Copernicus Access Platform Intermediate Layers Small Scale Demonstrator

D3.3 CANDELA Platform v1

Document Identification			
Status	Final	**Due Date**	31/12/2018
Version	1.3	**Submission Date**	31/05/2018

Related WP	WP3	**Document Reference**	D3.3
Related Deliverable(s)	D3.1 D3.5	**Dissemination Level (*)**	PU
Lead Participant	Atos France	**Lead Author**	Fabien CASTEL (Atos France)
Contributors	Anne-Sophie TONNEAU (Atos France) Jean-François ROLLAND (Atos France)	**Reviewers**	Michelle Aubrun (TAS FR)

Keywords:
Cloud, Platform as a Service, data processing, earth observation

This document is issued within the frame and for the purpose of the CANDELA project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 776193. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

Document Information

List of Contributors	
Name	Partner
Fabien CASTEL	Atos France
Anne-Sophie TONNEAU	Atos France
Jean-François ROLLAND	Atos France

Document History			
Version	Date	Change editors	Changes
0.1	04/12/2018	Fabien Castel (ATOS FR)	Initial Table of Content
0.2	11/12/2018	Anne-Sophie Tonneau (ATOS FR)	Version for partner review
0.3	13/12/2018	Anne-Sophie Tonneau (ATOS FR)	Version for quality review
0.4	18/12/2018	Juan Alonso (ATOS ES)	Quality Assessment
1.0	18/12/2018	Jose Lorenzo (ATOS ES)	Added Annex on WPSLib documentation Final revision for submission
1.1	25/04/2019	Jean-François Rolland (ATOS FR)	Update of the document to handle remarks issued at the first-year review
1.2	30/05/2019	Juan Alonso (ATOS ES)	Quality Assessment
1.3	31/05/2019	Jose Lorenzo (ATOS ES)	Final revision before re-submission

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Jean-François Rolland (ATOS FR)	29/05/2019
Quality manager	Juan Alonso (ATOS ES)	30/05/2019
Project Coordinator	Jose Lorenzo (ATOS ES)	31/05/2019

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
CPU	Central processing unit
EC	European Commission
D3.3	Deliverable number 3 y belonging to WP3
DIAS	Copernicus Data and Information Access Service
HTML	HyperText Markup Langage
OGC	Open Geospatial Consortium
PaaS	Platform as a Service
SAR	Synthetic Aperture Radar
WP	Work Package
WPS	Web Processing Service, OGC standard for geospatial processing services

Executive Summary

This report is an accompanying document to the CANDELA platform with the objective to list the content of the first version of the platform and to provide a user guide describing how to use all the features provided by the platform.

The current version of the document (the first release took place in December 2018) has been updated following the requirements from the first project review meeting. The main evolution in the second version of D3.3 is the addition of section two. After the previous version of the platform, several tools have been integrated.

- The semantic search tool from IRIT has been deployed on the platform. Technically it consists of two components: the website hosted on a Tomcat server and a geospatial database based on PostGIS. These two components are deployed as two different docker containers.
- The DLR algorithm performing semantic classification on earth observation products has been integrated on the platform. This needs to access to a monet database. An instance of this database has been deployed in a separate container.
- The JupyterLab notebook has been modified to include client libraries for both monetDB and PostGIS. A command line client for monetDB has also been included to this JupyterLab environment.
- In addition to the existing change detection algorithm on optical images provided by TAS France a new change detection algorithm for SAR (Synthetic Aperture Radar) images from TAS Italy has been integrated.

The document is structured as follows:

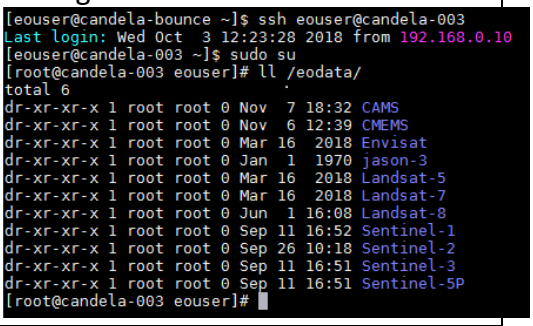
- In section 1, a release note presents the components that are available for this version, and how to access them.
- In section 2 it is presented the functional architecture of the CANDELA platform and how the different components implement this architecture. It is also described the links between CANDELA and past or present projects like SparkInData, EO4wildlife [7], or EUXDAT [6]. Finally, section 2 also includes a chapter about compatibility of CANDELA with other DIAS (Copernicus Data and Information Access Service).
- Section 3 presents first a user guide to help user to familiarize with the development environment, then it describes the CANDELA library, that is a tool to interact with the data analytics and management services offered by the platform.

1 Release Note

The following table details the components that are delivered, their version and access point. The detailed functionality of each component is presented in the document D3.5[5].

