1) First serial job
   a) Submit a serial job that:
      i) Is a serial (1 core) job
      ii) Emails you when is starts, ends and aborts
      iii) Has a maximum wall time of 2 minutes
      iv) Runs the 'hostname' command
   b) Make a note of the jobid when your job is submitted
   c) Watch your job run with the following command:
      i) "qstat -a -u $USER"
      ii) "showq –u $USER"
      iii) "jobinfo –j"
   d) Did you get the result emailed to your account
   e) Display the job output and error files.
   f) Examine the emails you have received.

2) Second serial job
   a) Submit a serial job that:
      i) As in part 1
         - Is a serial (1 core) job
         - Emails you when is starts, ends and aborts
         - Has a maximum wall time of 2 minutes
         - Runs the 'hostname' command
      ii) Runs the non existent command 'hello'
   b) Make a note of the jobid when your job is submitted
   c) Watch your job run with the following command:
      i) "qstat -a -u $USER"
      ii) "showq –u $USER"
      iii) "jobinfo –j"
   d) Did you get the result emailed to your account
   e) Display the job output and error files.
   f) Examine the emails you have received.

3) Third serial job
   a) Submit a serial job that:
      i) Is a serial (1 core) job
      ii) Emails you when is starts, ends and aborts
      iii) Has a maximum walltime of 2 minutes
      iv) Sleeps for 200 seconds
   b) Think about what will you think happen when this jobs runs?
      i) "qstat -a -u $USER"
      ii) The job output and error files.
      iii) The emails.
   c) Run your job and see what happens
      i) "qstat -a -u $USER"
      ii) The job output and error files.
      iii) The emails.

4) Forth serial job (Job Names)
   a) Submit a serial job that:
      i) Is a serial (1 core) job
      ii) Has a maximum wall time of 2 minutes
      iii) Sleeps for 30 seconds
      iv) Is named "my-4th-job"
   b) Look at your job in the output of the following commands
      i) "showq –u $USER"
      ii) "qstat –a –u $USER"
      iii) "jobinfo –j"

5) Interactive serial Job
   a) Open a second ssh session/terminal to the workshop cluster
   b) In the second ssh session submit a job that:
      i) Is a serial (1 core) job
      ii) Has a maximum wall time of 20 minutes
      iii) Emails you when the job is aborted, before it runs and a after it ends
      iv) Is named "my-first-interactive-job"
      v) Is interactive
   c) Wait for the job to start, then after the job starts:
      i) Find out on which node does your job ran on?
      ii) Look at the command line you are on.
      iii) Look at the PBS environment variables
         Hint: "printenv | grep -i pbs"
      iv) What is the jobs name?
         Hint: "echo $PBS_JOBNAME"
      v) What is the jobs id?
         Hint: "echo $PBS_JOBID"
      vi) Which directory is the job in?
         Hint: "pwd" command
      vii) Which directory has the job been submitted from?
         Hint: "echo $PBS_O_WORKDIR"
      viii) What is the path to executable for this job?
         Hint: "echo $PBS_O_PATH"

   d) Optional for advanced Unix users: In the interactive job
      i) (Advanced Unix) List the jobs cpuset. A cpuset is used to assigning a set of processor and memory to a set of processes, and can be used by the scheduling system to keep a job from using resources assigned to other jobs.
         Hint: "ls /dev/cpuset/torque/$PBS_JOBID"

      ii) (Advanced Unix) Verify that your current shell's process id is inside the cpuset.

