HARVARD UNIVERSITY

FAS RC

FAS Research Computing
Division of Science
https://rc.fas.harvard.edu

FAS RC FAS RESEARCH COMPUTING
HARVARD UNIVERSITY
FACULTY OF ARTS & SCIENCES

Getting Started on the FASRC clusters with Command Line Interface

Learning objectives

- Log in via `ssh` to Cannon and FASSE
- How to start an interactive job with `salloc`
- How to submit a batch job with `sbatch`
- Check job status
- Cluster software modules

Login to Cannon and FASSE – `ssh`

Documentation: https://docs.rc.fas.harvard.edu/kb/terminal-access/


Mac: Terminal, iTerm2


Linux: Xterm or Terminal

Windows


SSH client: Putty


Bash emulator: Git bash

Cannon

```
$ ssh jharvard@login.rc.fas.harvard.edu
Password:
Verification code:
```

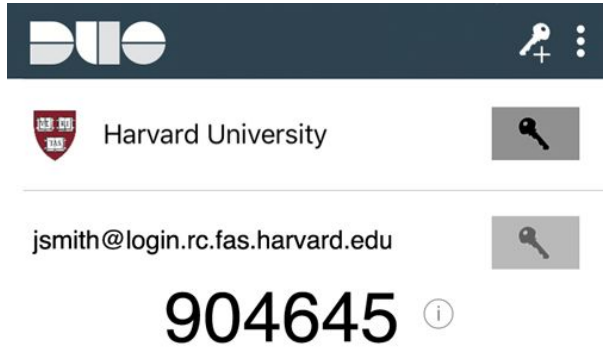
FASSE

```
$ ssh jharvard@fasselogin.rc.fas.harvard.edu
Password:
Verification code:
```

Login to Cannon and FASSE – 2 factor authentication

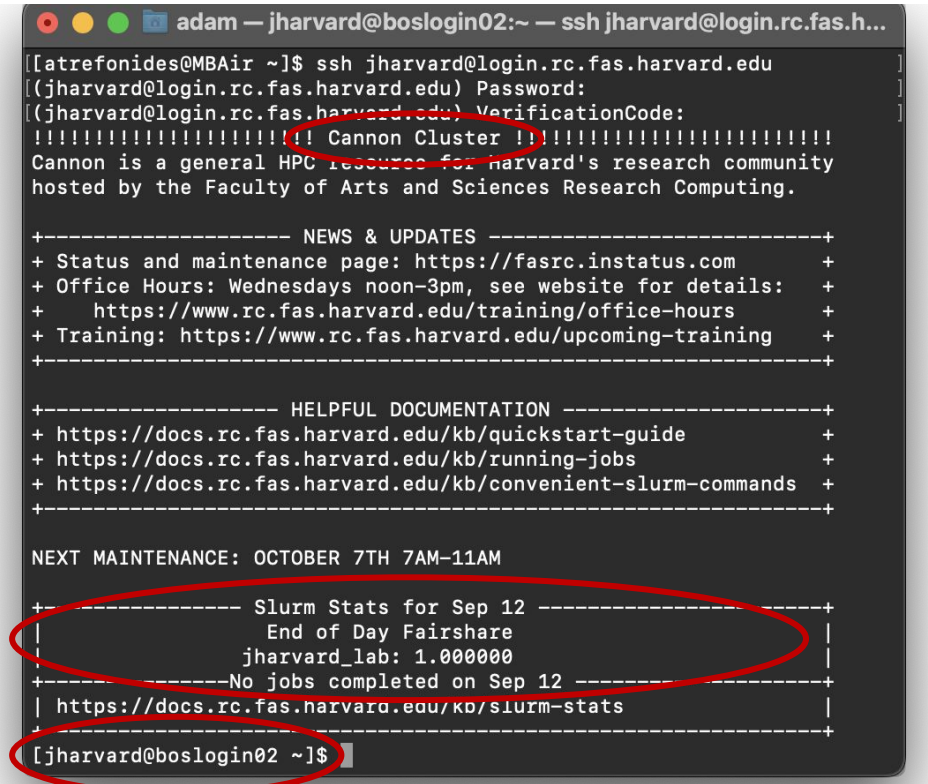
- Execute the ssh command, then:
  - Type your password (*cursor won't move!*), press enter
  - Type the 6-digit verification code (2-Factor Authentication)
    - Separate from HarvardKey
    - Updates token every 30 seconds
    - You can only use a 2FA code once

