MATLAB® Parallel Server™

System Administrator's Guide

MATLAB®

R2023a

MathWorks®

How to Contact MathWorks

Latest news:	`www.mathworks.com`
Sales and services:	`www.mathworks.com/sales_and_services`
User community:	`www.mathworks.com/matlabcentral`
Technical support:	`www.mathworks.com/support/contact_us`
Phone:	508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

MATLAB® Parallel Server™ System Administrator's Guide

Revision History

November 2005	Online only	New for Version 2.0 (Release 14SP3+)
December 2005	Online only	Revised for Version 2.0 (Release 14SP3+)
March 2006	Online only	Revised for Version 2.0.1 (Release 2006a)
September 2006	Online only	Revised for Version 3.0 (Release 2006b)
March 2007	Online only	Revised for Version 3.1 (Release 2007a)
September 2007	Online only	Revised for Version 3.2 (Release 2007b)
March 2008	Online only	Revised for Version 3.3 (Release 2008a)
October 2008	Online only	Revised for Version 4.0 (Release 2008b)
March 2009	Online only	Revised for Version 4.1 (Release 2009a)
September 2009	Online only	Revised for Version 4.2 (Release 2009b)
March 2010	Online only	Revised for Version 4.3 (Release 2010a)
September 2010	Online only	Revised for Version 5.0 (Release 2010b)
April 2011	Online only	Revised for Version 5.1 (Release 2011a)
September 2011	Online only	Revised for Version 5.2 (Release 2011b)
March 2012	Online only	Revised for Version 6.0 (Release 2012a)
September 2012	Online only	Revised for Version 6.1 (Release 2012b)
March 2013	Online only	Revised for Version 6.2 (Release 2013a)
September 2013	Online only	Revised for Version 6.3 (Release 2013b)
March 2014	Online only	Revised for Version 6.4 (Release 2014a)
October 2014	Online only	Revised for Version 6.5 (Release 2014b)
March 2015	Online only	Revised for Version 6.6 (Release 2015a)
September 2015	Online only	Revised for Version 6.7 (Release 2015b)
March 2016	Online only	Revised for Version 6.8 (Release 2016a)
September 2016	Online only	Revised for Version 6.9 (Release 2016b)
March 2017	Online only	Revised for Version 6.10 (Release 2017a)
September 2017	Online only	Revised for Version 6.11 (Release 2017b)
March 2018	Online only	Revised for Version 6.12 (Release 2018a)
September 2018	Online only	Revised for Version 6.13 (Release 2018b)
March 2019	Online only	Revised for Version 7.0 (Release 2019a)
September 2019	Online only	Revised for Version 7.1 (Release 2019b)
March 2020	Online only	Revised for Version 7.2 (Release 2020a)
September 2020	Online only	Revised for Version 7.3 (Release 2020b)
March 2021	Online only	Revised for Version 7.4 (Release 2021a)
September 2021	Online only	Revised for Version 7.5 (Release 2021b)
March 2022	Online only	Revised for Version 7.6 (Release 2022a)
September 2022	Online only	Revised for Version 7.7 (Release 2022b)
March 2023	Online only	Revised for Version 7.8 (Release 2023a)

Contents

1 Introduction

MATLAB Parallel Server Product Description **1-2**

Product Overview **1-3**
Parallel Computing Concepts **1-3**
Determining Product Installation and Versions **1-3**

2 Network Administration

Requirements and Ports for MATLAB Parallel Server **2-2**
Port Configuration **2-2**
Fully Qualified Domain Names **2-2**
Security Considerations **2-2**

Use Different MPI Builds on UNIX Systems **2-4**
Build MPI **2-4**
Use Your MPI Build **2-4**

Shut Down a Job Manager Cluster **2-6**
Linux Operating Systems **2-6**
Microsoft Windows Operating Systems **2-7**

Define MATLAB Job Scheduler Startup Parameters **2-9**
MJS Process and Logging **2-10**
Job Manager Security **2-14**
Job Manager and Worker Settings **2-18**

Customize Startup Parameters **2-21**
Modify Script Defaults **2-21**
Override Script Defaults **2-22**
Increase Heap Memory **2-22**

Set up MATLAB Job Scheduler Cluster for Auto-Resizing **2-24**
Set Maximum Number of Workers **2-24**
Start MJS Cluster **2-24**
Set up Auto-Resizing Process **2-25**

Access Service Record Files **2-27**
Locate Log Files **2-27**
Locate Checkpoint Folders **2-27**

Set MATLAB Job Scheduler Cluster Security **2-29**
Set Security Level **2-29**
Local, MATLAB Job Scheduler, and Network Passwords **2-32**
Authorize Users for Job and Task Access **2-32**
Create Shared Secret File **2-32**
Set Encrypted Communication **2-33**
Set MATLAB Client Verification **2-33**
Set Cluster Command Verification **2-34**

Troubleshoot Common Problems **2-37**
License Errors **2-37**
Memory Errors on UNIX Operating Systems **2-38**
Run Server Processes on Windows Network Installation **2-38**
Required Ports **2-38**
Ephemeral TCP Ports with Job Manager **2-39**
Host Communications Problems **2-40**
Verify Network Communications for Cluster Discovery **2-41**

Configure LDAP Server Authentication for MATLAB Job Scheduler **2-43**
Prerequisites **2-43**
Edit MATLAB Job Scheduler Parameter File **2-43**
Configure LDAP over SSL (LDAPS) **2-45**
Start MATLAB Job Scheduler and Test LDAP Server Authentication **2-46**

3 Product Installation

Choose Solution to Install and Configure MATLAB Parallel Server **3-2**
Activate the License for, Install, and Configure MATLAB Parallel Server on the Cluster **3-2**
Configure a MATLAB Client for a User to Submit Jobs to the Cluster **3-3**
Upgrade MATLAB Parallel Server **3-4**

Install and Configure MATLAB Parallel Server for Slurm **3-6**
Activate Your MATLAB Parallel Server License **3-6**
Get the Installation Files **3-6**
Install License Manager **3-7**
Install Software on Compute Nodes **3-8**
Install Software on Local Desktop **3-8**
Configure Client Machine **3-9**
Validate the Cluster Profile **3-10**
Run Parallel Code **3-11**

Install and Configure MATLAB Parallel Server for Third-Party Schedulers **3-12**
Activate Your MATLAB Parallel Server License **3-12**
Get the Installation Files **3-12**
Install License Manager **3-13**
Install Software on Compute Nodes **3-14**
Install Software on Local Desktop **3-14**
Configure Your Cluster **3-14**

Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager **3-16**
Activate Your MATLAB Parallel Server License **3-16**
Get the Installation Files **3-16**
Install Software on the Head Node **3-17**
Install Software on Compute Nodes **3-18**
Configure the MATLAB Job Scheduler **3-18**
Connect MATLAB Client to MATLAB Parallel Server Cluster **3-20**

Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Online Licensing **3-23**
Check License Type and Users **3-23**
Get the Installation Files **3-24**
Install Software on All Nodes **3-25**
Configure the MATLAB Job Scheduler with Admin Center **3-25**
Connect the MATLAB Client to the MATLAB Parallel Server Cluster **3-27**

Configure Advanced Options for MATLAB Job Scheduler Integration . . **3-29**
Run Multiple MATLAB Parallel Server Versions **3-29**
Set Up Windows Cluster Hosts **3-30**
Open Required Ports on Server **3-30**
Stop mjs Services of Old Installation **3-31**
Set the MATLAB Job Scheduler Security Level **3-32**
Start mjs Service, MATLAB Job Scheduler, and Workers (Command-Line) **3-32**
Install the mjs Service to Start Automatically at Boot Time (UNIX) **3-35**
Validate Installation with MATLAB Job Scheduler **3-36**

Configure for HPC Pack **3-37**
Configure Cluster for Microsoft HPC Pack **3-37**
Configure Client Computer for HPC Pack **3-37**
Validate Installation Using Microsoft HPC Pack **3-38**

Configure for Slurm, PBS Pro, OpenPBS, LSF, TORQUE **3-41**
Create Cluster Profile and Validate Installation **3-41**
Configure LSF Scheduler on Windows Cluster **3-43**
Configure Windows Firewalls on Server **3-44**

Configure Using the Generic Scheduler Interface **3-45**
Interface with Third-Party Schedulers **3-45**
Create a Generic Cluster Profile **3-46**
Special Configurations **3-51**

Distribute a Generic Cluster Profile and Plugin Scripts **3-53**
Decide How Users Access the Plugin Scripts **3-53**
Shared PluginScriptsLocation Folder **3-53**
Distribute Copies of the PluginScriptsLocation Folder **3-54**
Further Considerations for Clusters with Shared File Systems **3-54**

Customize Behavior of Sample Plugin Scripts **3-55**
Set Additional Properties **3-61**

Configure a Hadoop Cluster **3-64**
Cluster Configuration **3-64**
Client Configuration **3-64**

Kerberos Authentication 3-64
Hadoop Version Support 3-65

Configure a Spark Cluster **3-66**
Cluster Configuration 3-66
Client Configuration 3-66
Kerberos Authentication 3-66
Spark Version Support 3-67

Configure for Third-Party Scheduler Cluster Discovery **3-68**
Prerequisites 3-68
Cluster Configuration File Format 3-68
Store Configuration File 3-69
Discover Clusters on Users Machine 3-70

4 **Admin Center**

Start Admin Center **4-2**

Set Up Resources **4-3**
Add Hosts 4-3
Start mjs Service 4-4
Start a MATLAB Job Scheduler 4-5
Start Workers 4-6
Stop, Destroy, Resume, Restart Processes 4-7
Move a Worker 4-7
Update the Display 4-8

Test Connectivity **4-9**

Export and Import Sessions **4-12**

Prepare for Cluster Profiles **4-13**

5 **Control Scripts**

6 **Reference Architectures**

Run MATLAB Parallel Server on Amazon Web Services **6-2**
Requirements 6-2
Run from GitHub 6-2

Run MATLAB Parallel Server on AWS Batch **6-4**
Requirements **6-4**
Run from GitHub **6-4**

Run MATLAB Parallel Server on Microsoft Azure Using Reference Architecture **6-5**
Requirements **6-5**
Run from GitHub **6-5**

Run MATLAB Parallel Server from Microsoft Azure Marketplace **6-7**
Requirements **6-7**
Deploy MATLAB Parallel Server Resources in Azure **6-8**
Delete MATLAB Parallel Server Resources from Azure **6-13**
Architecture and Resources for MATLAB Parallel Server in Azure **6-14**
Marketplace vs. Reference Architecture **6-15**
Support **6-15**

Configure MATLAB Parallel Server Licensing for Cloud Platforms **6-17**
Before You Begin **6-17**
Select Licensing Configuration Option **6-17**
Change License Manager Type **6-18**
Add End Users to License **6-18**
Disable Network License Manager for MATLAB Parallel Server **6-19**

Introduction

- “MATLAB Parallel Server Product Description” on page 1-2
- “Product Overview” on page 1-3

MATLAB Parallel Server Product Description

Perform MATLAB and Simulink computations on clusters and clouds

MATLAB Parallel Server lets you scale MATLAB programs and Simulink® simulations to clusters and clouds. You can prototype your programs and simulations on the desktop and then run them on clusters and clouds without recoding. MATLAB Parallel Server supports batch jobs, interactive parallel computations, and distributed computations with large matrices.

All cluster-side licensing is handled by MATLAB Parallel Server. Your desktop license profile is dynamically enabled on the cluster, so you do not need to supply MATLAB licenses for the cluster. The licensing model includes features to support unlimited scaling.

MATLAB Parallel Server runs your programs and simulations as scheduled applications on your cluster. You can use the MATLAB optimized scheduler provided with MATLAB Parallel Server or your own scheduler. A plugin framework enables direct communication with popular cluster scheduler submission clients.

Product Overview

In this section...
"Parallel Computing Concepts" on page 1-3
"Determining Product Installation and Versions" on page 1-3

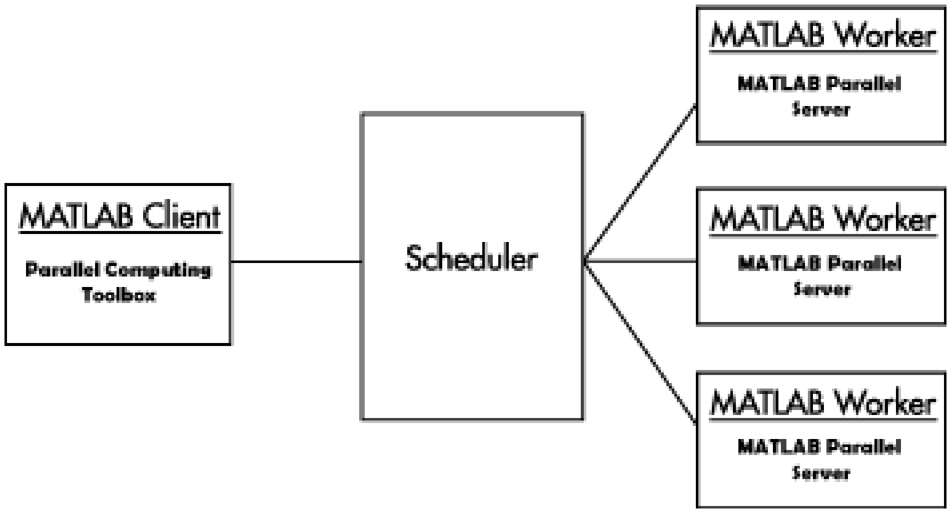
Parallel Computing Concepts

A *job* is some large operation that you need to perform in your MATLAB session. A job is broken down into segments called *tasks*. You decide how best to divide your job into tasks. You could divide your job into identical tasks, but tasks do not have to be identical.

The MATLAB session in which the job and its tasks are defined is called the *client* session. Often, this is on the machine where you program MATLAB. The client uses Parallel Computing Toolbox software to perform the definition of jobs and tasks. The MATLAB Parallel Server product performs the execution of your job by evaluating each of its tasks and returning the result to your client session.

Parallel Computing Toolbox software allows you to run a cluster of MATLAB workers on your local machine in addition to your MATLAB client session. MATLAB Parallel Server software allows you to run as many MATLAB workers on a remote cluster of computers as your licensing allows.

The MATLAB Job Scheduler is the part of the server software that coordinates the execution of jobs and the evaluation of their tasks. The MATLAB Job Scheduler distributes the tasks for evaluation to the server's individual MATLAB sessions called *workers*. Use of the MATLAB Job Scheduler is optional; the distribution of tasks to workers can also be performed by a third-party scheduler, such as Windows® HPC Server (including CCS), an IBM Spectrum® LSF® scheduler, or a PBS Pro® scheduler.



Basic Parallel Computing Configuration

Determining Product Installation and Versions

To determine if Parallel Computing Toolbox software is installed on your system, type this command at the MATLAB prompt:

```
ver
```

When you enter this command, MATLAB displays information about the version of MATLAB you are running, including a list of all toolboxes installed on your system and their version numbers.

You can run the `ver` command as part of a task in a distributed or parallel application to determine what version of MATLAB Parallel Server software is installed on a worker machine. Note that the toolbox and server software must be the same version.

Network Administration

This chapter provides information useful for network administration of Parallel Computing Toolbox software and MATLAB Parallel Server software.

- “Requirements and Ports for MATLAB Parallel Server” on page 2-2
- “Use Different MPI Builds on UNIX Systems” on page 2-4
- “Shut Down a Job Manager Cluster” on page 2-6
- “Define MATLAB Job Scheduler Startup Parameters” on page 2-9
- “Customize Startup Parameters” on page 2-21
- “Set up MATLAB Job Scheduler Cluster for Auto-Resizing” on page 2-24
- “Access Service Record Files” on page 2-27
- “Set MATLAB Job Scheduler Cluster Security” on page 2-29
- “Troubleshoot Common Problems” on page 2-37
- “Configure LDAP Server Authentication for MATLAB Job Scheduler” on page 2-43

Requirements and Ports for MATLAB Parallel Server

In this section...
"Port Configuration" on page 2-2
"Fully Qualified Domain Names" on page 2-2
"Security Considerations" on page 2-2

This section discusses the requirements and port configuration for your network to support parallel computing.

Port Configuration

Before you can use MATLAB Parallel Server, you must configure certain required ports. For details, see "Required Ports" on page 2-38. If you need more help during configuration, see this information from MathWorks® Support Team on MATLAB Answers: MATLAB Job Scheduler, or Third-Party Scheduler.

Fully Qualified Domain Names

MATLAB Parallel Server software and Parallel Computing Toolbox software support both short host names and fully qualified domain names. The default usage is short host names. Check the following considerations depending on your scheduler type:

Scheduler	Consideration
MATLAB Job Scheduler	• If your network requires fully qualified host names, you can use the `mjs_def` file to identify the worker nodes by their full names. See "Customize Startup Parameters" on page 2-21. • To set the host name used for a MATLAB client session, see the `pctconfig` reference page.
Third-Party scheduler	• To set the host name used for a MATLAB client session, see the `pctconfig` reference page.

Security Considerations

Check the following table for security considerations when using MATLAB Parallel Server:

Scheduler	Security Consideration
MATLAB Job Scheduler	• MATLAB workers run as whatever user the administrator starts the node's mjs service under. By default, the mjs service starts as `root` on UNIX® operating systems, and as `LocalSystem` on Microsoft® Windows operating systems. Because MATLAB provides system calls, users can submit jobs that execute shell commands. If you want to run tasks as the user that submitted the job, use security level 3. For more information, see "Set Security Level" on page 2-29. • By default, the mjs service does not enforce any access control or authentication. Anyone with local or remote access to the mjs services can start and stop their workers and job managers, and query for their status. For authentication and access control options, see "Set MATLAB Job Scheduler Cluster Security" on page 2-29. • The job manager does not restrict access to the cluster, nor to job and task data. For information on security options, see "Set MATLAB Job Scheduler Cluster Security" on page 2-29. Using a third-party scheduler instead of the MathWorks job manager could allow you to take advantage of the security measures it provides. • The parallel computing processes must all be on the same side of a firewall, or you must take measures to enable them to communicate with each other through the firewall. Workers running tasks of the same communicating job cannot be firewalled off from each other, because their MPI-based communication will not work. • If certain ports are restricted, you can specify the ports used for parallel computing. See "Modify Script Defaults" on page 2-21. • If your organization is a member of the Internet Multicast Backbone (MBone), make sure that your parallel computing cluster is isolated from MBone access if you are using multicast for parallel computing. Isolation is generally the default condition. If you have any questions about MBone membership, contact your network administrator.
Third-Party scheduler	• The parallel computing processes must all be on the same side of a firewall, or you must take measures to enable them to communicate with each other through the firewall. Workers running tasks of the same communicating job cannot be firewalled off from each other, because their MPI-based communication will not work. • If users will schedule communicating jobs on the cluster, each cluster host participating in communicating jobs requires unchallenged SSH. For example, to enable unchallenged SSH the cluster admin can set up host based authentication/host based authorization for approved cluster users.

See Also

"Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager" on page 3-16 | "Install and Configure MATLAB Parallel Server for Third-Party Schedulers" on page 3-12

Use Different MPI Builds on UNIX Systems

In this section...

"Build MPI" on page 2-4
"Use Your MPI Build" on page 2-4

Build MPI

On Linux® operating systems, you can use an MPI build that differs from the one provided with Parallel Computing Toolbox. This topic outlines the steps for creating an MPI build for use with the generic scheduler interface. If you already have an alternative MPI build, proceed to "Use Your MPI Build" on page 2-4.

1. Unpack the MPI sources into the target file system on your machine. For example, suppose you have downloaded `mpich2-distro.tgz` and want to unpack it into `/opt` for building:

   ```
   # cd /opt
   # mkdir mpich2 && cd mpich2
   # tar zxvf path/to/mpich2-distro.tgz
   # cd mpich2-1.4.1p1
   ```
2. Build your MPI using the `enable-shared` option (this is vital, as you must build a shared library MPI, binary compatible with `MPICH2-1.4.1p1` for R2013b to R2018b, or `MPICH3.2.1` for R2019a and later). For example, the following commands build an MPI with the `nemesis` channel device and the `gforker` launcher.

   ```
   #./configure -prefix=/opt/mpich2/mpich2-1.4.1p1 \
    --enable-shared --with-device=ch3:nemesis \
    --with-pm=gforker 2>&1 | tee log
   # make 2>&1 | tee -a log
   # make install 2>&1 | tee -a log
   ```

Use Your MPI Build

When your MPI build is ready, this stage highlights the steps to use it with a generic scheduler. To get your cluster working with a different MPI build, follow these steps.

1. Test your build by running the `mpiexec` executable. The build should be ready to test if its `bin/mpiexec` and `lib/libmpich.so` are available in the MPI installation location.

   Following the example in "Build MPI" on page 2-4, `/opt/mpich2/mpich2-1.4.1p1/bin/mpiexec` and `/opt/mpich2/mpich2-1.4.1p1/lib/libmpich.so` are ready to use, so you can test the build with:

   ```
   $ /opt/mpich2/mpich2-1.4.1p1/bin/mpiexec -n 4 hostname
   ```
2. Create an `mpiLibConf` function to direct Parallel Computing Toolbox to use your new MPI. Write your `mpiLibConf.m` to return the appropriate information for your build. For example:

   ```
   function [primary, extras] = mpiLibConf
   primary = '/opt/mpich2/mpich2-1.4.1p1/lib/libmpich.so';
   extras  = {};
   ```

   The `primary` path *must* be valid *on the cluster*; and your `mpiLibConf.m` file must be higher on the cluster workers' path than `*matlabroot*/toolbox/parallel/mpi`. (Sending `mpiLibConf.m` as an attached file for this purpose does not work. You can get the `mpiLibConf.m` function on the worker path by either moving the file into a folder on the path, or by having the scheduler use `cd` in its command so that it starts the MATLAB worker from within the folder that contains the function.)

