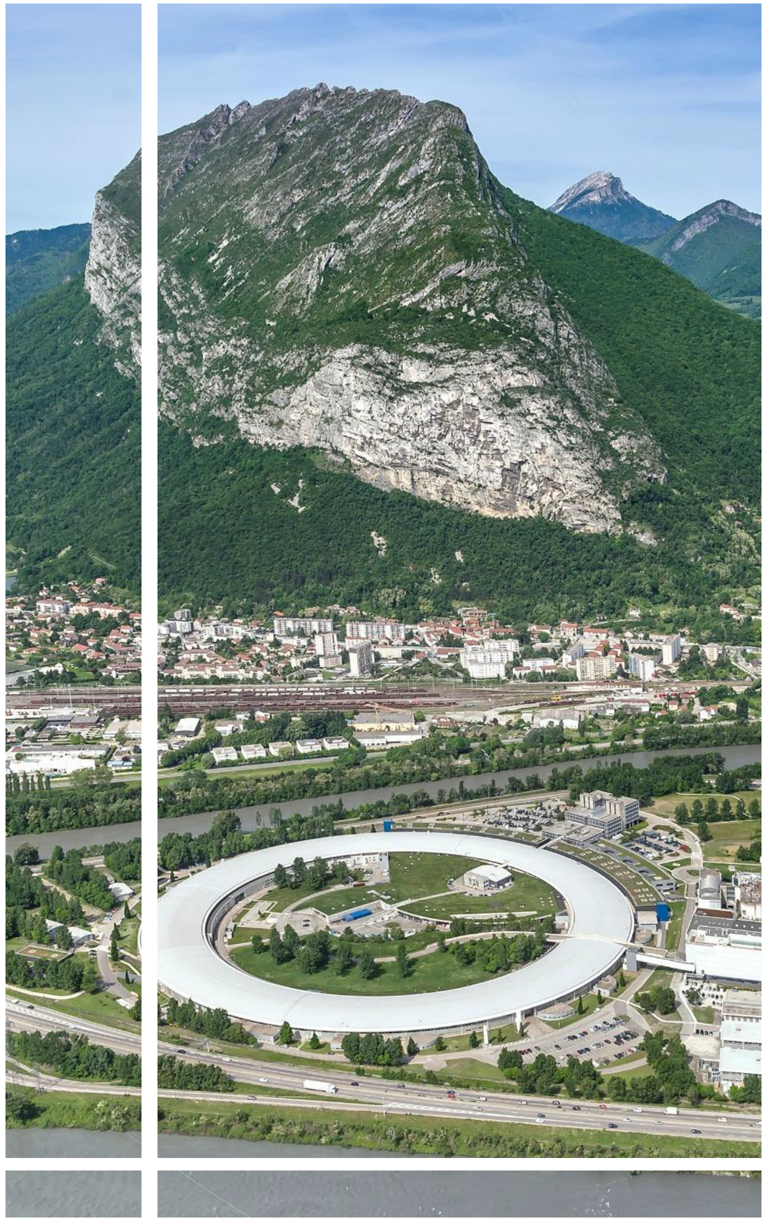
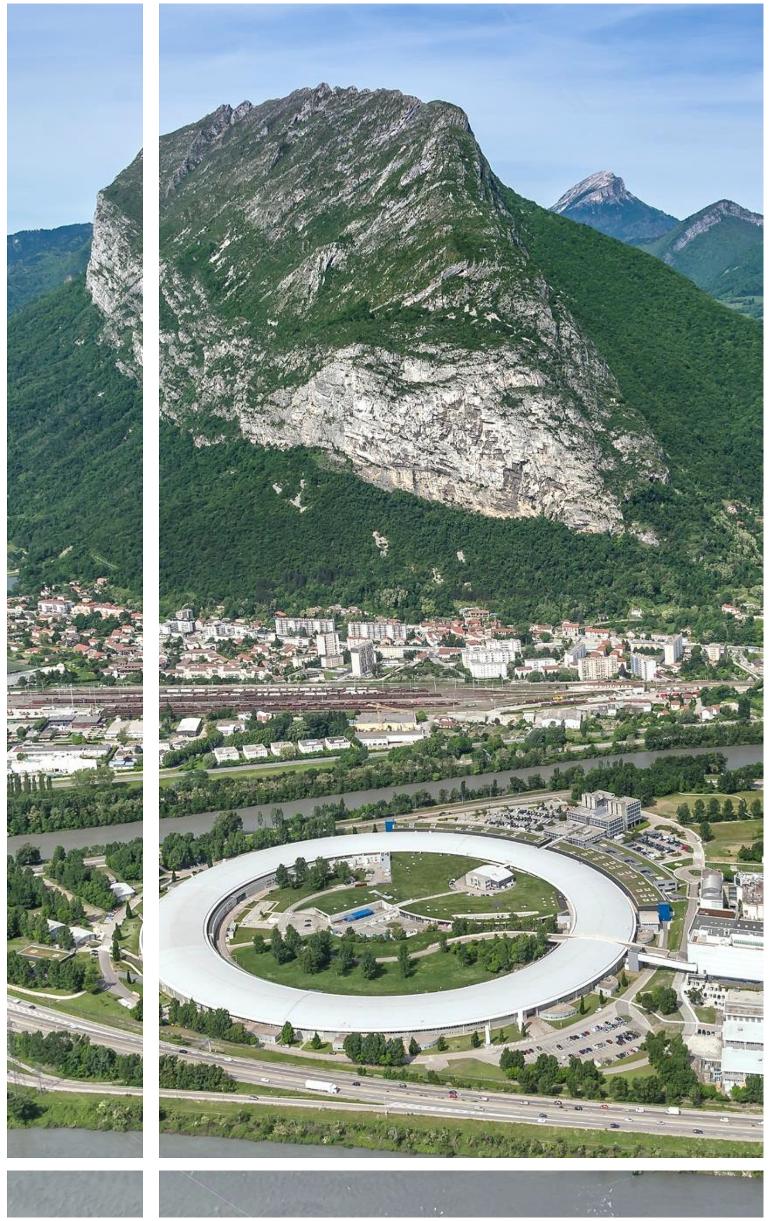
ESRF | The European Synchrotron



BLISS and EWOKS as collaborative platform
EWOKS (Extensible Workflow System)

Wout De Nolf
ESRF (Data Automation Unit)

STREAMLINE has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 870313

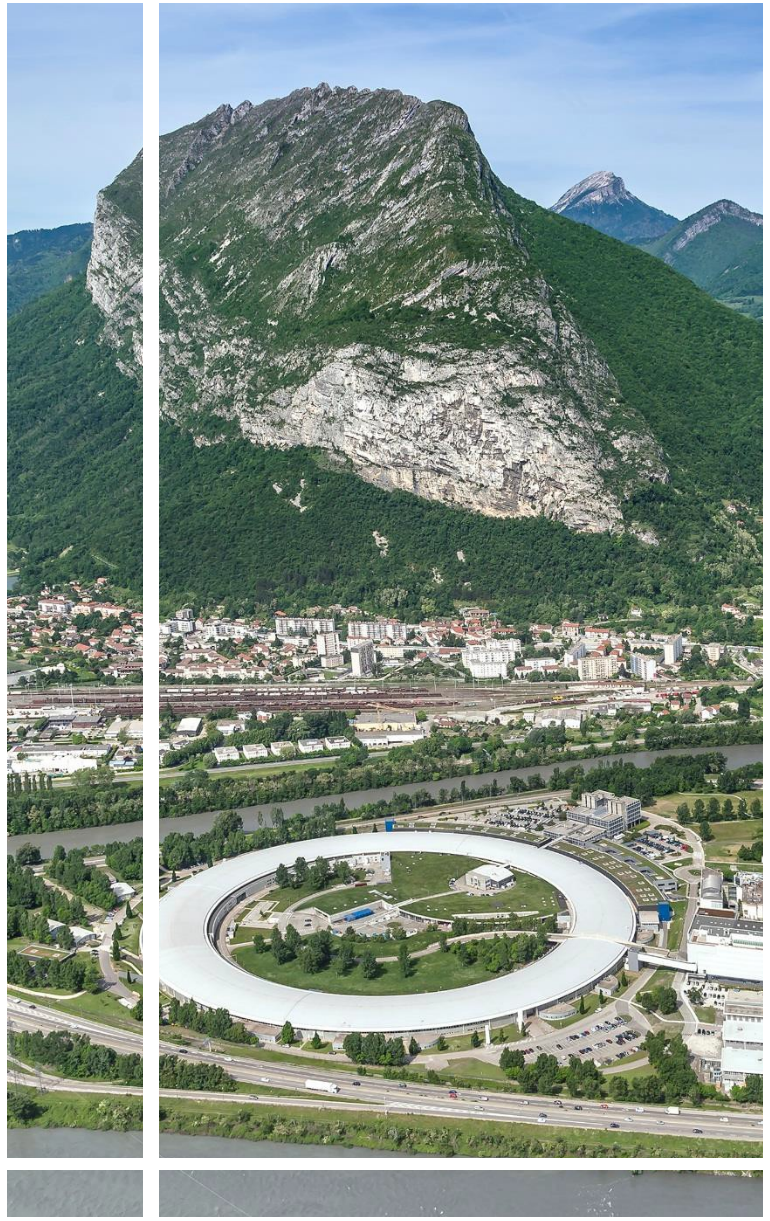


Goal of this demo

Getting started with EWOKS in general.

Use blissdata and EWOKS together for online data analysis.

WHAT IS EWOKS?



EWOKS: a workflow-based solution to

1. **automate** data processing and beamline operation
2. and make data processing results **FAIR** (the traceable and reproducible aspect).

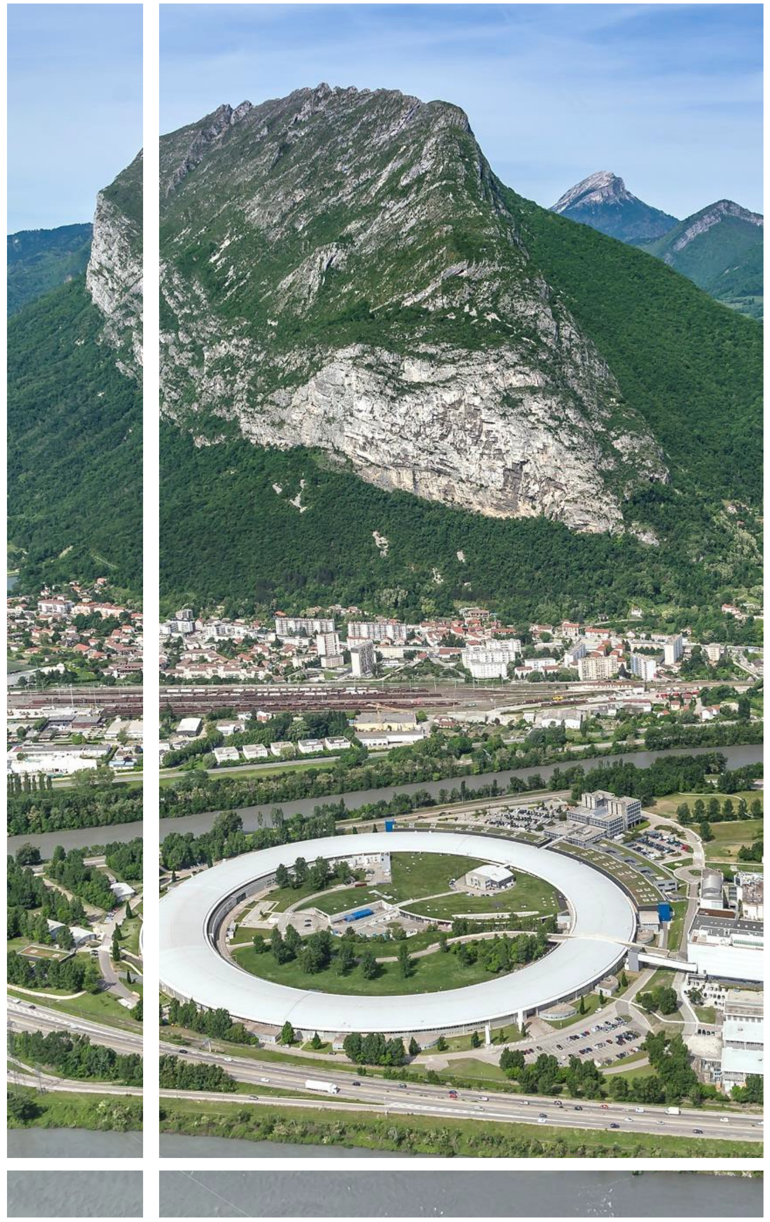
Meta workflow system: decouple workflows and their representation from the workflow management system that executes, visualizes and manages them.

