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:
```

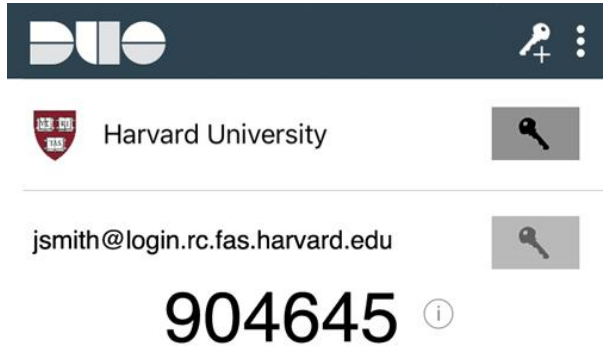
FASSE

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

Login to Cannon and FASSE – 2 factor authentication

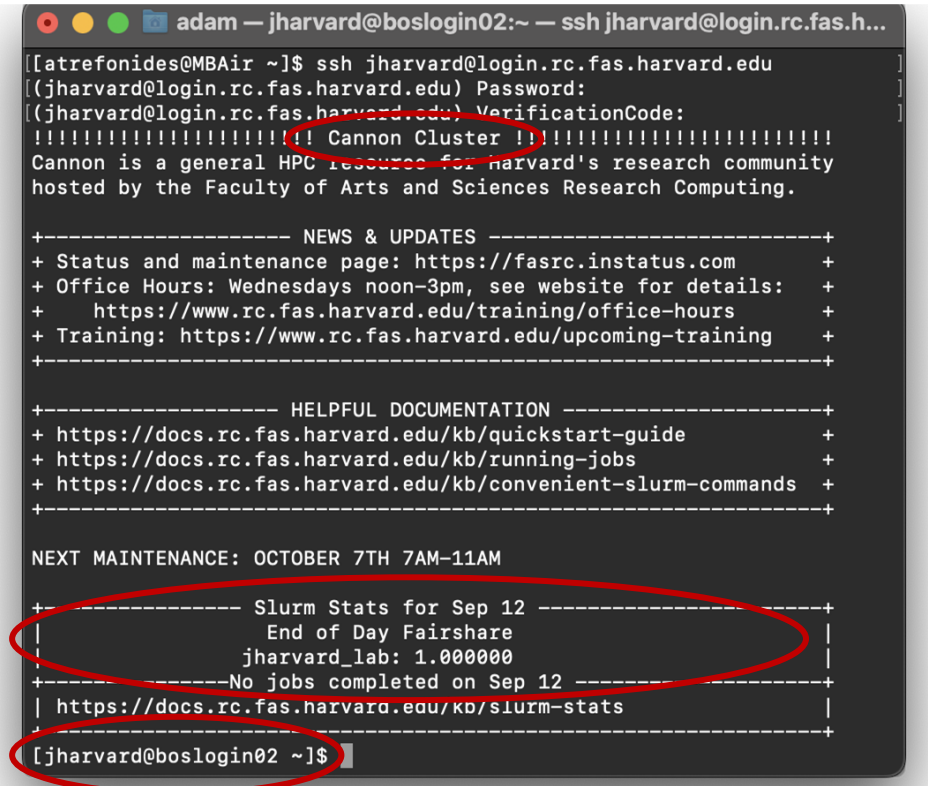
- Execute the ssh command, then:
  - Type your password (*cursor won't move!*), press enter
