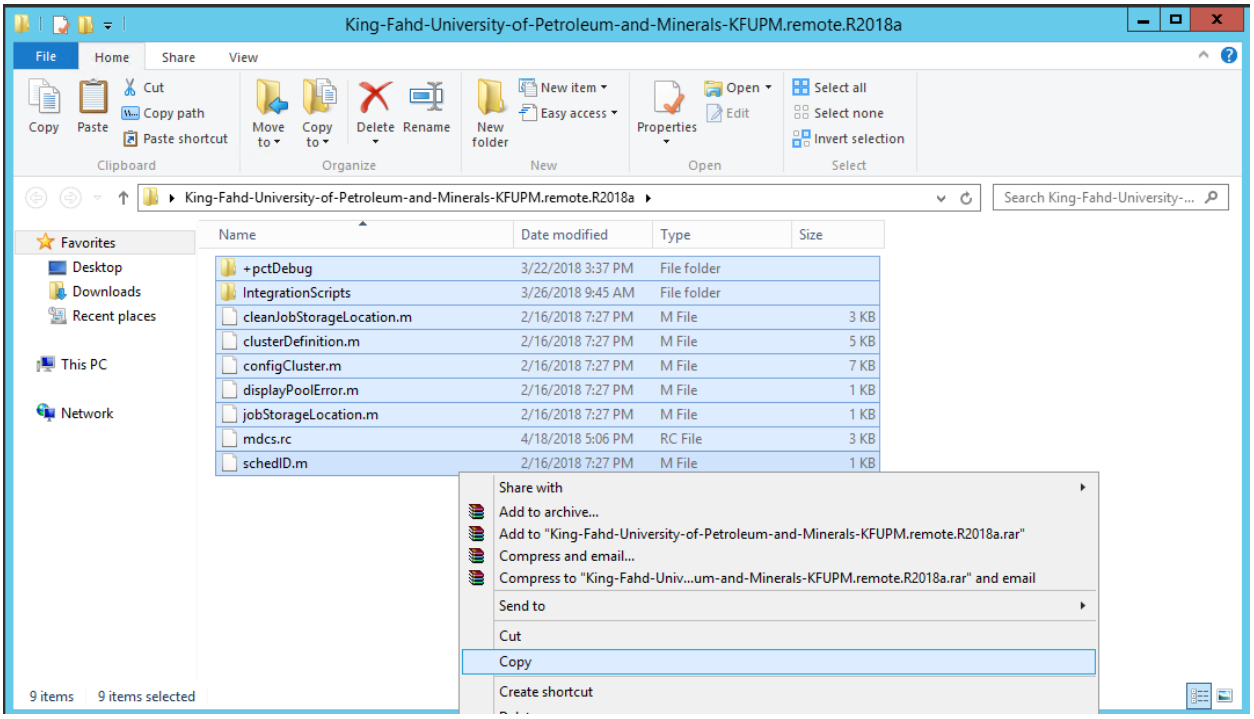
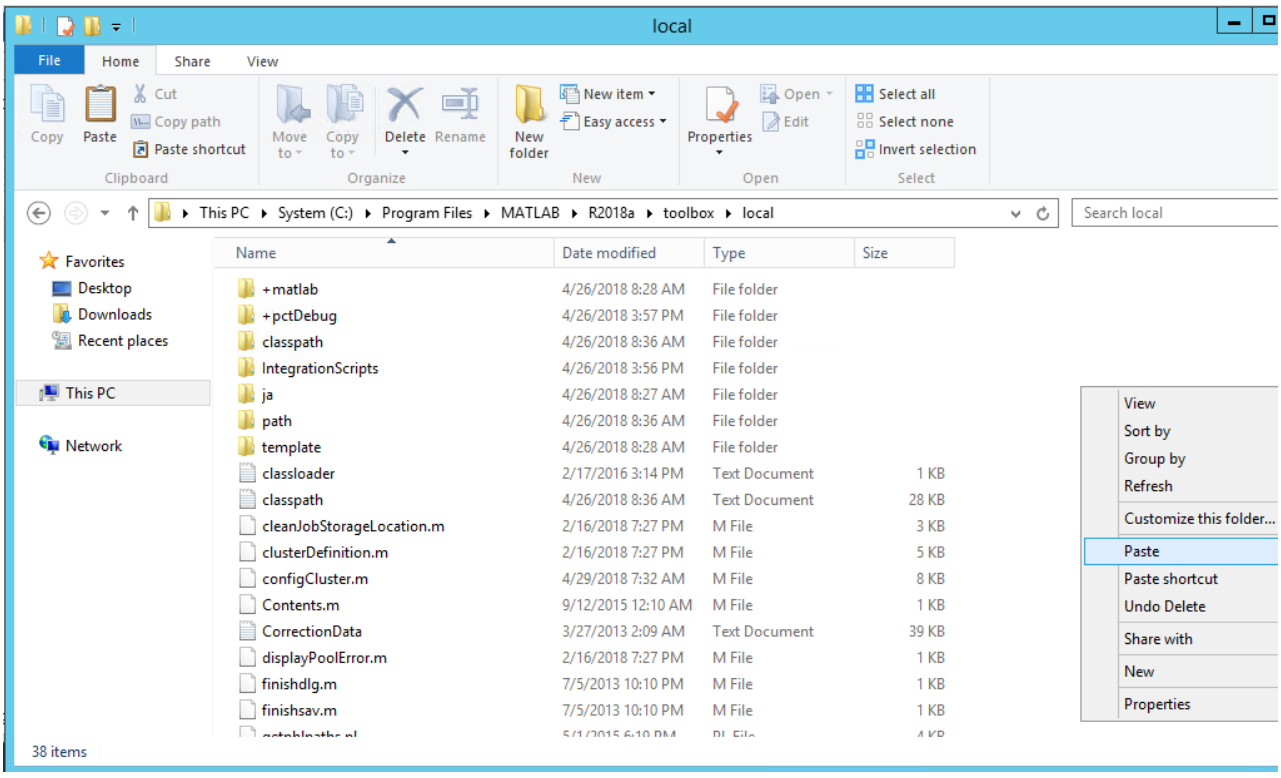
MATLAB Job Submission on Alfahd