Thursday 26 September

ESRF Auditorium**: workflow engines**

16:40 → 16:55: EWOKS presentation (Loic Huder - ESRF)

EWOKS INTERFACES



Getting started: https://ewoks.esrf.fr

Install (python package)

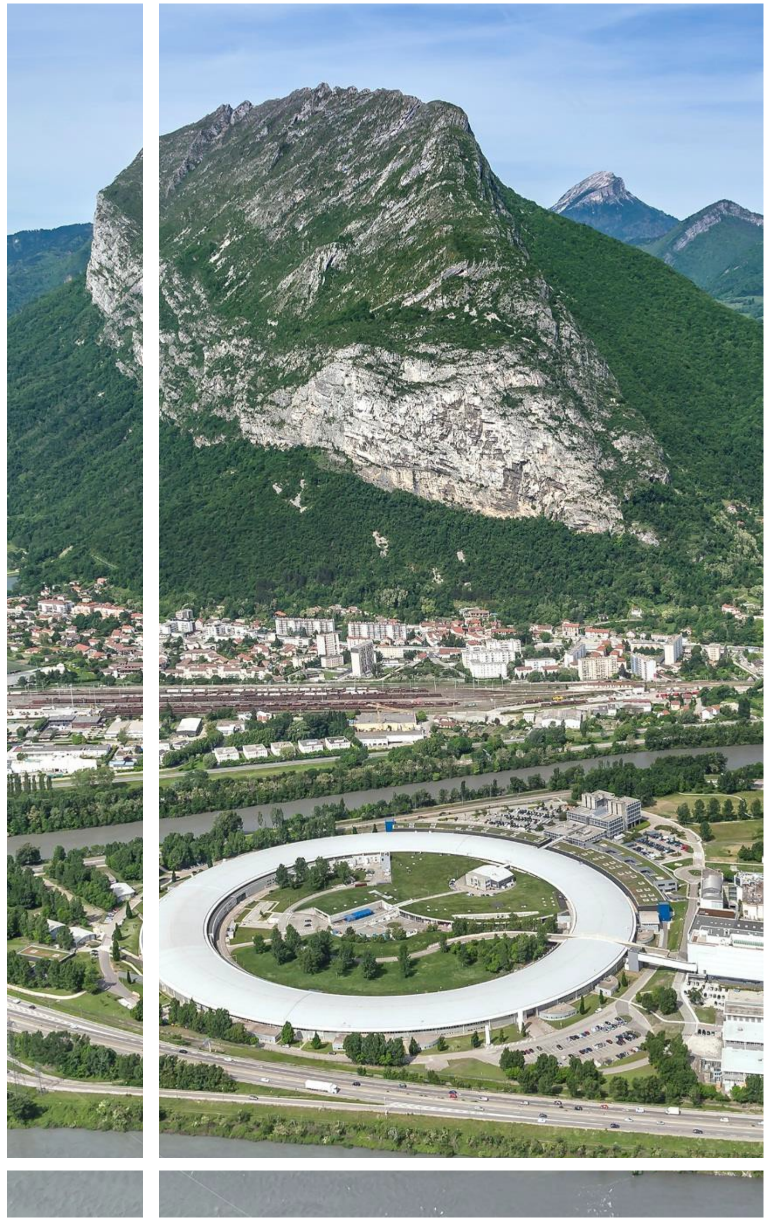
```
pip install ewoks
```

Execute a demo workflow with parameters and print the result of each workflow node

```
ewoks execute demo -p a=10 -p b=3 --test --outputs=all
```

This is the **command line interface** of EWOKS. Different interfaces exist for different purposes.

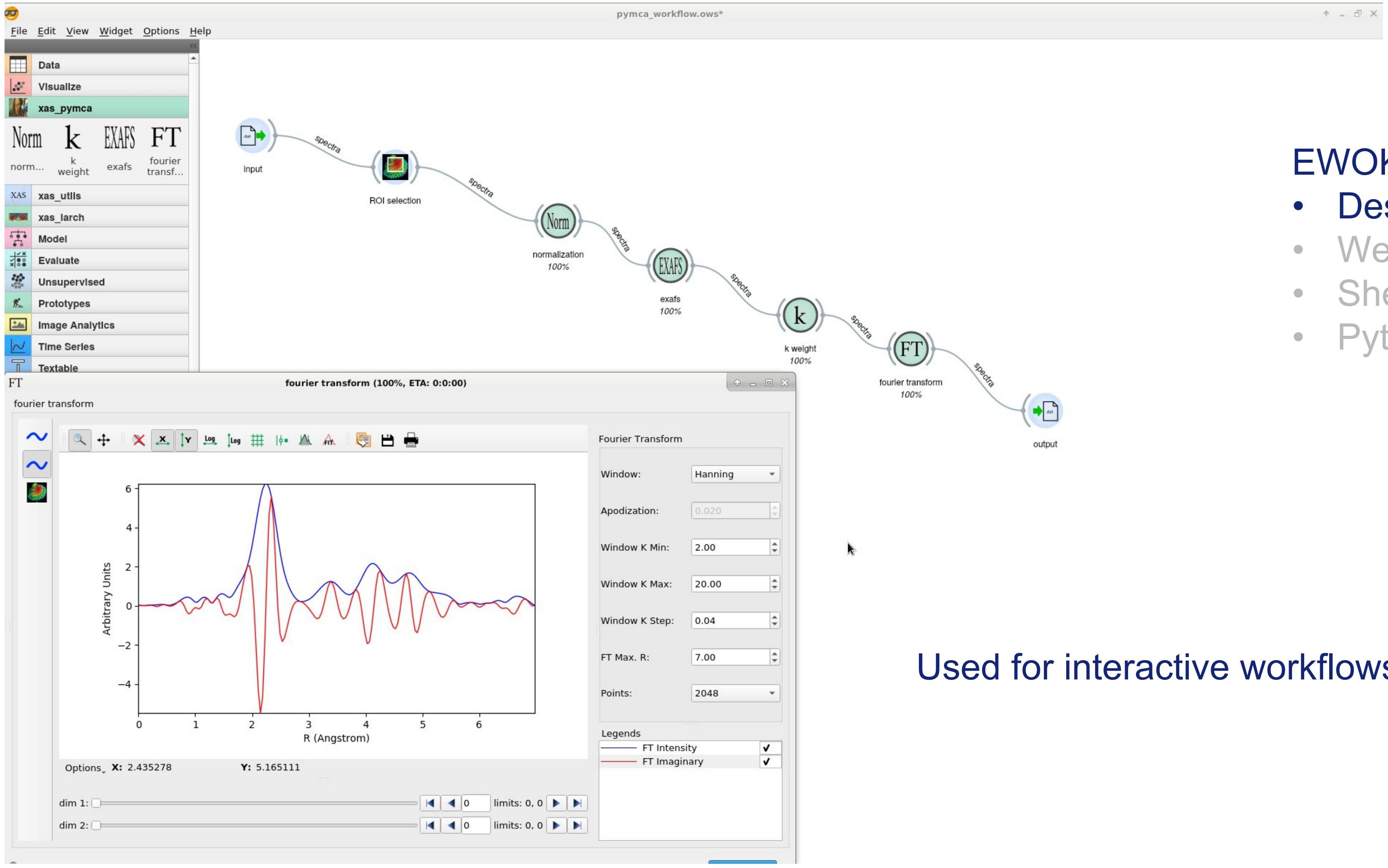
EWOKS INTERFACES



EWOKS interfaces and why they exist

- Desktop (interactive workflows)
- Web (workflows as a service)
- Shell (headless execution)
- Python (integration for developers)

EWOKS INTERFACES



Options X: 2.435278 Y: 5.165111

dim 1: 0 limits: 0, 0

dim 2: 0 limits: 0, 0

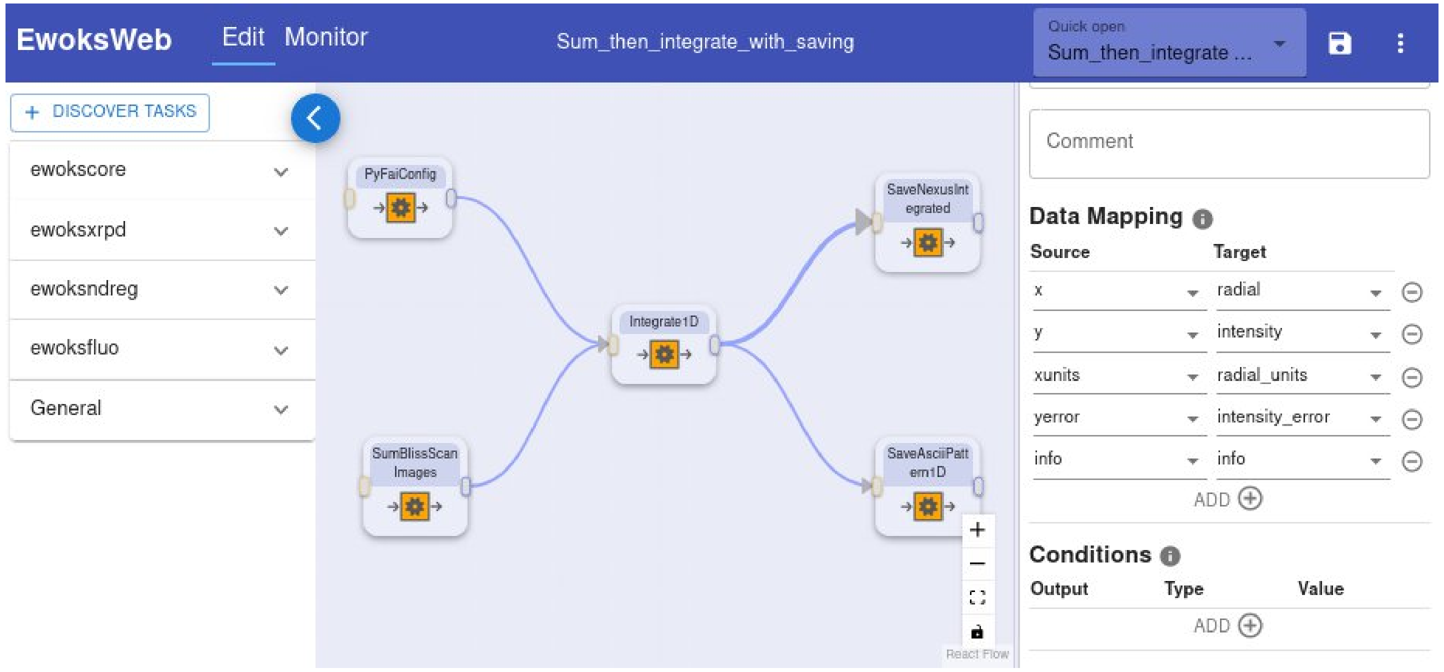
Window:	Hanning
Apodization:	0.020
Window K Min:	2.00
Window K Max:	20.00
Window K Step:	0.04
FT Max. R:	7.00
Points:	2048

EWOKS interfaces
- Desktop
- Web
- Shell
- Python

Used for interactive workflows

The European Synchrotron | ESRF

EWOKS INTERFACES



Data Mapping

Source	Target
x	radial
y	intensity
xunits	radial_units
yerror	intensity_error
info	info

ADD

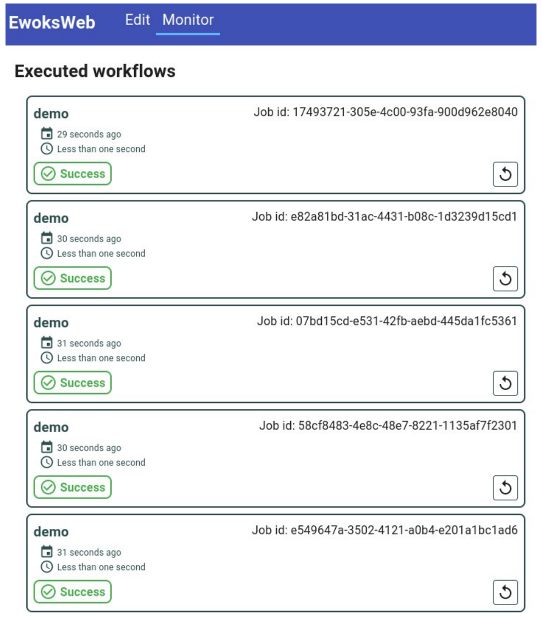
Conditions

Output	Type	Value

ADD

EWOKS interfaces

- Desktop
- Web
- Shell
- Python


EwoksWeb Edit Monitor

Executed workflows

demo — Job id: 17493721-305e-4c00-93fa-900d962e8040
29 seconds ago
Less than one second
Success

demo — Job id: e82a81bd-31ac-4431-b08c-1d3239d15cd1
30 seconds ago
Less than one second
Success

demo — Job id: 07bd15cd-e531-42fb-aebd-445da1fc5361
31 seconds ago
Less than one second
Success