3. Determine necessary daemons and command-line options.

   - Determine all necessary daemons (often something like `mpdboot` or `smpd`). The `gforker` build example in this section uses an MPI that needs no services or daemons running on the cluster, but it can use only the local machine.
   - Determine the correct command-line options to pass to `mpiexec`.

4. To set up your cluster to use your new MPI build, modify your communicating job wrapper script to pick up the correct `mpiexec`. Additionally, there might be a stage in the wrapper script where the MPI process manager daemons are launched.

   The communicating job wrapper script must:

   - Determine which nodes are allocated by the scheduler.
   - Start required daemon processes. For example, for the MPD process manager this means calling `"mpdboot -f <nodefile>"`.
   - Define which `mpiexec` executable to use for starting workers.
   - Stop the daemon processes. For example, for the MPD process manager this means calling `"mpdallexit"`.

   For examples of communicating job wrapper scripts, see “Sample Plugin Scripts” (Parallel Computing Toolbox).

Shut Down a Job Manager Cluster

In this section...

"Linux Operating Systems" on page 2-6
"Microsoft Windows Operating Systems" on page 2-7

If you are done using the job manager and its workers, you might want to shut down the server software processes so that they are not consuming network resources. You do not need to be at the computer running the processes that you are shutting down. You can run these commands from any machine with network access to the processes. The following sections explain shutting down the processes for different platforms.

Linux Operating Systems

Enter the commands of this section at the prompt in a shell.

Stopping the Job Manager and Workers

1. To shut down the job manager, enter the commands

   ```
   cd matlabroot/toolbox/parallel/bin
   ```

   (Enter the following command on a single line.)

   ```
   stopjobmanager -remotehost <job manager hostname> -name
   <MyJobManager> -v
   ```

   If you have more than one job manager running, stop each of them individually by host and name.

   For a list of all options to the script, type

   ```
   stopjobmanager -help
   ```
2. For each MATLAB worker you want to shut down, enter the commands

   ```
   cd matlabroot/toolbox/parallel/bin
   stopworker -remotehost <worker hostname> -v
   ```

   If you have more than one worker session running, you can stop each of them individually by host and name.

   ```
   stopworker -name worker1 -remotehost <worker hostname>
   stopworker -name worker2 -remotehost <worker hostname>
   ```

   For a list of all options to the script, type

   ```
   stopworker -help
   ```

Stop and Uninstall the mjs Daemon

Normally, you configure the mjs daemon to start at system boot time and continue running until the machine shuts down. However, if you plan to uninstall the MATLAB Parallel Server product from a machine, you might want to uninstall the mjs daemon also, because you no longer need it.

Note You must have `root` privileges to stop or uninstall the mjs daemon.

1 Use the following command to stop the mjs daemon:

```
/etc/init.d/mjs stop
```

2 Remove the installed link to prevent the daemon from starting up again at system reboot:

```
cd /etc/init.d/
rm mjs
```

Stop the Daemon Manually

If you used the alternative manual startup of the mjs daemon, use the following commands to stop it manually:

```
cd matlabroot/toolbox/parallel/bin
mjs stop
```

Microsoft Windows Operating Systems

Stop the Job Manager and Workers

Enter the commands of this section at the prompt in a Windows command prompt window.

1 To shut down the job manager, enter the commands

```
cd matlabroot\toolbox\parallel\bin
```

(Enter the following command on a single line.)

```
stopjobmanager -remotehost <job manager hostname> -name
<MyJobManager> -v
```

If you have more than one job manager running, stop each of them individually by host and name.

For a list of all options to the script, type

```
stopjobmanager -help
```

2 For each MATLAB worker you want to shut down, enter the commands

```
cd matlabroot\toolbox\parallel\bin
stopworker -remotehost <worker hostname> -name <worker name> -v
```

If you have more than one worker session running, you can stop each of them individually by host and name.

```
stopworker -remotehost <worker hostname> -name <worker1 name>
stopworker -remotehost <worker hostname> -name <worker2 name>
```

For a list of all options to the script, type

```
stopworker -help
```

Stop and Uninstall the mjs Service

Normally, you configure the mjs service to start at system boot time and continue running until the machine shuts down. If you need to stop the mjs service while leaving the machine on, enter the following commands at a Windows command prompt:

```
cd matlabroot\toolbox\parallel\bin
mjs stop
```

If you plan to uninstall the MATLAB Parallel Server product from a machine, you might want to uninstall the mjs service also, because you no longer need it.

You do not need to stop the service before uninstalling it.

To uninstall the mjs service, enter the following commands at a Windows command prompt:

```
cd matlabroot\toolbox\parallel\bin
mjs uninstall
```

Define MATLAB Job Scheduler Startup Parameters

MATLAB Job Scheduler service runs using several parameters. These parameters set the process name, the user name, log file location, ports, and so on of the MATLAB Job Scheduler cluster.

You can edit the parameters in the `mjs_def` before installing or starting the mjs service. Find this file in these locations:

- `*matlabroot*\toolbox\parallel\bin\mjs_def.bat` on Microsoft Windows operating systems
- `*matlabroot*/toolbox/parallel/bin/mjs_def.sh` on Linux operating systems.

The `mjs` service reads the `mjs_def` file when you start or stop the `mjs` service, run the `mjs` service with the arguments console, or install or uninstall the `mjs` service on a Windows operating system. When you start and stop workers and job managers, they contact the `mjs` service they are running under to obtain the values of the definitions and defaults stored in this file. Thus, the `mjs` service does not read the `mjs_def` file again when starting and stopping workers and job managers.

For a working cluster setup, the `mjs_def` files must use consistent settings across all hosts of the cluster.

Note If you want to run more than one job manager on the same host, they must all have unique names. Specify the names using flags with the startup commands.

The user-configurable parameters in the `mjs_def` file are defined below.

MJS Process and Logging

Parameter	Description
HOSTNAME	Name of the host reading the MJS file. The processes under the `mjs` advertise themselves with the host name, `HOSTNAME`. The job manager must be able to resolve the host name that the MATLAB workers advertise, and all MATLAB workers and clients must be able to resolve the host name that the job manager advertises. On Windows operating systems, the default `HOSTNAME` is the fully qualified domain name (FQDN) of the host. The `mjs` service can use the FQDN if the client resolves the short name of the host to a different host. The `mjs` service uses this command to determine the FQDN of the host. `for /F "usebackq skip=2 tokens=4" %%A in (``reg query HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Servi` `        set strHostname=%%A` `)`  `for /F "usebackq skip=2 tokens=4" %%A in (``reg query HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Servi` ` set strDomain=%%A` `)` `HOSTNAME=%strHostname%.%strDomain%`  On Linux operating systems, the `mjs` service determines the default `HOSTNAME` with this command. ``HOSTNAME=`hostname -f` `` Change this setting only if: • the code or command above returns an incorrect host name, or • the command above is not available, or • the code or command above returns a host name that cannot be resolved by all the other computers that the processes under the `mjs` service communicates with

Parameter	Description
BASE_PORT	Base port of the mjs service. The mjs service uses as many ports as required, starting with BASE_PORT. By default, BASE_PORT is 27350. If you use a host that runs a total of nJ job managers and nW workers, the mjs service reserves a total of 6+2*nJ+4*nW consecutive ports for its own use. All job managers and workers, even those on different hosts, that are going to work together must use the same base port. Otherwise the job managers and workers will not be able to contact each other. In addition, MPI communication occurs on ports starting at BASE_PORT +1000 and use 2*nW consecutive ports. For example, if you use a host with 1 job manager and 16 workers, then you need the following ranges of ports to be open: • 27350 – 27422 for the mjs service. • 28350 – 28382 for MPI communication. To connect from MATLAB to a cluster with a non-default BASE_PORT, you must append the value of BASE_PORT to the 'Host' property in the MATLAB Job Scheduler cluster profile. You must do this in the form Hostname:BASE_PORT, for example myMJSHost:44001.
MJSUSER	Username to start the mjs service. By default, all the processes that the mjs service manages are associated with the user who starts the mjs service. If MJSUSER is not specified, the mjs service runs as LocalSystem on Windows operating systems, or as root user on Linux operating systems. You can set this parameter to run the mjs service as a different user from the user who starts the service. On Windows operating systems, set the value before installing the service; on Linux operating systems, set the value before starting the service. The MJSUSER must have access to the MATLAB installation folder. You can specify MJSUSER as DOMAINNAME\USERNAME, or as .\USERNAME if the user is a local user. For more information about the privileges and requirements for the MJSUSER, see “Set the User” on page 2-21.

Parameter	Description
`MJSPASS`	Password for username specifies in the `MJSUSER` parameter (Windows operating systems only). To run the `mjs` service as `MJSUSER`, you can specify a password for that user. You can either set the password in the `mjs.def` file or, if unset, the `mjs` service prompts you for a password when you start the `mjs` service. If you specify the password in the `mjs_def` file and the password uses characters that are treated as special characters in a batch file, they must be replaced with an appropriate escape sequence: • A single exclamation point (!) in the password must be given as `^^!`. • A single percent (%) in the password must be given as %%. • A single caret (`^`) in the password must be given as `^^^^`. • Other special characters & `< > [ ] { } = ; ' + , `` ` `` ~` must be escaped with a single preceding `^`.
`PIDBASE` and `LOCKBASE`	Folder used to store PID and lock files. (Linux operating systems only.) The `PIDBASE` and `LOCKBASE` folders contain the PID and lock files for the `mjs` service. The default for `PIDBASE` is `/var/run` and the default for `LOCKBASE` is `/var/lock/subsys`. You must ensure that the user the `mjs` service runs as has write access to this folder.
`LOGBASE`	Folder used to store `mjs` log files. The `LOGBASE` folder contains the log files the `mjs` service generates during the normal course of operation. • On Microsoft Windows operating systems, the default `LOGBASE` folder path is `%PROGRAMDATA%\MJS\Log`, where `%PROGRAMDATA%` is the system program data environment variable. For example, if `%PROGRAMDATA%` is set to `C:\ProgramData`, the log files are placed in `C:\ProgramData\MJS\Log`. You must not enclose the folder name in double quotes and the folder name must not contain parenthesis. • On Linux operating systems, the default is `/var/log/mjs`. You must ensure that the user the `mjs` service runs as has write access to this folder. For security level 3 worker hosts, the user the `mjs` service runs must also be able to grant write privileges to this folder to all users who run jobs and tasks on the host.

Parameter	Description
`CHECKPOINTBASE`	Folder used to store `mjs` checkpoint folders. The `CHECKPOINTBASE` folder contains persistent information related to the `mjs` service. • On Microsoft Windows operating systems, the default `CHECKPOINTBASE` folder path is `%PROGRAMDATA%\MJS\Checkpoint`, where `%PROGRAMDATA%` is the system program data environment variable. For example, if `%PROGRAMDATA%` is set to `C:\ProgramData`, the checkpoint folders are placed in `C:\ProgramData\MJS\Checkpoint`. You must not enclose the folder name in double quotes and the folder name must not contain parenthesis. • On Linux operating systems, the default is `/var/lib/mjs`. On the host that runs the job manager, the checkpoint folder stores database information related to the job manager and can require substantial diskspace. On the hosts that runs the workers, the checkpoint folder stores data transferred with tasks for workers. You must ensure that the user the `mjs` service runs as has write access to this folder.
`MJS_JAVA`	Set the Java® Runtime Environment (JRE™) path for the `mjs` service. By default, the `mjs` service uses the Java installation that ships with MATLAB. To use your own JRE installation for the `mjs` service, specify a path to an installed JRE. For information about Java versions compatible with MATLAB, see MATLAB Interfaces to Other Languages.

Job Manager Security

Parameter	Description
`SECURITY_LEVEL`	Security level for the cluster. You can specify the security level for your MATLAB Job Scheduler cluster. Select from one of these levels: **1** Level 0 – No security features are enabled. This is the default security level. Jobs are not protected. **2** Level 1 – this level warns users when they try to access other users' jobs and tasks. However, all users can still perform all actions. **3** Level 2 – users must enter a password to access their jobs and tasks. Other users do not have access unless it is granted by the job owner through the `AuthorizedUsers` job property. **4** Level 3 – same as level 2 except this level runs jobs and tasks on the workers as the user to which they belong. The password must be the system or network password used to log on to a worker host. Only trusted workers (i.e. workers with an identical shared secret file) can connect to the job manager, see `SHARED_SECRET_FILE` below. You must set the `USE_SECURE_COMMUNICATION` parameter to `true`. On Linux operating systems, you must run the `mjs` service as the root user. For more information, see “Set Security Level” on page 2-29.
`ADMIN_USER`	Username of the cluster administrator account. At Security Level 2 and 3, the administrator can access and manipulate all users' jobs and tasks. They can also change users' passwords if LDAP server authentication is not enabled. The `mjs` service prompts you to provide a password for the administrator account when you start the job manager or perform administrator operations. If LDAP server authentication is enabled, the username specified in the `ADMIN_USER` parameter must be a valid username in the LDAP server. To authenticate the administrator's username against the LDAP server, you must provide the password for the username specified in `ADMIN_USER` when you start the job manager. If you do not specify a username, the cluster administrator account username defaults to `admin`.

Parameter	Description
USE_SECURE_COMMUNICATION	Use encrypted communication between services. Set up encrypted communication between job manager, client and workers. By default, job managers and workers communicate over non-encrypted channels. This can be useful when you do not have a requirement to protect the data or the connection is already protected from unauthorized access (for example, if your cluster network is isolated and has no access to the Internet). Additionally, encrypted communication can cause a performance decrease due to the extra overhead incurred when encrypting messages. When you set USE_SECURE_COMMUNICATION to true, communication between the MATLAB client, job managers, and workers is encrypted. Encrypted communication requires a shared secret file on each participating host (excluding the MATLAB client). For more information about the shared secret file, including how to generate one, see the description for SHARED_SECRET_FILE below. USE_SECURE_COMMUNICATION is enabled by default when you set the cluster to Security Level 3. To establish encrypted communication between MATLAB Job Scheduler, client and workers, see “Set Encrypted Communication” on page 2-33.
REQUIRE_SCRIPT_VERIFICATION	Require verification for privileged commands sent to the cluster. By default, REQUIRE_SCRIPT_VERIFICATION is set to false and the mjs service will run commands sent to it without any verification. This allows anyone to modify the state of the cluster, for example, by starting or stopping worker or job manager processes. Privileged commands include any command which can modify the cluster state. When you set REQUIRE_SCRIPT_VERIFICATION to true, the mjs service verifies each command with the secret file before execution. Command verification is enabled by default when you set the cluster to Security Level 3. For more information, see “Set Cluster Command Verification” on page 2-34.

Parameter	Description
`SHARED_SECRET_FILE`	Location of the shared secret file . To establish encrypted connections between job managers and workers, a shared secret file is required on all participating hosts. You can create a shared secret file with the `createSharedSecret` script. For more information, see “Create Shared Secret File” on page 2-32. If the shared secret file does not already exist when you start the `mjs`, the `mjs` service creates one in the default location: `%CHECKPOINTBASE%\security\secret`. When you set the cluster to Security Level 3, the `mjs` uses the shared secret file to ensure only trusted workers can connect to the job manager. **Note** Secret files contain sensitive data and you must protect it against unauthorized access. Anyone that can access the secret file might be able to eavesdrop on the connections between services.
`ALLOW_CLIENT_PASSWORD_CACHE`	Remember user passwords for future sessions. If set to `true`, this option allows users to let the MATLAB client remember their login information for future client sessions. Users can still choose to not store any information at the password prompt in MATLAB.
`ALLOWED_USERS`	List of users allowed to log on to the job manager. This property defines a list of users that are allowed to access the job manager. Multiple usernames must be separated by commas. By default, `ALLOWED_USERS` is set to `ALL`. To allow any user to access the job manager, use the keyword `ALL` instead of a list of usernames.
`REQUIRE_CLIENT_CERTIFICATE`	Require MATLAB clients to present a certificate to connect to the job manager when using encrypted communication. By default, `REQUIRE_CLIENT_CERTIFICATE` is set to `false`. If you set `REQUIRE_CLIENT_CERTIFICATE` to `true`, cluster users must have a cluster profile containing the certificate to connect to the job manager. For more information, see “Set MATLAB Client Verification” on page 2-33.

Parameter	Description
`WORKER_DOMAIN`	Windows domain workers use when logging on at security level 3 (Windows operating systems only). To run tasks as the user that submitted them, Windows requires a domain in addition to the user name. By default, if a task belongs to `USER`, it will be run as `%WORKER_DOMAIN%\USER`. In most circumstances the default value is correct and must not be altered. **Note** This property is required only when running on Windows at security level 3.
`USE_LDAP_SERVER_AUTHENTICATION`	Option to use an LDAP server to authenticate user credentials. By default, `USE_LDAP_SERVER_AUTHENTICATION` is set to `false`. To enable LDAP server authentication for a Security Level 2 or 3 cluster, set this parameter to `true`. For more information, see “Configure LDAP Server Authentication for MATLAB Job Scheduler” on page 2-43. Cluster users must use their LDAP server username and password to connect to the cluster.

`LDAP_URL`	URL of the LDAP server to authenticate users.
	
	If you have configured the cluster to use LDAP server authentication, you must set the `LDAP_URL`. Specify the `LDAP_URL` as:
	
	`ldap://HOST:PORT`
	
	If you have configured your LDAP server over SSL, specify the URL as:
	
	`ldaps://HOST:PORT`
	
	For more information, see “Configure LDAP Server Authentication for MATLAB Job Scheduler” on page 2-43.

`LDAP_SECURITY_PRINCIPAL_FORMAT`	Format of a security principal for the LDAP server. You can specify the format of a security principal the LDAP server uses for authentication. Common formats include `"[username]@domain.com"` and `"cn=[username],ou=Users,dc=domain,dc=com"`. The username of the principal replaces the `[username]` token during authentication.

Parameter	Description
`LDAP_SYNCHRONIZATION_INTERVAL_SECS`	Frequency at which the cluster synchronizes with the LDAP server. The default value is `1800` seconds, which will synchronize the cluster with the LDAP server every half hour. You can specify one of the following: • A positive number corresponding to the number of seconds between synchronizations. Access to the cluster might be granted for old/expired credentials until the next synchronization. • `0`. The cluster synchronizes with the LDAP server on every cluster access requiring authentication.

Job Manager and Worker Settings

Parameter	Description
`DEFAULT_JOB_MANAGER_NAME`	Default name of the job manager. When you start a new job manager, it needs to be identified by a name on the network, and when a new worker is started, it needs to know the name of the job manager to register with. The values specified with the `DEFAULT_JOB_MANAGER_NAME` parameter is the default job manager name used in both cases. To override the default job manager name, use the `startjobmanager` command with the `-name` flag. To change the job manager name on a worker host, use the `startworker` command with the `-jobmanager` flag.
`JOB_MANAGER_HOST`	Host on which the job manager lookup process runs. The MATLAB worker processes and the job manager process contact the job manager lookup process on the specified host. You can override the job manager host using the `startworker` command with the `-jobmanagerhost` flag.
`JOB_MANAGER_MAXIMUM_MEMORY`	Maximum heap size of the job manager Java process. You can adjust the amount of heap memory for the job manager Java process. By default, `JOB_MANAGER_MAXIMUM_MEMORY` is set to `1024m`. For recommendations on when to adjust the heap memory for the job manager, see “Increase Heap Memory” on page 2-22
`DEFAULT_WORKER_NAME`	Default name of the worker. You can override the default worker by calling the `startworker` command with the `-name` flag. Worker names must be unique on each host, therefore, you must use the `-name` flag with the `startworker` command if you want to start more than one worker on a single host.

Parameter	Description
WORKER_MAXIMUM_MEMORY	Maximum heap size of all worker Java process. You can adjust the amount of heap memory for the worker Java process. By default, WORKER_MAXIMUM_MEMORY is set to 1024m. The WORKER_MAXIMUM_MEMORY does not impact the amount of memory the MATLAB worker can use.
WORKER_START_TIMEOUT	Number of seconds for worker sessions wait for MATLAB to start before detecting a stall. The default value is 600s. You must select a value greater than the time it takes for a MATLAB session to start.
MATLAB_SHELL	Shell workers use in system calls (Linux operating systems only). You can specify the shell that is spawned when a worker invokes a MATLAB system command. For example, to use the Bourne shell, set MATLAB_SHELL to /bin/sh. MATLAB checks internally for the MATLAB_SHELL variable first and, if empty or not defined, then checks SHELL. If SHELL is also empty or not defined, MATLAB uses the Bourne shell, /bin/sh. The value of MATLAB_SHELL must be an absolute path, that is /bin/sh, not simply sh. Note that some shells (for example tcsh) might choose to interpret a system command in MATLAB as a request to start up a new login shell, rather than a subshell, which might have consequences for environment variable changes you have made in MATLAB, for example to the PATH environment variable that would allow other applications to run.
USE_ONLINE_LICENSING	Use a license that is managed online. Set USE_ONLINE_LICENSING to true to enable on-demand licensing or to use a license that is managed online. When enabled, users must log in to their MathWorks account to connect to the cluster, and their account must be linked to a MATLAB Parallel Server license that is managed online.
ALL_SERVER_SOCKETS_IN_CLUSTER	Initiate connections from the client to the cluster. To make all connections outbound from client, set ALL_SERVER_SOCKETS_IN_CLUSTER to true. To allow inbound connections to the client from the job manager and workers in addition to outbound connections from the client, set ALL_SERVER_SOCKETS_IN_CLUSTER to false.