Java desktop app

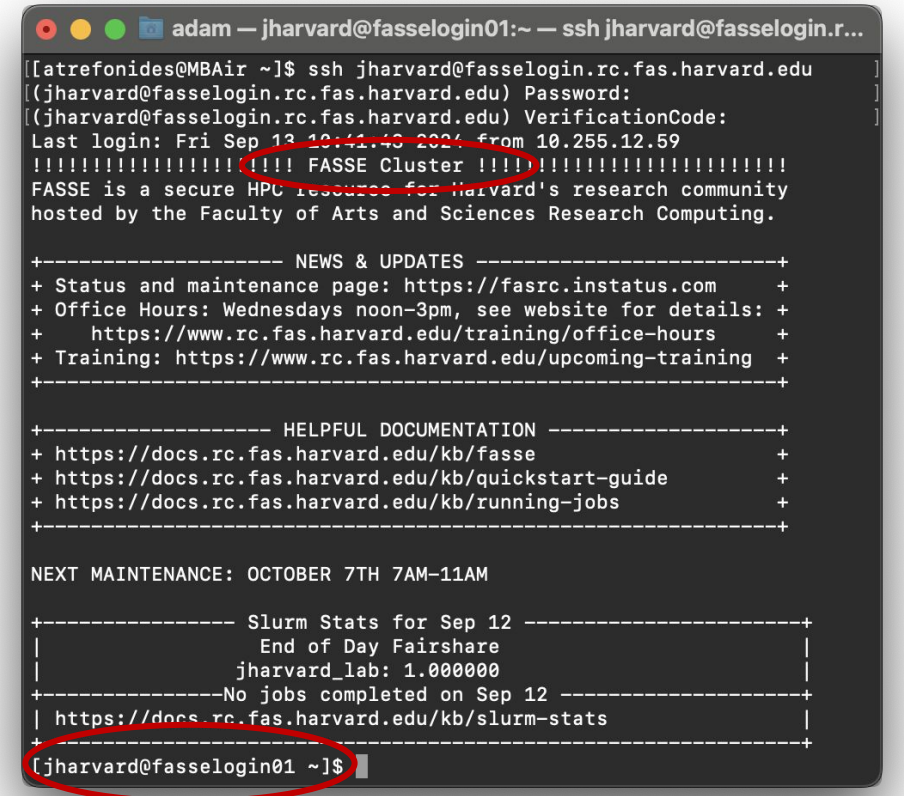


Login to Cannon and FASSE – at login node (1)

Cannon



FASSE



Login to Cannon and FASSE – at login node (2)

