces that the platform provides. It includes both the public APIs and User Interfaces.
- Infrastructure - management and operation of the resources at the level of the hosting environment. It provides a uniform, secure and scalable execution environment for all the domains.
- Machine learning - services supporting or applying Machine Learning expertise, techniques and technologies.
- Data Preparation - preparation of the EO data that the processing services can consume. This includes obtaining data from external sources and pre-processing (e.g. Analysis-Ready-Data generation)
- Applications – integration and operation of many thematic EO processing applications in different thematic areas, e.g., Agriculture, Forestry, Fisheries, Aquaculture, Coastal Risks, Water Quality.

Within this ecosystem the R&I Environment is included in the Exploitation User Interfaces component with connections to several of the other components. An overall description of the architecture of services4EO and how the R&I environment connects with those components is given in the next section.

2.1. Architecture of the R&I Environment

The R&I Environment made available within ScaleAgData is based on the services4EO Virtual Lab solution. This solution provides an interactive development environment that allows the users to develop algorithms online coded in Python and execute them over discovered data. This functionality can be accessed via Jupyter Notebooks with a subset of SDKs that helps the user during the data discovery, data access, data visualization and data execution operations.

Once the development is finished it is possible to deploy the new/updated script so that it is available to be triggered by the rest of users through the ScaleAgData R&I Environment as a standard toolbox.

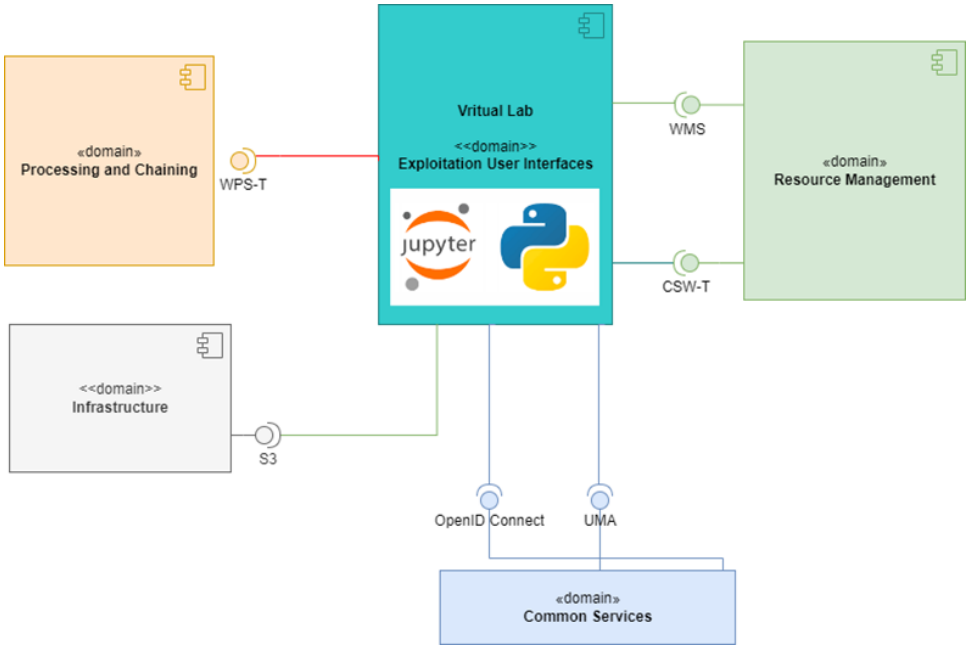


Figure 1 - R&I Environment (Virtual Lab) relationship with the other services4EO components

The Virtual Lab exposes a Jupyter Notebook interface and consumes the following main interfaces provided by the services4EO Platform:

- Open ID Connect (OIDC) and User Managed Access (UMA) for **Identity and Access Management** functionalities provided by the Common Services Domain of services4EO.
- OGC Catalogue Services for the Web (CSW) for **Data Discovery** provided by the Resource Management Domain of services4EO.

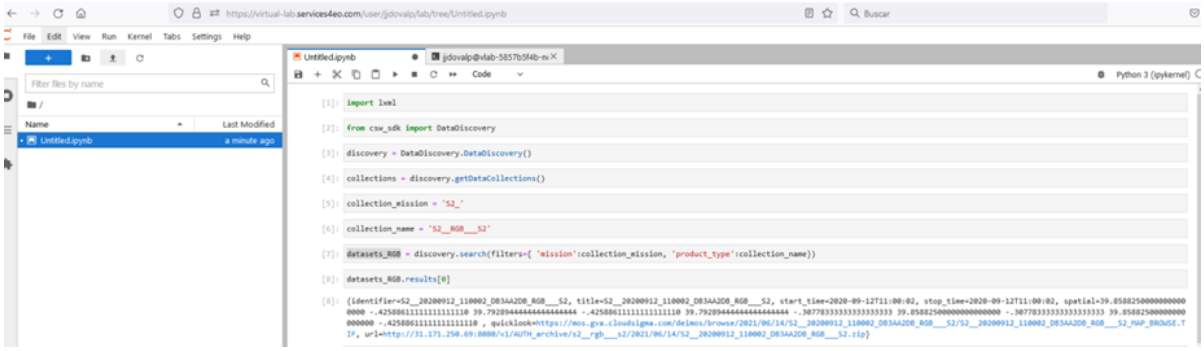


Figure 2 - Example of usage of the data discovery capabilities of the R&I Environment (Virtual Lab)