Parameter	Description
MJS_ADDITIONAL_MATLABROOTS	Path to additional installations of previous MATLAB releases. MJS_ADDITIONAL_MATLABROOTS is required only when specifying installations of MATLAB that are different from the MATLAB Parallel Server version in use. You must list the path to the MATLAB installation without quotes. Multiple entries are allowed separated by semi-colons, for example: `MJS_ADDITIONAL_MATLABROOTS=C:\Program Files\MATLAB\R2016a;C:\Program File` Only versions of MATLAB released prior to the MATLAB Parallel Server version you are using are supported, starting with R2016a. MATLAB clients can only use this cluster if a path to their version is specified.
MAX_LINUX_WORKERS	The maximum number of Linux workers you can resize the cluster to on demand. Only set this value if you have set up your cluster for resizing.
MAX_WINDOWS_WORKERS	The maximum number of Windows workers you can resize the cluster to on demand. Only set this value if you have set up your cluster for resizing.

See Also

`startworker` | `stopworker` | `mjs`

Related Examples

- “Customize Startup Parameters” on page 2-21
- “Set up MATLAB Job Scheduler Cluster for Auto-Resizing” on page 2-24
- “Set MATLAB Job Scheduler Cluster Security” on page 2-29
- “Configure Advanced Options for MATLAB Job Scheduler Integration” on page 3-29

Customize Startup Parameters

In this section...
"Modify Script Defaults" on page 2-21
"Override Script Defaults" on page 2-22
"Increase Heap Memory" on page 2-22

The MATLAB Parallel Server scripts run using several default parameters. You can customize the scripts, as described in this section.

Modify Script Defaults

The scripts for the server services require values for several parameters. You can set some parameters using flags at the command line, but the full set of user-configurable parameters are in the `mjs_def` file. To learn more about all the parameters in the `mjs_def` file, see "Define MATLAB Job Scheduler Startup Parameters" on page 2-9.

The default parameters used by the server service scripts are defined in the file:

- `*matlabroot*\toolbox\parallel\bin\mjs_def.bat` (on Microsoft Windows operating systems)
- `*matlabroot*/toolbox/parallel/bin/mjs_def.sh` (on Linux operating systems)

To modify the default parameters, edit this file before installing or starting the `mjs` service.

Set the User

By default, the job manager and worker services run as the user who starts them. To run the services as a different user, modify the `MJSUSER` and `MJSPASS` parameters in the `mjs_def` file.

On UNIX operating systems, `MJSUSER` requires that the current machine has the `sudo` utility installed, and that the current user be allowed to use `sudo` to execute commands as the user identified by `MJSUSER`. For further information, refer to your system documentation on the `sudo` and `sudoers` utilities (for example, `man sudo` and `man sudoers`).

The `MJSUSER` is granted these permissions on Windows systems:

Privilege	Purpose	Local Security Settings Policy
SeServiceLogonRight	Required to log on using the service logon type.	Log on as a service
SeAssignPrimaryTokenPrivilege	Required to start a process under a different user account.	Replace a process level token
SeIncreaseQuotaPrivilege	Required to start a process under a different user account.	Adjust memory quotas for a process

To modify or remove these privileges,

1. Select the Windows menu **Start** > **Settings** > **Control Panel**.

2 Double-click **Administrative Tools**, then **Local Security Policy**.
3 In the tree, select **Local Policies > User Rights Assignment**.

The table above indicates which policies are affected by `MJSUSER`. Double-click any of the listed policies in the Local Security Settings GUI to alter its setting or remove a user from that policy.

Override Script Defaults

Specify an Alternative Defaults File

The default parameters used by the `mjs` service, job managers, and workers are defined in the file:

- *`matlabroot`*`\toolbox\parallel\bin\mjs_def.bat` (on Windows operating systems)
- *`matlabroot`*`/toolbox/parallel/bin/mjs_def.sh` (on Linux operating systems)

Before installing and starting the `mjs` service, you can edit this file to set the default parameters with values you require.

Alternatively, you can make a copy of this file, modify the copy, and specify that this copy be used for the default parameters.

On Linux operating systems, enter the command

```
mjs start -mjsdef my_mjs_def.sh
```

On Windows operating systems, enter the command

```
mjs install -mjsdef my_mjs_def.bat
mjs start -mjsdef my_mjs_def.bat
```

If you specify a new `mjs_def` file instead of the default file for the service on one computer, the new file is not automatically used by the `mjs` service on other computers. If you want to use the same alternative file for all your `mjs` services, you must specify it for each `mjs` service you install or start.

For more information, see “Modify Script Defaults” on page 2-21.

Note The startup script flags take precedence over the settings in the `mjs_def` file.

Start in a Clean State

When a job manager or worker starts up, it normally resumes its session from the past. This way, a job queue is not destroyed or lost if the job manager machine crashes or if the job manager is inadvertently shut down. To start up a job manager or worker from a clean state, with all history deleted, use the `-clean` flag on the `start` command:

```
startjobmanager -clean -name MyJobManager
startworker -clean -jobmanager MyJobManager
```

Increase Heap Memory

An `mjs` service can use up to 4000 workers across a collection of nodes. When you scale up the number of workers or tasks in your cluster, you must increase the heap memory available to the job manager. To do so, set the `JOB_MANAGER_MAXIMUM_MEMORY` parameter in the `mjs_def` file based on the following recommendations.

- Use a minimum value of `1000m` (1000 MiB).
- Use `1000m` for every 1000 workers.
- Add `1000m` for every 100,000 tasks expected to be queued at peak load.

For example, for a cluster with 4000 workers and a peak queue size of 200,000 tasks, set `JOB_MANAGER_MAXIMUM_MEMORY` to `6000m`.

See Also

Related Examples

- "Define MATLAB Job Scheduler Startup Parameters" on page 2-9
- "Set up MATLAB Job Scheduler Cluster for Auto-Resizing" on page 2-24

Set up MATLAB Job Scheduler Cluster for Auto-Resizing

You can customize your MATLAB Job Scheduler (MJS) cluster to resize automatically. By default, an MJS cluster does not have the resizing functionality enabled. This means that MJS immediately rejects any work you submit to the cluster that requires more than the current number of workers in the cluster. Auto-resizing, also called auto-scaling, allows you to submit such work to the cluster and makes the number of workers in the cluster change automatically with the amount of work submitted. The cluster grows (scales up) when there is more work to do and shrinks (scales down) when there is less work to do. This allows you to use your compute resources more efficiently and can result in cost savings.

To configure your MJS cluster to resize automatically, you need to:

1. Set the maximum number of workers in the `mjs_def` file.
2. Start an MJS cluster.
3. Set up an auto-resizing process.

Set Maximum Number of Workers

To make an MJS cluster resizable, you need to define the maximum number of workers of your cluster by editing the `mjs_def` file as follows:

1. Open the file `mjs_def.sh` (on Linux) or `mjs_def.bat` (on Windows) located at `matlabroot/toolbox/parallel/bin`, where `matlabroot` is the directory of your MATLAB installation.
2. Uncomment one or both of the lines `#MAX_LINUX_WORKERS=` and `#MAX_WINDOWS_WORKERS=` and set them to the desired values. These variables define the maximum number of Linux and Windows workers to which you can resize the cluster, respectively.

A resizable MJS cluster allows jobs in the queue that require more than the current number of workers in the cluster, up to the amount specified in MAX_LINUX_WORKERS and MAX_WINDOWS_WORKERS. Other jobs are cancelled immediately.

Note To change the maximum number of Linux and Windows workers after the cluster has started, use the `resize` script located at `matlabroot/toolbox/parallel/bin` to run the `resize update` command. For example:

```
% cd matlab/toolbox/parallel/bin
% ./resize update -jobmanager myJobManager -maxlinuxworkers 4 -maxwindowsworkers 8
```

Start MJS Cluster

To create a cluster with the options defined in the `mjs_def` file, start an MJS cluster after editing and saving this file. For more information about how to install, configure and start an MJS cluster, see "Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager" on page 3-16.

Set up Auto-Resizing Process

To make a resizable MJS cluster change size automatically, you must set up a background process to periodically adjust the size of the cluster. The specific implementation of this background process depends on many factors, but you can follow these general recommended steps:

1. Identify the desired size of the cluster. The desired size of a resizable MJS cluster is reported as the total number of workers for each operating system and hence includes all busy workers and some idle workers that are already in the cluster. The desired size changes based on running jobs and jobs in the queue. Use the `resize` script located at `matlabroot/toolbox/parallel/bin` to run the `resize status` command:

   ```
   % cd matlab/toolbox/parallel/bin
   % ./resize status
   ```

   The `resize status` command above returns information about the resizable cluster in JSON format:

   ```
   {
     "jobManagers": [
       {
         "name": "myJobManager",
         "host": "myhostname",
         "desiredWorkers": {
           "linux": 1,
           "windows": 0
         },
         "maxWorkers": {
           "linux": 4,
           "windows": 8,
         },
         "workers": [
           {
             "name": "worker_1",
             "host": "myhostname",
             "operatingSystem": "linux",
             "state": "busy",
             "secondsIdle": 0
           },
           {
             "name": "worker_2",
             "host": "myhostname",
             "operatingSystem": "linux",
             "state": "idle",
             "secondsIdle": 60
           }
         ]
       }
     ]
   }
   ```

   Parse the JSON output to extract the `desiredWorkers` values that represent the desired number of Linux and Windows workers for the MJS cluster.
2. Compare the desired number of workers with the workers in the cluster to decide whether you need to start or stop workers. Use the `workers` array in the output of the `resize status` command to examine the workers in the cluster. To ensure that jobs in the queue eventually run,

you must start enough workers to match or exceed the desired number of workers. You can optionally stop idle workers that exceed the desired number of workers.

Note If workers take a long time to start in your environment, you might want to wait for excess workers to be idle for some time before stopping them. This approach can be more efficient than immediately stopping excess idle workers if they are needed again soon after they become idle. To check how long a worker has been idle, examine the `secondsIdle` value for the worker.

---

3 Start or stop workers as necessary. To do this, use the `startworker` and `stopworker` utility scripts. To avoid interrupting any work when stopping workers, it is recommended that you use the `-onidle` flag with the `stopworker` command.

See Also
`startworker` | `stopworker` | mjs

Related Examples
- "Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager" on page 3-16
- "Start mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)" on page 3-32

Access Service Record Files

In this section...
“Locate Log Files” on page 2-27
“Locate Checkpoint Folders” on page 2-27

The MATLAB Parallel Server services generate various record files in the normal course of their operations. The `mjs` service, job manager, and worker sessions all generate such files. This section describes the types of information stored by the services.

Locate Log Files

Log files for each service contain entries for the service’s operations. These might be of particular interest to the network administrator in cases when problems arise.

Operating System	File Location
Windows	The default location of the log files is `<PROGRAMDATA>\MJS\Log`, where `<PROGRAMDATA>` is the value of the system `PROGRAMDATA` variable. For example, if `PROGRAMDATA` is set to `C:\ProgramData`, the log files are placed in `C:\ProgramData\MJS\Log`. You can set alternative locations for the log files by modifying the `LOGBASE` setting in the `mjs_def.bat` file before starting the `mjs` service.
Linux	The default location of the log files is `/var/log/mjs/`. You can set alternative locations for the log files by modifying the `LOGBASE` setting in the `mjs_def.sh` file before starting the `mjs` service.

Locate Checkpoint Folders

Checkpoint folders contain information related to persistence data, which the server services use to create continuity from one instance of a session to another. For example, if you stop and restart a job manager, the new session continues the old session, using all the same data.

A primary feature offered by the checkpoint folders is in crash recovery. This allows server services to automatically resume their sessions after a system goes down and comes back up, minimizing the loss of data. However, if a MATLAB worker goes down during the evaluation of a task, that task is neither reevaluated nor reassigned to another worker. In this case, a finished job may not have a complete set of output data, because data from any unfinished tasks might be missing.

Note If a job manager crashes and restarts, its workers can take up to 2 minutes to reregister with it.

Platform	File Location
Windows	The default location of the checkpoint folders is `<PROGRAMDATA>\MJS\Checkpoint`, where `<PROGRAMDATA>` is the value of the system `PROGRAMDATA` variable. For example, if `PROGRAMDATA` is set to `C:\ProgramData`, the checkpoint folders are placed in `C:\ProgramData\MJS\Checkpoint`. You can set alternative locations for the checkpoint folders by modifying the `CHECKPOINTBASE` setting in the `mjs_def.bat` file before starting the `mjs` service.
Linux	The checkpoint folders are placed by default in `/var/lib/mjs/`. You can set alternative locations for the checkpoint folder by modifying the `CHECKPOINTBASE` setting in the `mjs_def.sh` file before starting the `mjs` service.

Set MATLAB Job Scheduler Cluster Security

In this section...

“Set Security Level” on page 2-29

“Local, MATLAB Job Scheduler, and Network Passwords” on page 2-32

“Authorize Users for Job and Task Access” on page 2-32

“Create Shared Secret File” on page 2-32

“Set Encrypted Communication” on page 2-33

“Set MATLAB Client Verification” on page 2-33

“Set Cluster Command Verification” on page 2-34

Set Security Level

Set the MATLAB Job Scheduler security level with the `SECURITY_LEVEL` parameter in the `mjs_def` file before starting the `mjs` service on your cluster nodes. The `mjs_def` file indicates what values you can set and briefly describes each security level.

This table describes the available security levels for accessing MATLAB Job Scheduler and its jobs.

Security Level	Description	User Restrictions
0	No security. • Any user can access any job. • Tasks are associated with the user who started the mjs process on the worker machines (typically root or Local System). • This is the default level and is the security level in all releases prior to R2010b. • Jobs are associated with the default username, but the software provides no protection.	None
1	Jobs are associated with the submitting user. • Any user can access any job. A dialog box warns if the accessed job belongs to another user. • Tasks are associated with the user who starts the `mjs` process on the worker machines (typically root or Local System).	• A dialog box prompts you to specify a username when you first access the job manager. • Your MATLAB Job Scheduler username does not have to match your system or network username. • You do not require a password.

Security Level	Description	User Restrictions
2	Jobs have MATLAB Job Scheduler password protection. • Jobs and tasks are associated with the submitting user and are password protected. The submitting user can authorize other users to access their jobs and tasks. In this case, an authorized user can enter their own password to access the jobs and tasks. Other unauthorized users cannot access your jobs. • Tasks are associated with the user who started the `mjs` process on the worker machines (typically root or Local System).	• When you start MATLAB Job Scheduler, you must provide a new password for the job manager administrator account. You can use this account to access all jobs and tasks. If you use LDAP server authentication, you must provide the LDAP server password of the administrator account when MATLAB Job Scheduler prompts you. • A dialog box prompts you to specify a username and password when you first access MATLAB Job Scheduler from your MATLAB client session. • Your MATLAB Job Scheduler username and password do not have to match your system or network username and password. If you use LDAP server authentication, your MATLAB Job Scheduler username and password must match the username and password in the LDAP server.

Security Level	Description	User Restrictions
3	In addition to the security of level 2, tasks are associated with the submitting user on worker machines. • Jobs and tasks are associated with the submitting user and are password protected. Other unauthorized users cannot access your jobs. • Tasks are associated with the user who submitted the job.	• MATLAB Job Scheduler must use encrypted communication with the workers. For more information, see "Set Encrypted Communication" on page 2-33. • When you start MATLAB Job Scheduler, you must provide a new password for the job manager's administrator account. You can use this account to access all jobs and tasks. If you use LDAP server authentication, you must provide the LDAP server password of the administrator account when MATLAB Job Scheduler prompts you. • A dialog box prompts you to specify a user name and password when you first access MATLAB Job Scheduler from your MATLAB client. • Your job manager MATLAB Job Scheduler username and password must be the same as your system or network username and password because the parallel workers must log you in to run the task as you. • Read and write permissions to the `CHECKPOINTBASE` folder and all its subfolders must be restricted to the user who starts the `mjs` process. • On UNIX systems, the root user must start the `mjs` process on the cluster nodes. • On Windows systems, the submitting user must be able to log on locally to every worker machine to successfully run jobs on the cluster. You must grant each submitting user account the `"Allow log on locally"` right. If you disable this right, all the jobs you submit will fail when the cluster starts them. To enable this right, change the `SeInteractiveLogonRight` constant for the user in the User Rights Assignment security policy settings for each machine in the cluster.

Tip Run the job manager and the workers at the same security level. The job manager does not register a worker running at a lower security level.

Local, MATLAB Job Scheduler, and Network Passwords

For any security level above level 0, when you start MATLAB Job Scheduler (for example, with the `startjobmanager` command), the software creates a cluster administrator account with the username specified in the `ADMIN_USER` parameter in the `mjs_def` file. If you do not specify a username, the administrator account username defaults to `admin`. The software prompts you to provide a password for the new administrator account. The administrator account has all the necessary permissions for accessing the cluster and all its jobs. To use LDAP server authentication, the username specified in `ADMIN_USER` must be in the LDAP server.

For any security level, MATLAB Job Scheduler associates every job with the user who submits it. Therefore, whenever you access MATLAB Job Scheduler or a job, MATLAB Job Scheduler must verify your identity.

At security level 0, the software sets the `Username` property to the login name of the person who creates the job. You can change this value at any time. For all higher security levels, the first time you access MATLAB Job Scheduler, a dialog box prompts you for your username. If the security level is 2 or 3, you must also provide a password. The username and password you provide for MATLAB Job Scheduler must match your network username and password if you are using security level 3 or if the MATLAB Job Scheduler cluster has LDAP server authentication configured. Otherwise, you can create a new username and password for MATLAB Job Scheduler. For convenience, you can choose to save your username and password on the local computer so you do not need to enter them every time you access your job.

For information about changing a password and logging out of a MATLAB Job Scheduler cluster, see `changePassword` and `logout`. For more information about LDAP server authentication for MATLAB Job Scheduler clusters, see "Configure LDAP Server Authentication for MATLAB Job Scheduler" on page 2-43.

Authorize Users for Job and Task Access

This example shows how to authorize users to access your job on a MATLAB Job Scheduler cluster with security level 2 or 3. When you create a job and submit it to a MATLAB Job Scheduler cluster, jobs and tasks are associated with the submitting user. These jobs and tasks are password protected so unauthorized users cannot access your jobs.

Use `parcluster` (Parallel Computing Toolbox) to create a cluster object using the cluster profile `'MyMJSCluster'`. Replace `'MyMJSCluster'` with the name of your cluster profile. Then, use `batch` (Parallel Computing Toolbox) to create and submit a job on the cluster.

```
c = parcluster('MyMJSCluster');
j = batch(c,@rand,1,{2});
```

You can set the `AuthorizedUsers` property of a job to authorize user access to that job and its tasks. Each user that you specify must have already used the MATLAB Job Scheduler cluster. Authorize access to a job for users `"user1"` and `"user2"`.

```
j.AuthorizedUsers = ["user1","user2"];
```

Create Shared Secret File

The secret file establishes trust between the processes on different machines.

To create this file, run one of these scripts:

- `matlabroot/toolbox/parallel/bin/createSharedSecret` on Linux operating systems
- `matlabroot\toolbox\parallel\bin\createSharedSecret.bat` on Windows operating systems

Specify the location of the secret file in the `SHARED_SECRET_FILE` parameter in the `mjs_def` file to enable MATLAB Job Scheduler to find it. The shared secret file contains sensitive data and must be read-only for the user who starts the `mjs` process

- In a shared file system, all the nodes can point to the same secret file. The nodes can also share the same `mjs_def` file.
- In a nonshared file system, create a secret file with the provided script, then copy the file to each node and make sure the `mjs_def` file of each node indicates the location of its secret file.

Set Encrypted Communication

To set encrypted communication between MATLAB Job Scheduler, the client, and the workers, set these values in the `mjs_def` file:

- `USE_SECURE_COMMUNICATION = true`
- `ALL_SERVER_SOCKETS_IN_CLUSTER = true`

Encrypted communication is provided using TLSv1.3.

Encrypted communication is provided via an SSLSocket using TLSv1.2.

Note If you specify `ALL_SERVER_SOCKETS_IN_CLUSTER` as `false` in the `mjs_def` file, then the `mjs` service establishes encrypted communication between MATLAB Job Scheduler and workers only. Communication between workers is never encrypted. If communication between a worker and the client is sent via another worker, only the communication between that worker and the client is encrypted.

Additionally, all hosts that run job managers or workers require the secret file at the location specified by the `SHARED_SECRET_FILE` parameter in the `mjs_def` file. To create the secret file, see "Create Shared Secret File" on page 2-32.

Note Encrypted communication is required when you use MATLAB Job Scheduler security level 3.

Set MATLAB Client Verification

Verify whether a MATLAB client can connect to your MATLAB Job Scheduler cluster.

You must use the same secret file as the cluster to create a certificate file. Use the certificate when you start the job manager and to create a certified cluster profile. To create the secret file, see "Create Shared Secret File" on page 2-32.

Connections between the MATLAB client and MATLAB Job Scheduler cluster are verified using mutual TLS (mTLS).

Configure MATLAB Job Scheduler Cluster

In the `mjs_def` file, set `REQUIRE_CLIENT_CERTIFICATE` to `true`.

Navigate to one of these folders:

- `matlabroot\toolbox\parallel\bin` on Windows operating systems
- `matlabroot/toolbox/parallel/bin` on Linux operating systems

When you have a location for the secret file, use the `generateCerticate` command to generate the certificate. Specify the path to the secret file and the name of the certificate.

```
generateCertificate -secretfile path_to_shared_secret_file/secret -certfile mjsClusterClientCert
```

To start the job manger, specify the file path to the certificate to the `startjobmanager` command using the `-certificate` flag.

```
startjobmanager -certificate mjsClusterClientCert
```

Create Certified Cluster Profile

The MATLAB client must also have a cluster profile with the correct certificate to connect to the job manager.

