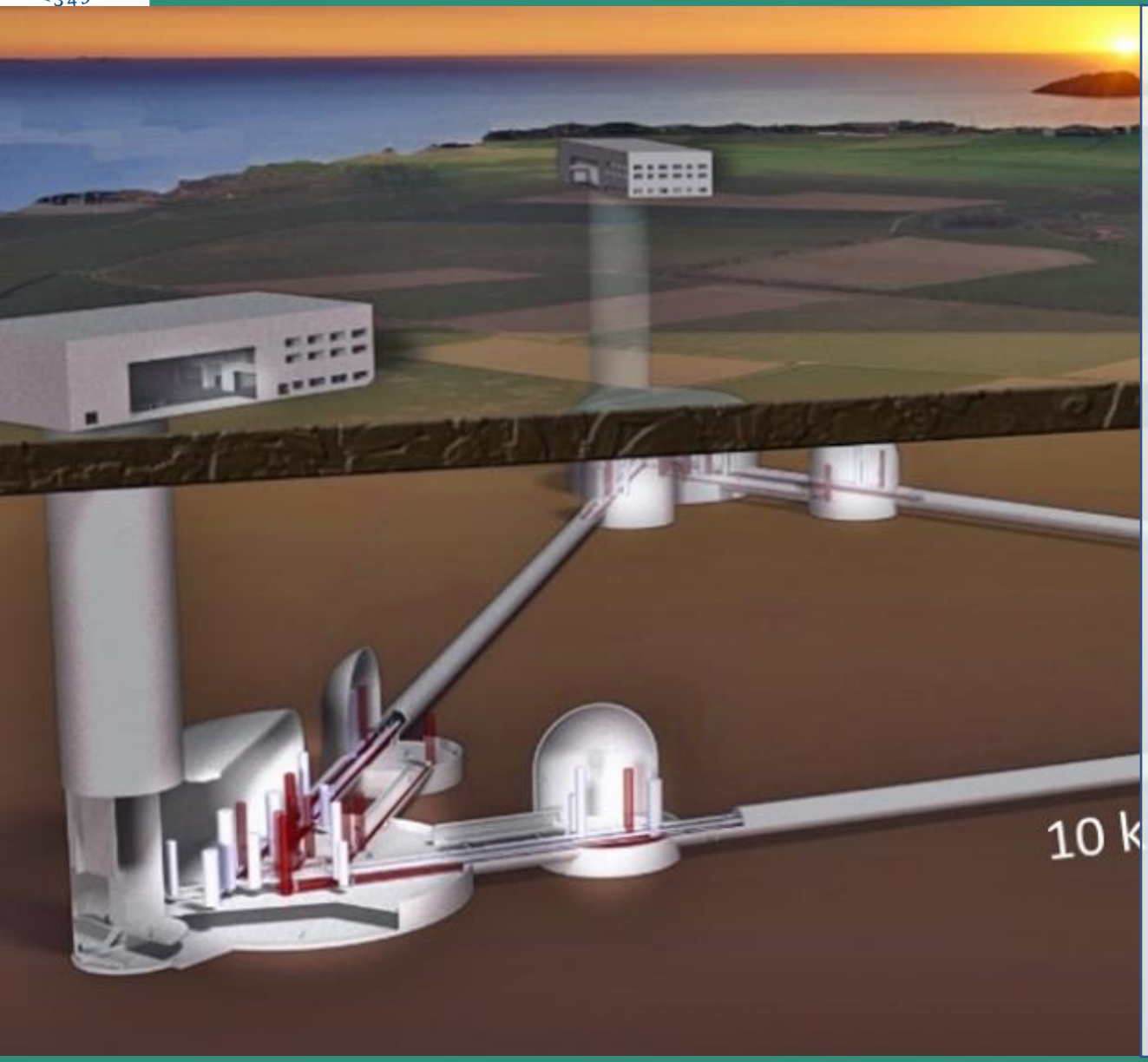


Repository and data format
update and proposals

Massimiliano Razzano

(in collaboration with many others)

University of Pisa & INFN-Pisa

SPB Meeting – 15 Jun 2022

ET Data Repository (etrepo)

- **Overview**
  - Hosted on Green Data Center @ University of Pisa
  - Online since 2019
  - Virtual machine, easy to customize
- **Specs**
  - 16 CPU cores Intel Xeon 5120 (28 thread/core)
  - 5 Tb hard disk
  - 32 Gb RAM
  - Centos7 OS
- **Recent Updates**
  - Increased CPUs, RAM
  - Backup

