Port forwarding for jupyter notebooks

> author: Katie Chamberlain
> date: 03 Nov 2022

---

0. Prelude

Make sure you have a functioning ssh config file on your local machine!

1. Setting up a `conda` installation on HPC

▼ more details can be found:

https://public.confluence.arizona.edu/display/UAHPC/Using+and+Installing+Python

1. Start a new interactive session on the hpc. This will put you on a compute node where you can run commands like `conda` etc.

   ```
   interactive -a gbesla -t 04:00:00 -Q user_qos_gbesla
   ```

   > Note: the generic command to get an interactive session gives you an interactive session using the windfall queue. This means your interactive session may be killed ***at any time.*** Therefore, I recommend setting an alias for the command above.

2. Load the anaconda module (you will NOT have to do this every time) and activate the base conda environment

   ```
   module load anaconda/2020
   conda init bash
   source ~/.bashrc && conda activate
   ```

3. Create a new conda environment with all of the packages and extensions you need!

```
conda create --name <ENV_NAME> python=3.8
conda activate <ENV_NAME>
conda install matplotlib pandas h5py <WHATEVER_OTHER_PACKAGES_YOU_WANT>
conda install -c conda-forge jupyter_contrib_nbextensions
```

> Note: Each time you long onto an interactive node on the hpc, conda will run automatically, and you will be using the `base` conda environment (the environment downloaded on the hpc by the hpc team). In order to use your own environment with your personally downloaded packages, you must use `source activate <ENV_NAME>` upon log in. It is possible to make it such that every time you log onto an interactive node, you automatically activate your environment, but I would recommend making this command an easy alias (I usually just use the name of my environment) so that you can more easily switch if you have more than one environment.

4. If you start a jupyter notebook in this environment, it will still use the base jupyter notebook installation on the hpc (in `/opt/ohpc/pub/…`) which we cannot modify. So, we need to make sure that if we start a jupyter notebook, it will initiate using the conda environment we just set up so that it has nbextensions.

```
pip install environment_kernels
```

> Note: if all you want is to have an environment where you have your own personally downloaded packages (for example, if the hpc did not have pandas, you could use a python environment that has pandas installed to run your code!), then you will not

> need this step.
> Note 2: in general, it is not GREAT practice to use a combo of conda and pip, HOWEVER I didn't find a way around this using conda

5. (I don't know if this step was *strictly * necessary, but here it is just in case you run into errors with passwords, etc). Now we need to make sure that our notebooks will open with the right preferences and settings, with whatever other info comes along with it.

   ```
   cd ~/.jupyter/
   jupyter notebook --generate-config
   ```

2. Creating job script to run a jupyter notebook

We need a slurm submission script so that we are able to submit a job to assign our jupyter notebook session to a node! In the script below, I have used ntasks=4 because I needed more memory than 1 cpu provided.

This job submission script is located at `~/jn.slurm`

```
#!/bin/bash

#SBATCH --output=~/jn.out
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=1
#SBATCH --time=8:00:00
#SBATCH --partition=high_priority
#SBATCH --account=gbesla
#SBATCH --qos=user_qos_gbesla

### move to research directory
cd /xdisk/gbesla/katiechambe

### activate conda environment
source activate <ENV_NAME>

###
jupyter notebook --no-browser --port=1234
```

In this script, the important things to note or change for yourself are:

- `—-output=~/jn.out` — change this to your preferred location for your slurm output files (which you will hopefully never have to look at)
- `--time=8:00:00` — change this if you want to increase or decrease the wall time of your job! This is, effectively, how long your jupyter notebook kernel will be accessible after you start your job
- `--partition=high_priority` — you can also use “standard” or “windfall”, however, don’t use windfall unless you have ZERO other options (i.e. someone used ALL the group hours)
- `cd /xdisk/gbesla/katiechambe` — this moves to the top directory under which all of your notebooks are stored. In general, this could be changed to `/` or `~/`. In my case, this is the same directory where I go when I log into puma.
- `jupyter notebook --no-browser --port=1234` — you can change the port number to whatever you want (for a port number that is not used, might take a few tries but 1234, 8888, and 9000 all worked for me). Whatever port number you choose, this MUST be the same as in the `ssh -L 1234:local…` command in the next section.

