Rucio-SWAN Integration Project

Google Summer of Code 2020 with CERN-HSF

Muhammad Aditya Hilmy
mhilmy@hey.com

THE BIG QUESTION

How can we help scientists work productively in the Exabyte-scale era?

Rucio

- Keeps track of data locations
- Moves data around as needed
- De facto standard for scientific data management

SWAN

- Online interactive Jupyter notebook
- No installation needed
- Enables collaboration through notebook sharing

INTEGRATING RUCIO AND SWAN

Integrating those would enable scientists to perform analyses on large datasets with ease. No installation and configuration needed.

What it takes to use Rucio-managed data in SWAN

- Download the data on your local machine and upload it back to SWAN
    - This is the simplest way of doing this
    - Suitable for end user analysis
    - If the work is shared, everyone running the notebook should do the same
    - There is no association between the Logical File Name (LFN) and notebook

What it takes to use Rucio-managed data in SWAN (2)

- Download the data directly from SWAN using Rucio CLI client
  - This is more practical than download-reupload
  - Requires users to install `rucio-clients`
  - Adds clutter to the notebook
  - If the work is shared, everyone running the notebook should do the same
  - There is no association between the Logical File Name (LFN) and notebook

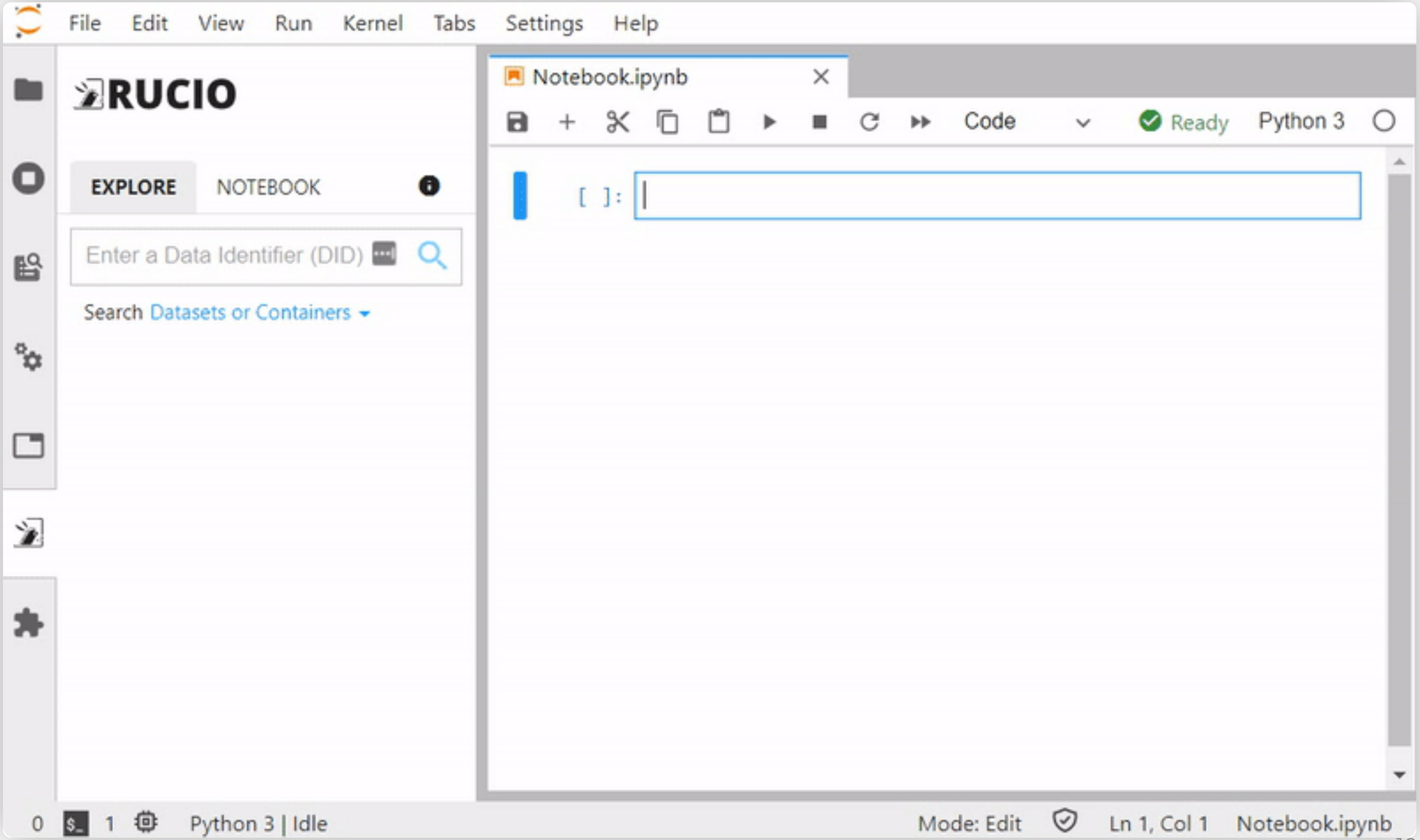
What it takes to use Rucio-managed data in SWAN (3)

