Real Use Case Session (Part I)

A practical introduction to ECASLab and Ophidia for data analytics

Donatello Elia | CMCC Foundation & University of Salento
Sandro Fiore | CMCC Foundation & University of Salento
On behalf of the ECAS Team

EOSC-hub receives funding from the EU's Horizon 2020 research and innovation programme under grant agreement No. 777536.

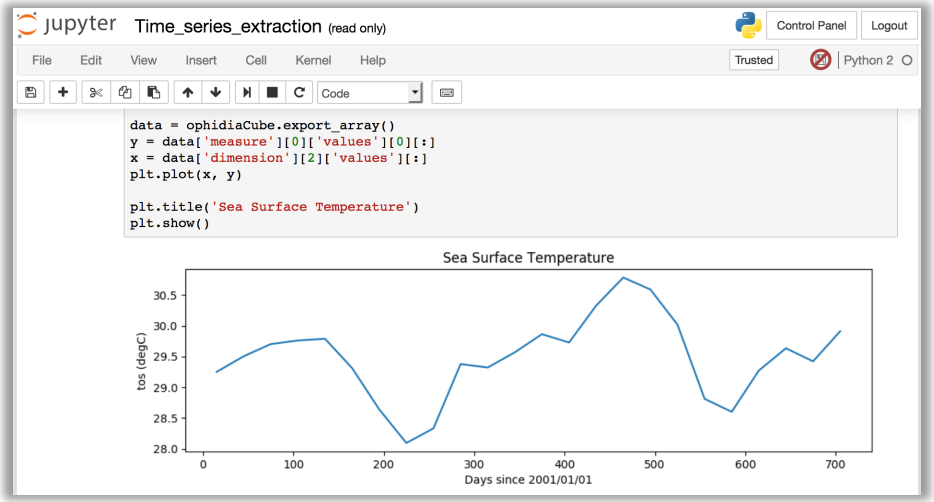
Outline

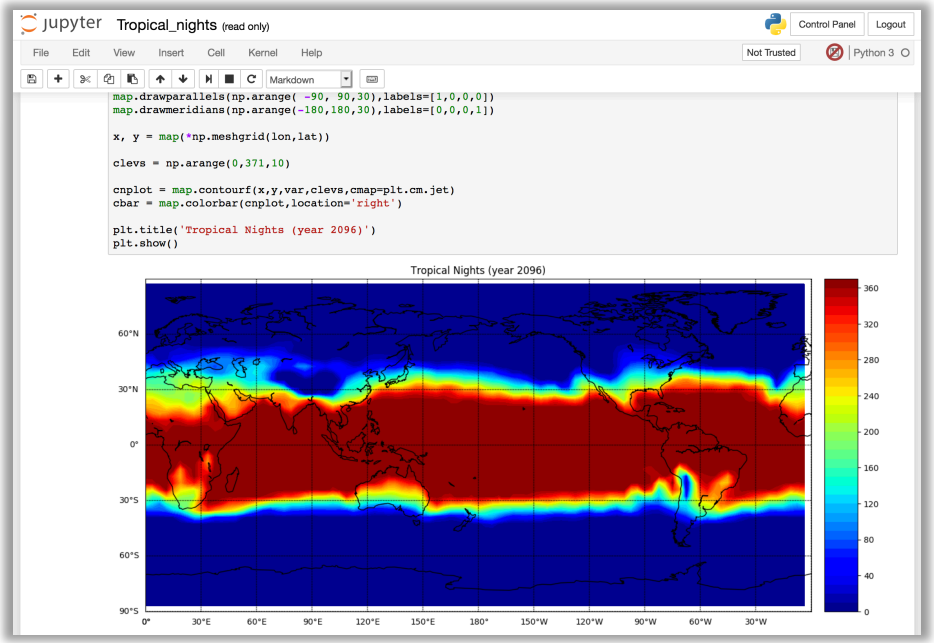
- ✔ ***ECASLab***
- ✔ *Server-side data analytics*
- ✔ *ECAS Terminal*
- ✔ *PyOphidia*
- ✔ *Live demo @ ECASLab*

ECASLab: Python environment for Data Science

ECASLab provides a ready-to-use environment based on JupyterHub, Ophidia and other services from EUDAT and EGI, bundled with a wide set of well-known Python scientific and data managment modules, some examples:

- ✓ NumPy, SciPy, Pandas
- ✓ NetCDF, PyOphidia
- ✓ Matplotlib, basemap, Cartopy





ECASLab: Jupyter file browsing

Control Panel | Logout

Files | Running | Clusters

Select items to perform actions on them.

Upload | New | ↻

	Name	Last Modified
data		6 months ago
Demo_EUDAT_PRACE_SummerSchool		a minute ago
Hands-on-EUDAT_PRACE_SummerSchool		19 minutes ago
notebooks		2 hours ago
onedata		3 months ago
quickstart		4 months ago
Training		2 days ago
workflows		6 months ago

ECASLab: Jupyter Notebooks

jupyter Time_series_extraction (read only)

Control Panel | Logout

File Edit View Insert Cell Kernel Help | Not Trusted | Python 2

Markdown

Import PyOphidia and connect to server instance

In [ ]:
```
from PyOphidia import cube, client
cube.Cube.setclient(read_env=True)
```

Import data and extract a single time series

In [ ]:
```
mycube = cube.Cube.importnc(src_path='/public/data/tos_O1_2001-2002.nc',measure='tos',imp_dim='time',ncores=5)
mycube2 = mycube.subset2(subset_dims="lat|lon",subset_filter="0:1|0:1",ncores=5)
data = mycube2.export_array()
```