3. Starting a Jupyter notebook job and connecting to it on your local machine’s browser

more information at:

▼ links

https://github.com/ua-astro-grads/ua-astro-grads.github.io/wiki/High-Performance-Computing-Lore

https://github.com/ua-astro-grads/ua-astro-grads.github.io/wiki/Off-campus-computing-and-remote-access#forward-jupyter-notebooks--jupyterlab

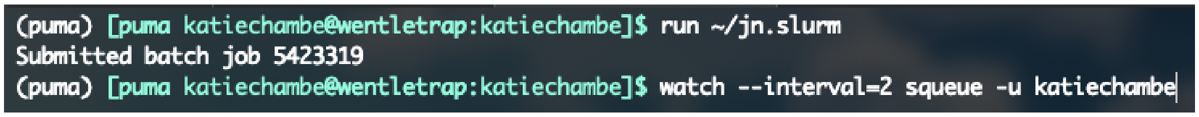
https://thedatafrog.com/en/articles/remote-jupyter-notebooks/

1. On your local machine, ssh into the hpc:

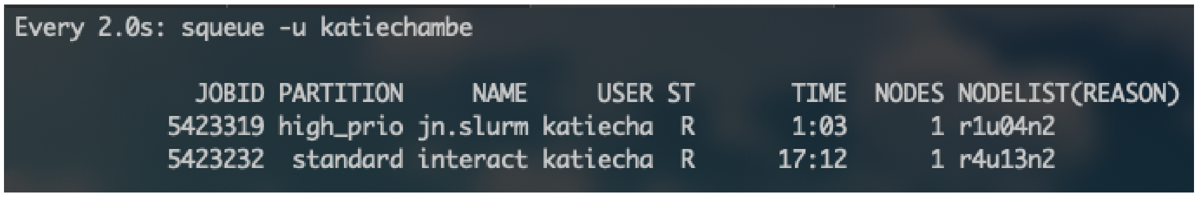
```
ssh puma
```

2. Once on the puma login node (wentletrap), run the job script that submits your job that starts a jupyter notebook on port `1234` (or whichever port number you choose):

```
run ~/jn.slurm
watch --interval=2 squeue -u $USER
```



```
(puma) [puma katiechambe@wentletrap:katiechambe]$ run ~/jn.slurm
Submitted batch job 5423319
(puma) [puma katiechambe@wentletrap:katiechambe]$ watch --interval=2 squeue -u katiechambe
```

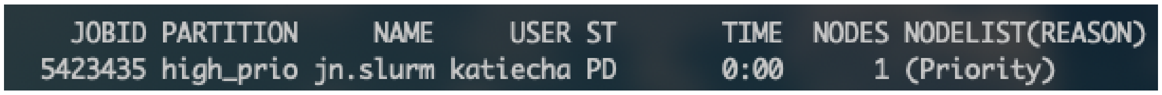


```
Every 2.0s: squeue -u katiechambe

             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           5423319 high_prio jn.slurm katiecha  R       1:03      1 r1u04n2
           5423232  standard interact katiecha  R      17:12      1 r4u13n2
```

This is the expected output of the two preceding lines of code. Look for the line with the correct JOBID (or name of your script). The node associated with your jupyter notebook job will be listed here under `NODELIST`. In the example above, the jupyter notebook was assigned to node **`r1u04n2`** **. * COPY THIS NODE ID ***