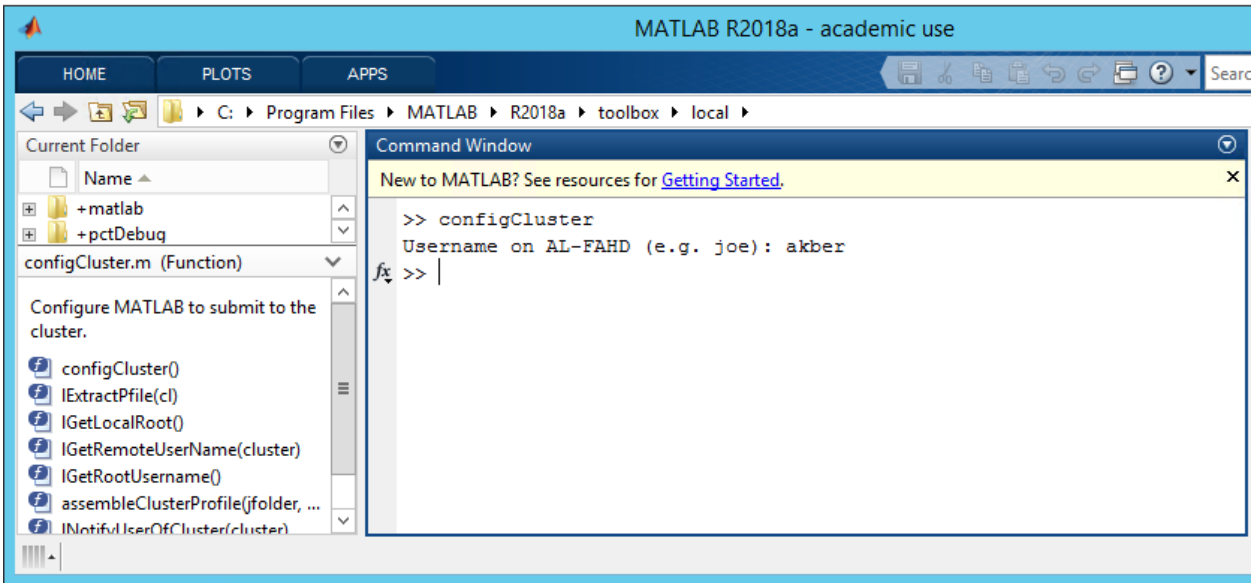
5/7/2018

Step1: Download the integration files zip folder. After unzipping it copy all the files to location **" C:\Program Files\MATLAB\R2018a\toolbox\local "** where Matlab installation files are present and paste it.

