Jupyter on CINECA HPC

January 30, 2026

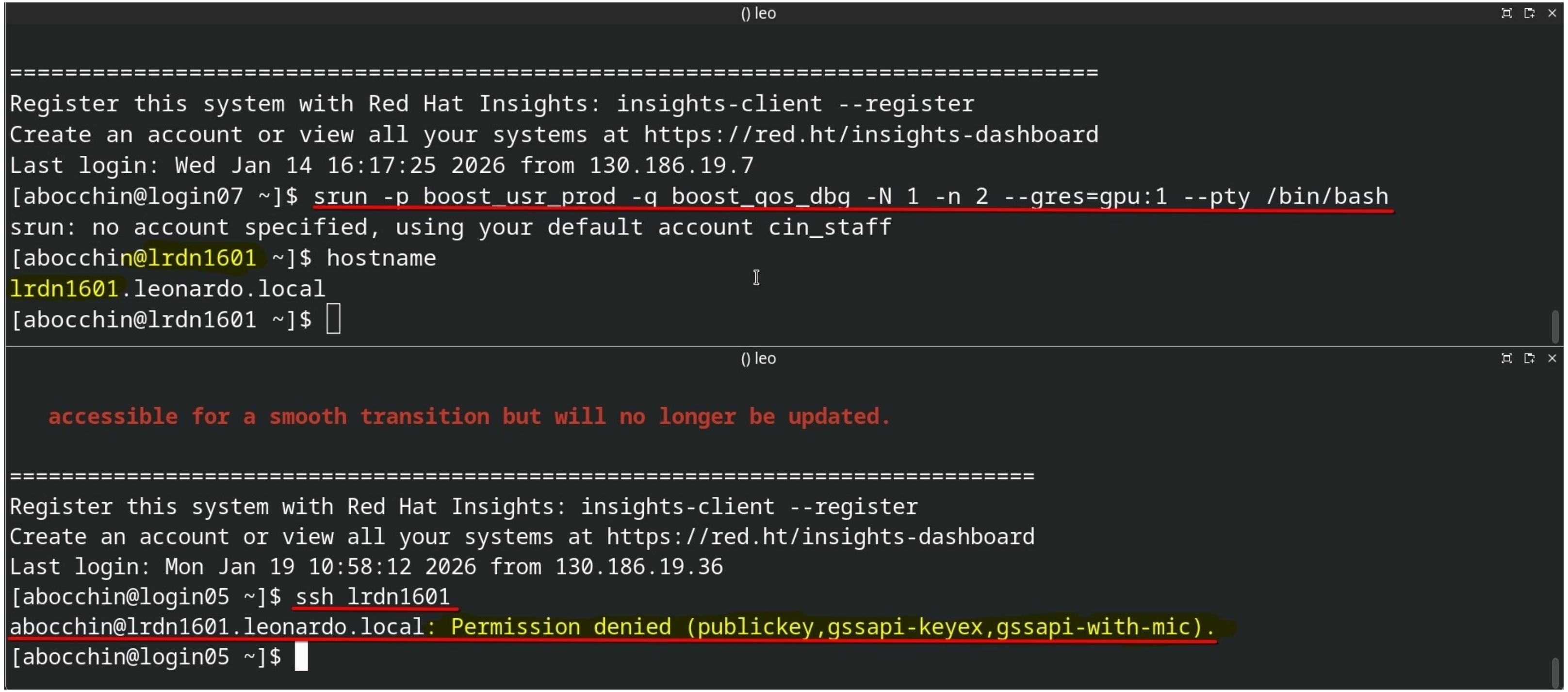
Alberto Bocchinfuso
a.bocchinfuso@cineca.it

Outline

- ➢ Pre-requirement → enable password-less ssh between nodes
- ➢ Pre-requirement → create a python virtual environment
- ➢ Launch a slurm Job → load a venv and launch a Jupyter Kernel
- ➢ Connect to the Kernel → ssh double tunnel
- ➢ Connect to Jupyter notebook → web browser on local workstation
- ➢ Additional Cineca resources → Chappyner & VS code template

ssh on Leonardo

- **Users CANNOT ssh to compute nodes, UNLESS THEY HAVE AN ACTIVE JOB ON THE NODES**
- **Experiment:** start a job, then try to conneto to the compute node from another shell.



() leo

```
================================================================================
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Wed Jan 14 16:17:25 2026 from 130.186.19.7
[abocchin@login07 ~]$ srun -p boost_usr_prod -q boost_qos_dbg -N 1 -n 2 --gres=gpu:1 --pty /bin/bash
srun: no account specified, using your default account cin_staff
[abocchin@lrdn1601 ~]$ hostname
lrdn1601.leonardo.local
[abocchin@lrdn1601 ~]$ 
```