Plot time series

In [ ]:
```
import matplotlib.pyplot as plt
y = data['measure'][0]['values'][0][:]
x = data['dimension'][2]['values'][:]
plt.figure(figsize=(11, 3), dpi=100)
plt.plot(x, y)

plt.ylabel(data['measure'][0]['name'] + " (degK)")
plt.xlabel("Days since 2001/01/01")
plt.title('Sea Surface Temperature (point 0.5, 1)')
plt.show()
```

Convert from Kelvin to Celsius degrees

In [ ]:
```
mycube3 = mycube2.apply(query="oph_sum_scalar('OPH_FLOAT','OPH_FLOAT',measure,-273.15)",description="celsius")
data = mycube3.export_array()
```

Plot time series

In [ ]:
```
y = data['measure'][0]['values'][0][:]
x = data['dimension'][2]['values'][:]
plt.figure(figsize=(11, 3), dpi=100)
plt.plot(x, y)

plt.ylabel(data['measure'][0]['name'] + " (degC)")
```

ECASLab Terminal (from Jupyter)

jupyter

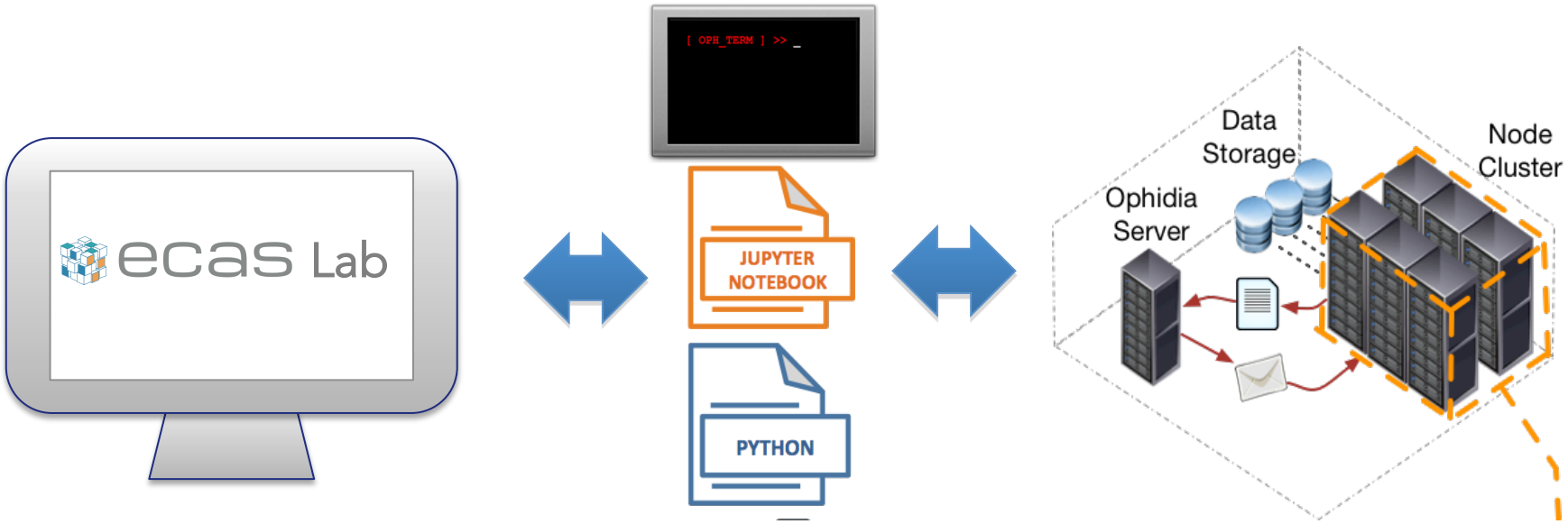
Control Panel | Logout

```
| c | tos_01_2001-2002.nc | https://ophidialab.cmcc.it/ophidia/2017/260995 | celsius |

Execution time: 0.11 seconds
[40..1913] >> oph_
oph_aggregate          oph_duplicate          oph_if                 oph_manage_session     oph_rollup
oph_aggregate2         oph_else               oph_importfits         oph_merge              oph_script
oph_apply              oph_elseif             oph_importnc           oph_mergecubes         oph_search
oph_cancel             oph_endfor             oph_importnc2          oph_mergecubes2        oph_service
oph_cluster            oph_endif              oph_importnc4          oph_metadata           oph_set
oph_concatnc           oph_explorecube        oph_importnc5          oph_movecontainer      oph_showgrid
oph_containerschema    oph_explorenc          oph_importnc6          oph_operators_list     oph_split
oph_createcontainer    oph_exportfits         oph_importsac          oph_permute            oph_subset
oph_cubeelements       oph_exportnc           oph_input              oph_primitives_list    oph_subset2
oph_cubeio             oph_exportnc2          oph_instances          oph_publish            oph_tasks
oph_cubeschema         oph_folder             oph_intercube          oph_randcube           oph_unpublish
oph_cubesize           oph_for                oph_list               oph_reduce             oph_wait
oph_delete             oph_fs                 oph_log_info           oph_reduce2
oph_deletecontainer    oph_get_config         oph_loggingbk          oph_restorecontainer
oph_drilldown          oph_hierarchy          oph_man                oph_resume
[40..1913] >> oph_cube
oph_cubeelements  oph_cubeio        oph_cubeschema    oph_cubesize
[40..1913] >> oph_cubeschema
[Request]:
operator=oph_cubeschema;sessionid=https://ophidialab.cmcc.it/ophidia/sessions/401947901415763327651512041204211913/experiment;exec_m
ode=sync;cube=https://ophidialab.cmcc.it/ophidia/2017/260995;cwd=/;cdd=/home/sfiore;host_partition=test;

[JobID]:
https://ophidialab.cmcc.it/ophidia/sessions/401947901415763327651512041204211913/experiment?241#9269

[Response]:
Datacube Information
--------------------

| PID | CREATION DATE | MEASURE | MEASURE TYPE | LEVEL | NUMBER OF FRAGMENTS | SOURCE FILE |
| https://ophidialab.cmcc.it/ophidia/2017/260995 | 2018-06-25 03:15:30 | tos | FLOAT | 2 | 1 | |

Datacube Additional Information
-------------------------------

| DESCRIPTION | HOST x CUBE | DBMS x HOST | DATABASES x DBMS | FRAGMENTS x DATABASE | ROWS x FRAGMENT | ELEMENTS x ROW | COMPRESSED | CUBE SIZE | UNIT | NUMBER OF ELEMENTS |
| celsius | 1 | 1 | 1 | 1 | 1 | 24 | no | | | 24 |

Dimension Information
---------------------

| NAME | TYPE | SIZE | HIERARCHY | CONCEPT LEVEL | ARRAY | LEVEL | LATTICE NAME |
| lat | double | 1 | oph_base | cell | no | 1 | |
| lon | double | 1 | oph_base | cell | no | 2 | |
| time | double | 24 | oph_base | cell | yes | 1 | |

Execution time: 0.12 seconds
[40..1913] >>
```

