Providing secure Interactive access to the HTCondor batch resources with JupyterHub

International Symposium on Grids & Clouds

Carmelo Pellegrino and Stefano Dal Pra - ISGC 2022 - 25/03/2022

Outline

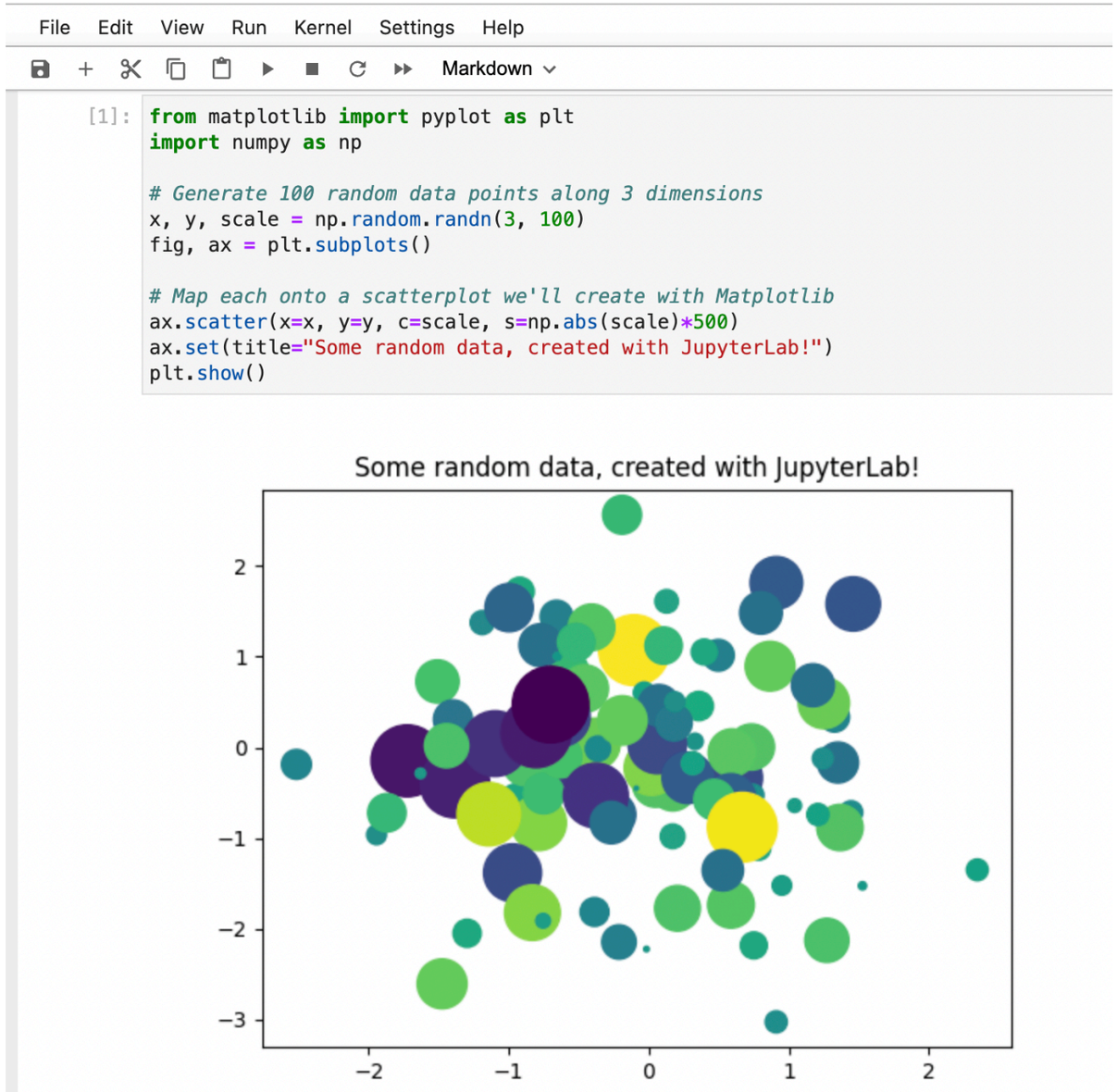
- The Jupyter ecosystem
- JupyterHub and HTCondor
  - batchspawner
- Implementation and security enforcements
  - singularity
  - custom batchspawner
- Additional features
  - Custom readme file
  - User-defined configuration file
  - Persistency