Hint: “echo $$” to find the process id of your current shell
Hint: “ cat /dev/cpuset/torque/$PBS_JOBID/tasks”
to list processes in the jobs cpuset.

iii) (Advanced Unix) Show which cores and which memory set that your job is running on?
Hint: “cat /dev/cpuset/torque/$PBS_JOBID/cpus”
for core numbers that the job processes are running on.
Hint: “cat /dev/cpuset/torque/$PBS_JOBID/mems”
for memory location(s) that the job memory may run on

e) The current interactive job only uses 1 process but for future comparison with other more complex job types run the following commands and write down the results.
   i) Give a list of node names where each process (there is only one in this case) of this job runs?
   Hint: “cat $PBS_NODEFILE”
   ii) On how many nodes (there is only one in this case) does this job run?
   Hint: “echo $PBS_NUM_NODES”
   iii) On how many processors (there is only one in this case) does this job run?
   Hint: “echo $PBS_NP”
   iv) On how many processors per node (there is only one in this case) does this job run?
   Hint: “echo $PBS_NUM_PPN”
   v) What is the array ID of this job?
   Hint: “echo $PBS_ARRAYID”
   Answer: In this case there is none.

f) From your first terminal on the login node look at your job in the output of the following commands how does it compare:
   i) “showq –u $USER”
   ii) “qstat –a –u $USER”
   iii) “jobinfo –j”
g) In your interactive session please exit the job
Hold down Control key and press the d key (ctr-d)

6) Job Arrays
    a) Submit a serial job array that:
        i) Has a maximum wall time of 2 minutes
        ii) Sleeps 30 seconds
        iii) Is named “my-array-job”
        iv) Has 12 tasks
        v) Writes a files to $PBS_JOBNAME

    b) Run the following commands to see your job running
        i) “showq –u $USER”
        ii) “qstat –a –u $USER”
        iii) “qstat –t –u $USER”
        iv) “jobinfo –j”

7) Job Arrays
    a) Submit a serial job array
        i) Has a maximum wall time of 2 minutes
        ii) Sleeps 30 seconds
        iii) Is named “my-array2-job”
        iv) Has 12 tasks
        v) Writes a files to $PBS_JOBNAME
        vi) Runs at most 2 jobs at once
    b) Run the following commands to see your job running
        i) “showq –u $USER”
        ii) “qstat –a –u $USER”
        iii) “qstat –t –u $USER”
        iv) “jobinfo –j”

8) Job Arrays
    a) Submit a serial job array that
        i) Sleeps 30 seconds
        ii) Is named “my-array3-job”
        iii) Has 4 tasks with indexes of: 1, 2, 7, -13
        iv) Writes a files $PBS_JOBNAME
    b) Run the following commands to see your job running
        i) “showq –u $USER”
        ii) “qstat –a –u $USER”
        iii) “qstat –t –u $USER”
        iv) “jobinfo –j”

9) Interactive Job arrays
(Having an interactive job array may not make sense in practice, but is educational)
   a) Open a second ssh session/terminal to the workshop cluster
   b) In the second ssh session submit a job that:
      i) Has 2 tasks with indexes of: 1, 4
      ii) Asks for 1 core per task
      iii) Has a maximum wall time of 20 minutes
      iv) Emails you when the job is aborted, before it runs and a after it ends
      v) Is named "my-array4-job"
      vi) Is interactive
   c) Wait for the job to start, then after the job starts:
      i) Look at the PBS environment variables
         Hint: "printenv | grep -i pbs"
      ii) Give a list of node names where each process of this job runs?
         Hint: "cat $PBS_NODEFILE"
      iii) On how many nodes (there is only one in this case) does this job run?
         Hint: "echo $PBS_NUM_NODES"
      iv) On how many processors (there is only one in this case) does this job run?
         Hint: "echo $PBS_NP"
      v) On how many processors per node (there is only one in this case) does this job run?
         Hint: "echo $PBS_NUM_PPN"
      vi) What is the array ID of this job?
         Hint: "echo $PBS_ARRAYID"
   d) From your first terminal on the login node look at your job in the output of the following commands how does it compare:
      i) "showq –u $USER"
      ii) "qstat –a –u $USER"
      iii) "qstat –t –u $USER"
      iv) "jobinfo –j"
   e) In your interactive session please exit the job
      Hold down Control key and press the d key (ctr-d)

10) Working example of an array job taking input from a file (This is advanced example, It was included as a request from a prior workshop, we may not have time to write this in the session, in that case just look at and run the answer pbs script.)
    a) Submit a serial job array that reads from a single file and runs a job for each line in the input file.
        i) Is named "my-array5-job"
        ii) Has 4 tasks with 1 procs each.
        iii) Emails you when your job is complete.
        iv) The file that is used as input is named: "input.q10.pbs"
    b) Have each job array output multiply the fist number and adding the second.
    c) Run the job and see the output
    d) As advanced work if time permits see if you can output in a single file as opposed to many array files.