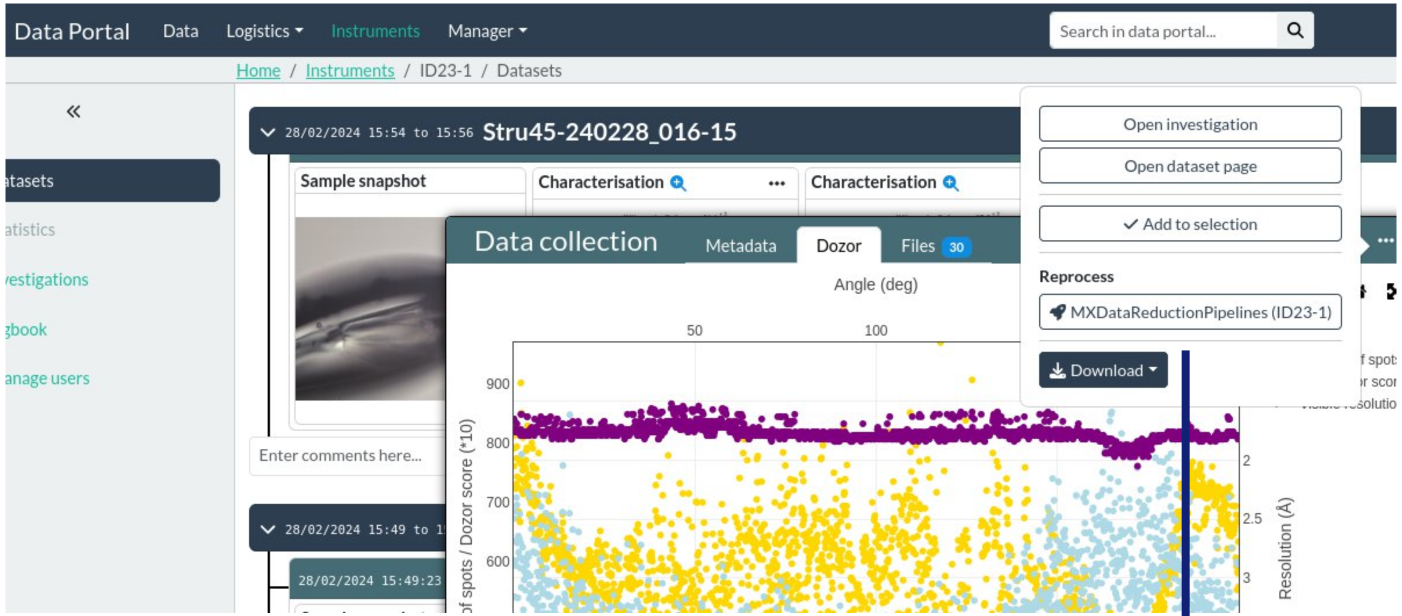
demo — Job id: 58cf8483-4e8c-48e7-8221-1135af7f2301
30 seconds ago
Less than one second
Success

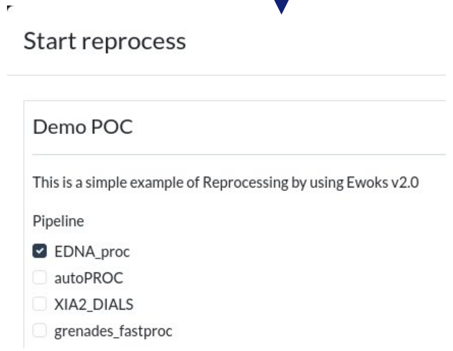
demo — Job id: e549647a-3502-4121-a0b4-e201a1bc1ad6
31 seconds ago
Less than one second
Success

Used to visualize workflows that don’t have graphical components

Standalone + frontend (similar to Jupyter notebooks)

EWOKS INTERFACES




Start reprocess

Demo POC

This is a simple example of Reprocessing by using Ewoks v2.0

Pipeline

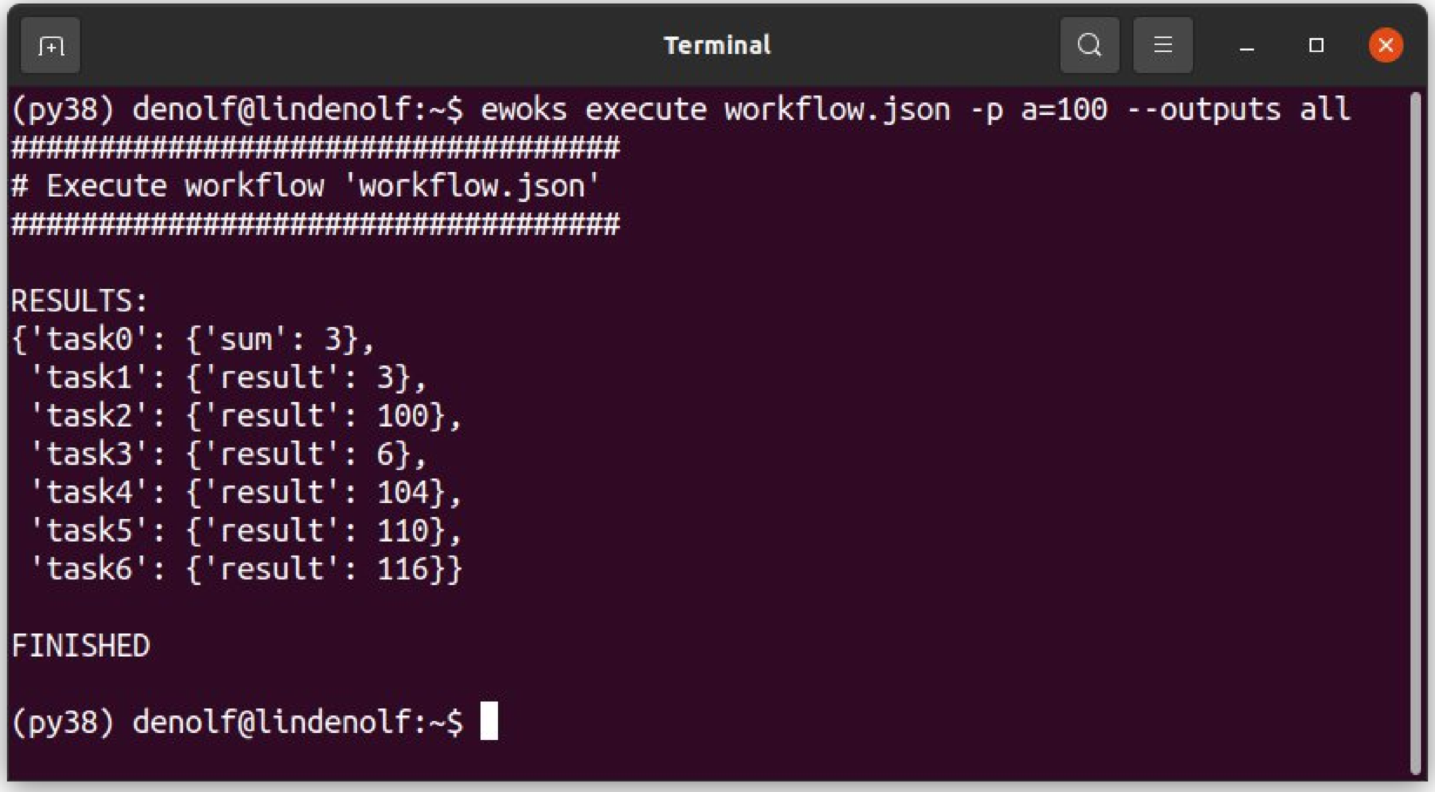
- [✓] EDNA_proc
- [ ] autoPROC
- [ ] XIA2_DIALS
- [ ] grenades_fastproc

EWOKS interfaces

- Desktop
- Web
- Shell
- Python

Web service used by other web services (e.g. ESRF data portal)

EWOKS INTERFACES

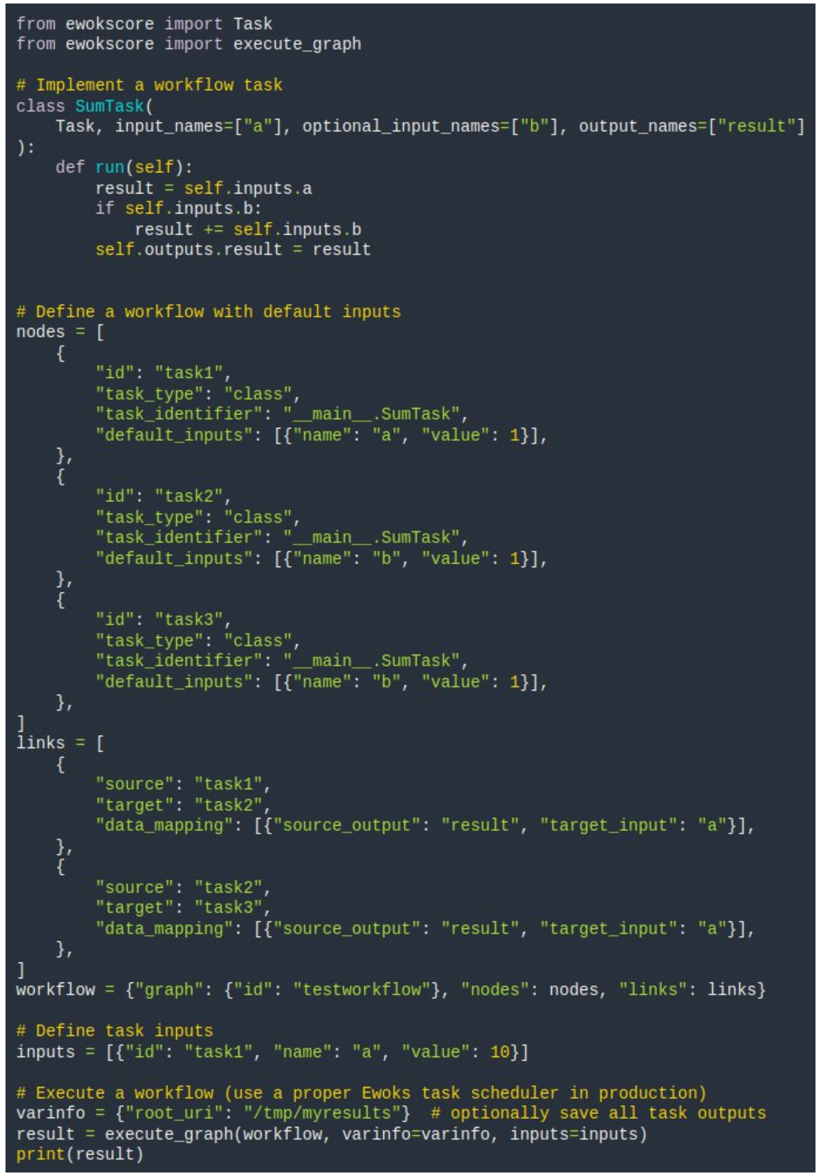


```
(py38) denolf@lindenolf:~$ ewoks execute workflow.json -p a=100 --outputs all
###################################
# Execute workflow 'workflow.json'
###################################

RESULTS:
{'task0': {'sum': 3},
 'task1': {'result': 3},
 'task2': {'result': 100},
 'task3': {'result': 6},
 'task4': {'result': 104},
 'task5': {'result': 110},
 'task6': {'result': 116}}

FINISHED

(py38) denolf@lindenolf:~$ 
```