Outline

- ✔ *ECASLab*
- ✔ ***Server-side data analytics***
- ✔ *ECAS Terminal*
- ✔ *PyOphidia*
- ✔ *Live demo @ ECASLab*

Server-side data analytics



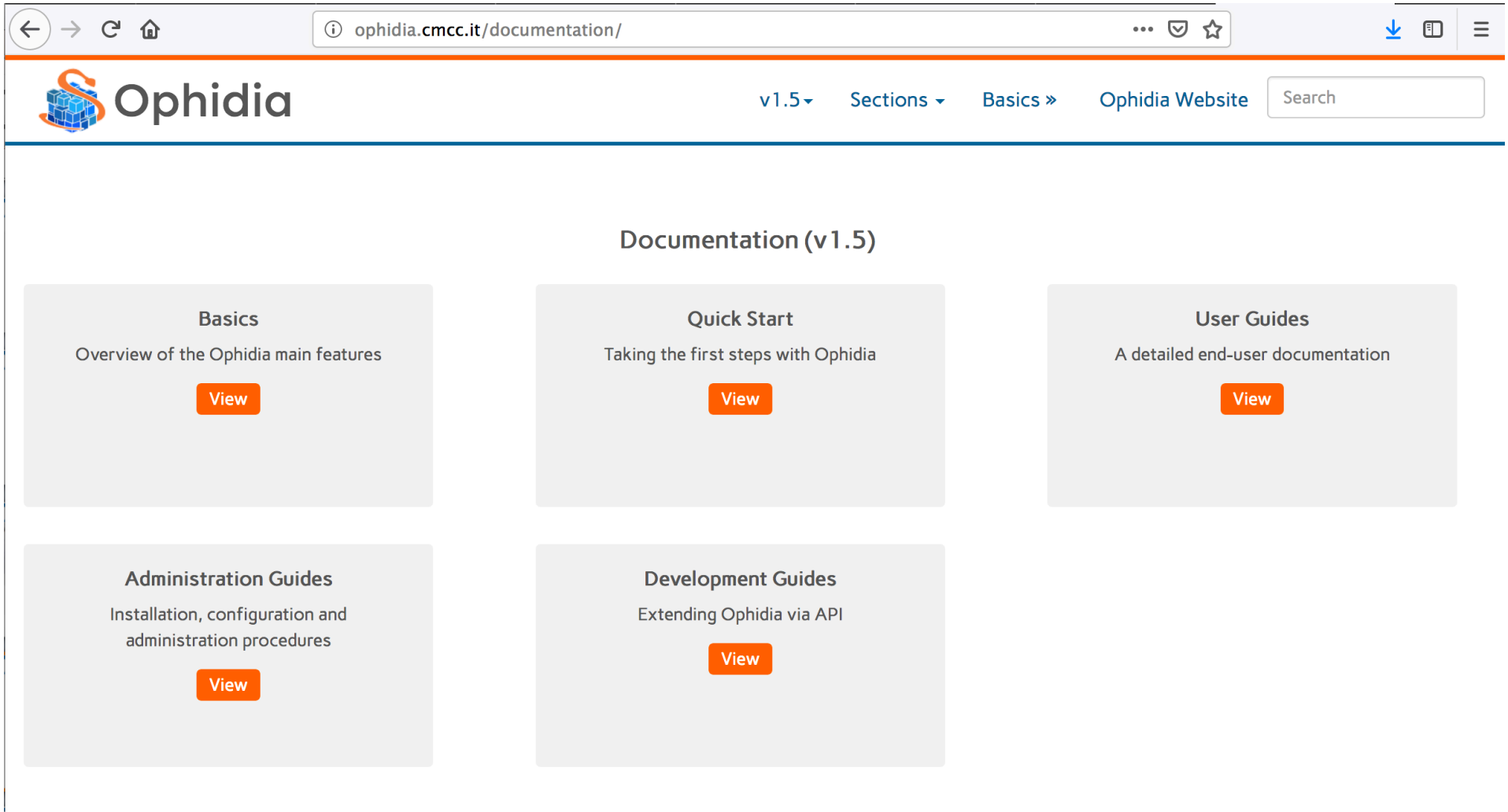
Oph_Term: a terminal-like commands interpreter serving as a client for the Ophidia framework