Table 1: V1 delivered components (May-2019)

Component	Version	Use	Access point
GeoServer	2.14RC	exposes the data processing algorithms as WPS processes	http://185.178.85.62/geoserver/web/ https://185.178.85.62/geoserver/web/
JupyterHub	0.9.4	manages multi-users access to the platform	http://185.178.85.62/hub/
Keycloak	4.4.0.Final	authentication role	http://185.178.85.62/auth/admin/master/console/#/realms/master
Notebook Server-JupyterLab	0.34.0	offers a dedicated environment with a web interface to user to interact with the platform	http://185.178.85.62/user/test%20user/lab
Python3 kernel	3.6.6	Python 3 is installed in the user environment	
MonetDB	v11.33.3	Database used by Eominer tool from DLR	Accessible from jupyter hub using: - Pymonetdb in a python environment - Mclient from a terminal
PostGIS	9.6.11	Database used by semsearch tool from Irit	Accessible from jupyter hub using psycopg2 library Accessible using any PostGIS client on this address 185.178.85.62 on port 30023
Semsearch		Web based application hosted by a tomcat server on the candela platform	http://185.178.85.62/semsearch

Component	Version	Use	Access point
CreoDIAS connector		access to CreoDIAS data	The CreoDIAS data is accessible through a s3fs volume: 
Data processing algorithms	1.0	algorithms for data analytics provided by candela partners	The algorithms are available through: GeoServer: by requesting them with WPS standard User's Notebook: by requesting them with the custom WPSLib library
REST backend	1.0	Provides access to monet DB through REST services	https://185.178.85.62/rest/
WPSLib	1.0	Python library for interacting with the WPS processes	The custom library is available through user's Notebook

2 Functional architecture

In this section it is described the functional architecture of the CANDELA platform. It is explained in a second part how the components used on the CANDELA platform cover the needs of the functional architecture. In the third part it is presented the adaptation needed to deploy this architecture on a different DIAS (Mundi).

2.1 Description of the functional architecture

This functional architecture of CANDELA platform shows its interface between the DIAS-like back-office services providing data, storage and processing resources and front office services dedicated to applications development and operation. The Figure 1 represents the CANDELA platform and its link with back office services (DIAS) and front office services.

CANDELA platform represents the intermediate layers where the workflow starts with the task definition in the so-called "B1. Job manager". This corresponds to the *what? who? where?* questions. From that definition, data and application are retrieved in the "B2. Retrieve data", "B4. Data Preparation" and "B3. Prepare Analytics" functions. Then, Data is processed using DIAS processing resources in the step represented by the box "B5. Process Analytics". After each data source has been processed separately, the "B6. Align Information" function performs the tasks of geo-temporal alignment and fusion. Finally, "B7. Push information" collects, formats and transfers extracted information to the Front-office "C3. Information Exploitation" on request of "C2. Search & Discovery".

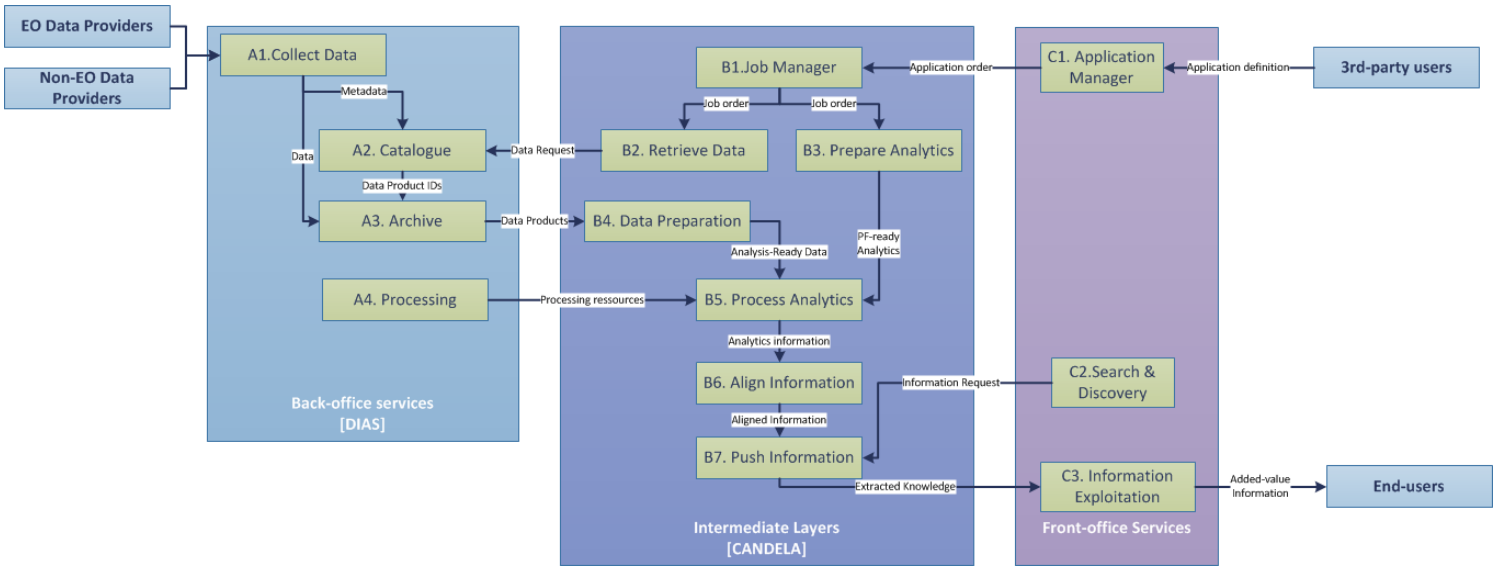


Figure 1: Candela functional architecture

2.2 Technical components

The Figure 2 represents the basic components deployed on the CANDELA platform. All these components are deployed as docker containers over a Kubernetes layer. The details of the system integration are presented in the deliverable D3.5 [5].