EWOKS interfaces
- Desktop
- Web
- **Shell**
- Python

Used for headless execution

EWOKS INTERFACES



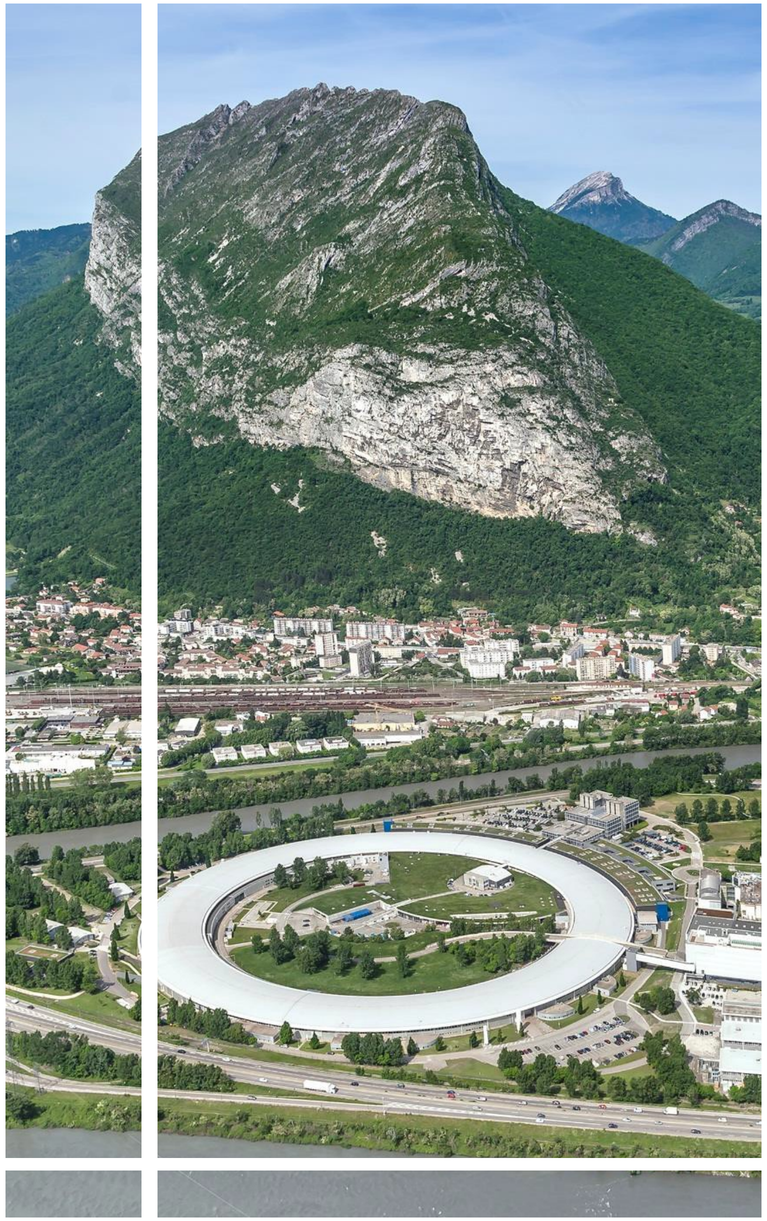
```python
from ewokscore import Task
from ewokscore import execute_graph

# Implement a workflow task
class SumTask(
    Task, input_names=["a"], optional_input_names=["b"], output_names=["result"]
):
    def run(self):
        result = self.inputs.a
        if self.inputs.b:
            result += self.inputs.b
        self.outputs.result = result


# Define a workflow with default inputs
nodes = [
    {
        "id": "task1",
        "task_type": "class",
        "task_identifier": "__main__.SumTask",
        "default_inputs": [{"name": "a", "value": 1}],
    },
    {
        "id": "task2",
        "task_type": "class",
        "task_identifier": "__main__.SumTask",
        "default_inputs": [{"name": "b", "value": 1}],
    },
    {
        "id": "task3",
        "task_type": "class",
        "task_identifier": "__main__.SumTask",
        "default_inputs": [{"name": "b", "value": 1}],
    },
]
links = [
    {
        "source": "task1",
        "target": "task2",
        "data_mapping": [{"source_output": "result", "target_input": "a"}],
    },
    {
        "source": "task2",
        "target": "task3",
        "data_mapping": [{"source_output": "result", "target_input": "a"}],
    },
]
workflow = {"graph": {"id": "testworkflow"}, "nodes": nodes, "links": links}

# Define task inputs
inputs = [{"id": "task1", "name": "a", "value": 10}]

# Execute a workflow (use a proper Ewoks task scheduler in production)
varinfo = {"root_uri": "/tmp/myresults"}  # optionally save all task outputs
result = execute_graph(workflow, varinfo=varinfo, inputs=inputs)
print(result)
```

EWOKS interfaces

- Desktop
- Web
- Shell
- Python

Developer usage like triggering workflows from the acquisition control system.

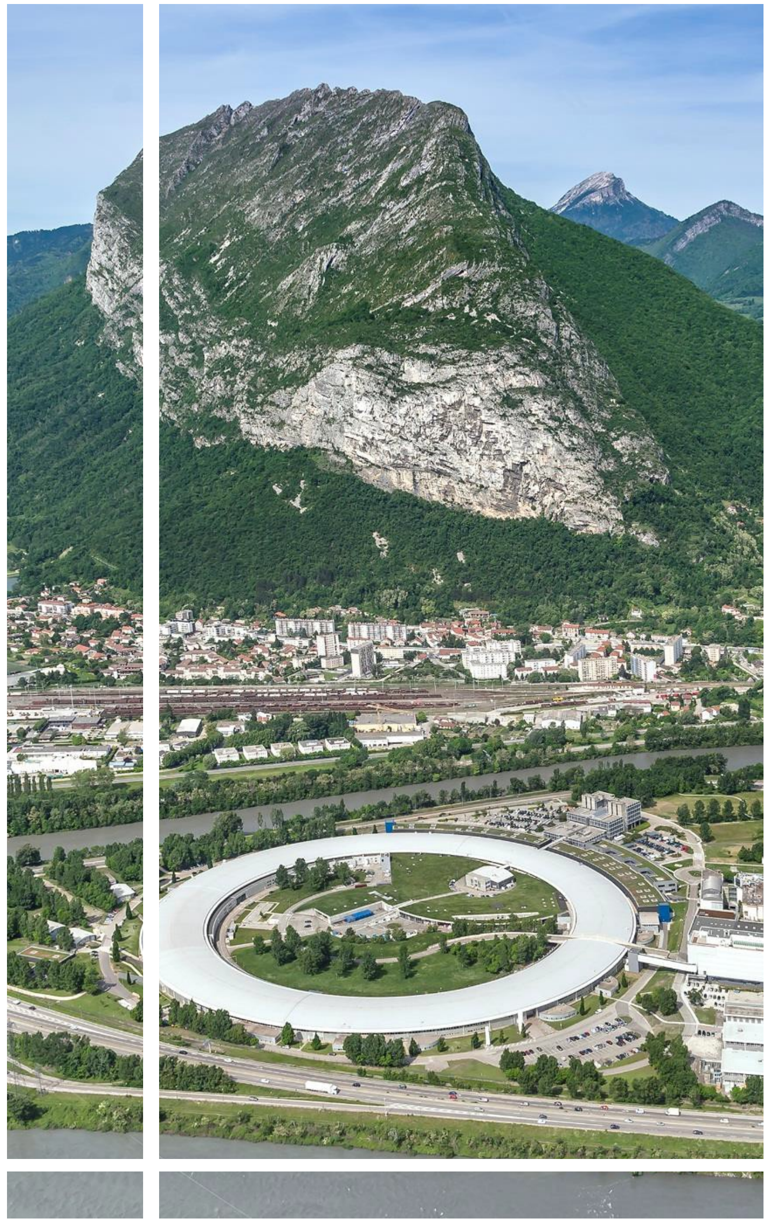
EWOKS DEMO



This EWOKS demo consists of two parts:

1. Getting started with EWOKS
2. Online diffraction data processing with blissdata and pyFAI

EWOKS DEMO

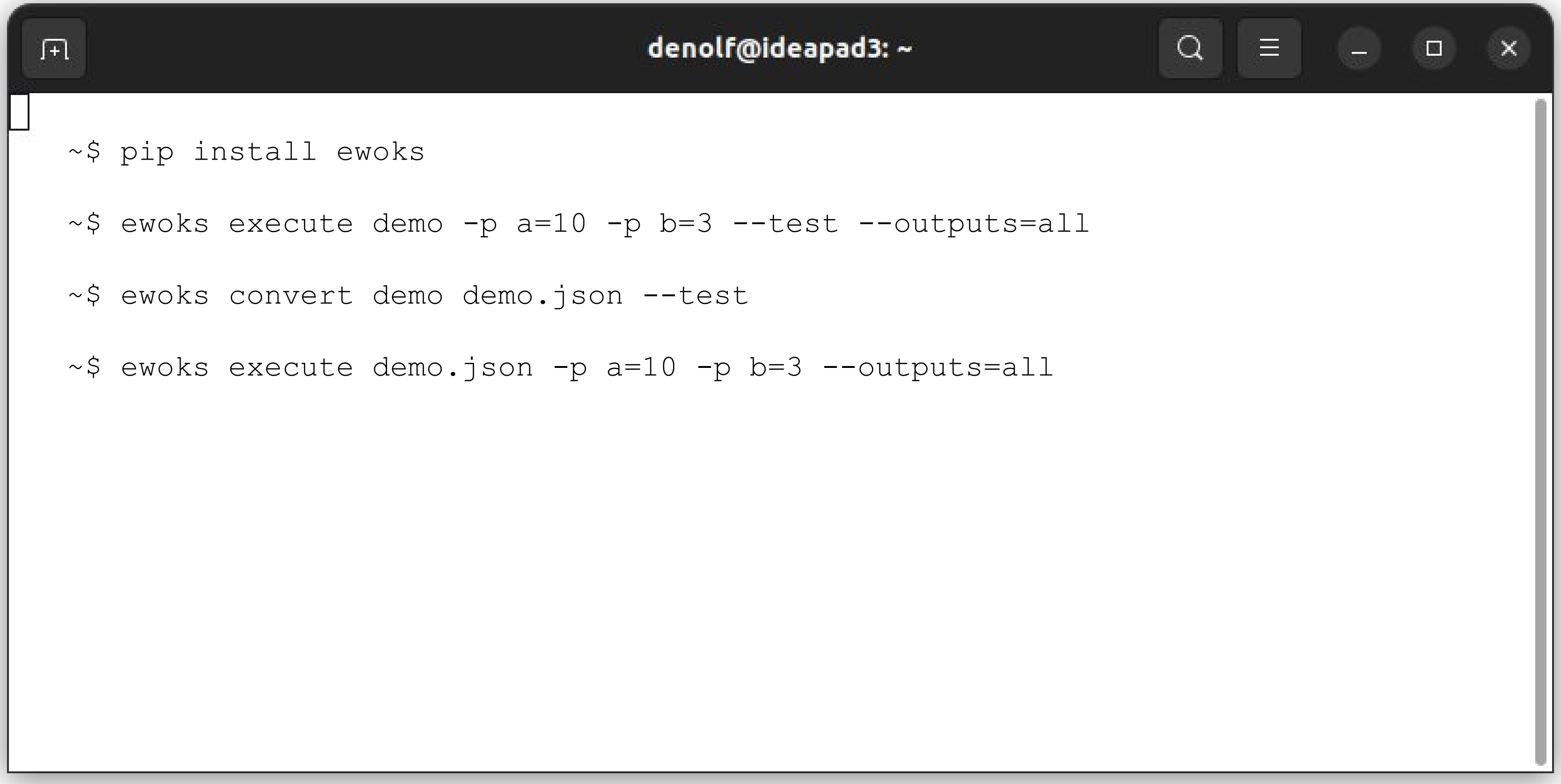


Getting Started with EWOKS

1. pip install ewoks
2. ewoks CLI
3. Web GUI

EWOKS DEMO

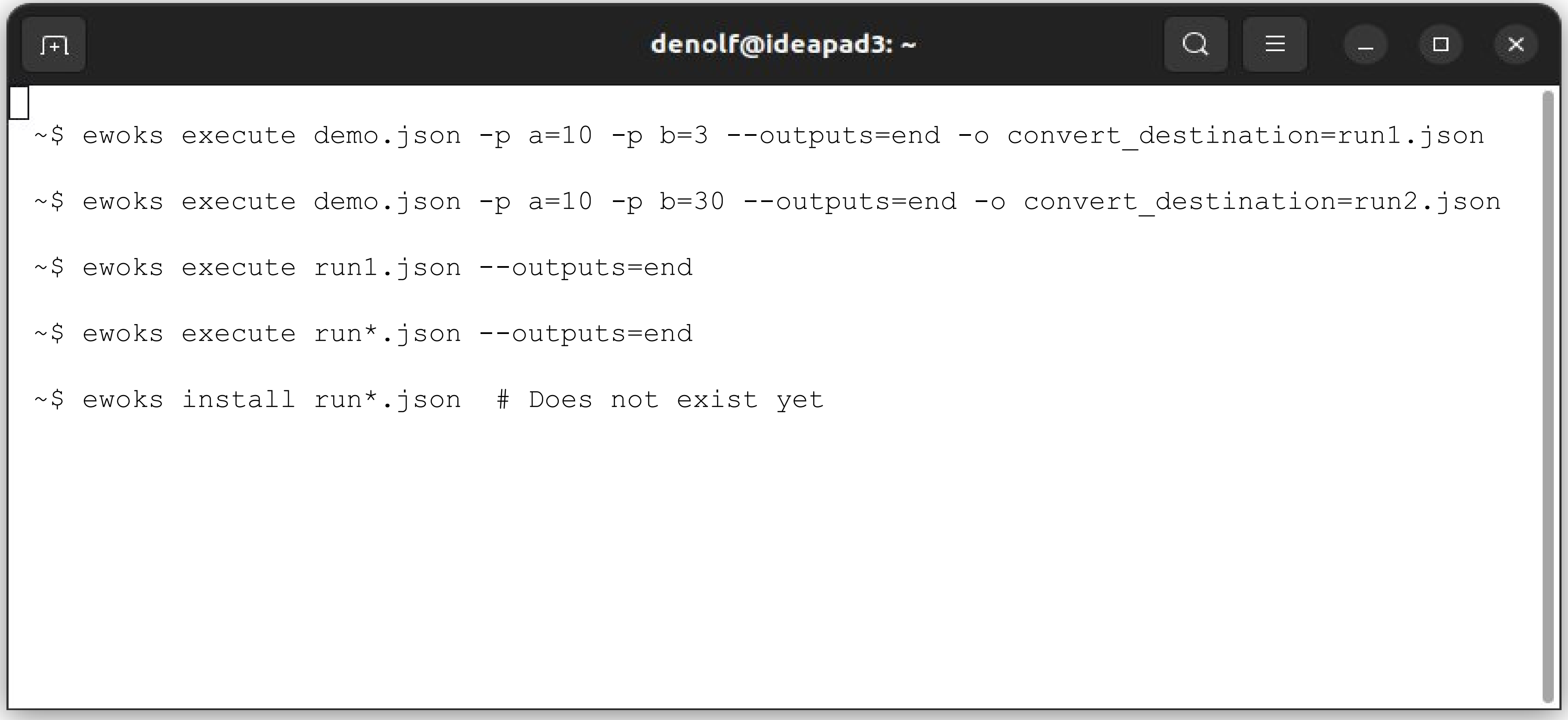
Getting started (CLI)



```
~$ pip install ewoks

~$ ewoks execute demo -p a=10 -p b=3 --test --outputs=all

~$ ewoks convert demo demo.json --test

~$ ewoks execute demo.json -p a=10 -p b=3 --outputs=all
```