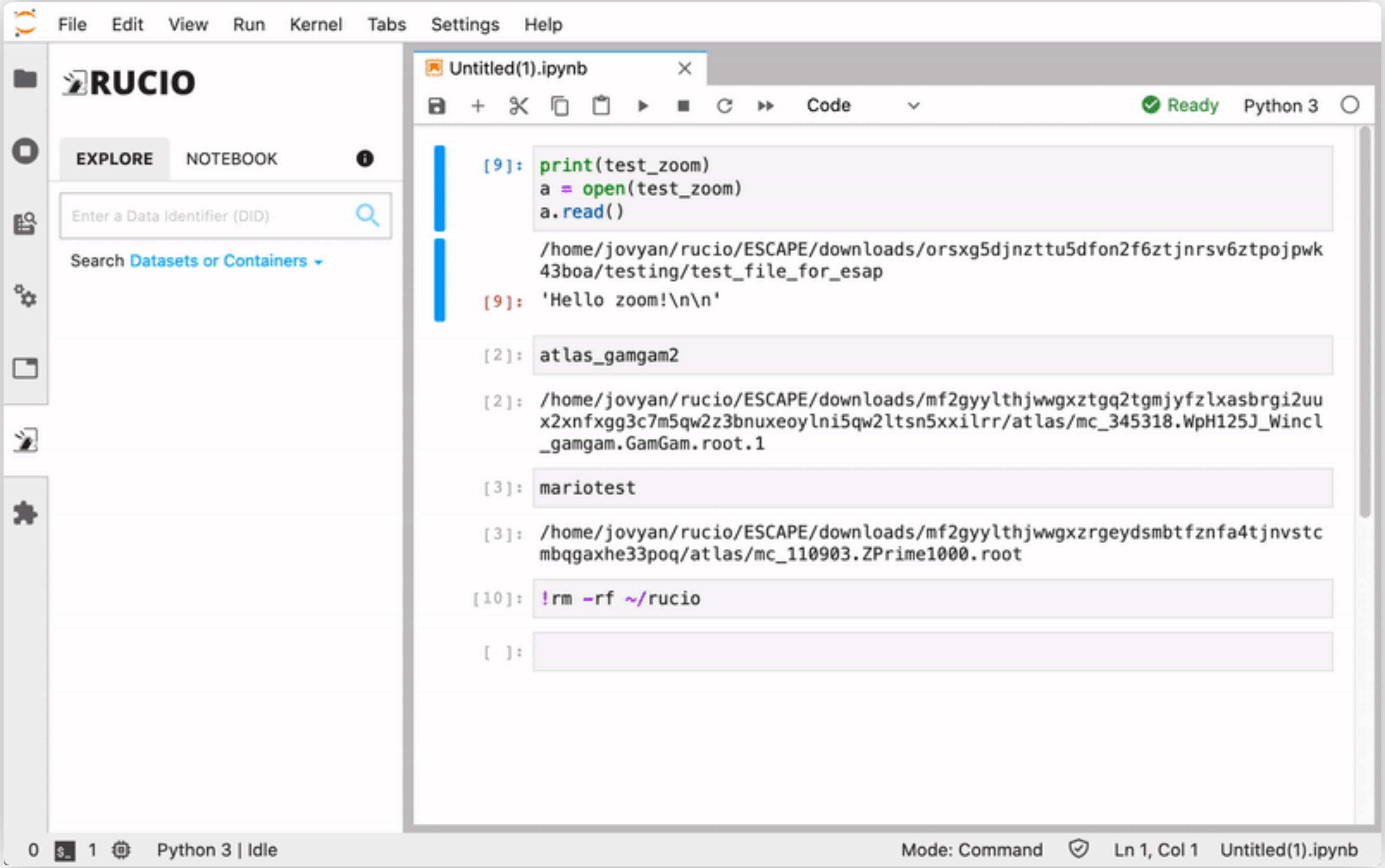
- Use Rucio to move the data to a network storage attached to SWAN
  - Users can use rucio-cli or Rucio web interface
    - The CLI and web interface are feature-packed
    - But might not be relevant to scientists
  - There is no association between the Logical File Name (LFN) and notebook

What if, we can make it as easy as online shopping?

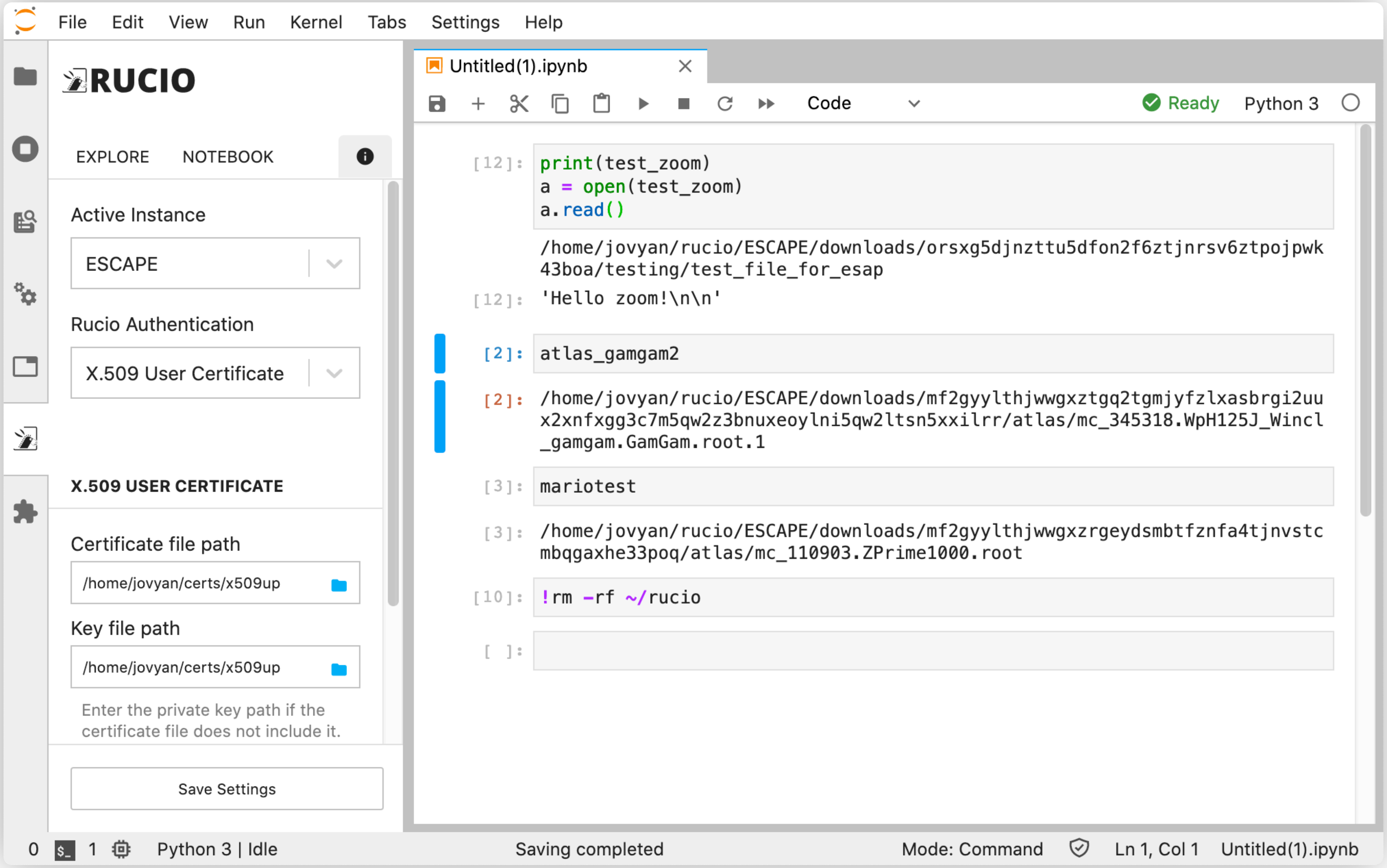
Introducing, Rucio JupyterLab Extension.

