Google R&D: PanDA and Dask setups

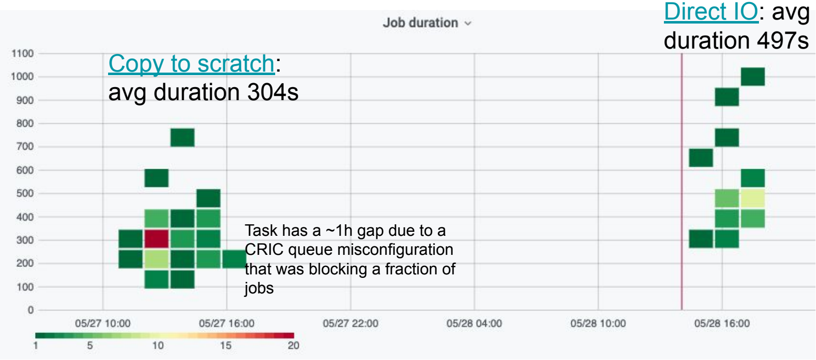
Fernando Barreiro Megino

UNIVERSITY OF
TEXAS
ARLINGTON

PanDA + Rucio setup

PanDA GKE **analysis** queue: GOOGLE100

- Cluster details
  - 0-10 **autoscaled, preemptible** nodes: n2-standard-8 (=8 cores, 32GB RAM)
    - Scaling down under discussion with Usman and Jason Nichols (GKE specialist)
  - Local **SSD** at each node
- Queue status “Brokeroff”: you need to specifically set the queue when submitting to PanDA
- Data needs to be pre-placed to RSE GOOGLE_EU
  - Jobs will stay in “**assigned**” if data not present
  - Requires **special permission/quota** in Rucio to interact with this storage element
- 11TB input task completed successfully (thanks to Nikolai Hartmann)
  - ~12GB input per job

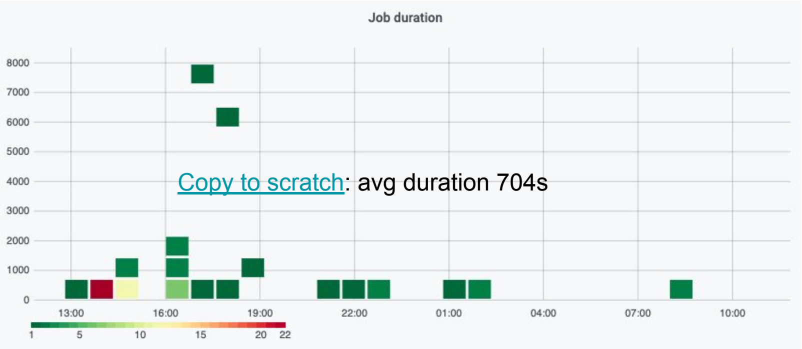


Copy to scratch: avg duration 304s

Direct IO: avg duration 497s

Task has a ~1h gap due to a CRIC queue misconfiguration that was blocking a fraction of jobs