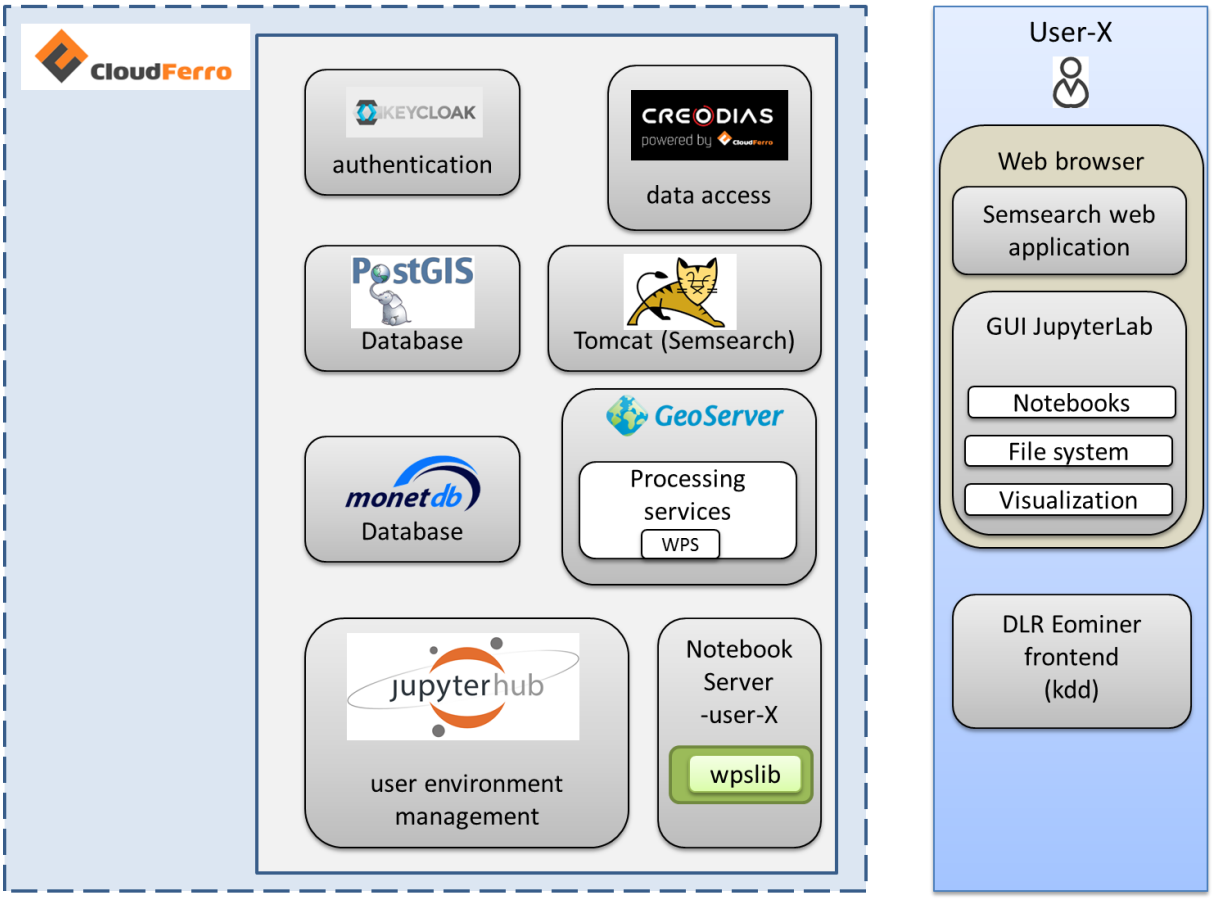


Figure 2: Technical architecture

The CANDELA Functional Architecture (represented in Figure 1) and the relationship with the Technical architecture (Figure 2) is presented below.

- B1 to B4 steps from the functional architecture are supported by the Jupyter environment. The application manager (C1) corresponds to the Jupyter frontend available for the user. From this user interface, the user can setup process analytics, search for satellites products, copy and prepare the data for analytics processing. The CreoDIAS data access allow the data research and data retrieval for process analytics.
- The GeoServer and algorithms from CANDELA partners implements B5, processing analytics and are executed.
- It should be noted that the *Align information* part or data fusion is not yet deployed.
- The *search and discovery* elements (C2) are implemented by the semsearch web application. This application exploits the data base PostGIS represented by the B7 building block in the functional architecture.

2.3 Links with other projects and DIAS

2.3.1 Legacy and synergies

The architecture of the CANDELA platform inherits elements from past projects and has numerous components in common with other ongoing projects.

The EUXDAT[1] [6] cloud environment relies for its low-level layers on a PaaS system provided by Atos. This system has been initially designed in the frame of the SparkInData[2] project launched in 2015, and it has been continuously reused and improved in many H2020 projects (EO4wildlife[3] [7], EUXDAT) and in the Mundi DIAS platform. It provides a built-in PaaS based on containerization using Docker and Kubernetes. Kubernetes is in charge of the cluster management, it deals across the cloud cluster with application deployment, resource management, network configuration and service discovery. All the Docker/Kubernetes environment is deployed and configured through specific scripts that automatize the process.

The first studies of deploying GeoServer and use its processing pipeline capabilities of scientific algorithm started in SparkInData, it has been reused and enhanced in EO4wildlife.

Several means were provided by the SparkInData platform to design and execute services workflows, also named processing pipelines. Each one of them provides a different level of flexibility and requires from the processing developer more or less time to integrate its piece of software into the platform. The WPS Processing Pipeline Service is one of these solutions. It aims to make as simple as possible the integration of a given processing block (or processing service) and expose it through a WPS interface.

The WPS standard (version 2.0) enables in particular:

- To document processes and make them discoverable.
- To monitor the execution of the process (execute synchronously or asynchronously, dismiss the WPS process)
- To chain processing services with processing blocks

This component and the way algorithms are embedded as OGC WPS processes have been adapted for CANDELA.