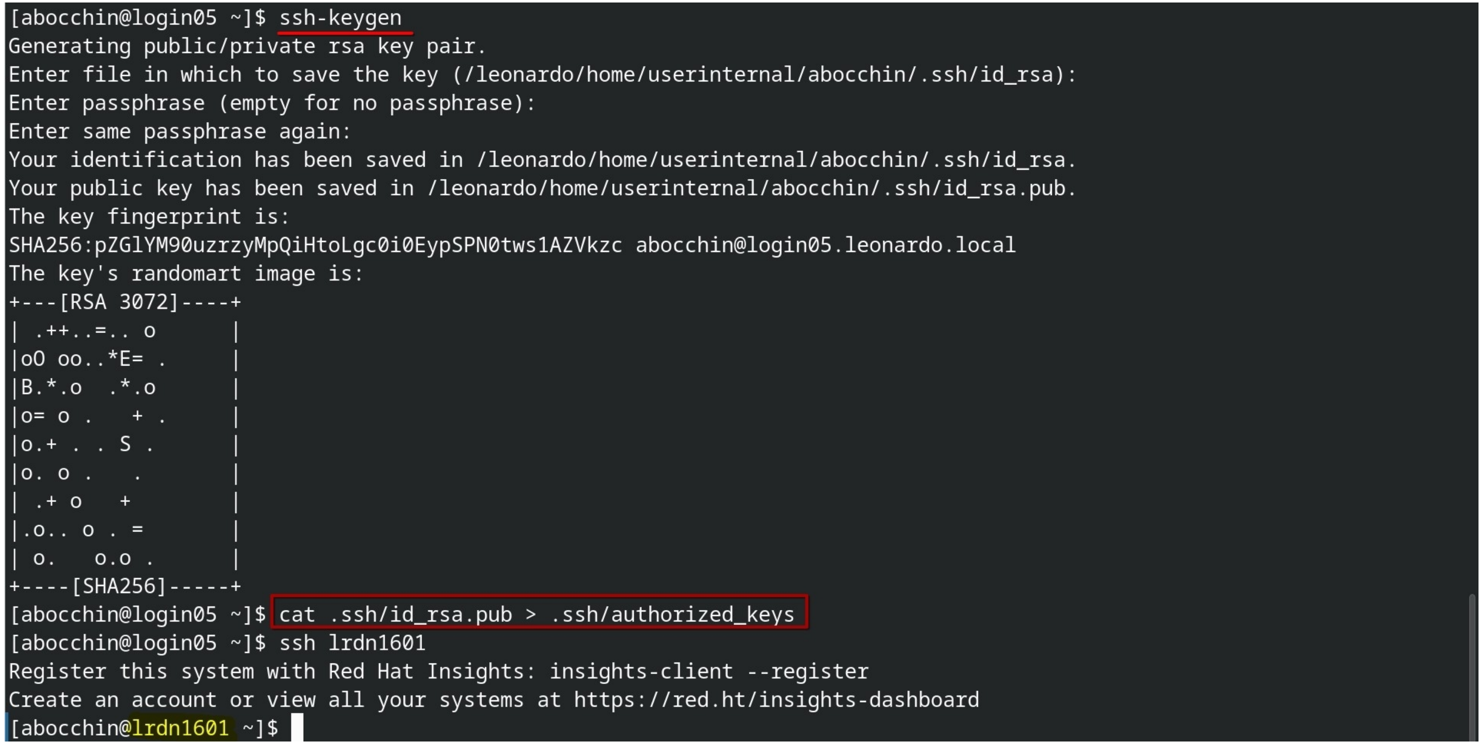
() leo

```
   accessible for a smooth transition but will no longer be updated.

================================================================================
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Mon Jan 19 10:58:12 2026 from 130.186.19.36
[abocchin@login05 ~]$ ssh lrdn1601
abocchin@lrdn1601.leonardo.local: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[abocchin@login05 ~]$ 
```