EWOKS DEMO

FAIR data processing (CLI)

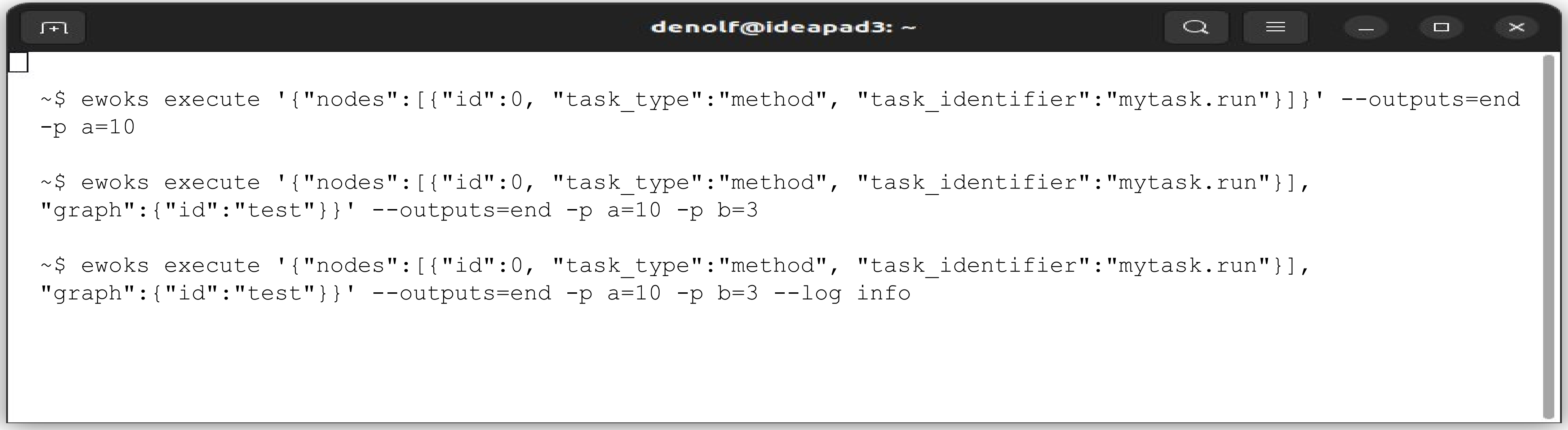


```
~$ ewoks execute demo.json -p a=10 -p b=3 --outputs=end -o convert_destination=run1.json

~$ ewoks execute demo.json -p a=10 -p b=30 --outputs=end -o convert_destination=run2.json

~$ ewoks execute run1.json --outputs=end

~$ ewoks execute run*.json --outputs=end

~$ ewoks install run*.json  # Does not exist yet
```

EWOKS DEMO

Python function and Ewoks events (CLI)

```
# mytask.py

def run(a, b=None):
     if b is None:
         return a
     return a + b
```

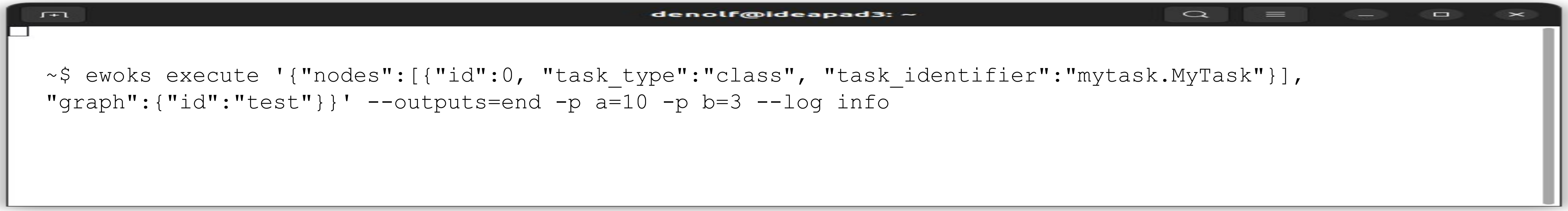


```
~$ ewoks execute '{"nodes":[{"id":0, "task_type":"method", "task_identifier":"mytask.run"}]}' --outputs=end
-p a=10

~$ ewoks execute '{"nodes":[{"id":0, "task_type":"method", "task_identifier":"mytask.run"}],
"graph":{"id":"test"}}' --outputs=end -p a=10 -p b=3

~$ ewoks execute '{"nodes":[{"id":0, "task_type":"method", "task_identifier":"mytask.run"}],
"graph":{"id":"test"}}' --outputs=end -p a=10 -p b=3 --log info
```

EWOKS DEMO

Python class (CLI)

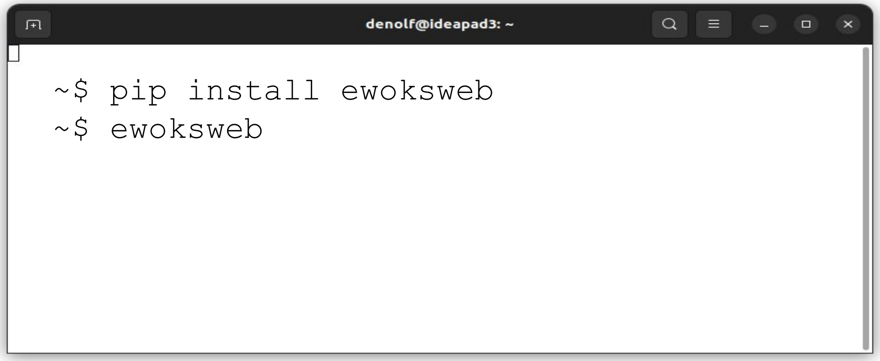
```
# mytask.py

from ewokscore import Task

class MyTask(
    Task,
    input_names=["a"],
    optional_input_names=["b"],
    output_names=["sum"],
):

    def run(self):
        if self.missing_inputs.b:
            self.outputs.sum = self.inputs.a
        else:
            self.outputs.sum = self.inputs.a + self.inputs.b
```



```
~$ ewoks execute '{"nodes":[{"id":0, "task_type":"class", "task_identifier":"mytask.MyTask"}],
"graph":{"id":"test"}}' --outputs=end -p a=10 -p b=3 --log info
```

EWOKS DEMO

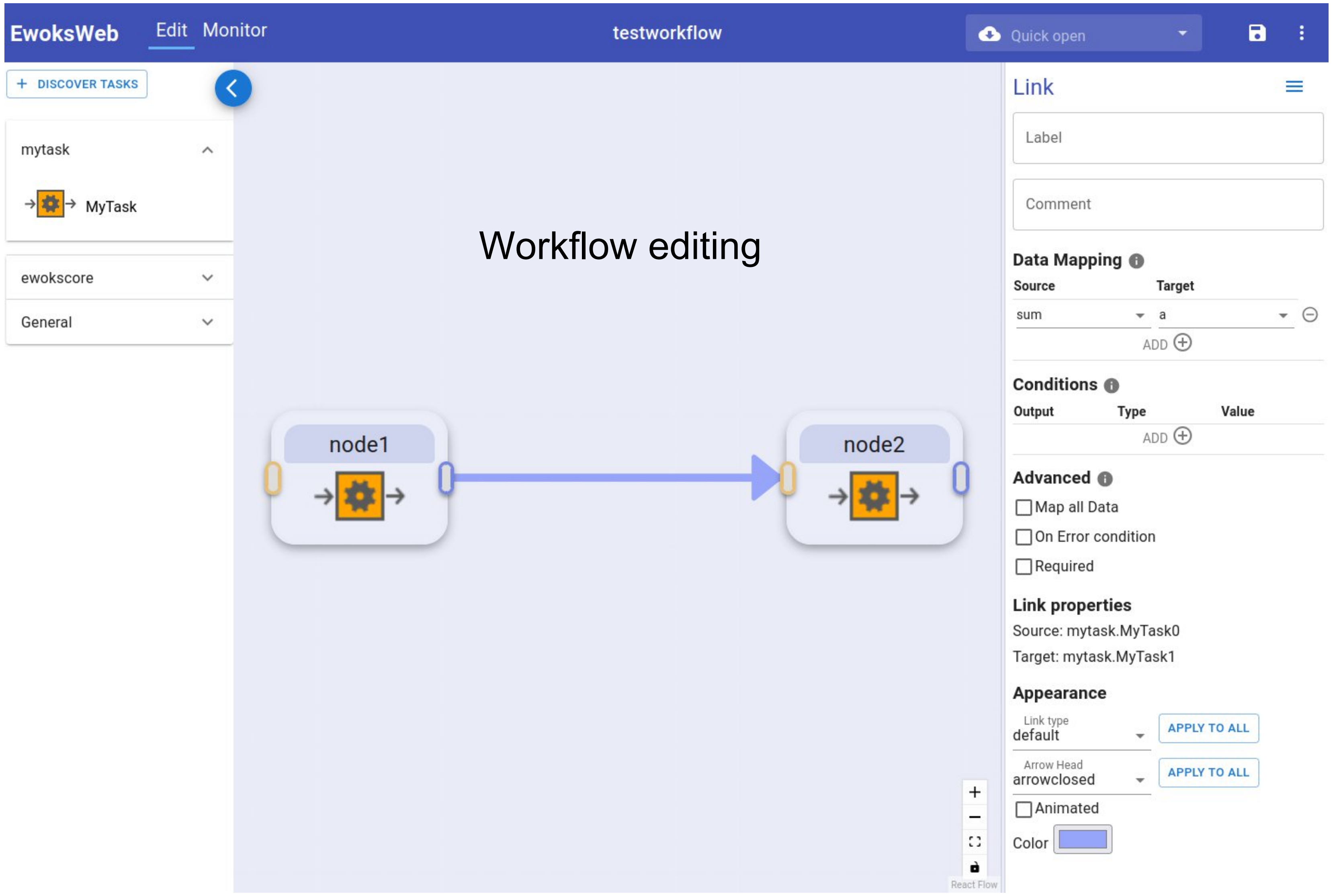
Graphical interface (Web GUI)



```
~$ pip install ewoksweb
~$ ewoksweb
```

Front-end: http://127.0.0.1:8000/
REST API: http://127.0.0.1:8000/api/docs or http://127.0.0.1:8000/api/redoc