The Jupyter ecosystem

Notebook

https://jupyter.org/ - https://jupyter-notebook.readthedocs.io/en/latest/

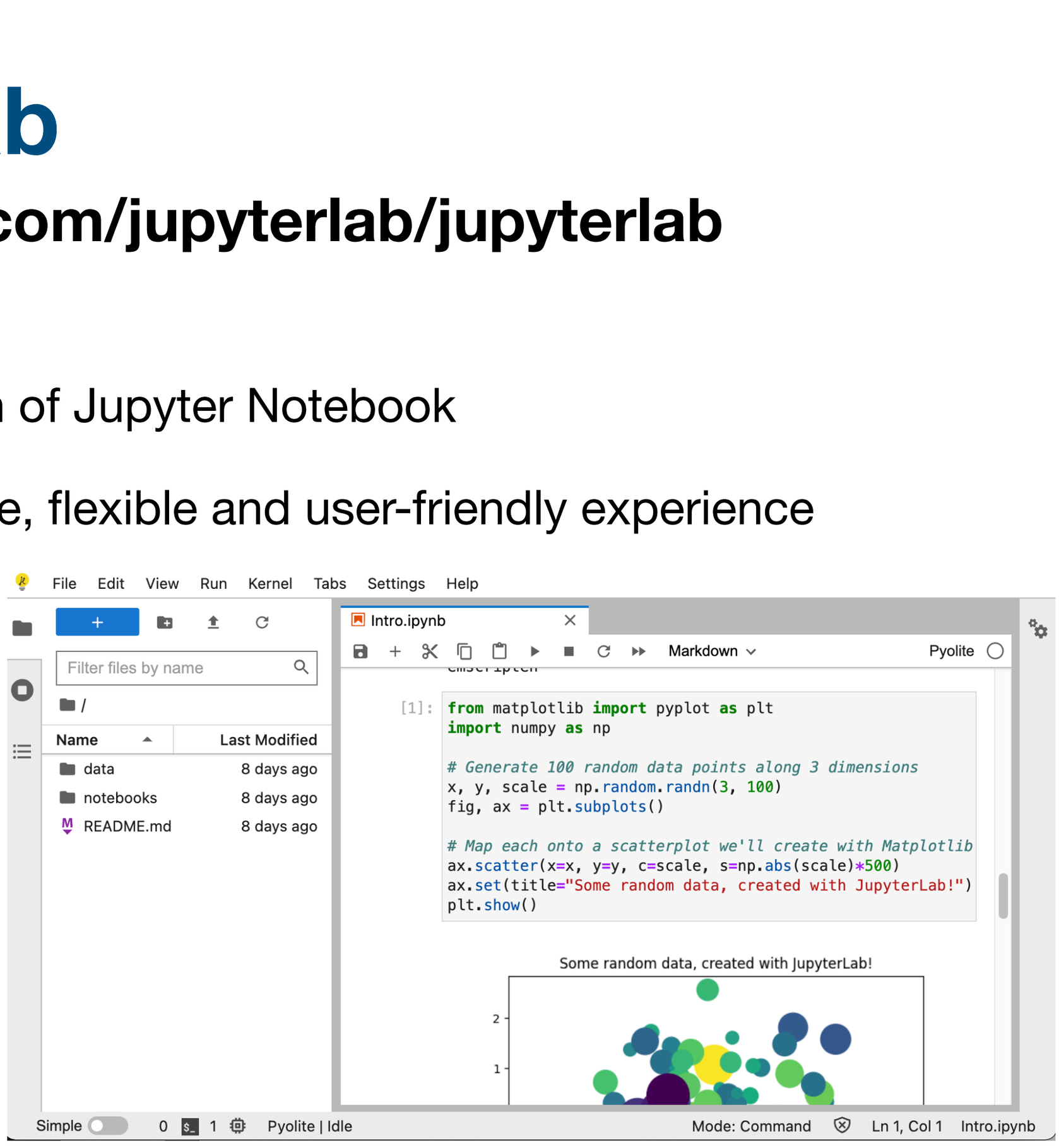
- Jupyter stands for **Ju**lia, **Pyt**hon, **R**
- Providing a novel interaction method initially for these languages, now supports over 150 different systems (called kernels)
- *The Jupyter Notebook is a web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience*
- Block of code-oriented workflow
- Supports inline pictures, tables, markdown-formatted text, etc...
- Everyone can run a notebook on its own PC



JupyterLab

https://github.com/jupyterlab/jupyterlab

- The new evolution of Jupyter Notebook
- Much more mature, flexible and user-friendly experience
- Alternative but active by default



JupyterHub

https://jupyterhub.readthedocs.io/en/latest

- Multi-user installation
- Supports several authentication methods, named *Authenticators*
- Supports several *Spawners*, i.e. notebook providers (e.g.: local, docker, **batchspawner**)
- Supports SSL both on the front-end and the back-end HTTP channels
- Configuration via a Python script (a.k.a. config.py)