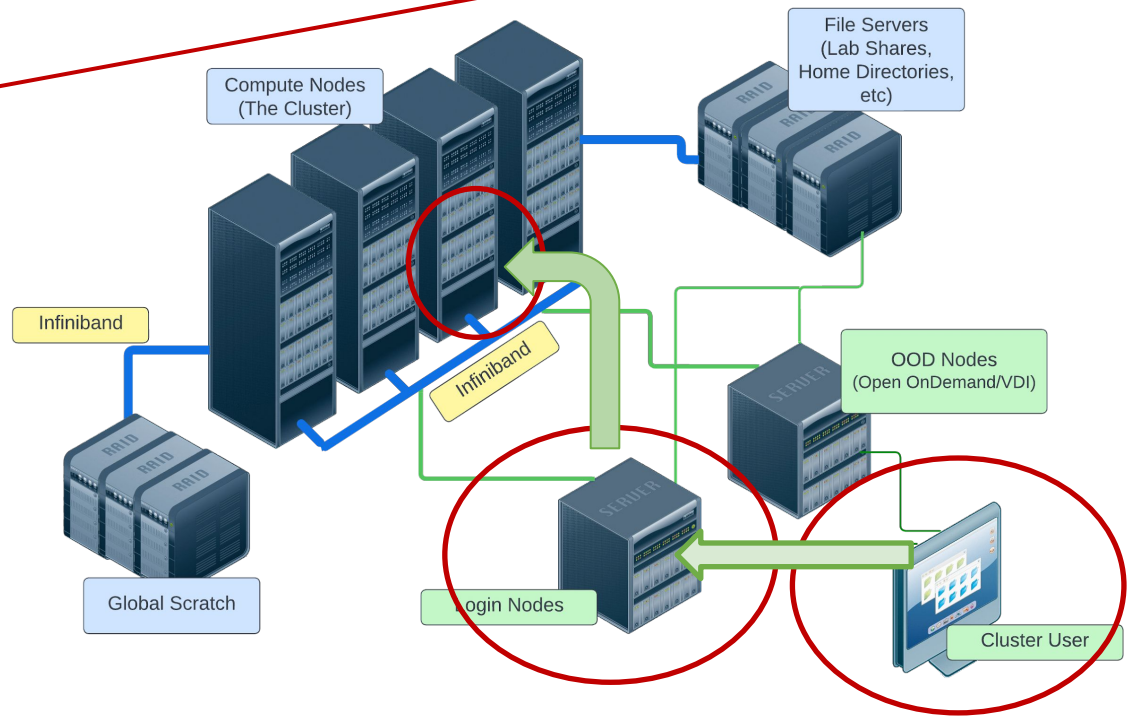
Cannon

```
[jharvard@boslogin01 ~]$
```

Name of the login node assigned to you

FASSE

```
[jharvard@fasselogin01 ~]$
```



Login vs. compute nodes

- Login nodes
  - Limited to 1 core and 4G of memory
  - Not designed for analysis
  - Not anything compute or memory intensive
  - Best practice is to request a compute node as soon as you login
- Compute node via interactive job
  - Work on a compute node interactively – testing, debugging, installing software
  - Request resources from slurm using `salloc` command
  - Session will only last as long as the network connection is active
  - Cannot be idle for more than 1h, session will freeze

Interactive job on Cannon (1)

Requesting an interactive job

```
[jharvard@boslogin01 ~]$ salloc --partition test --mem-per-cpu 1G --time 01:00:00
salloc: Pending job allocation 2741096
salloc: job 2741096 queued and waiting for resources
salloc: job 2741096 has been allocated resources
salloc: Granted job allocation 2741096
salloc: Nodes holy7c02410 are ready for job
[jharvard@holy7c02410 ~]$
```

Name of the compute node assigned to you

`salloc` – slurm command to request interactive job

`--partition test` – requesting a compute node in a specific partition

`--mem-per-cpu 1G` – memory requested in GB (if no unit is specified, the default is MB)

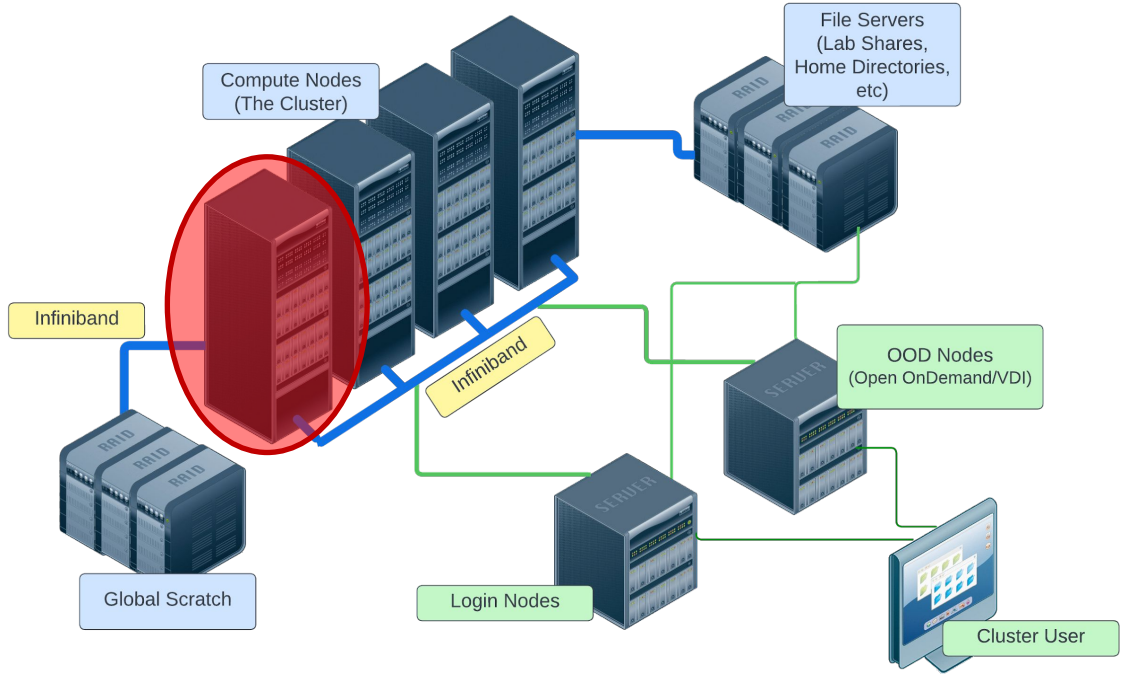
`--time 00:01:00` – time requested (1 hour, format HH:MM:SS or D-HH:MM)

