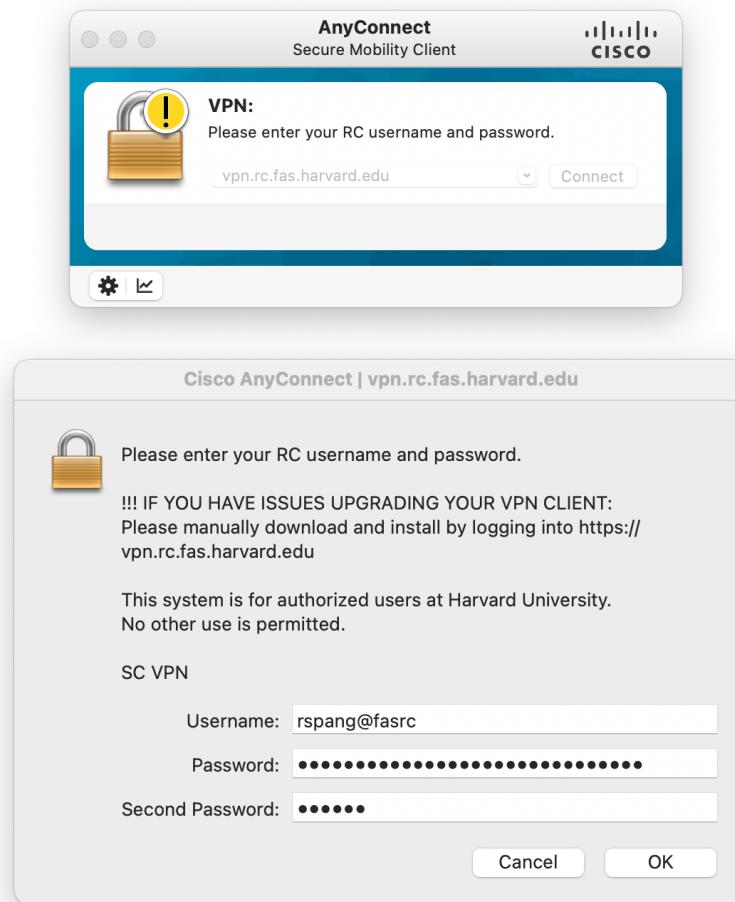


Exercise 1

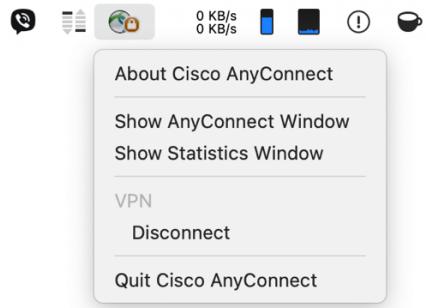
1. Connect to the VPN using Cisco AnyConnect.

To interact with the FASRC from your laptop, you have to be connected to the VPN. Start the Cisco AnyConnect App. Upon click on “Connect”, you’ll be asked to provide two passwords. Password: your FASRC account password. Second Password: the six digit one-time-password from your authenticator app, e.g., Google Authenticator



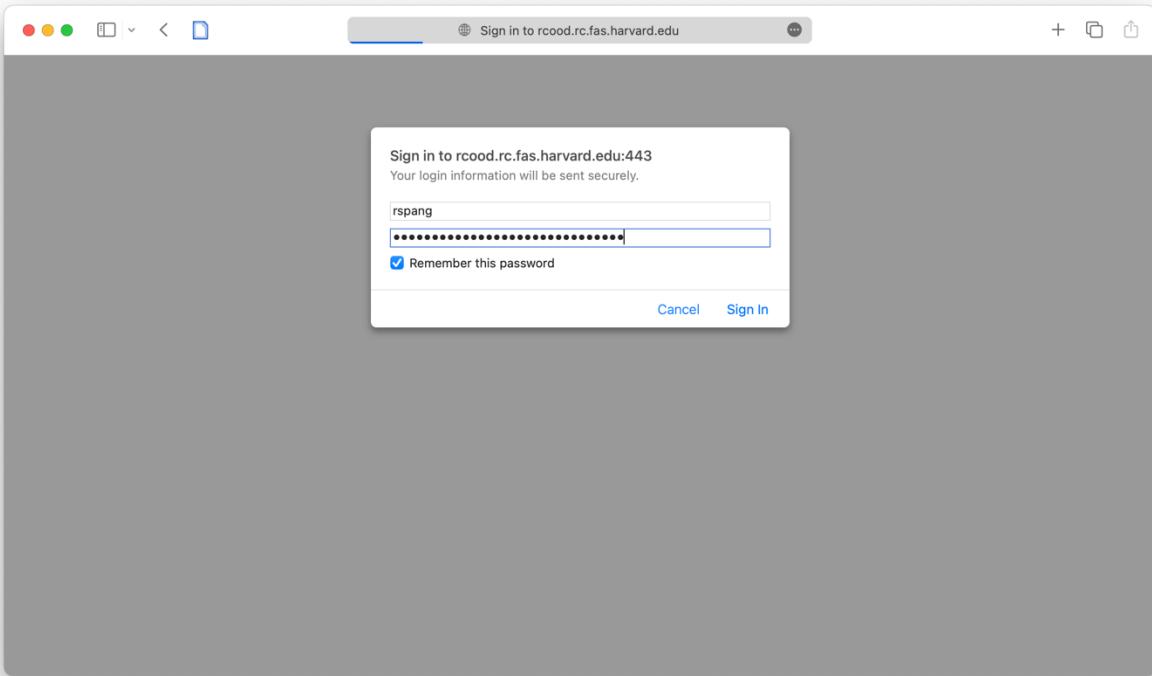
After you clicked “OK”, the VPN connection will be established, both windows disappear after a successful connection.