(I haven’t thought of a cool name for this project, so let’s stick to this extraordinarily ordinary name)


File
Edit
View
Run
Kernel
Tabs
Settings
Help
RUCIO
EXPLORE
NOTEBOOK
Enter a Data Identifier (DID)
Search Datasets or Containers
Notebook.ipynb
Code
Ready
Python 3
[ ]:
0
1
Python 3 | Idle
Mode: Edit
Ln 1, Col 1
Notebook.ipynb



```
[9]: print(test_zoom)
     a = open(test_zoom)
     a.read()

     /home/jovyan/rucio/ESCAPE/downloads/orsxg5djnzttu5dfon2f6ztjnrsv6ztpojpwk
     43boa/testing/test_file_for_esap
[9]: 'Hello zoom!\n\n'

[2]: atlas_gamgam2

[2]: /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjwwgxztgq2tgmjyfzlxasbrgi2uu
     x2xnfxgg3c7m5qw2z3bnuxeoylni5qw2ltsn5xxilrr/atlas/mc_345318.WpH125J_Wincl
     _gamgam.GamGam.root.1

[3]: mariotest

[3]: /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjwwgxzrgeydsmbtfznfa4tjnvstc
     mbqgaxhe33poq/atlas/mc_110903.ZPrime1000.root

[10]: !rm -rf ~/rucio

[ ]:
```



Active Instance: ESCAPE

Rucio Authentication: X.509 User Certificate

X.509 USER CERTIFICATE

Certificate file path: /home/jovyan/certs/x509up

Key file path: /home/jovyan/certs/x509up

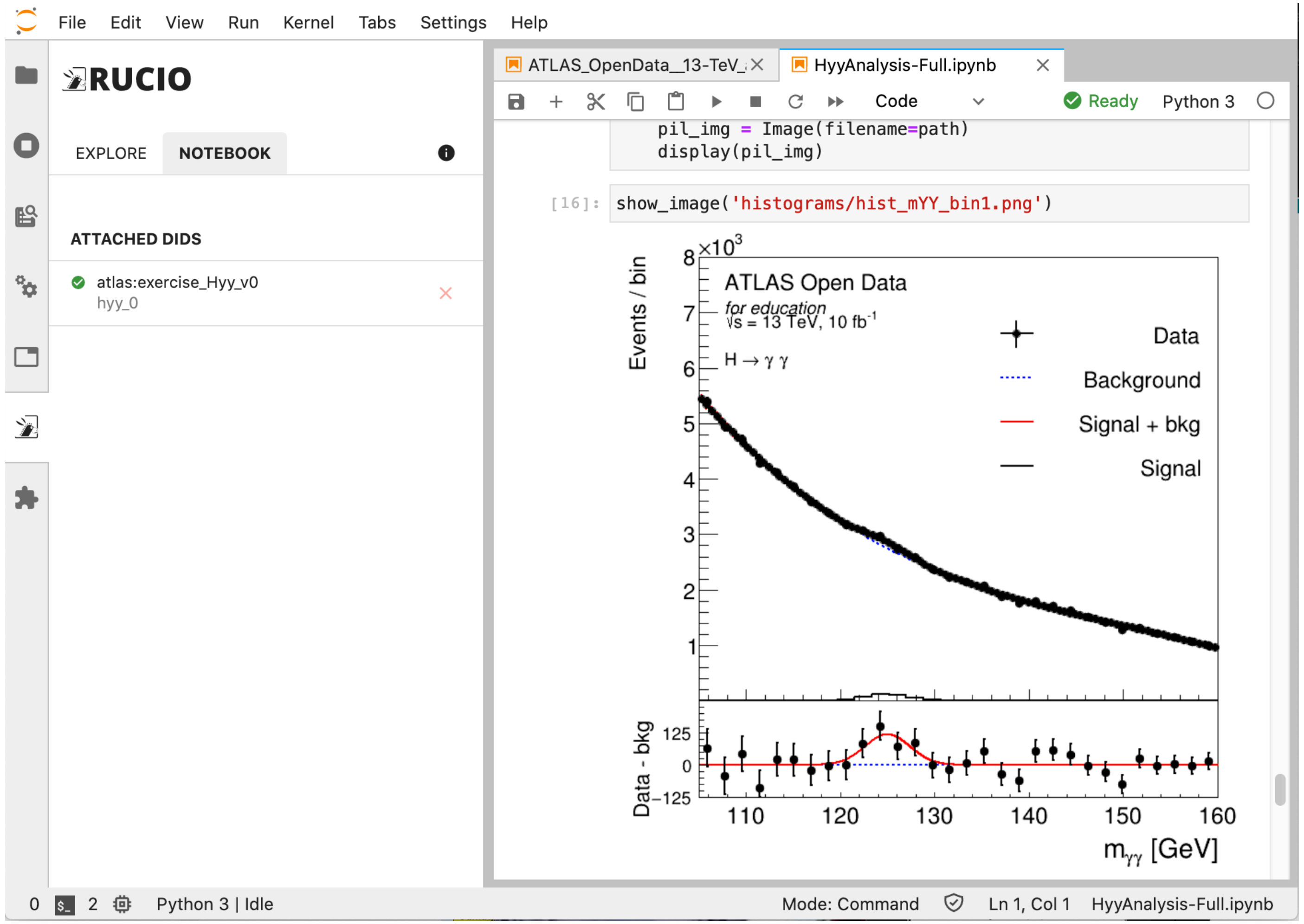
Enter the private key path if the certificate file does not include it.

```
[12]: print(test_zoom)
      a = open(test_zoom)
      a.read()

      /home/jovyan/rucio/ESCAPE/downloads/orsxg5djnzttu5dfon2f6ztjnrsv6ztpojpwk
      43boa/testing/test_file_for_esap
[12]: 'Hello zoom!\n\n'

[2]: atlas_gamgam2

[2]: /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjwwgxztgq2tgmjyfzlxasbrgi2uu
     x2xnfxgg3c7m5qw2z3bnuxeoylni5qw2ltsn5xxilrr/atlas/mc_345318.WpH125J_Wincl
     _gamgam.GamGam.root.1

[3]: mariotest

[3]: /home/jovyan/rucio/ESCAPE/downloads/mf2gyylthjwwgxzrgeydsmbtfznfa4tjnvstc
     mbqgaxhe33poq/atlas/mc_110903.ZPrime1000.root

[10]: !rm -rf ~/rucio

[ ]:
```