  - Type the 6-digit verification code (2-Factor Authentication)
    - Separate from HarvardKey
    - Updates token every 30 seconds
    - You can only use a 2FA code once

Java desktop app

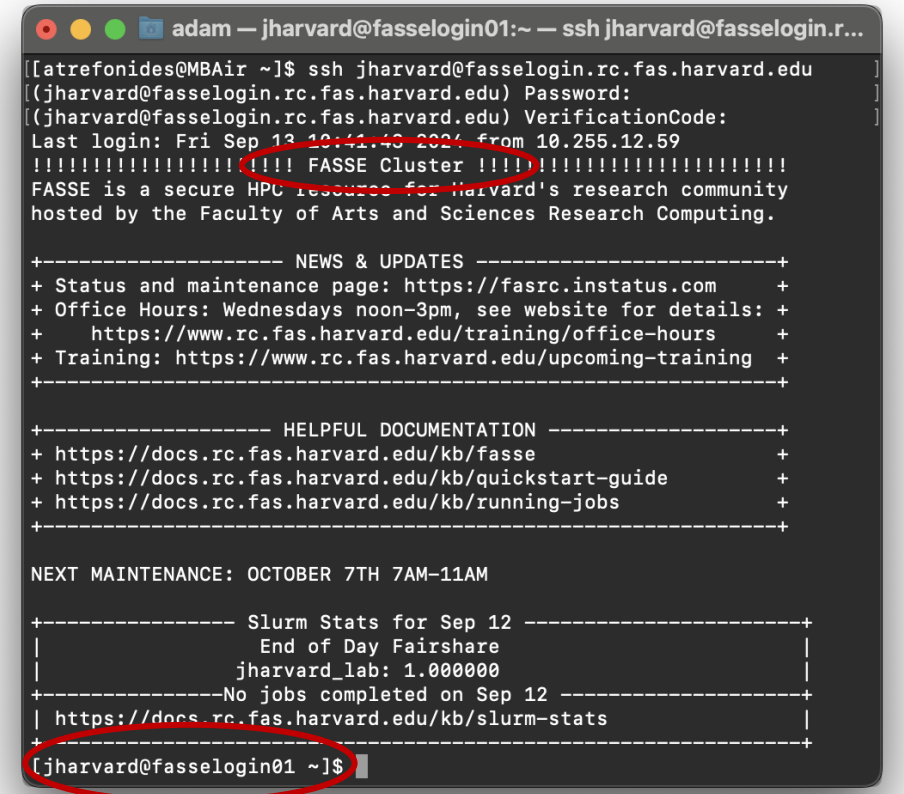


Login to Cannon and FASSE – at login node (1)

Cannon



FASSE



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

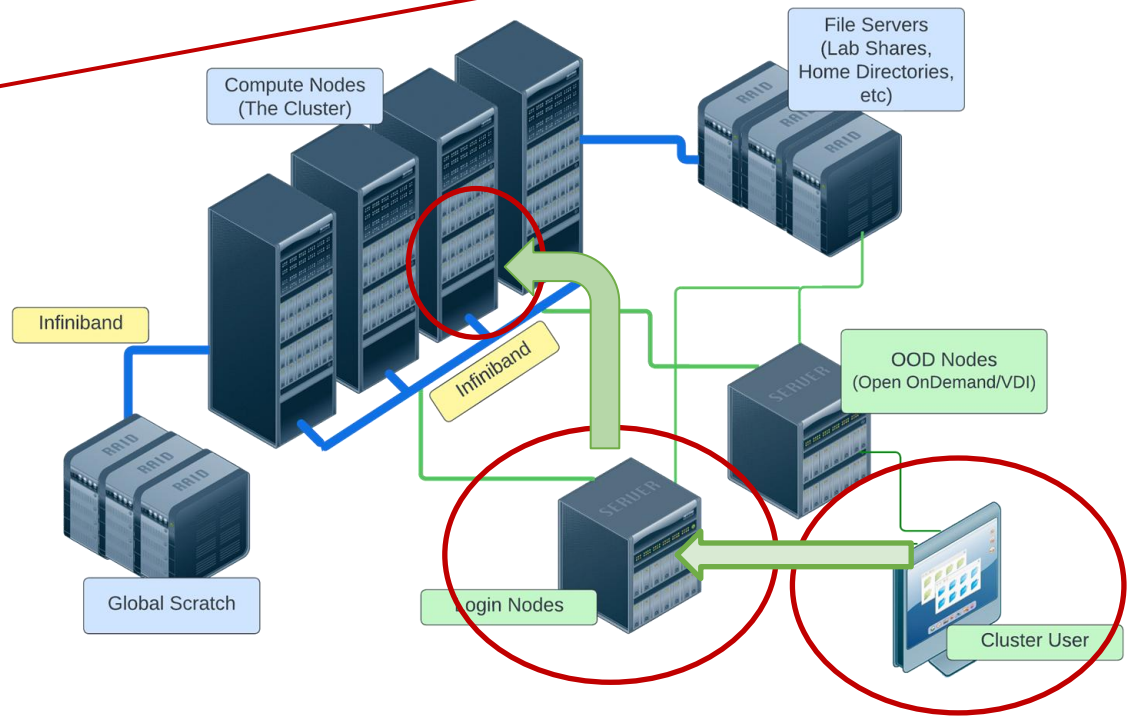
Cannon