PyOphidia: a Python interface for datacube management & analytics with Ophidia

Through **oph_term/PyOphidia** the user run (“send”) commands (“operators”) to the Ophidia server to manipulate datasets (“datacubes”) on the cluster resources

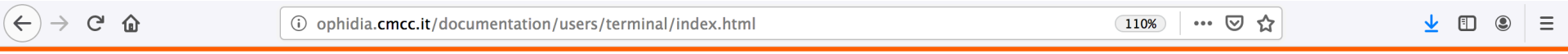
Both **oph_term** and **PyOphidia** are natively integrated into **ECASLab** and can be easily accessed from the browser

Ophidia Documentation



http://ophidia.cmcc.it/documentation

Ophidia Term documentation



Ophidia v1.5 Sections « I/O Server Cl... Ophidia Termi... » Ophidia Website Search

User Guides
- Partition Usage
- Terminal Usage
  - Basic usage
  - Advanced features
- PyOphidia
- Operators Manual
- Primitives Manual
- Data model
- Virtual File System
- Massive Operations
- Workflows Usage
- Session Management
- Time management
- Examples
- Appendix

Ophidia Terminal Manual

Ophidia Terminal is a robust, comprehensive, effective and extremely usable client, developed with characteristics similar to the bash program present in almost all Unix-like environments. The terminal manuals have been divided in two groups that describe:

- basic functionalities and concept to immediately use the CLI;
- advanced features useful for more skilled users.

Several examples of real-world usage of the terminal are also available on the website tutorial section.

- Basic usage
  - Starting the terminal
  - Submitting a request
  - Some examples
- Advanced features
  - Client-server interaction
  - Oph-Terminal environment
  - History management
  - Embedded help and auto-completion
  - Rendering output
  - Cube Provenance
  - Variable substitution
  - Aliases

http://ophidia.cmcc.it/documentation/users/terminal/index.html

PyOphidia repository

PyOphidia	Update interfaces in cube.py	5 months ago
conda/recipe	Adding Conda Recipe (#5)	2 years ago
.gitignore	Add .gitignore	4 years ago
AUTHORS.rst	Update author information	2 years ago
CONTRIBUTING.rst	Initial commit	4 years ago
HISTORY.rst	Update history for release	5 months ago
LICENSE	Initial commit	4 years ago
MANIFEST.in	Initial commit	4 years ago
README.rst	Update readme for release	5 months ago
setup.cfg	Initial commit	4 years ago
setup.py	Update history for release	5 months ago

README.rst

PyOphidia: Python bindings for Ophidia

PyOphidia is a GPLv3-licensed Python package for interacting with the Ophidia framework.

It is an alternative to Oph_Term, the Ophidia no-GUI interpreter component, and a convenient way to submit SOAP HTTPS requests to an Ophidia server or to develop your own application using Python.

https://github.com/OphidiaBigData/PyOphidia

Outline

- ✔ *ECASLab*
- ✔ *Server-side data analytics*
- ✔ ***ECAS Terminal***
- ✔ *PyOphidia*
- ✔ *Live demo @ ECASLab*

ECASLab Terminal

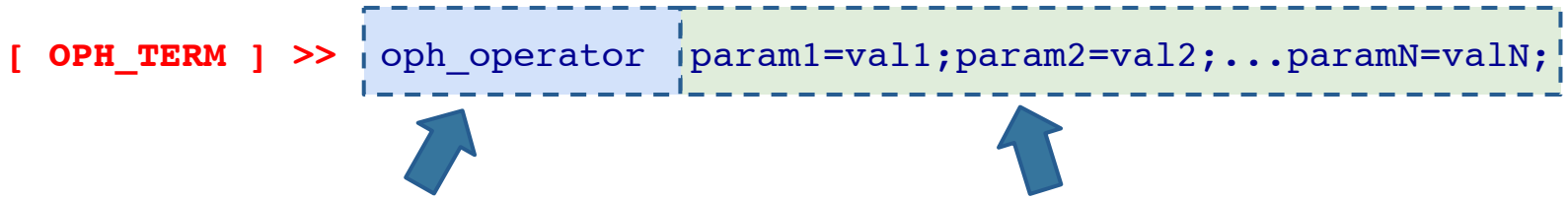
The **ECASLab terminal** is a shell interface integrated in ECASLab JupyterHub interface

Based on the **Ophidia Terminal**, a CLI bash-like client for the Ophidia framework.

It can be used for:

- ✓ Executing **interactive** data analytics sessions;
- ✓ Executing **batch** data analytics tasks of **workflows;**
- ✓ Experiment and operators **debugging;**
- ✓ FS **exploration** and environment **management**.

ECASLab Terminal: operator submission



Special arguments:

- "exec_mode": specifies if the command is executed in synchronous (”sync") or asynchronous mode (”async") which is the default;
- "ncores": it specifies the number of parallel processes requested for the execution of the operator (default is 1);
- "nthreads": it specifies the number of parallel threads per each processes requested for the execution of the operator (default is 1);
- "cube": it specifies the input datacube. It is automatically added by the terminal exploiting the last produced cube.

For information concerning the parameters of each operator, the manual can be exploited.