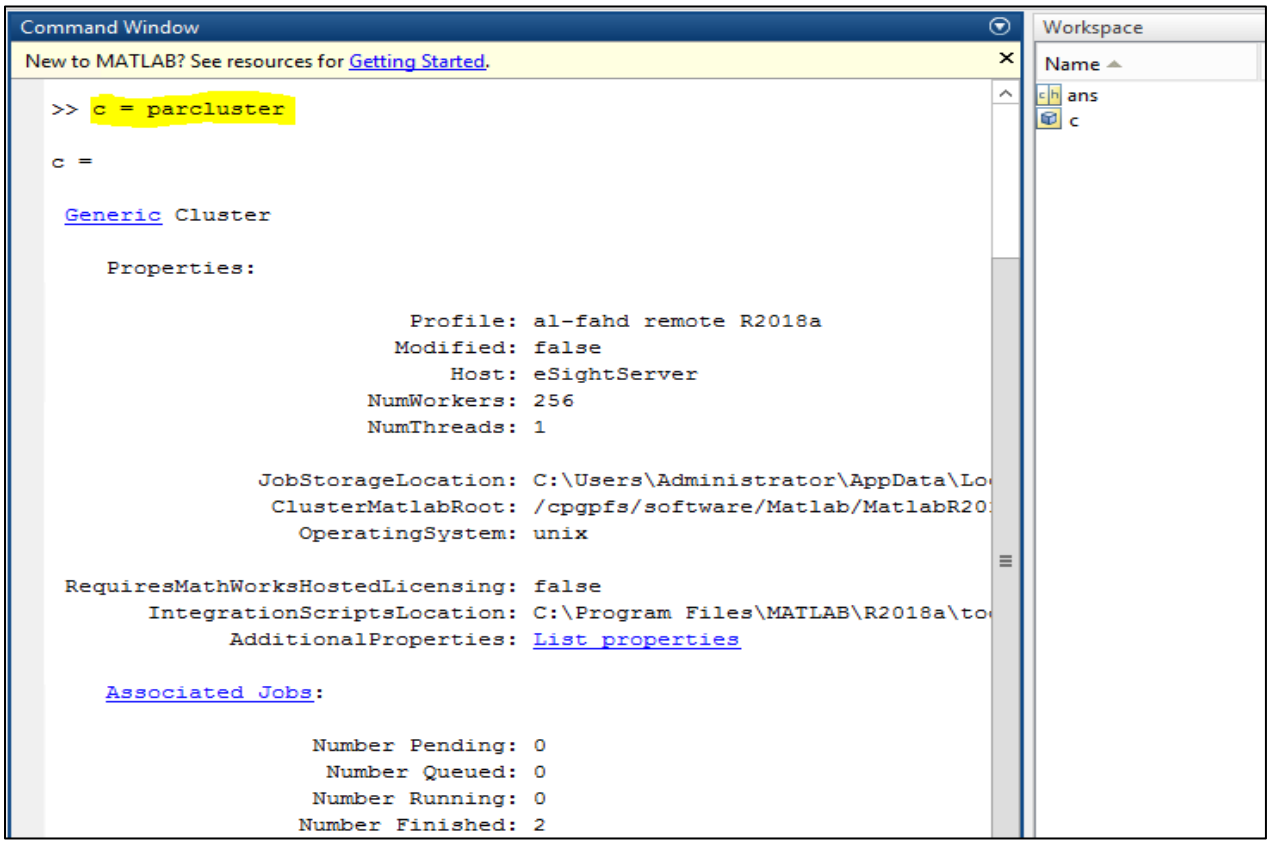




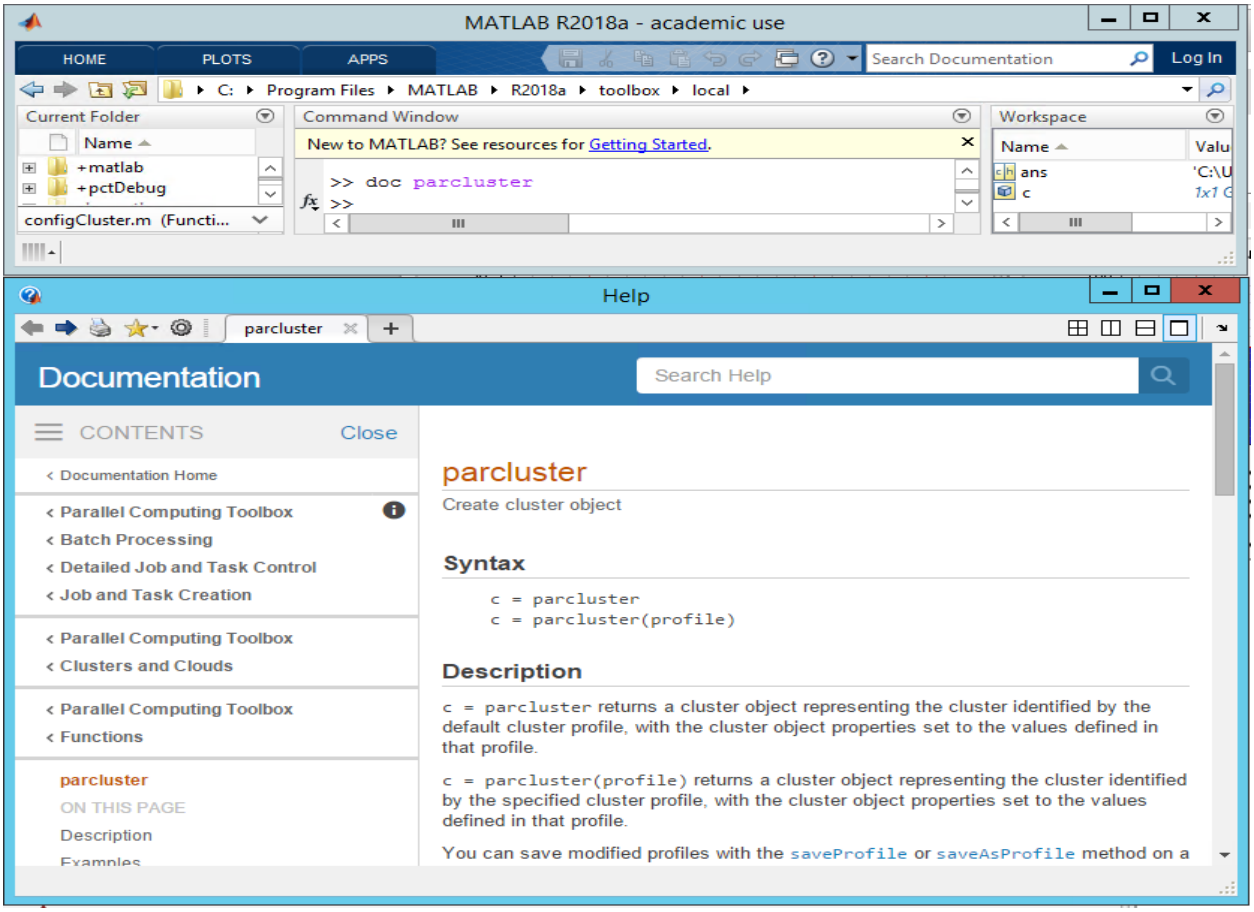
Step2: Now you have to connect your MATLAB session to the Alfahd cluster you're your user details using below command "configCluster" as shown below. The activity in first two steps is needed only once, later on you may skip directly to step-3.