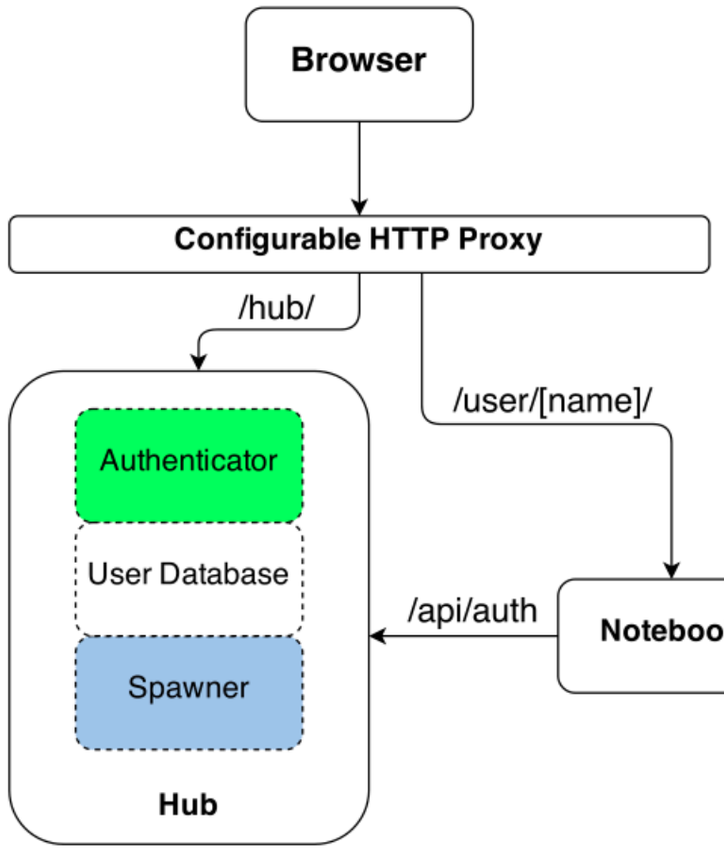
Network basics

https://jupyterhub.readthedocs.io/en/stable/getting-started/networking-basics.html

- **configurable-http-proxy** is a proxy of messages from and to the user's browser to interface with the Hub and the Notebooks

- **Hub** is interrogated by either the proxy and the notebooks.

- The **notebooks** can run on remote hosts. Expose a randomly generated port. Each notebook tells to the Hub which port it listens on. In turn the Hub configures the Proxy to forward user traffic to the notebook



credits: https://jupyterhub.readthedocs.io/en/0.9.1/

Notebook authentication

- On notebook submit, JupyterHub creates a unique token and passes it to the notebook
- The notebook uses the token to autenticate back to the Hub
- This token has to be kept **secure**!

JupyterHub and HTCondor

batchspawner

https://github.com/jupyterhub/batchspawner

- Is an implementation of the `Spawner` Python class
- Handles the interaction with several batch systems:
  - LSF, Slurm, PBS, Torque, ..., **HTCondor**
- Can configure, submit, poll, and remove remotely executed jobs that run the notebook applications and interface them with the user and the JupyterHub controls

batchspawner.CondorSpawner

- The default submit file looks like:

```
Executable = /bin/sh
RequestMemory = {memory}
RequestCpus = {nprocs}
Arguments = \"-c 'exec {cmd}'\"
Remote_Initialdir = {homedir}
Output = {homedir}/.jupyterhub.condor.out
Error = {homedir}/.jupyterhub.condor.err
ShouldTransferFiles = False
GetEnv = True
{options}
Queue
```

- {cmd} is replaced by batchspawner with the notebook executable and arguments
- Notebook authentication token is passed via environment variables (!)
- No HTCondor spool mechanism foreseen

batchspawner configuration

- The Hub configuration is all handled at once in a Python script
- The following lines are to enable and configure the CondorSpawner

```
import batchspawner                                          # mandatory
c.JupyterHub.spawner_class = 'batchspawner.CondorSpawner'    # set the spawner class
c.CondorSpawner.ip = '0.0.0.0'                               # IP address to bind => any
c.CondorSpawner.http_timeout = 1800                          # a reasonable timeout for the job
                                                             # to be in RUN state
```

Implementation and security enforcement

JupyterHub at INFN-Tier1

Why JupyterHub?

- Since several years, INFN-Tier1 local users ask the admins to support Jupyter notebooks
- Several autonomous and **insecure** attempts on Tier1 user interfaces by the users:
  - always chased by the admins to prevent security incidents
  - abuse of the resources in user interface
- Desiderata:
  - allow notebook execution on dedicated resources
  - accounting of the consumed CPU hours
  - security
  - avoid waste of unused computing resources
- The solution is https://jupyterhub-t1.cr.cnaf.infn.it

Implementation

- A set of worker nodes is dedicated to run notebooks
- Jupyter jobs are forced to have a "recognisable" name:

  condor_submit -spool **-batch-name jupyter-$USER**
- Job transform rule in the schedd configuration:

```
JOB_TRANSFORM_JupyterNotebook @=end
   REQUIREMENTS split(MY.JobBatchName, "-")[0] =?= "jupyter"
   if defined My.Requirements
      SET Requirements RegExp("wn-XX-YY", TARGET.Machine) && ( $(MY.Requirements) )
   else
      SET Requirements RegExp("wn-XX-YY", TARGET.Machine)
   endif
@end
```