Interactive job on Cannon (2)

Requesting an interactive job

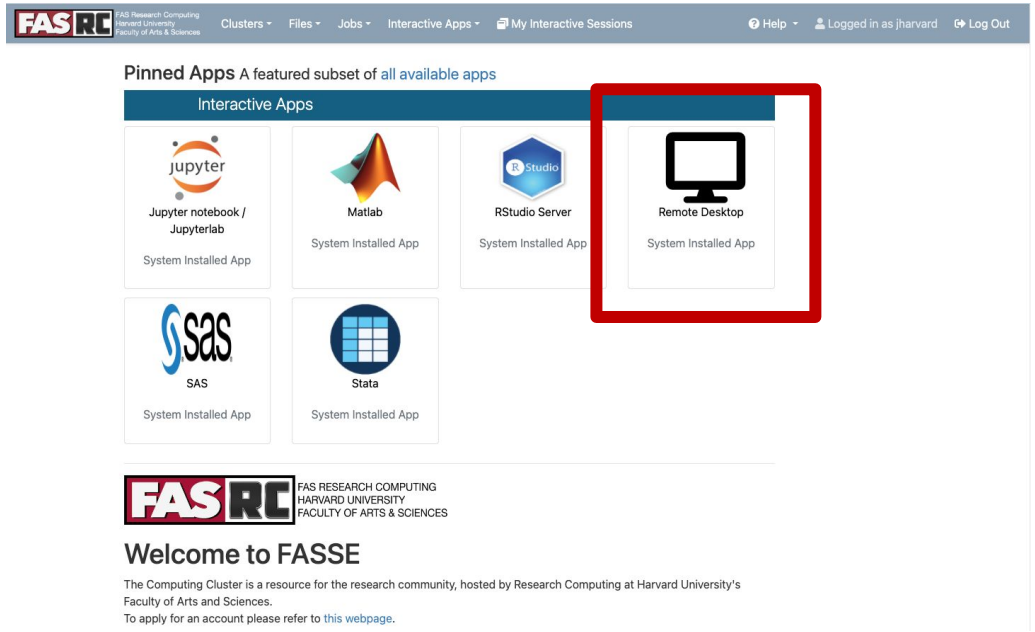
```
[jharvard@holy7c02410 ~]$
```

Name of the compute node assigned to you



Interactive job on FASSE

- You cannot request an interactive job on FASSE
- You must use Remote Desktop app on Open OnDemand https://fasseood.rc.fas.harvard.edu and launch terminal



Test first!!