Step3: Now you have to define connection parameter "c" to connect your MATLAB session to the Alfahd cluster using below command "c = parcluster" as shown below.



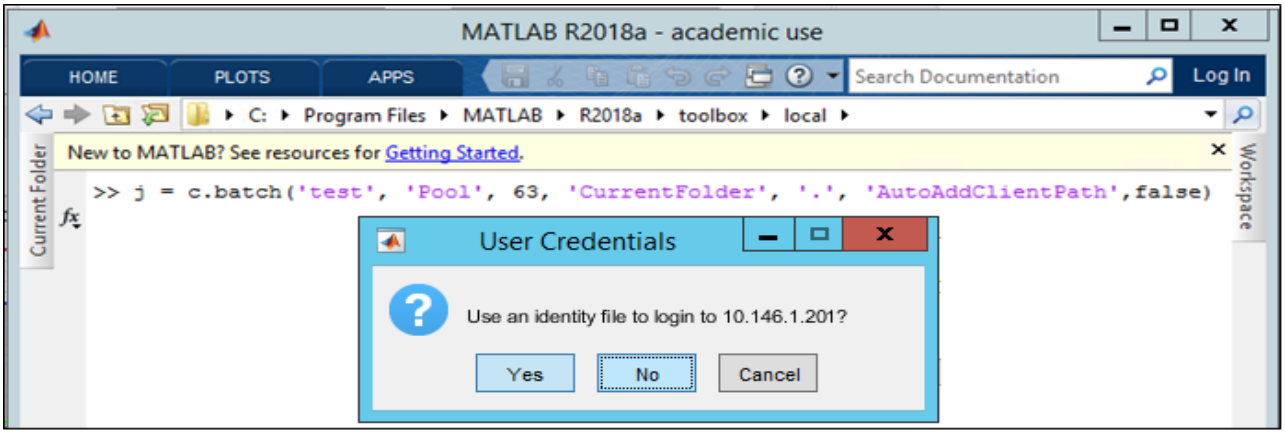
For more details about parcluster type “doc **parcluster**” at command, prompt.



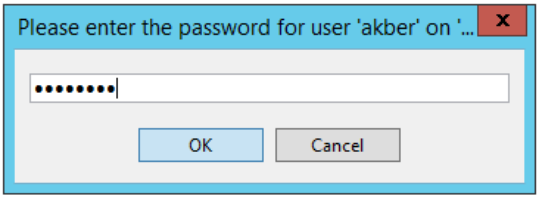
Step4: You can now submit your job using connection parameter handle to batch command as show below,