The AnyConnect icon indicates a successful connection with a padlock-icon.



2. Login to the FASRC web interface

Open <https://rcood.rc.fas.harvard.edu/> and login with your FASRC credentials (not your HarvardKey, but the separate account you created at sign up with the FASRC)



Start an interactive Jupyter notebook session

The screenshot shows the FAS RC web interface for managing interactive applications. At the top, there's a navigation bar with the FAS RC logo, user authentication, and a search bar. Below it, a section titled "Pinned Apps A featured subset of all available apps" displays five system-installed applications:

- Jupyter notebook / Jupyterlab (System Installed App)
- Matlab (System Installed App)
- RStudio Server (System Installed App)
- Remote Desktop (System Installed App)
- SAS (System Installed App)
- Stata (System Installed App)

Default settings are ok... (scroll down)

The screenshot shows the configuration page for starting a Jupyter notebook session. The left sidebar lists various application categories. The main content area is titled "Jupyter notebook / Jupyterlab" and displays the following configuration options:

- version:** a4f8202
- Partition:** test (input field)
- Description:** This app will launch [Jupyter](#) on a compute node on the [FAS-RC cluster](#).
- Use JupyterLab instead of Jupyter Notebook?** (checkbox)
- JupyterLab** is described as the next generation of Jupyter, and is completely compatible with existing [Jupyter Notebooks](#).
- Memory Allocation in GB:** 1 (input field)
- Memory required per node (--mem):** (text)
- Number of cores per task:** (text)

... and click “Launch” to request a new Jupyter session

The screenshot shows a web browser window with the URL `rcood.rc.fas.harvard.edu`. The page contains the following text and form fields:

Also, please make sure you use **long format option only**
(e.g. `--nodelist=holy7c24502` instead of `-w holy7c24502`)

If you are not familiar with Slurm we recommend to leave this blank.

Slurm Account (Optional)
cgaa

Account to charge for this job (--account). This will default to your main account (primary group). More information about [multiple slurm accounts here](#).

email address for status notification (Optional)

I would like to receive an email when the session starts

Launch

* The Jupyter notebook / Jupyterlab session data for this session can be accessed under the [data root directory](#).

powered by **OnDemand** You are on rcood02 OnDemand version: 2.0.29

After a few seconds, the session is running. Click “Connect to Jupyter” to connect.

The screenshot shows a web browser window with the FAS RC logo at the top. The page displays the following information:

Session was successfully created.

Home / My Interactive Sessions

Interactive Apps

- Desktop Apps
 - Matlab
 - SAS
 - Stata
 - Desktops
 - Containerized FAS-RC Remote Desktop
 - Remote Desktop
 - Web Apps
 - HockeyAI

