Finanziato dall'Unione europea
NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA

ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

INFN

ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Platforms for High Rate Analysis

Tommaso Tedeschi on behalf of WP5

Workshop on "Quasi-Interactive Analysis of Big Data with High Throughput" - 8/10 Jan 2024 - Bologna

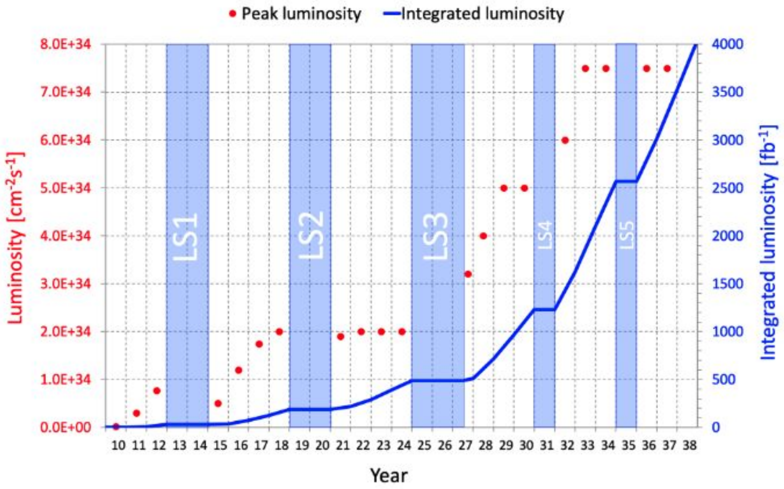
A recap - the context

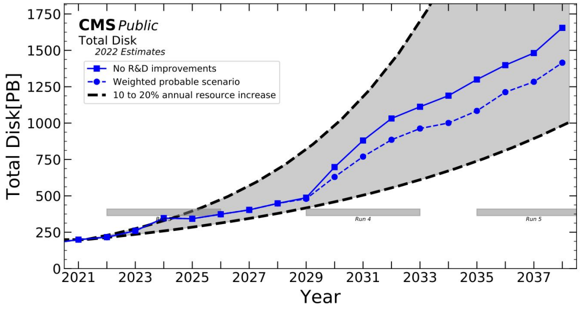
From 2029/30 onwards: huge increase in HEP experiments computing resources requests (HL-LHC above all)

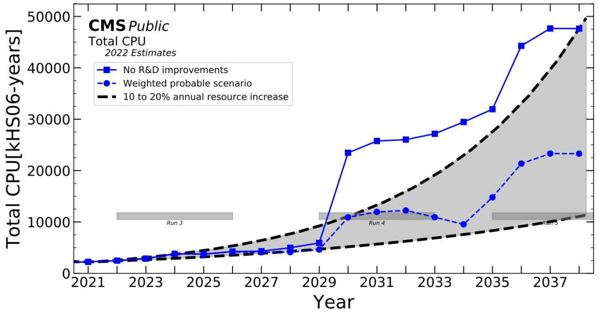
New analysis paradigm is arising: **high-rate, declarative, interactive or quasi-interactive data analysis approach**

The development of infrastructural solutions (high rate platform) to implement such a new model is done **inside WP2 and WP5:**

- use case-driven approach and tests with real-world analyses
- **and synergically with Spoke0**:
  - adopting/proposing infrastructural solutions







High rate analysis Tools

Main ingredients for high rate analysis:

- cutting edge tools:
  - **ROOT's RDataFrame (RDF)** - modern, high-level interface for analysis of data stored in TTree , CSV and other data formats, in C++ or Python
  - **Scikit-HEP echosystem** (uproot + awkward-array + vector + coffea + ...) - array-based syntax for manipulating HEP event data in an efficient and numpythonic way
  - MORE ON THIS IN THE NEXT CONTRIBUTIONS….
- reduced data formats
- Scaling out with **Dask**!

A typical high rate platform

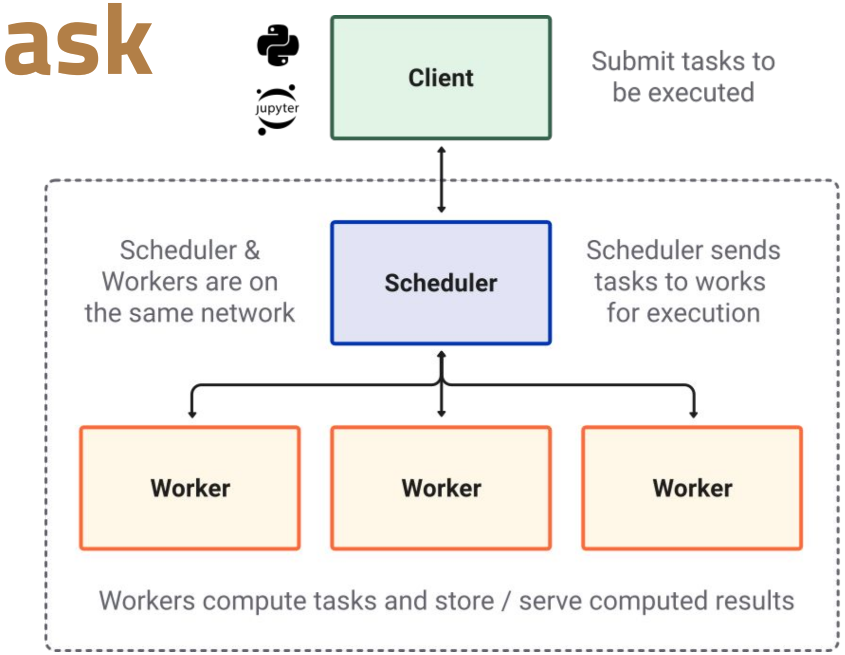
INFN

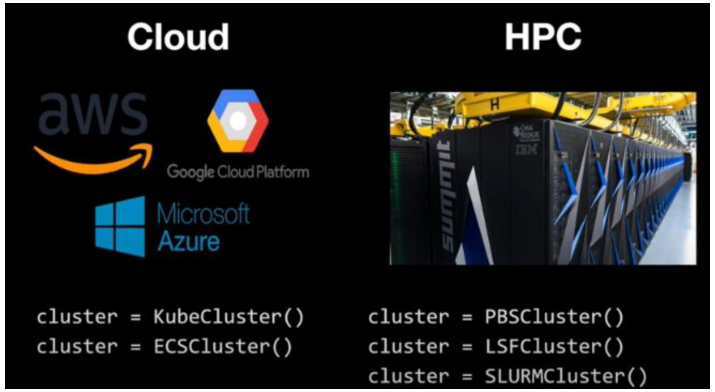
The computing infrastructures should be:

- **agnostic** with respect to the analysis software which can be, in principle, experiment-specific
- based on **containerized** cluster solutions integrated with storage systems and caches:
  - containers are executable software packages that contain everything that is needed to run an application, from code to system libraries, therefore abstracting it from the host infrastructure.
- **multi-user**
- capable of handling **different back-ends, most importantly Dask!**

Key technology enabler: Distributed Dask

- **Dask.distributed** is a centrally managed, distributed, dynamic task scheduler:
  - The central dask scheduler process coordinates
    - actions of several Dask worker processes on multiple machines
    - the concurrent requests of several clients
- Users interact by connecting a local Python session to the scheduler and submitting work
- Best to use a **cluster manager** utility class:
  - It deploys a scheduler and the necessary workers as determined by communicating with the resource manager
  - **KubeCluster** is a cluster manager for Kubernetes
  - **Dask-jobqueue** is a set of cluster managers for job queueing systems
    - Supports PBS, Slurm, LSF, HTCondor, …


Client
Submit tasks to be executed
Scheduler & Workers are on the same network
Scheduler
Scheduler sends tasks to works for execution
Worker
Worker
Worker
Workers compute tasks and store / serve computed results



```
cluster = KubeCluster()
cluster = ECSCluster()
```

```
cluster = PBSCluster()
cluster = LSFCluster()
cluster = SLURMCluster()
```

…

This obviously fits very well with the python analysis ecosystem, but also ROOT's RDataFrame can use Dask as backend: Dask therefore enables analysis on very different resource providers.. provided you get access to your data!

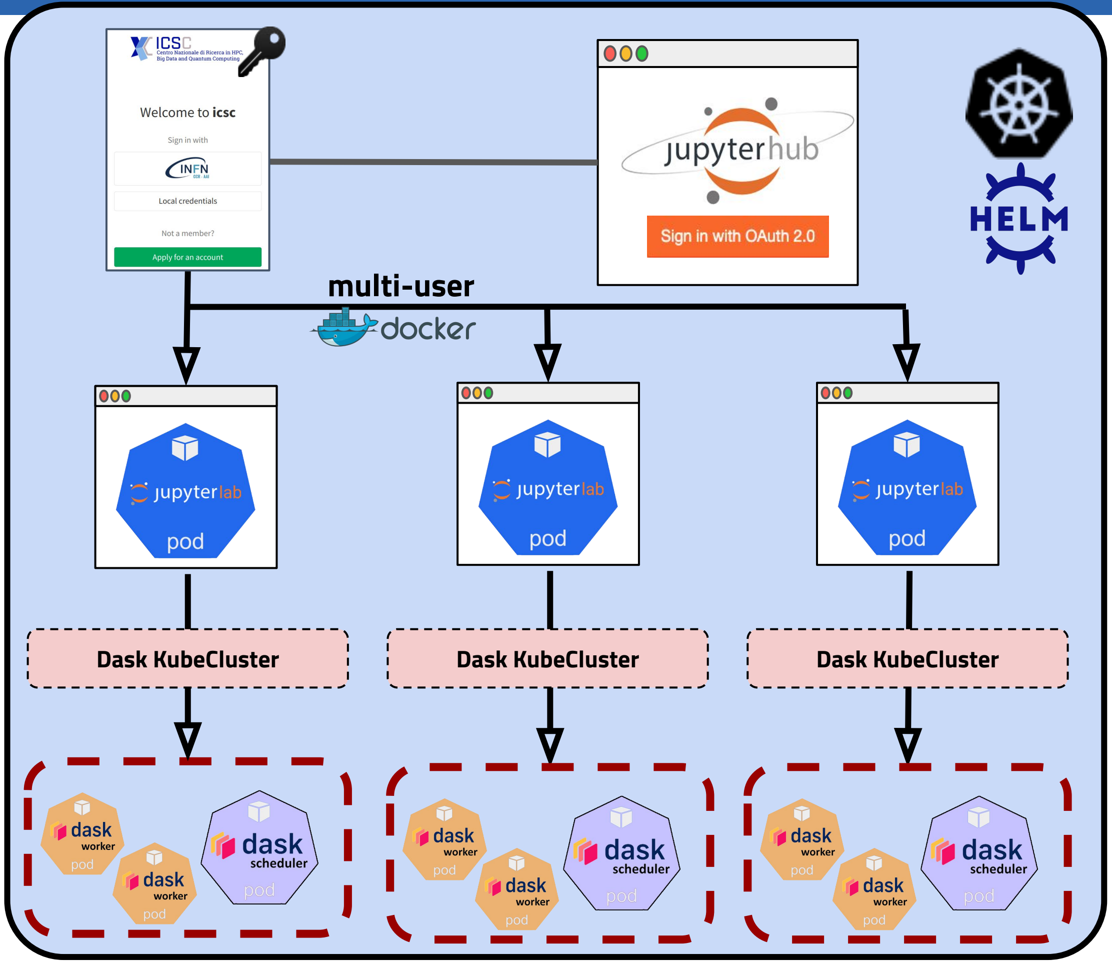
The ICSC Spoke2 high rate platform

As Spoke2 WP5 we are ready, using ICSC resources as per RAC allocation

A high rate platform deployed on a Kubernetes cluster (128 vCPUs and 258 GB)