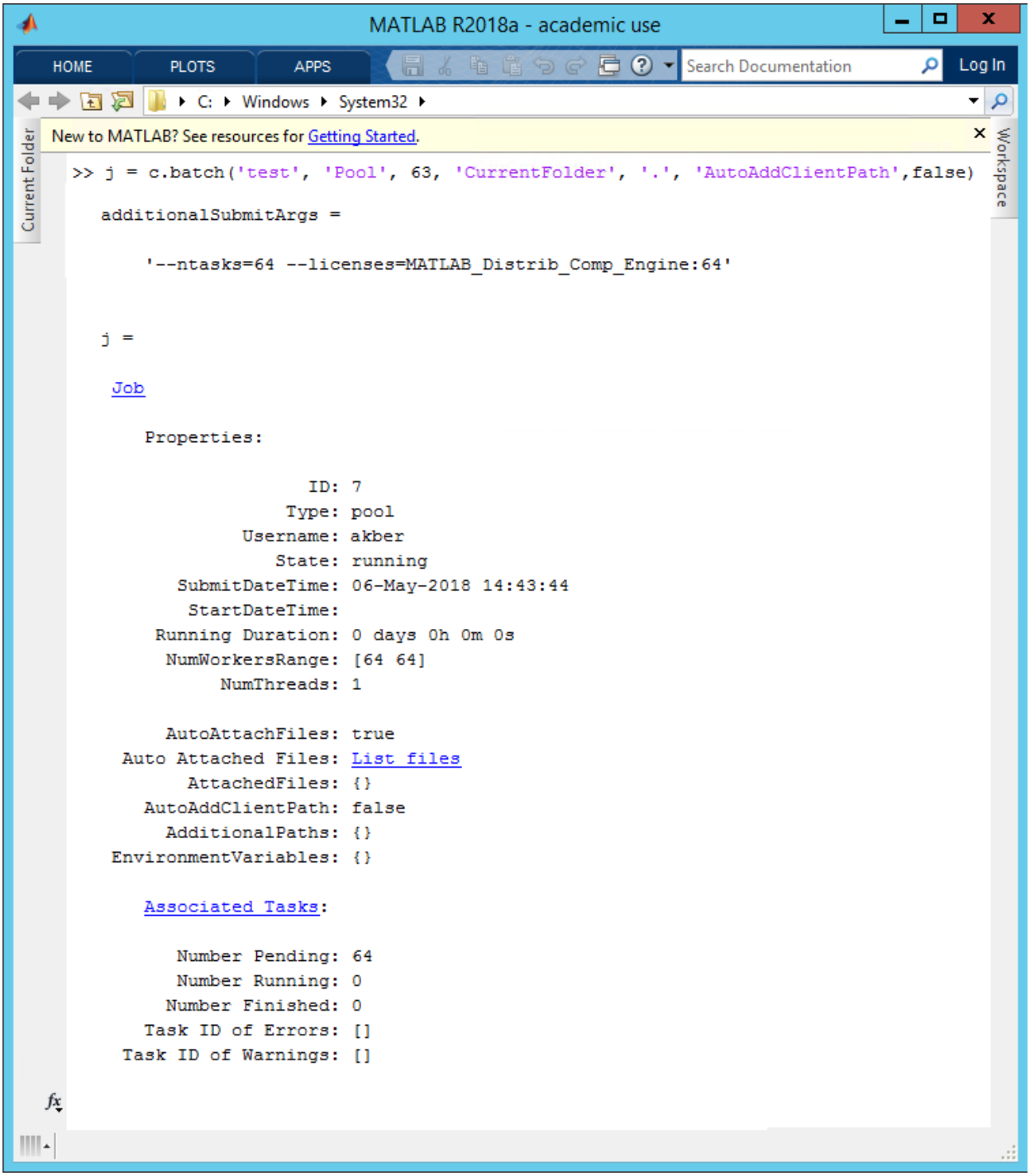
eg: `j = c.batch('MainProgramName', 'Pool', 63, 'CurrentFolder', '.', 'AutoAddClientPath',false)`

In above command **'Pool', 63** will start **64 parallel MATLAB workers i.e.** (63+1 Manager Worker) that will parallelize any of your parfor loops or similar part in the code.