ETrepo Organization

- **Data Directories**
  - Temp data-sandbox for manual transfer
  - Data-sites (1 Tb so far)
  - Periodic transfer to data-sites
- **Software directories**
  - General software directory (e.g. miniconda)
  - Et-software (e.g. shared jupyter notebooks)
  - Probably just one dir is enough
- **Users workspaces**
  - Linked from each home user directory
  - Use these for your work, not your /home/user directory
  - So far 340 Gb

Accessing ETrepo

- **Automatic accounting system**
  - Fill the form at https://forms.gle/n2MpK1cg2Mxfdz1o8 (sent around by email some time ago, will send again if needed)
  - Scripts will take your requests, make an account for you, set up directories and send an email to you with username and temp pwd
  - Usernames as name+surname → nsurname
  - Latency half hour, can be longer in some cases

Github vs Gitlab?

- Initially created Github repo, now we have the ET GitLab
- Should we drop Github account to et-sw and leave GitLab?

Documentation at https://tinyurl.com/y4ukh98d

Jupyter Notebooks

- **Jupyterhub infrastructure**
  - Works smoothly most of the time
  - SSL security
  - Access to data and software
  - Linked to local accounts

- **Issues and upgrades**
  - Some occasional glitches (easily fixed)
  - Resources limit (upgraded resources, now should be ok)
  - Issues with library compatibility (e.g. recent version of obspy)

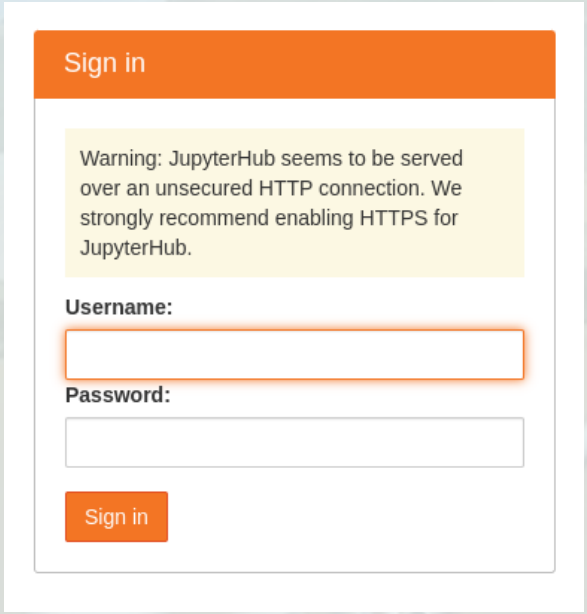
JupyterLab V2

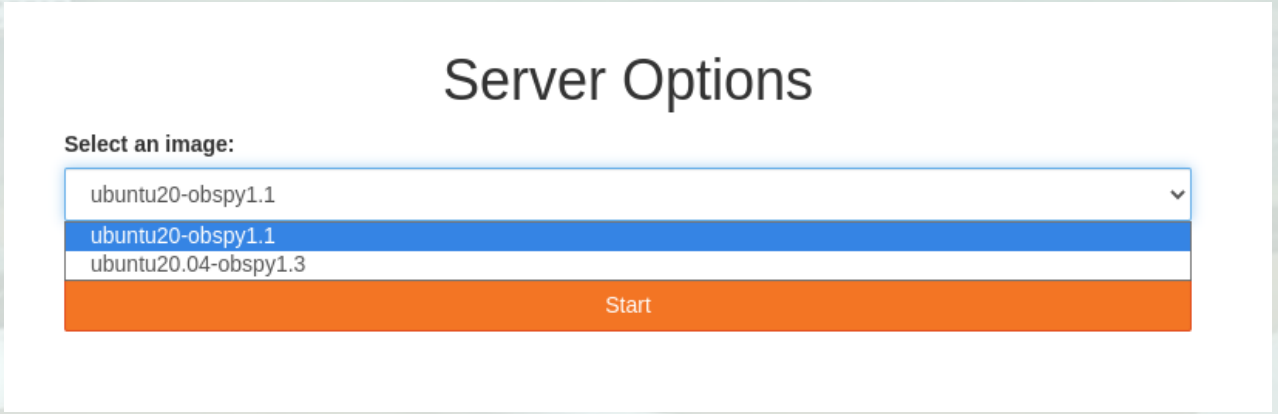
- **Upgrade & improvements on the JupyterLab**
  - Move to Docker containers instead of local system installation
  - Decoupled from OS, possible to install new libraries
  - Possibility to host different configurations, libraries
  - User experience almost the same, minor changes
  - Access using local accounts