To create the certified cluster profile, use the `createProfile` command. Specify the name and hostname of the cluster and the path of the certificate file. For example, create a cluster profile for the cluster `clusterName`, host name `mjsHost`, and certificate file `mjsClusterClientCert`. The command creates a cluster profile file `clusterName` with the extension `.mlsettings`. This file contains the certificate that the MATLAB client needs to connect to the job manager.

```
createProfile -name clusterName -host mjsHost -certfile mjsClusterClientCert
```

The certificate and associated cluster profile control which users can connect to the job manager. You must store this data securely and distribute the cluster profile to users through a secure channel. You can use the **Cluster Profile Manager** to import a profile into the MATLAB client. For more information, see “Discover Clusters and Use Cluster Profiles” (Parallel Computing Toolbox).

Set Cluster Command Verification

MATLAB Job Scheduler cluster administrators can restrict use of MATLAB Job Scheduler cluster commands to only specified users. Restrict command use to prevent unauthorized users from sending privileged commands to the cluster. Privileged commands are commands that can change the state of the cluster.

Privileged Commands

This table lists the privileged commands that require verification. You can find the executables for these commands in these folders:

- `matlabroot\toolbox\parallel\bin` on Windows operating systems
- `matlabroot/toolbox/parallel/bin` on Linux operating systems

Command	Description
`pausejobmanager`	Pause a job manager that is running under the `mjs` service.
`resize`	Determine or update resizing information for job manager processes under the `mjs` service.
`resumejobmanager`	Resume a job manager that is running under the `mjs` service.
`startjobmanager`	Start a job manager process and the associated job manager lookup process under the `mjs` service.
`startworker`	Start a MATLAB worker process under the `mjs` service.
`stopjobmanager`	Stop a job manager process and the associated job manager lookup process under the `mjs` service.
`stopworker`	Stop a MATLAB worker process under the `mjs` service.
`util/clusterlogs`	Set or get the log level of the `mjs` service.
`util/workerRegisterWithJobManager`	Register a MATLAB worker to a specified job manager.

Note Command verification is enabled by default when you set MATLAB Job Scheduler to Security Level 3.

Set Command Verification in mjs_def File

To require verification before the `mjs` service executes a privileged command on the cluster, set the `REQUIRE_SCRIPT_VERIFICATION` parameter to `true` in the `mjs_def` file.

You must also set the `SHARED_SECRET_FILE` parameter to the location of the secret file used by the `mjs` process you are trying to send commands to.

Specify the secret file using one of these options.

- Provide a secret file at the command line. You can provide the path to the secret file when you send a privileged command to the cluster.

  For example, to stop a worker in the cluster, enter this command in a Windows or Linux command:

  ```
  stopworker -name worker1 -secretfile path_to_shared_secret_file/secret
  ```
- Provide a path in the `mjs_def` file. If you are using the same cluster host as the location of the secret file, you can provide a path to the secret file in the `mjs_def` file.

  Only users that have read access to the secret file can run privileged commands on the cluster.

See Also
`startjobmanager` | `changePassword` | `logout` | `mjs`

Related Examples

- “Define MATLAB Job Scheduler Startup Parameters” on page 2-9
- “Configure LDAP Server Authentication for MATLAB Job Scheduler” on page 2-43

Troubleshoot Common Problems

In this section...

"License Errors" on page 2-37

"Memory Errors on UNIX Operating Systems" on page 2-38

"Run Server Processes on Windows Network Installation" on page 2-38

"Required Ports" on page 2-38

"Ephemeral TCP Ports with Job Manager" on page 2-39

"Host Communications Problems" on page 2-40

"Verify Network Communications for Cluster Discovery" on page 2-41

This section offers advice on solving problems you might encounter with MATLAB Parallel Server software.

License Errors

When starting a MATLAB worker, a licensing problem might result in the message

```
License checkout failed. No such FEATURE exists.
License Manager Error -5
```

There are many reasons why you might receive this error:

- This message usually indicates that you are trying to use a product for which you are not licensed. Look at your `license.dat` file located within your MATLAB installation to see if you are licensed to use this product.
- If you are licensed for this product, this error may be the result of having extra carriage returns or tabs in your license file. To avoid this, ensure that each line begins with either `#`, `SERVER`, `DAEMON`, or `INCREMENT`.

  After fixing your `license.dat` file, restart the network license manager and MATLAB should work properly.
- This error may also be the result of an incorrect system date. If your system date is before the date that your license was made, you will get this error.
- If you receive this error when starting a worker with MATLAB Parallel Server software:

  - You may be calling the `startworker` command from an installation that does not have access to a worker license. For example, starting a worker from a client installation of the Parallel Computing Toolbox product causes the following error:

    ```
    The mjs service on the host hostname
    returned the following error:

    Problem starting the MATLAB worker.

    The cause of this problem is:

    ==============================================================
       Most likely, the MATLAB worker failed to start due to a
       licensing problem, or MATLAB crashed during startup.  Check
    ```

```
   the worker log file
   /tmp/mjs_user/node_node_worker_05-11-01_16-52-03_953.log
   for more detailed information.  The mjs log file
   /tmp/mjs_user/mjs-service.log
   may also contain some additional information.
===============================================================
```

In the worker log files, you see the following information:

```
License checkout failed.
License Manager Error -15
MATLAB is unable to connect to the license server.
Check that the license manager has been started, and that the
MATLAB client machine can communicate with the license server.

Troubleshoot this issue by visiting:
https://www.mathworks.com/support/lme/R2009a/15

Diagnostic Information:
Feature: MATLAB_Distrib_Comp_Engine
License path: /apps/matlab/etc/license.dat
FLEXnet Licensing error: -15,570. System Error: 115
```

- If you installed only the Parallel Computing Toolbox product, and you are attempting to run a worker on the same machine, you will receive this error because the MATLAB Parallel Server product is not installed, and therefore the worker cannot obtain a license.

Memory Errors on UNIX Operating Systems

If the number of processes created by the server services on a machine running a Linux operating system exceeds the operating system limits, the services fail and generate an out-of-memory error. It is recommended that you adjust your system limits. For more information, see “Recommended System Limits for Macintosh and Linux” (Parallel Computing Toolbox).

Run Server Processes on Windows Network Installation

Many networks are configured not to allow `LocalSystem` to have access to UNC or mapped network shares. In this case, run the mjs process under a different user with rights to log on as a service. See “Set the User” on page 2-21.

Required Ports

With Job Manager

BASE_PORT

The `mjs_def` file specifies and describes the ports required by the job manager and all workers. See the following file in the MATLAB installation used for each cluster process:

- *`matlabroot`*`/toolbox/parallel/bin/mjs_def.sh` (on UNIX operating systems)
- *`matlabroot`*`\toolbox\parallel\bin\mjs_def.bat` (on Windows operating systems)

Communicating Jobs

On worker machines running a UNIX operating system, the number of ports required by MPICH for the running of communicating jobs ranges from `BASE_PORT + 1000` to `BASE_PORT + 2000`.

With Third-Party Scheduler

Communication Between Workers

Before the worker processes start, you can control the range of ports used by the workers for communicating jobs by defining the environment variable `MPICH_PORT_RANGE` with the value `minport:maxport`.

Open Ports on Workers for Inbound Communication from Client

You can control the listening port range workers open to connect to clients for interactive parallel pool jobs.

- Use the `pctconfig` function to specify which listening ports workers must open or
- Define the environment variable `PARALLEL_SERVER_OVERRIDE_PORT_RANGE` with the value `"minport maxport"`. This will override the port range specified with `pctconfig`.
  - For Microsoft HPC Pack, set `PARALLEL_SERVER_OVERRIDE_PORT_RANGE` in the job template with an addition to the Environments field. For example, to open a listening port in the range 30000 to 31000, add this code to the job template.

    ```
    PARALLEL_SERVER_OVERRIDE_PORT_RANGE=30000 31000;
    ```
  - For other third-party schedulers, set `PARALLEL_SERVER_OVERRIDE_PORT_RANGE` in the `communicatingJobWrapper.sh` script. For example, to open a listening port in the range 29000 to 31000, add this code to the `communicatingJobWrapper.sh` script.

    ```
    export PARALLEL_SERVER_OVERRIDE_PORT_RANGE="29000 31000"
    ```

    To learn more about the `communicatingJobWrapper.sh` script, see “Wrapper Scripts” (Parallel Computing Toolbox).

Client Ports

With the `pctconfig` function, you specify the ports used by the client. If the default ports cannot be used, this function allows you to configure ports separately for communication with the job scheduler and communication with a parallel pool.

Ephemeral TCP Ports with Job Manager

If you use the job manager on a cluster of nodes running Windows operating systems, you must make sure that a large number of ephemeral TCP ports are available on the job manager machine. By default, the maximum valid ephemeral TCP port number on a Windows operating system is 5000, but transfers of large data sets might fail if this setting is not increased. In particular, if your cluster has 32 or more workers, you should increase the maximum valid ephemeral TCP port number using the following procedure:

1. Start the Registry Editor.
2. Locate the following subkey in the registry, and click **Parameters**:

   ```
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
   ```

3 On the Registry Editor window, select **Edit** > **New** > **DWORD Value**.

4 In the list of entries on the right, change the new value name to `MaxUserPort` and press **Enter**.

5 Right-click on the `MaxUserPort` entry name and select **Modify**.

6 In the Edit DWORD Value dialog, enter `65534` in the **Value data** field. Select `Decimal` for the **Base** value. Click **OK**.

   This parameter controls the maximum port number that is used when a program requests any available user port from the system. Typically, ephemeral (short-lived) ports are allocated between the values of 1024 and 5000 inclusive. This action allows allocation for port numbers up to 65534.

7 Quit the Registry Editor.

8 Reboot your machine.

Host Communications Problems

If a worker is not able to make a connection with its MATLAB Job Scheduler, or if a client session cannot validate a profile that uses that scheduler, this might indicate communications problems between nodes.

With Command-Line Interface

First, be sure that the machines in question agree on their IP resolutions. The IP address for a particular host should be the same for itself as it is from the perspective of another host. For example, if a process on `hostB` cannot connect to one on `hostA`, find out the `hostA` IP address for itself, then see what the IP address for `hostA` is from `hostB`. They should be the same.

If the machines can identify each other, the `nodestatus` command can be useful for diagnosing problems between their processes. Use the function to determine what MATLAB Parallel Server processes are running on the local host, and which are accessible from remote hosts. If a worker on `hostA` cannot register with its job manager on `hostB`, run `nodestatus` on both hosts to see what each can see on `hostB`.

On `hostB`, execute:

```
nodestatus -remotehost hostB
```

Then on `hostA`, run exactly the same command:

```
nodestatus -remotehost hostB
```

The results should be the same, showing the same listing of job managers and workers.

If the output indicates problems, run the command again with a higher information level to receive more detailed information:

```
nodestatus -remotehost hostB -infolevel 3
```

With Admin Center GUI

You can diagnose some communications problems using Admin Center.

If you cannot successfully add hosts to the listing by specifying host name, you can use their IP addresses instead (see “Add Hosts” on page 4-3). If you suspect any communications problems, in the Admin Center GUI click **Test Connectivity** (see “Test Connectivity” on page 4-9). This testing verifies that the nodes can identify each other and allow their processes to communicate with each other.

Verify Network Communications for Cluster Discovery

If you want to use the discover cluster capabilities in Parallel Computing Toolbox, your network must be configured to use DNS SRV or DNS TXT records.

DNS SRV Record

When you use DNS for MATLAB Job Scheduler cluster discovery, you require a DNS SRV record for each domain. You can have multiple DNS SRV records for multiple MATLAB Job Schedulers. Use the following general form for each DNS SRV record.

```
_mdcs._tcp.<domain> <TTL> IN SRV <priority> <weight> <port> <hostname>.
```

Construct a DNS SRV record for a MATLAB Job Scheduler server using the following parts.

- `<domain>` is the domain name (like `company.com` or `university.edu`) that the client machine searches.
- `<TTL>` indicates how long (in seconds) the DNS record can be cached. `3600` is recommended.
- `IN SRV` is required as shown, indicating that this is a service record.
- `<priority>` and `<weight>` indicate priority and weight values. If you create multiple DNS SRV records, you can specify their priority with these fields. A value of `0` is recommended for each. The lower `<priority>` is, the *higher* priority the host has. When two records have the same `<priority>`, the record with the highest `<weight>` is used first. Use the `<weight>` value to specify server preference.
- `<port>` is the port on which you connect to the MATLAB Job Scheduler server. The default port is `27350`. If you change port for the MATLAB Job Scheduler server, change `<port>` accordingly.
- `<hostname>` is the fully qualified domain name for the host serving the MATLAB Job Scheduler. The machine `mjs-1` on the domain `company.com` has a fully qualified domain name `mjs-1.company.com`.

A valid DNS SRV record for the `company.com` network running a MATLAB Job Scheduler on machine `mjs-1` might look like this:

```
_mdcs._tcp.company.com 3600 IN SRV 0 0 27350 mjs-1.company.com.
```

Note If multiple domains are required to locate the cluster, use a DNS SRV record for each domain. If the network accessed by users via VPN has different DNS SRV records to your internal network, ensure that a DNS SRV record exists for each domain.

Use the standard procedure for your DNS system to create appropriate DNS SRV records. You can use standard utilities such as the `nslookup` command to verify that your network is configured with the necessary DNS SRV records. To examine MATLAB Job Scheduler DNS SRV records for the `company.com` domain, use the following command.

```
nslookup -type=SRV _mdcs._tcp.company.com
```

DNS TXT Record

Use DNS TXT records for third-party scheduler cluster discovery. A DNS TXT record associates a text string with a particular domain. To let MATLAB know where to find cluster discovery configuration files, store the locations of cluster discovery configuration files as text strings in DNS TXT records.

You can have multiple DNS TXT records for multiple clusters. Use this general form for each DNS TXT record.

```
_mdcs._tcp.<domain> IN TXT "discover_folder=<folder>"
```

Construct a DNS TXT record to discover a third-party scheduler using these parts.

- `<domain>` is the domain name (like `company.com` or `university.edu`) that the client machine searches.
- `IN TXT` is required as shown, indicating that this is a text record.
- `"discover_folder=<folder>"` where <folder> is the location of your cluster discovery configuration files.

A valid DNS TXT record for the `company.com` network running a Slurm scheduler cluster with a cluster discovery configuration file stored in `/network/share/discovery` might look like this:

```
_mdcs._tcp.company.com IN TXT "discover_folder=/network/share/discovery"
```

Note If multiple domains are required to locate the cluster, use a DNS TXT record for each domain. If the network accessed by users via VPN has different DNS TXT records to your internal network, ensure that a DNS TXT record exists for each domain.

Use the standard procedure for your DNS system to create appropriate DNS TXT records. You can use standard utilities such as the `nslookup` command to verify that your network is configured with the necessary DNS TXT records. To examine DNS TXT records for the `company.com` domain, use the following command.

```
nslookup -type=TXT _mdcs._tcp.company.com
```

See Also

Related Examples

- "Configure for Third-Party Scheduler Cluster Discovery" on page 3-68

Configure LDAP Server Authentication for MATLAB Job Scheduler

Configure MATLAB Job Scheduler to use your company Lightweight Directory Access Protocol (LDAP) server to authenticate user credentials. Follow these instructions to configure LDAP server authentication when you integrate MATLAB Job Scheduler with your cluster.

Prerequisites

If this is the first time you are integrating MATLAB Job Scheduler with your cluster, see this page for the most common configuration options: “Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager” on page 3-16.

In these instructions, `matlabroot` refers to the location of your installed MATLAB Parallel Server software. Where you see this term used in these instructions, substitute the path with the location of your installation.

Edit MATLAB Job Scheduler Parameter File

To configure LDAP server authentication, you must edit the `mjs_def` file on your headnode before installing the `mjs` service and starting MATLAB Job Scheduler. You can find this file in these locations:

- `matlabroot\toolbox\parallel\bin\mjs_def.bat` on Windows operating systems
- `matlabroot/toolbox/parallel/bin/mjs_def.sh` on Linux operating systems

To learn more about the parameters in the `mjs_def` file, see “Define MATLAB Job Scheduler Startup Parameters” on page 2-9.

Use these parameters to configure your company's LDAP server with your MATLAB Job Scheduler cluster. Edit the parameters in the `mjs_def` file with the required values.

Parameter	Description	Values
`SECURITY_LEVEL`	Security level for the cluster. To learn more about security levels and other parameters, see “Set MATLAB Job Scheduler Cluster Security” on page 2-29.	`Level 2` or `Level 3`

Parameter	Description	Values
`ADMIN_USER`	Username of the cluster administrator. `ADMIN_USER` must be a valid username in the LDAP server. **Note** When you start the job manager, the `mjs` service must authenticate the cluster administrator against the LDAP server to proceed. You must provide the LDAP server password of the cluster administrator when the mjs service prompts you.	Valid username in the LDAP server
`USE_LDAP_SERVER_AUTHENTICATION`	Option to use an LDAP server to authenticate user credentials.	`true`

`LDAP_URL`	URL of the LDAP server.
	
	Note Security Considerations: Use LDAP over SSL (LDAPS) to encrypt communication between the LDAP server and clients. For additional LDAPS configuration steps, see "Configure LDAP over SSL (LDAPS)" on page 2-45.
		Specify the LDAP_URL as:
		
		`ldaps://HOST:PORT`
		
		If you have not configured your LDAP server over SSL, specify the URL as:
		
		`ldap://HOST:PORT`

`LDAP_SECURITY_PRINCIPAL_FORMAT`	Format of a security principal (user) for your LDAP server.	Common formats include: • `cn=[username],ou=Users,dc=domain,dc=com` • `[username]@domain.com`
`LDAP_SYNCHRONIZATION_INTERVAL_SECS`	Frequency at which the cluster synchronizes with the LDAP server.	Positive number corresponding to the number of seconds between synchronizations. Default value is 1800 seconds. To synchronize the cluster with the LDAP server every time the cluster requires user authentication, set this parameter to `0`.

Configure LDAP over SSL (LDAPS)

When you use an LDAP server configured over SSL, you must add the LDAPS SSL certificate to the Java certificate trust store of your MATLAB Parallel Server installation. The `mjs` service validates the certificate against the LDAPS server to establish an encrypted connection.

The LDAPS SSL certificate must be formatted using PEM. For details about PEM, see:

- PEM format requirements for certificates and domain keys
- PEM, DER, CRT, and CER: X.509 Encodings and Conversions

These instructions show how to get an SSL certificate and add it to the Java certificate trust store.

Connect to LDAP Server to Get Server SSL Certificate

You can use the `openssl` toolkit with the `s_client` command to get the LDAP server SSL diagnostic information.

For example, to get the SSL diagnostic information from the LDAP server `my.LDAP.Server.com` at port 636, run this command in a Linux or Windows command window:

```
echo | openssl s_client -connect my.LDAP.Server.com:636 > myLDAPServer.com.cert.pem
```

The command generates the `myLDAPServer.com.cert.pem` file, which contains the LDAP server SSL diagnostic information. Edit the `myLDAPServer.com.cert.pem` file so that it contains only this text:

```
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
```

Add Certificate to Java Certificate Trust Store

The default Java certificate trust store is in these folders:

- `matlabroot\sys\java\jre\win64\jre\lib\security\cacerts` on Windows operating systems
- `matlabroot/sys/java/jre/glnxa64/jre/lib/security/cacerts` on Linux operating systems

To add the SSL certificate to the Java certificate trust store of your MATLAB Parallel Server installation, use the `keytool` key and certificate management utility. The `keytool` utility is available with your MATLAB Parallel Server installation at these locations:

- `matlabroot\sys\java\jre\win64\jre\bin` on Windows operating systems
- `matlabroot/sys/java/jre/glnxa64/jre/bin` on Linux operating systems

For more information, see keytool.

Before you import the server certificate to the Java certificate trust store, you must first make the `cacerts` file writable. For example, on a Linux host, run these commands:

```
cd matlabroot/sys/java/jre/glnxa64/jre/lib/security
chmod +w cacerts
```

Import the server certificate to the Java certificate trust store of the MATLAB Parallel Server installation. The default password for the `keytool` that comes with your MATLAB installation is `changeit`. You must enter the password when the `keytool` prompts you for a password.

```
matlabroot/sys/java/jre/glnxa64/jre/bin/keytool -import -keystore cacerts -file /path/to/server/certificate/myLDAPServer.com.cert.pem
```

Start MATLAB Job Scheduler and Test LDAP Server Authentication

To start MATLAB Job Scheduler, see Start `mjs` Service, MATLAB Job Scheduler, and Workers (Command-Line) on page 3-32.

You can connect to MATLAB Job Scheduler cluster and validate the cluster profile. You need your LDAP login username and password to access the cluster. For instructions that show how to validate your new MATLAB Job Scheduler cluster, see “Connect MATLAB Client to MATLAB Parallel Server Cluster” on page 3-20.

See Also

Related Examples

- “Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager” on page 3-16
- “Start mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)” on page 3-32
- “Define MATLAB Job Scheduler Startup Parameters” on page 2-9
- “Set MATLAB Job Scheduler Cluster Security” on page 2-29

Product Installation

Choose Solution to Install and Configure MATLAB Parallel Server

Consult the following tables for help with the most common tasks while setting up MATLAB Parallel Server. The tables help you to choose when you want to:

In this section...