- JupyterHub authenticates the users via their LDAP uid and Krb5 password

Security issues

with the stock design

- The notebook authentication token is passed via environment variables
  - accessible with a simple `condor_q -af environment`!
- Back-end communication (Hub <-> notebook) in plain HTTP
  - no cryptography, the official batchspawner code doesn't support SSL
- Direct interactive access to worker nodes can favour privilege escalation due to slowly patched bugs

Securing the authentication token

- submit wrapper override

```
c.CondorSpawner.batch_submit_cmd = '/etc/jupyterhub/submit_wrapper'
```

- remove GetEnv=true from the submit file

- Write the needed environment variables to a file to be spooled

```
#!/bin/sh

cd ~/

umask 0077


cat > env <<EOF
JUPYTERHUB_SERVER_NAME=$JUPYTERHUB_SERVER_NAME
JUPYTERHUB_SERVICE_PREFIX=$JUPYTERHUB_SERVICE_PREFIX
JUPYTERHUB_USER=$JUPYTERHUB_USER
JUPYTERHUB_API_TOKEN=$JUPYTERHUB_API_TOKEN
JUPYTERHUB_OAUTH_CALLBACK_URL=$JUPYTERHUB_OAUTH_CALLBACK_URL
JUPYTERHUB_BASE_URL=$JUPYTERHUB_BASE_URL
JUPYTERHUB_API_URL=$JUPYTERHUB_API_URL
JPY_API_TOKEN=$JPY_API_TOKEN
JUPYTERHUB_ACTIVITY_URL=$JUPYTERHUB_ACTIVITY_URL
JUPYTERHUB_HOST=$JUPYTERHUB_HOST
JUPYTERHUB_CLIENT_ID=$JUPYTERHUB_CLIENT_ID
JUPYTERHUB_ADMIN_ACCESS=$JUPYTERHUB_ADMIN_ACCESS
JUPYTERHUB_SSL_CLIENT_CA=/etc/jupyter/ca.pem
JUPYTERHUB_SSL_CERTFILE=/etc/jupyter/notebook.pem
JUPYTERHUB_SSL_KEYFILE=/etc/jupyter/notebook.key
EOF

condor_submit -spool -batch-name jupyter-$USER
```

Modified submit file

- Override the stock submit file in config.py

```
with open("/etc/jupyterhub/submit_file") as submit_file:
    c.CondorSpawner.batch_script = submit_file.read()
```

- The new submit file

```
Executable = /bin/sh
Transfer_executable = False
Arguments = "-c '/usr/share/htc/jupyter/notebook.sh {cmd}'"
RequestMemory = 3500
RequestCpus = 1
Output = jupyterhub.condor.out
Error = jupyterhub.condor.err
transfer_input_files = env,notebook.pem,notebook.key,ca.pem,jupyter_notebook_config.py
encrypt_input_files = *
transfer_output_files = jupyter_notebook_config.py
periodic_remove = JobStatus == 2 && time() - EnteredCurrentStatus > 259200
{options}
Queue
```

Securing the back-end communication

- JupyterHub has all the facilities to secure the back-end communications
- Enable "internal" SSL in config.py

```
c.JupyterHub.internal_ssl = True
```

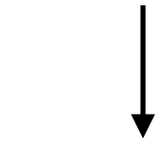
- JupyterHub handles its own set of CAs
- The NotebookCA issues RSA keys and x509 certificates to each user
- Key-cert user's pair is used by its notebook to provide TLS communication

Securing the back-end communication

https://github.com/carmelopellegrino/batchspawner

- The official batchspawner doesn't support internal SSL!
- Forked from the official GitHub repo
- Implements the `move_certs()` method of the `batchspawner.CondorSpawner` class that copies cert+key to a user-readable folder

- cert+key copied on the wn via spool



```
transfer_input_files
=notebook.pem,notebook.key,ca.pem,...
encrypt_input_files = *
```

```
922  924
     925  +    async def move_certs(self, paths):
     926  +        """Make a copy of the SSL certificates and key JupyterHub creates for
     927  +        each user when internal_ssl=True"""
     928  +        mask = os.umask()
     929  +        try:
     930  +            username = self.user.name
     931  +            entry = pwd.getpwnam(username)
     932  +
     933  +            home = entry.pw_dir
     934  +            uid = entry.pw_uid
     935  +            gid = entry.pw_gid
     936  +
     937  +            nb_paths = {
     938  +                    'cafile': home + '/ca.pem',
     939  +                    'keyfile': home + '/notebook.key',
     940  +                    'certfile': home + '/notebook.pem'
```

Securing the wn environment

- Run the notebook in a **singularity** container
- Isolation of the wn environment from the notebook one
- singularity image built in **continuous integration** every day
  - latest security updates are applied often
- Max job duration set to **72 hours** via HTCondor `periodic_remove` submit-file command