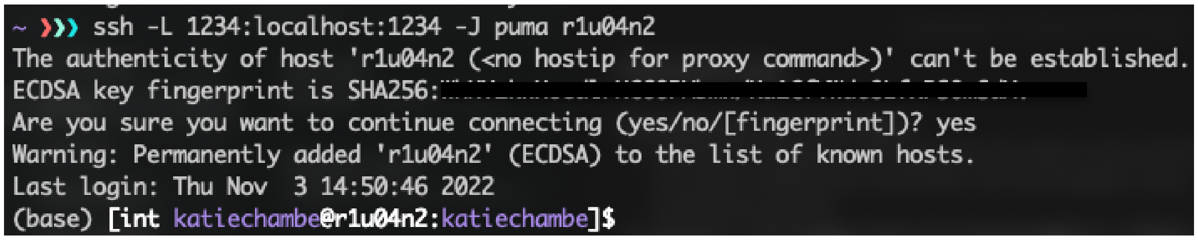
> Note: jobs can take a while to start and may sit quietly in the queue until assigned to a node. When watching the queue status (with the `watch` command above), the `NODELIST` column may read `(Priority)` as seen in the pic below. Simply wait for your job to be assigned to a node and start running, so that you can be sure to copy the correct node ID.



```
  JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
5423435 high_prio jn.slurm katiecha PD       0:00      1 (Priority)
```

3. Once you have **copied the node ID**, you can close your connection to the hpc (or open a new terminal window or tab on your local machine.) Then, on your local machine, start a screen session and establish your connection to the hpc node where your jupyter notebook is running.

```
screen -S jupy
ssh -L 1234:localhost:1234 -J puma r1u04n2. #(your port number name not be 1234)
```

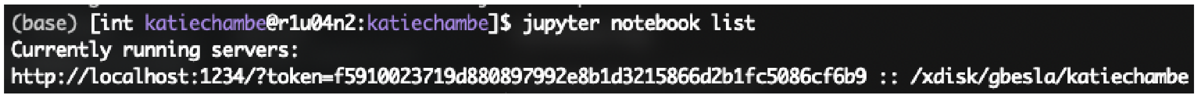


```
~ ❯❯❯ ssh -L 1234:localhost:1234 -J puma r1u04n2
The authenticity of host 'r1u04n2 (<no hostip for proxy command>)' can't be established.
ECDSA key fingerprint is SHA256:[redacted]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'r1u04n2' (ECDSA) to the list of known hosts.
Last login: Thu Nov  3 14:50:46 2022
(base) [int katiechambe@r1u04n2:katiechambe]$
```

> Note: The screen session will allow you to keep the connection to the hpc active, even if you close your terminal window. This also means you don't have to look at the output from your jupyter notebooks anymore! Additionally, the first time you connect to a node that your computer does not recognise, you will have to type "yes" to continue connecting to the node. (You can force the connection and auto-accept, but that's up to you)

4. Get the link to the jupyter notebook that you will need to open the notebook in your browser on your local machine,

```
jupyter notebook list
```



```
(base) [int katiechambe@r1u04n2:katiechambe]$ jupyter notebook list
Currently running servers:
http://localhost:1234/?token=f5910023719d880897992e8b1d3215866d2b1fc5086cf6b9 :: /xdisk/gbesla/katiechambe
```