The authentication components have been developed for both EUXDAT and CANDELA platforms.

The deployment, configuration, and customization of Jupyter Hub have synergies with EUXDAT and DIAS Mundi projects. Jupyter hub is a standard component when it comes to providing a development

[1] **EUXDAT**, EU H2020 (e-INFRA), 2017-2020 (www.euxdat.eu). *European e-Infrastructure for extreme data analytics in sustainable development*, enabling users to fully benefit from underlying High Processing capacities to explore new methods, build new innovative services, perform predictions and simulations with extremely large and heterogeneous datasets. CORDIS: https://cordis.europa.eu/project/rcn/216632/factsheet/en

[2] **SparkInData**, a National French R&D project, led by Atos, to build a cloud platform assembling all sources of Earth Observation data, selected under the call for "Cloud Computing & Big Data" projects issued through the "Investing for the Future" programme. Consisting of 11 partners: Atos, TerraNis, Geomatys, Geosigweb, Mercator-Ocean, CNES-Centre National d'Etudes Spatiales, IGN-Institut national de l'information géographique et forestier, BRGM-Bureau des Recherches Géologiques et Minières, IRIT-Institut de Recherche Informatique de Toulouse – Université Paul Sabatier, EI Purpan-Ecole d'Ingénieurs de Purpan, Aerospace Valley

[3] **EO4wildlife**, EU H2020 (SPACE), 2016-2018 (www.eo4wildlife.eu). Funded under the topic "Stimulating wider research use of Copernicus Sentinel Data", EO4wildlife aimed to build a platform for wildlife monitoring integrating Copernicus and ARGOS data, plus additional real time thematic databank portals. EO4wildlife was coordinated by Atos, being also the key partner providing the data platform based on the internal asset SparkInData. CORDIS: https://cordis.europa.eu/project/rcn/199237/factsheet/en

environment to the user. In each of these projects the needs are not the same and each project can beneficiate from the customization created for other ones.

Table 2 summarizes the links between components of CANDELA, the functional architecture and the different projects where they have been used.

Table 2: Function coverage and links with other projects

Component	Functionality (Ref. Figure 1)	Connected with project
Kubernetes		SparkInData, EO4wildlife, EUXDAT, DIAS Mundi
GeoServer	B5	SparkInData, EO4wildlife
JupyterHub	B1, B2, B3, B4, C1	EUXDAT, DIAS Mundi
Keycloak		EUXDAT
MonetDB		
PostGIS	B7	
Semsearch	C2	

2.3.2 How to adapt to another DIAS

The CANDELA platform can be deployed on any DIAS that allows to deploy a PaaS on virtual machines. The platform is based on a set of virtual machines hosted on a DIAS. Once these machines have been provisioned the first step is to deploy and configure Kubernetes.

The platform itself can be viewed as a set of Docker containers executed on top of Kubernetes. The configuration of the deployment may change a little bit considering possible restriction of a specific DIAS (range of available ports for example).

The main difference would be how the platform access to the data. Each DIAS propose different ways to access to satellites products. Part of the method used to search and retrieve images would need to be redefined.

3 User Guide

3.1 Development Environment

A Jupyter Notebook development environment is available for users, including a Python 3 kernel.

The Notebook Server exposes a web interface called JupyterLab, that allows to interact with the platform through:

- a file explorer
- a terminal
- Notebook files

When the user logs in the JupyterLab user interface, it accesses the functionalities offered by Notebook Server, as in the following figure:

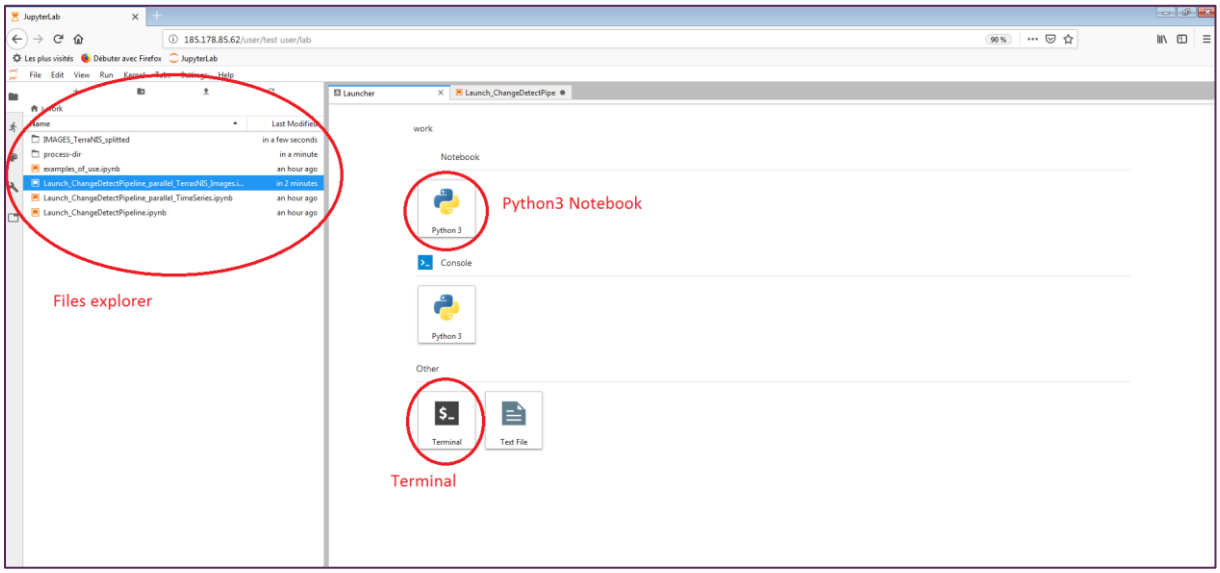


Figure 3: JupyterLab user interface after log in

3.1.1 File explorer

When navigating to the root of the file system, it has three folders available:

- **/lib**: contains the python WPSLib library described in section 3.1.4.
  User has only the right to read this folder content.
- **/public**: contains data that is public, meaning that it is common to all users.
  - Some examples of Notebooks:

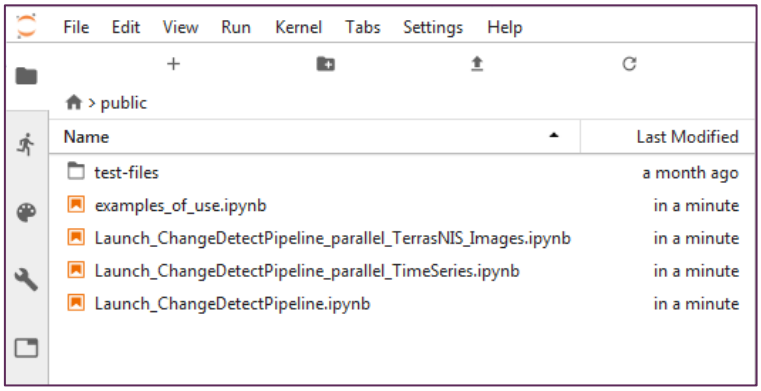


Figure 4: Public notebooks

- Some test data such as images provided for the TerraNIS use case in */public/test-files/Images/IMAGES_TerraNIS/*:

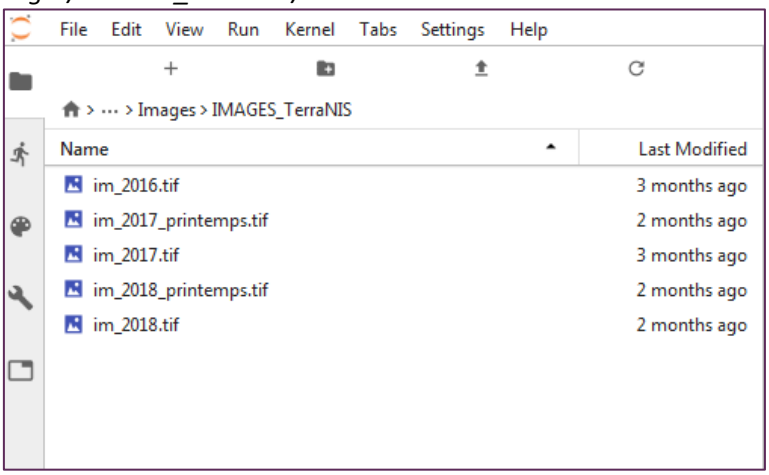


Figure 5: Public datasets

User has only the right to read this folder content.

- **/work**: this is the private workspace of the user. It contains for convenience a */process-dir* that is supposed to be used as output directory for the processes.
  User has the right to read and write in this folder.
- **Readme.md**: it contains some notes about how to use the JupyterLab, as shown in Figure 6.

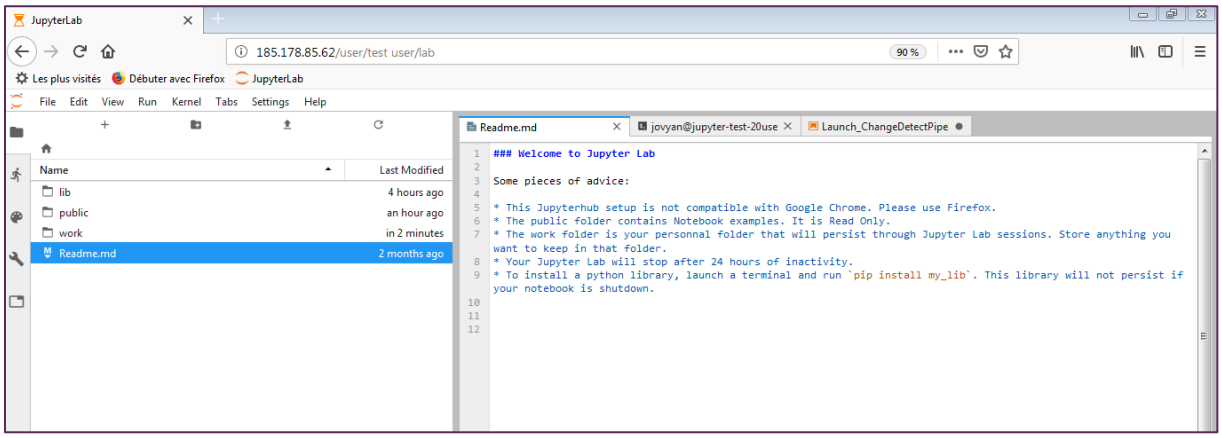


Figure 6: JupyterLab - root of the file system

3.1.2 Terminal

As explained in the *Readme.md* file, the user can install new Python libraries on his environment, using pip [1] or Conda [2].

To open a terminal at any time, click on the "+" symbol to open a new window, and then on "Terminal".