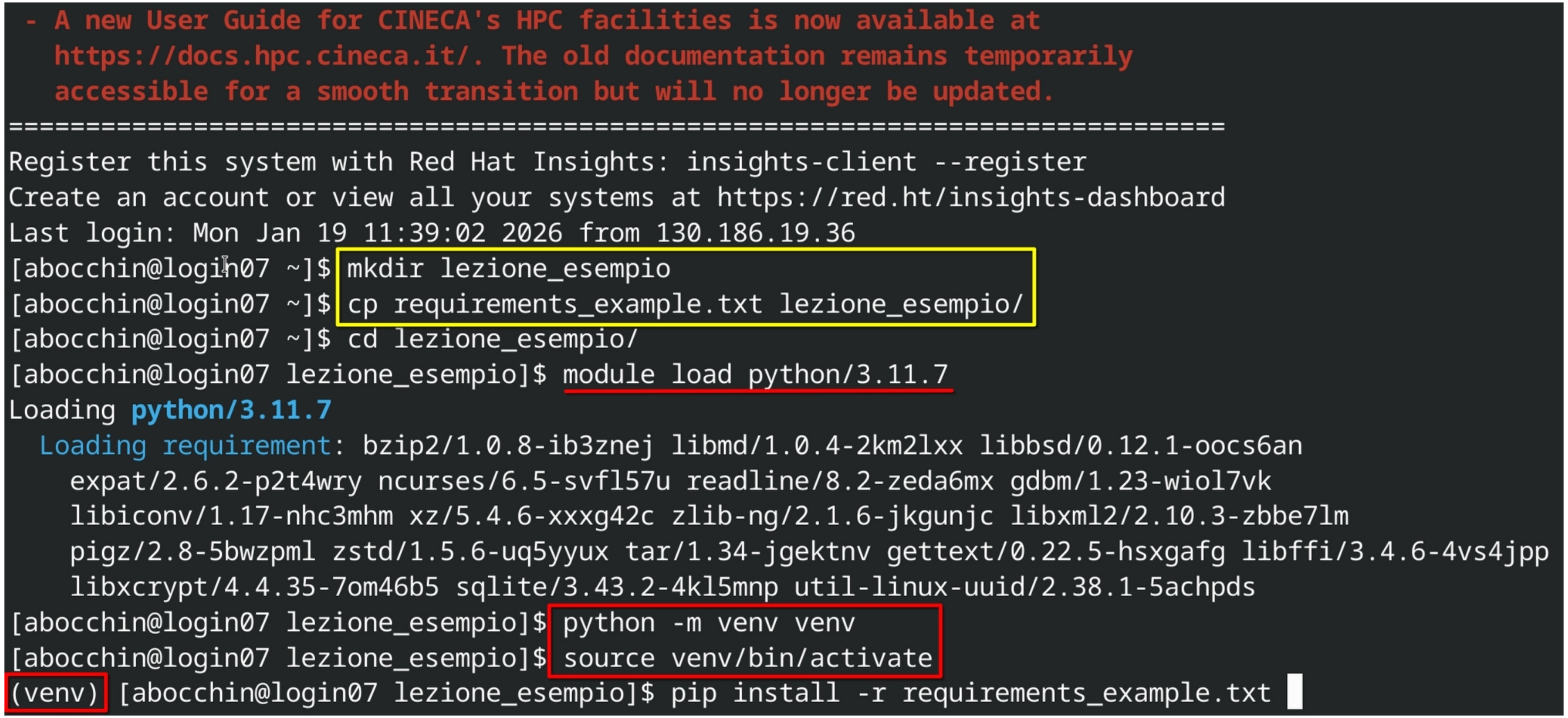
ssh on Leonardo

- ➢ **Users CANNOT ssh to compute nodes, UNLESS THEY HAVE AN ACTIVE JOB ON THE NODES**
- ➢ **Create and exchange a key, then try again.**



```
[abocchin@login05 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/leonardo/home/userinternal/abocchin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /leonardo/home/userinternal/abocchin/.ssh/id_rsa.
Your public key has been saved in /leonardo/home/userinternal/abocchin/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:pZGlYM90uzrzyMpQiHtoLgc0i0EypSPN0tws1AZVkzc abocchin@login05.leonardo.local
The key's randomart image is:
+---[RSA 3072]----+
| .++..=.. o      |
|oO oo..*E= .     |
|B.*.o  .*.o      |
|o= o .   + .     |
|o.+ . . S .      |
|o. o .   .       |
| .+ o   +        |
|.o.. o . =       |
| o.   o.o .      |
+----[SHA256]-----+
[abocchin@login05 ~]$ cat .ssh/id_rsa.pub > .ssh/authorized_keys
[abocchin@login05 ~]$ ssh lrdn1601
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
[abocchin@lrdn1601 ~]$
```

Creating a venv



```
 - A new User Guide for CINECA's HPC facilities is now available at
   https://docs.hpc.cineca.it/. The old documentation remains temporarily
   accessible for a smooth transition but will no longer be updated.
===============================================================================
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Mon Jan 19 11:39:02 2026 from 130.186.19.36
[abocchin@login07 ~]$ mkdir lezione_esempio
[abocchin@login07 ~]$ cp requirements_example.txt lezione_esempio/
[abocchin@login07 ~]$ cd lezione_esempio/
[abocchin@login07 lezione_esempio]$ module load python/3.11.7
Loading python/3.11.7
  Loading requirement: bzip2/1.0.8-ib3znej libmd/1.0.4-2km2lxx libbsd/0.12.1-oocs6an
    expat/2.6.2-p2t4wry ncurses/6.5-svfl57u readline/8.2-zeda6mx gdbm/1.23-wiol7vk
    libiconv/1.17-nhc3mhm xz/5.4.6-xxxg42c zlib-ng/2.1.6-jkgunjc libxml2/2.10.3-zbbe7lm
    pigz/2.8-5bwzpml zstd/1.5.6-uq5yyux tar/1.34-jgektnv gettext/0.22.5-hsxgafg libffi/3.4.6-4vs4jpp
    libxcrypt/4.4.35-7om46b5 sqlite/3.43.2-4kl5mnp util-linux-uuid/2.38.1-5achpds
[abocchin@login07 lezione_esempio]$ python -m venv venv
[abocchin@login07 lezione_esempio]$ source venv/bin/activate
(venv) [abocchin@login07 lezione_esempio]$ pip install -r requirements_example.txt
```

pip will install all requirements listed; slow, depending on the venv requirements.