"Activate the License for, Install, and Configure MATLAB Parallel Server on the Cluster" on page 3-2

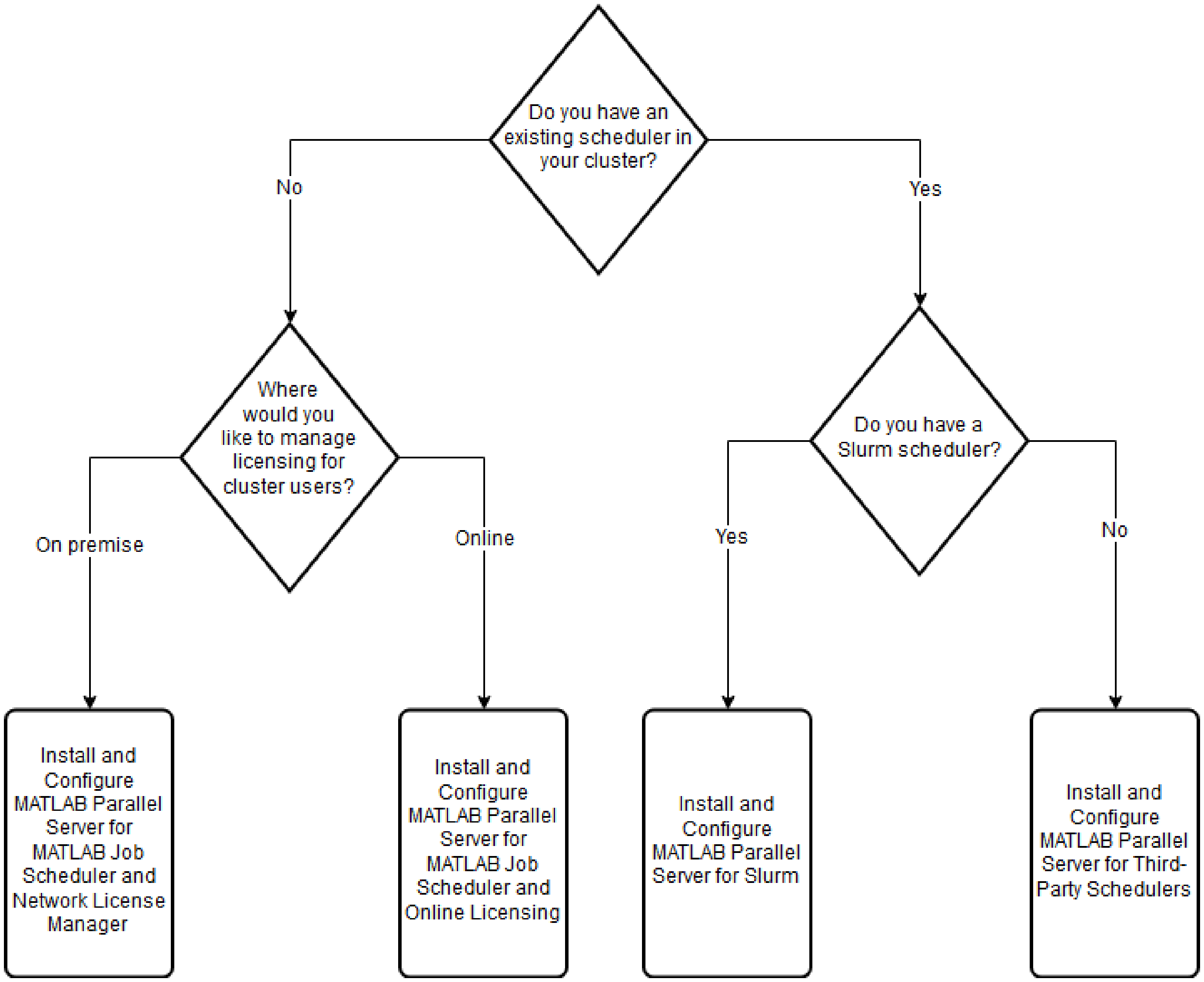
"Configure a MATLAB Client for a User to Submit Jobs to the Cluster" on page 3-3

"Upgrade MATLAB Parallel Server" on page 3-4

Activate the License for, Install, and Configure MATLAB Parallel Server on the Cluster

Consult the following table if you are getting started and you want to install and configure MATLAB Parallel Server for your cluster.

Scenario		Solution
You have a Slurm scheduler in your cluster.		Follow "Install and Configure MATLAB Parallel Server for Slurm" on page 3-6.
You have an existing scheduler in your cluster, such as • LSF • PBS • Torque • Grid Engine • HPC Pack • Hadoop®		Follow "Install and Configure MATLAB Parallel Server for Third-Party Schedulers" on page 3-12.
You do not have an existing scheduler in your cluster.	You want to manage licensing of your cluster users on-premise.	Set up the MATLAB Job Scheduler, which comes with MATLAB Parallel Server. Follow "Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager" on page 3-16.
	You want to use MathWorks online licensing for your cluster users.	Set up the MATLAB Job Scheduler, which comes with MATLAB Parallel Server. Follow "Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Online Licensing" on page 3-23.


Do you have an existing scheduler in your cluster?
No
Yes
Where would you like to manage licensing for cluster users?
On premise
Online
Do you have a Slurm scheduler?
Yes
No
Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager
Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Online Licensing
Install and Configure MATLAB Parallel Server for Slurm
Install and Configure MATLAB Parallel Server for Third-Party Schedulers

Configure a MATLAB Client for a User to Submit Jobs to the Cluster

Consult the following table if you already have a cluster configured and you want to configure a new MATLAB client to submit jobs to the cluster.

Scenario	Solution
The cluster uses a third-party scheduler and you meet all of these conditions: • The scheduler has out-of-the-box support in MATLAB. The schedulers with out-of-the-box support are Slurm, LSF, PBS Pro, OpenPBS, Grid Engine, Torque, HPC Pack. • There is a shared file system between the client machine and the cluster nodes. • The client machine has the third-party scheduler submission tools installed. You can check this by executing the relevant commands in a command line. For example, the Slurm submission script is `sbatch`.	See "Install Software on Local Desktop" on page 3-14. Then, • Slurm, LSF, PBS Pro, OpenPBS, Grid Engine, Torque – If the client is a Windows machine, start from "Configure Windows Firewalls on Server" on page 3-44. Otherwise, start from "Create Cluster Profile and Validate Installation" on page 3-41. • HPC Server – Start from "Configure Client Computer for HPC Pack" on page 3-37.
The cluster is a Hadoop cluster.	Start from "Client Configuration" on page 3-64.
The cluster uses a third-party scheduler.	Follow "Configure Using the Generic Scheduler Interface" on page 3-45.
The cluster uses the MATLAB Job Scheduler.	• If the cluster is using MathWorks online licensing, ask your cluster administrator to add the new user. For more information, see "Add Licensed End Users" on page 3-23. Then, follow "Connect the MATLAB Client to the MATLAB Parallel Server Cluster" on page 3-27. • Otherwise, follow "Connect MATLAB Client to MATLAB Parallel Server Cluster" on page 3-20.

Note You can export a cluster profile and import it in another client machine. For more information, see "Import and Export Cluster Profiles" (Parallel Computing Toolbox). This is useful when you are configuring another client machine with the same configuration, to avoid recreating the cluster profile manually.

Upgrade MATLAB Parallel Server

Consult the following table if you want to configure a new release of MATLAB Parallel Server.

Scenario	Solution
The cluster uses a third-party scheduler.	Follow "Install and Configure MATLAB Parallel Server for Third-Party Schedulers" on page 3-12 to • Activate and install the new version of MATLAB Parallel Server on the cluster. • Configure the cluster and client machines.

Scenario	Solution
The cluster uses the MATLAB Job Scheduler.	**1** Stop the mjs service of the older release. For instructions, see “Stop mjs Services of Old Installation” on page 3-31. **2** Install the new version of MATLAB Parallel Server on the cluster and configure the MATLAB Job Scheduler. For more information, see “Run Multiple MATLAB Parallel Server Versions” on page 3-29. For help with activation, installation, and configuration, consult the table in “Activate the License for, Install, and Configure MATLAB Parallel Server on the Cluster” on page 3-2.

Install and Configure MATLAB Parallel Server for Slurm

If you have a cluster with Slurm, follow these instructions to integrate MATLAB with your scheduler using MATLAB Parallel Server. If you do not have an existing scheduler in your cluster, see: “Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager” on page 3-16.

These instructions guide you through the following tasks:

In this section...
“Activate Your MATLAB Parallel Server License” on page 3-6
“Get the Installation Files” on page 3-6
“Install License Manager” on page 3-7
“Install Software on Compute Nodes” on page 3-8
“Install Software on Local Desktop” on page 3-8
“Configure Client Machine” on page 3-9
“Validate the Cluster Profile” on page 3-10
“Run Parallel Code” on page 3-11

After you integrate MATLAB with Slurm, you can access workers in your cluster from a desktop MATLAB client session with Parallel Computing Toolbox. Workers are MATLAB computational engines that typically correspond to a core.

The setup in these steps uses the network license manager.

Activate Your MATLAB Parallel Server License

To install MATLAB Parallel Server, you must activate your license. To activate your MATLAB Parallel Server license:

1. Navigate to https://www.mathworks.com/licensecenter.
2. Log into the Administrator’s MathWorks Account.
3. Select your MATLAB Parallel Server license, and click the **Install and Activate** tab.
4. At the rightmost side, under **RELATED TASKS**, select **Activate to Retrieve License File**.
5. Fill in the requested information. This information must refer to the machine that hosts the license manager. In these instructions, it is the head node. For more information, see “Install License Manager” on page 3-7.
6. After filling in the information, download or email the License File and copy the File Installation Key. These are used later in the process.

Note Activation is not necessary for trials. Contact your sales representative to obtain the License File and the File Installation Key.

Get the Installation Files

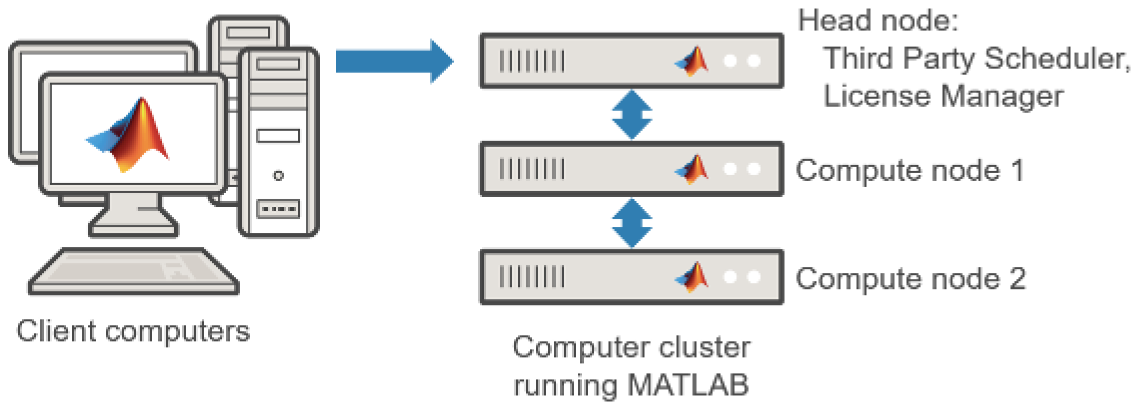
To save time and eliminate the need for the installer-based download process for each computer in your cluster, download the installation files prior to installation. Doing so facilitates installation in a

large number of machines. If you have access to an Administrator's account for your license, you can use the installer to download files without installing them. If not, contact the administrator of your license to obtain a copy of the installation files. For more information, see "Download Products Without Installing". When using the installer to download the files, choose the following options:

- Select the appropriate operating system for the cluster nodes.
- Select all products for download. MATLAB Parallel Server cannot run jobs requiring products that are not installed.

There are three server-side components of MATLAB Parallel Server:

1. The license manager, which hosts the MATLAB Parallel Server license used by each worker. For more information, see "Install License Manager" on page 3-7.
2. Your third-party job scheduler, which runs on the head node and manages jobs on your cluster. You integrate this scheduler with MATLAB Parallel Server. For more information, see "Install License Manager" on page 3-7 and "Configure Your Cluster" on page 3-14.
3. MATLAB Parallel Server, which runs on the compute nodes. For more information, see "Install Software on Compute Nodes" on page 3-8.



Install License Manager

Choose a computer node to host the license manager. For the installation, use the offline installer from the previous step. For more information on the offline installation procedure, see "Install Products Using File Installation Key".

1. Start the MATLAB installer from the installation files acquired in "Get the Installation Files" on page 3-6.
2. Select **Advanced Options > I want to install network license manager**.
3. In the License File step, browse to your `license.lic` file (obtained from "Activate Your MATLAB Parallel Server License" on page 3-6).
4. Continue through the prompts to complete the network manager installation. For more information about the procedure, see "Install License Manager on License Server"
5. Start the license manager.

Install Software on Compute Nodes

This procedure is similar to "Install License Manager" on page 3-7. For more information on the offline installation procedure, see "Install Products Using File Installation Key".

1. Start the MATLAB installer from the installation files acquired in "Get the Installation Files" on page 3-6.
2. Select **Advanced Options > I have a File Installation Key** and continue through the prompts.
3. Select all products. Alternatively, to save space, install only the products that the users of the cluster are licensed for.
4. Use the `license.dat` file from the head node. You can obtain this file from the `matlabroot/etc` folder, where `matlabroot` is the MATLAB installation folder.

For best performance, install locally on each node. However, you can also install in a network share location.

Note Install noninteractively (silently) instead if you want to

- Install the software on a machine without graphical user interface (GUI).
- Save the time that it takes to configure the installer for each compute node.

For more information, see "Install Noninteractively".

You can test the installation and licensing by running the following command in a command-line interface. *`matlabroot`* is the MATLAB installation folder. *`filename`* is the location to write the log file to, for example, a temporary location. You must have write permissions in this location.

```
matlabroot/toolbox/parallel/bin/checkLicensing -logfile filename
```

Note You do not need to start the mjs service when you want to configure MATLAB Parallel Server for Slurm.

Install Software on Local Desktop

To use MATLAB Parallel Server, you must use a local desktop running MATLAB and Parallel Computing Toolbox. Install the MathWorks products for which you are licensed, including Parallel Computing Toolbox, on the local desktops from which you want to submit jobs to the cluster. For help with this step, see "Installation and Licensing".

Any MATLAB cluster workers that you start use dynamic licensing: they can use all the functionality you are licensed for in the MATLAB client, while checking out only MATLAB Parallel Server licenses in the cluster.

Configure Client Machine

Next step	More information
Use out-of-the-box support If you meet all of these conditions. • The client machine uses Linux. • There is a shared file system between the client machine and the cluster nodes. • The client machine has the third-party scheduler submission tools installed. You can check this by executing the relevant commands in a command line. For example, the submission script is `sbatch`.	Follow "Create a Cluster Profile Using Out-of-the-Box Support" on page 3-9.
Use the generic scheduler interface.	Follow "Create a Cluster Profile Using the Generic Scheduler Interface" on page 3-10.

Create a Cluster Profile Using Out-of-the-Box Support

1. Start the Cluster Profile Manager. On the **Home** tab, in the **Environment** area, select **Parallel > Create and Manage Clusters**.
2. Create a new Slurm profile. In the Cluster Profile Manager, select **Add Cluster Profile > Slurm**.
3. To give it a different name, select the new profile and click **Rename**.
4. In the **Properties** tab, provide settings for the following fields.

   a. (Optional) Set the **Description** field.

   b. Set the **JobStorageLocation** to the location where you want job and task data to be stored. This location must be accessible to all the worker machines.

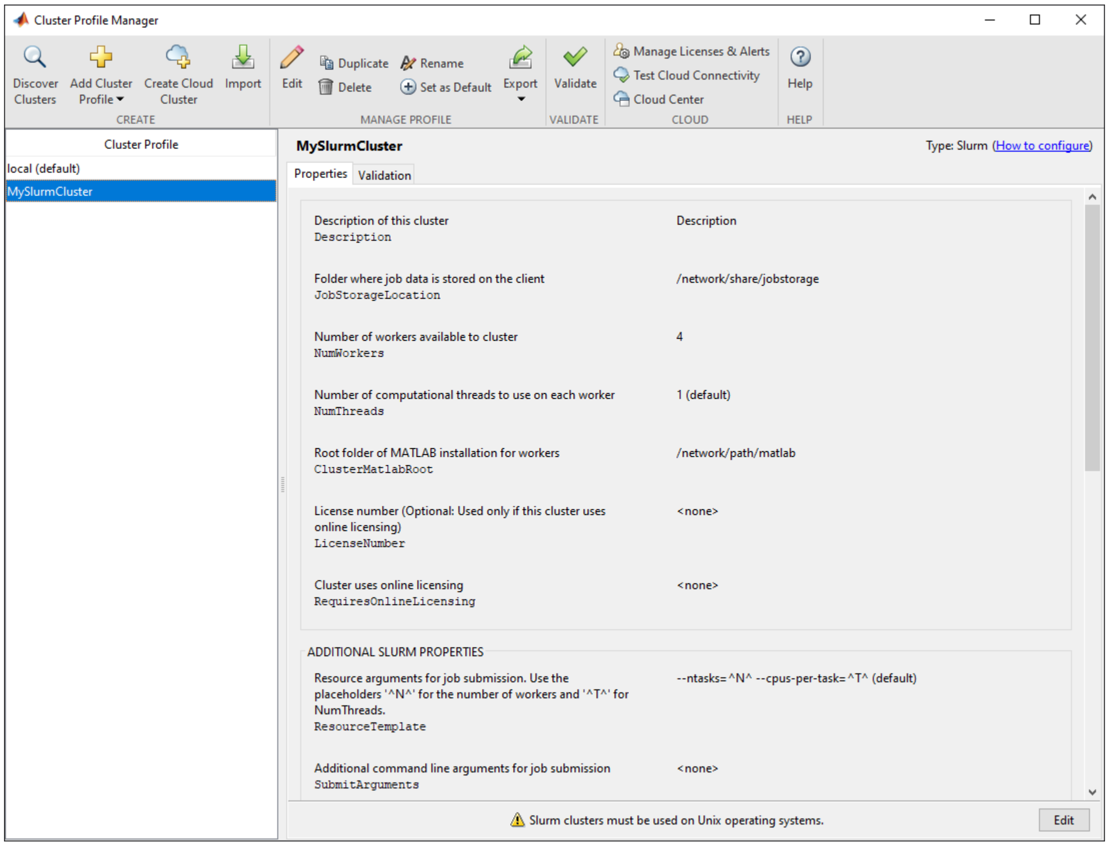
   Note Sharing `JobStorageLocation` with parallel computing products running different versions is not recommended; each version on your cluster should have its own `JobStorageLocation`.

   c. Set the **NumWorkers** field to the number of workers you want to use with this profile, within the limitation of your licensing.

   d. Set the **ClusterMatlabRoot** to the installation location of the MATLAB to be executed by the worker machines.

   e. (Optional) Set the **SubmitArguments** to include any additional command arguments required by your particular cluster and scheduler.

   f. After filling the fields, the dialog box looks like this:



MySlurmCluster

Type: Slurm (How to configure)

Description of this cluster `Description`	Description
Folder where job data is stored on the client `JobStorageLocation`	/network/share/jobstorage
Number of workers available to cluster `NumWorkers`	4
Number of computational threads to use on each worker `NumThreads`	1 (default)
Root folder of MATLAB installation for workers `ClusterMatlabRoot`	/network/path/matlab
License number (Optional: Used only if this cluster uses online licensing) `LicenseNumber`	<none>
Cluster uses online licensing `RequiresOnlineLicensing`	<none>

ADDITIONAL SLURM PROPERTIES

Resource arguments for job submission. Use the placeholders '^N^' for the number of workers and '^T^' for NumThreads. `ResourceTemplate`	--ntasks=^N^ --cpus-per-task=^T^ (default)
Additional command line arguments for job submission `SubmitArguments`	<none>

Slurm clusters must be used on Unix operating systems.

5 Click **Done** to save your cluster profile.

Create a Cluster Profile Using the Generic Scheduler Interface

Download the Slurm plugin scripts to interface with Slurm from the MATLAB client. On the **Home** tab, in the **Environment** area, click **Add-Ons**. In the Add-On Explorer, search for the add-on ***Parallel Computing Toolbox plugin for MATLAB Parallel Server with Slurm*** and install it. Alternatively, you can download the add-on from here: Slurm. To open it, double click it or drag it and drop it in MATLAB.

For more information on the generic scheduler interface, see “Configure Using the Generic Scheduler Interface” on page 3-45.

Validate the Cluster Profile

In this step you verify your cluster profile, and thereby your installation. You can specify the number of workers to use when validating your profile. If you do not specify the number of workers in the **Validation** tab, then the validation will attempt to use as many workers as the value specified by the `NumWorkers` property on the **Properties** tab. You can specify a smaller number of workers to validate your configuration without occupying the whole cluster.

1 If it is not already open, start the Cluster Profile Manager. On the **Home** tab, in the **Environment** area, select **Parallel** > **Create and Manage Clusters**.

2 Select your cluster profile in the listing.

3 Click the **Validation** tab.

4. Use the checkboxes to choose all tests, or a subset of the validation stages, and specify the number of workers to use when validating your profile.
5. Click **Validate**.

The Validation Results tab shows the output. The following figure shows the results of a profile that passed all validation tests.

Note If your validation does not pass, contact the MathWorks install support team.

If your validation passed, you now have a valid profile that you can use in other parallel applications. You can make any modifications to your profile appropriate for your applications, such as `NumWorkersRange`, `AttachedFiles`, `AdditionalPaths`, etc.

To save your profile for other users, select the profile and click **Export**, then save your profile to a file in a convenient location. Later, when running the Cluster Profile Manager, other users can import your profile by clicking **Import**.

Run Parallel Code

After you complete the configuration, you can submit jobs to the cluster. For examples, see “Running Code on Clusters and Clouds”.

See Also

“Running Code on Clusters and Clouds”

Install and Configure MATLAB Parallel Server for Third-Party Schedulers

If you already have a cluster with a scheduler, follow these instructions to integrate MATLAB with your scheduler using MATLAB Parallel Server. If you do not have an existing scheduler in your cluster, see: “Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager” on page 3-16.

These instructions guide you through the following tasks:

In this section...