- OGC Web Map Services (WMS) for **Data Visualization** provided by the Resource Management Domain of services4EO.

```
array_list=[]
for image in images:
    img=io.imread(image)
    array_list.append(skimage.util.img_as_ubyte(img))
```

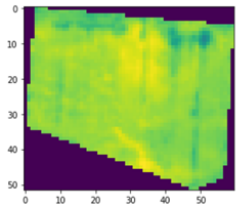
```
array_out=np.mean(array_list, axis=0)
```

```
np.mean(array_list)
```

```
60.988247863247864
```

```
plt.imshow(array_out)
```

```
<matplotlib.image.AxesImage at 0x7fc13a1718d0>
```




```
io.imsave(os.path.join(outputs_dir,"test.tif"), arr=array_out)
```

Figure 3 - Example of usage of the data visualization capabilities of the R&I Environment (Virtual Lab)

- Simple Storage Service (S3) for **Data Access and Download** provided by the Infrastructure Domain of services4EO.

Download products and store them in a folder inside virtual lab

```
for product in subset:
    url = product.url
    product_download_path = os.path.join(os.path.join(download_dir, url.split("/")[-1]))
    print(product_download_path)

    # check if not already downloaded
    if not os.path.isfile(product_download_path):
        print("Downloading from Archive product " + product.title + '... ')
        urllib.request.urlretrieve(product.url, product_download_path)

    else:
        print('Product: '+ product.title + ' already downloaded in path: ' + product_download_path)

print("Download has finished")
```

```
./Downloads/zips/S2__20220430_000000_FE1D9F21_VITONDVI
Downloading from Archive product S2__20220430_000000_FE1D9F21_VITONDVI...
./Downloads/zips/S2__20220515_000000_1CBC5008_VITONDVI
Downloading from Archive product S2__20220515_000000_1CBC5008_VITONDVI...
./Downloads/zips/S2__20220510_000000_70955634_VITONDVI
Downloading from Archive product S2__20220510_000000_70955634_VITONDVI...
Download has finished
```

Figure 4 - Example of usage of the data access and download capabilities of the R&I Environment (Virtual Lab)

The Virtual Lab can be extended for generating an SDK that allows the users to create online an Application Package (composed of the algorithm artefact and an Application Metadata file) that can be deployed as a data processing service and executed, monitored and controlled via OGC Web Processing Service (WPS) provided by the Processing and Chaining Domain of services4EO.

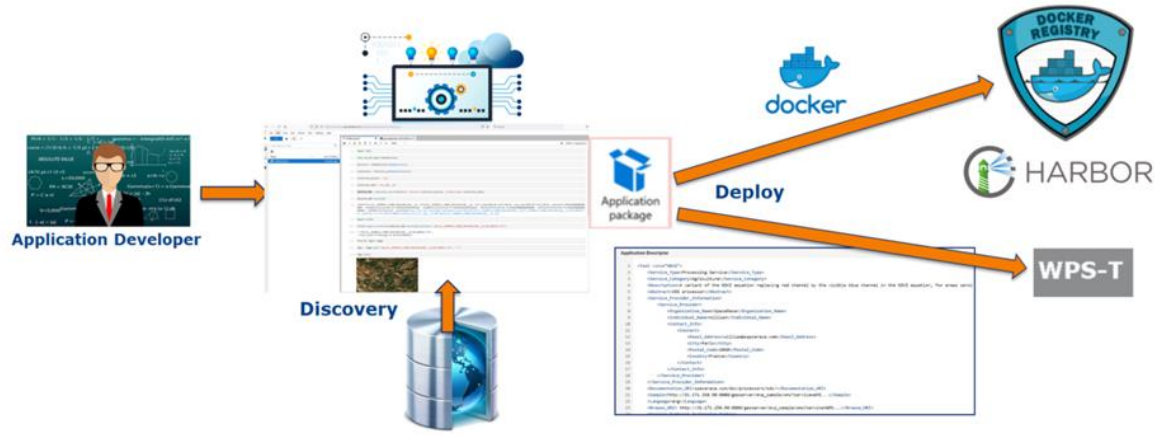


Figure 5 - Overview of the workflow to package and deploy R&I Environment developed algorithms/applications

Those processing services could be also consumed from end users via Data Exploitation or Marketplace interfaces.