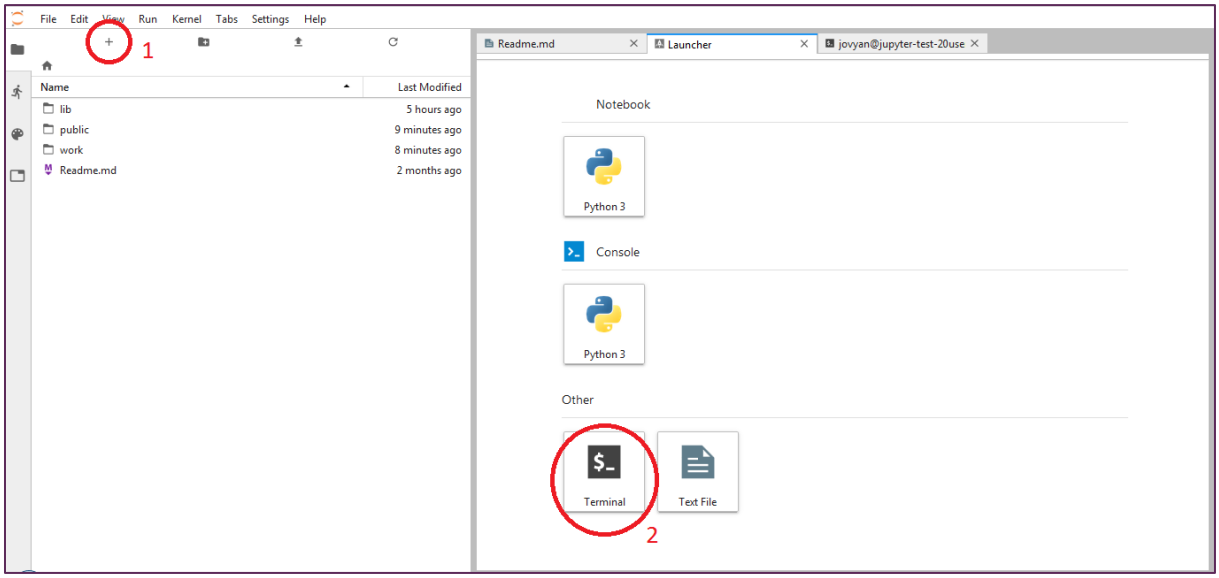


Figure 7: JupyterLab - open a terminal

Then it is possible to install the python library.

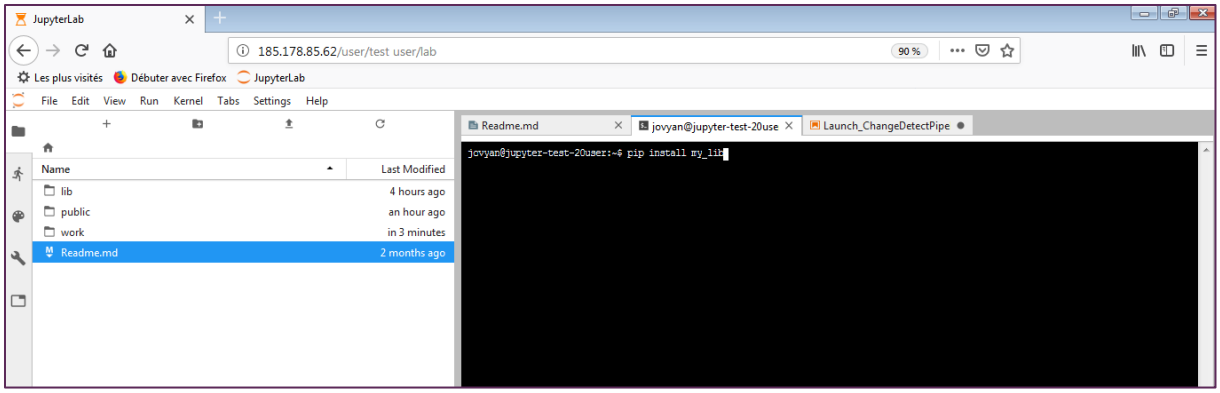


Figure 8: JupyterLab - install a library

3.1.3 Notebook file

Using the Python3 kernel, the user can write Notebook files to interact with the platform.
To create a new Notebook file, click on the "+" symbol, and then on "Python3" below "Notebook".