- endpoint is here

Let's see its components in details



Deployment

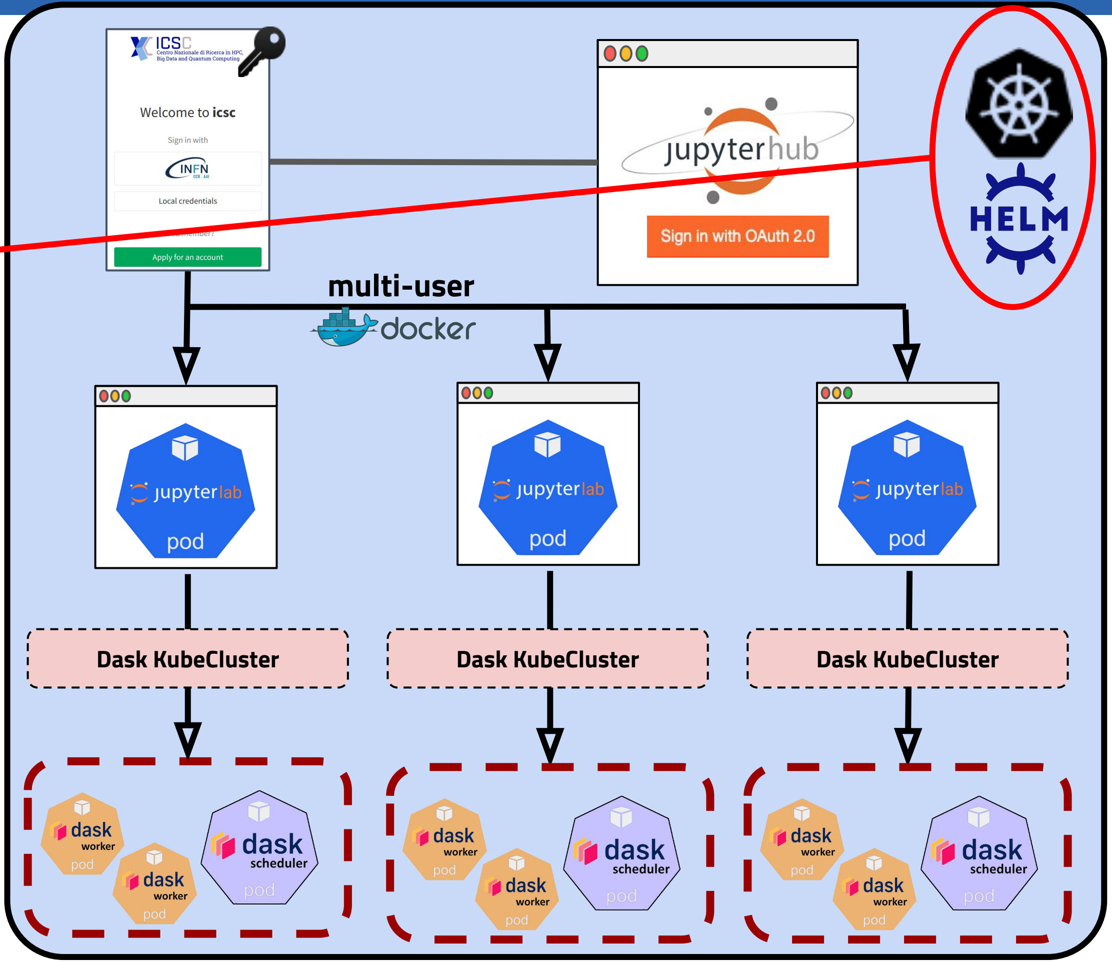
Kubernetes (K8s):

- The industrial standard for the management and orchestration of the deployment of containerized services on a set of machines
- Kubernetes cluster: set of nodes that run containerized applications (organized in Pods), controlled by a control plane

HELM:

- The "package manager" for K8s
- The deployment of the K8s resources needed for the spawning of this platform, is handled via HELM charts available in the GitHub organization https://github.com/ICSC-Spoke2-repo/HighRateAnalysis-WP5.

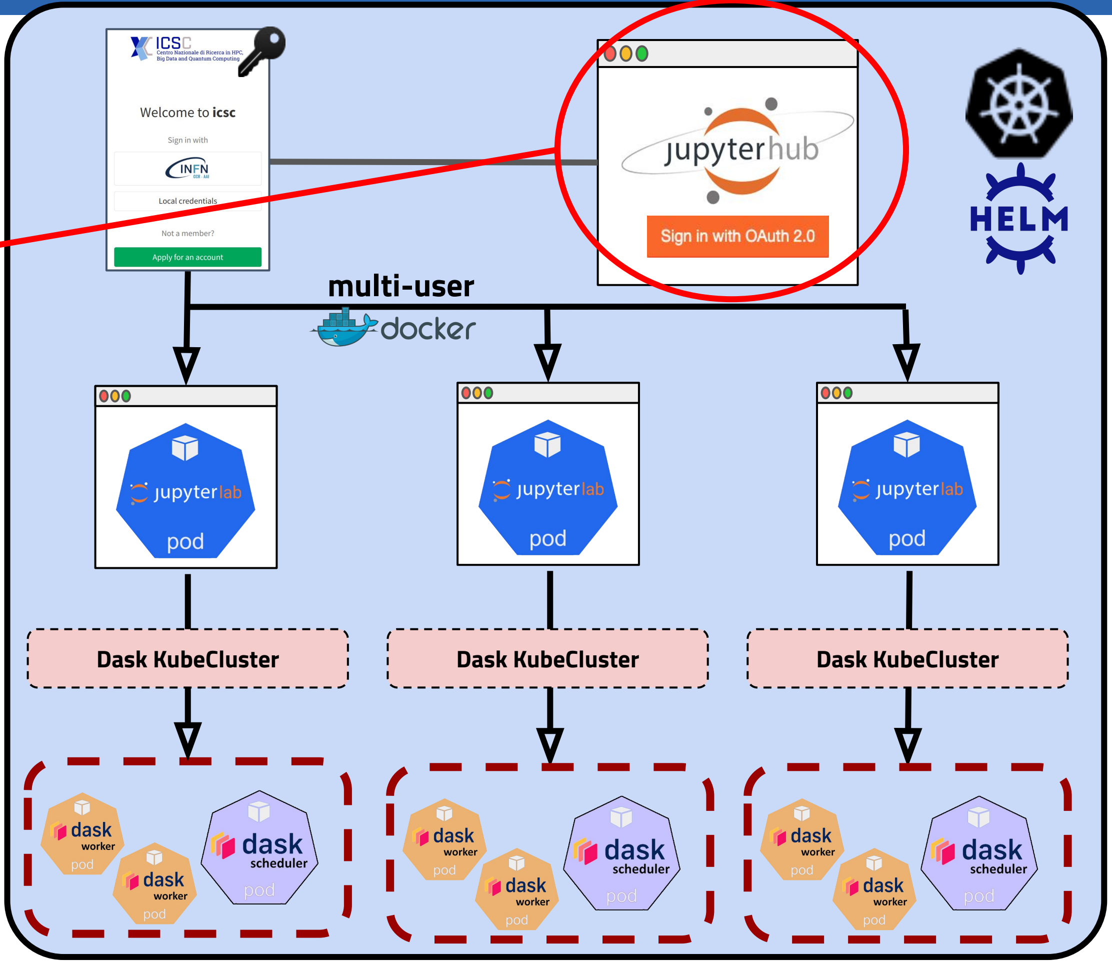
This allows a seamless, flexible, scalable and fault-tolerant deployment on the available resources, with a limited impact on the admin's work time.



Access

Connecting to an entrypoint URL, the user reaches a **Jupyterhub** instance

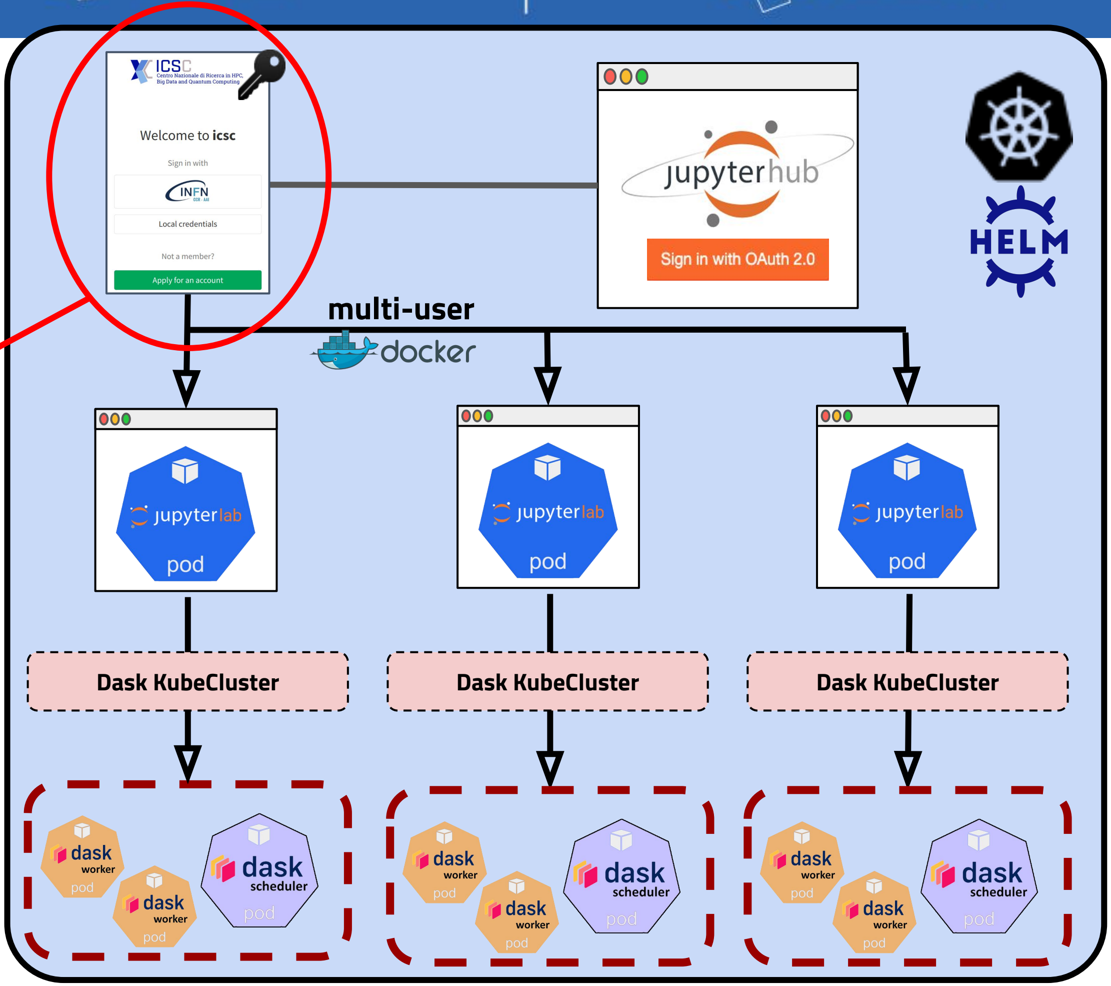
It is a multi-user Hub that spawns, manages, and proxies **multiple instances of the single-user Jupyter notebook server**



Security

Authentication and authorization via INDIGO-IAM
https://iam-icsc.cloud.infn.it/login

- An Open Source Identity and Access management solution for scientific computing