ALWAYS test the job submission script first:

- To ensure the job will complete without errors
- To ensure you understand the resource needs and have requested them appropriately

Submitting a batch job

```
[jharvard@boslogin01 python]$ sbatch runscript.sh
Submitted batch job 2742999
[jharvard@boslogin01 python]$
```

Batch job

Documentation:
https://docs.rc.fas.harvard.edu/kb/running-jobs/

- Automate job
- No interaction
- Can close your terminal/laptop and job will keep running

- Partitions
  - Cannon: https://docs.rc.fas.harvard.edu/kb/running-jobs/
  - FASSE: https://docs.rc.fas.harvard.edu/kb/fasse/

slurm script `runscript.sh`

slurm directives

```
#!/bin/bash
#SBATCH -J py_job           # Job name
#SBATCH -p test             # Partition(s) (separate with
                            # commas if using multiple)
#SBATCH -c 1                # Number of cores
#SBATCH -t 0-00:30:00       # Time (D-HH:MM:SS)
#SBATCH --mem=500M          # Memory
#SBATCH -o py_%j.o          # Name of standard output
file
#SBATCH -e py_%j.e          # Name of standard error file

# load software environment
module load python/3.10.12-fasrc01

# print a statement
echo "This is our test slurm script"

cd ~jharvard/user_training/intro_cli
# execute python code
python hello_world.py
```

Job monitoring – `sacct`

Documentation:

- `sacct`: slurm accounting database
  - Every 30 sec the node collects the amount of cpu and memory usage that all of the process ID are using for a given job. After the job ends this data is sent to slurm database
- Common flags (i.e., options)
  - `-j jobid or --name=jobname`
  - `-S starttime YYYY-MM-DD and -E endtime YYYY-MM-DD`
  - `-o output_options`
  - See slurm docs for more options: https://slurm.schedmd.com/sacct.html

```
[jharvard@boslogin01 ~]$ sacct --format=JobID,Jobname,partition,state,time,start,end,elapsed,MaxRss,MaxVMSize,nnodes,ncpus,nodelist --units=G -j 2742999
JobID           JobName  Partition      State  Timelimit               Start                 End    Elapsed     MaxRSS  MaxVMSize   NNodes      NCPUS        NodeList
------------ ---------- ---------- ---------- ---------- ------------------- ------------------- ---------- ---------- ---------- -------- ---------- ---------------
2742999          py_job       test  COMPLETED   00:30:00 2023-09-21T12:03:20 2023-09-21T12:03:21   00:00:01                              1          1     holy7c02410
2742999.bat+      batch             COMPLETED            2023-09-21T12:03:20 2023-09-21T12:03:21   00:00:01      0.01G      0.21G        1          1     holy7c02410
2742999.ext+     extern             COMPLETED            2023-09-21T12:03:20 2023-09-21T12:03:21   00:00:01      0.00G      0.17G        1          1     holy7c02410
```

Memory usage

1. Run a test batch job
2. Check memory usage after the job has completed (with `sacct` command)

```
[jharvard@boslogin01 ~]$ sacct -j 2742999 -o ReqMem,MaxRSS
    ReqMem     MaxRSS
---------- ----------
      500M
                7512K
                4348K
[jharvard@boslogin01 ~]$ sacct -j 2742999 -o ReqMem,MaxRSS --units=G
    ReqMem     MaxRSS
---------- ----------
     0.49G
                0.01G
                0.00G
```

Job efficiency summary – `seff`

1. Run a test batch job
2. Check job efficiency after the job has completed (with `seff` command)

```
[jharvard@boslogin01 ~]$ seff 2742999
Job ID: 2742999
Cluster: odyssey
User/Group: jharvard/jharvard_lab
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 00:00:00
CPU Efficiency: 0.00% of 00:00:01 core-walltime
Job Wall-clock time: 00:00:01
Memory Utilized: 7.34 MB
Memory Efficiency: 1.47% of 500.00 MB
```

```
[user@boslogin01 home]# seff 1234567
Job ID: 1234567
Cluster: odyssey
User/Group: user/user_lab
State: COMPLETED (exit code 0)
Nodes: 8
Cores per node: 64
CPU Utilized: 37-06:17:33
CPU Efficiency: 23.94% of 155-16:02:08 core-walltime
Job Wall-clock time: 07:17:49
Memory Utilized: 1.53 TB (estimated maximum)
Memory Efficiency: 100.03% of 1.53 TB (195.31
GB/node)
```