ECASLab Terminal: main features

Example of submission of *oph_list* command

```
[11..4495] >> oph_list level=2;
[Request]:
operator=oph_list;path=;level=2;sessionid=https://ophidialab.cmcc.it/ophidia/sessions/1112
38695229505952271558621818154495/experiment;exec_mode=sync;cube=https://ophidialab.cmcc.it
/ophidia/2924/374978;cwd=/;cdd=/home/delia;host_partition=auto;

[JobID]:
https://ophidialab.cmcc.it/ophidia/sessions/11123869522950595227155862181815449 5/experimen
t?224089#455500

[Response]:
Ophidia Filesystem: /
---------------------

+===+====================+================================================+=============+
| T | PATH               | DATACUBE PID                                   | DESCRIPTION |
+===+====================+================================================+=============+
| f | testFolder/        |                                                |             |
|---|--------------------|------------------------------------------------|-------------|
| c | test               | https://ophidialab.cmcc.it/ophidia/2917/374976 |             |
|---|--------------------|------------------------------------------------|-------------|
| c | tos.nc             | https://ophidialab.cmcc.it/ophidia/2924/374978 |             |
+===+====================+================================================+=============+

Execution time: 0.12 seconds
```

ECASLab Terminal: main features

Besides command submission, the **Ophidia Terminal** supports:

- ✓ Search in command history: `[crtl + r] start typing…`

```
(reverse-i-search)`oph_l': oph_list level=2;
```

- ✓ Integrated help: `help command`

```
[OPH_TERM] >> help help
USAGE OF help COMMAND

        help [cmd|var]

        Without arguments, list all available commands and the
        environment variables.
        Optionally append a command/variable name to get its
        usage.
```

ECASLab Terminal: main features

Besides command submission, the **Ophidia Terminal** supports:

- ✓ Commands and (pre-defined) alias or environment variables auto-completion: `start typing and press [tab] twice`

```
[OPH_TERM] >> oph_l
oph_list        oph_log_info    oph_loggingbk
```

- ✓ Inline argument/parameter suggestion: `after typing the command press [tab] twice`

```
[OPH_TERM] >> oph_list
** cwd                                  measure_filter (all)
container_filter (all)                  ncores (1)
cube (all)                              ntransform (all)
exec_mode [async|sync (async)]          recursive [yes|no (no)]
level [0|1|2|3|4|5|6|7|8 (1)]          src_filter (all)
...
```

ECASLab Terminal: main features

Besides command submission, the **Ophidia Terminal** supports:

- ✓ Complete operators/primitives manual: `oph_man function=oph_operator`

```
[OPH_TERM] >> oph_man function=oph_list
...

[Response]:
Function Name and Version
-------------------------
OPH_LIST v1.0

Function Info
-------------
+===========+=============================================================================+
| INFO TYPE | INFO VALUE                                                                  |
+===========+=============================================================================+
| Abstract  | [Type]                                                                      |
|           | Metadata Access.                                                            |
|           |                                                                             |
|           | [Behaviour]                                                                 |
|           | It shows information about folders, container and datacubes fragmentation   |
|           | (file system).                                                              |
|           | Levels from 0 to 2 show information about the folder tree. If the cwd is empty |
|           | the current path is shown.                                                  |
|           | Level 3 is similar to level 2 but it also shows additional information about |
|           | the data cubes.                                                             |
...
```

ECASLab Terminal: main features

Besides command submission, the **Ophidia Terminal** supports:

- ✓ Custom aliases and environment variables: `setalias/setenv`

```
[OPH_TERM] >> setalias lsla="oph_list level=3;cwd=$1;"
[OPH_TERM] >> getalias lsla
oph_list level=3; cwd=$1;
[OPH_TERM] >> setenv TEST_PATH="/testFolder"
[OPH_TERM] >> getenv TEST_PATH
/testFolder
[OPH_TERM] >> lsla $TEST_PATH
[Request]:
operator=oph_list;level=3;cwd=/testFolder;...
OPH_TERM] >> unsetalias lsla
```

- ✓ Task submission history: `resume nwf / view #wfid`

```
[OPH_TERM] >> resume 2
[224117] 2019-09-13 17:29:46 [OPH_STATUS_ERROR]     lsla $TEST_PATH
[224119] 2019-09-13 17:29:59 [OPH_STATUS_COMPLETED] lsla $TEST_PATH
[OPH_TERM] >> view 224119
[224119] lsla $TEST_PATH [https://ophidialab.cmcc.it/ophidia/sessions/
11123869522950595227155862181815449 5/experiment?224119#455560]
...
```

Outline

- ✔ *ECASLab*
- ✔ *Server-side data analytics*
- ✔ *ECAS Terminal*
- ✔ ***PyOphidia***
- ✔ *Live demo @ ECASLab*

Python programmatic access to Ophidia

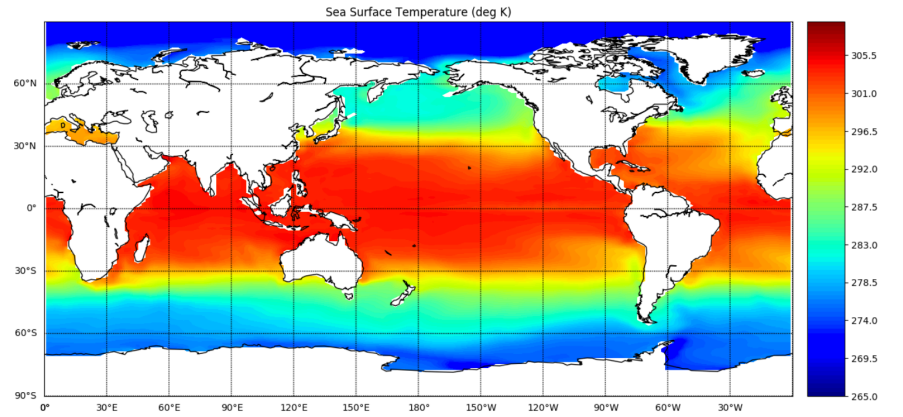
PyOphidia is a GPLv3-licensed Python module to interact with the Ophidia framework, integrated within the ECASLab environment.