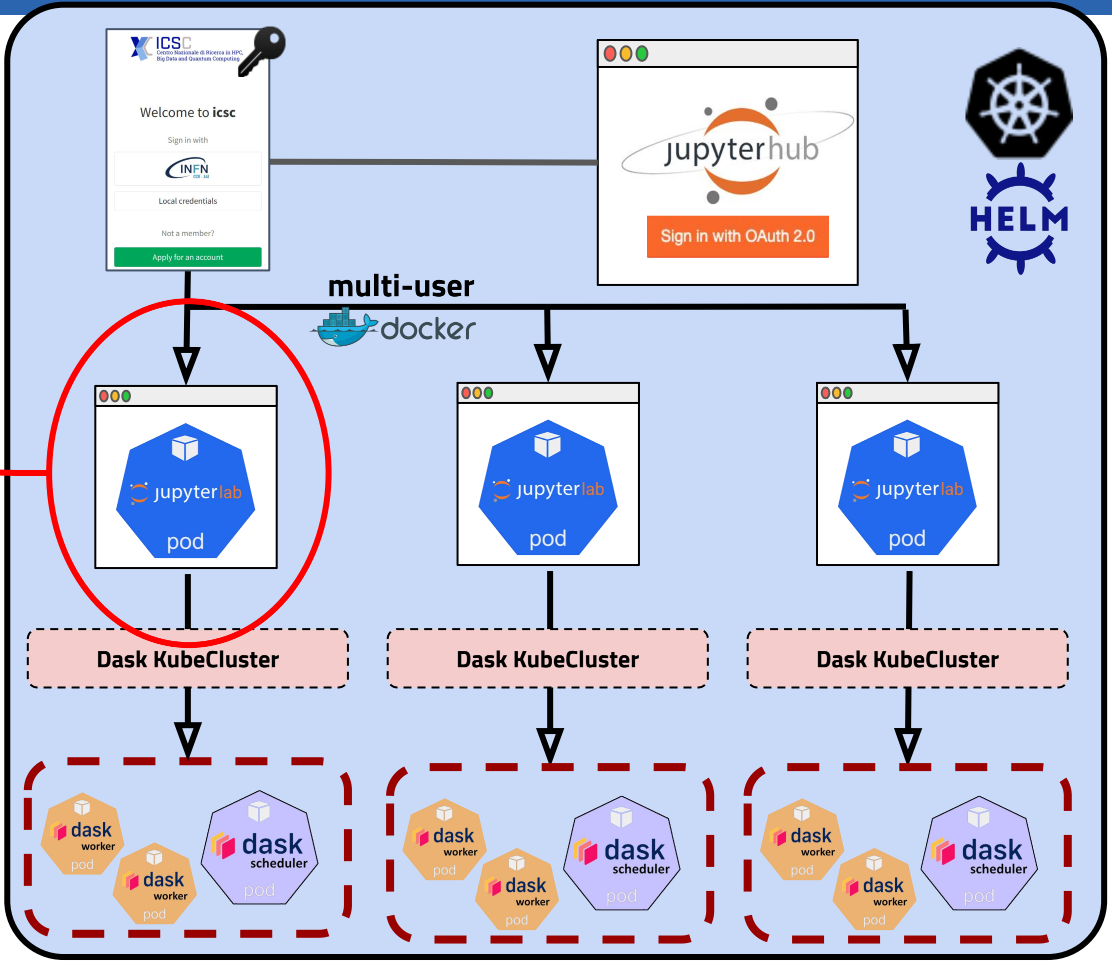
User interface

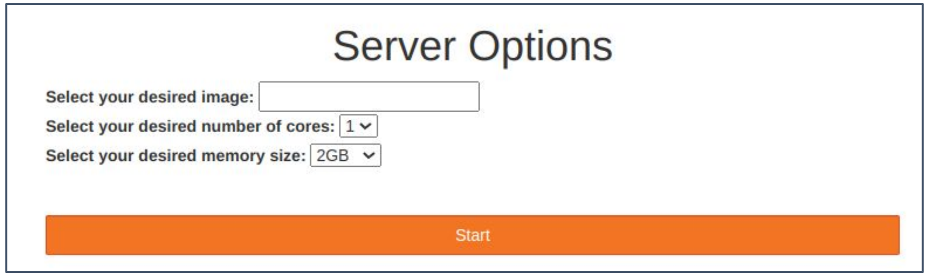
The user interface is based on **Jupyterlab** customised with specific plugins for specific purposes (e.g. Dask).

The working environment is highly customizable:

- using tailored Docker containers
- this is important when analyses require specific software (collaboration-wise)

Jupyterlab images Dockerfiles for Spoke2 are hoster here https://github.com/ICSC-Spoke2-repo/wp5-custom-images/tree/highrate





Server Options

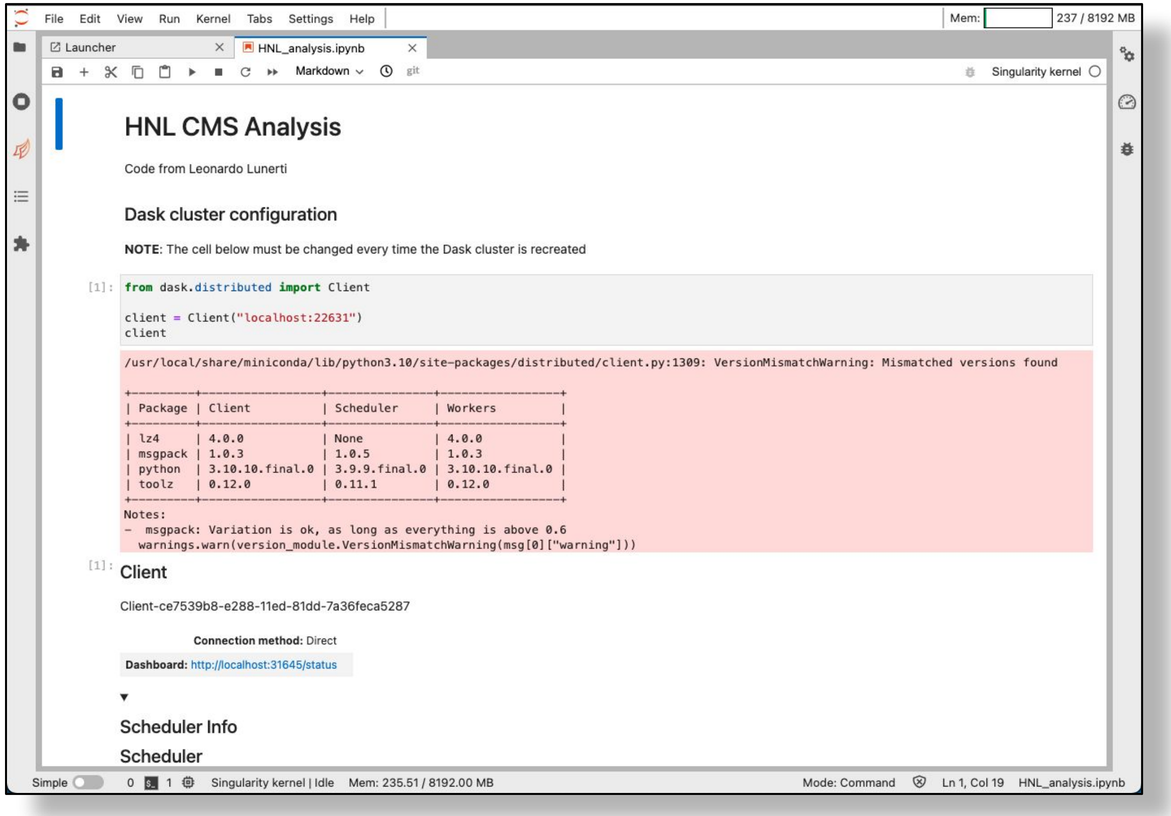
Select your desired image:

Select your desired number of cores: 1

Select your desired memory size: 2GB

User interface

The user gets access to a full IDE!



HNL CMS Analysis

Code from Leonardo Lunerti

Dask cluster configuration

NOTE: The cell below must be changed every time the Dask cluster is recreated

```
[1]: from dask.distributed import Client

     client = Client("localhost:22631")
     client
```

```
/usr/local/share/miniconda/lib/python3.10/site-packages/distributed/client.py:1309: VersionMismatchWarning: Mismatched versions found

+---------+-----------------+---------------+-----------------+
| Package | Client          | Scheduler     | Workers         |
+---------+-----------------+---------------+-----------------+
| lz4     | 4.0.0           | None          | 4.0.0           |
| msgpack | 1.0.3           | 1.0.5         | 1.0.3           |
| python  | 3.10.10.final.0 | 3.9.9.final.0 | 3.10.10.final.0 |
| toolz   | 0.12.0          | 0.11.1        | 0.12.0          |
+---------+-----------------+---------------+-----------------+
Notes:
-  msgpack: Variation is ok, as long as everything is above 0.6
  warnings.warn(version_module.VersionMismatchWarning(msg[0]["warning"]))
```

[1]: **Client**

Client-ce7539b8-e288-11ed-81dd-7a36feca5287

Connection method: Direct

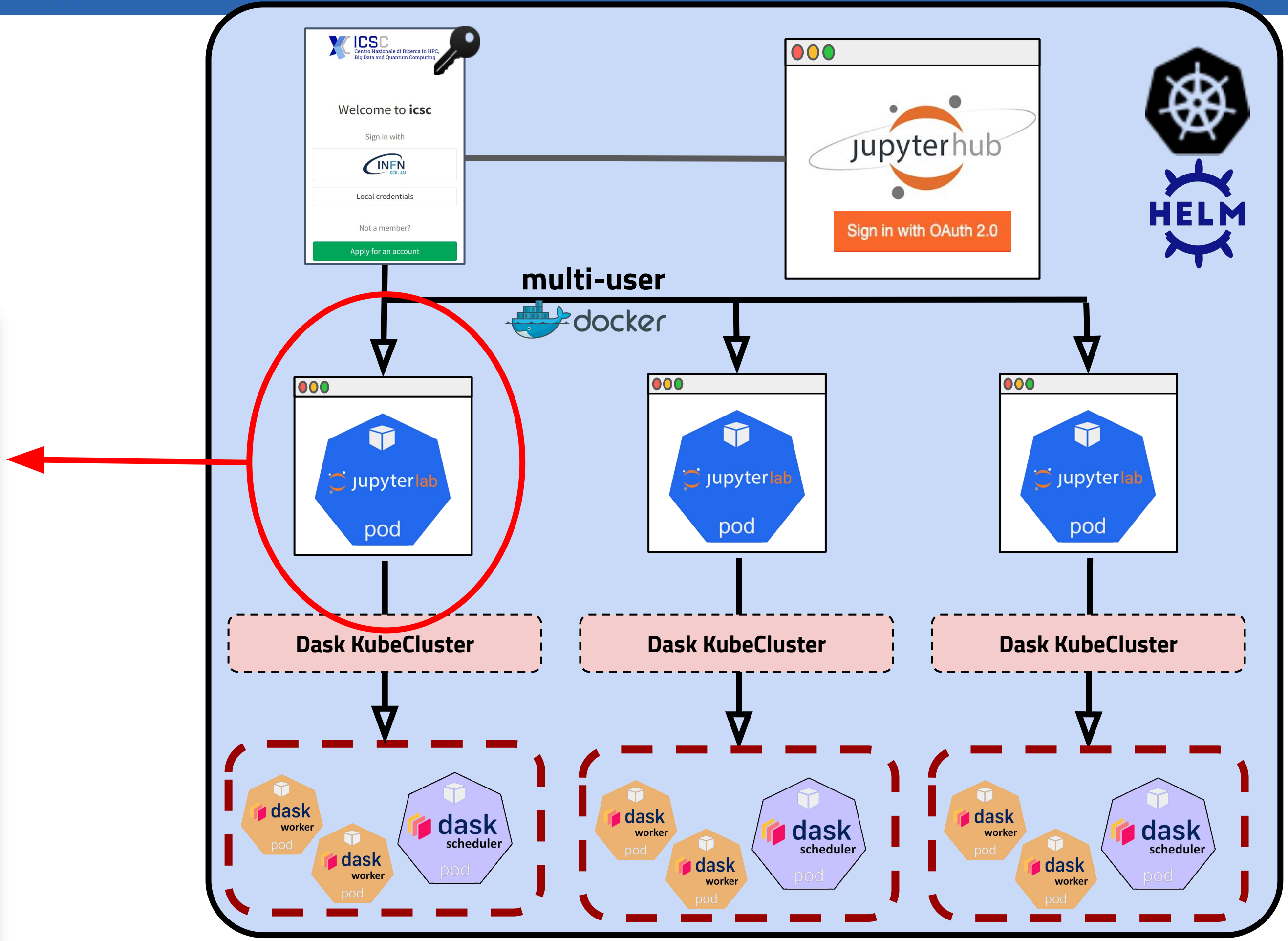
Dashboard: http://localhost:31645/status