When prompted for an identity file in User Credentials prompt select ‘No’ and enter your password as below:





Then your job is submitted on the Alfahd cluster whose details are furnished in the command prompt with its status. You can then monitor the status of your jobs, see command prompt outputs on cluster, download results once job is done and have a glance at list of jobs on the cluster using commands: j.State , j.diary, j.fetchoutputs{:} & c.Jobs respectively, as shown below.

```
>> j.State

ans =

    'running'
```

```
>> j.diary
--- Start Diary ---
Start Simulation

t =

   10.2122

Simulation Completed

--- End Diary ---
```

```
>> j.fetchOutputs{:}

    ans =

       10.2122
```

```
>> c.Jobs

ans =

 7x1 Job array:

        ID          Type        State              FinishDateTime  Username  Tasks
    ------------------------------------------------------------------------------
   1     1          pool      finished        29-Apr-2018 08:18:02   akber         5
   2     2          pool      finished        29-Apr-2018 08:18:32   akber         5
   3     4          pool      finished                               akber        64
   4     5          pool      finished                               akber        64
   5     6          pool      finished                               akber        64
   6     7          pool      finished                               akber        64
   7     8          pool      finished                               akber        64
```

j.fetchoutputs{:} is used to retrieve function output arguments; if using batch with a script, use j.load instead. Data that has been written to files on the cluster will be retrieved/downloaded directly from the file system once j.load is executed.

```
>> j.State

ans =

    'running'

>> j.State

ans =

    'finished'

>> j.load
```

When submitting jobs, with multiple tasks, you will have to specify the task number to view results of a previously completed job using command listed below:

% Find the old job

```
j = c.findJob('ID', 9);
```

% Retrieve the state of the job

```
j.State
```

% Fetch the results

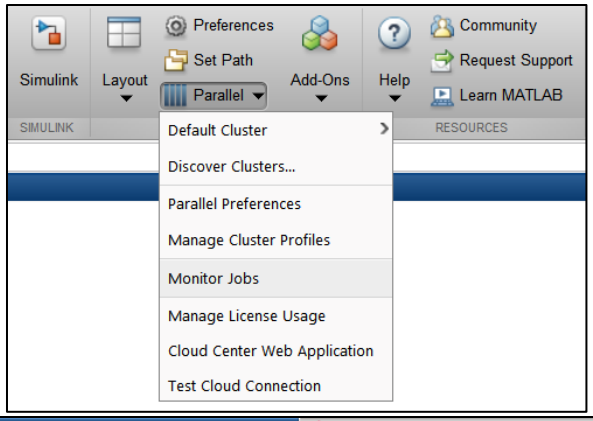
```
j.fetchOutputs{:};
```

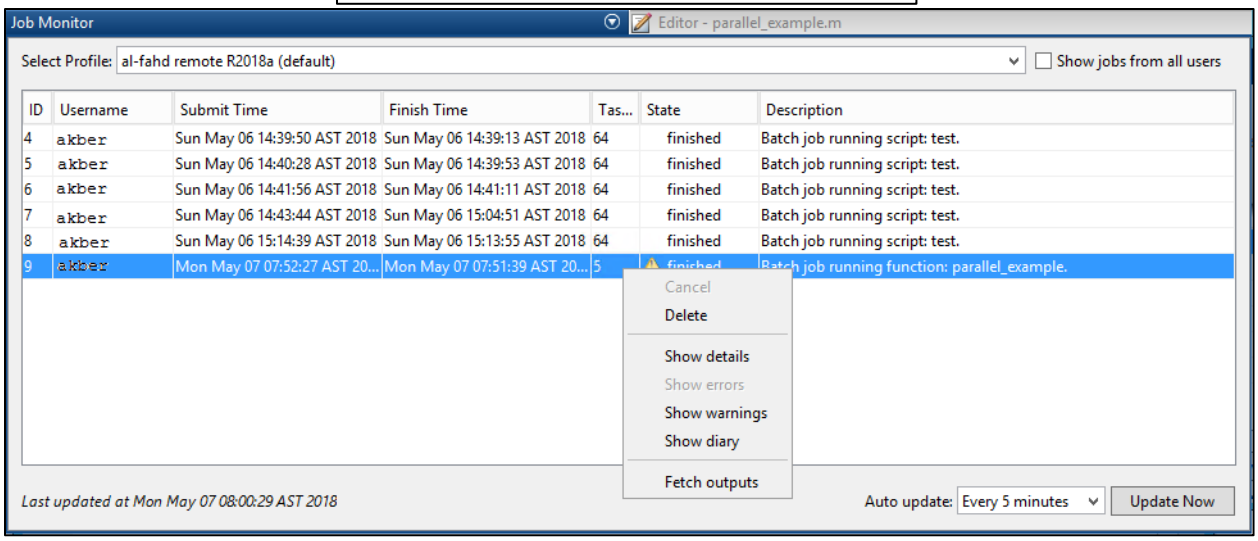
% If necessary, retrieve output/error log file

```
c.getDebugLog(j)
```