Jupyter notebook / Jupyterlab (2760468) | 1 node | 1 core | Running

Host: `>_holy7c02409.rc.fas.harvard.edu` Delete

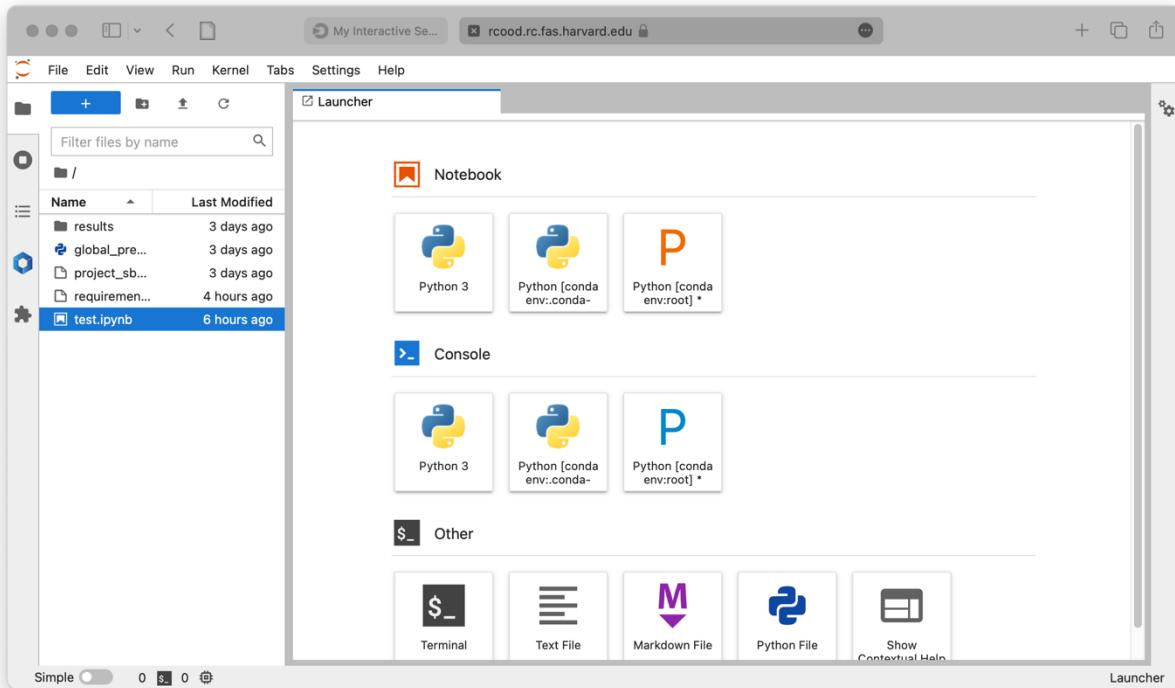
Created at: 2023-09-21 16:47:47 EDT

Time Remaining: 1 hour and 44 minutes

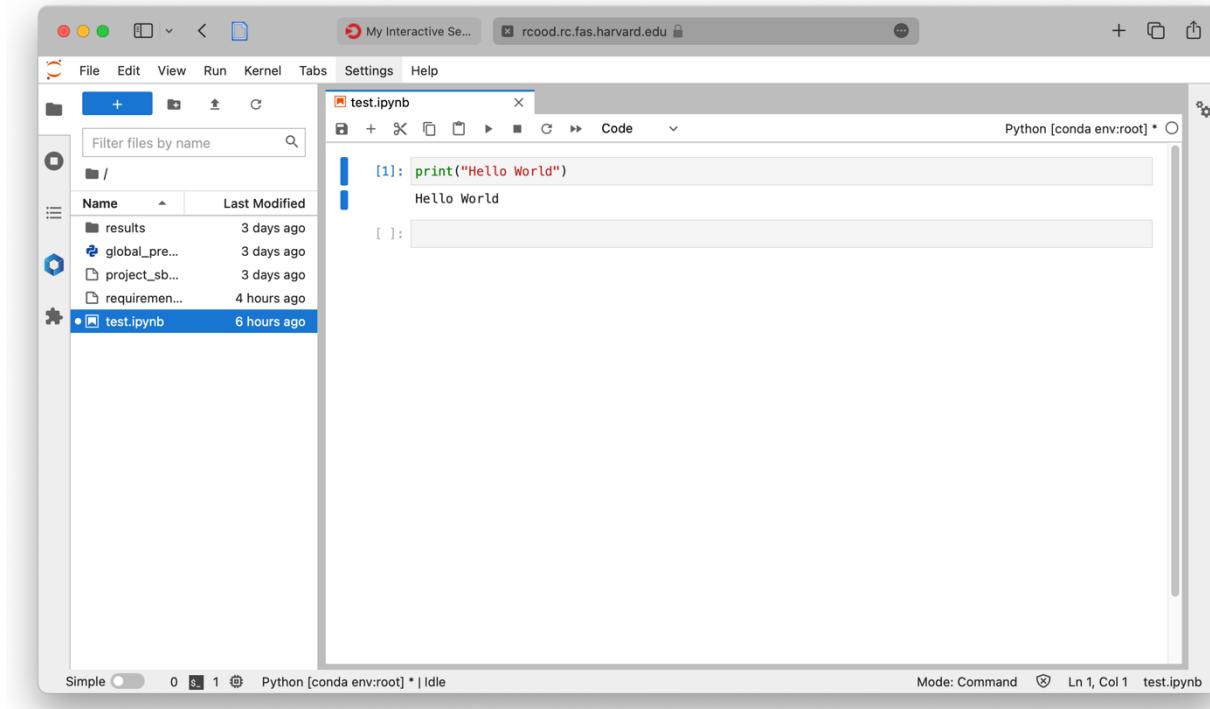
Session ID: `18236603-e4ca-4acf-abdc-c421bfefb7ea`

Connect to Jupyter

In Jupyter, create a new Python 3 Notebook, name it `test.ipynb`



Type and run a simple demo script. This code is executed on a FASRC compute node!