▼

Scheduler Info

Scheduler

Simple 0 1 Singularity kernel | Idle Mem: 235.51 / 8192.00 MB Mode: Command Ln 1, Col 19 HNL_analysis.ipynb



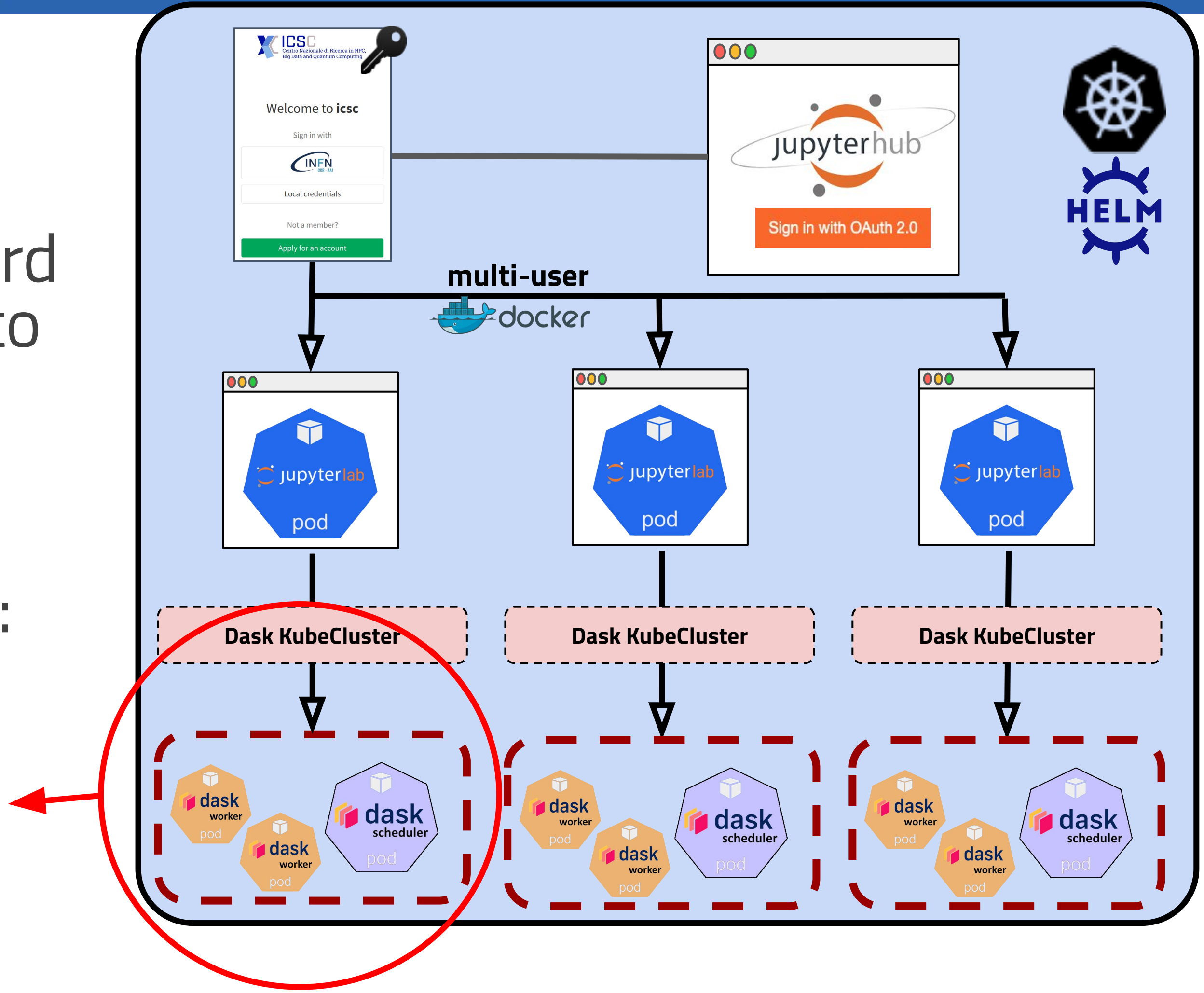
Distributing the workload

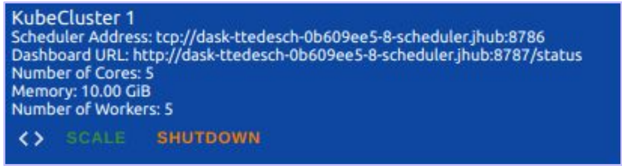
Dask Labextension plugin:

- allows to interact with the Dask dashboard directly in the Jupyterlab session access to useful monitoring panels
- deploys a Dask cluster using the KubeCluster cluster manager

The deployment of such cluster can be done: also via CLI/notebook cell

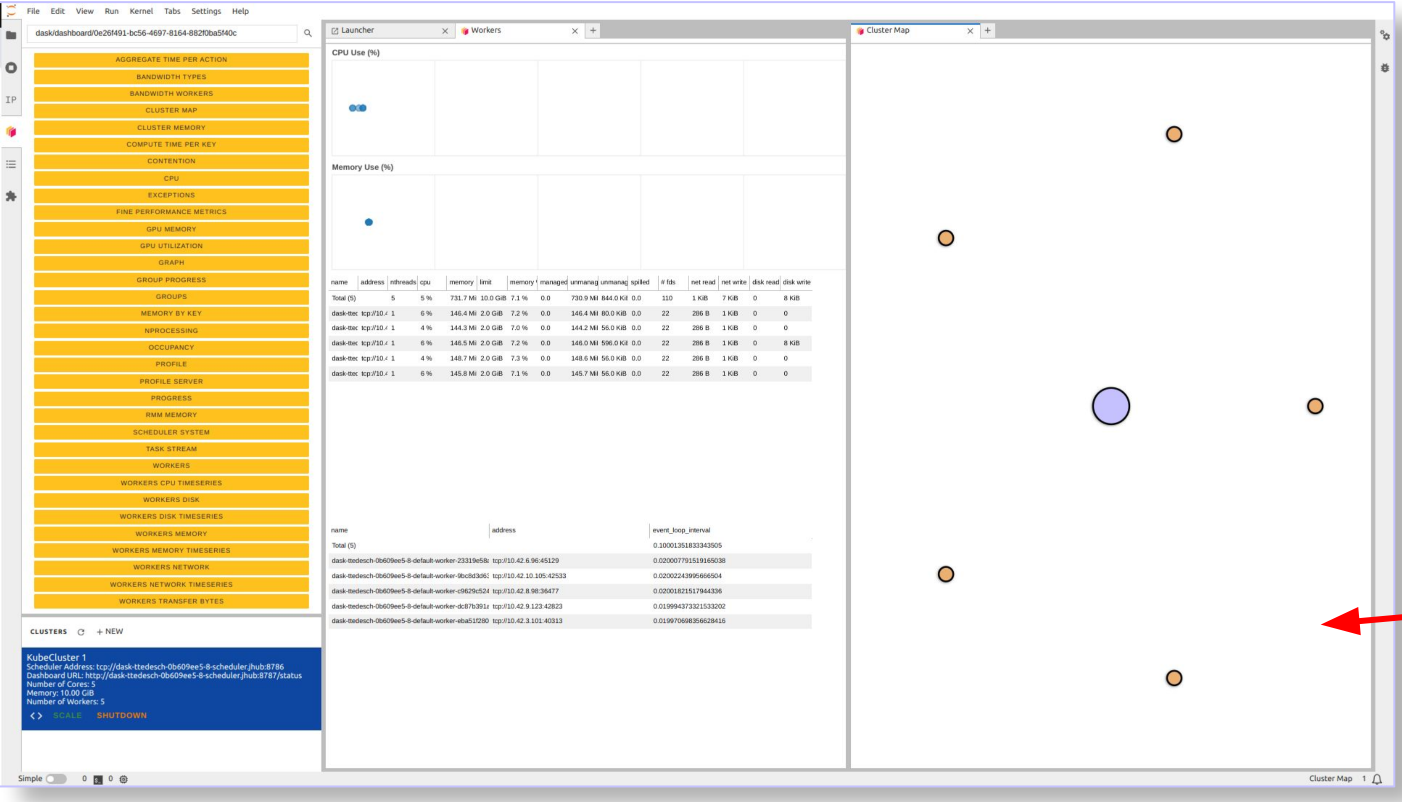
You can then handle your cluster from the plugin





KubeCluster 1
Scheduler Address: tcp://dask-ttedesch-0b609ee5-8-scheduler.jhub:8786
Dashboard URL: http://dask-ttedesch-0b609ee5-8-scheduler.jhub:8787/status
Number of Cores: 5
Memory: 10.00 GiB
Number of Workers: 5