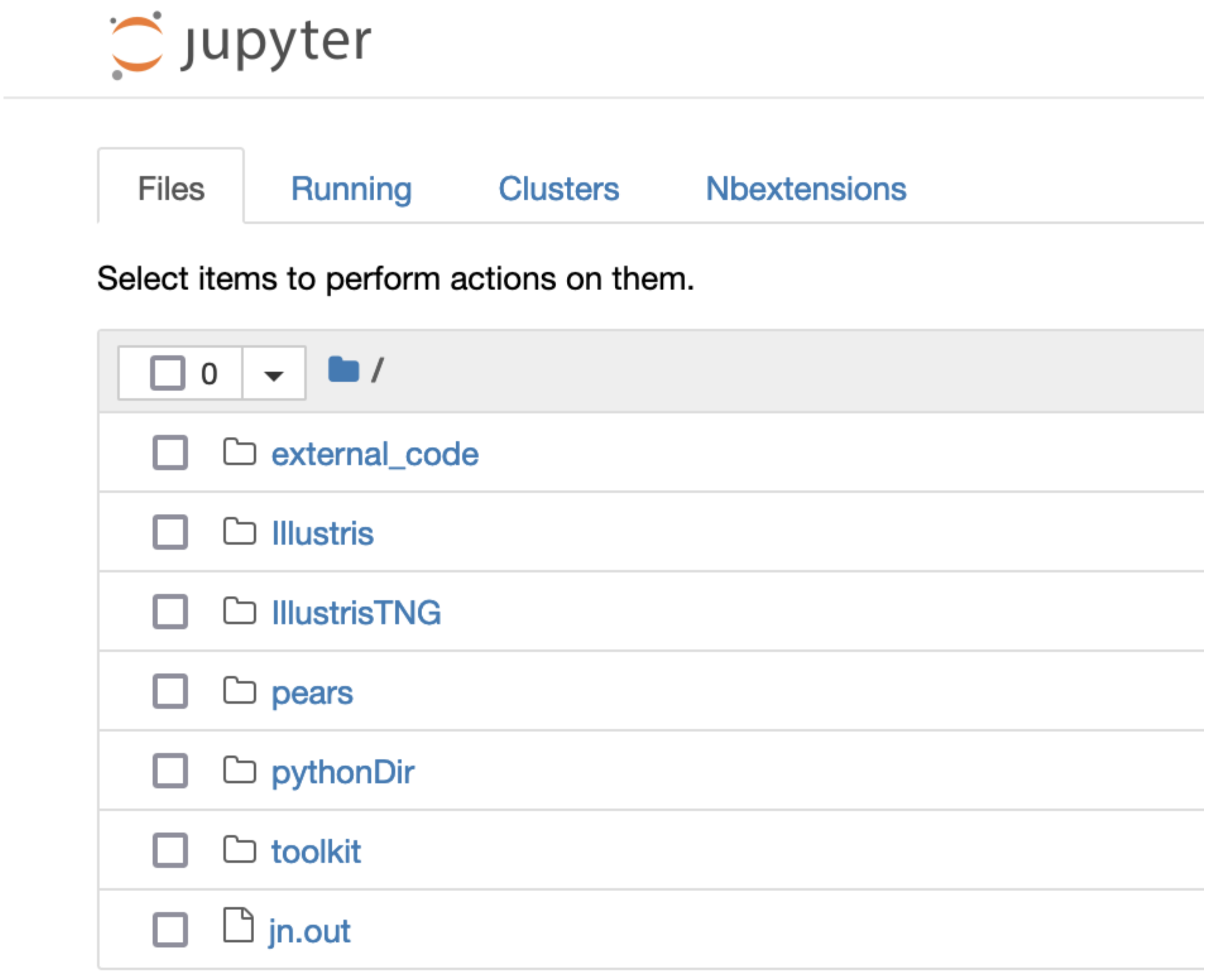
> Note: you can also simply type `http://localhost:1234/tree` but this may lead to password issues. If you run into those, you can use the method above which will specify the "token."

5. Now, copy and paste that link into your preferred browser! Yay!
6. If you want, you can "detach" from the screen on your local machine. To do this, simply press Ctrl+A then D (hold Ctrl the whole time). In order to reattach to the screen (say, if you want to close your connection to the hpc or look at your memory usage on the compute node you're using), you can use `screen -x jupy.` If you have more than one jupy screen, this will tell you that you have numerous options.

> Note: on a mac or linux machine, you can kill all of your screens with `pkill SCREEN`