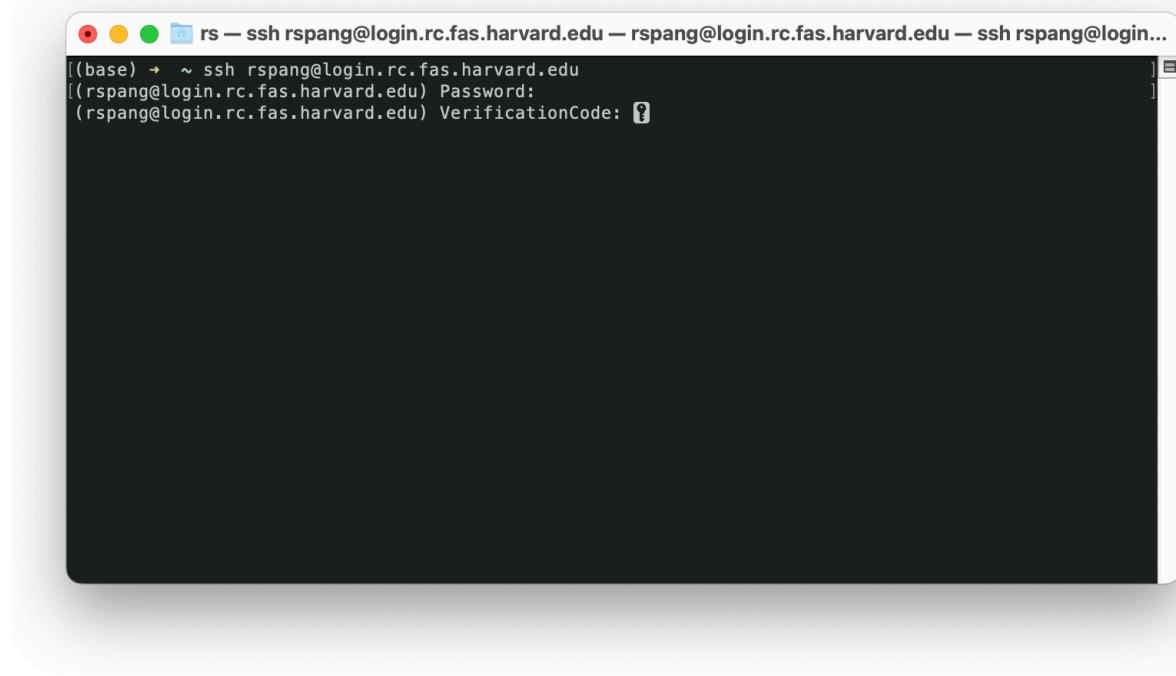
3. Login to the FASRC via the command line

Login to the cluster via SSH.

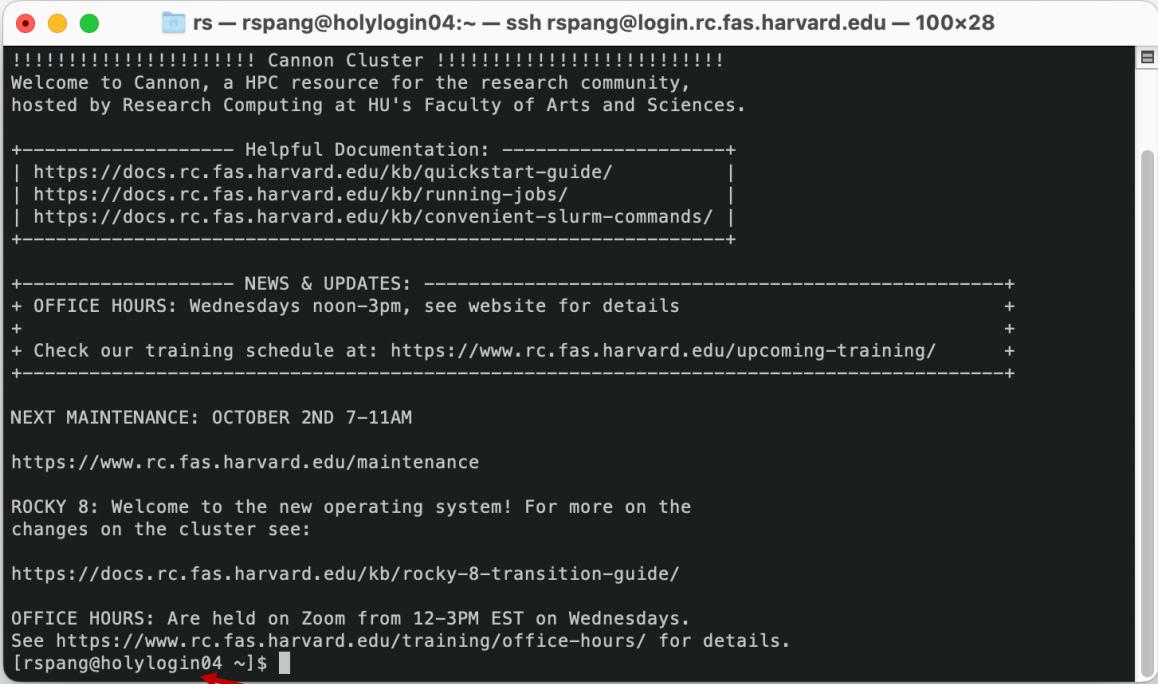
Open a terminal / console and connect to the FASRC login-node using the command:

ssh user@login.rc.fas.harvard.edu (replace “user” with your username)

Similarly to the VPN connection, you’ll be asked to provide two passwords: your FASRC account password, and a one-time-pad password from your authenticator app (“VerificationCode”).