Distributing the workload

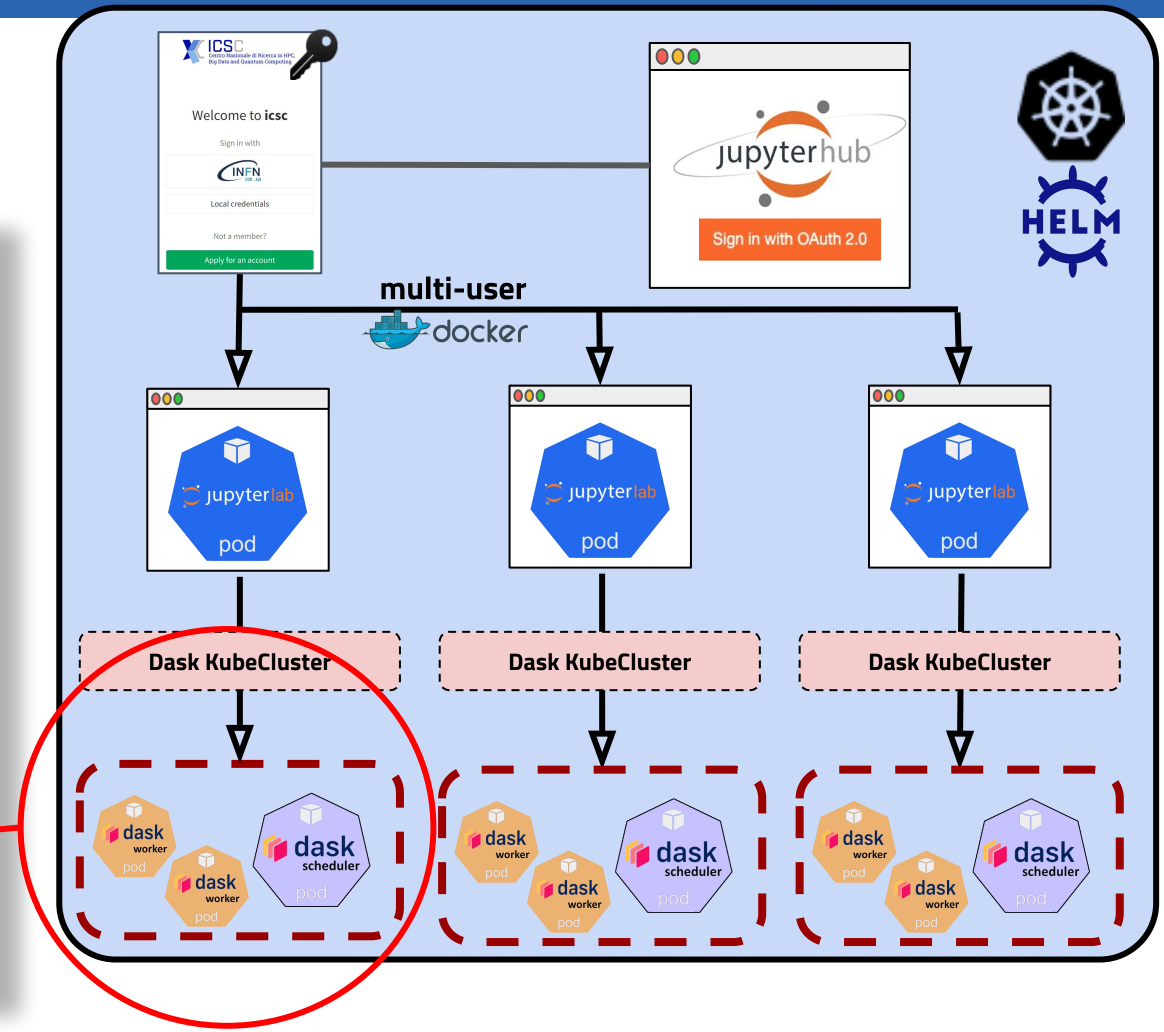


dask/dashboard/0e26f491-bc56-4697-8164-882f0ba5f40c

name	address	nthreads	cpu	memory	limit	memory '	managed	unmanag	unmanag	spilled	# fds	net read	net write	disk read	disk write
Total (5)		5	5 %	731.7 Mi	10.0 GiB	7.1 %	0.0	730.9 Mil	844.0 Kil	0.0	110	1 KiB	7 KiB	0	8 KiB
dask-ttec	tcp://10.4	1	6 %	146.4 Mi	2.0 GiB	7.2 %	0.0	146.4 Mil	80.0 KiB	0.0	22	286 B	1 KiB	0	0
dask-ttec	tcp://10.4	1	4 %	144.3 Mi	2.0 GiB	7.0 %	0.0	144.2 Mil	56.0 KiB	0.0	22	286 B	1 KiB	0	0
dask-ttec	tcp://10.4	1	6 %	146.5 Mi	2.0 GiB	7.2 %	0.0	146.0 Mil	596.0 Kil	0.0	22	286 B	1 KiB	0	8 KiB
dask-ttec	tcp://10.4	1	4 %	148.7 Mi	2.0 GiB	7.3 %	0.0	148.6 Mil	56.0 KiB	0.0	22	286 B	1 KiB	0	0
dask-ttec	tcp://10.4	1	6 %	145.8 Mi	2.0 GiB	7.1 %	0.0	145.7 Mil	56.0 KiB	0.0	22	286 B	1 KiB	0	0

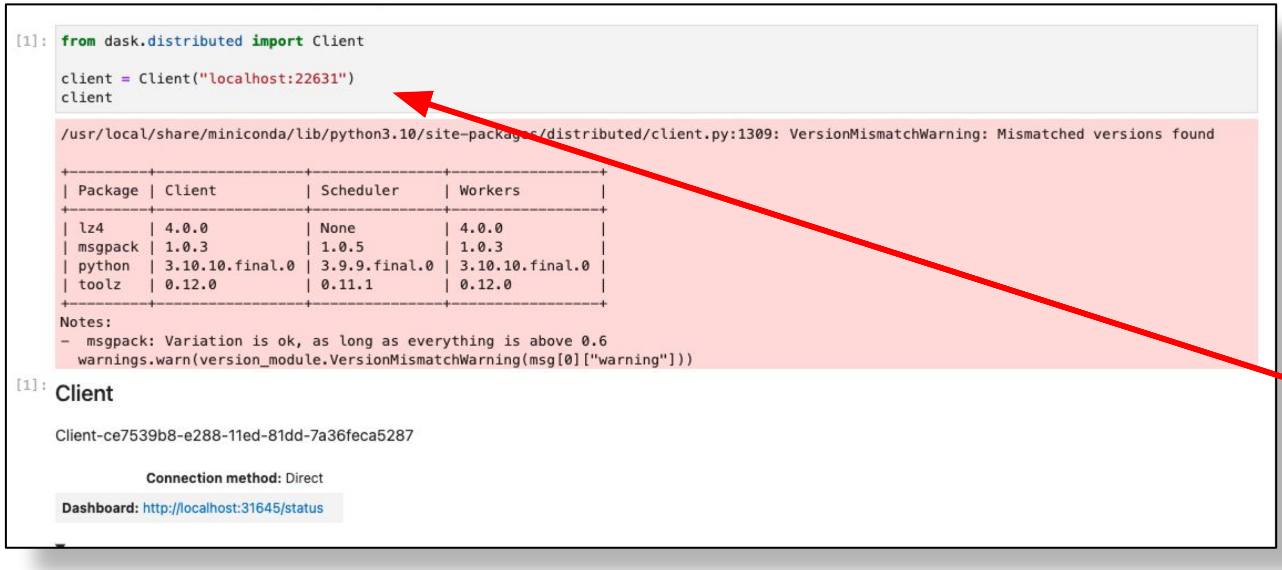
name	address	event_loop_interval
Total (5)		0.10001351833343505
dask-ttedesch-0b609ee5-8-default-worker-23319e58a	tcp://10.42.6.96:45129	0.020007791519165038
dask-ttedesch-0b609ee5-8-default-worker-9bc8d3d63	tcp://10.42.10.105:42533	0.02002243995666504
dask-ttedesch-0b609ee5-8-default-worker-c9629c524	tcp://10.42.8.98:36477	0.02001821517944336
dask-ttedesch-0b609ee5-8-default-worker-dc87b391a	tcp://10.42.9.123:42823	0.019994373321533202
dask-ttedesch-0b609ee5-8-default-worker-eba51f280	tcp://10.42.3.101:40313	0.019970698356628416

KubeCluster 1
Scheduler Address: tcp://dask-ttedesch-0b609ee5-8-scheduler.jhub:8786
Dashboard URL: http://dask-ttedesch-0b609ee5-8-scheduler.jhub:8787/status
Number of Cores: 5
Memory: 10.00 GiB
Number of Workers: 5



User interface

Connect the client and scale up your computations!



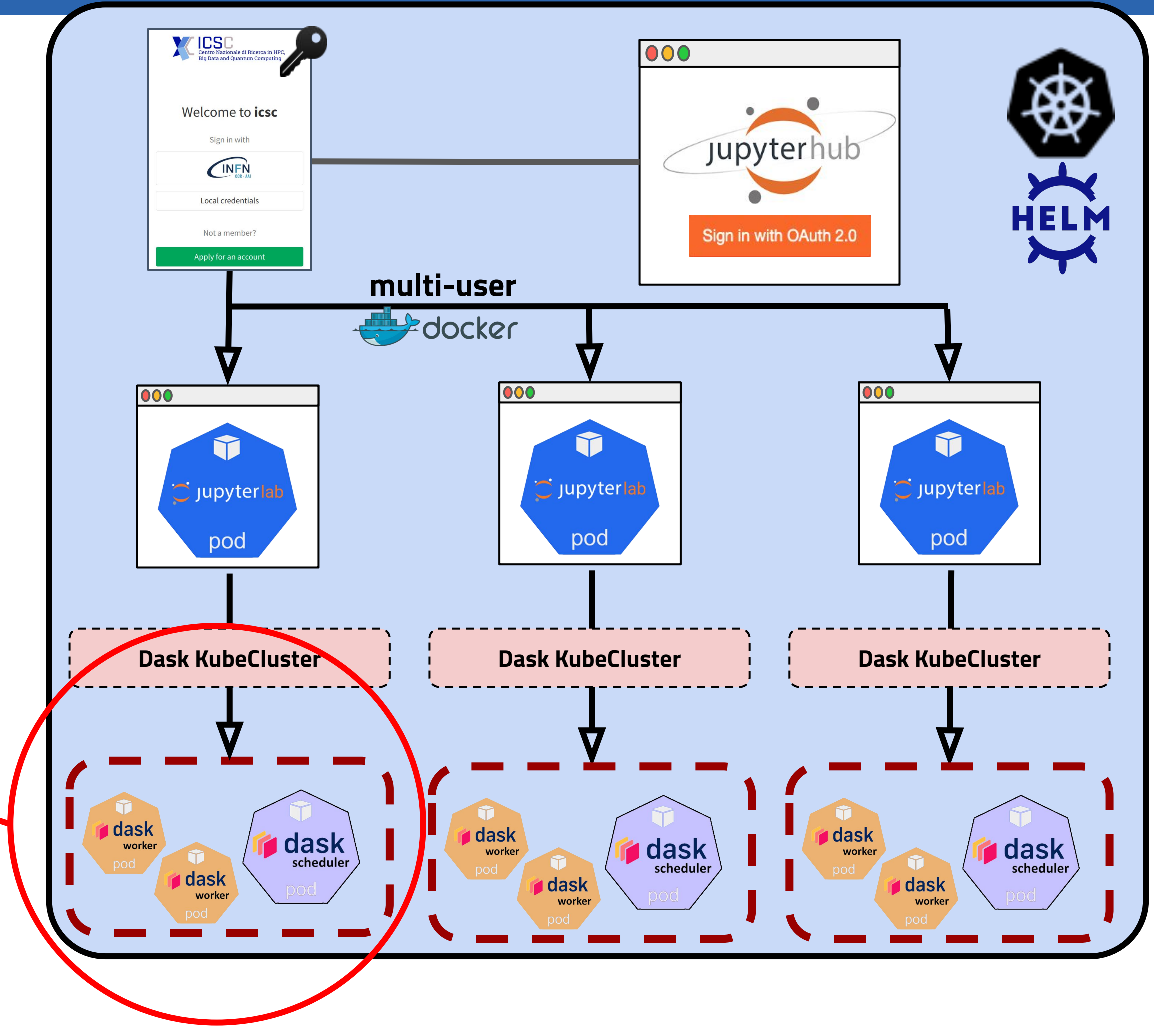
```
[1]: from dask.distributed import Client

     client = Client("localhost:22631")
     client

/usr/local/share/miniconda/lib/python3.10/site-packages/distributed/client.py:1309: VersionMismatchWarning: Mismatched versions found

+---------+-----------------+---------------+-----------------+
| Package | Client          | Scheduler     | Workers         |
+---------+-----------------+---------------+-----------------+
| lz4     | 4.0.0           | None          | 4.0.0           |
| msgpack | 1.0.3           | 1.0.5         | 1.0.3           |
| python  | 3.10.10.final.0 | 3.9.9.final.0 | 3.10.10.final.0 |
| toolz   | 0.12.0          | 0.11.1        | 0.12.0          |
+---------+-----------------+---------------+-----------------+
Notes:
-  msgpack: Variation is ok, as long as everything is above 0.6
  warnings.warn(version_module.VersionMismatchWarning(msg[0]["warning"]))
```