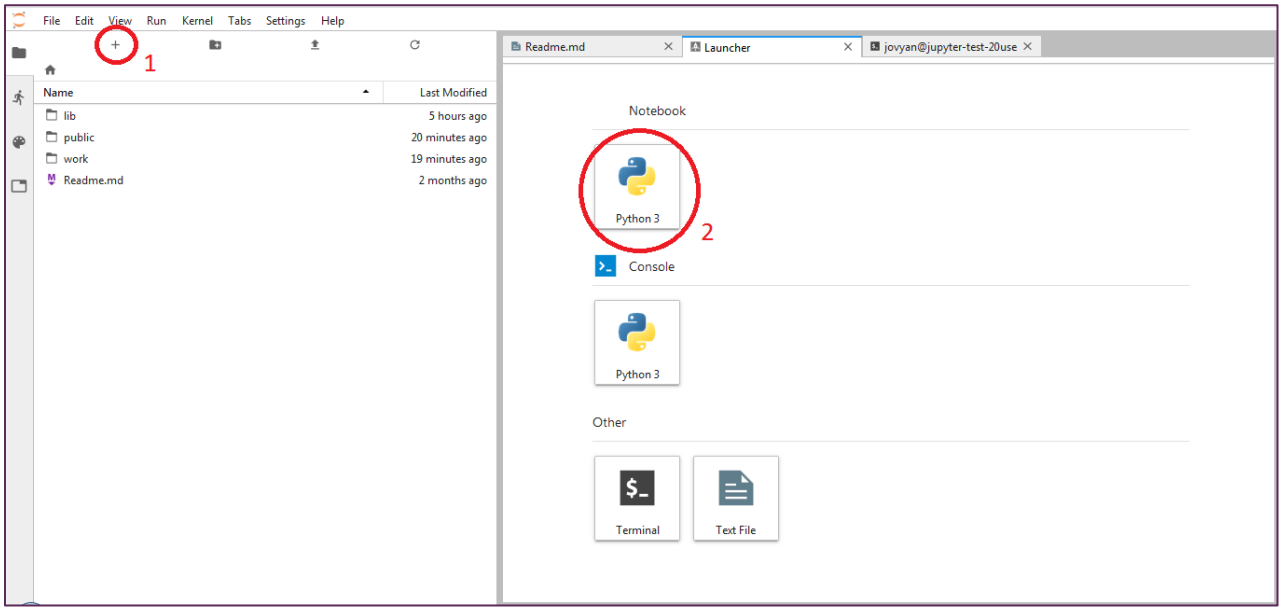


Figure 9: JupyterLab - Create a new Notebook

A new Notebook file is created. User can write Python3 code, execute it and see the outputs traces:

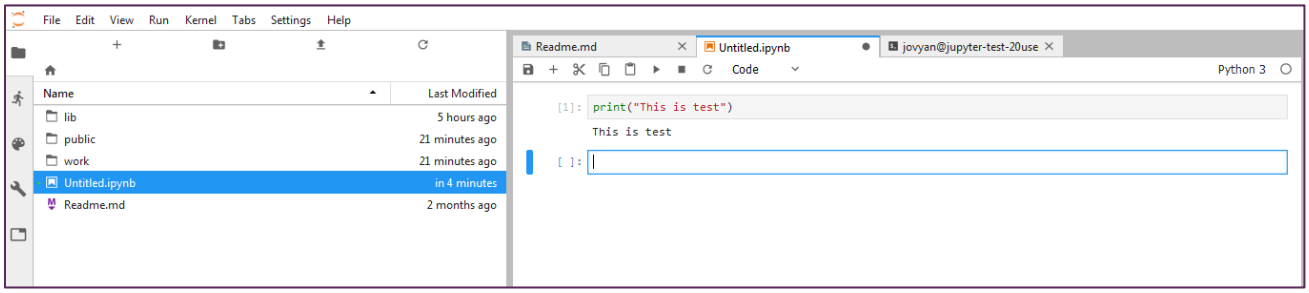


Figure 10: JupyterLab - New Notebook created

3.1.4 Python 3 console and Text File

For interacting with the Python environment, user can also use the Python 3 Console. The interface also offers the possibility to create text files (see Figure 9: JupyterLab - Create a new Notebook).

3.2 CANDELA Library

A custom Python library called WPSLib has been developed to interact with the platform, for manipulating the WPS processing services. It offers facilities for launching processes and interact with them. It is built on top of the OWSLib [3] that is a Python library for programming with OGC web services.

The documentation of WPSLib can be found here [4]. It is also available in pdf format (and attached as annex at the end of the Deliverable):

Adobe Acrobat Document

4 Conclusion

This document outlines the content of the first release of the CANDELA platform and provided a user guide.

This first version includes a single data analytics chain. This version has been updated tacking into account the requirements from the first project review meeting held in January 2019, by including the "Functional Architecture" chapter. The next versions of the platform shall include additional data analytics algorithms. The facilities for searching and getting CreoDIAS data will also be improved.

Moreover, the key feature expected from the CANDELA platform is to be able process large amount of data with a high level of performances. The first version of the platform is not compliant with this expectation, as it relies on a static cloud environment with limited computation resources. The main goal of the next version of the platform will be to implement solutions relying on cloud mechanism to scale up processes execution and allocate on-demand memory and CPU resources adapted to each process specific needs.

References

[1] Pip, the recommended tool for installing Python packages, https://pypi.org/project/pip/ , retrieved 2018/12/07

[2] Conda, an open-source Python package management system, https://conda.io/docs/ , retrieved 2018/12/07

[3] OWSLib Python library for OGC protocols, https://geopython.github.io/OWSLib/, retrieved 2018/12/10

[4] WPSLib documentation http://185.178.85.62/wpslib/index.html, retrieved 2018/12/10

[5] F. Castel, "D3.5 System Integration and Validation Test Plan", Deliverable of the CANDELA Project, 2018, retrieved 2019/05/24

[6] EUXDAT project, http://www.euxdat.eu/ retrieved 2019/05/24

[7] EO4wildlife project, http://eo4wildlife.eu/ retrieved 2019/05/29

WPSLib Documentation

Release 1.0

Atos Fr

Dec 06, 2018

CHAPTER

ONE

INDICES AND TABLES

- genindex
- search

CHAPTER

TWO

WPSLIB MAIN

This module is based on the OWSLib, a Python package for client programming with OGC web service interface standards. WPSLib helps manipulating WPS standard, it facilitates this interactions with WPS processing services : process description, process running, process chaining or launching several processes in parallel

https://geopython.github.io/OWSLib/

wpslib.**getCapabilities**()
Function for getting the list of WPS processes available in Geoserver

Returns list of WPS processes

Return type str

wpslib.**getLogger**(*stdout*, *log_file=''*)
Function for getting a logger

Parameters

- **stdout** (*bool*) – defines if the logger should write traces on stdout.
- **log_file** (*str*) – (optional) path to the file where logger should write traces (no logs file if empty).

