# LQTS Documentation

## 1. Introduction

LQTS (pronounced Locutus) is a *L*ightweight job *Q*ueueing *S*ystem. Its purpose is to manage the execution of a large number of computational jobs, only exeucuting a fixed amount of them concurrently. LQTS is intended to be very easy to run. There is no setup required, but some options may be controlled through environment variables or a .env file.

If you are familiar with PBS queueing system on Linux, this should be very familiar.

The major concepts involved in this are:

```
* Job Queue - a list of jobs to be executed
* Job Submission - putting a job in the queue
* Job Server - the process that gets submissions, manages the queue,
  and executes the jobs
* Worker - a child process of the server that executes the job
```

## 2. Starting the Job Server

LQST is a client/server app. The server runs on localhost (IP address is 127.0.0.1) and port 9200. The server has process pool and a queue that it uses to manage jobs that are submitted to it. The server is started using the qstart.exe program.

```
$ qstart.exe
```

On start up the server should display

```
Starting up LoQuTuS server - 1.0.8+5bc479c
Worker pool started with 32 workers
Visit http://127.0.0.1:9200/qstatus to view the queue status
```

By default there are (# CPUs - 2) workers avaialable. This can be queried by calling `qworkers.exe` with no arguments. The number of workers can be dynamically adjusted by calling `qworkers.exe` with one argument that is the number of workers desired.

## 3. Server Configuration

Ther server has several settings that can be adjusted by setting environment variables or by starting the server in a directory where a file name `.env` is present. Unless you have a specific requirement, it is not recommended to change these from their defaults. The available settings are:

- LQTS_PORT - The port the server is bound to (change if you want to run mulitple instances)
- LQTS_NWORKERS - Maximum number of workers/concurrent jobs
- LQTS_COMPLETED_LIMIT - Number of completed jobs the server "remembers" (i.e. that would show up in qstat)
- LQTS_RESUME_ON_START_UP - Whether or not to attempt to resume the job queue based on the contents of the LQTS_QUEUE_FILE (It is recommended not to set this to true currently, as it can be flakey.)
- LQTS_QUEUE_FILE – location of the file where LQTS writes its current queue every few minutes

## 4. Job Submission

Several commands are available for different use cases to submit jobs to the queue. Each qsub command returns the job ID for the submitted job(s).

- qsub - submit one job
- qsub-multi - submit multiple commands to the queue
- qsub-cmulti - submit mutliple jobs to the queue, where the command is the same but the input files are different
- qsub-argfile - submit mutliple jobs to the queue, where each job is defined by a line if the given file

### qsub

The `qsub` command submits one job to the queue.

Example:

‘ qsub –help Usage: qsub COMMAND [qsub ARGS] – [COMMAND ARGS]

Submits one job to the queue

qsub ARGS: –priority INTEGER [default: 1] –logfile TEXT Name of log file [default: ] –log Create a log file based on the command name [default: False]

-d, –depends LIST Specify one or more jobs that these batch of jobs depend on. They will be held until those jobs complete [default: <class 'list'>]

–debug Produce debug output [default: False] –walltime TEXT A amount of time a job is allowed to run. It will be killed after this amount [NOT IMPLEMENTED YET]

–cores INTEGER Number of cores/threads required by the job [default: 1]

–port INTEGER The port number of the server [default: 9200] –ip_address TEXT The IP address of the server [default: 127.0.0.1] -a, –alternate-runner Runs the submitted command in a slightly different manner. In rare cases an executable can start, then hang. However, the log file isn't updated until the process terminates. [default: False]

–help Show this message and exit. ‘

## qsub-cmulti

The `qsub-cmulti` command submits multiple jobs. This command is used you have one command with multiple input files, each of which is a different job. You should be able to reference your input files with a glob pattern (such as "*.inp")