GOOGLE100



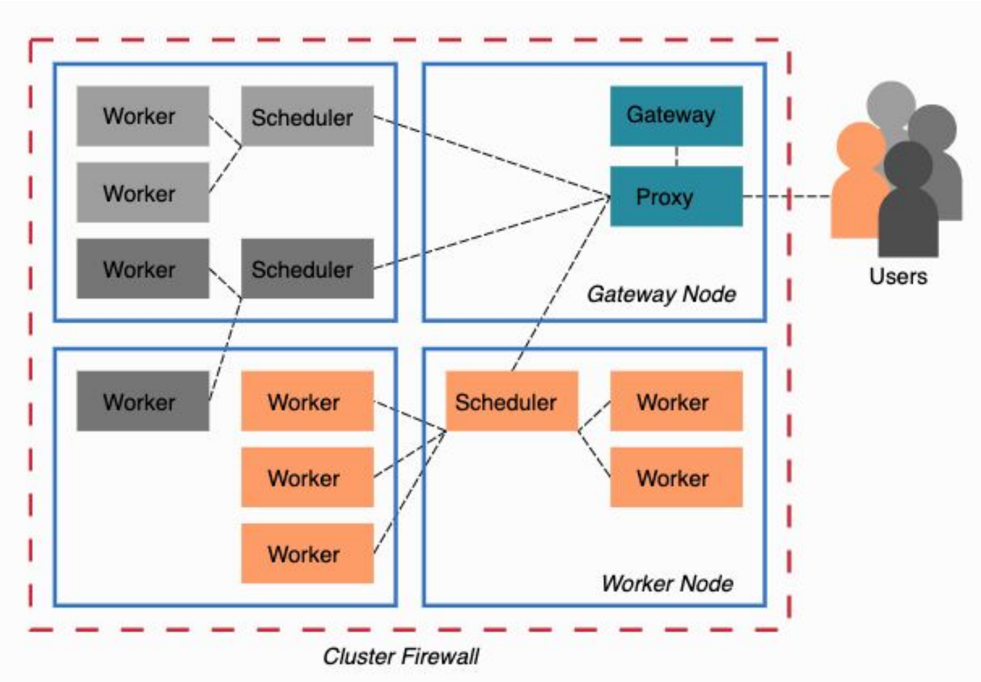
Copy to scratch: avg duration 704s

Grid

Colours are #occurrences, not failure rate!!!

Dask+JupyterHub

Dask + Jupyter setup



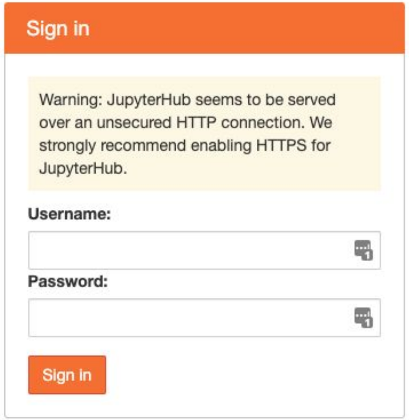
- Sets up common Dask cluster and JupyterHub for all users
- Users have access to JupyterHub and Dask, but not to GCP/GKE

- Disadvantage: less flexibility for individual customization. Needs central maintenance of a set of images that work for everybody

- Current installation on modest cluster: 3 e2-standard-8 nodes with 100GB disk

JupyterHub

http://jupyter.gcp4hep.org/



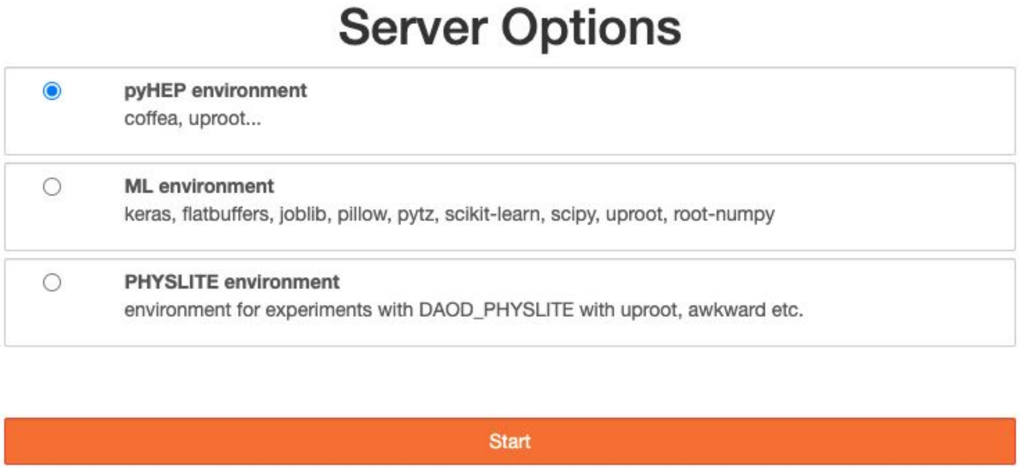
Sign in

Warning: JupyterHub seems to be served over an unsecured HTTP connection. We strongly recommend enabling HTTPS for JupyterHub.

Username:

Password:

- Local accounts. I need to add new users
- Integration with other identity providers possible. Could be done only if there is real usage



Server Options

● **pyHEP environment**
coffea, uproot...