If the connection was successful, you'll be greeted with a welcome message and a few announcements:



The screenshot shows a terminal window titled "rs — rspang@holyllogi04:~ — ssh rspang@login.rc.fas.harvard.edu — 100x28". The window displays a welcome message for the Cannon Cluster, which is a HPC resource for research at Harvard University's Faculty of Arts and Sciences. It provides links for documentation, news, and updates. It also mentions the next maintenance date (October 2nd) and links to the new operating system (Rocky 8). The prompt "[rspang@holyllogi04 ~]\$" is visible at the bottom.

```
!!!!!! Cannon Cluster !!!!!!!  
Welcome to Cannon, a HPC resource for the research community,  
hosted by Research Computing at HU's Faculty of Arts and Sciences.  
----- Helpful Documentation: -----  
| https://docs.rc.fas.harvard.edu/kb/quickstart-guide/ |  
| https://docs.rc.fas.harvard.edu/kb/running-jobs/ |  
| https://docs.rc.fas.harvard.edu/kb/convenient-slurm-commands/ |  
-----  
----- NEWS & UPDATES: -----  
+ OFFICE HOURS: Wednesdays noon-3pm, see website for details +  
+  
+ Check our training schedule at: https://www.rc.fas.harvard.edu/upcoming-training/ +  
+  
NEXT MAINTENANCE: OCTOBER 2ND 7-11AM  
https://www.rc.fas.harvard.edu/maintenance  
ROCKY 8: Welcome to the new operating system! For more on the  
changes on the cluster see:  
https://docs.rc.fas.harvard.edu/kb/rocky-8-transition-guide/  
OFFICE HOURS: Are held on Zoom from 12-3PM EST on Wednesdays.  
See https://www.rc.fas.harvard.edu/training/office-hours/ for details.  
[rspang@holyllogi04 ~]$
```

The shell is configured in a way that it displays the hostname of the server you are using. This is helpful to check if you're on a compute node or the login node. Here, we are on a login node.

Next, setup a python environment: To do so, we first have to load a “module” that makes python available for us to use. Use the following command:

```
module load Mambaforge/23.3.1-fasrc01
```



The screenshot shows a terminal window titled "rs — rspang@holyllogi04:~ — ssh rspang@login.rc.fas.harvard.edu — 100x14". The user runs the command "module load Mambaforge/23.3.1-fasrc01", which is shown in the terminal window.

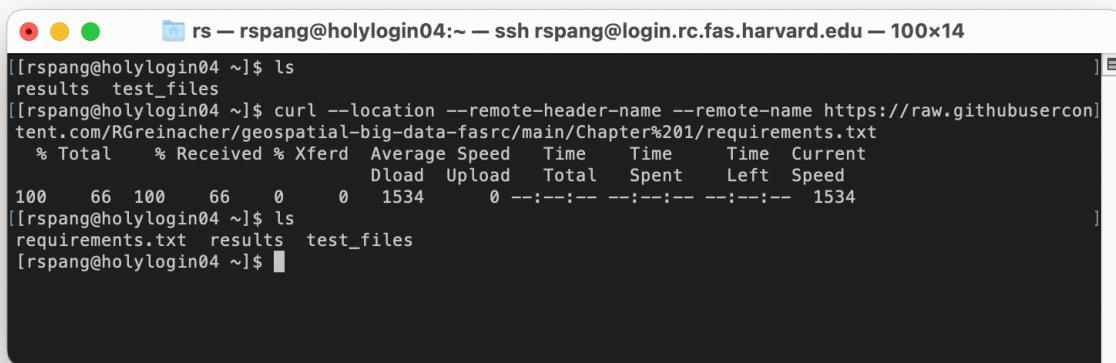
```
[rspang@holyllogi04 ~]$ module load Mambaforge/23.3.1-fasrc01  
[rspang@holyllogi04 ~]$
```

Next, we want to create a new Python environment to work with. We prepared a list of python packages that we will use today in the GitHub repository for this course. Download the file “requirements.txt” from the repo to your home-folder. Copy the following code and execute it:

```
curl --location --remote-header-name --remote-name  
https://raw.githubusercontent.com/cga-harvard/python-workshop-gis-big-  
data/main/Chapter%201/requirements.txt
```

The screenshot below shows the contents of my home folder, then the curl command, and lastly the new contents of my home folder, having a new file: requirements.txt

Caution: generally, be careful what to download from the internet and make sure you trust the resource you download. It makes sense to inspect the file in your browser first.