Singularity recipe

```
BootStrap: docker
From: centos:7

%help
Helper image to run jupyter-notebook in HTCondor jobs

%labels
AUTHOR Carmelo Pellegrino <carmelo.pellegrino@cnaf.infn.it>
VERSION 1.0
```

```
%post
yum update  -y                                                  && \
yum upgrade -y                                                  && \
yum install -y epel-release                                     && \
yum install -y conda vim nano git subversion htop less          && \
yum clean all
source /etc/profile.d/conda.sh
conda create -y --prefix /jupyter python=3.9.7
conda activate /jupyter
conda install -y jupyter jupyterlab
pip install batchspawner==1.1.0

%runscript
source /etc/profile.d/conda.sh
conda activate /jupyter
exec "$@"
```

Securing the Hub

- Fail2ban

```
# cat /etc/fail2ban/filter.d/jupyterhub.conf
[Definition]
failregex = PAM Authentication failed \(.*@<HOST>\)
            404 (GET|HEAD|POST|PUT|DELETE) .* \(@<HOST>\) .*ms
ignoreregex = ''
```

- HTTPS in config.py

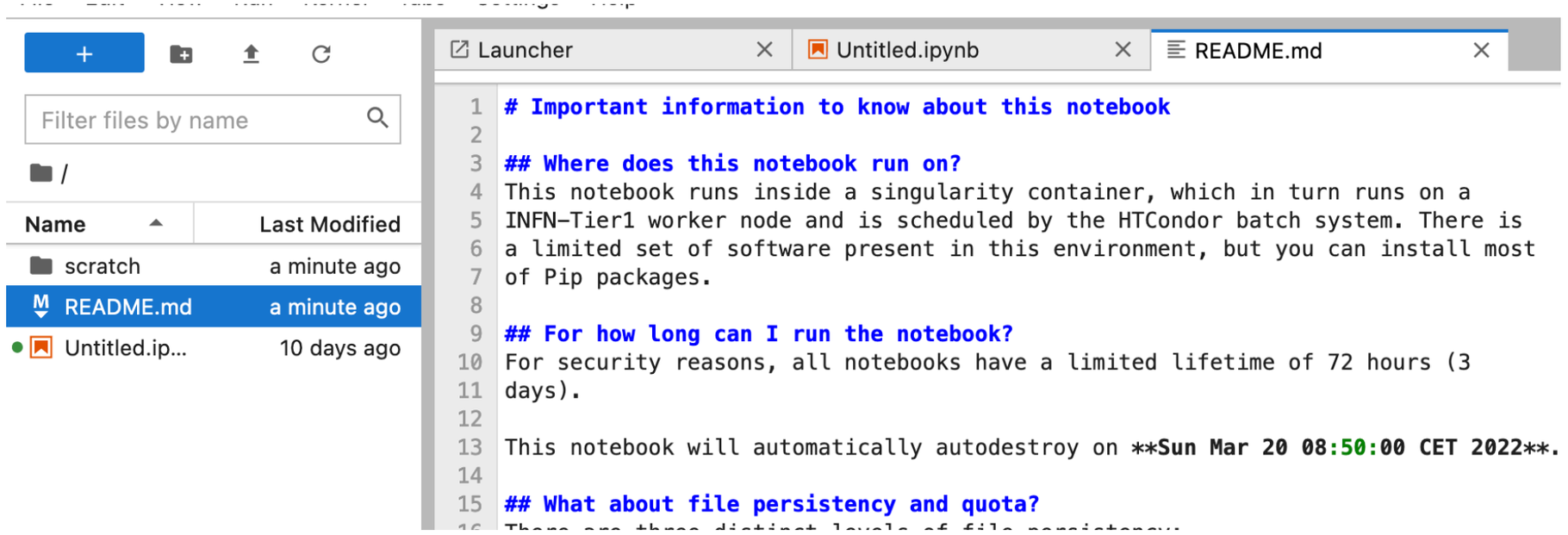
```
c.JupyterHub.port = 443
c.JupyterHub.ssl_cert = '/etc/jupyterhub/jupyterhub_cert.pem'
c.JupyterHub.ssl_key = '/etc/jupyterhub/jupyterhub_key.pem'
```

- Firewall rules
  - allow SSH only from a restricted set of IPs
  - allow back-end connection only from the dedicated worker nodes
  - DROP everything else

Additional features

Readme file

- Each new job receives a custom readme.md file
- basic information for the user
- useful links
- time and date of job termination