Submit the SLURM job

```
#!/bin/bash
#SBATCH --time=00:30:00
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=8
#SBATCH --gres=gpu:1
#SBATCH --partition=boost_usr_prod
#SBATCH --qos=boost_qos_dbg
#SBATCH --account=tra25_sumsch
source venv/bin/activate
worker_list=($(scontrol show hostnames "$SLURM_JOB_NODELIST"))
head_node=${worker_list[0]}
jupyter_port=$(($RANDOM%(64511-50000+1)+50000))
jupyter_token=${USER}_${jupyter_port}
echo “ssh -L ${jupyter_port}:${head_node}:${jupyter_port} ${USER}@login.leonardo.cineca.it -N”
echo "http://127.0.0.1:${jupyter_port}/lab?token=${jupyter_token}"
srun jupyter lab --ip=0.0.0.0 --port=${jupyter_port} --NotebookApp.token=${jupyter_token}
```

Connect to the remote Jupyter

- ➢ ssh tunnel: **workstation → Leonardo login**
  ssh tunnel: **Leonardo login → $head_node**
- ➢ For simplicity, select always the same port.

```
$ ssh -L $jupyter_port:$head_node:$jupyter_port <userid>@login.leonardo.cineca.it -N
```

- ➢ Now, connect through the browser

http://127.0.0.1:${jupyter_port}/lab?token=${jupyter_token}

NOTE: the script on the previous page echoes the command for ssh and the http address with the values of each variables. Please, read the slurm-JOBID.out

IMPORTANT: DO NOT send the ssh tunnel in background.

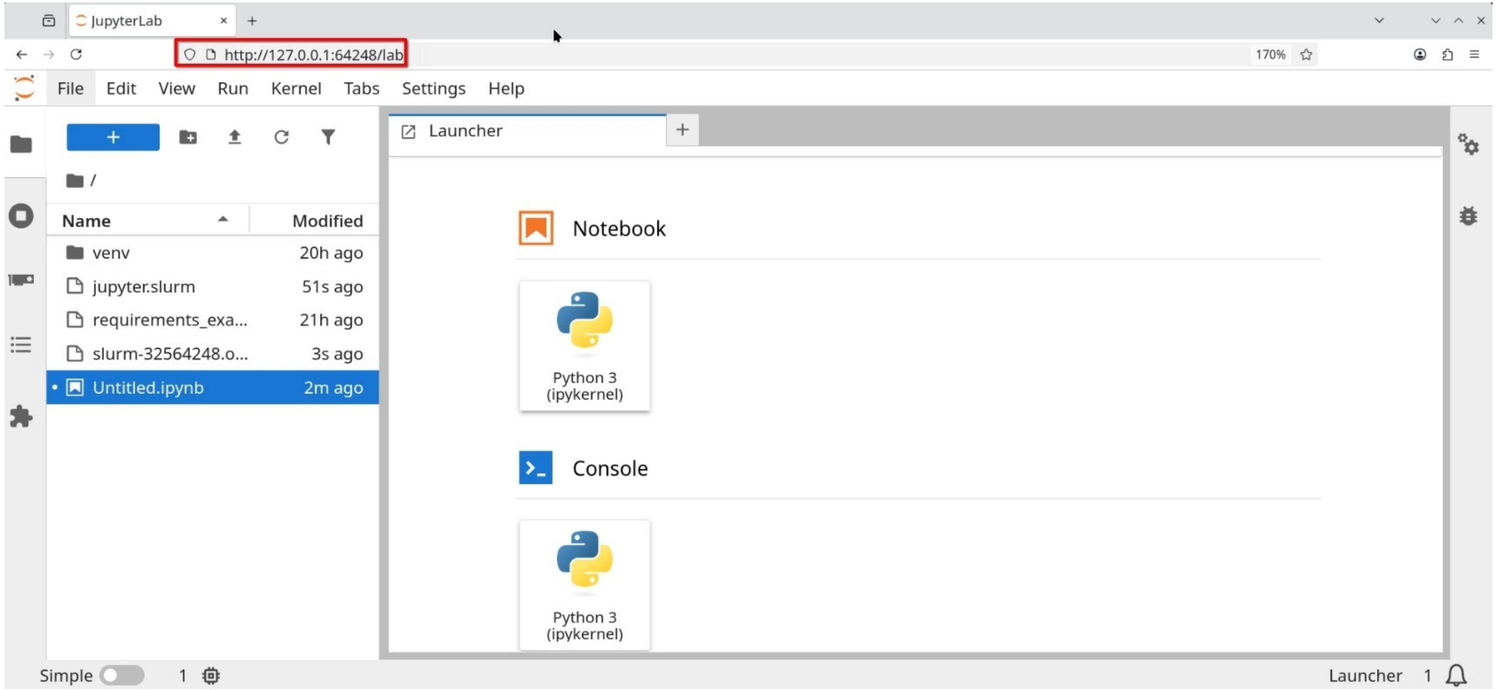
Leave it in the foreground of an open terminal and don’t forget to

close the tunnel after you’ve done.

Connect to the remote Jupyter

```
$ ssh -L $jupyter_port:$head_node:$jupyter_port <userid>@login.leonardo.cineca.it -N
```

http://127.0.0.1:${jupyter_port}/lab?token=${jupyter_token}



Name	Modified
venv	20h ago
jupyter.slurm	51s ago
requirements_exa...	21h ago
slurm-32564248.o...	3s ago
Untitled.ipynb	2m ago

Outline

- ➢ Pre-requirement → enable password-less ssh between nodes
- ➢ Pre-requirement → create a python virtual environment
- ➢ Launch a slurm Job → load a venv and launch a Jupyter Kernel
- ➢ Connect to the Kernel → ssh double tunnel
- ➢ Connect to Jupyter notebook → web browser on local workstation
- ➢ Additional Cineca resources → Chappyner & VS code template

Chappyner

- **Chappyner is a cineca tool by Fabio Pitari.** The source code is available at:

  https://gitlab.hpc.cineca.it/interactive_computing/chappyner

- Chappyner is able to open and manage Jupyter sessions through ssh
  the tool automates everything explained in the previous slides.