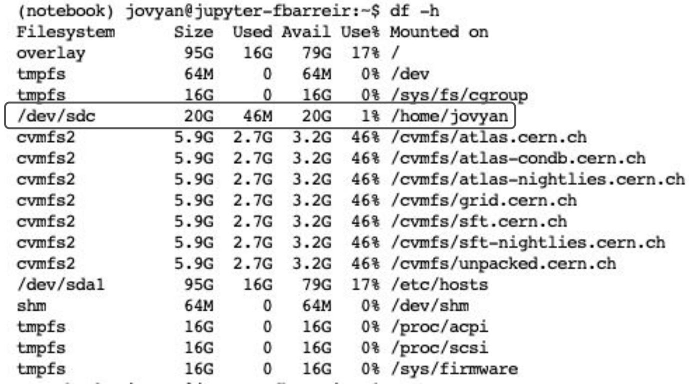
○ **ML environment**
keras, flatbuffers, joblib, pillow, pytz, scikit-learn, scipy, uproot, root-numpy

○ **PHYSLITE environment**
environment for experiments with DAOD_PHYSLITE with uproot, awkward etc.

- Available images
  - pyHEP environment: dependencies suggested in this tutorial
  - ML environment: dependencies requested by Fang-Ying
  - PHYSLITE environment: image provided by Nikolai
- Images hosted in GCP Container Registry
- Image management is **very time consuming**

Jupyter notebook specs

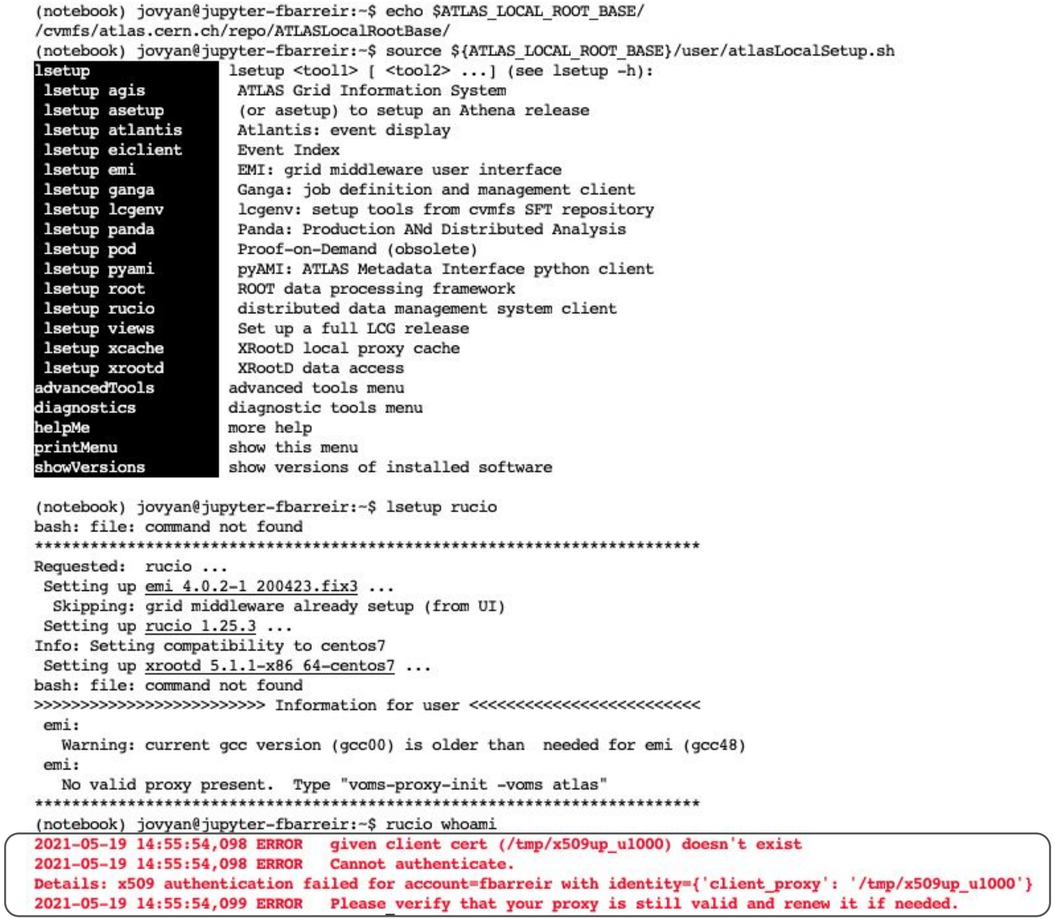
- Each user notebook runs on an **independent** pod with image selected at startup
  - “**Burstable**” QoS with 1GB RAM base request
  - How much you can burst depends on overall cluster usage and occupancy of the node
- User home directory: 10GB
  - Independent persistent disk
  - Value is configurable
  - Disk can also be manually extended
  - Anything outside the home directory gets **cleaned up** once notebook stops