EWOKS DEMO



Workflow editing

Link

Label

Comment

Data Mapping

Source	Target
sum	a

ADD

Conditions

Output	Type	Value

ADD

Advanced

☐ Map all Data
☐ On Error condition
☐ Required

Link properties

Source: mytask.MyTask0

Target: mytask.MyTask1

Appearance

Link type: default — APPLY TO ALL

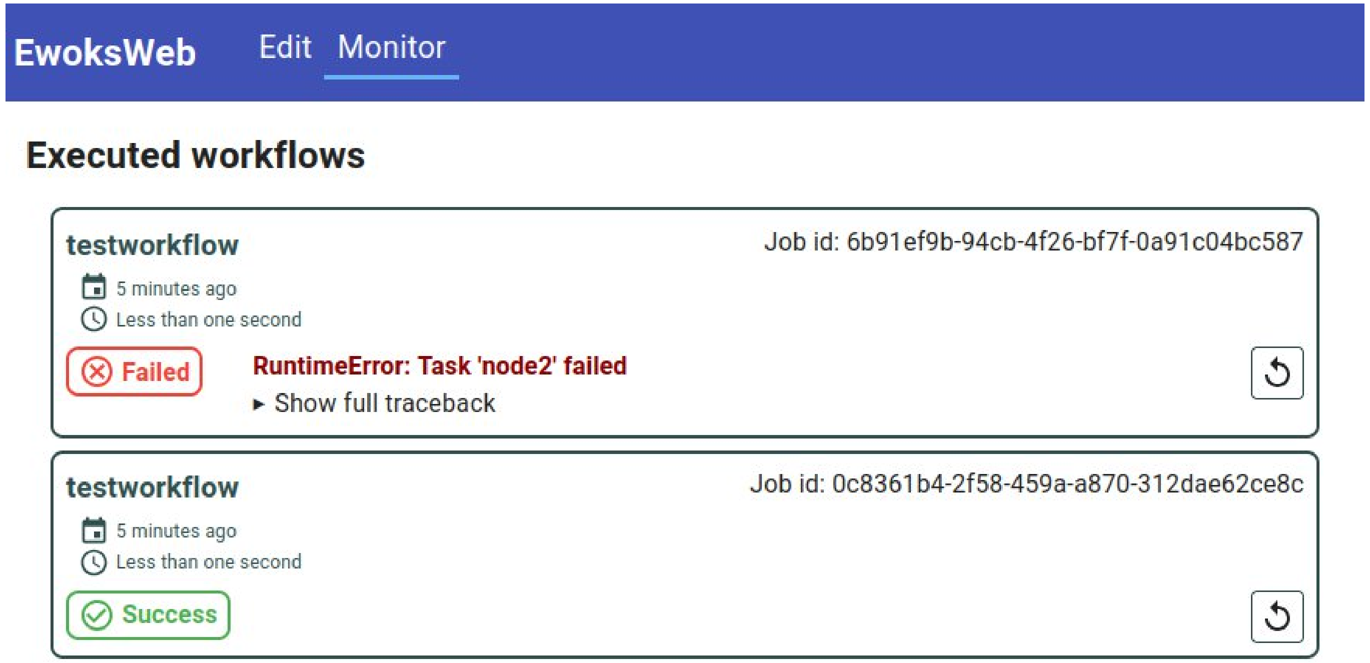
Arrow Head: arrowclosed — APPLY TO ALL

☐ Animated

Color

EWOKS DEMO

Workflow execution & monitoring

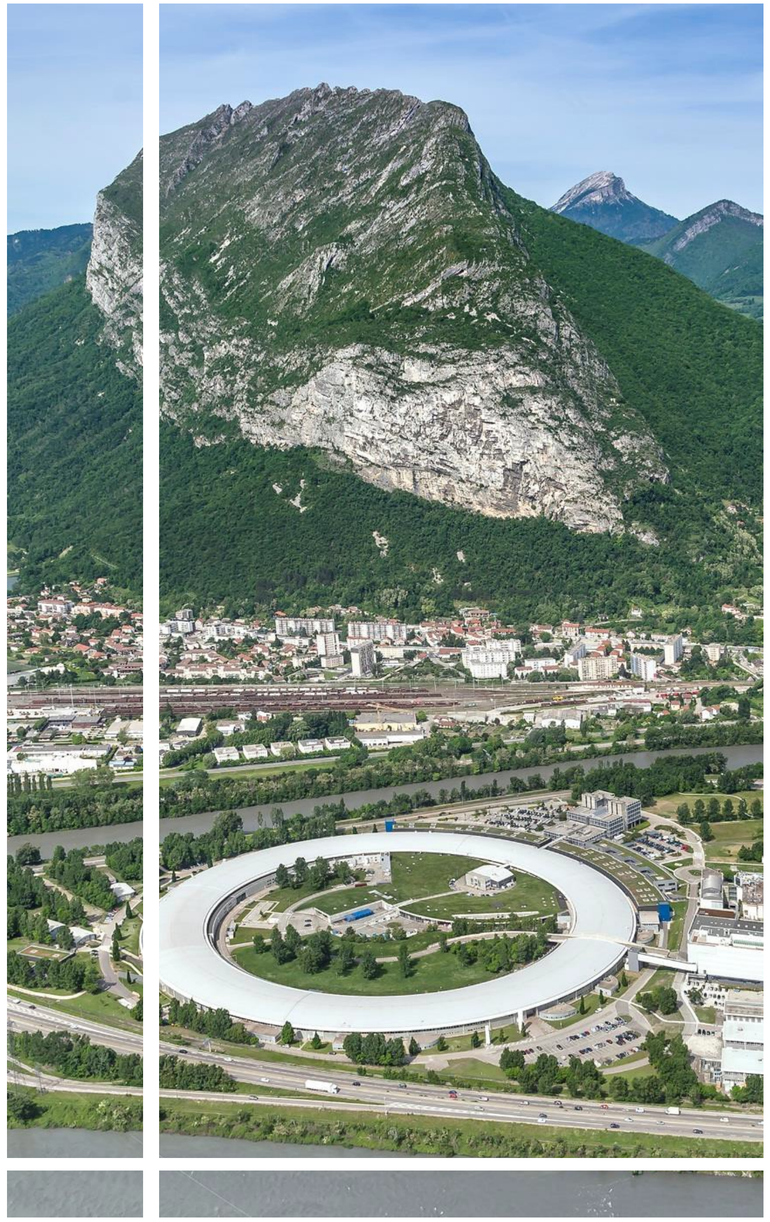


Executed workflows

testworkflow
Job id: 6b91ef9b-94cb-4f26-bf7f-0a91c04bc587
5 minutes ago
Less than one second
Failed
RuntimeError: Task 'node2' failed
▸ Show full traceback

testworkflow
Job id: 0c8361b4-2f58-459a-a870-312dae62ce8c
5 minutes ago
Less than one second
Success

EWOKS DEMO



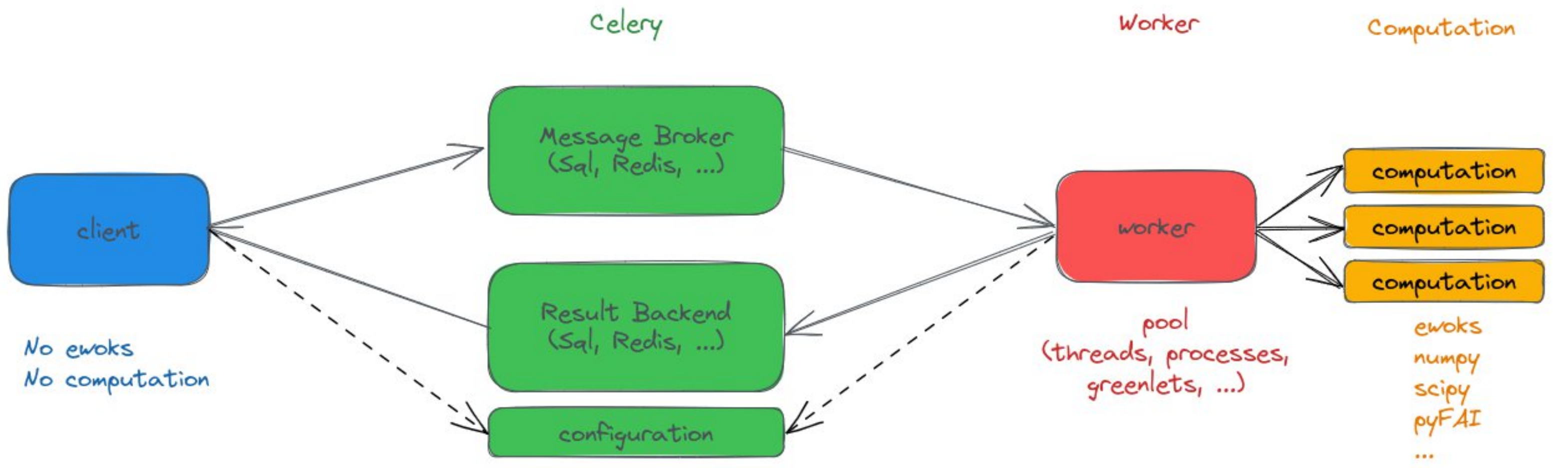
Online diffraction data processing with EWOKS, blissdata and pyFAI

1. pip install ewoksjob
2. celery configuration
3. publish PyFAI simulated images with blissdata
4. subscribe with blissdata
5. PyFAI integrate + plot

Monday 23 September (pyFAI user meeting)
14:00 → 15:00: 125 (Science Building)