It provides a more programmatic access to Ophidia features, allowing:

- ✓ Submission of commands to the Ophidia Server and retrieval of the results
- ✓ Management of (remote) data objects in the form of datacubes
- ✓ Easy exploitation from Jupyter Notebooks and integration with other Python modules

```
from PyOphidia import cube, client
cube.Cube.setclient(read_env=True)

mycube =
cube.Cube.importnc(src_path='/public/data/ecas_training
/file.nc', measure='tos', imp_dim='time',
import_metadata='yes', ncores=5)
mycube2 = mycube.reduce(operation='max',ncores=5)
mycube3 = mycube2.rollup(ncores=5)
data = mycube3.export_array()

mycube3.exportnc2(output_path='/home/test',
export_metadata='yes')
```

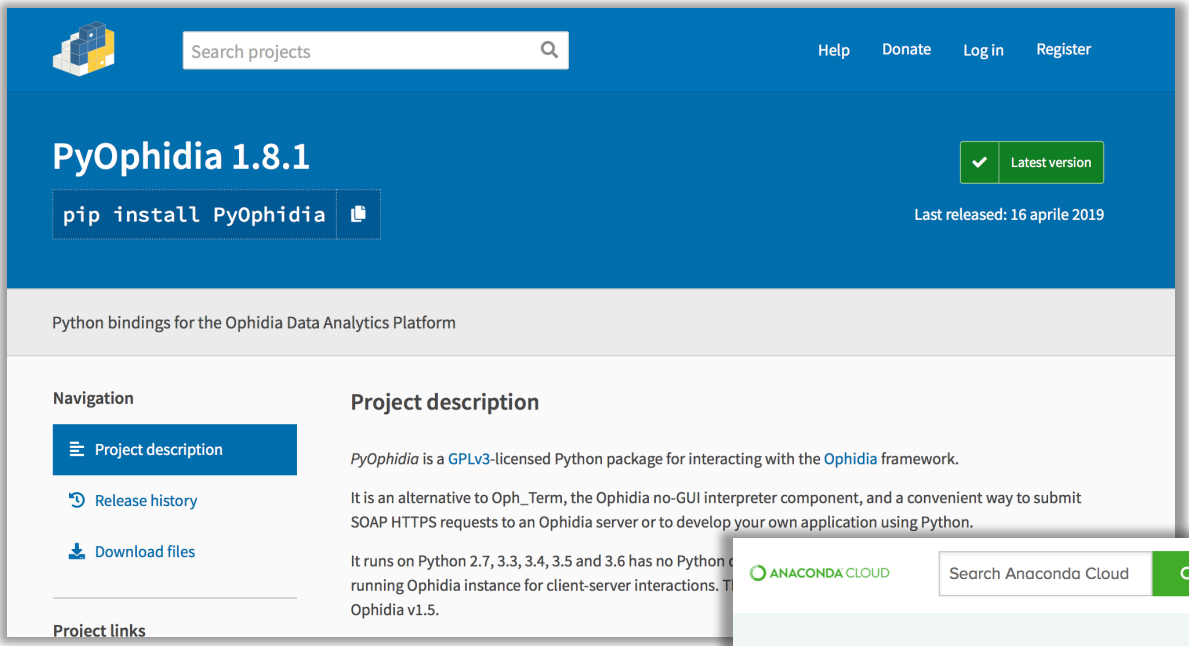
```
plt.title('Sea Surface Temperature (deg K)')
plt.show()
```



Export result to NetCDF file

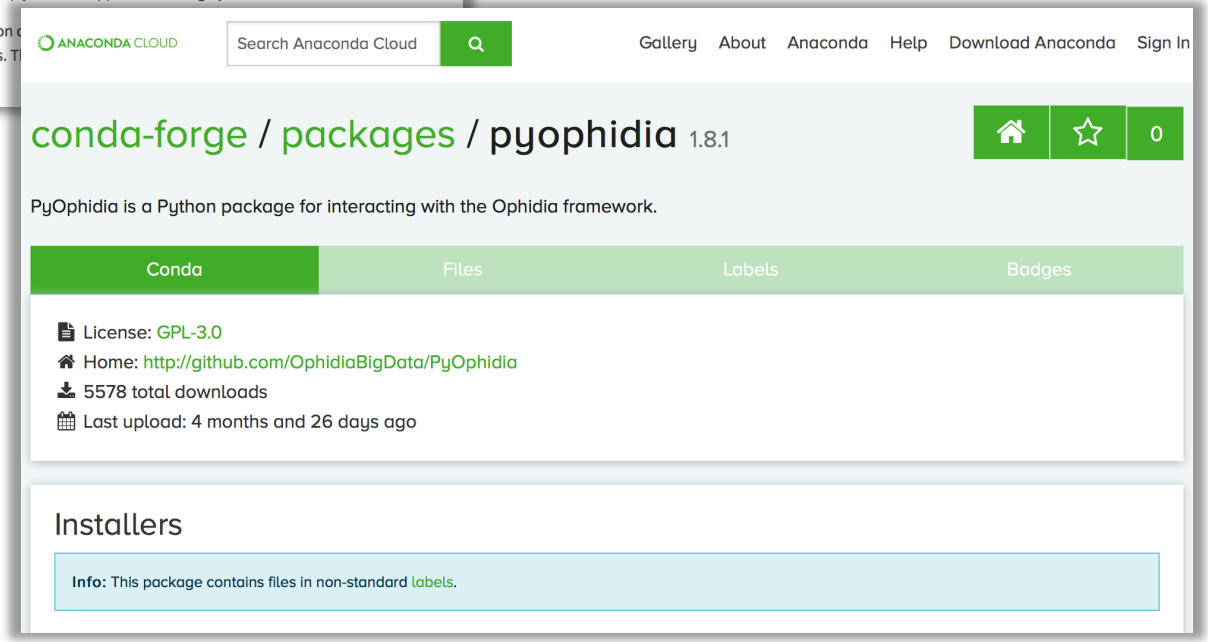
```
]: mycube3.exportnc2(output_path='/home/' + cube.Cube.client.username,export_metadata='yes')
```