11) MPI Jobs

Submit the start-mpi.pbs job

    a) Look at the job with the following commands:
        i) "showq –u $USER"
        ii) "qstat –a –u $USER"
        iii) "jobinfo –j"
        iv) "checkjob <jobid>"
    b) Note how long it took to run
    c) Edit the start-q11.pbs script to user 4 processors
    d) Submit the edited script
    e) Look at the job with the following commands:
        i) "showq –u $USER"
        ii) "qstat –a –u $USER"
        iii) "jobinfo –j"
        iv) "checkjob <jobid>"
    f) Please list which nodes and cores the job is running on or scheduled to run on and how long it took to run.

12) MPI Interactive Job

a) Submit a job
   i) Asks for 1 node with 4 processors per node
   ii) Has a walltime of 20 minutes
   iii) Emails you when the job is aborted, before it runs and a after it ends
   iv) Is named "my-interactive-mpijob"
   v) Is interactive

b) After the job starts look at the PBS environment variables
   (run: "printenv | grep -i pbs")
   i) What is the jobs id?
      Hint: "echo $PBS_JOBID"
   ii) On how many nodes does this job run?
      Hint: "echo $PBS_NUM_NODES"
   iii) On how many processors does this job run?
      Hint: "echo $PBS_NP"
   iv) On how many processors per node does this job run?
      Hint: "echo $PBS_NUM_PPN"
   v) Give a list of node names where each process of this job runs?
      Hint: "cat $PBS_NODEFILE"
   vi) What is the array ID of this job?
      Hint: "echo $PBS_ARRAYID"
   vii) List the jobs cpuset. (Advanced Unix users)
      Hint: "ls /dev/cpuset/torque/$PBS_JOBID"
   viii) Verify that your current shell's process id is inside the cpuset (Advanced Unix users)
      Hint: "echo $$" to find the process id of your current shell
      Hint: " cat /dev/cpuset/torque/$PBS_JOBID/tasks" to list processes in the jobs cpuset.
   ix) Show which cores and which memory set that your job is running on? (Advanced Unix users)
      Hint: "cat /dev/cpuset/torque/$PBS_JOBID/cpus" for core numbers that the job processes are running on.
      Hint: "cat /dev/cpuset/torque/$PBS_JOBID/mems" for memory location that the job memory may be used on

c) In another terminal/ssh session log into the cluster head node and look at the job with the following commands:
   i) "showq –u $USER"
   ii) "qstat –a –u $USER"
   iii) "jobinfo –j"
   iv) "checkjob <jobid>"

d) Exit your job with the exit command

13) MPI Interactive Jobs part2

a) Submit a job
   i) Asks for 4 procs
   ii) Has a walltime of 20 minutes
   iii) Emails you when the job is aborted, before it runs and a after it ends
   iv) Is named “my-interactive-job2”
   v) Is interactive
b) From the job’s interactive session determine
   i) What is the jobs id?
   ii) On how many nodes does this job run?
   iii) On how many processors does this job run?
   iv) On how many processors per node does this job run?
   v) Give a list of node names where each process of this job runs?
   vi) On which cores on which machine is your job is running on? Give the specific core number. (Advanced unix users)
   Hint: you may need to ssh into all the other nodes that were allocated to this jobs and read the /dev/cpuset/torque/<your jobid>/cpus file on each allocated node
c) In another terminal/ssh session log into the cluster head node and look at the job with the following command: “checkjob <jobid>”
   i) Does the allocated nodes output agree with result of b.ii and b.vi?
d) Exit your job with the exit command