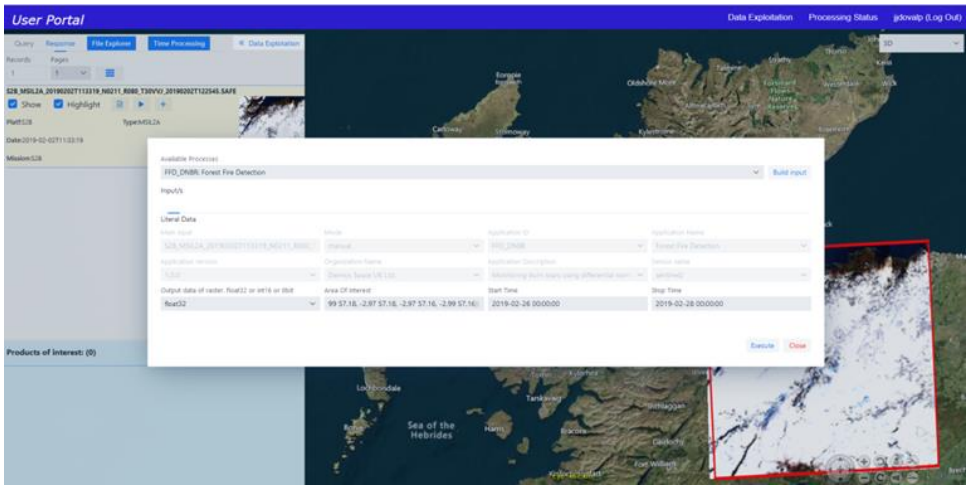


Figure 6 - Data Exploitation User Interface

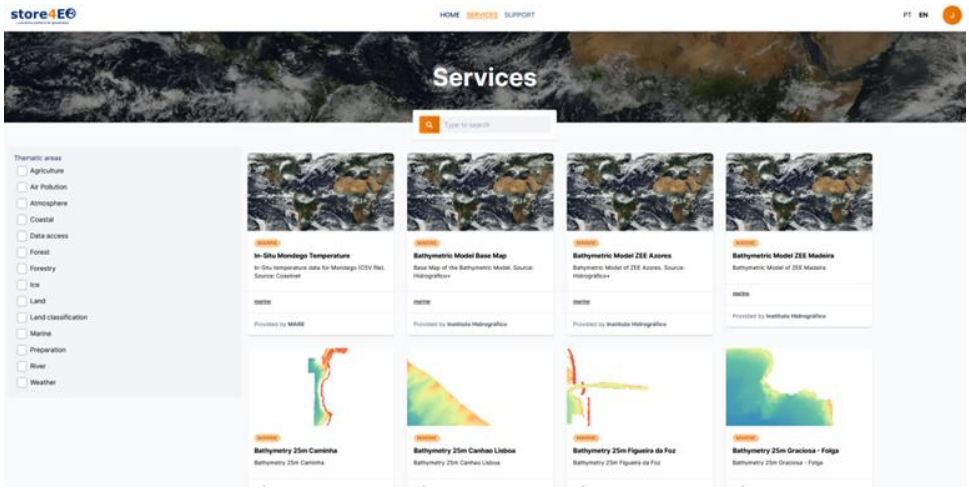


Figure 7 - Data Marketplace User Interface

3. User Manual

This development mode enables the operator to create new ScaleAgData scripts coded in Python using the framework JupyterHub. This new script is created as a Jupyter notebook.

This R&I Environment is accessed using the credentials sent to you by the operations team. With this user comes a personal storage space where the operator can develop its scripts.

Once the development is finished it is possible to deploy the new/updated script so that it is available to be triggered by the rest of users through the ScaleAgData R&I Environment as a standard toolbox. Using the Jupyter Hub enables the operator to use the following functionalities:

- Connect with ScaleAgData catalogue and archive to retrieve the inputs needed to execute the notebook.
- Use within Python of all the already existing libraries used by the MATLAB toolboxes:
    - Report Generation
    - EECFIs usage
    - Logs message
- Deployment to integrate the notebook as a ScaleAgData toolbox.

3.1. Login

To access the user's R&I Environment the operator must connect with a browser to the following url:

https://virtual-lab.services4eo.com/

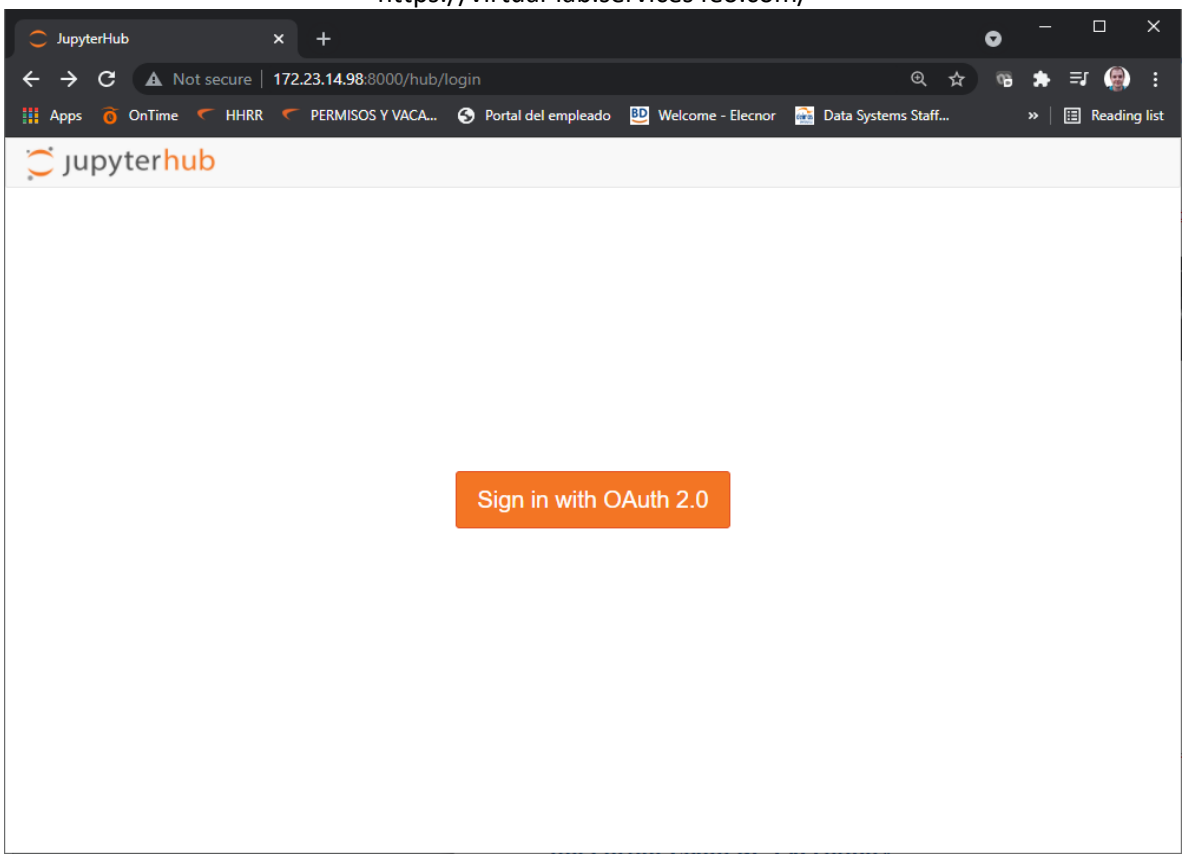


Figure 8 - Landing Page of the R&I Environment (Virtual Lab)

And enter the operator credentials:

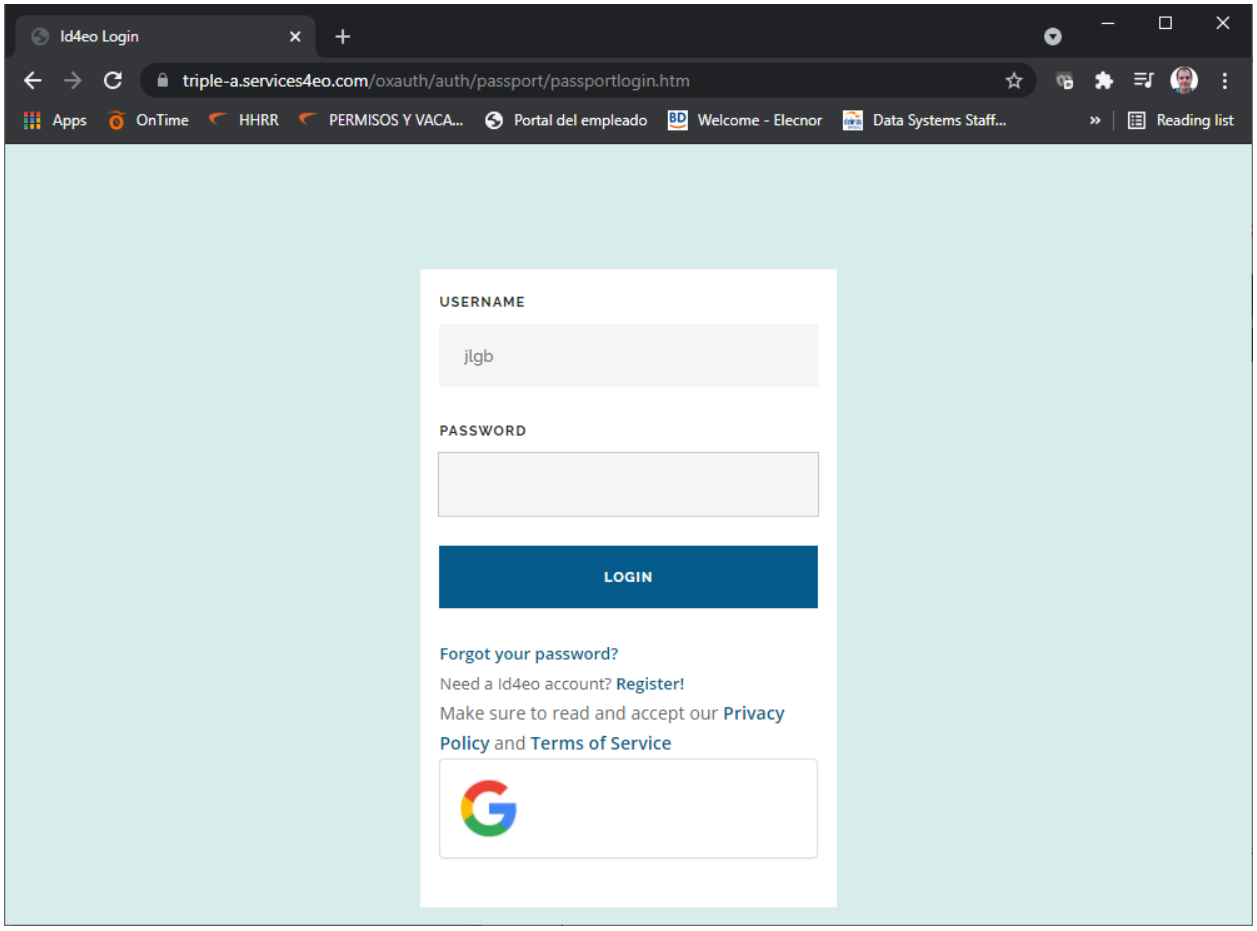


Figure 9 - Login Page of the R&I Environment (Virtual Lab)

Once logged in, the first window that appears is the following where the notebooks can be selected, opened and executed:

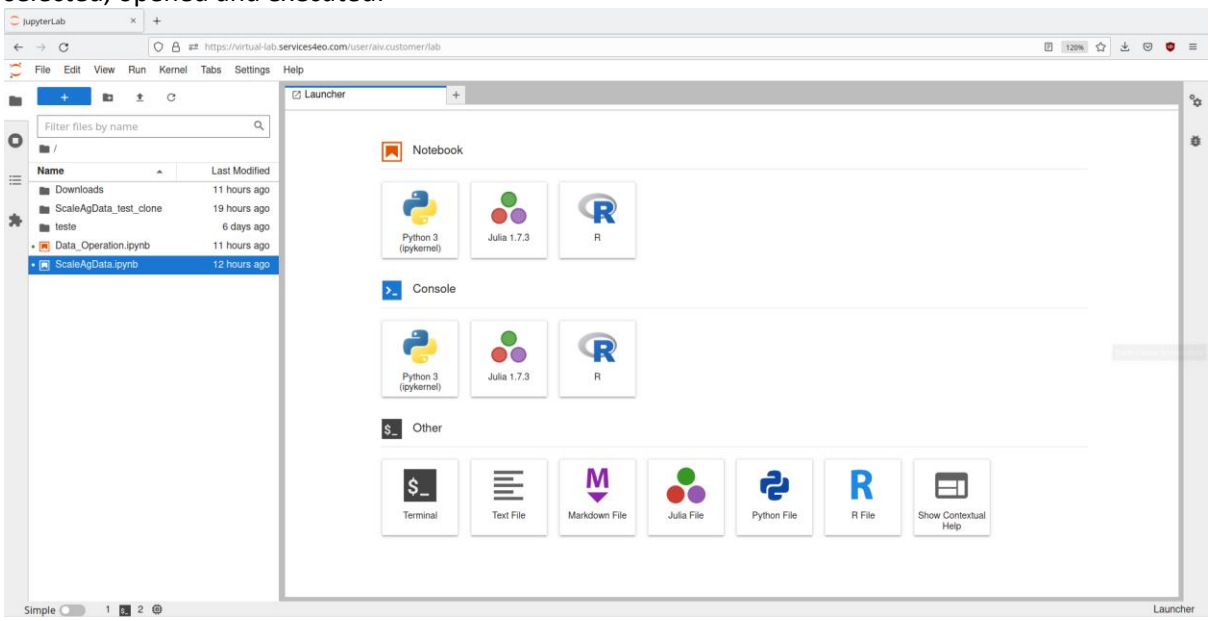


Figure 10 - Notebook type selection interface of the R&I Environment (Virtual Lab)

Note that specific python kernels can be set and configured depending on the needs. In order to develop python notebooks, it is possible to create kernels that are loaded with specific python SW packages:

```
sudo
 /opt/anaconda3/bin/conda create --yes --name python4eo scipy matplotlib
 rasterio ipykernel matplotlib ipywidgets pandas papermill

/opt/anaconda3/envs/jupyterhub/bin/python -m ipykernel install --name "python4eo" --display-name "python4eo"
```

If needed, a package can be added to an already existing kernel using **root user**:

```
conda activate jupyterhub
conda install jpype1
conda deactivate
```

3.2. Jupyter Notebook Execution

The jupyter notebooks in order to work within ScaleAgData infrastructure (log messages, report generation, common libraries availability) must have the following sections (as in the ScaleAgData notebook provided as example):

- **User defined directories:** The following variables must be defined by the operator:
    - downloadRepDir = 'Downloads'
    - zipRepDir = 'Downloads/zips/'
    - foldersRepDir = 'Downloads/folders/'
    - tmpDir = 'temp/'
- **User Data Discovery and Selection:** Section to perform the queries to the catalogue via CSW
- **User Data Output:** User defined variables to be used to generate the outputs and/or reports (if desired):
    - outputFileName = ''
    - outputRepFileName = 'Downloads/outputs/'
- **Loading toolbox Environment:** Section where all environment variables and libraries are loaded.
- **User Algorithm:** One or several sections where the main algorithm of the notebook is coded.
- **Generate Output:** Section where the function to generate the output report is called based on the variables set by the operator.

To execute a notebook it can be done by executing one cell at a time or by running the whole notebook:

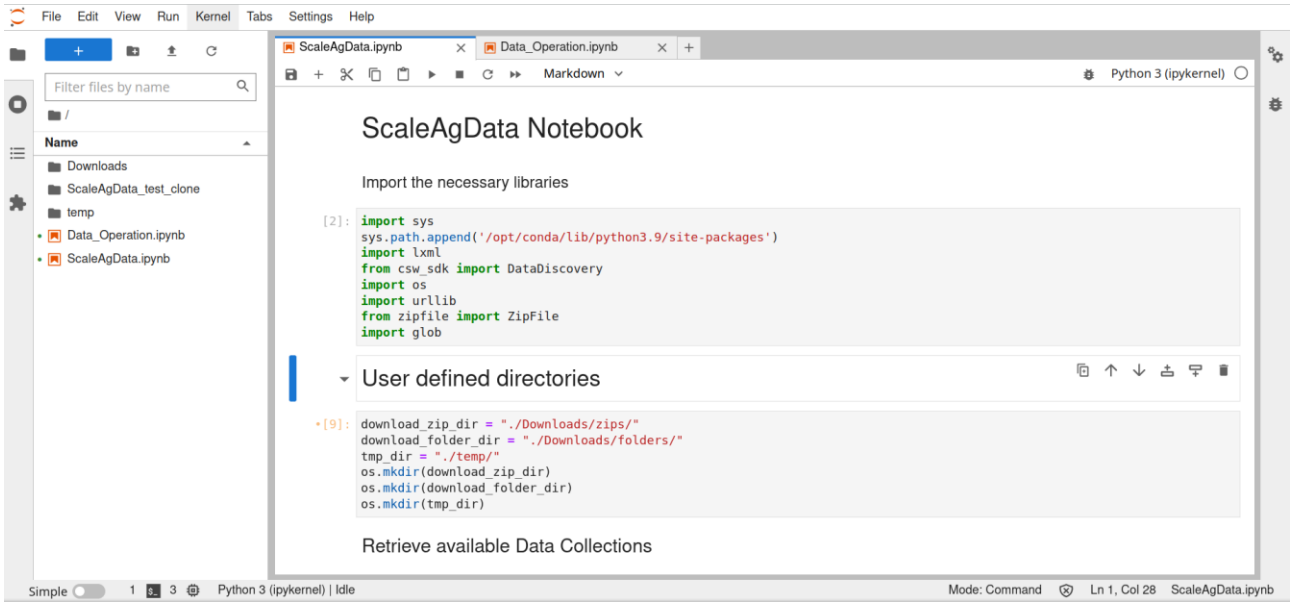


Figure 11 - Notebook execution example in the R&I Environment (Virtual Lab)

Also at any time a console can be opened that contains all variables defined at the moment:

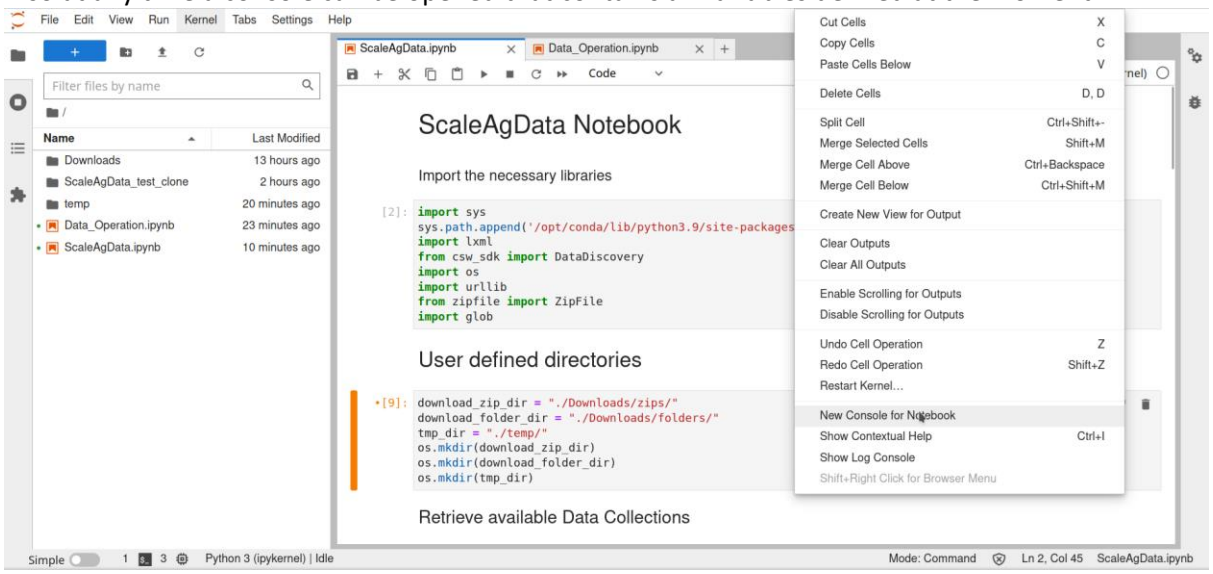


Figure 12 - Variable console of the R&I Environment (Virtual Lab)

3.3. Access to ScaleAgData catalogue

An SDK has been implemented to allow the functionality to allow users to discover collections and datasets in the archive4EO.

3.3.1. Class Structure

This is the class structure provided to the operator:

- **DataDiscovery:** contains discovery methods

  Methods:

  getDataCollections(): Returns an object of the class "DataCollections" with the list attribute filled with the collections available in the catalogue. Each collection in the list is an object of the class "DataCollection".

  search(filters={}, sortby='<parameter>', sort='<asc|desc>'): Returns an object of the class "EODatasets" with the results attribute filled with the list of datasets that meet the searching parameters.

  getSearchParameters(): Returns a list with the available search parameters.

  getSortParemeters(): Returns a list with the available sort parameters.

- **DataCollection:** containing attributes about a collection

  Attributes

  mission: collection's mission

  name: collection's name

DataCollections: containing a list of data collections and methods over that list

Attributes:

list: list of objects of class "DataCollection"

Methods:

searchCollections(mission='<substring>', name='<substring>'): returns a list of collections whose mission and name contain the substrings passed as arguments. Each collection on the list is an object of the class DataCollection.

EODataset: containing the attributes with the metadata values of an specific dataset

Attributes:

startTime

stopTime

spatial

url

quicklook

title

identifier

EODatasets: containing a list of datasets (results) and methods over that list

Attributes:

results: list of objects of class "EODataset"

Methods:

convertResultsToDict(): returns a list of dictionaries containing results information.

getDataset(id): returns an object of class "EODataset" with the id passed as argument (if it exists).