PyOphidia release



https://pypi.org/project/PyOphidia/

```
pip install pyophidia
```



https://anaconda.org/conda-forge/pyophidia

```
conda install -c conda-forge pyophidia
```

The PyOphidia library

PyOphidia implements two main classes:

- ✓ **Client class**: supports the submissions of Ophidia commands and workflows, as well as the management of session from Python code (similar to the Ophidia Terminal)
  - ✓ It allows to run all the Ophidia operators, including massive tasks and workflows
- ✓ **Cube class**: provides the datacube type abstraction and the methods to manipulate, process and get information on cubes objects
  - ✓ Defines a object-oriented approach allowing a handle the datacubes more naturally

The two modules provide different features and can/should be used jointly.

The PyOphidia library: client class

Example of **client class** usage:

- ✓ Load the module and start a new client

```
from PyOphidia import client

ophclient = client.Client(username="oph-user", password="oph-passwd",
                          server="127.0.0.1", port="11732")
```

- ✓ In ECASLab, the arguments can be automatically inferred by the environment

```
ophclient = client.Client(read_env=True)
```

- ✓ Commands follow the same structure as for the Oph_term (`oph_operator param1=val1;`)

```
ophclient.submit("oph_list level=1")
```

The PyOphidia library: client class

Example of **client class** usage:

- ✓ The `display` argument allows to show (with `True`) or suppress (with `False`, default) the command resulting output. It is recommended to be used mainly with metadata commands (such as oph_list, oph_metadata, oph_cubeschema).

```
ophclient.submit("oph_list level=2", display=True)
```

```
Ophidia Filesystem: /
------------------------------
+===+========+=================================================+=============+
| T | PATH   | DATACUBE PID                                    | DESCRIPTION |
+===+========+=================================================+=============+
| c | test   | https://ophidialab.cmcc.it/ophidia/2917/374976  |             |
|---|--------|-------------------------------------------------|-------------|
| c | tos.nc | https://ophidialab.cmcc.it/ophidia/2924/374978  |             |
+===+========+=================================================+=============+
Execution time: 0.12 seconds
```

The PyOphidia library: client class

Example of **client class** usage:

- ✓ Similarly to the Ophidia Terminal, special parameters can be set in the environment and used for all the commands submited (if not specified). The following command will be exectued with 2 cores

```
ophclient.ncores = 2

ophclient.submit("oph_reduce2 operation=avg;dim=time;\
cube=https://ophidialab.cmcc.it/ophidia/3274/375586;")
```

- ✓ The last cube produced can be accessed from the `cube` parameter. Again, if not specified, it will automatically added to the next command submitted

```
print(ophclient.cube)
```

```
https://ophidialab.cmcc.it/ophidia/3274/375592
```

The PyOphidia library: cube class

PyOphidia Cube class introduces the concept of **cube objects** and supports all the Ophidia operators as **methods.**

To this end, the class defines two types of methods according to the type of operator:

- ✓ **Class methods**: concerning the operators which do not refer to a particular cube object (e.g. the oph_list, the operators to manage the file system, etc.)

```
cube.Cube.list(level=2)
```

- ✓ **Instance methods**: concern the operators applied directly on a cube object to access and manipulate it (by creating a new cube object)

```
mycube.info()

mycube2 = mycube.reduce(operation='max',ncores=5)
```

The PyOphidia library: cube class

Example of **cube class** usage:

- ✓ Load the module and setup a connection to the server instance (similar to client class)

```
from PyOphidia import cube

cube.Cube.setclient(username="oph-user", password="oph-passwd",
                    server="127.0.0.1", port="11732")
```

- ✓ In ECASLab, the arguments can be automatically inferred by the environment

```
cube.Cube.setclient(read_env=True)
```

- ✓ Internally the cube class exploits the client module, so commands can still be specified as in the client class. However…

```
cube.Cube.client.submit("oph_list level=2", display=True)
```

The PyOphidia library: cube class

Example of **cube class** usage:

- ✓ All commands are mapped to specific class / instance methods and arguments for each operator parameter can be specified (default values will be used instead):

```
cube.Cube.list(level=2)
```

- ✓ To get the list of arguments and default values the python `help` command can be used

```
help(cube.Cube.list)
```

```
Help on method list in module PyOphidia.cube:

list(level=1, exec_mode='sync', path='-', cwd=None, container_filter='all',
cube='all', host_filter='all', dbms_filter='all', measure_filter='all',
ntransform='all', src_filter='all', db_filter='all', recursive='no',
objkey_filter='all', display=True) method of builtins.type instance

...
```

The PyOphidia library: cube class

Example of **cube class** usage:

- ✓ A cube object can be created in multiple ways. In case of pre-existing cube (pid):

```
mycube = cube.Cube(pid='https://ophidialab.cmcc.it/ophidia/1/1')
```

- ✓ A cube can be also created from a NetCDF file using the constructor function:

```
mycube = cube.Cube(exp_dim='lat|lon',imp_dim='time',ncores=2
          measure='tos',src_path='/path/tos.nc')
```