- Chappyner can be installed in a python virtual environment with pip

  ```
  pip install chappyner
  ```

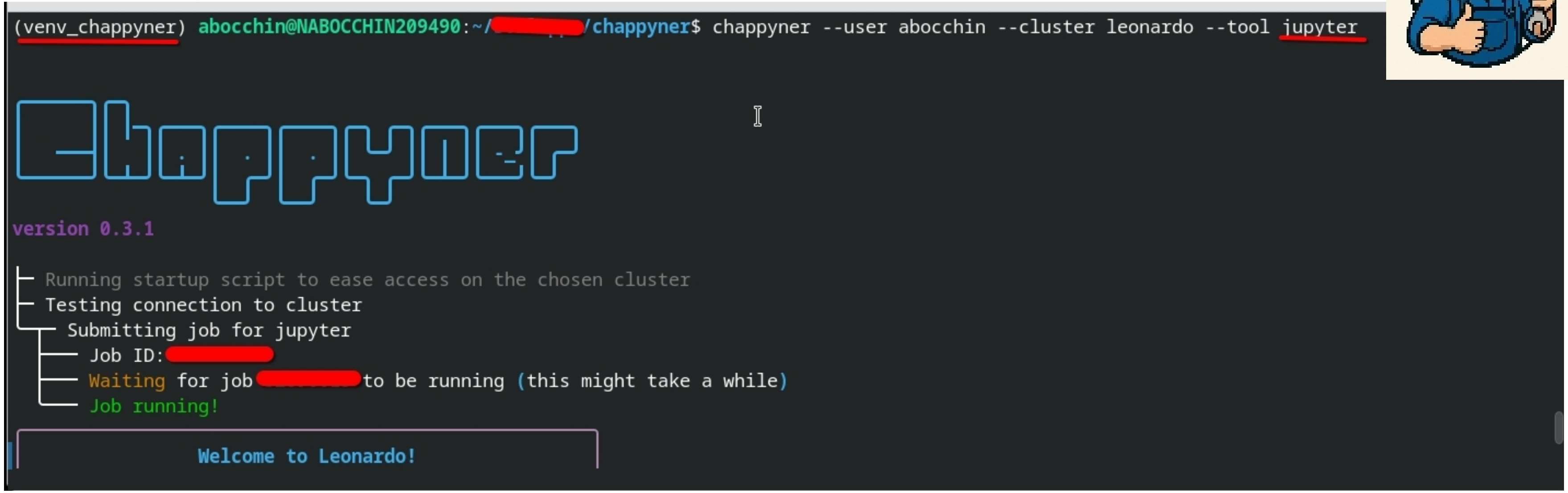
- Chappyner gitlab repo offfers **extensive documentation** in the form of README.md files

- Jupyter sessions on leonardo can be started with the command

  ```
  chappyner --user <my user name on leonardo> --cluster leonardo --tool jupyter
  ```

- **Users are encouraged to check the documentation for the other tools available.**

Chappyner

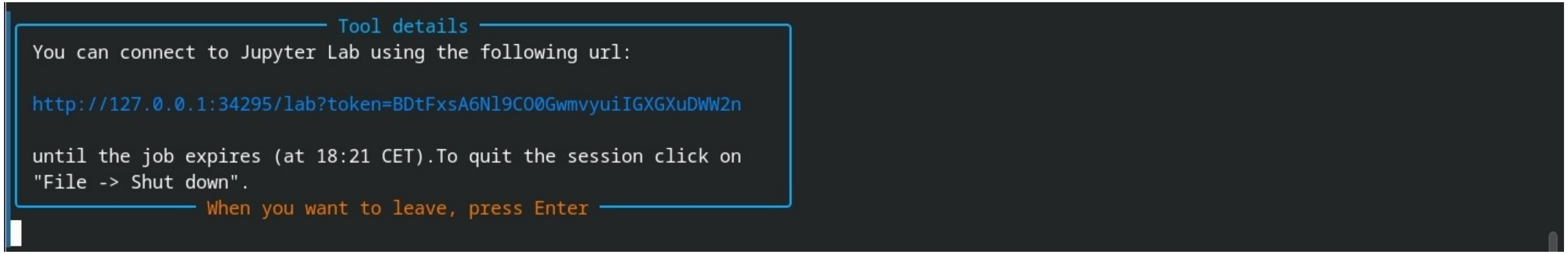


```
(venv_chappyner) abocchin@NABOCCHIN209490:~/[redacted]/chappyner$ chappyner --user abocchin --cluster leonardo --tool jupyter

version 0.3.1

├─ Running startup script to ease access on the chosen cluster
├─ Testing connection to cluster
└──── Submitting job for jupyter
   ├──── Job ID: [redacted]
   ├──── Waiting for job [redacted] to be running (this might take a while)
   └──── Job running!

           Welcome to Leonardo!
```



```
──────────────── Tool details ────────────────
 You can connect to Jupyter Lab using the following url:

 http://127.0.0.1:34295/lab?token=BDtFxsA6N19C00GwmvyuiIGXGXuDWW2n

 until the job expires (at 18:21 CET).To quit the session click on
 "File -> Shut down".
──────── When you want to leave, press Enter ────────
```