“Activate Your MATLAB Parallel Server License” on page 3-12
“Get the Installation Files” on page 3-12
“Install License Manager” on page 3-13
“Install Software on Compute Nodes” on page 3-14
“Install Software on Local Desktop” on page 3-14
“Configure Your Cluster” on page 3-14

After you integrate MATLAB with a scheduler, you can access workers in your cluster from a desktop MATLAB client session with Parallel Computing Toolbox. Workers are MATLAB computational engines that typically correspond to a core.

The setup in these steps uses the network license manager.

Activate Your MATLAB Parallel Server License

To install MATLAB Parallel Server, you must activate your license. To activate your MATLAB Parallel Server license:

1. Navigate to https://www.mathworks.com/licensecenter.
2. Log into the Administrator’s MathWorks Account.
3. Select your MATLAB Parallel Server license, and click the **Install and Activate** tab.
4. At the rightmost side, under **RELATED TASKS**, select **Activate to Retrieve License File**.
5. Fill in the requested information. This information must refer to the machine that hosts the license manager. In these instructions, it is the head node. For more information, see “Install License Manager” on page 3-13.
6. After filling in the information, download or email the License File and copy the File Installation Key. These are used later in the process.

Note Activation is not necessary for trials. Contact your sales representative to obtain the License File and the File Installation Key.

Get the Installation Files

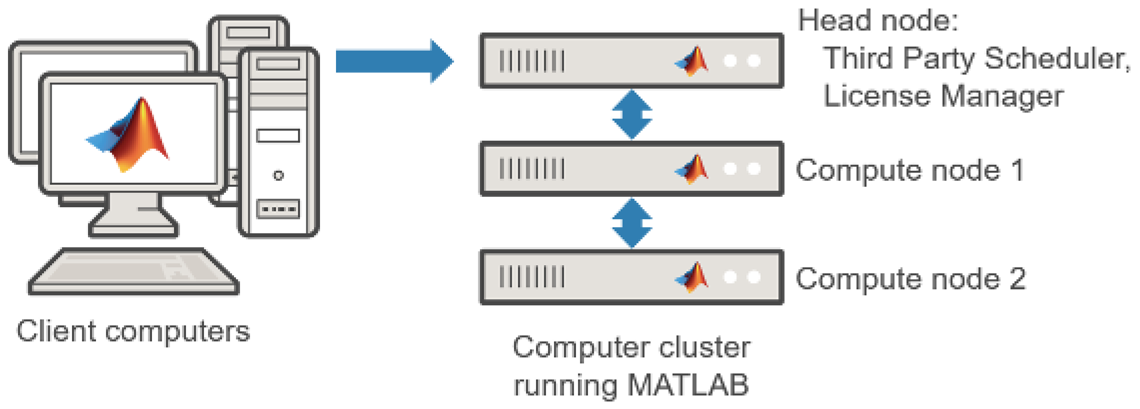
To save time and eliminate the need for the installer-based download process for each computer in your cluster, download the installation files prior to installation. Doing so facilitates installation in a

large number of machines. If you have access to an Administrator’s account for your license, you can use the installer to download files without installing them. If not, contact the administrator of your license to obtain a copy of the installation files. For more information, see “Download Products Without Installing”. When using the installer to download the files, choose the following options:

- Select the appropriate operating system for the cluster nodes.
- Select all products for download. MATLAB Parallel Server cannot run jobs requiring products that are not installed.

There are three server-side components of MATLAB Parallel Server:

1. The license manager, which hosts the MATLAB Parallel Server license used by each worker. For more information, see “Install License Manager” on page 3-13.
2. Your third-party job scheduler, which runs on the head node and manages jobs on your cluster. You integrate this scheduler with MATLAB Parallel Server. For more information, see “Install License Manager” on page 3-13 and “Configure Your Cluster” on page 3-14.
3. MATLAB Parallel Server, which runs on the compute nodes. For more information, see “Install Software on Compute Nodes” on page 3-14.



Install License Manager

Choose a computer node to host the license manager. For the installation, use the offline installer from the previous step. For more information on the offline installation procedure, see “Install Products Using File Installation Key”.

1. Start the MATLAB installer from the installation files acquired in “Get the Installation Files” on page 3-12.
2. Select **Advanced Options > I want to install network license manager**.
3. In the License File step, browse to your `license.lic` file (obtained from “Activate Your MATLAB Parallel Server License” on page 3-12).
4. Continue through the prompts to complete the network manager installation. For more information about the procedure, see “Install License Manager on License Server”
5. Start the license manager.

Install Software on Compute Nodes

This procedure is similar to "Install License Manager" on page 3-13. For more information on the offline installation procedure, see "Install Products Using File Installation Key".

1. Start the MATLAB installer from the installation files acquired in "Get the Installation Files" on page 3-12.
2. Select **Advanced Options > I have a File Installation Key** and continue through the prompts.
3. Select all products. Alternatively, to save space, install only the products that the users of the cluster are licensed for.
4. Use the `license.dat` file from the head node. You can obtain this file from the `matlabroot/etc` folder, where `matlabroot` is the MATLAB installation folder.

For best performance, install locally on each node. However, you can also install in a network share location.

Note Install noninteractively (silently) instead if you want to

- Install the software on a machine without graphical user interface (GUI).
- Save the time that it takes to configure the installer for each compute node.

For more information, see "Install Noninteractively".

You can test the installation and licensing by running the following command in a command-line interface. *`matlabroot`* is the MATLAB installation folder. *`filename`* is the location to write the log file to, for example, a temporary location. You must have write permissions in this location.

```
matlabroot/toolbox/parallel/bin/checkLicensing -logfile filename
```

Note You do not need to start the mjs service when you want to configure MATLAB Parallel Server for a third-party scheduler.

Install Software on Local Desktop

To use MATLAB Parallel Server, you must use a local desktop running MATLAB and Parallel Computing Toolbox. Install the MathWorks products for which you are licensed, including Parallel Computing Toolbox, on the local desktops from which you want to submit jobs to the cluster. For help with this step, see "Installation and Licensing".

Any MATLAB cluster workers that you start use dynamic licensing: they can use all the functionality you are licensed for in the MATLAB client, while checking out only MATLAB Parallel Server licenses in the cluster.

Configure Your Cluster

When the cluster and client installations are complete, you can proceed to configure the products for the job scheduler of your choice. Use this table to choose a suitable guide to complete your configuration.

Next step	**More information**
Use out-of-the-box support If • Your scheduler has out-of-the-box support in MATLAB. These schedulers are Slurm, PBS Pro, OpenPBS, Torque, LSF, and HPC Pack. • There is a shared file system between the client machine and the cluster nodes. • The client machine has the third-party scheduler submission tools installed. You can check this by executing the relevant commands in a command line. For example, the Slurm submission script is `sbatch`.	• “Configure for HPC Pack” on page 3-37 • “Configure for Slurm, PBS Pro, OpenPBS, LSF, TORQUE” on page 3-41
Use out-of-the-box support if you have a Hadoop cluster.	• “Configure a Hadoop Cluster” on page 3-64
Use the generic scheduler interface.	• “Configure Using the Generic Scheduler Interface” on page 3-45

See Also
“Running Code on Clusters and Clouds”

Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager

If you do not have an existing scheduler in your cluster, follow these instructions to integrate the MATLAB Job Scheduler, which is provided with MATLAB Parallel Server. If you already have a cluster with a scheduler, see “Install and Configure MATLAB Parallel Server for Third-Party Schedulers” on page 3-12.

These instructions guide you through the following tasks:

In this section...
“Activate Your MATLAB Parallel Server License” on page 3-16
“Get the Installation Files” on page 3-16
“Install Software on the Head Node” on page 3-17
“Install Software on Compute Nodes” on page 3-18
“Configure the MATLAB Job Scheduler” on page 3-18
“Connect MATLAB Client to MATLAB Parallel Server Cluster” on page 3-20

After you integrate MATLAB with a scheduler, you can access workers in your cluster from a desktop MATLAB client session with Parallel Computing Toolbox. Workers are MATLAB computational engines that typically correspond to a core.

The setup in these steps uses the network license manager.

Activate Your MATLAB Parallel Server License

To install MATLAB Parallel Server, you must activate your license. To activate your MATLAB Parallel Server license:

1. Navigate to https://www.mathworks.com/licensecenter.
2. Log into the Administrator’s MathWorks Account.
3. Select your MATLAB Parallel Server license, and click the **Install and Activate** tab.
4. At the rightmost side, under **RELATED TASKS**, select **Activate to Retrieve License File**.
5. Fill in the requested information. This information must refer to the machine that hosts the license manager. In these instructions, it is the head node. For more information, see “Install Software on the Head Node” on page 3-17.
6. After filling in the information, download or email the License File and copy the File Installation Key. These are used later in the process.

Note Activation is not necessary for trials. Contact your sales representative to obtain the License File and the File Installation Key.

Get the Installation Files

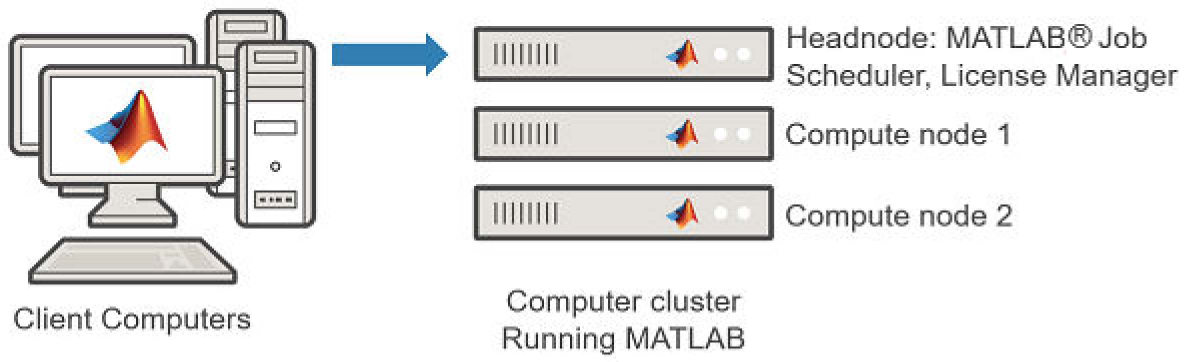
To save time and eliminate the need for the installer-based download process for each computer in your cluster, download the installation files prior to installation. Doing so facilitates installation in a

large number of machines. If you have access to an Administrator's account for your license, you can use the installer to download files without installing them. If not, contact the administrator of your license to obtain a copy of the installation files. For more information, see "Download Products Without Installing". When using the installer to download the files, choose the following options:

- Select the appropriate operating system for the cluster machines.
- Select all products for download. MATLAB Parallel Server cannot run jobs requiring products that are not installed.

There are three server-side components of MATLAB Parallel Server:

1. The license manager, which hosts the MATLAB Parallel Server license used by each worker. For more information, see "Install Software on the Head Node" on page 3-17.
2. The MATLAB Job Scheduler, which runs on the head node and manages jobs on your cluster. For more information, see "Install Software on the Head Node" on page 3-17.
3. MATLAB Parallel Server, which runs on the compute nodes. For more information, see "Install Software on Compute Nodes" on page 3-18.


Headnode: MATLAB® Job Scheduler, License Manager
Compute node 1
Compute node 2
Client Computers
Computer cluster Running MATLAB

Install Software on the Head Node

Use the offline installer from the previous step. For more information on the offline installation procedure, see "Install Products Using File Installation Key".

1. Choose a computer to host the license manager and the MATLAB Job Scheduler. This computer is your head node.
2. Start the MATLAB installer from the installation files acquired in "Get the Installation Files" on page 3-16.
3. Select **Advanced Options > I want to install network license manager**, and continue through the prompts to complete the network manager installation. In the License File step, browse to your `license.lic` file (obtained from "Activate Your MATLAB Parallel Server License" on page 3-16). For more information about the procedure, see "Install License Manager on License Server".
4. Restart the installer by clicking on `setup.exe`. This file is located in the top level of the folder where you extracted the files.
5. Under **Advanced Options** select **I have a File Installation Key** and continue through the prompts.
6. Select all products. Alternatively, to save space, install only the products that the users of the cluster are licensed for.

7. In the License File step, browse to your `license.dat` file. You can obtain this file from the `matlabroot/etc` folder, where `matlabroot` is the MATLAB installation folder.
8. Start the license manager.

You can test the installation and licensing by running the following command in a command-line interface. *`matlabroot`* is the MATLAB installation folder. *`filename`* is the location to write the log file to, for example, a temporary location. You must have write permissions in this location.

```
matlabroot/toolbox/parallel/bin/checkLicensing -logfile filename
```

Install Software on Compute Nodes

This procedure is similar to “Install Software on the Head Node” on page 3-17. For more information on the offline installation procedure, see “Install Products Using File Installation Key”.

1. Start the MATLAB installer from the installation files acquired in “Get the Installation Files” on page 3-16.
2. Under **Advanced Options** select **I have a File Installation Key** and continue through the prompts.
3. Select all products. Alternatively, to save space, install only the products that the users of the cluster are licensed for.
4. Use the `license.dat` file from the head node. You can obtain this file from the `matlabroot/etc` folder, where `matlabroot` is the MATLAB installation folder.

For best performance, install locally on each node. However, you can also install in a network share location.

Note Install noninteractively (silently) instead if you want to

- Install the software on a machine without graphical user interface (GUI).
- Save the time that it takes to configure the installer for each compute node.

For more information, see “Install Noninteractively”.

You can test the installation and licensing by running the following command in a command-line interface. *`matlabroot`* is the MATLAB installation folder.

```
matlabroot/toolbox/parallel/bin/checkLicensing
```

Configure the MATLAB Job Scheduler

The MATLAB Job Scheduler is a scheduler that ships with MATLAB Parallel Server. The MATLAB Job Scheduler is intended primarily for clusters that run only MATLAB jobs. The scheduler interface is a high-level abstraction that lets you submit jobs to your computation resources, so you do not have to deal with differences in operating systems and environments.

The following steps configure the MATLAB Job Scheduler with Admin Center, the graphical interface. If your machine does not provide graphics, use the command-line interface instead. For more information, see “Start mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)” on page 3-32.

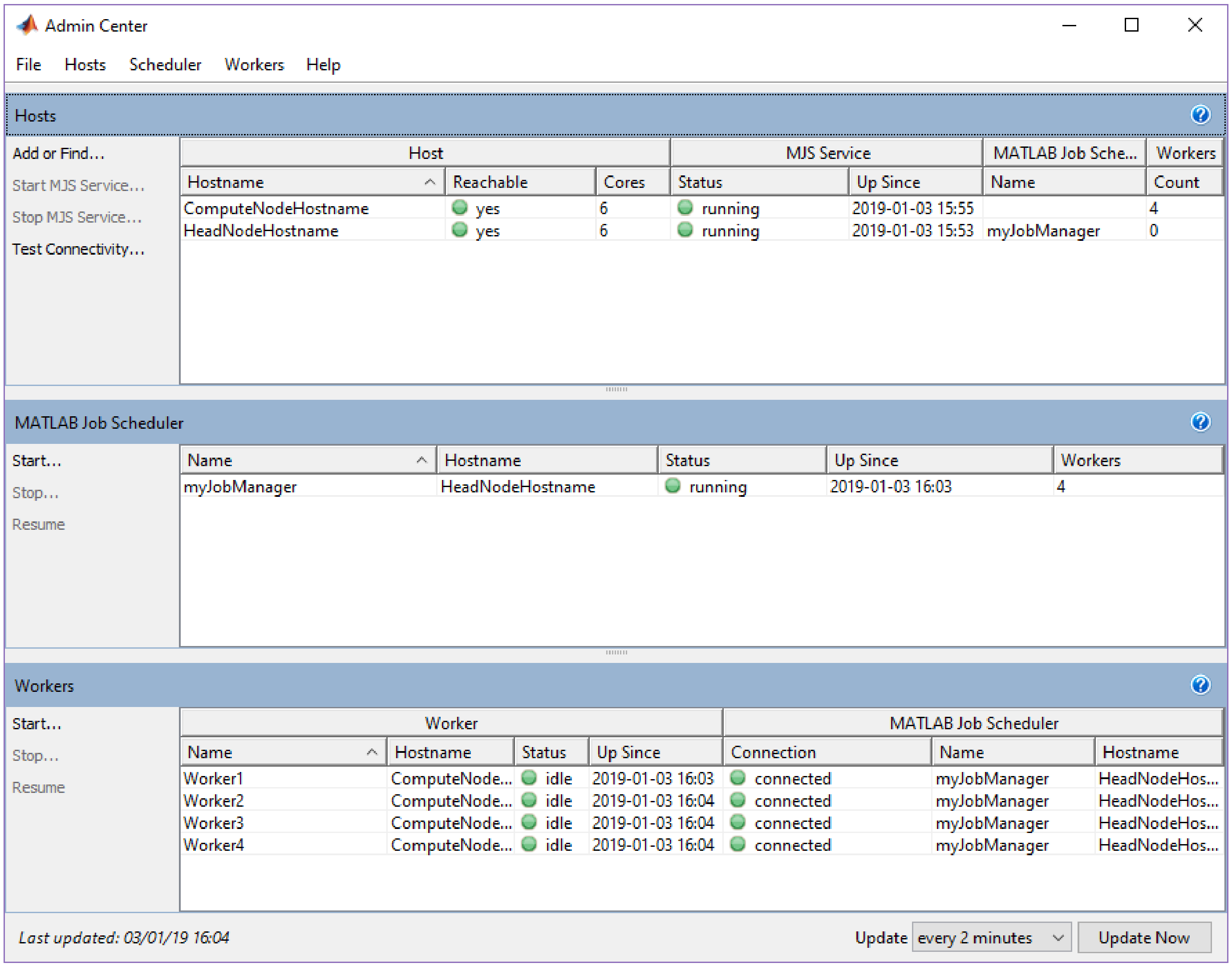
1. On the head node, start Admin Center. Browse to `matlabroot/toolbox/parallel/bin` and execute the file named `admincenter`, where `matlabroot` is the MATLAB installation folder.
2. Click **Add or Find**, and specify the computers that you are using as your head node and compute nodes.
3. Progress through the prompts, and confirm to start the mjs service. If necessary, manually start the mjs service using the command-line interface. For more information, see “Use the Command-Line Interface (Windows)” on page 3-32 or “Use the Command-Line Interface (UNIX)” on page 3-34.
4. In the MATLAB Job Scheduler section, click **Start**. Specify a name for your MATLAB Job Scheduler, and select the head node from the dropdown list.
5. To add the MATLAB Parallel Server workers, click **Start** in the **Workers** section of the Admin Center.

   a. Select the computers to host the workers.
   b. Select the number of workers per computer.
6. To verify your configuration, review worker status in the **Workers** section.
7. To troubleshoot issues, click **Test Connectivity** in the **Host** section.
8. If you are using UNIX, configure the mjs service to start automatically at start time. For instructions, see “Start mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)” on page 3-32.

The head node uses computational resources to run the MATLAB Job Scheduler. If you set up workers on the head node, they compete for resources with the MATLAB Job Scheduler.

Tip Avoid setting up workers on the head node. If workers use too many system resources, such as memory, processor, network, or local storage, the job manager can become unresponsive.

The following screenshot shows a final setup in Admin Center:



Admin Center

File Hosts Scheduler Workers Help

Hosts

Add or Find...
Start MJS Service...
Stop MJS Service...
Test Connectivity...

Host			MJS Service		MATLAB Job Sche...	Workers
Hostname	Reachable	Cores	Status	Up Since	Name	Count
ComputeNodeHostname	● yes	6	● running	2019-01-03 15:55		4
HeadNodeHostname	● yes	6	● running	2019-01-03 15:53	myJobManager	0

MATLAB Job Scheduler

Start...
Stop...
Resume

Name	Hostname	Status	Up Since	Workers
myJobManager	HeadNodeHostname	● running	2019-01-03 16:03	4

Workers

Start...
Stop...
Resume

Worker				MATLAB Job Scheduler		
Name	Hostname	Status	Up Since	Connection	Name	Hostname
Worker1	ComputeNode...	● idle	2019-01-03 16:03	● connected	myJobManager	HeadNodeHos...
Worker2	ComputeNode...	● idle	2019-01-03 16:04	● connected	myJobManager	HeadNodeHos...
Worker3	ComputeNode...	● idle	2019-01-03 16:04	● connected	myJobManager	HeadNodeHos...
Worker4	ComputeNode...	● idle	2019-01-03 16:04	● connected	myJobManager	HeadNodeHos...