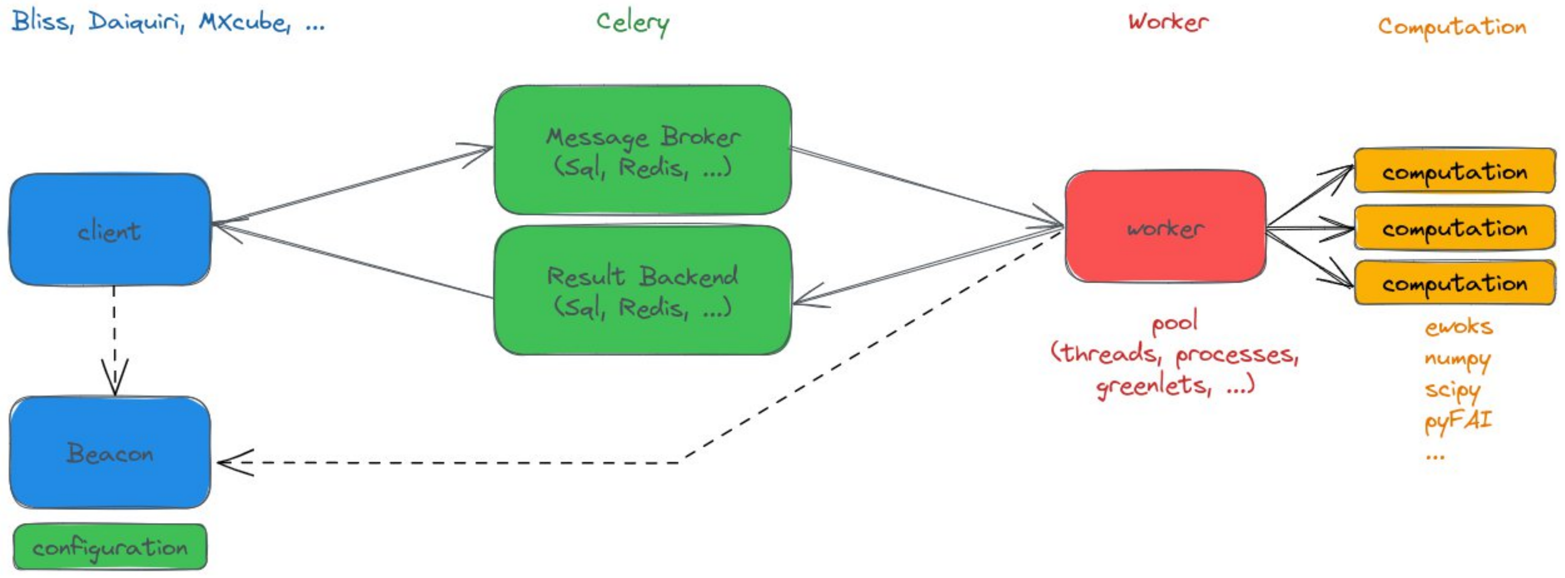
EWOKS DEMO

Online data processing in the demo today



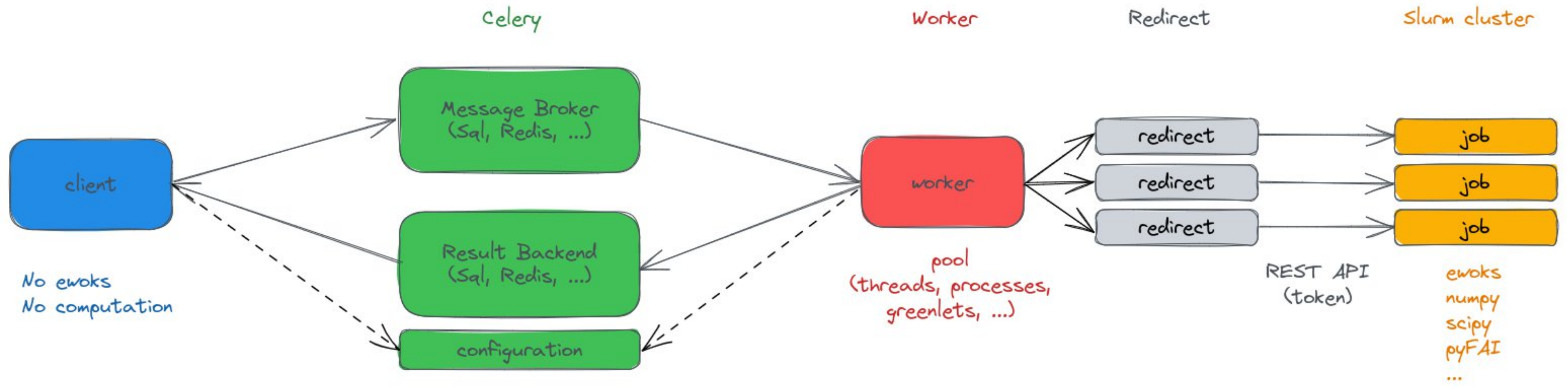
EWOKS DEMO

Online data processing at the ESRF



EWOKS DEMO

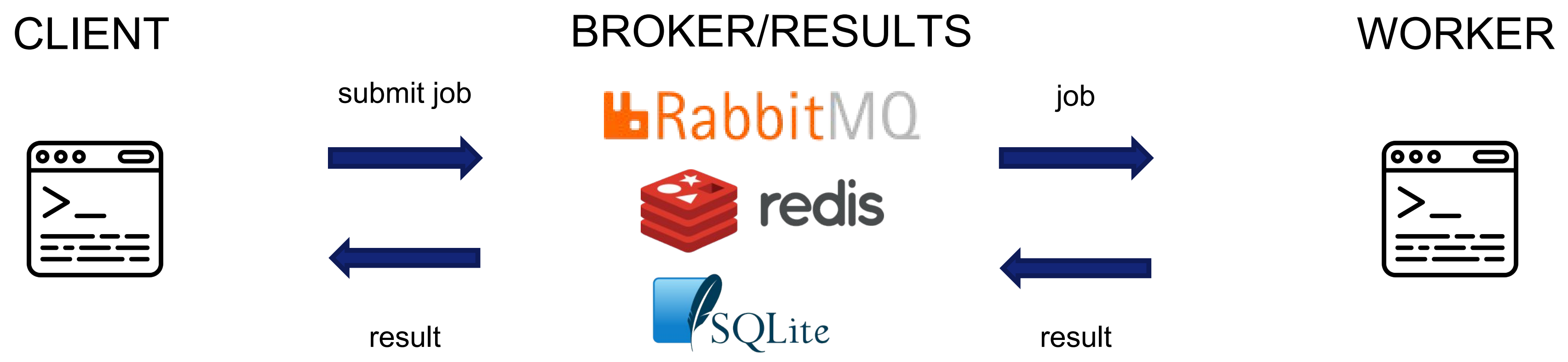
Online data processing using the cluster



The European Synchrotron | ESRF

EWOKS DEMO

Blissdata + EWOKS demo



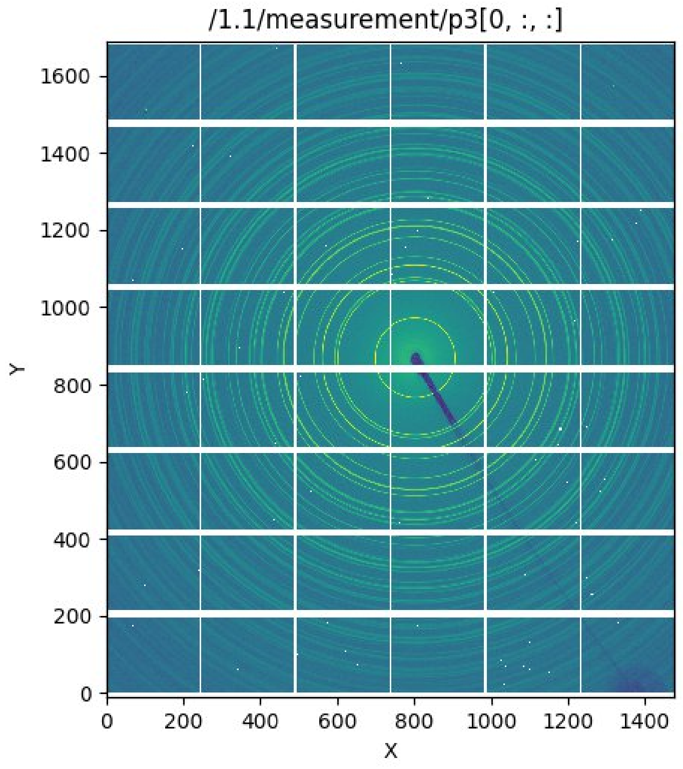
```
~$ pip install ewoksjob
~$ export EWOKS_CONFIG_URI=ewoksconfig.py
```

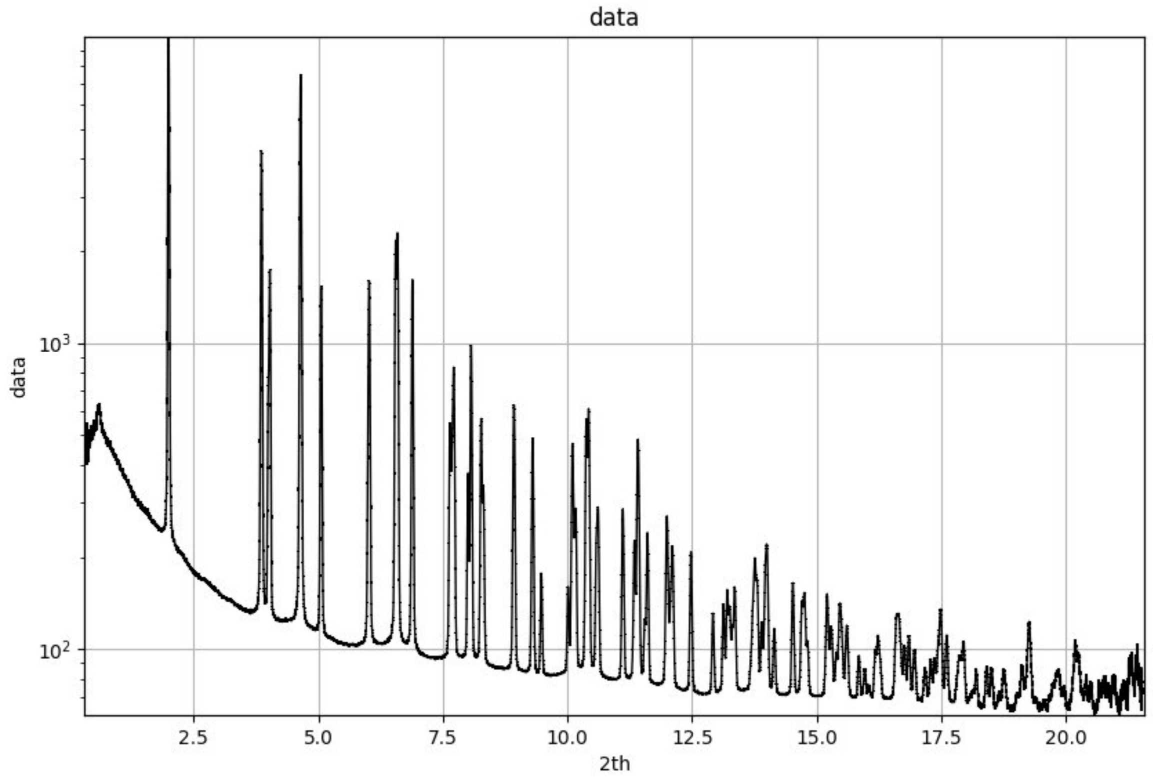
```
~$ pip install ewoksjob[worker]
~$ export EWOKS_CONFIG_URI=ewoksconfig.py
```

```
# ewoksconfig.py

broker_url = "sqla+sqlite:///celery.db"
result_backend = "db+sqlite:///celery_results.db"
```

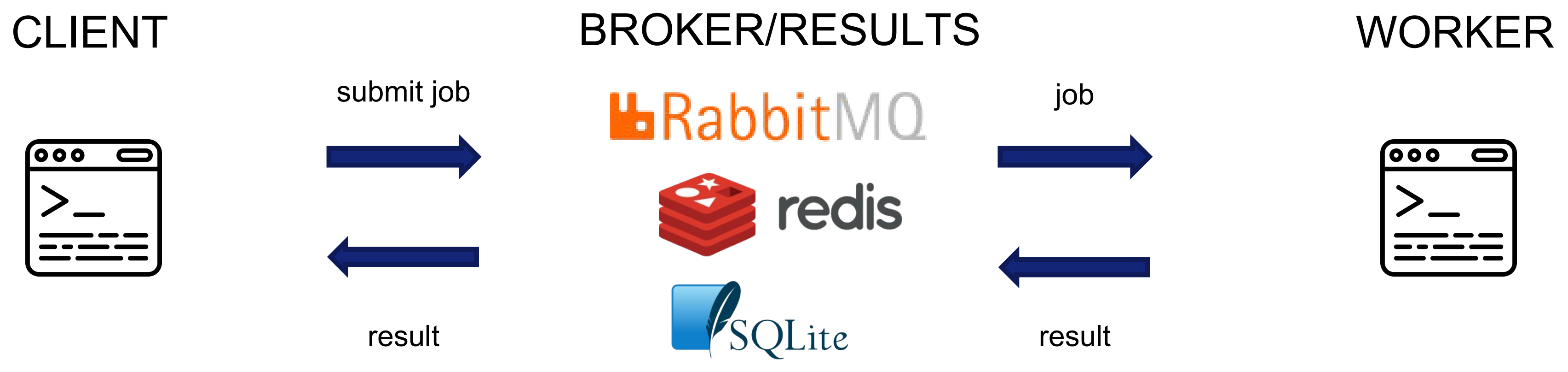
EWOKS DEMO

SAXS/WAXS: azimuthal integration of X-ray diffraction patterns from 2D detectors



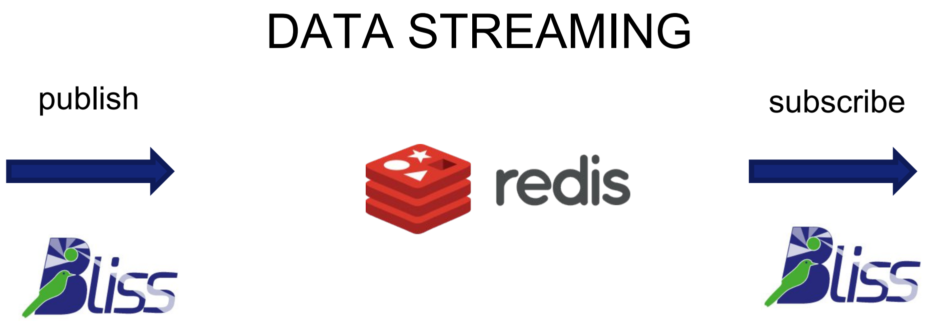


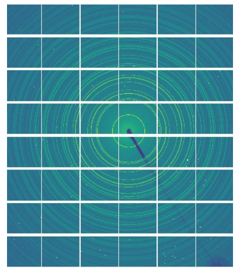
EWOKS DEMO



For every scan:
- **trigger workflow**
- **publish images one-by-one**

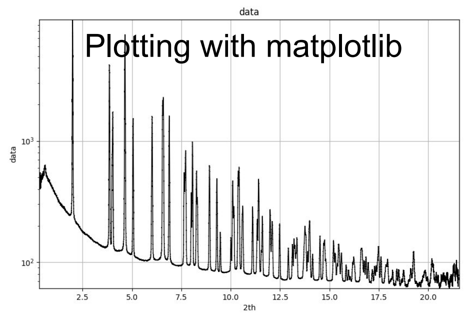
For every scan:
- **integrate images one-by-one**
- **plot integrated data**