A template for VS code

- **Visual Studio code users can rely on the template to debug python code on the cluster, by Alberto Bocchinfuso and Attilio Marcelli**

  https://gitlab.hpc.cineca.it/HpcUserSupport/py-compute-debug

- The user must log into the machine with VS code via Remote -SSH, **then the tool automates the procedure needed to debug code** (out of the scope of this lecture) **or launch a Jupyter Kernel.**

- Similarly to Chappyner, the user will get a link to connect to Jupyter.

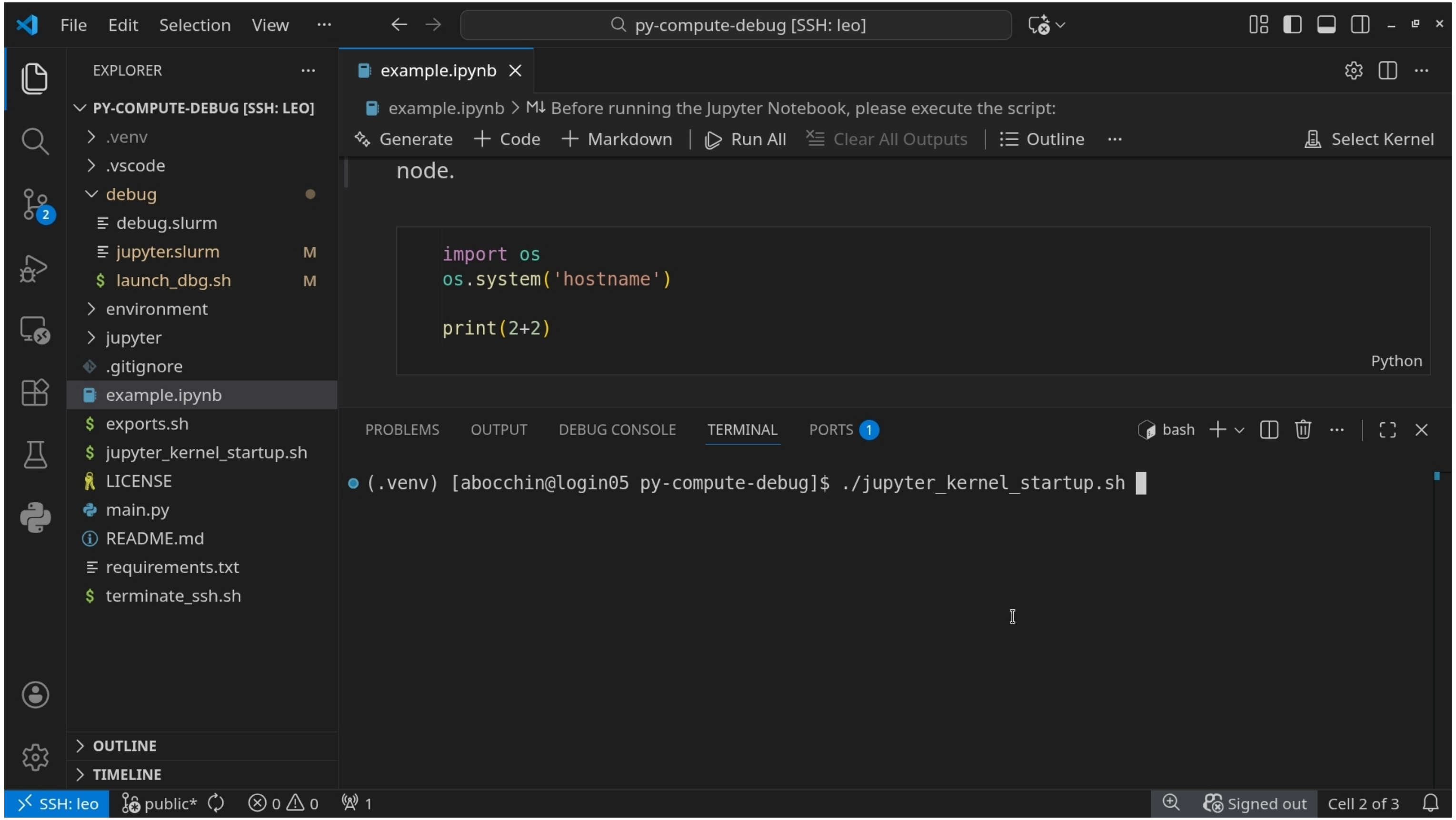
- Unlike Chappyner, the user will **not use an external web browser**; the Jupyter Notebook extension of VS code will be connected to the running kernel.

- The documentation of the template can be found at

  https://gitlab.hpc.cineca.it/HpcUserSupport/py-compute-debug/-/wikis/home

- **Users are encouraged to check the documentation to learn about both debugging and Jupyter**

A template for VS code



node.