    - A potential conda user environment installed on home directory would survive



```
(notebook) jovyan@jupyter-fbarreir:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay          95G   16G   79G  17% /
tmpfs            64M     0   64M   0% /dev
tmpfs            16G     0   16G   0% /sys/fs/cgroup
/dev/sdc         20G   46M   20G   1% /home/jovyan
cvmfs2          5.9G  2.7G  3.2G  46% /cvmfs/atlas.cern.ch
cvmfs2          5.9G  2.7G  3.2G  46% /cvmfs/atlas-condb.cern.ch
cvmfs2          5.9G  2.7G  3.2G  46% /cvmfs/atlas-nightlies.cern.ch
cvmfs2          5.9G  2.7G  3.2G  46% /cvmfs/grid.cern.ch
cvmfs2          5.9G  2.7G  3.2G  46% /cvmfs/sft.cern.ch
cvmfs2          5.9G  2.7G  3.2G  46% /cvmfs/sft-nightlies.cern.ch
cvmfs2          5.9G  2.7G  3.2G  46% /cvmfs/unpacked.cern.ch
/dev/sda1        95G   16G   79G  17% /etc/hosts
shm              64M     0   64M   0% /dev/shm
tmpfs            16G     0   16G   0% /proc/acpi
tmpfs            16G     0   16G   0% /proc/scsi
tmpfs            16G     0   16G   0% /sys/firmware
```