Partitions

`spart` allows you to see which partitions you have access to

Documentation: https://docs.rc.fas.harvard.edu/kb/convenient-slurm-commands/

```
[jharvard@holy8a26602 ~]$ spart
Partition                State      Cores      GPUs       Average Mem/Node(GB)      Nodes      Time Limit
bigmem                   UP         448        0          2015                      4          3-00:00:00
bigmem_intermediate      UP         192        0          2015                      3          14-00:00:00
gpu                      UP         2304       144        1007                      36         3-00:00:00
gpu_h200                 UP         2464       88         1007                      22         3-00:00:00
gpu_requeue              UP         24368      1320       1169                      312        3-00:00:00
gpu_test                 UP         768        96         503                       12         12:00:00
intermediate             UP         1344       0          1007                      12         14-00:00:00
remoteviz                UP         32         0          377                       1          3-00:00:00
sapphire                 UP         20832      0          1007                      186        3-00:00:00
serial_requeue           UP         110544     1320       605                       1586       3-00:00:00
shared                   UP         17760      0          188                       370        3-00:00:00
test                     UP         2016       0          1007                      18         12:00:00
unrestricted             UP         384        0          188                       8          UNLIMITED
```

Software – LMOD module system

- Software is loaded incrementally using modules, to set up your shell environment (e.g., `PATH`, `LD_LIBRARY_PATH`, and other environment variables)
- Why add `module load` commands in a slurm batch script? (instead of `.bashrc` file)
  - Keeps your interactive working environment simple
  - Is a record of your research workflow (reproducible research!)
  - Keeping `.bashrc` module loads to a minimum helps avoid software and library conflicts

```
module load matlab/R2022b-fasrc01    # recommended command
module load matlab                   # loads most recent version
module list                          # show loaded modules
module purge                         # unload all loaded modules
module spider matlab                 # search for modules with matlab in the name
module display matlab/R2022b-fasrc01 # show the details of the module
```

Python modules

- Miniforge+mamba replacing Anaconda+conda: fast, robust, and cross-platform package manager
  - Mamba is a tool to manage `conda` environments
  - Mamba uses the same commands and configuration options as `conda`
  - You can swap almost all commands between `conda` & `mamba`
- Python 3

```
[jharvard@holy7c24103 ~]$ module load python/3.12.5-fasrc01
[jharvard@holy7c24103 ~]$ module list

Currently Loaded Modules:
  1) Miniforge3/24.7.1-fasrc01   2) python/3.12.5-fasrc01
```

Python package install

Documentation
https://docs.rc.fas.harvard.edu/kb/python-package-installation

```
# Load a Python module
module load python/3.12.5-fasrc01

# Create conda environment in ~/.conda/envs/ENV_NAME
mamba create -n ENV_NAME PACKAGE_LIST -y

# Use the new environment
mamba activate ENV_NAME

# Install a new package named MY_PACKAGE
mamba install MY_PACKAGE

# If the package is not available with conda/mamba use pip
pip install MY_PACKAGE

# If you have problems updating a package first remove it
mamba uninstall MY_PACKAGE

# Deactivate the environment
mamba deactivate
```

Python – conda/mamba env in Lab storage

For optimal performance it is recommended to use the `--prefix` option to create your conda environments to your LAB space, e.g.,

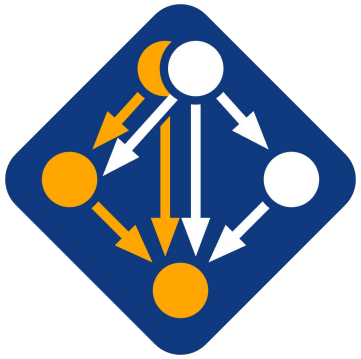
```
/n/holylabs/LABS/<PI_LAB>/Lab
```

```
# Load a Python module, e.g.,
module load python/3.12.5-fasrc01

# Create a conda environment in LAB space, e.g.,
mamba create -y --prefix=/n/holylabs/LABS/<PI_LAB>/Lab/conda/<ENV_NAME>PACKAGE_LIST

# Activate the conda environment
mamba activate /n/holylabs/LABS/<PI_LAB>/Lab/conda/<ENV_NAME>
```

Spack



- For software that does not have a module, you can install it with Spack: https://docs.rc.fas.harvard.edu/kb/spack/
- Install Spack in a Holyoke storage location, such as `holylabs`
  - Package installation is best done in an interactive session with 8 cores 12GB as Spack needs more resources

```
salloc --partition test --time 0-04:00 --mem 12G --cpus-per-task 8
```