14) OpenMP jobs

a) Submit a job
   i) Asking for 1 node with 12 cores
   ii) sleeps 60 seconds
   iii) Has a maximum walltime of 20 minutes
   iv) Is named "my-interactive-openmpjob"
   v) Is interactive

b) After the job starts look at the PBS environment variables
   (run: "printenv | grep -i pbs")
   i) What is the jobs id?
   Hint: "echo $PBS_JOBID"
   ii) On how many nodes does this job run?
   Hint: "echo $PBS_NUM_NODES"
   iii) On how many processors does this job run?
   Hint: "echo $PBS_NP"
   iv) On how many processors per node does this job run?
   Hint: "echo $PBS_NUM_PPN"
   v) Give a list of node names where each process of this job runs?
   Hint: "cat $PBS_NODEFILE"
   vi) What is the array ID of this job?
   Hint: "echo $PBS_ARRAYID"
   vii) Show which cores and which memory set that your job is running on?
   (Advanced Unix users)
   Hint: "cat /dev/cpuset/torque/$PBS_JOBID/cpus" for core numbers that the job processes are running on.
   Hint: "cat /dev/cpuset/torque/$PBS_JOBID/mems" for memory location that the job memory may be used on

c) Look at the job with the following commands:
   - "showq –u $USER"
   - "qstat –a –u $USER"
   - "jobinfo –j"
   - "checkjob <jobid>"

d) Exit your job with the exit command

15) Hybrid Jobs

Submit a job that:

a) Asking for 4 nodes with 4 cores
b) sleeps 60 seconds
c) Has a maximum walltime of 2 minutes
d) Look at the job with the following commands:
    - "showq –u $USER"
    - "qstat –a –u $USER"
    - "jobinfo –j"
    - "checkjob <jobid>"

    ii) Please list which nodes and cores the job is running on or scheduled to run on.

16) Hybrid Interactive Jobs

a) Submit a job
    i) Asks for 2 nodes with 5 processors per node
    ii) Has a walltime of 20 minutes
    iii) Emails you when the job is aborted, before it runs and a after it ends
    iv) Is named "my-interactive-hybrid-job"
    v) Is interactive
b) From the job's interactive session determine
    i) What is the jobs id?
    ii) On how many nodes does this job run?
    iii) On how many processors does this job run?
    iv) On how many processors per node does this job run?
    v) Give a list of node names where each process of this job runs?
    vi) Show which cores on which machine your job is running on? Give the specific core number. (Advanced unix users)
    Hint: you may need to ssh into all the other nodes that were allocated to this jobs and read the /dev/cpuset/torque/<your jobid>/cpus file on each allocated node
c) In another terminal/ssh session log into the cluster head node and look at the job with the following commands:
    - "showq –u $USER"
    - "qstat –a –u $USER"
    - "jobinfo –j"
    - "checkjob <jobid>"

    ii) Does the allocated nodes output agree with result of part b?
d) Exit your job with the exit command

17) Jobs and memory (mem,pmem)

a) Take the start-mem.pbs script and edit it so that is asks for: pmem=12000mb
b) Submit a job from the script you edited . Look at the job with the following commands:
   i) "showq –u $USER"
   ii) "qstat –a –u $USER"
   iii) "jobinfo –j"
   iv) "checkjob <jobid>"
c) How much memory does this job use?

18)Jobs and memory (mem,pmem)

a) Take the start-mem.pbs script and edit it so that is asks for: mem=12000mb
b) Submit a job from the script you edited . Look at the job with the following commands:
   i) "showq –u $USER"
   ii) "qstat –a –u $USER"
   iii) "jobinfo –j"
   iv) "checkjob <jobid>"
c) How much memory does this job use?

19)Jobs and memory (mem,pmem)

a) Take the start-mem.pbs script and edit it so that is asks for: pmem=3000mb
b) Submit a job from the script you edited . Look at the job with the following commands:
   i) "showq –u $USER"
   ii) "qstat –a –u $USER"
   iii) "jobinfo –j"
   iv) "checkjob <jobid>"
c) How much memory does this job use?