- ✓ or directly using the import method (exactly the same as the previous one):

```
mycube = cube.Cube.importnc(exp_dim='lat|lon',imp_dim='time',ncores=2
                            measure='tos',src_path='/path/tos.nc')
```

- ✓ After the processing, the cube can be deleted with the proper method:

```
mycube.delete()
```

The PyOphidia library: cube class

Example of **cube class** usage:

- ✓ The datacube structure (and its partitioning schema) can be inspected with:

```
mycube.info()
```

```
...
+======+========+======+===========+==============+=======+=======+
| NAME | TYPE   | SIZE | HIERARCHY | CONCEPT LEVEL | ARRAY | LEVEL |
+======+========+======+===========+==============+=======+=======+
| lat  | double | 170  | oph_base  | cell          | no    | 1     |
|------|--------|------|-----------|---------------|-------|-------|
| time | long   | ALL  | oph_base  | ALL           | yes   | 0     |
+======+========+======+===========+==============+=======+=======+
```

- ✓ Whereas the values can be accessed with:

```
mycube.explore(limit_filter=1, show_time='yes')
```

```
+=======+=====+=============================================================+
| lat   | lon | tos                                                         |
+=======+=====+=============================================================+
| -79.5 | 1.0 | 275.665710, 275.859649, 276.042022, 276.142547, 276.166168  |
+=======+=====+=============================================================+
...
```

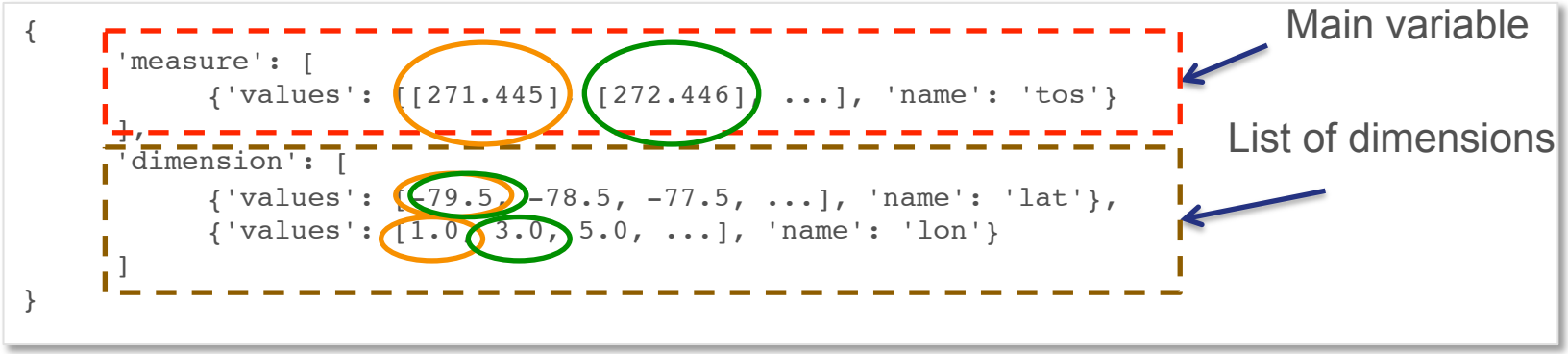
The PyOphidia library: cube class

Example of **cube class** usage:

- ✓ Moreover the datacube values can be exported into a Python data structure :

```
data = mycube.export_array()
```

- ✓ The structure created is a Python dictionary with nested dictionaries



- ✓ Data can then be accessed providing the proper dictionary keys

```
lat = data['dimension'][0]['values'][:]
lon = data['dimension'][1]['values'][:]
var = data['measure'][0]['values'][:]
```

The PyOphidia library: cube class

Example of **cube class** usage:

- ✓ Once a cube is available in the python code, various operators can be executed to produce new datacubes:

```
mycube2 = mycube.reduce(operation='max',ncores=5)

mycube3 = mycube2.subset2(subset_dims="lat|lon|time", ncores=5,
                          subset_filter="-80:30|30:120|151:240")

mycube4 = mycube3.aggregate(operation='max',ncores=5)
```

- ✓ Methods can also be concatenated into a single command:

```
mycube5 = mycube.reduce(operation='max',ncores=5).subset2(
                  subset_dims="lat|lon|time", ncores=5,
                  subset_filter="-80:30|30:120|151:240").aggregate(
                                operation='max',ncores=5)
```

Useful Resources

- *CMCC ECASLab instance: https://ecaslab.cmcc.it/*
- *ECASLab JupyterHub @ CMCC: https://ecaslab.cmcc.it/jupyter/*
- *ECAS repository: https://github.com/ECAS-Lab*
- *Ophidia Website: http://ophidia.cmcc.it*
- *Ophidia Doc : http://ophidia.cmcc.it/documentation*
- *PyOphidia repository: https://github.com/OphidiaBigData/PyOphidia*

Outline

- ✔ *ECASLab*
- ✔ *Server-side data analytics*
- ✔ *ECAS Terminal*
- ✔ *PyOphidia*
- ✔ ***Live demo @ ECASLab: https://ecaslab.cmcc.it/jupyter/***

Demo @ ECASLab

Control Panel | Logout

Files | Running | Clusters

Select items to perform actions on them.

Upload | New | ↻

	Name ↑	Last Modified ↑
☐	data	6 months ago
☐	Demo_EUDAT_PRACE_SummerSchool	a minute ago
☐	Hands-on-EUDAT_PRACE_SummerSchool	19 minutes ago
☐	notebooks	2 hours ago
☐	onedata	3 months ago
☐	quickstart	4 months ago
☐	Training	2 days ago
☐	workflows	6 months ago

Thanks

Donatello Elia | donatello.elia@cmcc.it