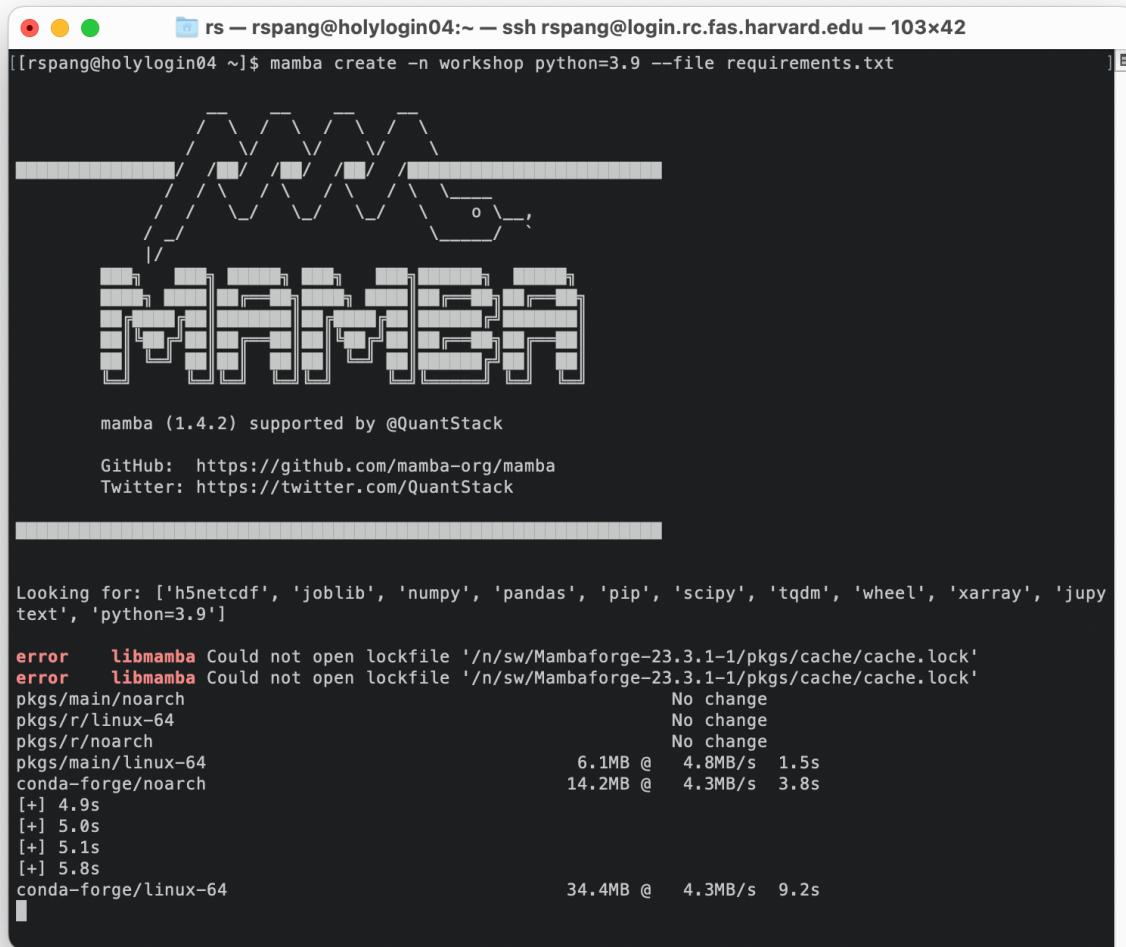


A screenshot of a terminal window titled "rs — rspang@holylaptop04:~ — ssh rspang@login.rc.fas.harvard.edu — 100x14". The window shows the following command and its execution:

```
[rspang@holylaptop04 ~]$ ls  
results test_files  
[[rspang@holylaptop04 ~]$ curl --location --remote-header-name --remote-name https://raw.githubusercontent.com/RGreinacher/geospatial-big-data-fasrc/main/Chapter%201/requirements.txt  
% Total % Received % Xferd Average Speed Time Time Current  
          Dload Upload Total Spent Left Speed  
100 66 100 66 0 0 1534 0 --:--:-- --:--:-- --:--:-- 1534  
[[rspang@holylaptop04 ~]$ ls  
requirements.txt results test_files  
[rspang@holylaptop04 ~]$ ]
```

Now, having the requirements file in place, we can go ahead and create a new Python environment with all packages from the requirements file:

```
mamba create -n workshop python=3.9 --file requirements.txt
```



A screenshot of a terminal window titled "rs — rspang@login04:~ — ssh rspang@login.rc.fas.harvard.edu — 103x42". The window shows the command "mamba create -n workshop python=3.9 --file requirements.txt" being run. The output includes a decorative ASCII art logo, information about mamba version 1.4.2, and links to GitHub and Twitter. It then lists packages being installed, showing progress bars and download speeds. The terminal has a dark background with light-colored text and icons.

```
[[rspang@login04 ~]$ mamba create -n workshop python=3.9 --file requirements.txt

          / \ \ / \ \ / \ \ / \ \ 
         / | \ / | \ / | \ / | \ 
        / / \ / / \ / / \ / / \ 
       / / / \ / / / \ / / / \ 
      / / / / \ / / / / \ / / / 
     / / / / / \ / / / / / \ / 
    / / / / / / \ / / / / / / \ 
   / / / / / / / \ / / / / / / \ 
  / / / / / / / / \ / / / / / / \ 
 mamba (1.4.2) supported by @QuantStack
 GitHub: https://github.com/mamba-org/mamba
 Twitter: https://twitter.com/QuantStack

Looking for: ['h5netcdf', 'joblib', 'numpy', 'pandas', 'pip', 'scipy', 'tqdm', 'wheel', 'xarray', 'jupyter', 'python=3.9']

error   libmamba Could not open lockfile '/n/sw/Mambaforge-23.3.1-1/pkg/cache/cache.lock'
error   libmamba Could not open lockfile '/n/sw/Mambaforge-23.3.1-1/pkg/cache/cache.lock'
pkgs/main/noarch                               No change
pkgs/r/linux-64                                 No change
pkgs/r/noarch                                   No change
pkgs/main/linux-64                             6.1MB @ 4.8MB/s 1.5s
conda-forge/noarch                            14.2MB @ 4.3MB/s 3.8s
[+] 4.9s
[+] 5.0s
[+] 5.1s
[+] 5.8s
conda-forge/linux-64                           34.4MB @ 4.3MB/s 9.2s
```