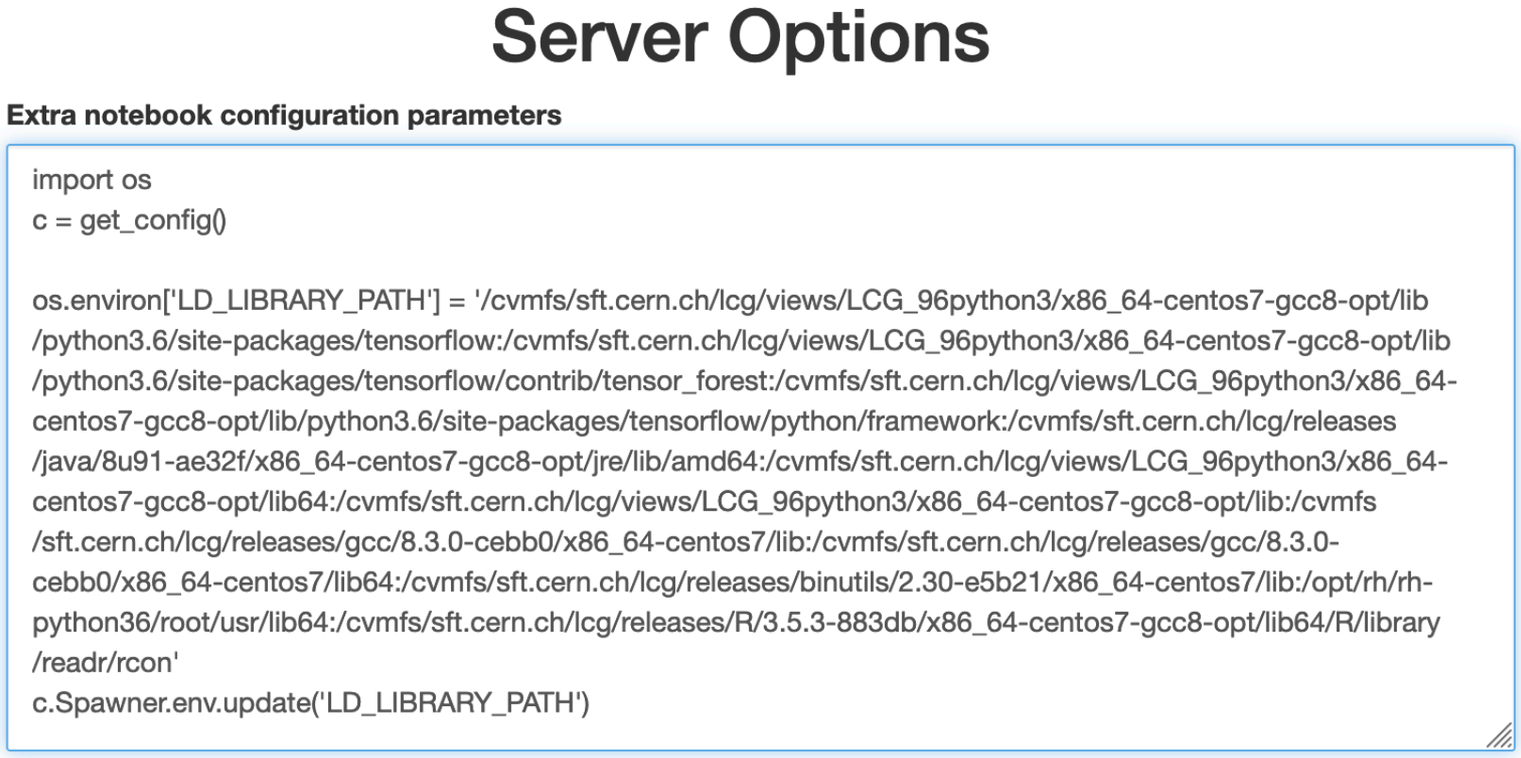


User-defined configuration file

- Editable by each user via web interface before the job is submitted by the Hub
- Copied at every job start via the spool mechanism
- Visible at ~/.jupyter/jupyter_notebook_config.py
- Useful to load an alternative version of Python from a different source (e.g.: cvmfs)

User-defined configuration file

- E.g.: to make sure Python 3.6 and its libraries are correctly loaded from CVMFS:



Data and session persistency

- Three levels of data persistency:
  - **scratch**: all data contained in this folder are to be considered absolutely **ephemeral**. This means that all files that are left in this directory will be **destroyed** at the end of this job. A quota of less than 2TB is shared among all notebooks running on the same worker node
  - **home**: all data contained in the home folder are kept between subsequent notebook runs via singularity **overlayfs** files. This storage is intended for notebook files and installation of small packages via *pip* or *conda*. A hard quota is set to 512MB
  - **/storage/gpfs_***: these are the usual CNAF disk folders, shared with user interfaces and worker nodes. Quotas are applied on a per-experiment basis and persistency is completely demanded to the experiment

Executable = notebook.sh

- What is actually run as executable:

storage
SSL
image

```
singularity run                                                    \
  --overlay "/storage/gpfs_[...]/$USERNAME.img"                    \
  --bind scratch/:$HOMEDIR/scratch/:rw                             \
  --bind README.md:$HOMEDIR/README.md:rw                           \
  --bind jupyter_notebook_config.py:$HOMEDIR/.jupyter/jupyter_notebook_config.py \
  --bind /cvmfs:/cvmfs:ro                                          \
  --bind /storage:/storage:rw                                      \
  --bind /opt/exp_software:/opt/exp_software:ro                    \
  --bind ca.pem:/etc/jupyter/ca.pem:ro                             \
  --bind notebook.pem:/etc/jupyter/notebook.pem:ro                 \
  --bind notebook.key:/etc/jupyter/notebook.key:ro                 \
  --env-file env --no-home                                         \
  jn.img "$@"                                                \
  --keyfile=/etc/jupyter/notebook.key                              \
  --certfile=/etc/jupyter/notebook.pem                             \
```