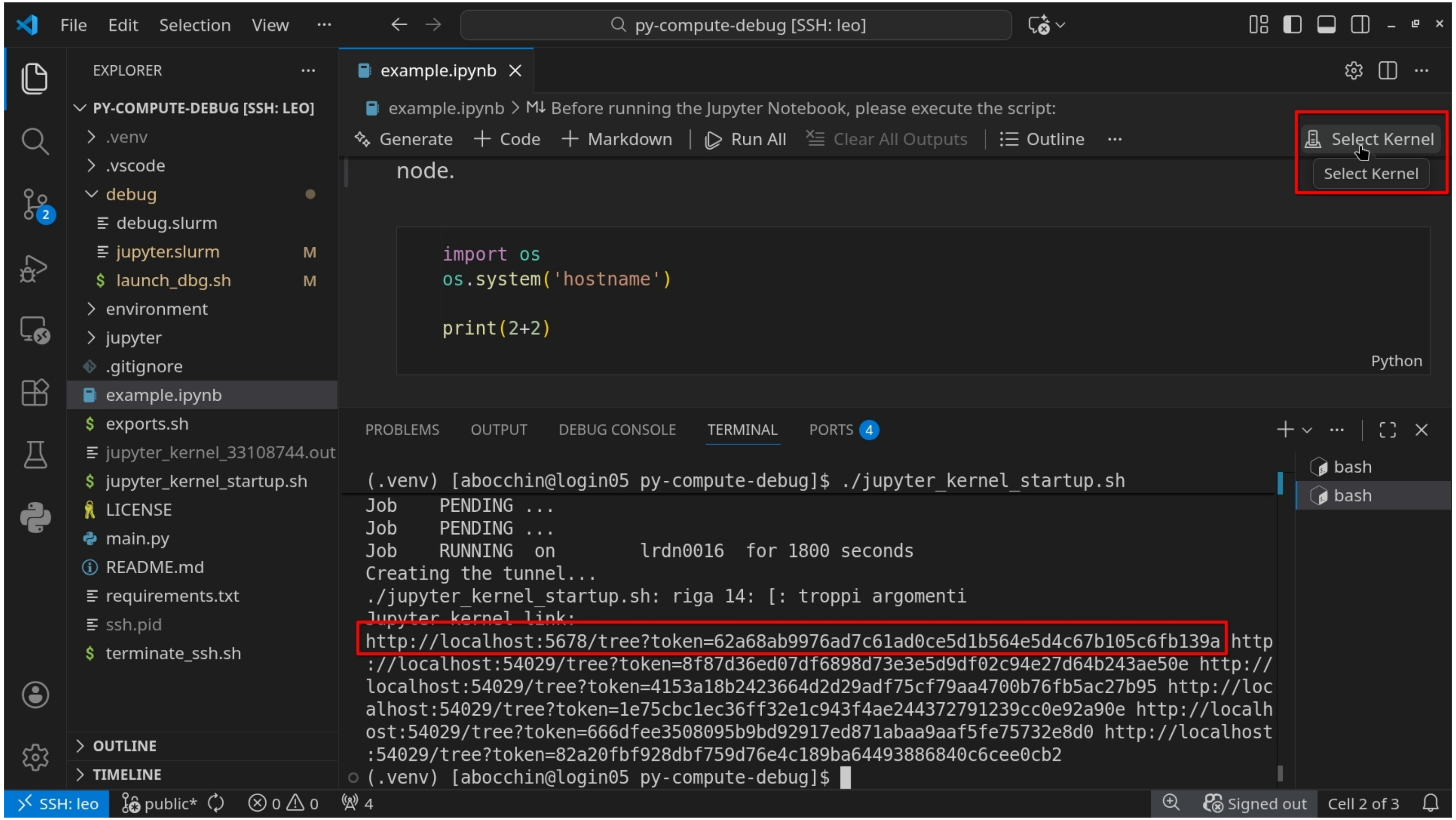
```python
import os
os.system('hostname')

print(2+2)
```

Python

```
(.venv) [abocchin@login05 py-compute-debug]$ ./jupyter_kernel_startup.sh
```

