# Rosalind Getting Started

## Important Notices

For technical support, please email hpcsupport@ucdenver.edu

This is not a HIPAA compliant system. Please do not use it to store or process PHI.

Currently, nothing is backed up. Please make sure that you have your own backup of any data you keep on the HPC.

We are targeting both HIPAA compliance and backup of selected data within the next few months.

Getting Started Guide version 1.5

## Cluster Specifications

- Quantity 32 Linux compute nodes, each with 24 CPU cores and 128 GB of RAM
- Quantity 2 Linux high memory node, with 36 CPU cores and 1536 GB of RAM
- FDR InfiniBand high speed networking for parallel computing and storage interconnect
- 3.7 PB usable shared storage (DDN GS14Ke with approx. 650 hard drives)
- Red Hat Enterprise Linux 7.3 operating system, SLURM job scheduler

## Logging In

The cluster can only be reached by using a jump host, which is an intermediary system which provides enhanced security by sitting between campus networks and the Rosalind HPC cluster. The jump host is only reachable from campus networks and the campus VPN (but not directly from the internet)

To log in to the jump host, you will need a SSH client. If you have a Linux or Mac computer, open a terminal and log in using:

```
ssh username@cubipmaccess.ucdenver.pvt
```

Where username is your CU "short" username. (example: jonesp, not paul.jones@ucdenver.edu)

If you have a Windows computer, you will have to download a SSH client first. One simple option is PuTTY, downloadable here

If you are using Putty on Windows, enter the jump host's address, `cubipmaccess.ucdenver.pvt` , into the Host Name blank and press Open to begin.

Once you are logged into the jump host, all you need to do to log into the HPC cluster is to type `ssh hpc` at the jump host's command prompt.

## Transferring data

Similarly, to transfer data into the Rosalind HPC storage system, you must use a data transfer jump host. To do this, use a SFTP client to make a SFTP protocol connection to `cubipmsftp.ucdenver.pvt`. Once connected, you will see a directory named after your username. You may upload your data into this directory, and it will appear at /gpfs/transfer/username on the Rosalind CLI, where username represents your CU username. Please move any data you have uploaded from the transfer area to your home directory or project directory as soon as possible. This ensures that the security function of the data transfer jump host remains effective. The transfer area will never be backed up and data may be removed from it any time, since it is not intended as a location for permanent data storage.

## Home and Scratch

By default, the shell variables `$HOME` and `$SCRATCH` refer to your home directory and your scratch area. You can use these just like normal file paths. For example:

```
mkdir $HOME/newdir

echo "test" > $SCRATCH/testfile
```

It is always a good idea to refer to your home directory using `$HOME` instead of hard coding it's path into your code. This way, if your home directory is moved, your code will still work, and it will also work for other people you give it to.

`$SCRATCH` points to storage that is much faster than the local disks on the compute nodes. Instead of writing temporary data to `/tmp` on a node, write it to `$SCRATCH` for better performance.

`$SCRATCH` is temporary storage. It should never be used to store data you intend to keep beyond the runtime the job that's using the data. In the future, we may automatically delete files from `$SCRATCH` that haven't been accessed within a week or so. `$SCRATCH` will never be backed up.

By default, your disk quotas in home and scratch are:

- $HOME - 100 GB and 150,000 files (whichever comes first)

- $SCRATCH - 1000 GB and 150,000 files (whichever comes first)

Additionally, you will be notified in your welcome letter of a storage location for your project. This area can be shared between members of your project and has, by default a limit of 100,000 files and unlimited GB.

Please do not, in any situation, set permissions on any of your directories to world-accessible. If you need to share data, please contact hpcsupport@ucdenver.edu with your request to share to avoid HIPAA violations.

World accessible permissions are monitored on your top-level directories. If they are set to world accessible, we will receive an alert and our standard procedures require that we investigate.

## Running Jobs

In order to run a job, you must make a submit script for the SLURM job queueing software. The purpose of the submit script is to tell SLURM how to run your job, including how many resources it requires. Below, a minimal submit script:

```
#!/bin/bash

#SBATCH --export=ALL
#SBATCH --time=60
#SBATCH --ntasks=1
#SBATCH --mem=1000
#SBATCH --job-name=MyJob
#SBATCH --output=MyJob.log

date
sleep 60
```

To submit the script above as a job, first save it to a text file on Rosalind. For example, you could save it to MyJob.sh. Then, while logged in, submit it with sbatch:

```
[you@cubipmlgn01 ~]$ sbatch ./myjob.sh
```

To check to see if it is running, you can use squeue, and to check status on it, you can use use sacct.

```
[you@cubipmlgn01 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```

```
             15852      defq    MyJob      you PD       0:00      1 (Priority)

[you@cubipmlgn01 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             15852      defq    MyJob      you  R       0:48      1 cubipmcmp028

[you@cubipmlgn01 ~]$ sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
15852              MyJob       defq        you          1  COMPLETED      0:0
15852.batch        batch                   you          1  COMPLETED      0:0

[you@cubipmlgn01 ~]$ sacct --format=jobid,elapsed,ncpus,ntasks,state
       JobID    Elapsed      NCPUS   NTasks      State
------------ ---------- ---------- -------- ----------
15852          00:01:01          1           COMPLETED
15852.batch    00:01:01          1        1  COMPLETED
```

The manual pages on Rosalind for sacct, squeue, and sbatch contain a lot of useful information. Don't forget to try typing `man sbatch`, `man squeue`, and `man sacct`

Explaining the above submit script one line at a time:

```
#!/bin/bash
```

Your submit script is a Linux shell script that is put together in a specific way that SLURM understands. Since it's a regular shell script, it begins with a call to a shell script interpreter, in this case, bash.