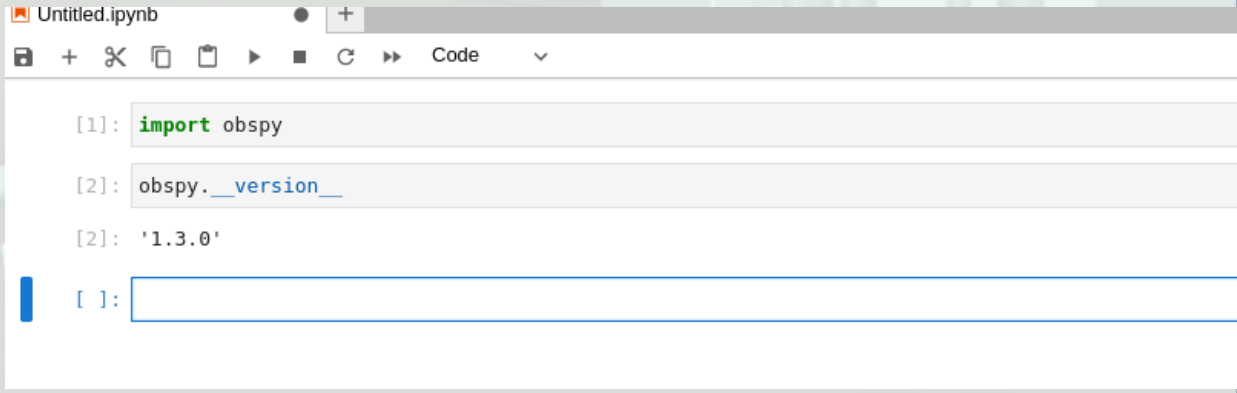
- **Status update**
  - Prototype infrastructure online (under unipi VPN), in collaboration with A. Fiori
  - 2 images, one with early obspy (as in the current implementation) and one with obspy1.3.0
  - Images managed under CI
  - Access to etrepo data as before

The new Jupyter on ETrepo

Same login as before







Last Tests and ready in ~ few days

Data organization & data format

- **Data Organizazion**
  - Data flows to etrepo from different sites, sensors
  - Provide an easy way to access by users
  - Data will keep flowing, data management plan needed

- **Data Format**
  - Data collected by different sources
  - Different data format depending on instruments
  - Accomodate new, different, sensors and source format
  - Easy to access and manipulate
  - Experience with auxiliary channels in Virgo/LIGO

Data format: proposal

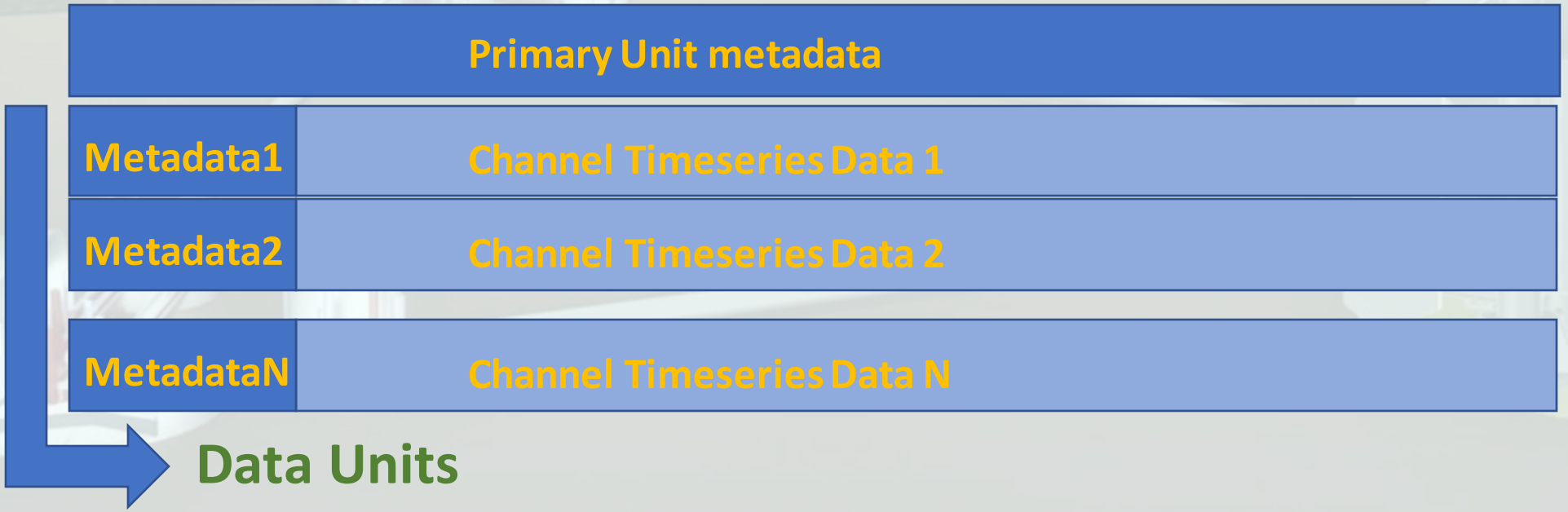
- **Requirements**
  - Manage data from different sources/instruments/format, both existing and future
  - Act intermediate layer for existing formats (e.g. miniseed)
  - Collect and manage metadata from different instruments
  - Hierarchical,multichannel structure, similar to existing aux channels in GW detectors
  - Easy and fast I/O