This starts a process of collecting packages that all are compatible to each other and can work together. This might take a little while. Eventually, the process lists all packages it identified it should download, and asks for a confirmation. Confirm these changes by typing "Y":

```
rs — rspang@holyllogon04:~ — ssh rspang@login.rc.fas.harvard.edu — 100x28
+ pytz          2023.3.post1    py39h06a4308_0    pkgs/main/linux-64  Cached
+ pyyaml        6.0            py39h5eee18b_1    pkgs/main/linux-64  Cached
+ readline      8.2            h5eee18b_0       pkgs/main/linux-64  Cached
+ scipy         1.11.1         py39heeff2f4_0   pkgs/main/linux-64  Cached
+ setuptools    68.0.0         py39h06a4308_0   pkgs/main/linux-64  Cached
+ six           1.16.0         pyhd3eb1b0_1     pkgs/main/noarch   Cached
+ sqlite         3.41.2         h5eee18b_0       pkgs/main/linux-64  Cached
+ tk             8.6.12         h1ccaba5_0      pkgs/main/linux-64  Cached
+ toml          0.10.2         pyhd3eb1b0_0     pkgs/main/noarch   Cached
+ tqdm          4.65.0         py39hb070fc8_0   pkgs/main/linux-64  Cached
+ traitlets     5.7.1          py39h06a4308_0   pkgs/main/linux-64  Cached
+ tzdata         2023c          h04d1e81_0      pkgs/main/noarch   Cached
+ wheel          0.38.4         py39h06a4308_0   pkgs/main/linux-64  Cached
+ xarray         2023.6.0       py39h06a4308_0   pkgs/main/linux-64  Cached
+ xz             5.4.2          h5eee18b_0       pkgs/main/linux-64  Cached
+ yaml           0.2.5          h7b6447c_0      pkgs/main/linux-64  Cached
+ zlib          1.2.13         h5eee18b_0       pkgs/main/linux-64  Cached

Summary:
Install: 63 packages
Total download: 5MB

Confirm changes: [Y/n] ■
```

Subsequently, all packages listed are being downloaded and installed. This also might take a moment.

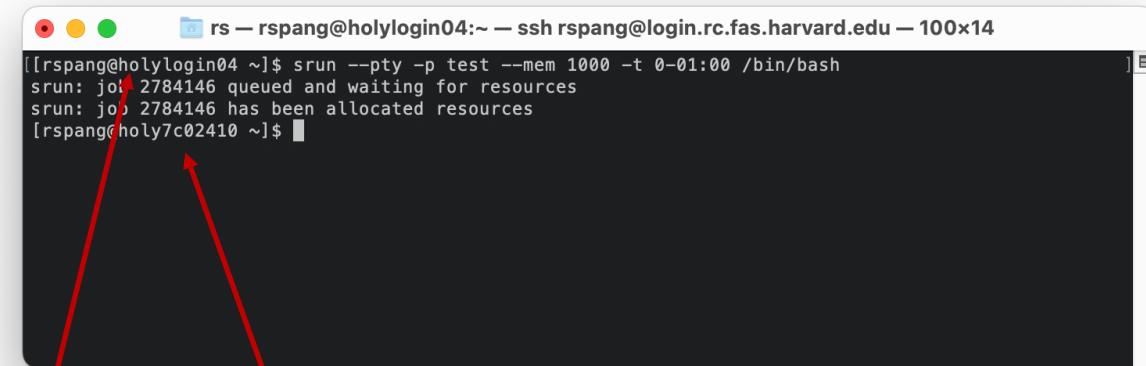
```
rs — rspang@holyllogon04:~ — ssh rspang@login.rc.fas.harvard.edu — 100x28
+ sqlite         3.41.2         h5eee18b_0       pkgs/main/linux-64  Cached
+ tk             8.6.12         h1ccaba5_0      pkgs/main/linux-64  Cached
+ toml          0.10.2         pyhd3eb1b0_0     pkgs/main/noarch   Cached
+ tqdm          4.65.0         py39hb070fc8_0   pkgs/main/linux-64  Cached
+ traitlets     5.7.1          py39h06a4308_0   pkgs/main/linux-64  Cached
+ tzdata         2023c          h04d1e81_0      pkgs/main/noarch   Cached
+ wheel          0.38.4         py39h06a4308_0   pkgs/main/linux-64  Cached
+ xarray         2023.6.0       py39h06a4308_0   pkgs/main/linux-64  Cached
+ xz             5.4.2          h5eee18b_0       pkgs/main/linux-64  Cached
+ yaml           0.2.5          h7b6447c_0      pkgs/main/linux-64  Cached
+ zlib          1.2.13         h5eee18b_0       pkgs/main/linux-64  Cached

Summary:
Install: 63 packages
Total download: 5MB

[Confirm changes: [Y/n] Y
openssl          5.5MB @ 17.9MB/s 0.3s
]■
Downloading and Extracting Packages
Preparing transaction: done
Verifying transaction: | ■
```