CVMFS

- Available from Jupyter session and Dask workers
- You can for example interact with Rucio to list file replicas and get signed URLs
  - Requires uploading your voms proxy to the notebook
- Not all CVMFS software is compatible with local environment, e.g. GFAL2 breaks rucio downloads

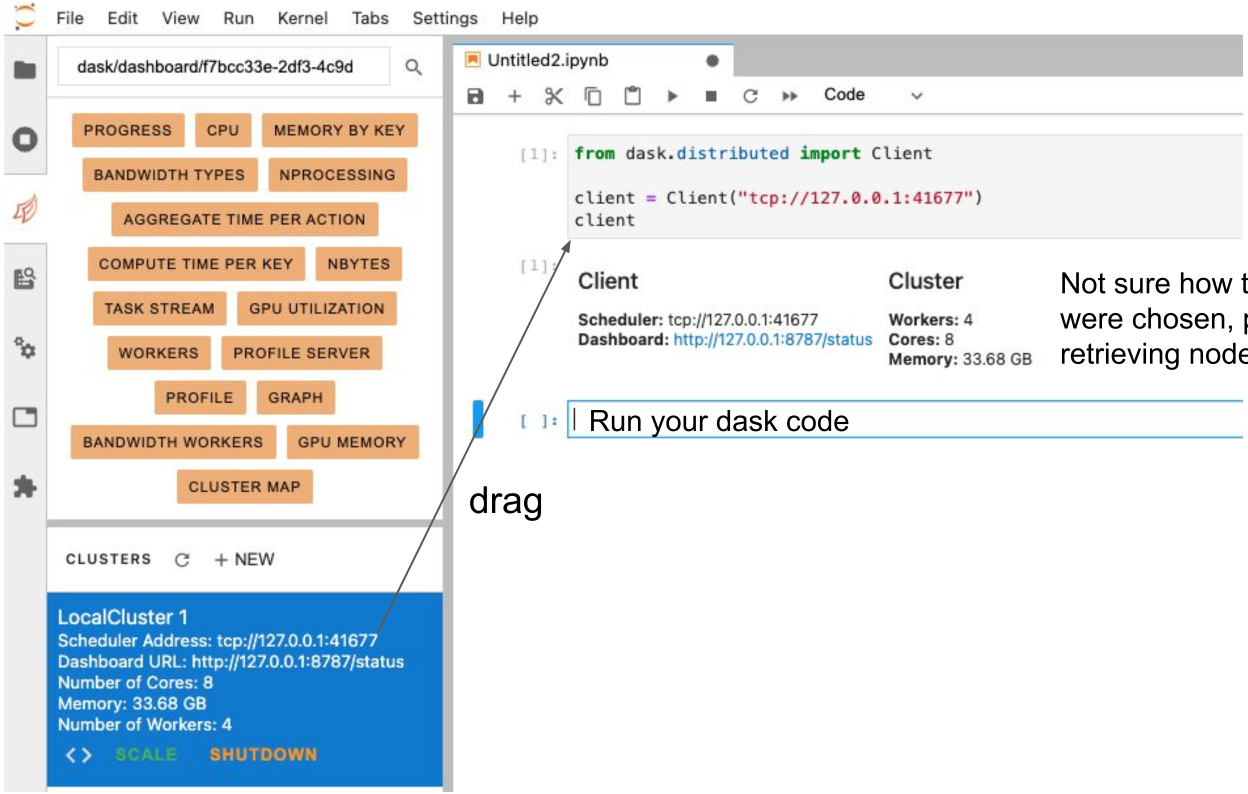


```
(notebook) jovyan@jupyter-fbarreir:~$ echo $ATLAS_LOCAL_ROOT_BASE/
/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase/
(notebook) jovyan@jupyter-fbarreir:~$ source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh
lsetup                   lsetup <tool1> [ <tool2> ...] (see lsetup -h):
 lsetup agis              ATLAS Grid Information System
 lsetup asetup            (or asetup) to setup an Athena release
 lsetup atlantis          Atlantis: event display
 lsetup eiclient          Event Index
 lsetup emi               EMI: grid middleware user interface
 lsetup ganga             Ganga: job definition and management client
 lsetup lcgenv            lcgenv: setup tools from cvmfs SFT repository
 lsetup panda             Panda: Production ANd Distributed Analysis
 lsetup pod               Proof-on-Demand (obsolete)
 lsetup pyami             pyAMI: ATLAS Metadata Interface python client
 lsetup root              ROOT data processing framework
 lsetup rucio             distributed data management system client
 lsetup views             Set up a full LCG release
 lsetup xcache            XRootD local proxy cache
 lsetup xrootd            XRootD data access
advancedTools            advanced tools menu
diagnostics              diagnostic tools menu
helpMe                   more help
printMenu                show this menu
showVersions             show versions of installed software

(notebook) jovyan@jupyter-fbarreir:~$ lsetup rucio
bash: file: command not found
************************************************************************
Requested:  rucio ...
 Setting up emi 4.0.2-1 200423.fix3 ...
  Skipping: grid middleware already setup (from UI)
 Setting up rucio 1.25.3 ...
Info: Setting compatibility to centos7
 Setting up xrootd 5.1.1-x86 64-centos7 ...
bash: file: command not found
>>>>>>>>>>>>>>>>>>>>>>>>> Information for user <<<<<<<<<<<<<<<<<<<<<<<<<
 emi:
   Warning: current gcc version (gcc00) is older than  needed for emi (gcc48)
 emi:
   No valid proxy present.  Type "voms-proxy-init -voms atlas"
************************************************************************
(notebook) jovyan@jupyter-fbarreir:~$ rucio whoami
2021-05-19 14:55:54,098 ERROR   given client cert (/tmp/x509up_u1000) doesn't exist
2021-05-19 14:55:54,098 ERROR   Cannot authenticate.
Details: x509 authentication failed for account=fbarreir with identity={'client_proxy': '/tmp/x509up_u1000'}
2021-05-19 14:55:54,099 ERROR   Please verify that your proxy is still valid and renew it if needed.
```