0 1 Python 3 | Idle   Saving completed   Mode: Command   Ln 1, Col 1   Untitled(1).ipynb

SHOWCASE

ATLAS Open Data

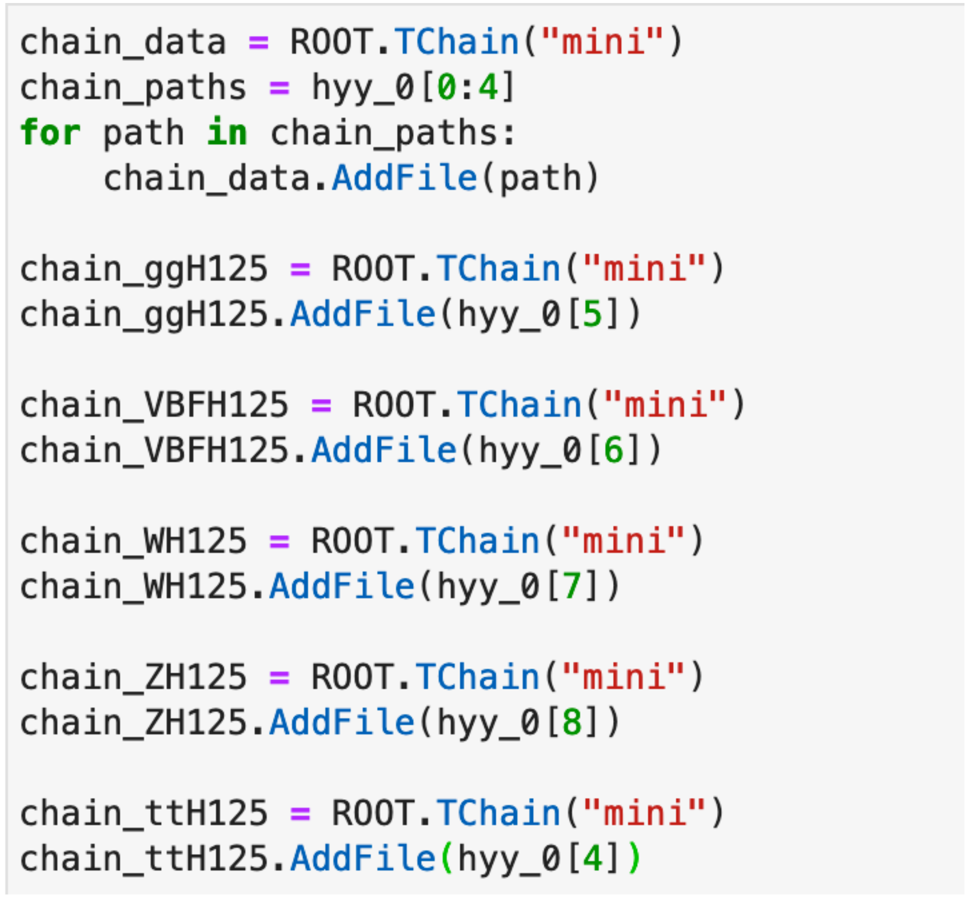


```
    pil_img = Image(filename=path)
    display(pil_img)

[16]: show_image('histograms/hist_mYY_bin1.png')
```

- Hyy analysis using ATLAS Open Data
- No hardcoded path to file

SHOWCASE

ATLAS Open Data (2)



```
chain_data = ROOT.TChain("mini")
chain_paths = hyy_0[0:4]
for path in chain_paths:
    chain_data.AddFile(path)

chain_ggH125 = ROOT.TChain("mini")
chain_ggH125.AddFile(hyy_0[5])

chain_VBFH125 = ROOT.TChain("mini")
chain_VBFH125.AddFile(hyy_0[6])

chain_WH125 = ROOT.TChain("mini")
chain_WH125.AddFile(hyy_0[7])

chain_ZH125 = ROOT.TChain("mini")
chain_ZH125.AddFile(hyy_0[8])

chain_ttH125 = ROOT.TChain("mini")
chain_ttH125.AddFile(hyy_0[4])
```

- This is the code to load the ROOT files (in PyROOT).
  - No need to know the file paths
- `hyy_0` is an array of paths to files in dataset `atlas:exercise_Hyy_v0` in ESCAPE datalake.
  - The paths are injected by the extension automatically.