```
[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

Basic Linux Commands

Change to working directory

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

Make a new directory (folder)

mkdir my_test

List directory contents

ls my_test

Getting linux help: man <command>

Text file editors: vim, emacs, nano

Resources

https://swcarpentry.github.io/shell-novice/

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)

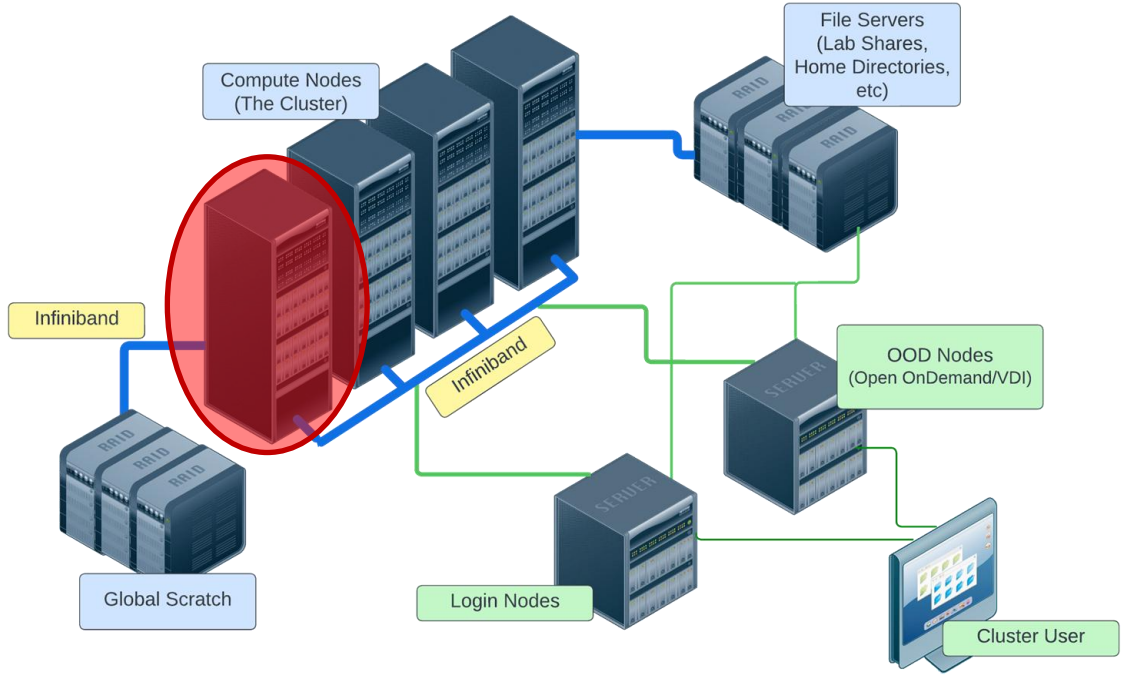
`--time 00:01:00` – time requested (1 hour, format HH:MM:SS or D-HH:MM)

Interactive job on Cannon (2)

Requesting an interactive job

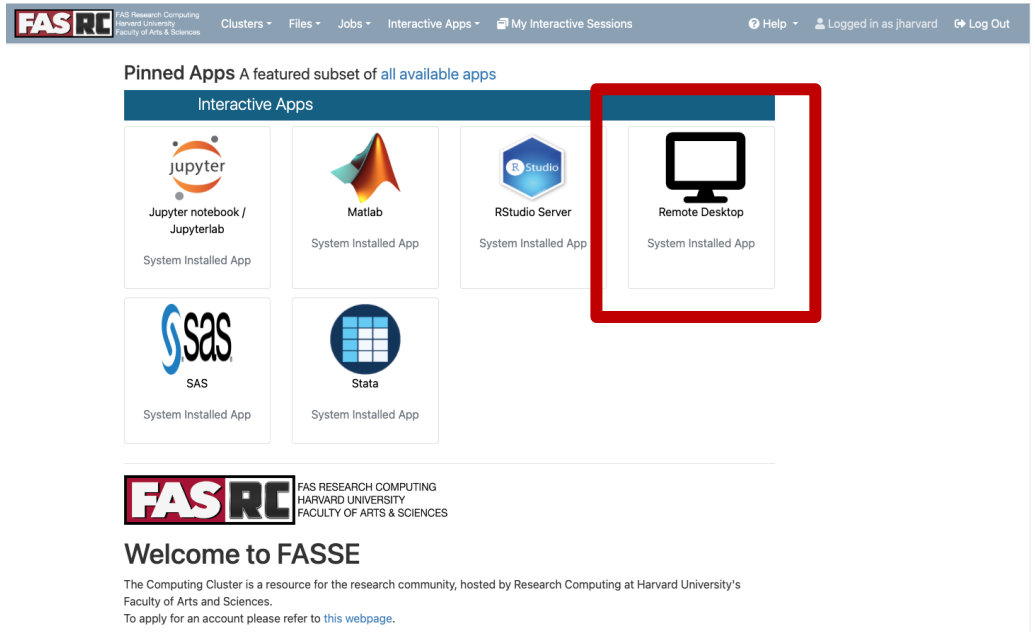
```
[jharvard@holy7c02410 ~]$
```

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 stats summary – `jobstats`

Documentation:
https://docs.rc.fas.harvard.edu/kb/jobstats/

- Run a test batch job
- Check job stats after the job has completed (with `jobstats` command)
- Overall section: Summary "bar" plot
- Detailed section
- Notes (where warnings appear)
- For email summary, add to your batch script `#SBATCH --mail-type=END`