Spinning up a Dask cluster: LOCAL

LOCAL: your cluster lives in your jupyter pod



LocalCluster 1
Scheduler Address: tcp://127.0.0.1:41677
Dashboard URL: http://127.0.0.1:8787/status
Number of Cores: 8
Memory: 33.68 GB
Number of Workers: 4

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

     client = Client("tcp://127.0.0.1:41677")
     client
```

[1]:

Client	Cluster
Scheduler: tcp://127.0.0.1:41677	**Workers:** 4
Dashboard: http://127.0.0.1:8787/status	**Cores:** 8
	Memory: 33.68 GB

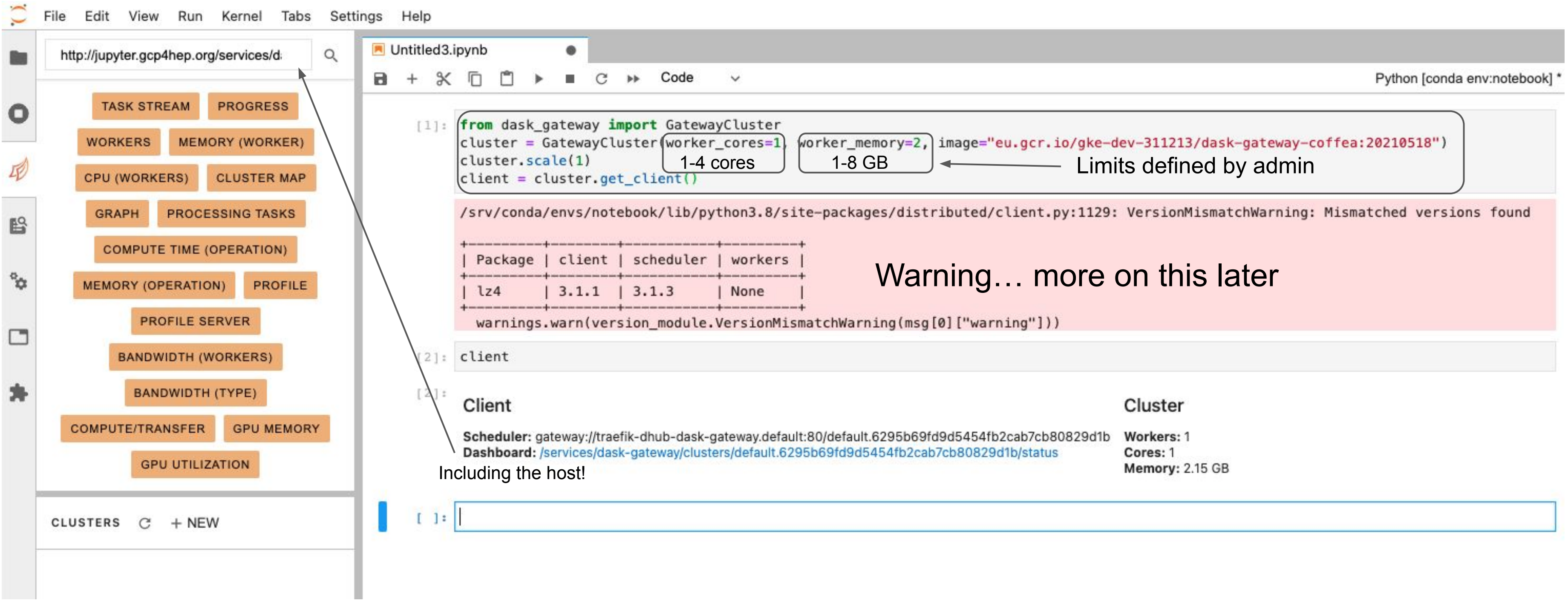
```
[ ]: Run your dask code
```

drag

Not sure how these values were chosen, probably by retrieving node size

Spinning up a Dask cluster: distributed

Distributed: each worker gets an independent pod, so you can scale to multiple nodes



```
[1]: from dask_gateway import GatewayCluster
     cluster = GatewayCluster(worker_cores=1, worker_memory=2, image="eu.gcr.io/gke-dev-311213/dask-gateway-coffea:20210518")
     cluster.scale(1)
     client = cluster.get_client()
```

1-4 cores

1-8 GB

Limits defined by admin