3.3.2. Use cases and examples

1. Search for data collections available in the archive:

```
from csw_sdk import DataDiscovery
discovery = DataDiscovery.DataDiscovery()
collections = discovery.getDataCollections()
```

Each collection (object from class "DataCollection") has mission and name as attributes, to access them:

```
collection = collections.list[x]
collection_mission = collection.mission
collection_name = collection.name
```

2. Filter collections by name and/or mission (substrings are allowed)

```
collections = collections.searchCollections(mission='S1B',name='S1PL1_SAFE')
collection = collections[x]
collection_mission = collection.mission
collection_name = collection.name
```

3. Search for datasets

```
datasets = discovery.search(filters={'file_name':'%ED0E%', 'mission': 'S2_', 'product_Type':'
S2__VITONDVI', start_time:'2022-05-01T00:00:00', stop_time:'2022-05-15T00:00:00', bbox:'1.45 49.418
6.95 52.0', ...}, sortby='creation_date', sort='asc,desc');
```

The allowed parameters for searching and sorting are in a configuration file. To check which parameters can be used, the user may use methods getSearchParameters() and getSortParameters() of DataDiscovery class:

```
valid_filters = discovery.getSearchParameters()
valid_sort_parameters = discovery.getSortParameters()
```

Select a Collection, defining the collection name and mission, start and stop time, bbox

```
collection_name = 'S2__VITONDVI'
collection_mission = 'S2_'
datasets = discovery.search(filters={'mission':collection_mission, 'product_type':collection_name, 'start_time':'2022-05-01T00:00:00', 'stop_time':'2022-05-15T00:00:00'})
datasets.results
```

```
[{identifier=None, title=S2__20220507_000000_C215D126_VITONDVI, start_time=2022-05-07T00:00:00, stop_time=2022-05-07T00:00:00, spatial=None, quicklook=None, url=https://data-lake.services4eo.com/storage/452b823d-b5fa-4b1c-88bd-d10a4b8d0648/output/S2__20220507_000000_C215D126_VITONDVI},
 {identifier=None, title=S2__20220505_000000_9AE01D3B_VITONDVI, start_time=2022-05-05T00:00:00, stop_time=2022-05-05T00:00:00, spatial=None, quicklook=None, url=https://data-lake.services4eo.com/storage/452b823d-b5fa-4b1c-88bd-d10a4b8d0648/output/S2__20220505_000000_9AE01D3B_VITONDVI},
 {identifier=None, title=S2__20220512_000000_70DCB658_VITONDVI, start_time=2022-05-12T00:00:00, stop_time=2022-05-12T00:00:00, spatial=None, quicklook=None, url=https://data-lake.services4eo.com/storage/452b823d-b5fa-4b1c-88bd-d10a4b8d0648/output/S2__20220512_000000_70DCB658_VITONDVI},
 {identifier=None, title=S2__20220515_000000_1CBC5008_VITONDVI, start_time=2022-05-15T00:00:00, stop_time=2022-05-15T00:00:00, spatial=None, quicklook=None, url=https://data-lake.services4eo.com/storage/452b823d-b5fa-4b1c-88bd-d10a4b8d0648/output/S2__20220515_000000_1CBC5008_VITONDVI},
 {identifier=None, title=S2__20220510_000000_70955634_VITONDVI, start_time=2022-05-10T00:00:00, stop_time=2022-05-10T00:00:00, spatial=None, quicklook=None, url=https://data-lake.services4eo.com/storage/452b823d-b5fa-4b1c-88bd-d10a4b8d0648/output/S2__20220510_000000_70955634_VITONDVI},
 {identifier=None, title=S2__20220502_000000_CB4A4FC1_VITONDVI, start_time=2022-05-02T00:00:00, stop_time=2022-05-02T00:00:00, spatial=None, quicklook=None, url=https://data-lake.services4eo.com/storage/452b823d-b5fa-4b1c-88bd-d10a4b8d0648/output/S2__20220502_000000_CB4A4FC1_VITONDVI}]
```

Figure 13 - Data Discovery Python code snippet

4. Convert list of EODataset objects to a list of dictionaries

```
results = datasets.convertResultsToDict()
#Example to use results with pandas
import pandas as pd
input_metadata = pd.DataFrame.from_records(results)
input_metadata.head()
```

Convert list of EODataset objects to a list of dictionaries

```
results = datasets.convertResultsToDict()
#Example to use results with pandas
import pandas as pd
input_metadata = pd.DataFrame.from_records(results)
input_metadata.head()
```

	identifier	title	start_time	stop_time	spatial	quicklook	url
0	None	S2__20220507_000000_C215D126_VITONDVI	2022-05-07T00:00:00	2022-05-07T00:00:00	None	None	https://data-lake.services4eo.com/storage/452b...
1	None	S2__20220505_000000_9AE01D3B_VITONDVI	2022-05-05T00:00:00	2022-05-05T00:00:00	None	None	https://data-lake.services4eo.com/storage/452b...
2	None	S2__20220512_000000_70DCB658_VITONDVI	2022-05-12T00:00:00	2022-05-12T00:00:00	None	None	https://data-lake.services4eo.com/storage/452b...
3	None	S2__20220515_000000_1CBC5008_VITONDVI	2022-05-15T00:00:00	2022-05-15T00:00:00	None	None	https://data-lake.services4eo.com/storage/452b...
4	None	S2__20220510_000000_70955634_VITONDVI	2022-05-10T00:00:00	2022-05-10T00:00:00	None	None	https://data-lake.services4eo.com/storage/452b...