Conclusions

Conclusions

- Jupyter Notebooks are an established technology that support flexible and solid computing workflows
- JupyterHub@INFN-Tier1 allows notebooks and interactive access to execute codes on servers identical of those where batch jobs run
- Notebooks are accounted as batch jobs
- Security is enforced

Thank you

Backup

Authenticators

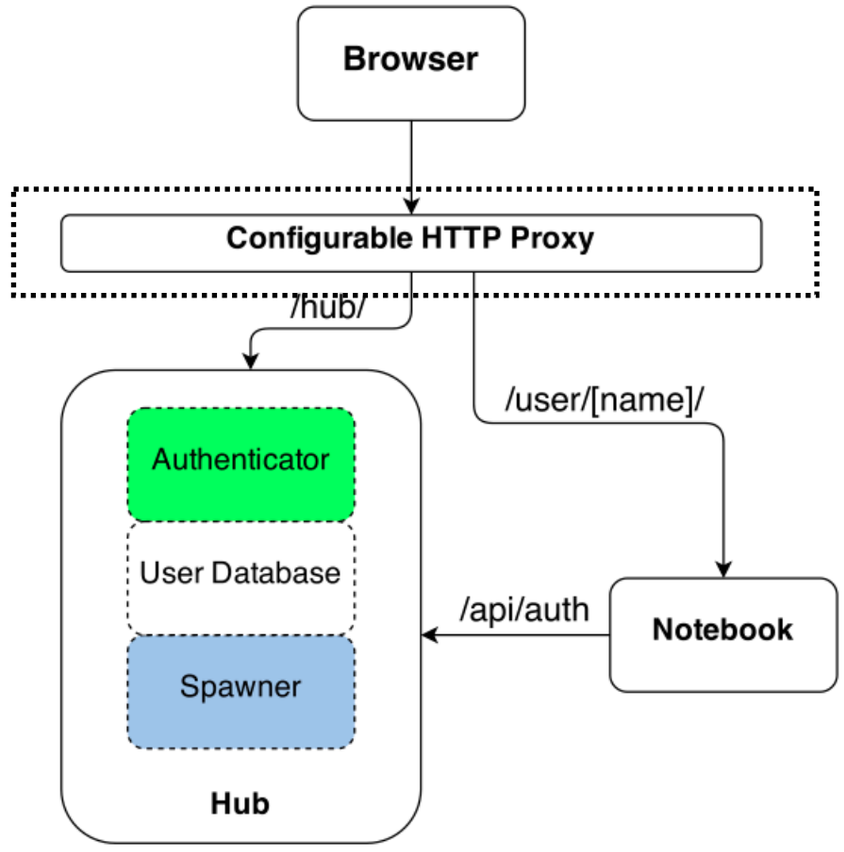
https://jupyterhub.readthedocs.io/en/stable/reference/authenticators.html

- PAMAuthenticator is the default
  - based on the PAM modules installed on the Hub
  - authentication can be delegated to any PAM authenticator
  - LDAP/Kerberos
  - password
  - etc...
- OAuthenticator supports GitHub the OAuth provider

Network basics

https://jupyterhub.readthedocs.io/en/stable/getting-started/networking-basics.html

- The central network element is the configurable-http-proxy
- It is a HTTP proxy written in JavaScript (nodejs) with REST API for configuration
- Dispatches messages back and forth the Hub and the Notebooks to the users' browsers
- Exposes two network ports:
  - [::]:443 if SSL else 80 (towards users)
  - localhost:8001 (configuration API)