Notebook preview on
https://nbviewer.jupyter.org/gist/didithilmy/28400804ed55b1e4ff683902fa1cc58d

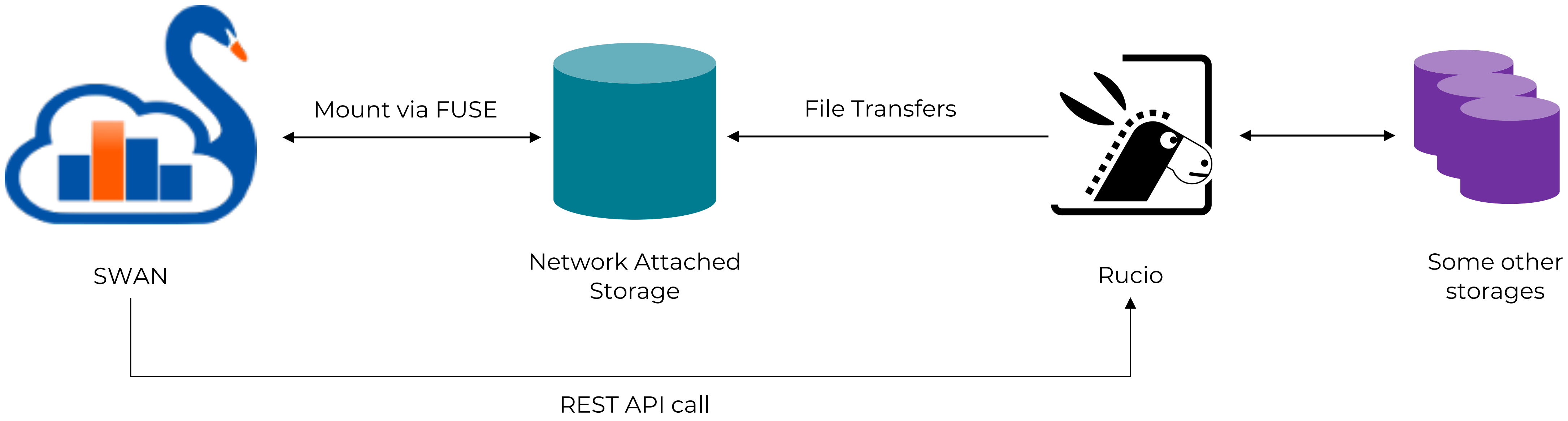
Key Features

- Browse Rucio data from the Lab sidebar
- Replicate data with just one click
- Resolves file path automagically
- Inject path to notebook as a variable
- Supports two methods of authentication (currently):
    - Username & Password
    - X.509 User Certificate (or Proxy)
- Supports two modes of operation:
    - Replica mode: uses network-attached storage as a Rucio Storage Element (RSE), utilizes Rucio's file transfer capability.
    - Download mode: downloads data directly to the user's directory using Rucio clients.
- Remote configuration

Replica Mode

- Uses a Rucio Storage Element (RSE) mounted via FUSE, shared with multiple users.
- Uses existing file transfer infrastructure to make files available.
- Extension creates a new replication rule when making data available.
- Suitable for larger installations with pre-existing data transfer infrastructure.