Last updated: 03/01/19 16:04

Update every 2 minutes Update Now

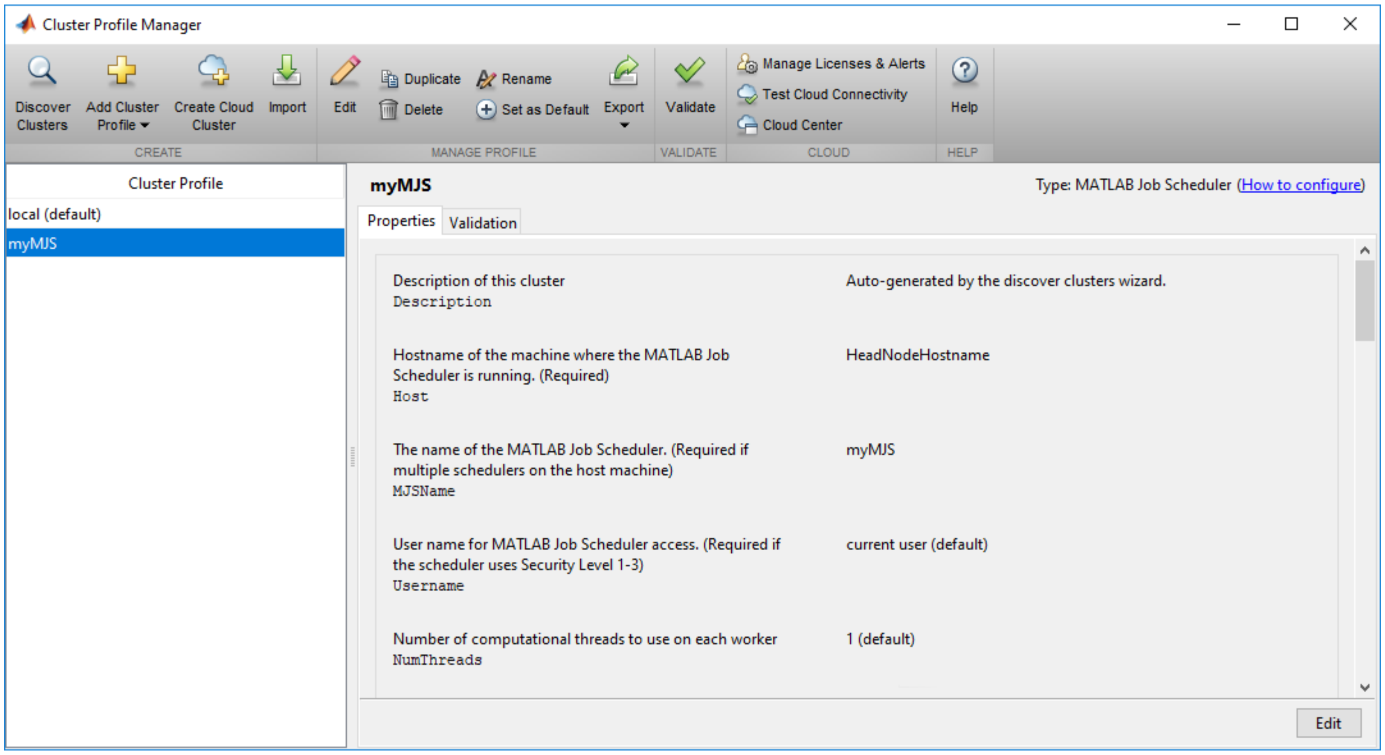
Note If you need more help during the configuration, such as your cluster requires firewall configuration or you want to set up multiple mjs installations, see this more detailed guide: "Configure Advanced Options for MATLAB Job Scheduler Integration" on page 3-29.

Connect MATLAB Client to MATLAB Parallel Server Cluster

To use MATLAB Parallel Server, you must use a client computer running MATLAB and Parallel Computing Toolbox. In the MATLAB toolstrip, use **Parallel** > **Discover Clusters** and follow the instructions to automatically discover and set up your cluster. Alternatively, you can configure it manually as follows:

1. In MATLAB, on the **Home** tab, select the **Parallel** menu. Select **Create and Manage Clusters**.
2. Click **Add Cluster Profile** > **MATLAB Job Scheduler**.
3. To modify the name of the MATLAB Job Scheduler profile, double click the profile name.
4. To edit the profile, select it and click **Edit** in the toolbar.

5. In the **Host** field, enter the host name of the head node.
6. Click **Done**. The following image shows a MATLAB Job Scheduler cluster profile after configuration:



myMJS

Type: MATLAB Job Scheduler (How to configure)

Description of this cluster `Description`	Auto-generated by the discover clusters wizard.
Hostname of the machine where the MATLAB Job Scheduler is running. (Required) `Host`	HeadNodeHostname
The name of the MATLAB Job Scheduler. (Required if multiple schedulers on the host machine) `MJSName`	myMJS
User name for MATLAB Job Scheduler access. (Required if the scheduler uses Security Level 1-3) `Username`	current user (default)
Number of computational threads to use on each worker `NumThreads`	1 (default)

7. To make this profile the default, select **Set as Default**.
8. **Validate** the cluster profile.

If validation of your cluster is successful, your MATLAB session can now submit jobs to the MATLAB Parallel Server cluster.

Note If your validation does not pass, contact the MathWorks install support team.

Any MATLAB cluster workers that you start use dynamic licensing: they can use all the functionality you are licensed for in the MATLAB client, while checking out only MATLAB Parallel Server licenses in the cluster.

To configure more advanced options for your cluster, see “MATLAB Job Scheduler Cluster Customization”. For example, you can set the security of the cluster in “Set MATLAB Job Scheduler Cluster Security” on page 2-29. After you finish your configuration, try some examples of cluster workflows in “Running Code on Clusters and Clouds”.

See Also

Related Examples

- “MATLAB Job Scheduler Cluster Customization”

- “Set MATLAB Job Scheduler Cluster Security” on page 2-29

Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Online Licensing

If you do not have an existing scheduler in your cluster, follow these instructions to integrate the MATLAB Job Scheduler, which is provided with MATLAB Parallel Server. If you already have a cluster with a scheduler, see “Install and Configure MATLAB Parallel Server for Third-Party Schedulers” on page 3-12.

These instructions guide you through the following tasks:

In this section...

“Check License Type and Users” on page 3-23
“Get the Installation Files” on page 3-24
“Install Software on All Nodes” on page 3-25
“Configure the MATLAB Job Scheduler with Admin Center” on page 3-25
“Connect the MATLAB Client to the MATLAB Parallel Server Cluster” on page 3-27

After you integrate MATLAB with a scheduler, you can access workers in your cluster from a desktop MATLAB client session (requires Parallel Computing Toolbox). Workers are MATLAB computational engines that typically correspond to a core.

The setup in these steps uses online licensing.

Check License Type and Users

Check License Type

Note If you have a campus-wide license for MATLAB Parallel Server, then you already have access to online licensing and can skip to “Add Licensed End Users” on page 3-23. If you are unsure about your license details, then contact the license administrator.

To install MATLAB Parallel Server using online licensing, you must check your license type.

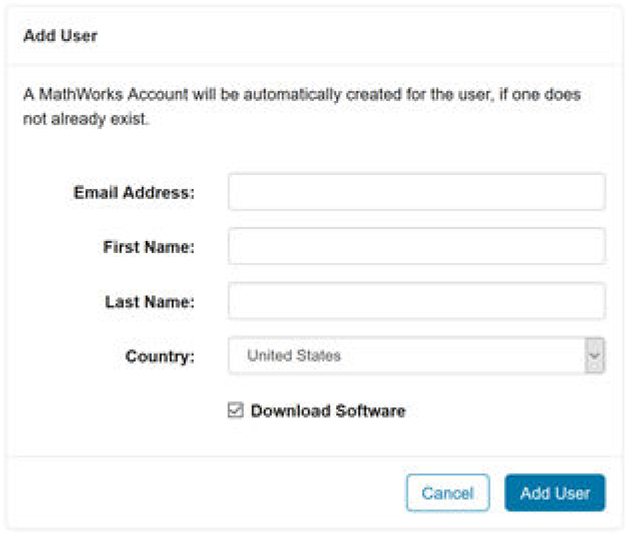
1. In your browser, go to License Center and log in with your MathWorks Administrator Account.
2. Select the MATLAB Parallel Server license that you plan to use.
3. On the **Install and Activate** tab, look for **License Manager:** followed by the license manager type currently assigned to this license.

   - If the license manager is already the one you want, then you do not need to do anything. Go to “Add Licensed End Users” on page 3-23.
   - To change the license manager, click the pencil icon and follow the onscreen instructions. When you have finished, go to “Add Licensed End Users” on page 3-23.

Add Licensed End Users

With online licensing, any user of MATLAB Parallel Server must be added as a licensed end user. To add licensed end users, you must be a license administrator. If you are not a license administrator, provide a list of required licensed end users to the license administrator along with the following steps. If you are a license administrator, follow these steps.

1 If you are not already logged in as an administrator, go to License Center and log in with your MathWorks Administrator Account.
2 Select your MATLAB Parallel Server license, and then click **Manage Users**.
3 Click **Add User** to add a user to the list.


Add User

A MathWorks Account will be automatically created for the user, if one does not already exist.

Email Address:	
First Name:	
Last Name:	
Country:	United States

☑ Download Software

4 Provide the user’s email address, first and last names, and country. Click **Add User**. Note that if the specified email address does not correspond to an existing MathWorks Account, a new account is created for that user.
5 Add end users as necessary.

Get the Installation Files

If you are not a license administrator, you can download the Internet-based installer. For more information, see “Install Products Using Internet Connection”. Note that this method performs a full download of the necessary files per compute node that you are setting up.

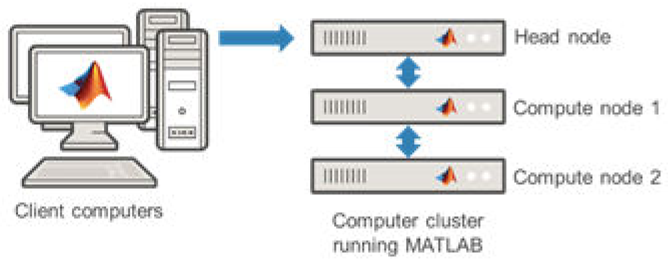
To save time and eliminate the need to run the installer-based download process on each computer in your cluster, download the installation files prior to installation. Doing so facilitates installation in a large number of machines. If you have access to an Administrator’s account for your license, you can use the installer to download files without installing them. If not, contact the administrator of your license to obtain a copy of the installation files. For more information, see “Download Products Without Installing”. Alternatively, proceed with the Internet-based installer. When you download the files using the installer, you must:

- Select the operating system for the cluster machines.
- Select all products for download. MATLAB Parallel Server cannot run jobs requiring products that are not installed.

MATLAB Parallel Server has two server-side components:

- The MATLAB Job Scheduler, which runs on the head node and manages jobs on your cluster. For more information, see “Install Software on All Nodes” on page 3-25.

- MATLAB Parallel Server, which runs on the compute nodes. For more information, see “Install Software on All Nodes” on page 3-25.



Install Software on All Nodes

To install the software on each computer in your cluster, follow these steps:

1. Start the MATLAB installer from the installation files downloaded in “Get the Installation Files” on page 3-24.
2. Select **Log in with a MathWorks Account** and follow the prompts.
3. Select all products that the end users will use.
4. After the installation completes, update the `mjs_def` file in `matlabroot/toolbox/parallel/bin`. Uncomment and set:
   - Unix: `USE_ONLINE_LICENSING="true"`
   - Windows: `USE_ONLINE_LICENSING=true`

For best performance, install the software locally on each node. However, you can also install the software in a network share location.

Note Install noninteractively (silently) instead if you want to

- Install the software on a machine without graphical user interface (GUI).
- Save the time that it takes to configure the installer for each compute node.

For more information, see “Install Noninteractively”.

You can test the installation and licensing by running the following command in a command-line interface. *`matlabroot`* is the MATLAB installation folder. *`filename`* is the location to write the log file to, for example, a temporary location. You must have write permissions in this location.

```
matlabroot/toolbox/parallel/bin/checkLicensing -logfile filename
```

Configure the MATLAB Job Scheduler with Admin Center

The MATLAB Job Scheduler is a scheduler that is provided with MATLAB Parallel Server. The MATLAB Job Scheduler is intended primarily for clusters that run only MATLAB jobs. The scheduler interface is a high-level abstraction that enables you to submit jobs to your computation resources, and allows you to avoid dealing with differences in operating systems and environments.

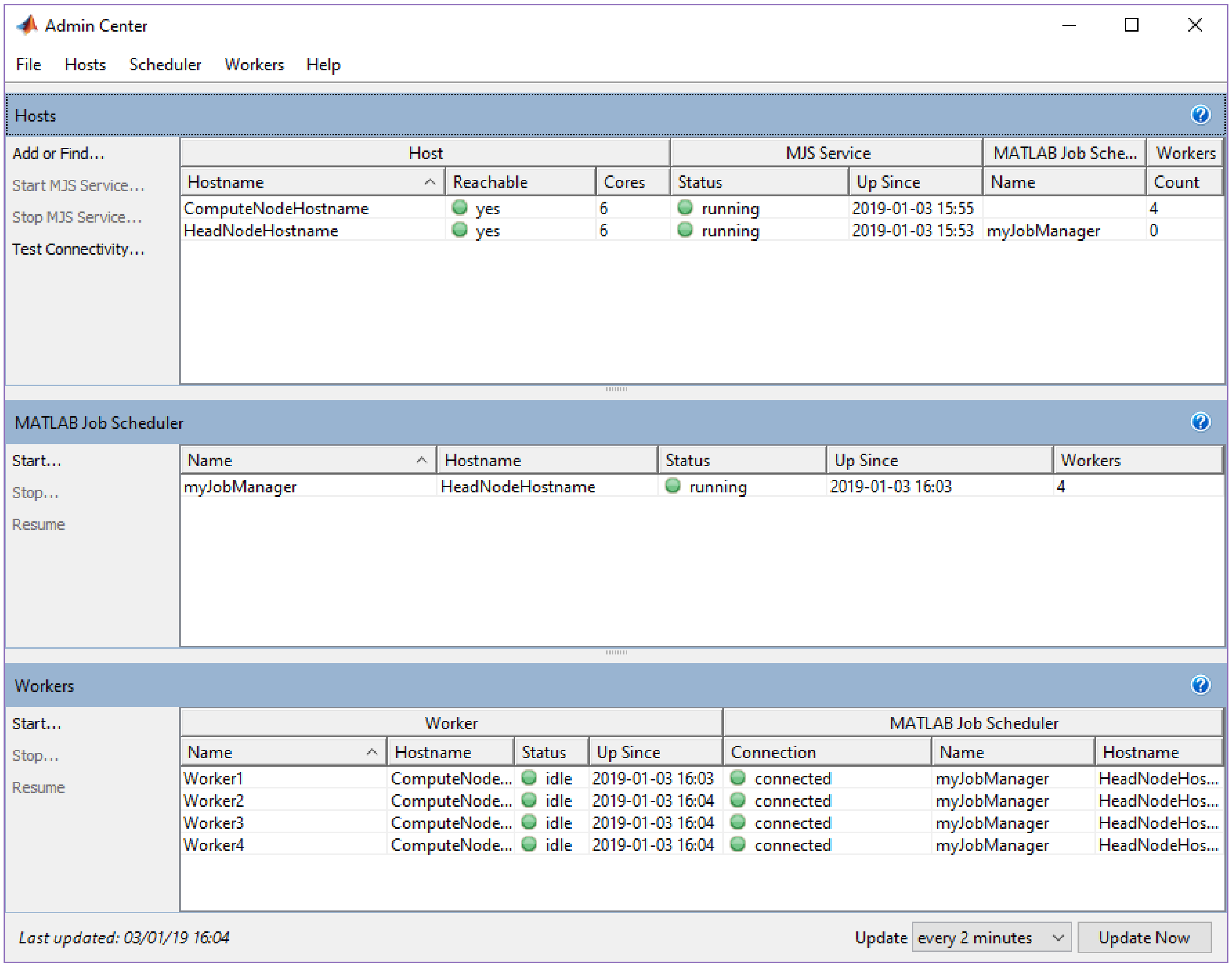
The following steps configure the MATLAB Job Scheduler with Admin Center, the graphical interface. If your machine does not provide graphics, use the command-line interface instead. For more information, see “Start mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)” on page 3-32.

1. On the head node, start Admin Center. Go to `matlabroot/toolbox/parallel/bin` and execute the file named `admincenter`. `matlabroot` is the MATLAB installation folder.
2. Click **Add or Find**, and specify the computers that you are using as your head node and compute nodes.
3. Follow the prompts and confirm to start the mjs service. If necessary, manually start the mjs service using the command-line interface. For more information, see “Use the Command-Line Interface (Windows)” on page 3-32 or “Use the Command-Line Interface (UNIX)” on page 3-34.
4. In the MATLAB Job Scheduler section, click **Start**. Specify a name for your MATLAB Job Scheduler and select the head node from the dropdown list.
5. To add MATLAB Parallel Server workers, click **Start** in the **Workers** section of the Admin Center.
   a. Select the computers to host the workers.
   b. Select the number of workers per computer.
6. Verify your configuration by checking worker status in the **Workers** section.
7. To troubleshoot issues, click **Test Connectivity** in the **Host** section.
8. If you use UNIX, configure the mjs service to start automatically at start time. For instructions, see “Start mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)” on page 3-32.

The head node uses computational resources to run the MATLAB Job Scheduler. If you set up workers on the head node, they compete for resources with the MATLAB Job Scheduler.

Tip Avoid setting up workers on the head node. If workers use too many system resources, such as memory, processor, network, or local storage, the job manager can become unresponsive.

The following screenshot shows the final setup in Admin Center.



Host			MJS Service		MATLAB Job Sche...	Workers
Hostname	Reachable	Cores	Status	Up Since	Name	Count
ComputeNodeHostname	● yes	6	● running	2019-01-03 15:55		4
HeadNodeHostname	● yes	6	● running	2019-01-03 15:53	myJobManager	0

Name	Hostname	Status	Up Since	Workers
myJobManager	HeadNodeHostname	● running	2019-01-03 16:03	4

Worker				MATLAB Job Scheduler		
Name	Hostname	Status	Up Since	Connection	Name	Hostname
Worker1	ComputeNode...	● idle	2019-01-03 16:03	● connected	myJobManager	HeadNodeHos...
Worker2	ComputeNode...	● idle	2019-01-03 16:04	● connected	myJobManager	HeadNodeHos...
Worker3	ComputeNode...	● idle	2019-01-03 16:04	● connected	myJobManager	HeadNodeHos...
Worker4	ComputeNode...	● idle	2019-01-03 16:04	● connected	myJobManager	HeadNodeHos...

Last updated: 03/01/19 16:04

Update: every 2 minutes

Note If you need more help during the configuration, such as your cluster requires firewall configuration or you want to set up multiple mjs installations, see this more detailed guide “Configure Advanced Options for MATLAB Job Scheduler Integration” on page 3-29.

Connect the MATLAB Client to the MATLAB Parallel Server Cluster

To use MATLAB Parallel Server, you must have a client computer running MATLAB and Parallel Computing Toolbox. In the MATLAB toolstrip, select **Parallel** > **Discover Clusters** and follow the instructions to automatically discover and set up your cluster.

Alternatively, you can configure it manually as follows:

1. In MATLAB, on the **Home** tab, select **Parallel** > **Create and Manage Clusters**.
2. Select **Add Cluster Profile** > **MATLAB Job Scheduler**.

- Create your MATLAB Job Scheduler profile and click **Edit**.
- Update the hostname of the head node.
- Update the license number.
- Click **Done** and select **Set as Default** (optional) .

After you successfully validate your cluster, you can now use your MATLAB session to submit jobs to the MATLAB Parallel Server cluster.

Note If your validation does not pass, contact the MathWorks install support team.

Any MATLAB cluster workers that you start use dynamic licensing: they can use all the functionality you are licensed for in the MATLAB client, while checking out only MATLAB Parallel Server licenses in the cluster.

For information on configuring more advanced options for your cluster, see “MATLAB Job Scheduler Cluster Customization”. For example, you can set the security of the cluster in “Set MATLAB Job Scheduler Cluster Security” on page 2-29. After you finish your configuration, try some examples of cluster workflows in “Running Code on Clusters and Clouds”.

See Also

Related Examples

- “MATLAB Job Scheduler Cluster Customization”
- “Set MATLAB Job Scheduler Cluster Security” on page 2-29

Configure Advanced Options for MATLAB Job Scheduler Integration

In this section...

“Run Multiple MATLAB Parallel Server Versions” on page 3-29
“Set Up Windows Cluster Hosts” on page 3-30
“Open Required Ports on Server” on page 3-30
“Stop mjs Services of Old Installation” on page 3-31
“Set the MATLAB Job Scheduler Security Level” on page 3-32
“Start mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)” on page 3-32
“Install the mjs Service to Start Automatically at Boot Time (UNIX)” on page 3-35
“Validate Installation with MATLAB Job Scheduler” on page 3-36

Follow these instructions to configure advanced options during integration of MATLAB Job Scheduler with your cluster.

Note If this is the first time you integrate MATLAB Job Scheduler, see the following for the most common configuration options: “Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager” on page 3-16.

In the following instructions, `matlabroot` refers to the location of your installed MATLAB Parallel Server software. Where you see this term used in the instructions that follow, substitute the path to your location.

Run Multiple MATLAB Parallel Server Versions

You can upgrade your MATLAB Job Scheduler clusters and continue to use the R2016a release onwards of Parallel Computing Toolbox on your MATLAB desktop client to connect to it. To take advantage of this backward compatibility feature:

1. Install the latest version of MATLAB Parallel Server on your cluster. You must use this version to run MATLAB Job Scheduler on your cluster.
2. Install MATLAB Parallel Server for each release that you want to support in the cluster. For example, to use R2016a and R2016b with your cluster, install both the R2016a and R2016b releases of MATLAB Parallel Server.
3. Configure MATLAB Job Scheduler with the location of these installations. In the `mjs_def` configuration file, specify the location of each installation of MATLAB Parallel Server in the `MJS_ADDITIONAL_MATLABROOTS` variable. You can find this file in *`matlabroot`*`/toolbox/parallel/bin` for Linux (`mjs_def.sh`) and Windows (`mjs_def.bat`). For more information, see mjs.

With this configuration, the MATLAB Job Scheduler allows MATLAB clients from the installed releases to submit jobs to the cluster. The MATLAB Job Scheduler dynamically starts the right version of the MATLAB worker to run the job.

Set Up Windows Cluster Hosts

If this is the first installation of MATLAB Parallel Server on a cluster of Windows machines, you need to configure these hosts for job communications.

Note If you do not have a Windows cluster, or if you have already installed a previous version of MATLAB Parallel Server on your Windows cluster, you can skip this step.

Configure Windows Firewalls on the Cluster Nodes

If you are using Windows firewalls on your cluster nodes,

1. Log in as a user with administrator privileges.
2. Execute the following in a Windows command prompt.

   ```
   matlabroot\toolbox\parallel\bin\addMatlabToWindowsFirewall.bat
   ```

   This command adds MATLAB as an allowed program. If you are using other firewalls, you must configure them for similar accommodation.