4. Setting up Nbextensions

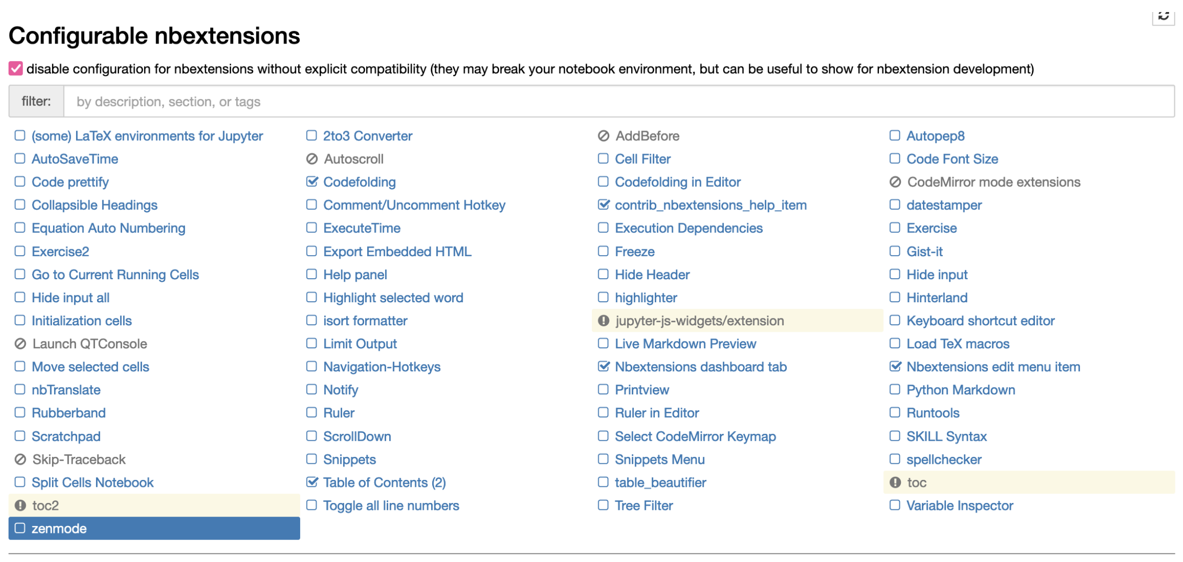
Extra: If you have installed Nbextensions, they should be available as a tab at the top of your tree home page.



Select items to perform actions on them.

external_code
Illustris
IllustrisTNG
pears
pythonDir
toolkit
jn.out

From here, you can edit which extensions you'd like to use. I am not familiar with most of them, but I'd highly recommend checking "Codefolding" and "Table of Contents (2)"



Configurable nbextensions

☑ disable configuration for nbextensions without explicit compatibility (they may break your notebook environment, but can be useful to show for nbextension development)

filter: by description, section, or tags

☐ (some) LaTeX environments for Jupyter	☐ 2to3 Converter	⊘ AddBefore	☐ Autopep8
☐ AutoSaveTime	⊘ Autoscroll	☐ Cell Filter	☐ Code Font Size
☐ Code prettify	☑ Codefolding	☐ Codefolding in Editor	⊘ CodeMirror mode extensions
☐ Collapsible Headings	☐ Comment/Uncomment Hotkey	☑ contrib_nbextensions_help_item	☐ datestamper
☐ Equation Auto Numbering	☐ ExecuteTime	☐ Execution Dependencies	☐ Exercise
☐ Exercise2	☐ Export Embedded HTML	☐ Freeze	☐ Gist-it
☐ Go to Current Running Cells	☐ Help panel	☐ Hide Header	☐ Hide input
☐ Hide input all	☐ Highlight selected word	☐ highlighter	☐ Hinterland
☐ Initialization cells	☐ isort formatter	❗ jupyter-js-widgets/extension	☐ Keyboard shortcut editor
⊘ Launch QTConsole	☐ Limit Output	☐ Live Markdown Preview	☐ Load TeX macros
☐ Move selected cells	☐ Navigation-Hotkeys	☑ Nbextensions dashboard tab	☑ Nbextensions edit menu item
☐ nbTranslate	☐ Notify	☐ Printview	☐ Python Markdown
☐ Rubberband	☐ Ruler	☐ Ruler in Editor	☐ Runtools
☐ Scratchpad	☐ ScrollDown	☐ Select CodeMirror Keymap	☐ SKILL Syntax
⊘ Skip-Traceback	☐ Snippets	☐ Snippets Menu	☐ spellchecker
☐ Split Cells Notebook	☑ Table of Contents (2)	☐ table_beautifier	❗ toc
❗ toc2	☐ Toggle all line numbers	☐ Tree Filter	☐ Variable Inspector
☐ zenmode			

5. Helpful aliases that I have in my bashrc file:

```
# on my local machine:
alias js='screen -S jupy'
alias sshjn='ssh -L 1234:localhost:1234 -J puma'

# on my remote machines (puma, etc):
alias runjn='run ~/jn.slurm'
alias cw='watch --interval=2 squeue -u <USERNAME>'
alias int='interactive -a gbesla -t 04:00:00 -Q user_qos_gbesla'
alias jnurl='jupyter notebook list
```