- **Hierarchical Multichannel Data Format**
  - Concept as data formats like FITS, GWframes, mseed,etc...
  - Data streams in Data Units, containing metadata+channel data
  - Possibility to group channels (e.g. same sensor) and add Data Quality flags
  - Rely on HDF format as container easy to read in Python and other languages
  - HDF files also used in Adaptable Seismic Data Format (ASDF) for seismic data
  - Needed a full definition and a package to manipulate this format

Data format: proposal

Parse from Instruments

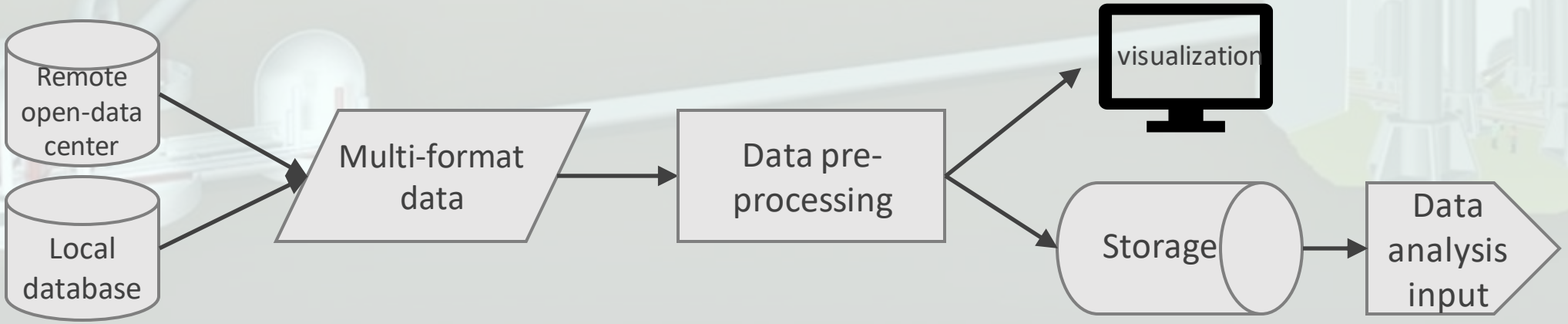
Read/Analysis



Data Manager package prototype

A first prototype has been developed:

- Based on existing library developed @unipi (M.Razzano,F. Di Renzo, N. Sorrentino,et al)
- Open-source, compatible with the main data analysis packages **GWpy**, **Obspy** and **Pandas**
- Multi-format/channel I/O: compatible with the standards in GW community (gwf) and geophysics (mseed, csv), and open data center (GWOSC)
- Data analysis ready: perform standard visualization and pre-processing operations
- Hdf5 based: datasets in a hierarchical structure for storage and retrieval
- Parse and manage from various sources into this format