Training session evaluation

Please, fill out our training session evaluation. Your feedback is essential for us to improve our trainings!!

https://tinyurl.com/FASRC-training

FASRC documentation

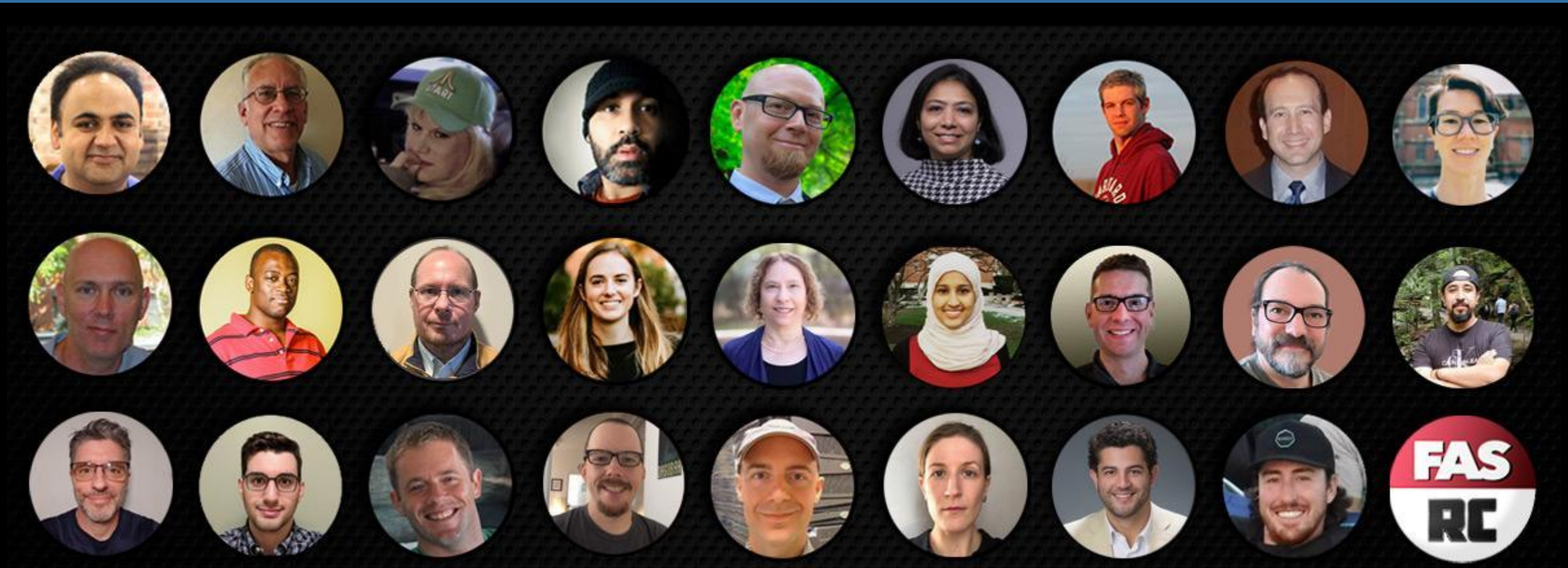
- FASRC docs: https://docs.rc.fas.harvard.edu/
- GitHub User_codes: https://github.com/fasrc/User_Codes/
- Getting help
  - Office hours: https://www.rc.fas.harvard.edu/training/office-hours/
  - Ticket
    - Email: rchelp@rc.fas.harvard.edu

Upcoming training sessions

Training calendar: https://www.rc.fas.harvard.edu/upcoming-training/

Getting started on the FASRC clusters with Open OnDemand

- Audience
  - New users not familiar with command-line interface
  - Wants to use a GUI
- Requirements
  - Single-node jobs
  - Working FASRC account with cluster access
- Content
  - Access Open OnDemand
  - Launch Jupyter, Rstudio Server, Remote Desktop
  - Install Rstudio Server packages
  - Install python packages for Jupyter
  - Launch software from Remote Desktop



Thank you :)

FAS Research Computing