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
#SBACTH --mail-type=END # Emails jobstats summary
```

jobstats – CPU job

```
[jharvard@holylogin08 ~]$ jobstats 44768161

================================================================================
                               Slurm Job Statistics
================================================================================
         Job ID: 44768161
   User/Account: jharvard/jharvard_lab
       Job Name: stress-ng
          State: COMPLETED
          Nodes: 1
      CPU Cores: 112
     CPU Memory: 1014833MB (9.1GB per CPU-core)
  QOS/Partition: normal/sapphire
        Cluster: odyssey
     Start Time: Tue Nov 4, 2025 at 1:35 PM
       Run Time: 2-23:00:29
     Time Limit: 2-23:30:00

                            Overall Utilization
================================================================================
  CPU utilization  [|||||||||||||||||||||||||||||||||||||||||||||||||97%]
  CPU memory usage [|||||||||||||||||||||||||||||||||||||||||||||||||98%]

                            Detailed Utilization
================================================================================
  CPU utilization per node (CPU time used/run time)
      holy8a28606: 323-02:24:06/331-08:54:08 (efficiency=97.5%)

  CPU memory usage per node - used/allocated
      holy8a28606: 971.4GB/991GB (8.7GB/8.8GB per core of 112)

                                    Notes
================================================================================
  * Have a nice day!
```

Summary “bar” plot

Detailed

Notes

jobstats – GPU job

```
[jharvard@boslogin08 ~]$ jobstats 47914764

================================================================================
                              Slurm Job Statistics
================================================================================
         Job ID: 47914764
   User/Account: jharvard/jharvard_lab
       Job Name: scan_kernel
          State: COMPLETED
          Nodes: 1
      CPU Cores: 32
     CPU Memory: 200GB (6.2GB per CPU-core)
           GPUs: 1
  QOS/Partition: normal/gpu_h200
        Cluster: odyssey
     Start Time: Tue Nov 25, 2025 at 10:52 AM
       Run Time: 02:59:53
     Time Limit: 1-00:00:00
```

Summary “bar” plot

```
                           Overall Utilization
================================================================================
  CPU utilization  [|                                                      3%]
  CPU memory usage [                                                       0%]
  GPU utilization  [||||||||||||||||||||||||||||||||||||||||||||||||||||100%]
  GPU memory usage [|||||||||||||||||                                     31%]
```

Detailed

```
                           Detailed Utilization
================================================================================
  CPU utilization per node (CPU time used/run time)
      holygpu8a12103: 03:00:12/3-23:56:16 (efficiency=3.1%)

  CPU memory usage per node - used/allocated
      holygpu8a12103: 431.3MB/200GB (13.5MB/6.2GB per core of 32)

  GPU utilization per node
      holygpu8a12103 (GPU 1): 100%

  GPU memory usage per node - maximum used/total
      holygpu8a12103 (GPU 1): 44.0GB/140.4GB (31.3%)

                                    Notes
================================================================================
  * The max Memory utilization of this job is 0%. This value is low compared
    to the target range of 80% and above. Please investigate the reason for
    the low efficiency. For more info:
      https://docs.rc.fas.harvard.edu/kb/job-efficiency-and-optimization-best-
practices/#Memory

  * Have a nice day!
```

Notes

warning

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/<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/<PI_LAB>/Lab/conda/<ENV_NAME> PACKAGE_LIST

# Activate the conda environment
mamba activate /n/holylabs/<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

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

Thank you :)

FAS Research Computing

Hands-on

Training Material

```
# Login to Cannon
ssh <username>@login.rc.fas.harvard.edu

# copy FASRC CLI examples:
cd /n/holylabs/<your_lab>/Lab/<username>
cp -r /n/holylabs/jharvard_lab/Everyone/user_training/intro_cli .
cd intro_cli
```

Hands-on

1) Run hello_world.py in an interactive session
    a) salloc --partition=test --time=30:00 --mem=500M
    b) module avail python
    c) module load python/XXXX
    d) python hello_world.py
    e) 'exit' interactive session

1) Run hello_world.py as an sbatch job
    a) edit runscript.sh
    b) sbatch runscript.sh

1) Use SLURM command line tools to evaluate job performance and efficiency
    a) sacct
    b) jobstats
    c) seff