Sample usage: I/O

Import **DataManager** class

Read data from multiple formats and locations

Names etc needs to be defined

Print the hierarchical structures of the imported datasets

and the metadata

```
[1]: from os import path
     import sys
     sys.path.append("../bin")

     from etdama import DataManager  # Main class for data managment

     # Path to data files
     gwf_path  = "../data/H-H1_GWOSC_4KHZ_R1-1126259447-32.gwf"
     txt_path  = "../data/H-H1_GWOSC_4KHZ_R1-1126259447-32.txt"
     mseed_path = path.join("/home/fdirenzo","data-sites/sosenattos/SOE0/2020/01/01/XX.SOE00.HHE_20200101_000000.seed")
     csv_path  = path.join("/home/fdirenzo", "data-sites/sosenattos/WEATHER/2020/2020-06.csv")

     dm = DataManager("Data")   # Create class instance

     dm.read(gwf_path, "H1:GWOSC-4KHZ_R1_STRAIN", key="grav/gwf")      # Read GW data from local GWF file
     dm.read(txt_path, comment="#", names=["Strain"], key="grav/txt")  #  -   -   -    -    -    txt  -
     dm.read(source="gwosc", ifo="V1",                                 # Read open GW data from remote
             start="2020-01-01",end="2020-01-01 0:15", key="grav/gwosc")

     dm.read(mseed_path, key="sosenattos")                             # Read mseed data
     dm.read(csv_path, key="weather", timecol="dateTime")              # Read csv file

     print(dm)   # Print the  structure and the attributes
```

```
Data:
  ├── grav
  │   ├── gwf
  │   ├── gwosc
  │   └── txt
  │       └── Strain
  ├── sosenattos
  │   └── HHE
  └── weather
      ├── barometer
      ├── dewpoint
      ├── outHumidity
      ├── outTemp
      ├── rain
      ├── rainRate
      ├── windDir
      ├── windGust
      ├── windGustDir
      └── windSpeed

 Attributes:
        name : Data
  time_stamp : 22-06-14_10h53m04s
```

Sample usage: plotting etc

- Show metadata of channels
- Access data/times as timeseries
- Quick visualization are also present (plot of multiple datasets, histograms, etc.)

```
dm["sosenattos/HHE"].attrs['starttime'] = unix2gps(dm["sosenattos/HHE"].attrs['starttime'])
```

```
[5]: dm["sosenattos/HHE"].data
```

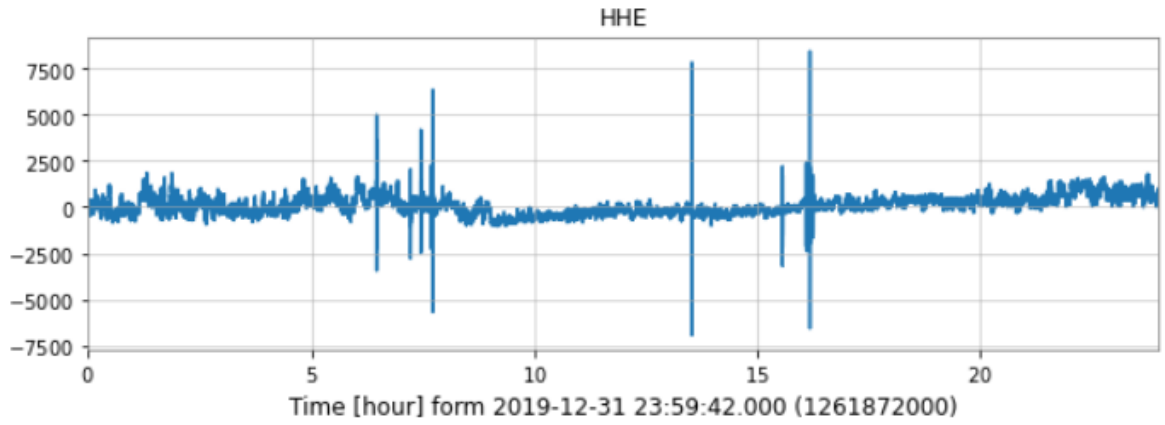
```
[5]: array([121,  62, 125, ..., 883, 886, 893], dtype=int32)
```

```
[6]: dm["sosenattos/HHE"].times
```

[6]: $[1.261872 \times 10^9,\ 1.261872 \times 10^9,\ 1.261872 \times 10^9,\ \ldots,\ 1.2619584 \times 10^9,\ 1.2619584 \times 10^9,\ 1.2619584 \times 10^9]$ s

```
[7]: dm["sosenattos/HHE"].plot()
```

[7]:



Conclusions

- **Repository**
  - ETrepo updated
  - New containerized implementation of JupyterLab (more flexibility)
  - Last tests, almost ready for deployment (will send an email when upgrading)
- **Data format**
  - Proposal for multichannel, hiearchical data format
  - Interface with various instruments/data sources
  - Efficient storage for data access/plot/analysis
  - Prototype for data access package, deploy on ETrepo for test
  - Write a data format description document, then work on the implementation
- **Data organization**
  - Parse collected data to and archive in the new multichannel format
  - Pipeline for automatic conversion based on data package
  - Data collected, send to ETrepo and added to the archive
  - Deploy on ETrepo/sites/wherever needed