Yeti - Submitting Jobs

- **Batch Jobs**
  - **Interactive Jobs**
  - Jobs with Graphical User Interface
- **Basic Job Directives**
  - Environment Variables
- **Advanced Job Requests**
  - Multiple copies of the same job - job arrays
  - Requesting more than one processor
  - Selection of specific types of nodes

Yeti has two main types of queues that accept job submission: the batch queue and the interactive queue. To run software applications on Yeti, users should submit jobs to one of these two queue types, according to the instructions below. Torque and Moab, which are the two middleware components currently running on the cluster, take care of job scheduling.

### Batch Jobs

Batch jobs are jobs that do not interact with the user during their execution. They are submitted once and are channeled to the batch queue (which actually consists of several queues for optimization).

The batch queue has a maximum walltime of 5 days for jobs of 16 GB or less and 3 days for larger jobs. If a job fails to specify either of these settings the defaults are 30 minutes walltime and 1 GB of memory. The specifications of the various settings and invocations of application software to be executed are contained in a submit script created/edited by the user. The general way to submit jobs to the batch queue is to use the Torque "qsub" command:

```
$ qsub submit_script.sh
```

where submit_script.sh contains job information (along with application invocation), as described below.

All jobs are submitted from the submit node (yetisubmit.cc.columbia.edu). An example of a simple script and how to submit it with the qsub command can be found in the Getting Started section of this documentation. More advanced examples can be found in the Job Examples page.

You should redirect output and error files to anywhere you have write permission in the /vega tree.

### Interactive Jobs

Interactive jobs, which are specified with the `-I` flag on the qsub command, allow user interaction during their execution. They deliver to the user a new shell from which applications can be launched. Interactive jobs are automatically channeled to the interactive queue.

The maximum walltime for the interactive queue is 4 hours (04:00:00), and the default walltime is one half hour (00:30:00). Users may have a maximum of four concurrent interactive jobs running. The general way to submit jobs to the interactive queue is:

```
$ qsub -I -W group_list=<GROUP> -l walltime=01:30:00,mem=1000mb
```

where `-I` is required and specifies an interactive job and `-W group_list` is also required and identifies the group a user belongs to. Most of the Torque directives noted below that begin with #PBS are available on the command line as well for interactive jobs. If a node is available, you will see a command line prompt seconds after pressing enter. If no nodes are available, your current shell will wait.

### Jobs with Graphical User Interface

Applications using Graphical User Interface can, by their nature, only be run via interactive jobs.

For information about how to set up a GUI, see the Graphical User Interfaces in Yeti page.
## Basic Job Directives

The following options may be found in a Torque job submit script (as lines in the script preceded by the "#PBS" string as in the examples below) or on the command line with qsub (as arguments to the qsub command). The latter syntax is the only way to specify them for interactive jobs. The options are presented roughly in the order in which they typically appear in a batch submit script, including the examples in this documentation. The first few are either mandatory or otherwise need to be considered and decided upon.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>#PBS -q queue name</td>
<td>Assigned queue</td>
<td>#PBS -q infiniband</td>
<td>Used for interactive jobs, as in the Example. When absent, jobs get directed onto a batch queue.</td>
</tr>
<tr>
<td>#PBS -N job name</td>
<td>Assigned job name</td>
<td>#PBS -N DateAndTime</td>
<td>Default: Torque job submit script.</td>
</tr>
<tr>
<td>#PBS -W group list</td>
<td>Group associated with the job</td>
<td>#PBS -W group list=yetistats</td>
<td>See the table of submit groups for possible values.</td>
</tr>
<tr>
<td>#PBS -l mem=RAM size</td>
<td>Maximum physical memory job</td>
<td>#PBS -l mem=200mb</td>
<td>Default: 1024mb (1 GB). Job terminated if exceeded.</td>
</tr>
<tr>
<td></td>
<td>requires in MBytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#PBS -l walltime</td>
<td>Maximum wall time for job.</td>
<td>#PBS -l walltime=12:00:00</td>
<td>Default: 30 minutes.</td>
</tr>
<tr>
<td></td>
<td>Format: DAYS:HRS:MIN:SEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#PBS -l nodes=node:ppn</td>
<td>Number of nodes and number</td>
<td>#PBS -l nodes=1:ppn=4</td>
<td>Default: nodes=1,ppn=1.</td>
</tr>
<tr>
<td></td>
<td>of processors per node (ppn)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#PBS -M email</td>
<td>Email address for job status</td>
<td>#PBS -M <a href="mailto:hpc2108@columbia.edu">hpc2108@columbia.edu</a></td>
<td>Recommended for batch jobs</td>
</tr>
<tr>
<td></td>
<td>messages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#PBS -m a/b/e</td>
<td>Codes for email notification</td>
<td>#PBS -m a/b/e</td>
<td>Recommended for batch jobs; a: abort, b: begin, e: end.</td>
</tr>
<tr>
<td></td>
<td>(1-3 can be specified)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#PBS -m n</td>
<td>Turn off all e-mail.</td>
<td>#PBS -m n</td>
<td>Used e.g. for tested jobs</td>
</tr>
<tr>
<td>#PBS -V</td>
<td>Environment variable control.</td>
<td>#PBS -V</td>
<td>Exports all environment variables to the job</td>
</tr>
<tr>
<td>#PBS -o localhost:&lt;path&gt;</td>
<td>Path to &lt;jobname&gt;.o&lt;jobid&gt;</td>
<td>#PBS -o localhost:/vega/stats/users/hpc2108/output</td>
<td>Standard output</td>
</tr>
<tr>
<td></td>
<td>file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#PBS -e localhost:&lt;path&gt;</td>
<td>Path to &lt;jobname&gt;.e&lt;jobid&gt;</td>
<td>#PBS -e localhost:/vega/stats/users/hpc2108/output</td>
<td>Standard error</td>
</tr>
<tr>
<td></td>
<td>file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#PBS -j oe</td>
<td>Merge option</td>
<td>#PBS -j oe</td>
<td>Merge error messages with regular output.</td>
</tr>
<tr>
<td>#PBS -t &lt;sequence&gt;</td>
<td>Job array syntax</td>
<td>#PBS -t 1-4</td>
<td>Submit multiple copies of same job</td>
</tr>
</tbody>
</table>