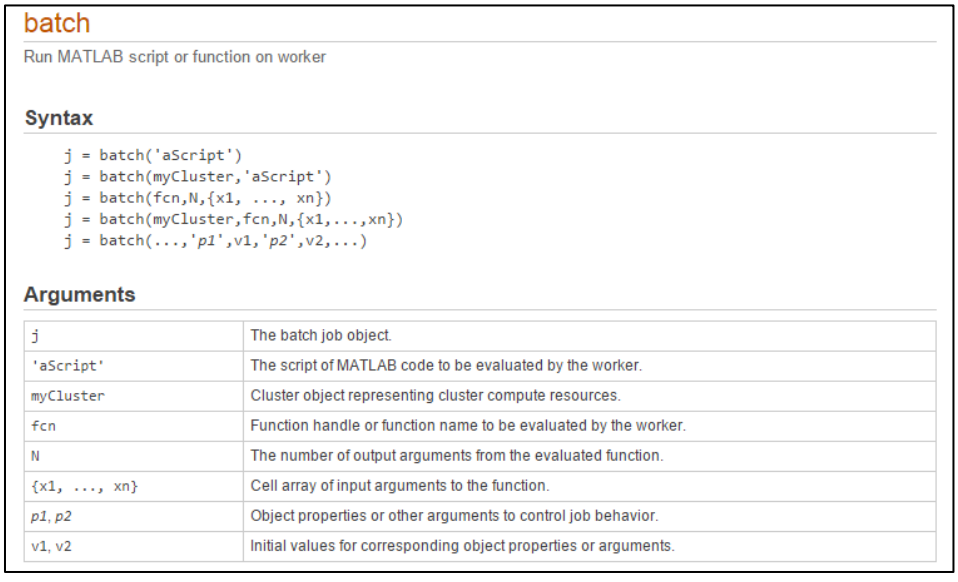
```
>> j = c.findJob('ID', 9);
>> j.State

ans =

    'finished'

>> c.getDebugLog(j)
LOG FILE OUTPUT:
Node list: cpg044
srun -l -n 5 /cpgpfs/software/Matlab/MatlabR2018a/bin/worker -parallel
0:
0:                            < M A T L A B (R) >
0:                  Copyright 1984-2018 The MathWorks, Inc.
0:                   R2018a (9.4.0.813654) 64-bit (glnxa64)
0:                             February 23, 2018
0:
1:
```

Alternatively, to retrieve job results via a GUI, use the Job Monitor (**Parallel** > **Monitor Jobs**).





For more details about batch, type "doc **batch**"



batch

Run MATLAB script or function on worker

Syntax

```
j = batch('aScript')
j = batch(myCluster,'aScript')
j = batch(fcn,N,{x1, ..., xn})
j = batch(myCluster,fcn,N,{x1,...,xn})
j = batch(...,'p1',v1,'p2',v2,...)
```

Arguments

j	The batch job object.
'aScript'	The script of MATLAB code to be evaluated by the worker.
myCluster	Cluster object representing cluster compute resources.
fcn	Function handle or function name to be evaluated by the worker.
N	The number of output arguments from the evaluated function.
{x1, ..., xn}	Cell array of input arguments to the function.
p1, p2	Object properties or other arguments to control job behavior.
v1, v2	Initial values for corresponding object properties or arguments.