```
#SBATCH --export=ALL
```

This line causes all of your login shell's environment variables to be visible to the programs running in your submit script.

```
#SBATCH --time=60
```

This line tells SLURM to give a time limit of 60 minutes to your job. Setting time limits is important because jobs with time limits will be allowed to run sooner than jobs with the default limit, which is 36 hours.

In a shell script, # denotes a comment that is normally not executed. However, SLURM recognizes lines that begin with #SBATCH as instructions to SLURM, In general, if you would pass an option like the –time option to SLURM on the command line, you can put it in your submit script on a line beginning with #SBATCH and it will have the same effect.

```
#SBATCH --ntasks=1
```

This line tells SLURM that you are requesting one task for your job, which on our system means requesting one CPU core. No matter how many threads your job has, it will only be allowed to run on the number of cores you request. You may request fewer cores than your job has threads, with no ill effect aside from the possibility of your job running more slowly.

```
#SBATCH --mem=1000
```

This line tells SLURM that you are requesting 1000 megabytes of memory for your job. The default, if you do not specify, is about 4 GB, but requesting smaller amounts can cause your job to be scheduled to run sooner. If you need more than 4 GB, you must explicitly request it. The maximum you can request on one node is about 100 GB. Be sure to request enough memory for your job, since a job will fail when it tries to use more memory than you requested.

```
#SBATCH --job-name=MyJob
#SBATCH --output=MyJob.log
```

These lines give your job a name and a file to send (by default) the job's stdin and stderr output streams to. If you don't give your job a name or an output file, it will pick generic defaults, which may make it inconvenient to track the progress of your job.

```
date
```

This line begins the part of the submit script which contains the actual Linux programs SLURM will run as part of your job. In this case, the job will simply run the date command and exit. Date's output will be written to MyJob.log

## Running R

A recent version of R is available in the default environment. Specifically, we currently provide the latest R provided by the EPEL Linux packages repo.

This copy of R is updated whenever EPEL ships an update (roughly, a few times per year). If you need to lock your R version to a particular release, we highly recommend installing a copy of R yourself in your home directory.

You may install CRAN packages into a personal library as needed - when you run a package install inside R, just say "yes" to using a personal library.

## Running Python

Although we provide some common Python libraries in the default environment, we recommend using the Intel Python Distribution instead. Both Intel Python and EPEL Python are updated whenever the vendor releases and update. If you need to lock your verson of Python, Scipy, Numpy, or any other library to a particular version, we highly recommend installing a copy yourself in your home directory.

Intel Python provides the `conda` tool in order to create custom Python environments that contain the Python libraries you need. It provides many more libraries than the default Python.

To enable Intel Python in your current session (or in a submit script)

```
source /opt/intel/intelpython2/bin/activate
```

You may replace the "2" above with a "3" to activate Python 3 instead.

To build a new Conda environment (you only need to do this once, unless you want multiple environments)

```
conda create -n MyPython numpy scipy   #do this just once. MyPython is just an arbitrary nam
```

This is just an example. If you don't need numpy or scipy, you should omit them. Other available libraries include tensorflow, scikit-learn, keras, and many more.

Once you have built an environment, you can enter it directly like so:

```
source /opt/intel/intelpython2/bin/activate MyPython
```

Much more info is available here

## Running MPI Jobs

In order to run MPI jobs, you only need to load a MPI module and specify the desired number of nodes and cores in your submit script. Example submit script:

```
#!/bin/bash

#SBATCH --time=60
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=24

module load mpich/ge/gcc/64/3.2rc2
```

```
module load slurm

# using srun here allows srun to set up the MPI environment for you
srun my_mpi_program
```

Optionally adding `#SBATCH --open-mode=append` would mean that your output log file will not be overwritten by subsequent runs of the same job. Instead, new results will be added to it.

## Best Practices

- The ideal batch job is longer than a few minutes, but shorter than a day. Jobs in this range of lengths will be scheduled more efficiently by SLURM, and so you will get more work through the system and, on average experience less queue wait time if you keep your jobs within these limits.
- Batch jobs should be able to be killed and restarted without losing too much progress. You can accomplish this by writing your code to checkpoint, or simply by breaking up a long running job into multiple shorter jobs.
- Do not hard code the UNIX path of your home directory into your SLURM scripts. If you need to reference a file in your scripts, use the shell's `$HOME` variable.
- Try it: Type `echo $HOME`
- If you have a file called myfile in the top level of your home, you can reference it as `$HOME/myfile`