Now, the environment is all set and ready to use. To start using Python, switch to a compute-node. Request an interactive session using SLURM:

```
srun --pty -p test --mem 1000 -c 2 -t 0-01:00 /bin/bash
```



```
[rspang@holylogin04 ~]$ srun --pty -p test --mem 1000 -t 0-01:00 /bin/bash
srun: job 2784146 queued and waiting for resources
srun: job 2784146 has been allocated resources
[rspang@holy7c02410 ~]$
```

This will connect you to a different machine. Note how the hostname changes from "holylogin04" to "holy7c02410". Your server names might be different. In any case, we started on a login-node, and switched to a compute-node.

Since we are on a new computer now, we have to load the Python module again. Also, we can now activate the Python environment we created earlier. Use the following two commands:

```
module load Mambaforge/23.3.1-fasrc01
mamba activate workshop
```



```
[rspang@holy7c02410 ~]$ module load Mambaforge/23.3.1-fasrc01
[[rspang@holy7c02410 ~]$ mamba activate workshop
(workshop) [rspang@holy7c02410 ~]$
```

The shell also indicates the active environment.

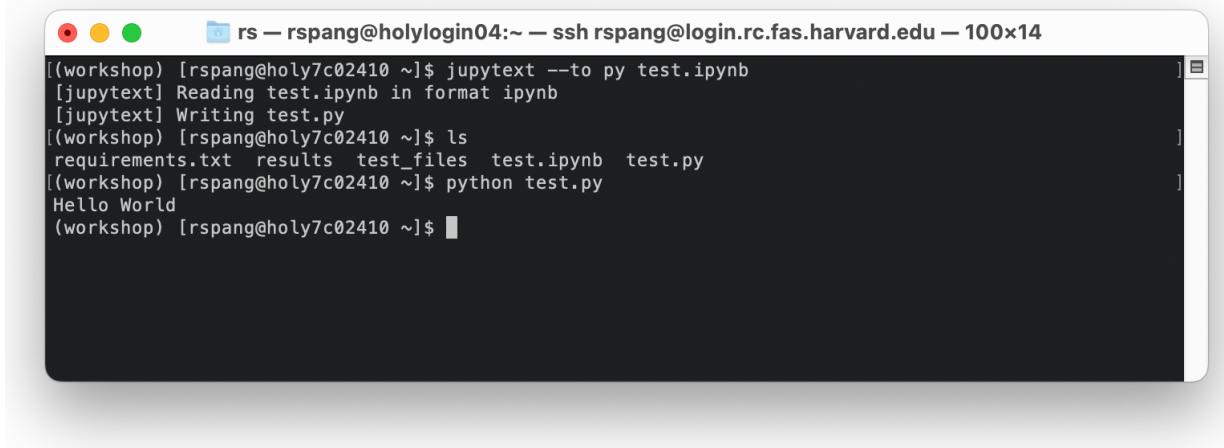
Now, we're finally ready to run Python code! For a little example, we can recycle the demo code we used in our Jupyter notebook earlier. Since Jupyter notebooks cannot be executed on the command line directly, we can first convert the notebook to plain python using:

```
jupytext --to py test.ipynb
```

This generates a new file that is named like the notebook, but it has a “.py” extension. To run

this code, simply use

```
python test.py
```



The image shows a terminal window titled "rs — rspang@holylaptop04:~ — ssh rspang@login.rc.fas.harvard.edu — 100x14". The terminal content is as follows:

```
((workshop) [rspang@holylaptop04 ~]$ jupytext --to py test.ipynb
[jupytext] Reading test.ipynb in format ipynb
[jupytext] Writing test.py
((workshop) [rspang@holylaptop04 ~]$ ls
requirements.txt  results  test_files  test.ipynb  test.py
((workshop) [rspang@holylaptop04 ~]$ python test.py
Hello World
((workshop) [rspang@holylaptop04 ~]$ )
```

What you learned in this exercise:

- How to start a FASRC based Jupyter session in your browser
- How to connect to the FASRC via SSH
- How to download a file from the internet
- How to load modules
- How to create a new Python environment, providing a list of packages
- How to start an interactive SLURM session
- How to convert Jupyter notebooks to Python scripts
- How to run a Python script on a FASRC compute-node