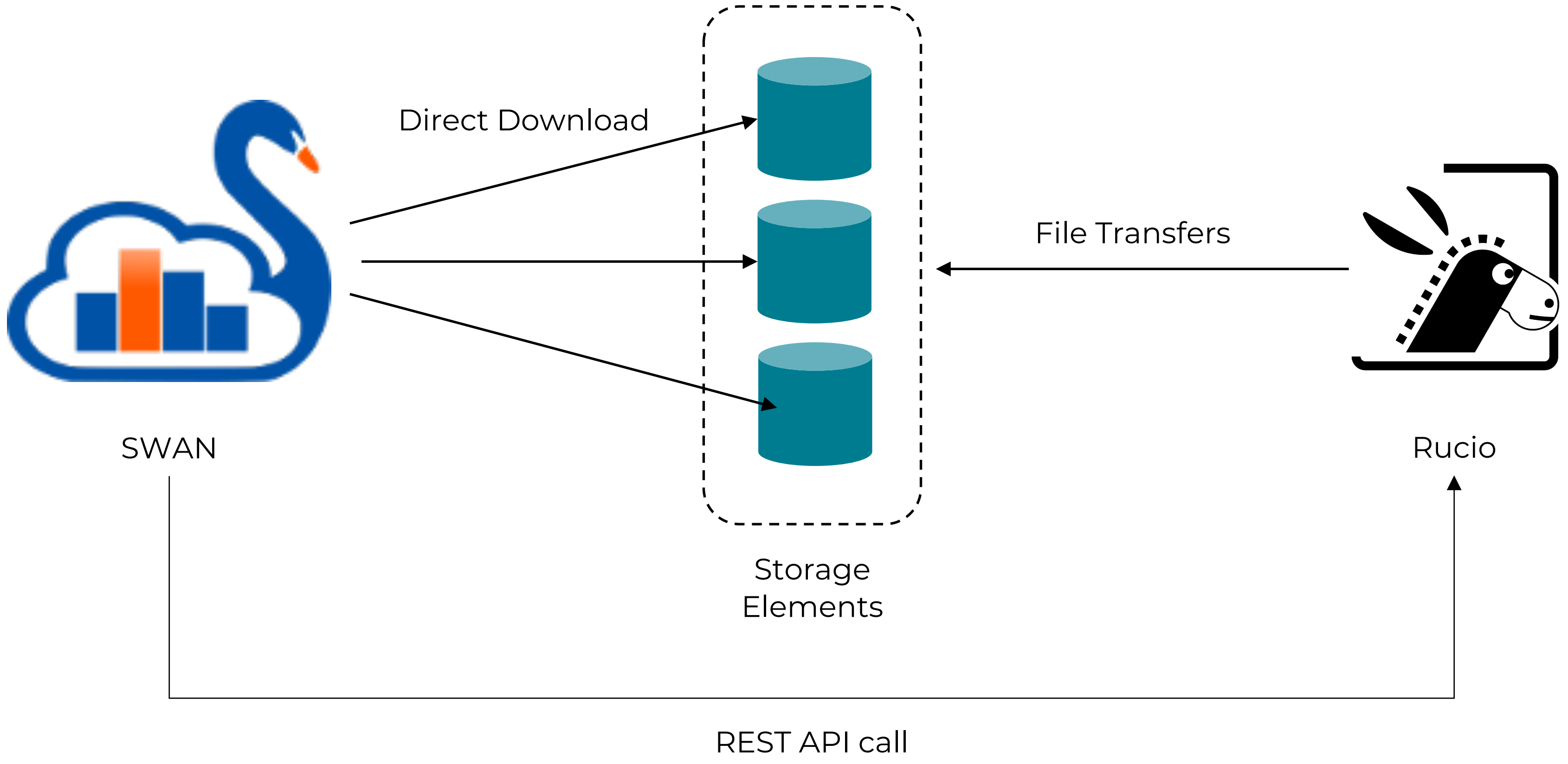
Replica Mode



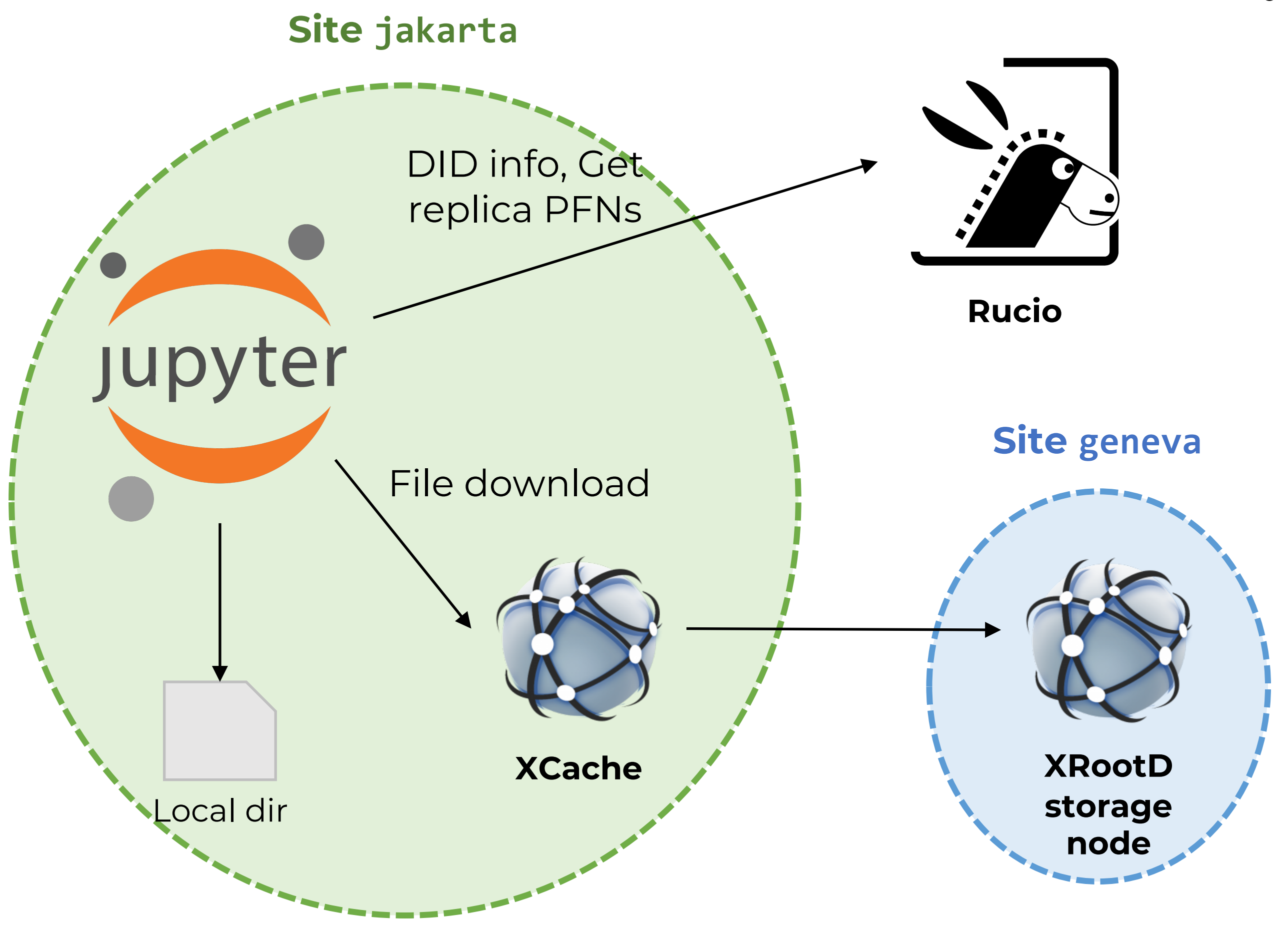
Download Mode

- Uses Rucio download client to make files available.
  - JupyterLab server downloads the file directly from the RSE to the user's directory
- No need for mounted RSE.
  - Self-contained within the JupyterLab installation.
- Suitable for simpler installations that don't have existing data transfer infrastructure.
- In Download mode, multiple users can download the same files.
  - Here, placing a caching layer would reduce network traffic.