20) Jobs and memory (appropriate resources)

a) Create a job run the "cryptic" program edit the start-mem2.pbs script
   i) Make sure your job emails you when is starts, ends and aborts
   ii) Make a guess and for enough RAM to run the program
b) Submit your edited Job script , look at your running Job with the following commands, look at the ,memory used by your job
   i) "checkjob <jobid>"
   ii) "qstat –f <jobid>"
c) Did your job run successfully? Or fail because of a lack of memory?
   i) If your job failed due to a lack of memory, increase the maximum memory requested and resubmit your job, and go back to point b)
d) Look at the email reporting on your job success, how much resources were reported used. Compare the memory used to the reported memory in point c).
e) Edit job script and request an appropriate amount of memory to run the Job .
f) Submit your new job
g) Verify that the jobs runs successfully.

21) Software licenses and generic resources

a) Submit a job asking for that asks for
   i) 1 proc
   ii) 1 MATLAB license
   iii) 1 Statistics_Toolbox license
b) Try to see resources used by your job, use the checkjob command:

22) Full nodes
   a) Submit a job asking for that asks for
      i) 4 procs
      ii) Not to run on any nodes with other users
          Useful if you are trying to debug your job
   b) See if you can see which nodes your job is running on.
      i) "checkjob <jobid>"
      ii) "qstat –f <jobid>"

23)Full nodes
   a) Submit a job asking for that asks for
      i) 4 procs
      ii) Not to run on any nodes with other jobs
          Useful if you are trying to debug your job
   b) See if you can see which nodes your job is running on.
      i) "checkjob <jobid>"
      ii) "qstat –f <jobid>"

24) Full nodes
   a) Submit a job asking for that asks for
      i) 4 procs
      ii) Each task should run on a separate node.
   b) See if you can see which nodes your job is running on.
      i) "checkjob <jobid>"
      ii) "qstat –f <jobid>"

25)Job dependencies

   a) Submit a serial job named dep1, that has
      i) Walltime of 2:00
      ii) Sleeps 120 seconds
   b) Submit a serial job 22b waits until job dep2 is done
      i) Walltime of 2:00
      ii) Sleeps 120 seconds
   c) Look at job dep2 with checkjob
   d) Run the command "showq –u $USER"
   e) Verify that Job dep1 complete before dep2 starts

26) Job epilogue
   a) Submit a job that runs the provided epilogue script "epilogue.script".

27) Job using temporary directory
   a) Submit a job that runs in the temporary directory used no more than 1000mb of space,

28) Job environment variables.

a) Submit a serial job that prints the queue that the job was ran in.

29) Basic Job info

a) Use the “Jobinfo –j” and “qstat -t -u <username>” commands to find out how many jobs your have running, queued, in hold state or complete.

b) Use the “showq” command what does active, eligible, blocked jobs mean, how many jobs are in each category

c) Use the “showq –b” command to see how many jobs are in what state?

30) Examining a job

a) Start a Job

b) Examine its priority with ”jobinfo –i” or “showq -i”

c) run qstat –f <jobid > and determine how much RAM the Job asks for/used

d) run checkjob –v –v <> determine and examine the result , what is its priority

31) Multiple accounting groups (This exercise will only be available to users with RAC allocated groups or multiple accounting groups. The answer will need to be modified with your accounting group.)

a) Submit a Job to a non default accounting group, that asks for 1 proc
b) Try to see which accounting group your job belongs to, use the checkjob command:
   i) "checkjob <jobid>"

32) Priority, Fairshare, and allocations

a) Start a Job is your priority what is its priority relative to other jobs.

b) What is your research groups allocation and usage

c) Which person in your group has used the most resources, have they used more than their share?

d) When were the most Jobs ran by your group

33) Job holds

a) See if any of your jobs in the queue have any job holds, if so identify the hold and the reason why.

34) Cluster info

a) How many idle nodes are on the cluster "mdiag –n"

b) How many cpus and memory are not being used on the cluster?

c) How many cores with 2 GB Ram are not being used on the cluster?

d) How many nodes are set offline on a cluster.