[1]: Client

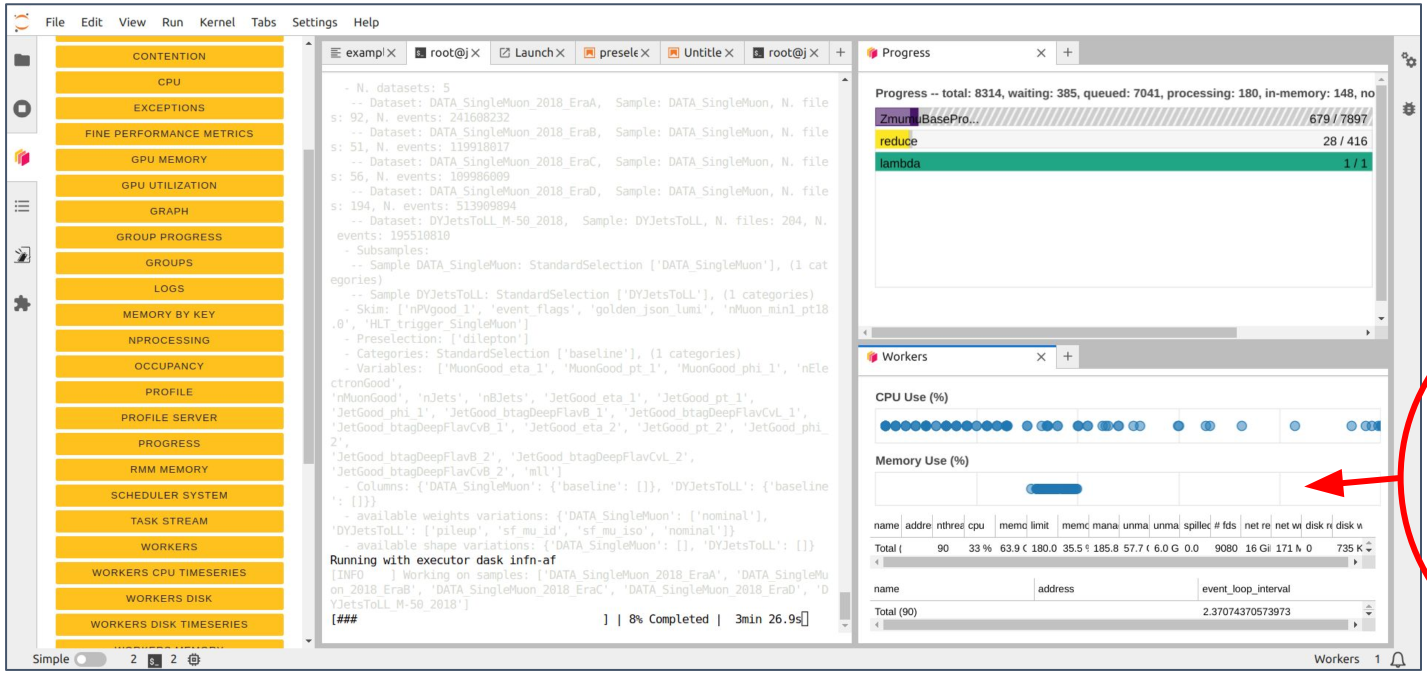
Client-ce7539b8-e288-11ed-81dd-7a36feca5287

Connection method: Direct

Dashboard: http://localhost:31645/status



Distributing the workload



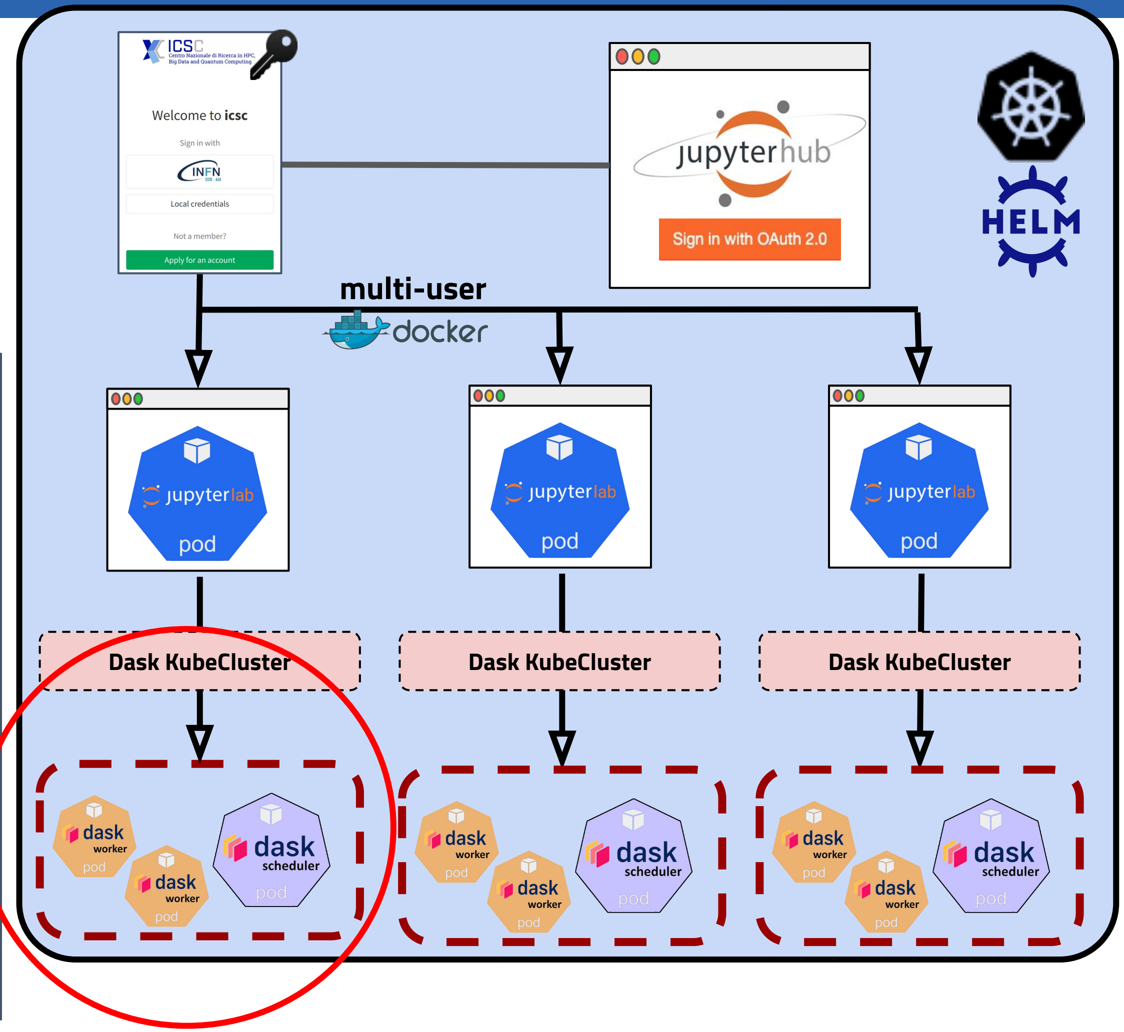
```
  - N. datasets: 5
   -- Dataset: DATA_SingleMuon_2018_EraA,  Sample: DATA_SingleMuon, N. file
s: 92, N. events: 241608232
   -- Dataset: DATA_SingleMuon_2018_EraB,  Sample: DATA_SingleMuon, N. file
s: 51, N. events: 119918017
   -- Dataset: DATA_SingleMuon_2018_EraC,  Sample: DATA_SingleMuon, N. file
s: 56, N. events: 109986009
   -- Dataset: DATA_SingleMuon_2018_EraD,  Sample: DATA_SingleMuon, N. file
s: 194, N. events: 513909894
   -- Dataset: DYJetsToLL_M-50_2018,  Sample: DYJetsToLL, N. files: 204, N.
 events: 195510810
  - Subsamples:
   -- Sample DATA_SingleMuon: StandardSelection ['DATA_SingleMuon'], (1 cat
egories)
   -- Sample DYJetsToLL: StandardSelection ['DYJetsToLL'], (1 categories)
  - Skim: ['nPVgood_1', 'event_flags', 'golden_json_lumi', 'nMuon_min1_pt18
.0', 'HLT_trigger_SingleMuon']
  - Preselection: ['dilepton']
  - Categories: StandardSelection ['baseline'], (1 categories)
  - Variables:  ['MuonGood_eta_1', 'MuonGood_pt_1', 'MuonGood_phi_1', 'nEle
ctronGood',
'nMuonGood', 'nJets', 'nBJets', 'JetGood_eta_1', 'JetGood_pt_1',
'JetGood_phi_1', 'JetGood_btagDeepFlavB_1', 'JetGood_btagDeepFlavCvL_1',
'JetGood_btagDeepFlavCvB_1', 'JetGood_eta_2', 'JetGood_pt_2', 'JetGood_phi_
2',
'JetGood_btagDeepFlavB_2', 'JetGood_btagDeepFlavCvL_2',
'JetGood_btagDeepFlavCvB_2', 'mll']
  - Columns: {'DATA_SingleMuon': {'baseline': []}, 'DYJetsToLL': {'baseline
': []}}
  - available weights variations: {'DATA_SingleMuon': ['nominal'],
'DYJetsToLL': ['pileup', 'sf_mu_id', 'sf_mu_iso', 'nominal']}
  - available shape variations: {'DATA_SingleMuon': [], 'DYJetsToLL': []}
Running with executor dask infn-af
[INFO    ] Working on samples: ['DATA_SingleMuon_2018_EraA', 'DATA_SingleMu
on_2018_EraB', 'DATA_SingleMuon_2018_EraC', 'DATA_SingleMuon_2018_EraD', 'D
YJetsToLL_M-50_2018']
[###                                       ] | 8% Completed |  3min 26.9s
```

Progress -- total: 8314, waiting: 385, queued: 7041, processing: 180, in-memory: 148, no

ZmumuBasePro...	679 / 7897
reduce	28 / 416
lambda	1 / 1

name	addre	nthrea	cpu	memo	limit	memo	mana	unma	unma	spilled	# fds	net re	net wi	disk re	disk w
Total (		90	33 %	63.9 (	180.0	35.5 °	185.8	57.7 (	6.0 G	0.0	9080	16 Gi	171 M	0	735 K

name	address	event_loop_interval
Total (90)		2.37074370573973

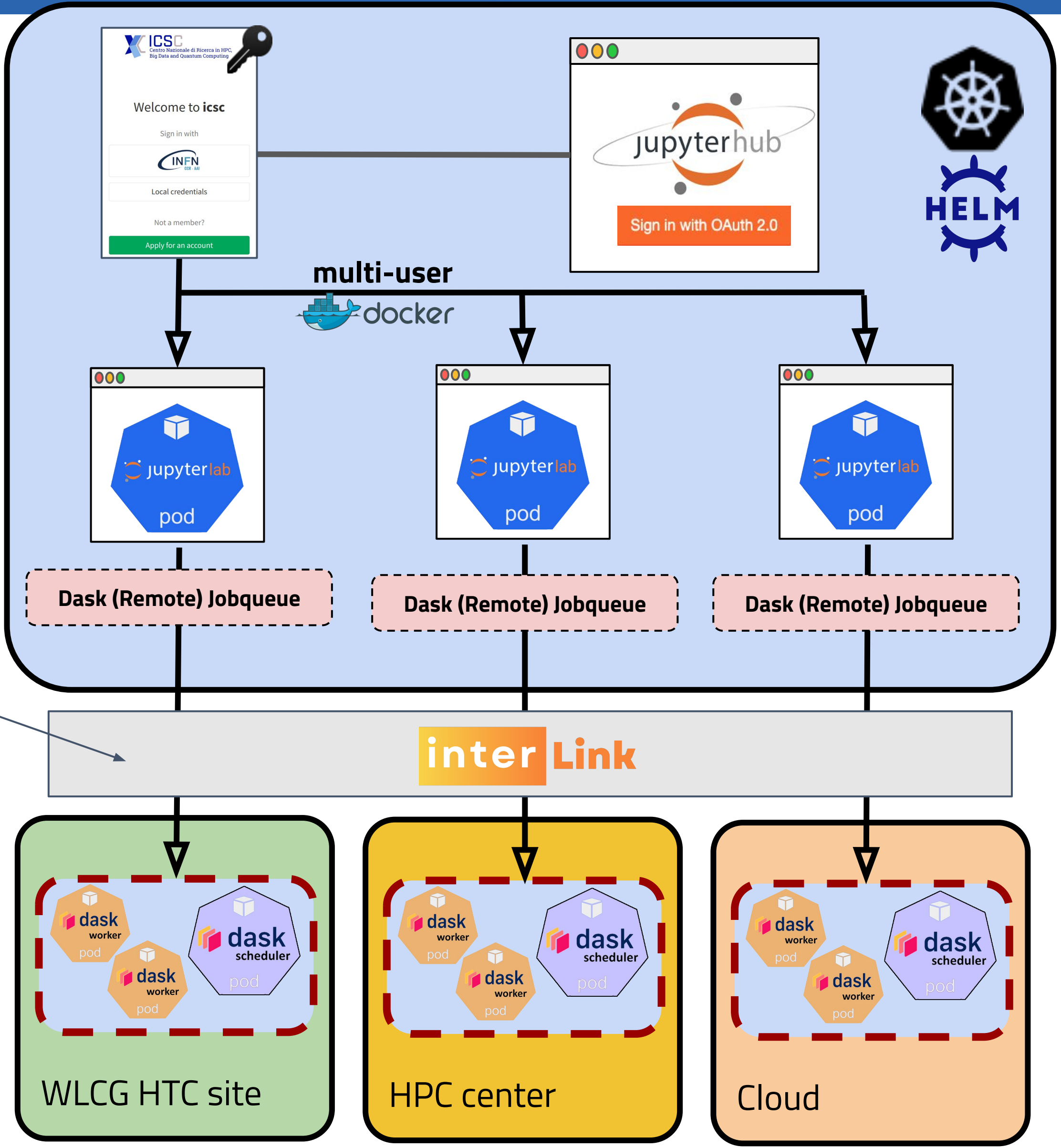


Offloading

Aim: enable the platform to **dynamically exploit all kinds of resources (HTC, HPC, Cloud)** transparently for the user