Download Mode



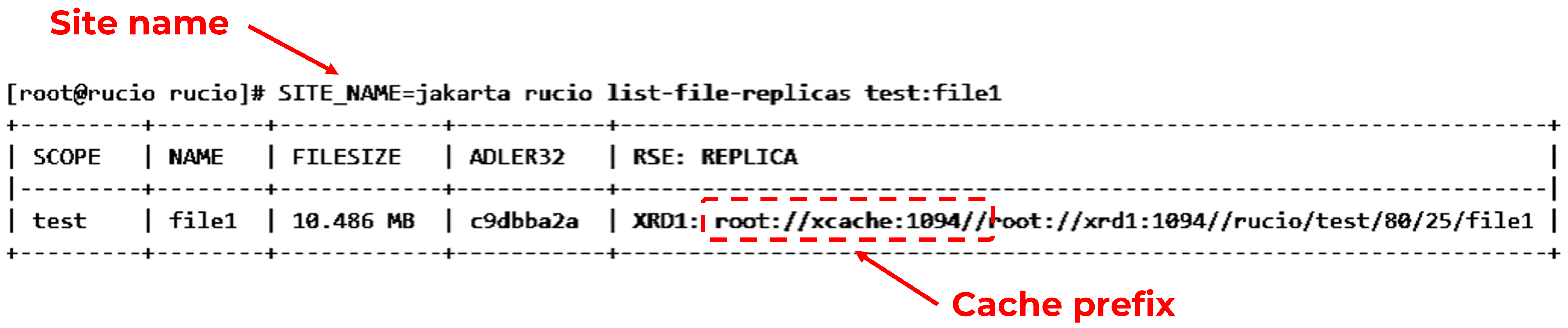
Using XCache in Download Mode



- Rucio has native support for XCache using `root-proxy-internal` config.
  - It will prepend a cache prefix (e.g. `root://xcache:1094//`) to the PFN if the client is on a different site.
  - The extension (utilizing Rucio DownloadClient) will retrieve files from that cache, when applicable.
  - Site admins need to specify a site name in the extension config.