```
/srv/conda/envs/notebook/lib/python3.8/site-packages/distributed/client.py:1129: VersionMismatchWarning: Mismatched versions found

+---------+--------+-----------+---------+
| Package | client | scheduler | workers |
+---------+--------+-----------+---------+
| lz4     | 3.1.1  | 3.1.3     | None    |
+---------+--------+-----------+---------+
  warnings.warn(version_module.VersionMismatchWarning(msg[0]["warning"]))
```

Warning… more on this later

```
[2]: client
```

[2]:

Client

Scheduler: gateway://traefik-dhub-dask-gateway.default:80/default.6295b69fd9d5454fb2cab7cb80829d1b
Dashboard: /services/dask-gateway/clusters/default.6295b69fd9d5454fb2cab7cb80829d1b/status

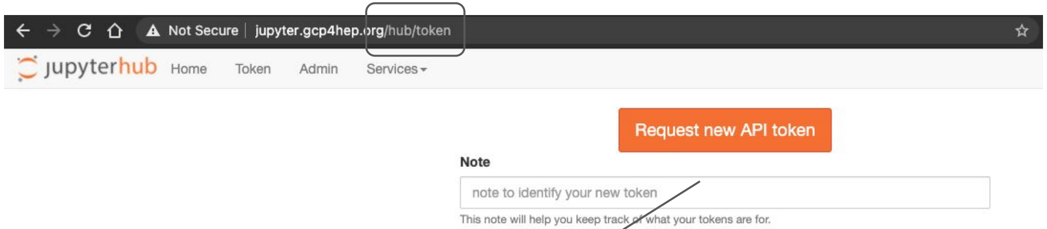
Cluster

Workers: 1
Cores: 1
Memory: 2.15 GB

Including the host!

```
[ ]:
```

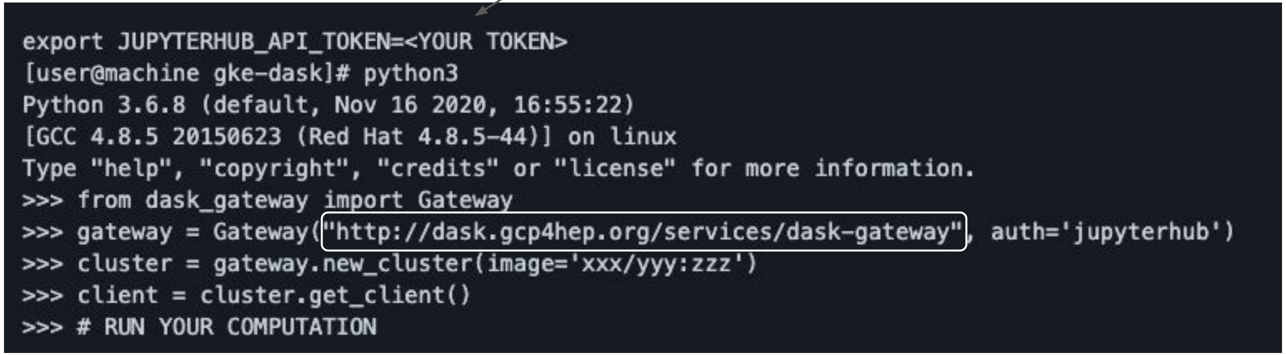
Spinning a dask cluster from an arbitrary python shell



Note

note to identify your new token

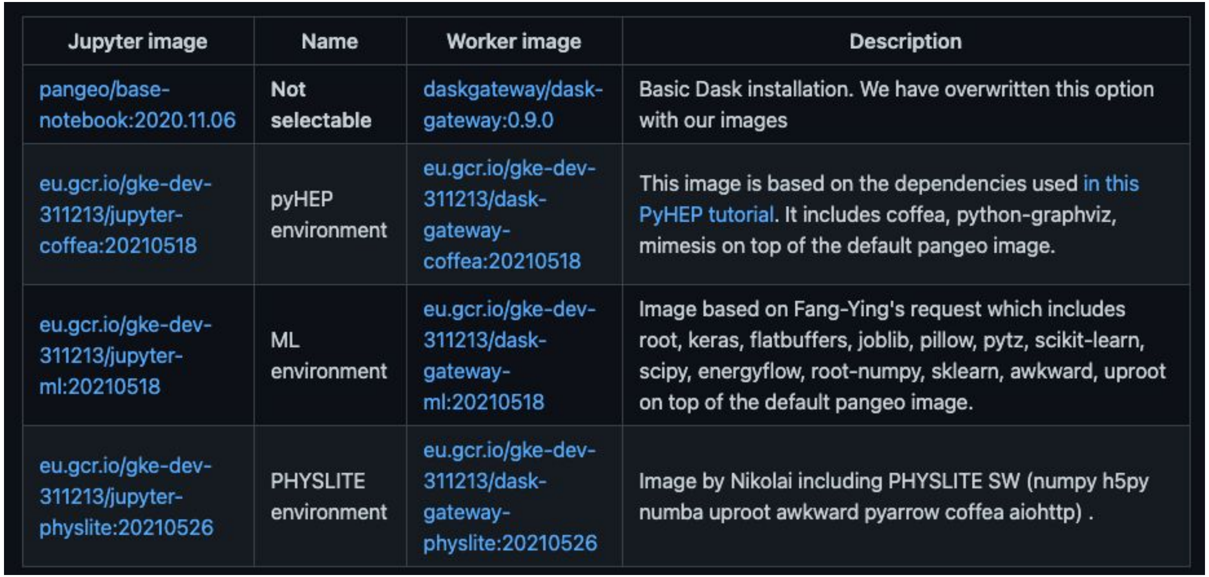
This note will help you keep track of what your tokens are for.