- looking for a synergy with active developments in this context, to delegate container execution on remote resources while keeping the very same user interface
- Possible solution: InterLink, which provides execution of a Kubernetes pod on almost any remote resource
  - Resources visible to the user thanks to an **HTCondor overlay**

CMS INFN Analysis Facility already implements this solution to integrate italian grid sites



Data are important: Rucio integration

INFN

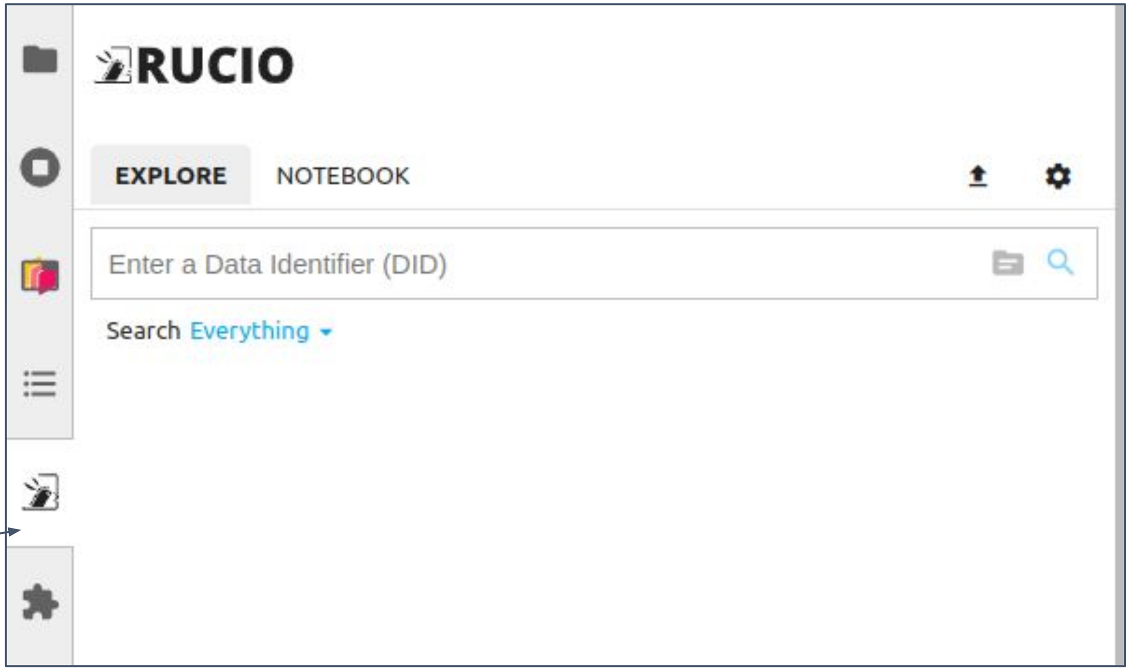
High rate platform also integrates with Rucio:

- a project that provides services and associated libraries for allowing scientific collaborations to **manage large volumes of data spread across facilities** at multiple institutions and organisations.

Rucio client enables users to interact with the system and access the distributed data:

- **The client can upload, download, manage and delete** everything from single files up to Petabyte sized datasets.

A **JupyterLab extension** integrates with Rucio to allow users to access some of Rucio's capabilities directly from the JupyterLab interface



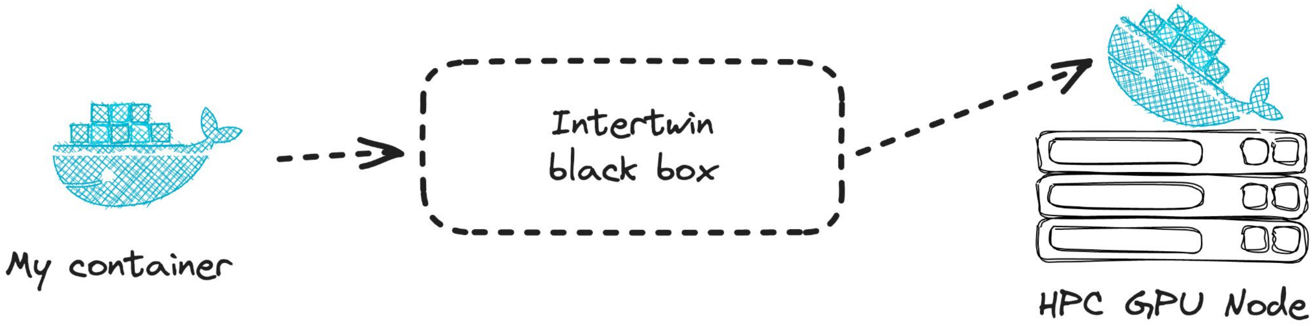
Conclusions

- Now you got an overview of each component of a typical high rate analysis platform…
  - IT'S TIME TO GET YOUR HANDS DIRTY!
    - Using some of the tools that will be demonstrated during the workshop
  - We will use the ICSC hub accessible at this endpoint https://hub.131.154.98.51.myip.cloud.infn.it/

Backup

What is offloading?

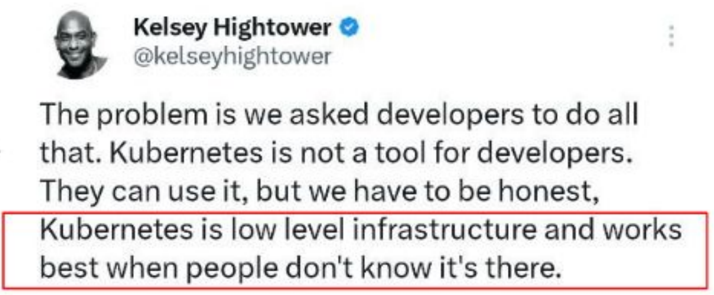
- **Delegate the execution of a container/workflow on remote resources while keeping the user interface unchanged.**
- Example:
  - "I have my own ML training container, and I want to run it on a node with 4 A100s"



How can we implement offloading?

We want a NATIVE integration with the Kubernetes primitives, acting underneath as a virtual node.

N.B. We aim to use Kubernetes as the workhorse for the "offloading", NOT as the user interface though →



Kelsey Hightower
@kelseyhightower

The problem is we asked developers to do all that. Kubernetes is not a tool for developers. They can use it, but we have to be honest, Kubernetes is low level infrastructure and works best when people don't know it's there.

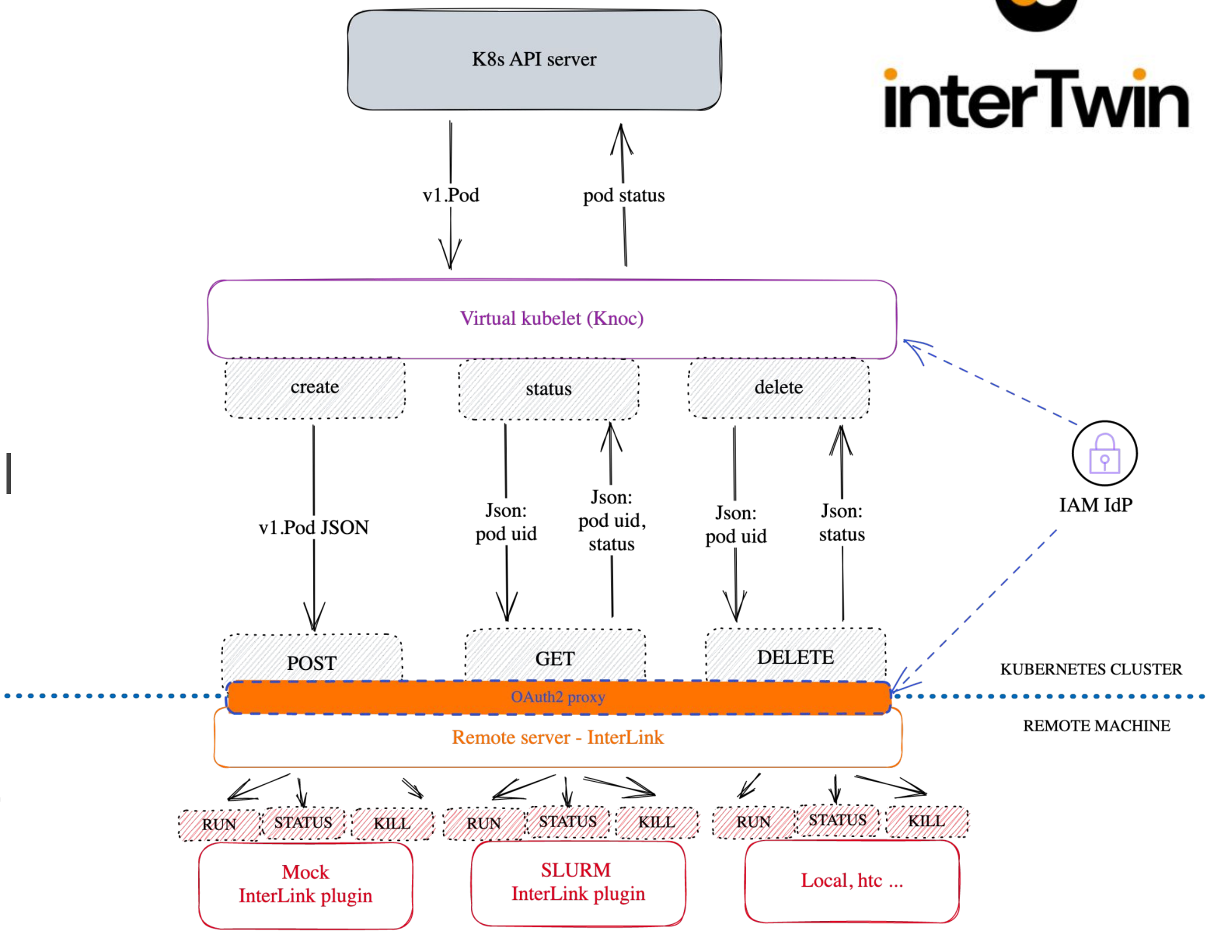
Offloading should be transparent for the users

A possible solution: InterLink

InterLink aims to provide an abstraction for the execution of a Kubernetes pod on any remote resource capable of managing a container execution lifecycle.