A template for VS code

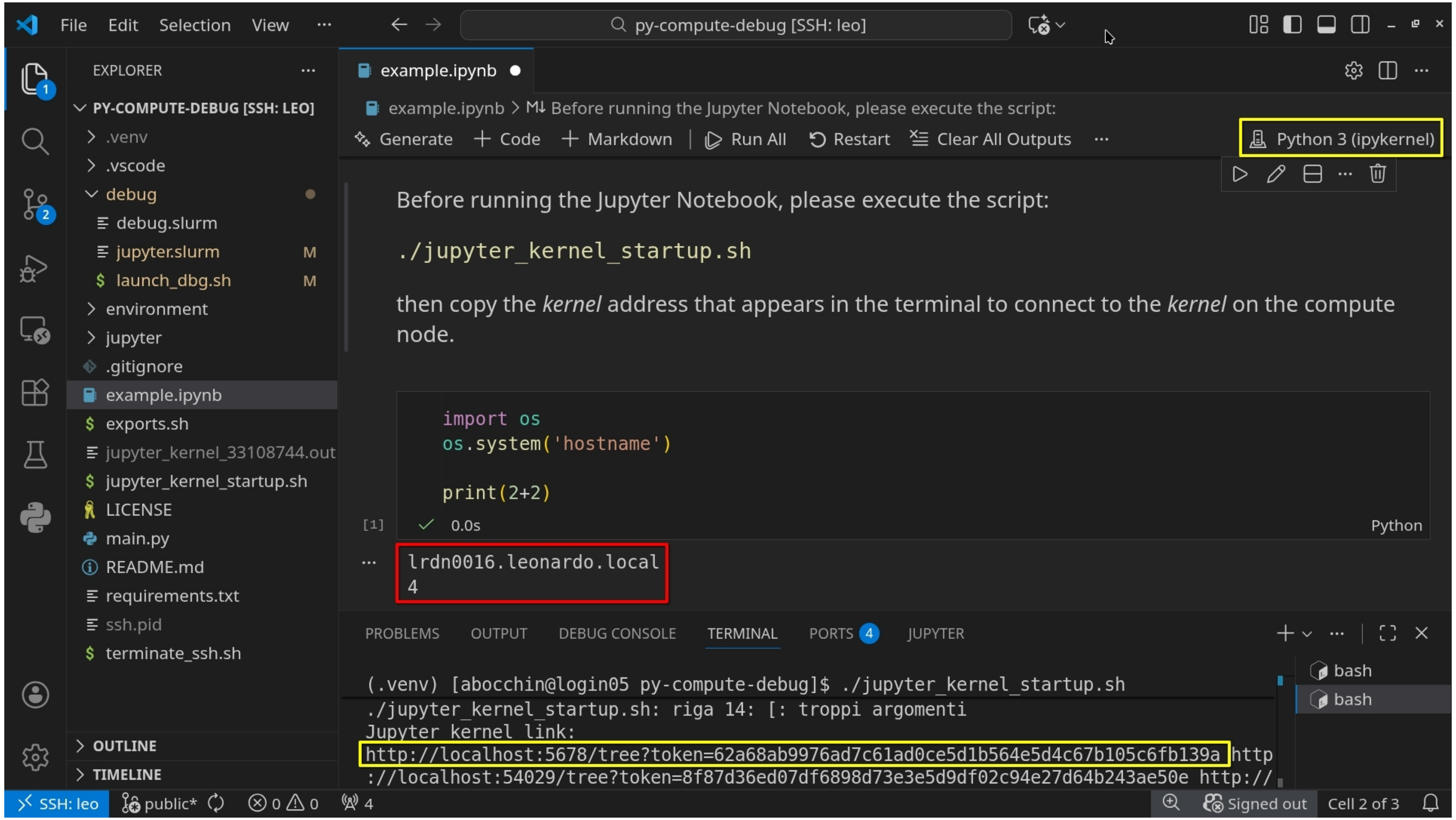


```python
import os
os.system('hostname')

print(2+2)
```

```
(.venv) [abocchin@login05 py-compute-debug]$ ./jupyter_kernel_startup.sh
Job    PENDING ...
Job    PENDING ...
Job    RUNNING  on        lrdn0016  for 1800 seconds
Creating the tunnel...
./jupyter_kernel_startup.sh: riga 14: [: troppi argomenti
Jupyter kernel link:
http://localhost:5678/tree?token=62a68ab9976ad7c61ad0ce5d1b564e5d4c67b105c6fb139a http
://localhost:54029/tree?token=8f87d36ed07df6898d73e3e5d9df02c94e27d64b243ae50e http://
localhost:54029/tree?token=4153a18b2423664d2d29adf75cf79aa4700b76fb5ac27b95 http://loc
alhost:54029/tree?token=1e75cbc1ec36ff32e1c943f4ae244372791239cc0e92a90e http://localh
ost:54029/tree?token=666dfee3508095b9bd92917ed871abaa9aaf5fe75732e8d0 http://localhost
:54029/tree?token=82a20fbf928dbf759d76e4c189ba64493886840c6cee0cb2
(.venv) [abocchin@login05 py-compute-debug]$
```

A template for VS code



example.ipynb > Before running the Jupyter Notebook, please execute the script:

Before running the Jupyter Notebook, please execute the script:

```
./jupyter_kernel_startup.sh
```

then copy the *kernel* address that appears in the terminal to connect to the *kernel* on the compute node.

```python
import os
os.system('hostname')

print(2+2)
```

[1] ✓ 0.0s Python

```
lrdn0016.leonardo.local
4
```

```
(.venv) [abocchin@login05 py-compute-debug]$ ./jupyter_kernel_startup.sh
./jupyter_kernel_startup.sh: riga 14: [: troppi argomenti
Jupyter kernel link:
http://localhost:5678/tree?token=62a68ab9976ad7c61ad0ce5d1b564e5d4c67b105c6fb139a http
://localhost:54029/tree?token=8f87d36ed07df6898d73e3e5d9df02c94e27d64b243ae50e http://
```

THANK YOU FOR YOUR ATTENTION

https://docs.hpc.cineca.it/

Write to superc@cineca.it in case of need!