```
export JUPYTERHUB_API_TOKEN=<YOUR TOKEN>
[user@machine gke-dask]# python3
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from dask_gateway import Gateway
>>> gateway = Gateway("http://dask.gcp4hep.org/services/dask-gateway", auth='jupyterhub')
>>> cluster = gateway.new_cluster(image='xxx/yyy:zzz')
>>> client = cluster.get_client()
>>> # RUN YOUR COMPUTATION
```

Software compatibility

- The client (Jupyter or your python shell) and the Dask workers need to have compatible SW (dask, tensorflow...) installed
- I tried to generate compatible images for both Jupyter images building on the default pangeo/daskgateway images



Jupyter image	Name	Worker image	Description
pangeo/base-notebook:2020.11.06	**Not selectable**	daskgateway/dask-gateway:0.9.0	Basic Dask installation. We have overwritten this option with our images
eu.gcr.io/gke-dev-311213/jupyter-coffea:20210518	pyHEP environment	eu.gcr.io/gke-dev-311213/dask-gateway-coffea:20210518	This image is based on the dependencies used in this PyHEP tutorial. It includes coffea, python-graphviz, mimesis on top of the default pangeo image.
eu.gcr.io/gke-dev-311213/jupyter-ml:20210518	ML environment	eu.gcr.io/gke-dev-311213/dask-gateway-ml:20210518	Image based on Fang-Ying's request which includes root, keras, flatbuffers, joblib, pillow, pytz, scikit-learn, scipy, energyflow, root-numpy, sklearn, awkward, uproot on top of the default pangeo image.
eu.gcr.io/gke-dev-311213/jupyter-physlite:20210526	PHYSLITE environment	eu.gcr.io/gke-dev-311213/dask-gateway-physlite:20210526	Image by Nikolai including PHYSLITE SW (numpy h5py numba uproot awkward pyarrow coffea aiohttp) .

https://github.com/gcp4hep/analysis-cluster/wiki/Daskhub-images

Conclusions

- Infrastructure is ready and required features implemented
  - Desirable Dask features (https, oAuth) can be implemented depending on evolution of activity
  - Data management setup in Dask to be explored further
- First tests in progress
  - PanDA:
    - Nikolai: 11TB input task done, next is 100TB task
    - Lukas also has a potential idea
  - Dask:
    - Nikolai: Small validation done
    - Fang-Ying: evaluating whether the notebooks fit her needs
      - Working on rucio client dependency issues
- Paul has setup a separate single-user cluster
  - This model can be more appropriate for a potential PanDA integration
  - Still requires more experience and agreeing on architecture