About this tutorial, and how to follow it

This tutorial assumes you are already comfortable enough in the NYU HPC environment to login, run jobs and move data to and from the cluster, and that you have set up and are using an SSH tunnel to access the cluster. If you are new to HPC, we recommend working through Tutorial 2 first. And if you have not used the command line before, we recommend starting at Tutorial 1.

This tutorial is run as a regular class in the library (see the calendar on the NYU HPC Wiki front page for times) but can also be used for self-directed learning. Much of the tutorial is a walk through relevant pages of the NYU HPC Wiki, interspersed with additional explanation and exercises to try - so have a terminal window open to try things as you go.

The panels on the left are for navigation through the tutorial. There is more material here than we will have time to go over in the class, so the essential topics are indicated with bold links in the navigation panels.

An example HPC Workflow

Here is a (slightly contrived) example of an HPC workflow: (even if yours is completely different, it probably shares some characteristics with this example)
Some things to note about it:

- The blue boxes are compute jobs. This is why you are using the HPC cluster - to perform either large or many computational tasks, more than is practical on a workstation.
- You have input data and output data to manage, as well as source code and configuration files.
- A significant portion of the workflow is running *almost the same task*, many times.
- The analysis cannot begin until all of the experiments are complete.
- You have source code, input data and result data, all of which must be managed appropriately.
- Certain things are to be shared with colleagues.

HPC components supporting each aspect of the workflow are:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing and storing source and configuration files</td>
<td>These are typically small, are used only once in a given run, and ought to be version controlled and backed up. At NYU, the $HOME filesystem is good for this. We also recommend using some version control such as git.</td>
</tr>
<tr>
<td>Getting input data onto the cluster</td>
<td>scp, rsync and wget are all options. This data should normally be put into $SCRATCH while being used, and may be copied to $WORK to prevent it being deleted after a period of non-use.</td>
</tr>
<tr>
<td>Making files (configuration, results, etc) accessible to colleagues</td>
<td>At NYU, $SCRATCH supports FACLS (&quot;file access control lists&quot;), for files on $HOME, $WORK or $ARCHIVE use chmod.</td>
</tr>
<tr>
<td>Running a set of independent experiments</td>
<td>Job arrays are a powerful and convenient way to manage large sets of similar jobs.</td>
</tr>
<tr>
<td>Running an analysis job after all experiments are complete</td>
<td>Torque, the batch system used at NYU, supports job dependencies, including arrayjob dependencies, so you can automate your workflow.</td>
</tr>
</tbody>
</table>
Archival of results

At NYU, $ARCHIVE is a good place to store important results. But $ARCHIVE should not be used for large numbers of small files. For more on using $ARCHIVE, see Storage.

In this tutorial, we will work through each aspect of the workflow, and how to use it.