Returns the logger

Return type logger

wpslib.**getOutputValue**(*execution*, *outputName*)
Function for getting an output value of an execution that is done

Parameters

- **execution** (*execution*) – the execution
- **outputName** (*str*) – the output name

Returns the output value

Return type str

wpslib.**getProcessDescription**(*identifier*)
Function for getting the description of a WPS process from Geoserver

Parameters **identifier** (*str*) – the identifier of the process

Returns WPS description of the process

Return type str

wpslib.**runPipelinesInParallel**(*pipelineList*, *logger*, *max_processes=0*)
Function for running pipelines of WPS processes in parallel

Parameters

- **pipelineList** (*list of dict*) – the list of pipelines to run at the same time
- **logger** (*logger*) – the logger to use
- **max_processes** (*int*) – the maximum number of pipelines to run at the same time (default value is 0, meaning no maximum)

Returns the list of OWSLib executions

Return type list of execution

wpslib.**runProcessesInParallel**(*processList*, *logger*, *max_processes=0*)

Function for running WPS processes in parallel

Parameters

- **processList** (*list of dict*) – the list of processes to run at the same time
- **logger** (*logger*) – the logger to use
- **max_processes** (*int*) – the maximum number of processes to run at the same time (default value is 0, meaning no maximum)

Returns the list of OWSLib executions

Return type list of execution

Examples: processList example: processList = [

{'PROCESS_ID': 'splitimages-im_2016', 'IDENTIFIER': 'candela:SplitImagesProcessing', 'INPUTS': [('IMAGES', '/public/test-files/Images/IMAGES_TAS/im_2016.tif'), ('PROCESS_ID', 'splitimages-im_2016'), ('OUTPUT_FOLDER', '/work/IMAGES_TAS_splitted/'), ('CONFIG_FOLDER', '/work/IMAGES_TAS_splitted/config'), ('N_SPLITS', '3')], 'OUTPUTS': [('outpath', True), ('logFiles', True)]}, {'PROCESS_ID': 'splitimages-im_2017', 'IDENTIFIER': 'candela:SplitImagesProcessing', 'INPUTS': [('IMAGES', '/public/test-files/Images/IMAGES_TAS/im_2017.tif'), ('PROCESS_ID', 'splitimages-im_2017'), ('OUTPUT_FOLDER', '/work/IMAGES_TAS_splitted/'), ('CONFIG_FOLDER', '/work/IMAGES_TAS_splitted/config'), ('N_SPLITS', '3')], 'OUTPUTS': [('outpath', True), ('logFiles', True)]}

]

wpslib.**runSinglePipeline**(*pipeline*, *logger*)

Function for running a pipeline of WPS process

Parameters

- **pipeline** (*list of dict*) – the pipeline definition
- **logger** (*logger*) – the logger to use

Returns the OWSLib execution of the last process ran

Return type execution

Examples: Pipeline example: Pipeline = [

{'PROCESS_ID': 'changedetection-timeseries', 'IDENTIFIER': 'candela:ChangeDetectionProcessing', 'INPUTS': [('IMAGES', '/public/test-files/Images/Harbour/TimeSeries/'),

('PROCESS_ID', 'changedetection-timeseries'), ('OUTPUT_FOLDER', '/work/process-dir/change_detection_pipeline/'), ('CONFIG_FOLDER', '/work/process-dir/change_detection_pipeline/config/'), ('OUTPUT_FILENAME', 'change_detect.tif')],

'OUTPUTS': [('outpath', True), ('logFiles', True)], 'PREVIOUS_PROCESS': '', 'INPUT': '', 'PREVIOUS_VALUE': ''}, {'PROCESS_ID': 'changeindex-timeseries', 'IDENTIFIER': 'candela:ChangeIndexProcessing', 'INPUTS': [('PROCESS_ID', 'changeindex-timeseries'),

('OUTPUT_FOLDER', '/work/process-dir/change_detection_pipeline/'), ('CONFIG_FOLDER', '/work/process-dir/change_detection_pipeline/config/'), ('OUTPUT_FILENAME', 'change_index.tif')],

'OUTPUTS': [('outpath', True), ('logFiles', True)], 'PREVIOUS_PROCESS': 'changedetection-timeseries', 'INPUT': 'IMAGE', 'PREVIOUS_VALUE': 'outpath'}, {'PROCESS_ID': 'changeclustering-timeseries', 'IDENTIFIER': 'candela:ChangeClusteringProcessing', 'INPUTS': [('PROCESS_ID', 'changeclustering-timeseries'),

('OUTPUT_FOLDER', '/work/process-dir/change_detection_pipeline/'), ('CONFIG_FOLDER', '/work/process-dir/change_detection_pipeline/config/'), ('OUTPUT_FILENAME', 'change_clustering.tif')],

'OUTPUTS': [('outpath', True), ('logFiles', True)], 'PREVIOUS_PROCESS': 'changeindex-timeseries', 'INPUT': 'IMAGE', 'PREVIOUS_VALUE': 'outpath'}

]

wpslib.**runSingleProcess**(*identifier*, *inputs*, *output*, *logger*)

Function for running a WPS process

Parameters

- **identifier** (*str*) – the identifier of the process
- **inputs** (*list of tuples*) – the inputs of the process
- **output** (*list of tuples*) – the expected output of the process
- **logger** (*logger*) – the logger to use

Returns the OWSLib execution of the process

Return type execution

Examples: Inputs example: inputs = [("IMAGES","/public/test-files/Images/IMAGES_TAS/im_2016.tif"),

("PROCESS_ID","splitimages-im_2016"), ("OUTPUT_FOLDER","/work/IMAGES_TAS_splitted/"), ("CONFIG_FOLDER","/work/IMAGES_TAS_splitted/config"), ("N_SPLITS","6")

]

Output example: outputs = [('outpath',True),('logFiles',True)]