Simulate Images

Azimuthal Integration

PyFAI Fast Azimuthal Integration

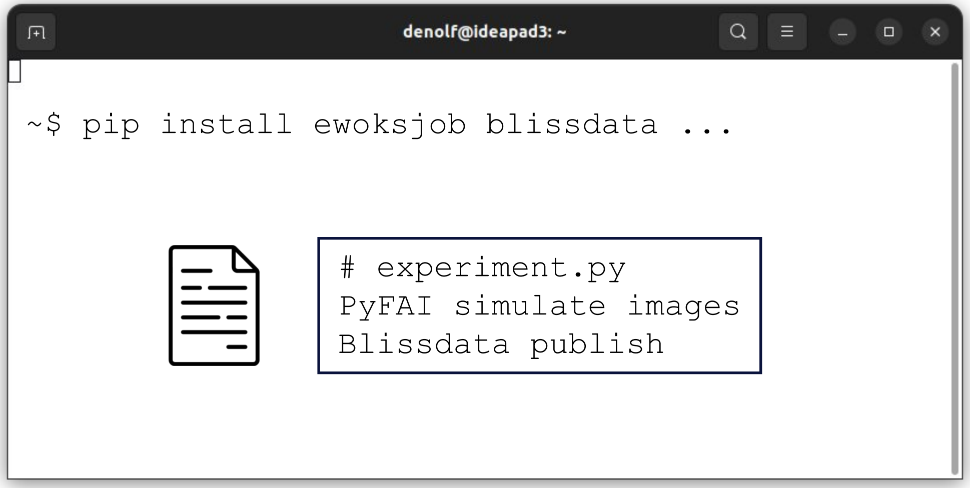


EWOKS DEMO

Online data processing: ewoksjob + blissdata + pyFAI

```
# ewoksconfig.py

broker_url = "sqla+sqlite:///celery.db"
result_backend = "db+sqlite:///celery_results.db"
```

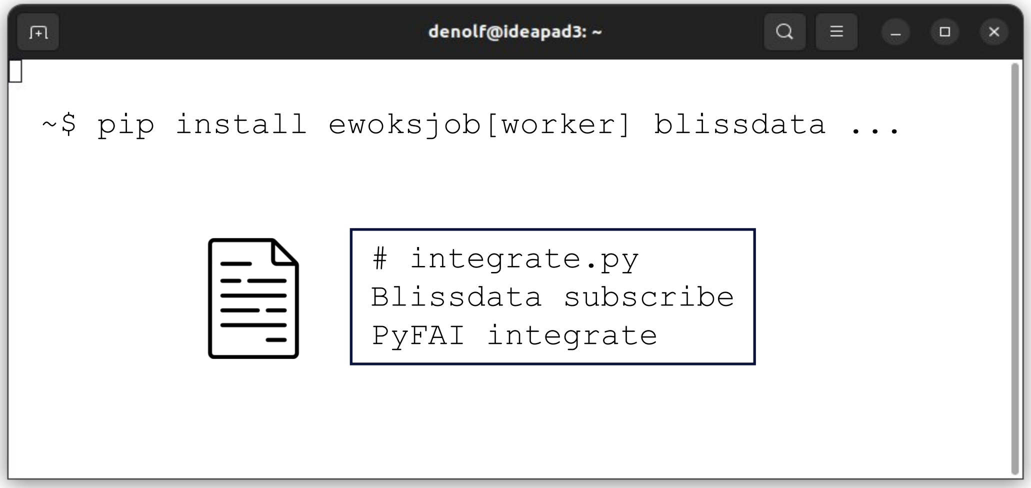
CLIENT



```
~$ pip install ewoksjob blissdata ...
```

```
# experiment.py
PyFAI simulate images
Blissdata publish
```

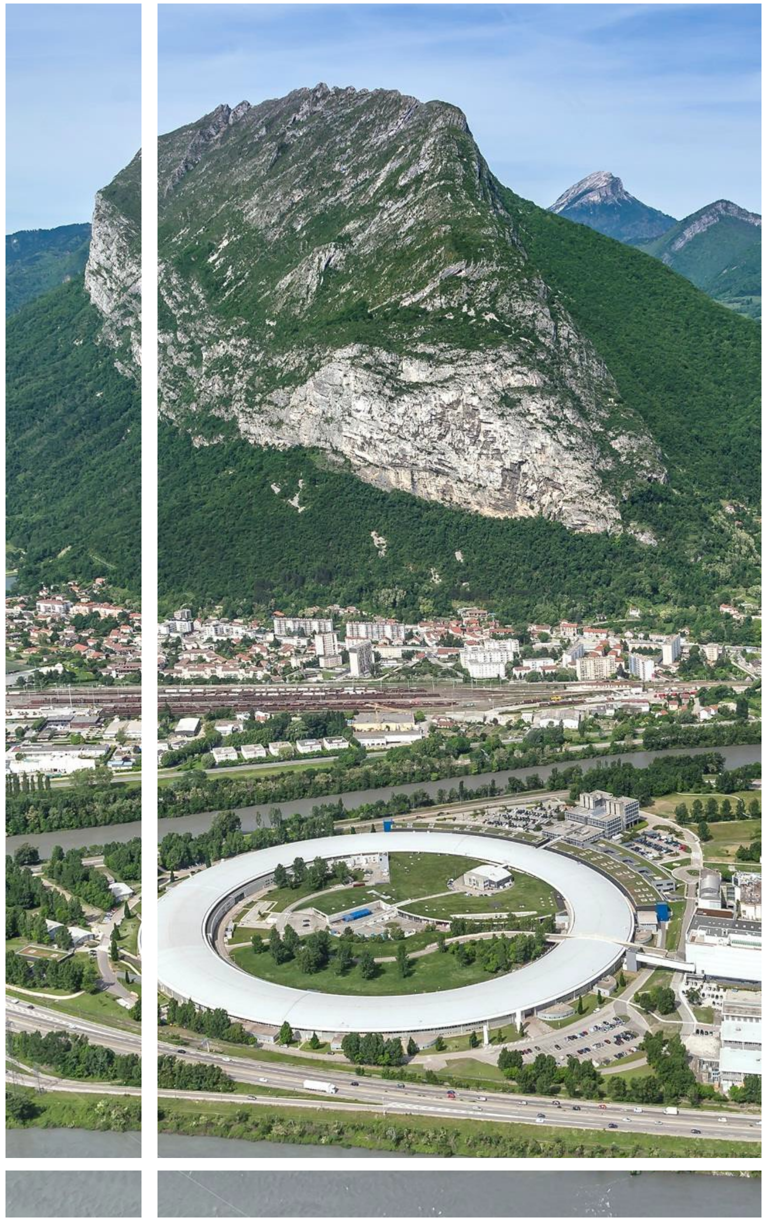
WORKER



```
~$ pip install ewoksjob[worker] blissdata ...
```

```
# integrate.py
Blissdata subscribe
PyFAI integrate
```

EWOKS DEMO



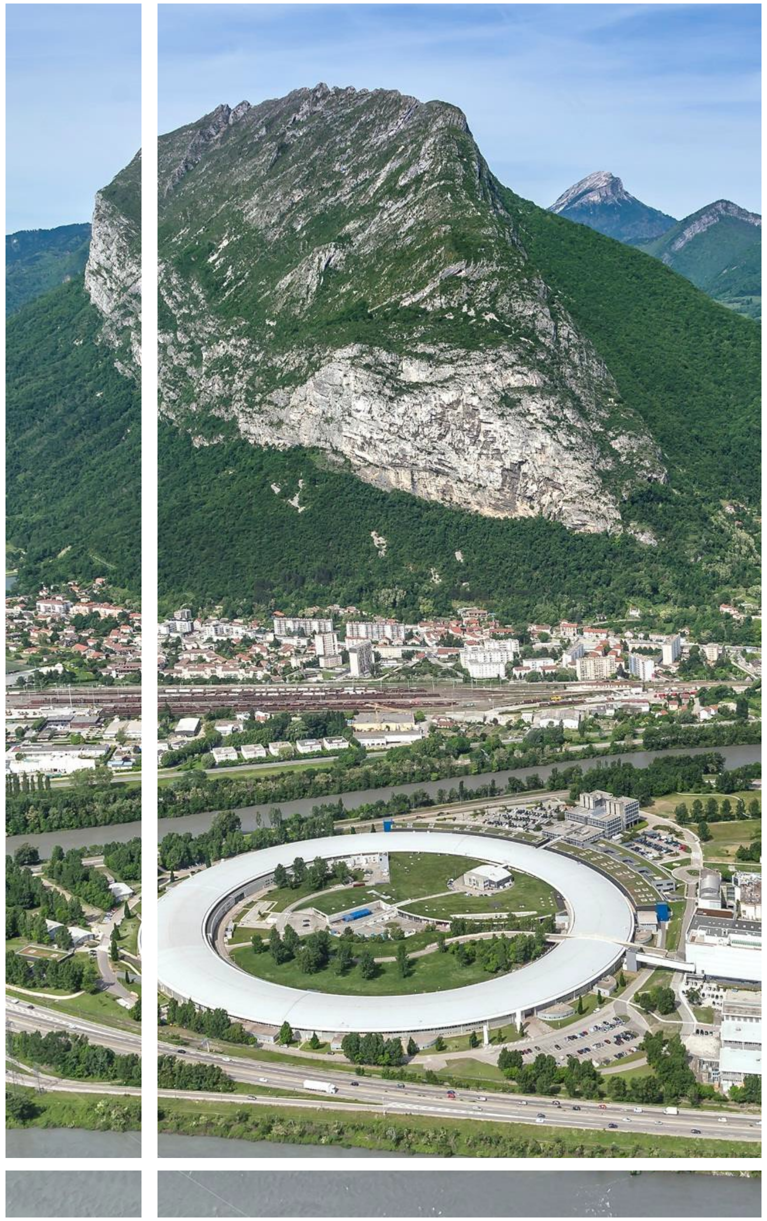
Online diffraction data processing with EWOKS, blissdata and pyFAI

Clone this repository and make it the current working directory
(instructions in README.md)

```
git clone https://gitlab.esrf.fr/workflow/ewokstutorials/ewokswithblissdata.git
cd ewokswithblissdata
```

The European Synchrotron | ESRF

CONCLUSION

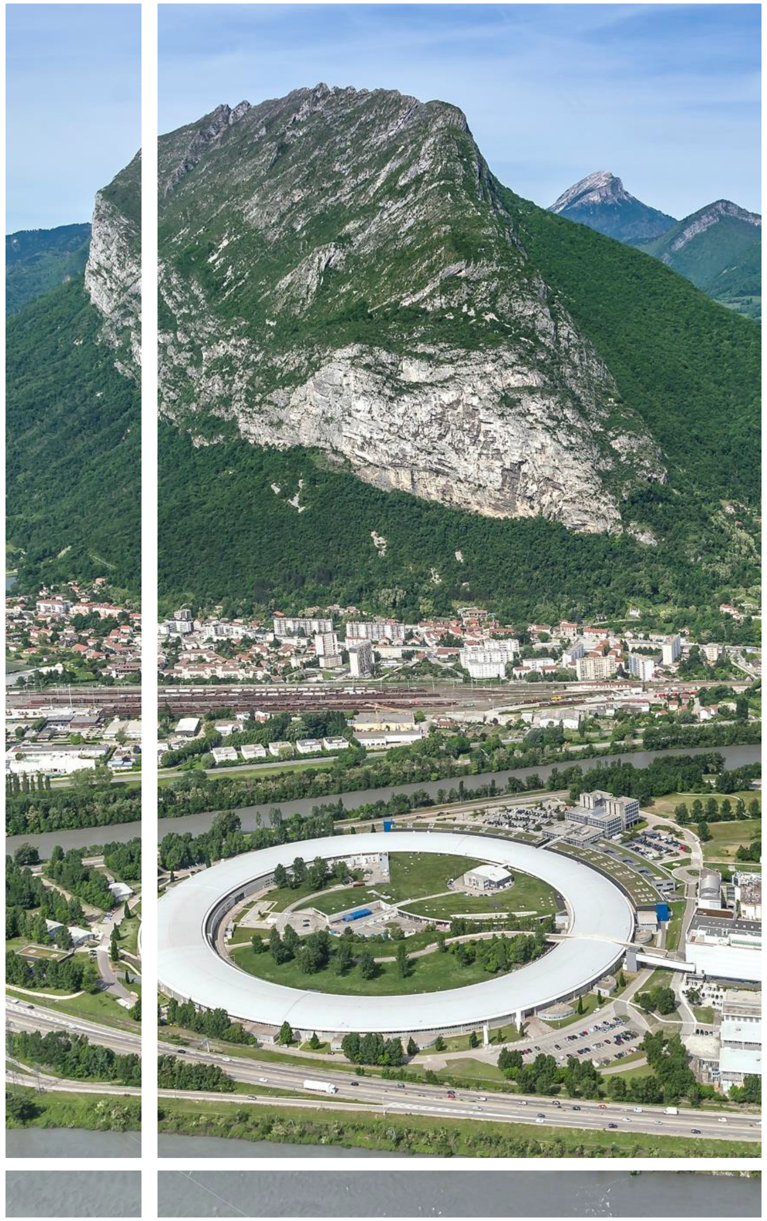


What did we cover:

1. pip install ewoks → **execute** workflows
2. pip install ewoksweb → **create** workflows
3. pip install ewoksjob[worker] → **job scheduling** of workflows
4. pip install blissdata → **data streaming**

Whatever technology you use: simplicity is a must!

CONCLUSION



Main documentation:
https://ewoks.esrf.fr

More detailed tutorial:
https://ewoksfordevs.readthedocs.io/