Your script may also include comments in the form of lines that start with # (without the letters PBS following).

**Temporary note on mail address**

The email address which is the parameters to the -M flag above must currently be a Columbia address in the format <UNI>@columbia.edu. Outside addresses are not accepted - but this is something we expect to be allowed in the not too distant future.

**General note on Output**

Torque usually generates two types of files, error files and output files, during job execution. The paths to store these files are controlled by the "#PBS -e" and "#PBS -o" directives. In addition to specifying these options, users can redirect program-specific output using ">" or ">>" as shown in the examples below.

## Environment Variables

Additional environment variables are available in scripts and jobs. Some of the most useful are:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
</table>
**Advanced Job Requests**

**Multiple copies of the same job - job arrays**

To submit multiple copies of your job, and have each run on any available processor, use the following notation in the submit script:

```bash
#PBS -t 1-4
```

This can also be accomplished by using the "-t 1-4" directive on the command line with `qsub`. Job arrays make available to each job an additional environment variable called `$PBS_ARRAYID`. To vary an input parameter based on the job number, you may reference `$PBS_ARRAYID` in your job script.

**Important note about array jobs**

Please be mindful that your submit script should describe the walltime and memory requirements for ONE of the array jobs. The `-t` flag is merely a way to avoid having to submit several of the same jobs individually. So if one of the array processes uses 300 MB RAM on one node and takes 45 minutes, the `-t` flag should read "-l nodes=1,mem=300m,walltime=00:45:00".

**Email in array jobs**

Large array jobs can generate a lot of email. Please consider turning off email for your array jobs by setting "#PBS -m n" in your submit file.

**Requesting more than one processor**

To request 4 processors on 1 server:

```bash
#PBS -l nodes=1:ppn=4
```

To request 16 processors on each of 2 servers (32 cores total):

```bash
#PBS -l nodes=2:ppn=16
```

Note that all servers on Yeti have 16 cores. For reasons of efficiency we recommend that you do not request more than 1 node unless you require more than 16 cores.
Selection of specific types of nodes

Yeti’s execute nodes can be divided into various categories, according to the physical attributes of the servers which act as nodes within the clusters. Users can specify categories of nodes they want to use in their jobs. However, there are certain rules which restrict the types of nodes that can be used within one job. Categories typically correspond to the different “generations” of nodes on the cluster: e.g. the original (“v1”) nodes from the Oct 2013 Yeti launch and nodes added during the expansion in Feb 2015 (“v2”).

For every category there is a corresponding label which can be specified in the submit file, in the all-important parameter to the “-l” flag. For example, the line:

```
#PBS -l nodes=2:ppn=16:v2
```

requests 2 of the newer nodes, and all 16 cores on each of the nodes.

The following are the current categories and their corresponding labels. When two labels appear on the same line, they have the same meaning and are otherwise equivalent.

```
First generation CPUs: v1, e5-2650l
Second generation CPUs: v2, e5-2650v2

Infiniband nodes (either generation): ib
Infiniband nodes (newest generation): ib2

GPU nodes (either generation): gpu
GPU nodes (older generation): gpu1, k20
GPU nodes (newest generation): gpu2, k40
```

One job can run on nodes of only one generation of CPU (i.e. “v1” or “v2”). If no generation is specified, the scheduler will select one based on system availability.