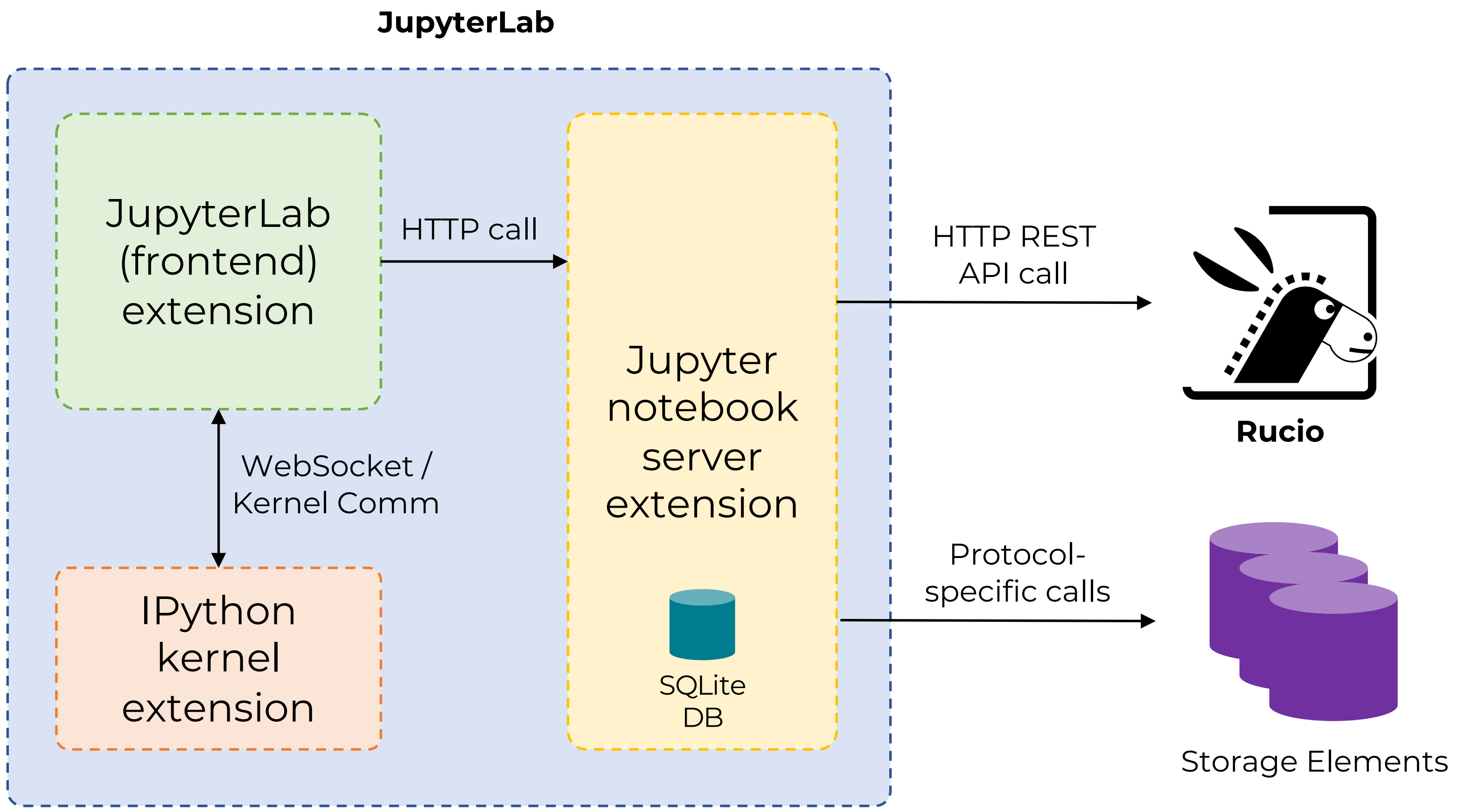
XCache Support in Rucio

1. Assign a site name to existing RSE
   - `rucio-admin rse set-attribute --rse XRD1 --key site --value geneva`
2. Register XCache host + port to Rucio
   - `rucio-admin config set --section root-proxy-internal --option jakarta --value xcache:1094`



```
[root@rucio rucio]# SITE_NAME=jakarta rucio list-file-replicas test:file1
+----------+--------+------------+-----------+----------------------------------------------------------------+
| SCOPE    | NAME   | FILESIZE   | ADLER32   | RSE: REPLICA                                                   |
|----------+--------+------------+-----------+----------------------------------------------------------------|
| test     | file1  | 10.486 MB  | c9dbba2a  | XRD1: root://xcache:1094//root://xrd1:1094//rucio/test/80/25/file1 |
+----------+--------+------------+-----------+----------------------------------------------------------------+
```

Architecture



- The server extension uses SQLite to store user config and cache Rucio responses.
- Frontend extension communicates with kernel extension to inject path string into Python variables.

Multi-VO Support

- The extension supports multi-VO deployments
- Site admins can configure the VO name
  - The extension will include the VO option when authenticating with Rucio.
  - When using X.509 User Certificate credential in Download mode, the extension can generate a Proxy certificate using `voms-proxy-init`.
    - The Proxy cert will be stored in a temporary directory and its path will be set as `X509_USER_PROXY` env variable when downloading the files.

Installing the Extension

- It comprises of several components:
  - Jupyter notebook server extension
  - JupyterLab (frontend) extension
  - IPython kernel extension
- The extension can be installed from Python Package Index (PyPI)
  - Simply do `pip install rucio-jupyterlab`
  - Complete installation instructions on https://github.com/didithilmy/rucio-jupyterlab
- Prebuild Docker image is also available for quick setup (see repo README)

Quick Docker installation

```
docker run -d -p 8888:8888 \
    --name jupyterlab \
    -e RUCIO_MODE=download \
    -e RUCIO_BASE_URL=https://escape-rucio.cern.ch:32300/ \
    -e RUCIO_AUTH_URL=https://escape-rucio.cern.ch:32301 \
    -e RUCIO_DISPLAY_NAME=ESCAPE \
    -e RUCIO_NAME=ESCAPE \
    -e RUCIO_CA_CERT=/certs/ca.pem \
    -e RUCIO_WILDCARD_ENABLED=1 \
    -v /etc/grid-security/:/etc/grid-security/ \
    -v /root/ca.pem:/certs/ca.pem:z,ro \
    -v /root/x509up:/home/jovyan/certs/x509up:z,ro \
    didithilmy/rucio-jupyterlab:latest
```

Configuring the Extension

- The extension uses standard Jupyter configuration system
  - Can be placed on `~/.jupyter/jupyter_notebook_config.json`
  - Refer to Jupyter docs
- Supports remote configuration
  - Config JSON placed in a static file, accessible via HTTP from the JupyterLab server
  - More on this later

Configuring the Extension (2)

- Configurable items:
  - Name & Display Name
  - Rucio URL & Auth URL
  - Wildcard search enable/disable
  - Destination RSE
  - Mount path prefix
  - Replication rule lifetime
  - Rucio CA path
  - VO, VOMS vomsdir, VOMS certdir, VOMS vomses, VOMS enabled
  - Site name
  - App ID
  - Mode of operation
- More on this: https://github.com/didithilmy/rucio-jupyterlab/blob/master/CONFIGURATION.md

Remote Configuration

- Rucio and JupyterLab instances can be managed by different site administrators
  - If a configuration item changes, it’s a challenge to coordinate both teams.
  - To address this, the extension supports reading a configuration remotely.

```
{
  "instances": [
    {
      "name": "experiment.cern.ch",
      "display_name": "Experiment",
      "$url": "https://url-to-rucio-configuration/config.json"
    }
  ]
}
```

Future Developments

- More Kernel compatibility
  - Octave, R, ROOT C++
- More authentication methods
  - OAuth/OpenID Connect
- Share notebooks across JupyterLab installations
  - Allows any JupyterLab instance to connect to publicly-accessible Rucio installations and their RSEs
  - Fetches Rucio configuration on-the-fly, URL known from notebook metadata

Conclusion

- This extension has the capability to bridge an exascale data management platform (Rucio) and an online data analytics platform (SWAN/JupyterLab).
  - It could become an important piece to enable easy data access from an analysis platform to ESCAPE datalake.
- The development of the extension has benefited from ESCAPE’s expertise and R&D.
- The development of this extension was aligned with the current challenges in distributed computing.
- The Rucio JupyterLab Extension can help to assess the new models and infrastructures being prototyped to address the future exabyte-scale and multi-experiment computing scenarios.

Acknowledgements

Huge thanks to my CERN mentors:

- Aristeidis FKIARAS
- Riccardo DI MARIA
- Martin BARISITS
- Diogo CASTRO
- Mario LASSNIG
- Enric TEJEDOR SAAVEDRA
- Enrico BOCCHI

Muhammad Aditya Hilmy
mhilmy@hey.com
didithilmy

Thank you.

Attributions:
CERN-HSF logo courtesy of hepsoftwarefoundation.org
Rucio logo courtesy of rucio.github.io
SWAN logo courtesy of swan.web.cern.ch
ESCAPE logo courtesy of projectescape.eu