Example:

```
$ qsub-cmulti myprogram.exe *.inp
```

Example2:

```
$ qsub-cmulti myprogram.exe *.in -- -arg1 --arg2 Arg2Value
```

### Full Command Help

```
Usage: qsub-cmulti COMMAND FILE_PATTERN [qsub-cmulti ARGS] -- [COMMAND args]

  Submits mutlitiple jobs to the queue.

  Runs **command** for each file in **files**.  Pass in args.

  $ qsub mycommand.exe MyInputFile*.inp  -d 2 --log -- --do --it
         [-----------] [--------------]  [--------]    [----------]
           command       filepattern    qsub ARGS     command args

Options:
  --priority INTEGER      [default: 1]
  -d, --depends TEXT      Specify one or more jobs that this batch of jobs
                          depends on. They will be held until those jobs
                          complete  [default: <class 'list'>]

  --debug                 [default: False]
  --log                   Create a log file for each command submitted
                          [default: False]

  --cores INTEGER         Number of cores/threads required by the job
                          [default: 1]

  --port INTEGER          The port number of the server  [default: 9200]
  --ip_address TEXT       The IP address of the server  [default: 127.0.0.1]
  -a, --alternate-runner  Runs the submitted command in a slightly different
                          manner.  In rare cases an executable can start, then
                          hang.  However, the log file isn't updated until the
                          process terminates.  [default: False]
  --help                  Show this message and exit.
```

### qsub-multi

The `qsub-multi` commnad submits multiple jobs. This command is used if you have multiple executables or scripts that you want to submit. You should be able to reference this executables or scripts with a glob pattern such as *.bat.

Example:

```
$ qsub-multi myprograms*.exe
```

### qsub-argfile

The `qsub-argfile` command submits mutliple jobs. For this command you need one executable file and a text file. Each line in the text file is represents a different job and has the arguments that will be passed to your executable command.

For example, imagine you have a script `echoit.bat` and its contents are

```
echo $1
sleep 2
```

Also there is an `argfile.txt` file:

```
jim
dwight
pam
MICHAEL_SCOTT
```

Here you want to pass each line separately `echoit.bat`

```
$ qsub-argfile echoit.bat argfile.txt --log
```

Specifying `--log` will case LQTS to write a log file for each job. The log file contains any output that would have been written to the screen as well as some job performance information.

## 5. Passing Additional Arguments to your Command

After the command and input file/arg file, arguments are typically interpreted as being arguments passed to the qsub command itself and are not passed on to your program. Like many Linux commands, specifying a double dash "–" will allow you to specify additional arguments you want passed to your command.

```
$ qsub myprogram.exe --log -- --flagForMyProgram true
```

Here "`--log`" is consumed by the qsub command and "`--flagForMyProgram true`" is passed along to myprogram.exe when it is executed.

## 6. Waiting for a job to complete

The `qwait.exe` command will block until the specified jobs have completed. If you have something you want to happen once a set of jobs has completed, use this. You can specifies the job IDs as arguments to `qwait`, or it will read them from stdin. This means you can pipe the output of the `qsub` commands directly into `qwait`.

Examples:

```
$ qsub myprogram.exe --log -- --flagForMyProgram true
10
$ qwait 10
```

Alternatively:

```
$ qsub myprogram.exe --log -- --flagForMyProgram true | qwait
```

## 7. Job Priority

Each job has a priortity associated with it. Jobs with higher priority are executed before jobs with lower priority. The `qsub` commands support a `--priority` argument. There is also the `qpriority` command that allows you to change the priority of a job after it has been submitted.

## 8. Deleting a Job

Use the `qdel` command to delete a job.

## 9. Job Status and Summary

The `qstat` and `qsummary` commands provide information about the state of jobs in the queue. `qstat` provides information about each job and `qsummary` simply tells you how many jobs are running and how many are queued.

You can also navigate to http://127.0.0.1:9200 (where 9200 is the port you have LQTS running on) to see an HTML table of the current status. This page auto-refreshes every two minutes.

## 10. Testing your setup

`qsub-test` provides an easy way to test that LQTS is functioning properly. It writes a script that sleeps for a user defined amount of time and an argfile with a user defined number of jobs.

The following example will write a script that sleeps for 5 seconds and submit 30 jobs to the queue.

```
$ qsub-test 5 --count 30
```