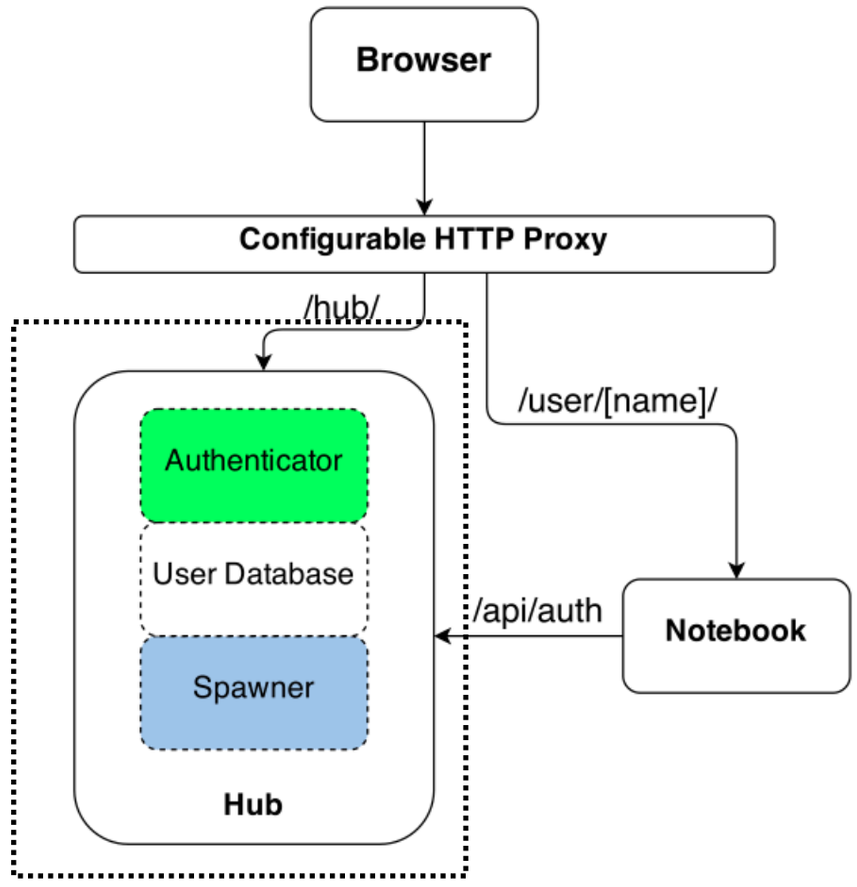


credits: https://jupyterhub.readthedocs.io/en/0.9.1/

Network basics

https://jupyterhub.readthedocs.io/en/stable/getting-started/networking-basics.html

- The Hub exposes the [::]:8081 port
- Used by both the Proxy and the notebooks
- By default, the Hub and the Proxy run on the same host

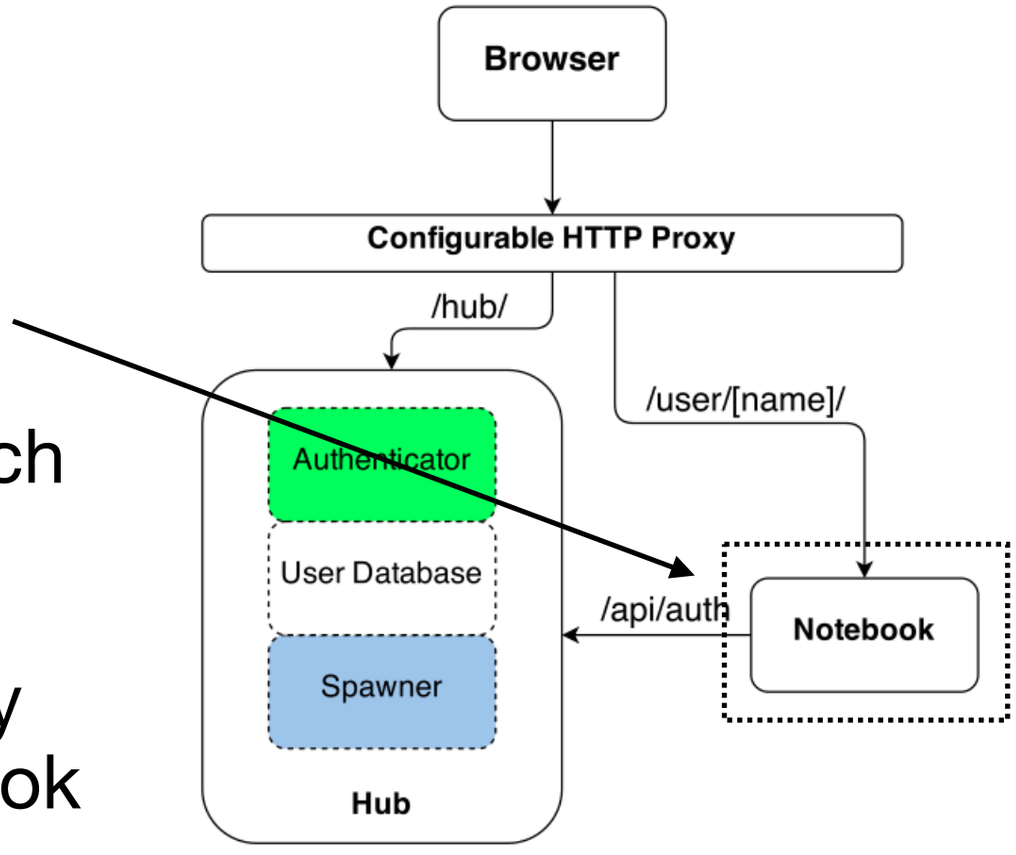


credits: https://jupyterhub.readthedocs.io/en/0.9.1/

Network basics

https://jupyterhub.readthedocs.io/en/stable/getting-started/networking-basics.html

- The notebooks can run on remote hosts
- Expose a randomly generated port
- Each notebook tells to the Hub which port it listens on
- In turn the Hub configures the Proxy to forward user traffic to the notebook



credits: https://jupyterhub.readthedocs.io/en/0.9.1/