The project consists of two main components:

- **A Kubernetes Virtual Node**: based on the VirtualKubelet technology. Translating request for a kubernetes pod execution into a remote call to the interLink API server.
- **The interLink API server**: a modular and pluggable REST server where you can create your own container manager plugin (called sidecar), or use the existing ones: remote docker execution on a remote host, singularity Container on a remote SLURM or **HTCondor batch system**, etc…



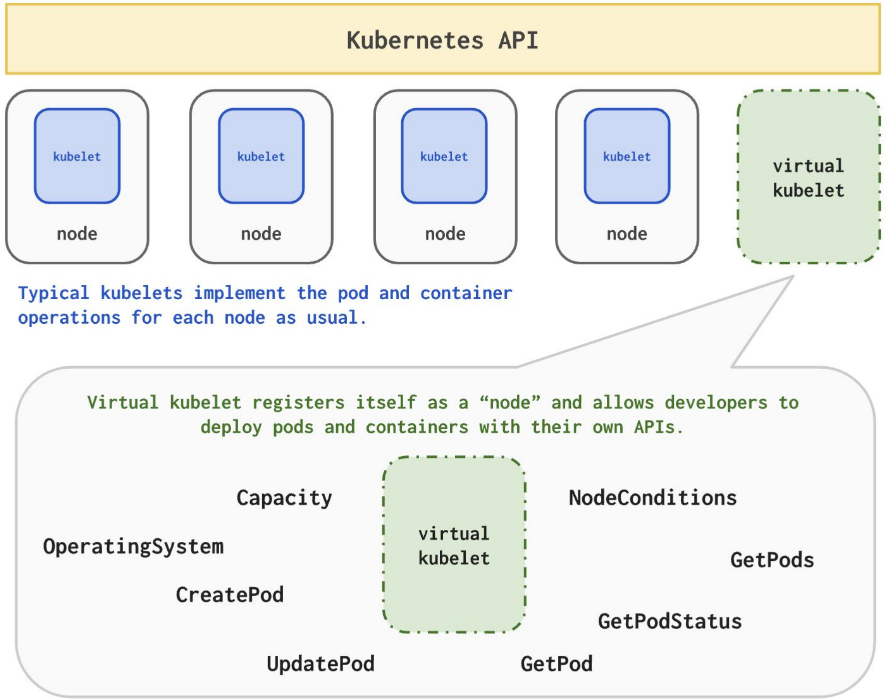
https://github.com/interTwin-eu/interLink

interTwin

Components: VK

https://virtual-kubelet.io/

- **Virtual kubelet (VK)**:
  - "Open-source Kubernetes kubelet implementation that masquerades as a kubelet. This allows Kubernetes nodes to be backed by Virtual Kubelet providers"
- Can be imagined as a translation layer:
  - "I take your pod and run your container wherever I want"
- Registers virtual node and pulls work to run
- The pod lifecycle is managed via interlink rest calls
- Oauth2 via service token kept "refreshed"

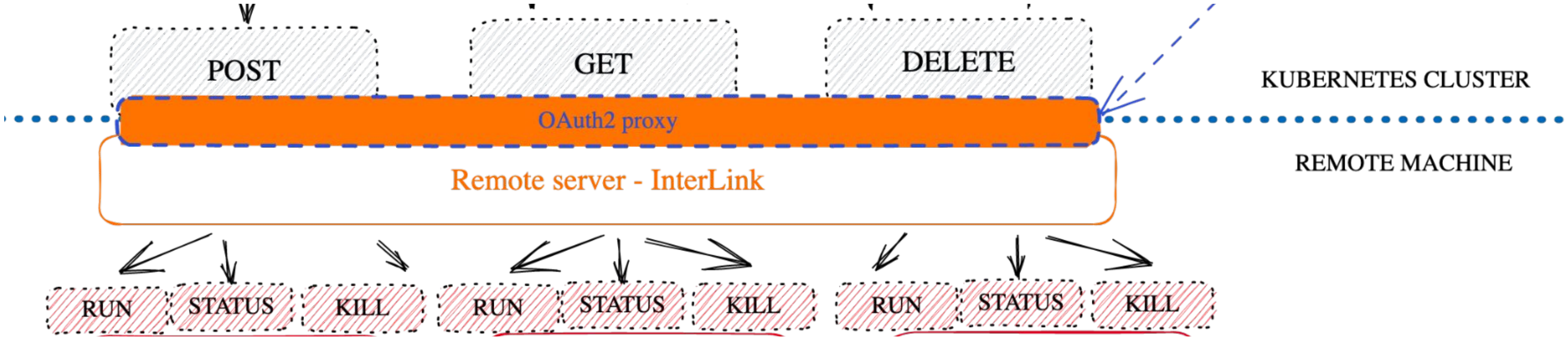


Typical kubelets implement the pod and container operations for each node as usual.

Virtual kubelet registers itself as a "node" and allows developers to deploy pods and containers with their own APIs.

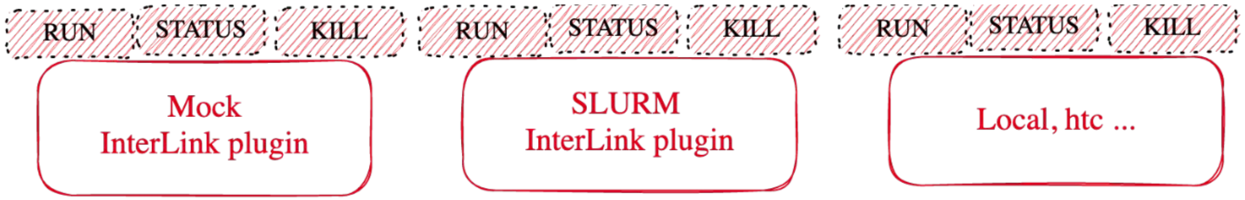
Components: Interlink + Oauth2 proxy

- Oauth2 proxy: authN with IAM and authZ configurable on aud and groups
- “Digests” and manipulates calls from VK to the sidecar
- Self contained binary, distributable on all OS without dependencies



Components: Sidecar/Plugin

- Agent that must expose a REST with defined specs, but which can be implemented in the language and with the methods you prefer:
  - creation of the pod: run local docker or submit a job on htc, slurm etc
  - collect the execution states
  - collect and forward logs upon request
  - kill
- Existing plugins: local Docker (Go), Slurm (Go), HTCondor (python), ARC (python), Kubernetes (python), Kueue (python)



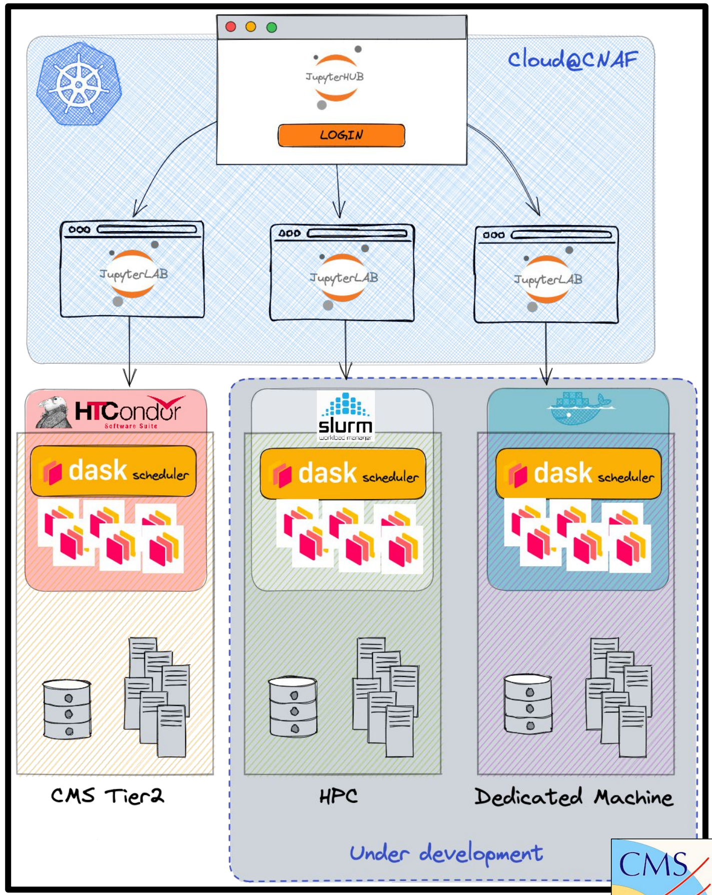
CMS INFN AF analysis

A growing project (activity started in 2020):

- Now 28 registered users: at least 4/5 of them can be considered “very active” users, authenticated via CMS IAM
- Containerized Jupyterlab environment:
  - Both “a-la-batch” and interactive processing allowed
- Integration of heterogeneous resources under the same pool:
  - Existing WLCG infrastructure and batch-systems for interactive use used for both legacy and interactive processing:
  - Using an HTCondor overlay and Dask in HTCondor mode
- offload on all Italian Tier2 sites via Interlink mechanism:
  - Deployment of Dask clusters on remote resources via RemoteHTCondor (Dask-jobqueue plugin)

What users see

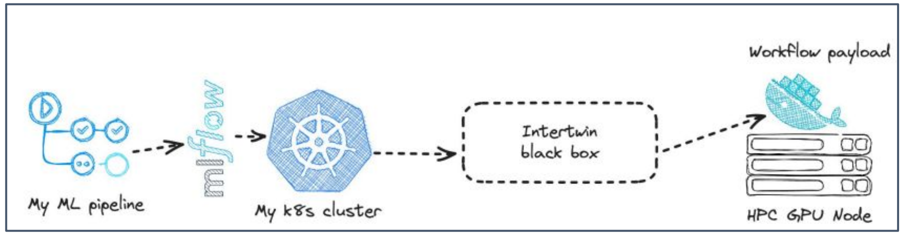
What the offloading hides to the user

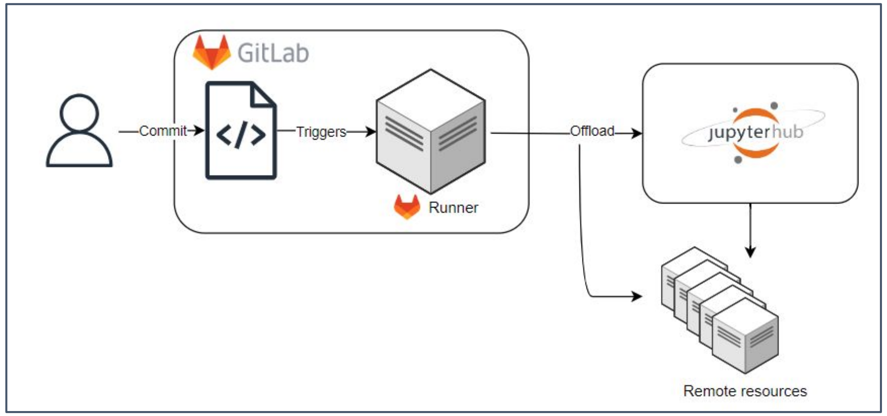


Enabling use cases

These are some use cases we have in mind as a scientific community:

- **Unlock full power of cutting-edge analysis tools**
  - Speed-up of factor O(10-100) for HEP analysis workflows
- **Easy GPU access**:
  - seamless access to HPC centers
  - ML training triggered via workflow automation, e.g. ML pipelining tools (Kubeflow, MLflow, …)
  - many GPUs at once == more/faster hyperparameter optimization
- Enable **CI/CD as a trigger for analysis execution**
- …





InterLink: development context and ICSC related activities

INFN

The technical solution (interLink) has been initially prototyped by INFN in the context of the interTwin EU Funded project and is now enhanced within the ICSC development/research programme.

In particular

- **It is part of the Spoke0** infrastructural toolkits. As such it is under consolidation, testing and improvement
- **It is part of the Spoke2 - WP5 work plan**
  - in this respect there is a ongoing integration effort to extend the High rate analysis platform over HTC/HPC computing resources
- **Also part of the Spoke3 integration plan**
  - idea is to benefit of the interLink capabilities to offer highly dynamic access to specialized HW (i.e. over Leonardo)
  - integrating offloading with data retrieval from the data-lake prototype

Many fruitful sinergies should lead toward a generic technical solution, versatile and extendible based on specific needs.