Figure 14 -Conversion of list of EO Dataset output into a list of dictionaries Python code snippet

5. Select a dataset from the list

```
product= datasets.results[x]
product= datasets.getDataset(id)
6. Access to dataset
link_to_storage = product.url
```

6. Download products and store them in a folder

Download products and store them in a folder inside virtual lab

```
for product in subset:
    url = product.url
    product_download_path = os.path.join(os.path.join(download_dir, url.split("/")[-1]))
    print(product_download_path)

    # check if not already downloaded
    if not os.path.isfile(product_download_path):
        print("Downloading from Archive product " + product.title + '... ')
        urllib.request.urlretrieve(product.url, product_download_path)

    else:
        print('Product: '+ product.title + ' already downloaded in path: ' + product_download_path)

print("Download has finished")
```

```
./Downloads/zips/S2__20220430_000000_FE1D9F21_VITONDVI
Downloading from Archive product S2__20220430_000000_FE1D9F21_VITONDVI...
./Downloads/zips/S2__20220515_000000_1CBC5008_VITONDVI
Downloading from Archive product S2__20220515_000000_1CBC5008_VITONDVI...
./Downloads/zips/S2__20220510_000000_70955634_VITONDVI
Downloading from Archive product S2__20220510_000000_70955634_VITONDVI...
Download has finished
```

Figure 15 - Example a data access and download code Python code snippet

After the download of product finishes you can take a look into the products inside the Dowload directory:

List the products inside the Download Directory

```
os.listdir(download_dir)
```

```
['S2__20220515_000000_1CBC5008_VITONDVI',
 'S2__20220510_000000_70955634_VITONDVI',
 'S2__20220430_000000_FE1D9F21_VITONDVI']
```

Figure 16 - Python code snippet of list of EO products in the download directory

7. Extract Files from Zip downloaded

The downloaded products will always be zip files so the next steps explain how to unzip and extract all files and store them in different folders.

Unzip files

```
zip_filepaths = glob.glob(download_zip_dir+"*")
zip_filepaths
```

```
['./Downloads/zips/S2__20220515_000000_1CBC5008_VITONDVI',
 './Downloads/zips/S2__20220510_000000_70955634_VITONDVI',
 './Downloads/zips/S2__20220430_000000_FE1D9F21_VITONDVI']
```

Extract files from the zip files

```
for zip_path in zip_filepaths:
    with ZipFile(os.path.join(zip_path), 'r') as zip_ref:
        # printing all the contents of the zip file
        zip_ref.printdir()
        zip_ref.extractall(os.path.join(download_folder_dir, zip_path.split("/")[-1]))
```

```
File Name                                             Modified             Size
metadata.json                                  2023-01-17 15:37:24          501
S2__20220515_000000_1CBC5008_VITONDVI.tif      2023-01-17 15:37:24         9190
File Name                                             Modified             Size
metadata.json                                  2023-01-17 15:37:04          501
S2__20220510_000000_70955634_VITONDVI.tif      2023-01-17 15:37:00         9843
File Name                                             Modified             Size
S2__20220430_000000_FE1D9F21_VITONDVI.tif      2023-01-17 15:37:14         9822
metadata.json                                  2023-01-17 15:37:14          501
```

Figure 17 - Example of Python code snippet to unzip downloaded products

3.3.3. Data Operations

After having data downloaded and stored in the respective directories, operations can be performed on the data. An example of this is available in the JupyterLab in the Jupyter notebook named "Data_Operation.ipynb" that is going to be explained below.

1. Define the Directories to retrieve the data and to store outputs created

2. Search for the data in the directories

Search for raster (".tif") files in the directories and store the paths to a list:

```
images=[]
for file in filepaths:
    images.append(glob.glob(os.path.join(file,"*.tif"))[0])
```

```
images
```

```
['./Downloads/folders/S2__20220515_000000_1CBC5008_VITONDVI/S2__20220515_000000_1CBC5008_VITONDVI.tif',
 './Downloads/folders/S2__20220510_000000_70955634_VITONDVI/S2__20220510_000000_70955634_VITONDVI.tif',
 './Downloads/folders/S2__20220430_000000_FE1D9F21_VITONDVI/S2__20220430_000000_FE1D9F21_VITONDVI.tif']
```

Figure 18 - Example of a data retrieval and store Python code snippet

3. Read the images, produce some operation and see the results

Read the images and produce the operation that you desire. In the image below a simple example is showcased of a mean operation for 3 raster images for the same area of interest.

The produced results are stored in a folder inside JupyterLab.

```
array_list=[]
for image in images:
    img=io.imread(image)
    array_list.append(skimage.util.img_as_ubyte(img))
```

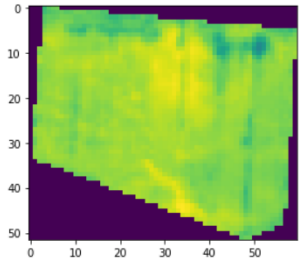
```
array_out=np.mean(array_list, axis=0)
```

```
np.mean(array_list)
```

```
60.988247863247864
```

```
plt.imshow(array_out)
```

```
<matplotlib.image.AxesImage at 0x7fc13a1718d0>
```




```
io.imsave(os.path.join(outputs_dir,"test.tif"), arr=array_out)
```

Figure 19 - Example of a data manipulation Python code snippet

3.3.4. Commit changes to Git remote repository

Changes to a Git repository can be made through the terminal. The steps are:

1. Open a terminal window
2. Change directory to your Git repository.

```
cd my_git_directory
```

3. Stage the changes. You can add all using "." or specify a single or folder.

```
git add .
```

4. Commit changes to your Git repository.

```
git commit -m "My commit message"
```

5. Push changes to the remote repository. You will have to type the username and password.

```
git push
```