Configure Windows User Access for mjs

The user that mjs runs as requires access to the cluster MATLAB installation location. By default, mjs runs as the user `LocalSystem`. If your network allows `LocalSystem` to access the install location, you can skip this step. (If you are not sure of your network configuration and the access provided for `LocalSystem`, contact the MathWorks install support team.)

Note If `LocalSystem` cannot access the install location, you must run mjs as a different user.

You can set a different user with these steps:

1. With any standard text editor (such as WordPad) open the `mjs_def` file found at:

   ```
   matlabroot\toolbox\parallel\bin\mjs_def.bat
   ```
2. Find the line for setting the `MJSUSER` parameter, and provide a value in the form `domain\username`:

   ```
   set MJSUSER=mydomain\myusername
   ```
3. Provide the user password by setting the `MJSPASS` parameter:

   ```
   set MJSPASS=password
   ```
4. Save the file.

Open Required Ports on Server

The `mjs` service uses as many ports as required, starting with `BASE_PORT`. By default, `BASE_PORT` is 27350.

If you use a host that runs a total of `nJ` job managers and `nW` workers, the `mjs` service reserves a total of `6+2*nJ+4*nW` consecutive ports for its own use. All job managers and workers, even those on different hosts, that are going to work together must use the same base port. Otherwise the job managers and workers will not be able to contact each other. In addition, MPI communication occurs on ports starting at `BASE_PORT+1000` and use `2*nW` consecutive ports.

For example, if you use a host with 1 job manager and 16 workers, then you need the following ranges of ports to be open:

- `27350 – 27422` for the `mjs` service.
- `28350 – 28382` for MPI communication.

To connect from MATLAB to a cluster with a non-default `BASE_PORT`, you must append the value of `BASE_PORT` to the `'Host'` property in the MATLAB Job Scheduler cluster profile. You must do this in the form `Hostname:BASE_PORT`, for example `myMJSHost:44001`.

Stop mjs Services of Old Installation

If you have an older version of MATLAB Parallel Server running on your cluster nodes, you should stop the mjs services before starting the services of the new installation.

Stop mjs on Windows

1. Open a Windowscommand window with administrator privileges.
2. In the command window, navigate to the folder of the old installation that contains the control scripts.

   ```
   cd oldmatlabroot\toolbox\parallel\bin
   ```
3. Stop and uninstall the old service and remove its associated files by typing the following command.

   ```
   mjs uninstall -clean
   ```

   In releases before R2019a, the service is called mdce. Type the following commands instead.

   ```
   cd oldmatlabroot\toolbox\distcomp\bin
   mdce uninstall -clean
   ```

   Note Using the `-clean` flag permanently removes all existing job data. Be sure this data is no longer needed before removing it.
4. Repeat the instructions of this step on all worker nodes.

Stop mjs on UNIX

1. Log in as root. If you cannot log in as root, you must alter the following parameters in the *`oldmatlabroot`*`/toolbox/parallel/bin/mjs_def.sh` file to point to a folder for which you have write privileges: `CHECKPOINTBASE`, `LOGBASE`, `PIDBASE`, and `LOCKBASE` if applicable. In releases before R2019a, this file is *`oldmatlabroot`*`/toolbox/distcomp/bin/mdce_def.sh` instead.
2. **On each cluster node**, stop the mjs service and remove its associated files by typing the commands:

   ```
   cd oldmatlabroot/toolbox/parallel/bin
   ./mjs stop -clean
   ```

In releases before R2019a, the service is called mdce. Type the following command instead.

```
cd oldmatlabroot/toolbox/distcomp/bin
./mdce stop -clean
```

Note Using the `-clean` flag permanently removes all existing job data. Be sure this data is no longer needed before removing it.

Set the MATLAB Job Scheduler Security Level

Before starting the mjs service on your cluster nodes, set a security level. For instructions, see “Set Security Level” on page 2-29. For additional security considerations, see “Set MATLAB Job Scheduler Cluster Security” on page 2-29.

Start mjs Service, MATLAB Job Scheduler, and Workers (Command-Line)

You can start MATLAB Job Scheduler using a graphical interface or the command line. For instructions on how to use the graphical interface, see “Configure the MATLAB Job Scheduler” on page 3-18. To use the graphical interface, Admin Center, you must run it on a computer that has direct network connectivity to all the nodes of your cluster. If you cannot run Admin Center on such a computer, you must use the command-line interface. For instructions on how to use the command-line interface, follow the next steps.

Use the Command-Line Interface (Windows)

1. **Start the mjs Service**

   You must install the mjs service on all nodes (head node and worker nodes). Begin on the head node.

   a. Open a Windowscommand window with administrator privileges.
   b. In the Windows command window, navigate to the folder with the control scripts:

      ```
      cd matlabroot\toolbox\parallel\bin
      ```
   c. Install the mjs service by typing the command:

      ```
      mjs install
      ```
   d. Start the mjs service by typing the command:

      ```
      mjs start
      ```
   e. Repeat the instructions of this step on all worker nodes.

   As an alternative to items 3–5, you can install and start the mjs service on nodes remotely from one machine by typing:

   ```
   cd matlabroot\toolbox\parallel\bin
   ssh hostA mjs install
   ssh hostA mjs start
   ```

   where `hostA` refers to a remote host. For more information on your `ssh` utility, see the usage reminder by typing:

```
ssh
```

Once installed, the mjs service starts running each time the machine reboots. The mjs service continues to run until explicitly stopped or uninstalled, regardless of whether a MATLAB Job Scheduler or worker session is running.

2 **Start the MATLAB Job Scheduler**

To start the MATLAB Job Scheduler, enter the following commands in a Windows command prompt. You do not have to be at the machine on which the MATLAB Job Scheduler runs, as long as you have access to the MATLAB Parallel Server installation.

a In your Windows command prompt, navigate to the folder with the startup scripts:

```
cd matlabroot\toolbox\parallel\bin
```

b Start the MATLAB Job Scheduler, using any unique text you want for the name `<MyMJS>`:

```
startjobmanager -name <MyMJS> -remotehost <MATLAB Job Scheduler host name> -v
```

c Verify that the MATLAB Job Scheduler is running on the intended host.

```
nodestatus -remotehost <MATLAB Job Scheduler host name>
```

Note If you are executing `startjobmanager` on the host where the MATLAB Job Scheduler runs, you do not need to specify the `-remotehost` flag.

If you have more than one MATLAB Job Scheduler on your cluster, each must have a unique name.

3 **Start the Workers**

Note Before you can start a worker on a machine, the mjs service must already be running on that machine. If you are using the network license manager, it must be running on the network.

For each node used as a worker, enter the following commands in a Windows command prompt. You do not have to be at the machines where the MATLAB workers will run, as long as you have access to the MATLAB Parallel Server installation.

a Navigate to the folder with the startup scripts:

```
cd matlabroot\toolbox\parallel\bin
```

b Start the workers on each node, using the text for `<MyMJS>` that identifies the name of the MATLAB Job Scheduler you want this worker registered with. Enter this text on a single line:

```
startworker -jobmanagerhost <MATLAB Job Scheduler host name>
    -jobmanager <MyMJS> -remotehost <worker host name> -v
```

To run more than one worker session on the same node, give each worker a unique name by including the `-name` option on the `startworker` command, and run it for each worker on that node:

```
startworker ... -name <worker1 name>
startworker ... -name <worker2 name>
```

c Verify that the workers are running.

```
nodestatus -remotehost <worker host name>
```

d Repeat items 2–3 for all worker nodes.

For more information about mjs, MATLAB Job Scheduler, and worker processes, such as how to shut them down or customize them, see “MATLAB Job Scheduler Cluster Customization”.

Use the Command-Line Interface (UNIX)

1 **Start the mjs Service**

On each cluster node, start the mjs service by typing the commands:

```
cd matlabroot/toolbox/parallel/bin
./mjs start
```

Alternatively, you can start the mjs service on nodes remotely from one machine by typing

```
cd matlabroot/toolbox/parallel/bin
ssh hostA matlabroot/toolbox/parallel/bin/mjs start
ssh hostB matlabroot/toolbox/parallel/bin/mjs start
ssh hostC matlabroot/toolbox/parallel/bin/mjs start
```

where `hostA`, `hostB`, and `hostC` refers to your remote host names. For a long list of host names, you can start the mjs service on several nodes remotely from one machine by typing:

```
for host in hostA hostB hostC;
 do ssh $host matlabroot/toolbox/parallel/bin/mjs start;
done
```

For more information on your `ssh` utility, see the usage reminder by typing:

```
ssh
```

Alternatively, you can access the system reference manual by typing:

```
man ssh
```

2 **Start the MATLAB Job Scheduler**

To start the MATLAB Job Scheduler, enter the following commands. You do not have to be at the machine on which the MATLAB Job Scheduler runs, as long as you have access to the MATLAB Parallel Server installation.

a Navigate to the folder with the startup scripts:

```
cd matlabroot/toolbox/parallel/bin
```

b Start the MATLAB Job Scheduler, using any unique text you want for the name `<MyMJS>`. Enter this text on a single line.

```
./startjobmanager -name <MyMJS> -remotehost <MATLAB Job Scheduler host name> -v
```

c Verify that the MATLAB Job Scheduler is running on the intended host:

```
./nodestatus -remotehost <MATLAB Job Scheduler host name>
```

Note If you have more than one MATLAB Job Scheduler on your cluster, each must have a unique name.

3 Start the Workers

Note Before you can start a worker on a machine, the mjs service must already be running on that machine. If you are using the network license manager, it must be running on the network.

For each computer hosting a MATLAB worker, enter the following commands. You do not have to be at the machines where the MATLAB workers run, as long as you have access to the MATLAB Parallel Server installation.

a. Navigate to the folder with the startup scripts:

   ```
   cd matlabroot/toolbox/parallel/bin
   ```

b. Start the workers on each node, using the text for `<MyMJS>` that identifies the name of the MATLAB Job Scheduler you want this worker registered with. Enter this text on a single line:

   ```
   ./startworker -jobmanagerhost <MATLAB Job Scheduler host name>
      -jobmanager <MyMJS> -remotehost <worker host name> -v
   ```

   To run more than one worker session on the same machine, give each worker a unique name with the `-name` option:

   ```
   ./startworker ... -name <worker1>
   ./startworker ... -name <worker2>
   ```

c. Verify that the workers are running. Repeat this command for each worker node:

   ```
   ./nodestatus -remotehost <worker host name>
   ```

For more information about mjs, MATLAB Job Scheduler, and worker processes, such as how to shut them down or customize them, see “MATLAB Job Scheduler Cluster Customization”.

Install the mjs Service to Start Automatically at Boot Time (UNIX)

Although this step is not required, it is helpful in case of a system crash. Once configured for this, the mjs service starts running each time the machine reboots. The mjs service continues to run until explicitly stopped, regardless of whether a MATLAB Job Scheduler or worker session is running.

You must have root privileges to do this step.

Debian, Fedora, SUSE, and Red Hat (non-Fedora) Platforms

On each cluster node, register the mjs service as a known service and configure it to start automatically at system boot time by following these steps:

1. Create the following link, if it does not already exist:

   ```
   ln -s matlabroot/toolbox/parallel/bin/mjs /etc/mjs
   ```

2. Create the following link to the boot script file:

   ```
   ln -s matlabroot/toolbox/parallel/bin/mjs /etc/init.d/mjs
   ```

3. Set the boot script file permissions:

   ```
   chmod 555 /etc/init.d/mjs
   ```

4 Find your default run level. If you have a SysV Linux machine, you can determine the default run level by booting your machine and immediately executing the `$runlevel` command. The second number output is the default run level of your system. If your Linux machine does not support SysV, look in `/etc/inittab` for the default run level.

5 When you have determined the run level, create a link in the `rc` folder associated with that run level. For example, if the run level is `5`, execute one of the following sets of platform-specific commands.

   - Debian and Fedora platforms:

     ```
     cd /etc/rc5.d;
     ln -s ../init.d/mjs S99MJS
     ```
   - SUSE platform:

     ```
     cd /etc/init.d/rc5.d;
     ln -s ../mjs S99MJS
     ```
   - Red Hat platform (non-Fedora):

     ```
     cd /etc/rc.d/rc5.d;
     ln -s ../../init.d/mjs S99MJS
     ```

Validate Installation with MATLAB Job Scheduler

To verify that your MATLAB Parallel Server products are installed and configured correctly, create a cluster profile and validate it. For instructions, see “Connect MATLAB Client to MATLAB Parallel Server Cluster” on page 3-20. You can specify the number of workers to use when validating your profile, to avoid occupying the whole cluster. If your validation does not pass, contact the MathWorks Install Support Team, or see “Troubleshoot Common Problems” on page 2-37.

After you create a cluster profile, you can make any modifications appropriate for your applications, such as `NumWorkersRange`, `AttachedFiles`, or `AdditionalPaths`. To save your profile for other users, in the Cluster Profile Manager, select the profile and click **Export**, then save your profile to a file in a convenient location. Later, when running the Cluster Profile Manager, other users can import your profile by clicking **Import**. For more information about cluster profiles, see “Discover Clusters and Use Cluster Profiles” (Parallel Computing Toolbox).

See Also

Related Examples

- “Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager” on page 3-16

More About

- “MATLAB Job Scheduler Cluster Customization”
- “Troubleshooting in MATLAB Parallel Server”
- “Get Started with MATLAB Parallel Server”

Configure for HPC Pack

In this section...
"Configure Cluster for Microsoft HPC Pack" on page 3-37
"Configure Client Computer for HPC Pack" on page 3-37
"Validate Installation Using Microsoft HPC Pack" on page 3-38

Configure Cluster for Microsoft HPC Pack

Follow these instruction to configure your MATLAB Parallel Server installation to work with Microsoft HPC Pack or Compute Cluster Server (CCS). In the following instructions, *`matlabroot`* refers to the MATLAB installation location.

Supported versions: MATLAB Compute Cluster Server 2003, Windows HPC Server 2008, Windows HPC Server 2008 R2, Microsoft HPC Pack 2012, Microsoft HPC Pack 2012 R2, and Microsoft HPC Pack 2016.

Note If you are using an HPC Pack in a network share installation, the network share location must be in the "Intranet" zone. You might need to adjust the Internet Options for your cluster nodes and add the network share location to the list of Intranet sites.

1. Log in on the cluster head node as a user with administrator privileges.
2. Open a command window with administrator privileges and run the following file command

   ```
   matlabroot\toolbox\parallel\bin\MicrosoftHPCServerSetup.bat -cluster
   ```

   If you are using an HPC Pack in a network share installation, this command performs some of the setup required for all machines in the cluster. The location of the MATLAB installation must be the same on every cluster node. If you are not using an HPC Pack in a network share installation, then you must run this command on every cluster node.

   Note If you need to override the script default values, modify the values defined in `MicrosoftHPCServerSetup.xml` before running `MicrosoftHPCServerSetup.bat`. Use the `-def_file` argument to the script when using a `MicrosoftHPCServerSetup.xml` file in a custom location. For example:

   ```
   MicrosoftHPCServerSetup.bat -cluster -def_file <filename>
   ```

   You modify the file only on the node where you actually run the script.

   An example of one of the values you might set is for `CLUSTER_NAME`. If you provide a friendly name for the cluster in this parameter, it is recognized by MATLAB's discover clusters feature and displayed in the resulting cluster list.

Configure Client Computer for HPC Pack

This configuring applies to all versions of HPC Pack.

Note If you are using HPC Pack in a network share installation, the network share location must be in the "Intranet" zone. You might need to adjust the Internet Options for your cluster nodes and add the network share location to the list of Intranet sites.

1. Open a command window with administrator privileges and run the following file command

   ```
   matlabroot\toolbox\parallel\bin\MicrosoftHPCServerSetup.bat -client
   ```

   This command performs some of the setup required for a client machine.

   Note If you need to override the default values the script, modify the values defined in `MicrosoftHPCServerSetup.xml` before running `MicrosoftHPCServerSetup.bat`. Use the `-def_file` argument to the script when using a `MicrosoftHPCServerSetup.xml` file in a custom location. For example:

   ```
   MicrosoftHPCServerSetup.bat -client -def_file <filename>
   ```

2. To submit jobs or discover the cluster from MATLAB, the Microsoft HPC Pack client utilities must be installed on your MATLAB client machine. If they are not already installed and up to date, ask your system administrator for the correct client utilities to install. The utilities are available from Microsoft download center.

   If you have installed multiple versions of the Microsoft HPC Pack client utilities, MATLAB uses the most recent install. To configure MATLAB to use a specific install, set the environment variable `'MATLAB_HPC_SERVER_HOME'` to the install location of the client utilities you want to use.

Validate Installation Using Microsoft HPC Pack

This procedure verifies that your parallel computing products are installed and configured correctly for using Microsoft Windows HPC Pack or Compute Cluster Server (CCS).

Step 1: Create a Cluster Profile

In this step you create a cluster profile to use in subsequent steps. To create a cluster profile, try discovering your cluster. On the **Home** tab, in the **Environment** area, select **Parallel** > **Discover Clusters**. For more information, see "Discover Clusters" (Parallel Computing Toolbox). Alternatively, if your cluster is not available for discovery, follow these steps.

1. Start the Cluster Profile Manager. On the **Home** tab, in the **Environment** area, select **Parallel** > **Create and Manage Clusters**.
2. Create a new profile in the Cluster Profile Manager by selecting **Add Cluster Profile** > **HPC Server**.
3. With the new profile selected in the list, click **Rename** and edit the profile name to be `HPCtest`. Press **Enter**.
4. In the Properties tab, provide text for the following fields:
   - a. Set the **Description** field to `For testing installation with HPC Server`.
   - b. Set the **NumWorkers** field to the number of workers you want to run the validation tests on, within the limitation of your licensing.

c Set the **Host** field to the name of the host on which your scheduler is running. Depending on your network, this might be a simple host name, or it might have to be a fully qualified domain name.

Note: The following four property settings (`JobStorageLocation`, `ClusterMatlabRoot`, `ClusterVersion`, and `UseSOAJobSubmission`) are optional, and need to be set in the profile here only if you did not run `MicrosoftHPCServerSetup.bat` as described in "Configure Cluster for Microsoft HPC Pack" on page 3-37, or if you want to override the setting established by that script.

d Set the **JobStorageLocation** to the location where you want job and task data to be stored. This must be accessible to all the worker machines.

Note `JobStorageLocation` should not be shared by parallel computing products running different versions; each version on your cluster should have its own `JobStorageLocation`.

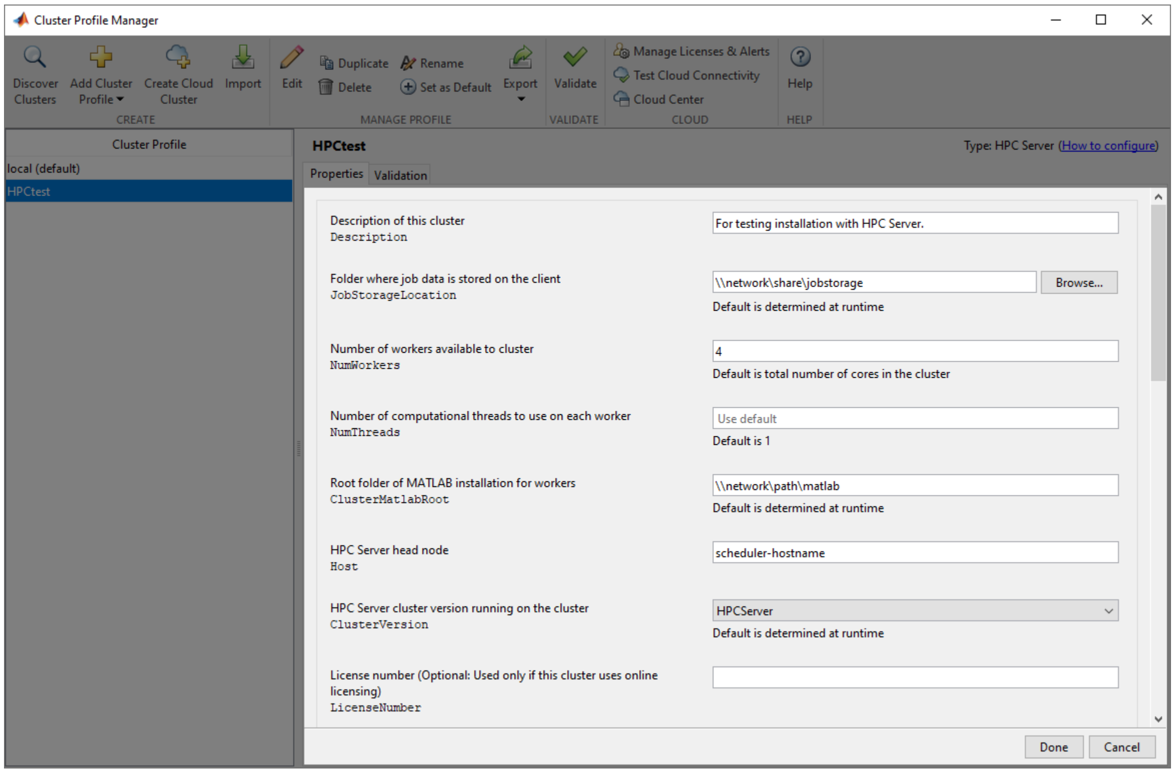
---

e Set the **ClusterMatlabRoot** to the installation location of the MATLAB to be executed by the worker machines, as determined in Chapter 1 of the installation instructions.

f Set the **ClusterVersion** field to `HPCServer` or `CCS`.

g If you want to test SOA job submissions on an HPC Server cluster, set **UseSOAJobSubmission** to `true`. If you plan to use SOA job submissions with your cluster, you should test this first without SOA submission, then later return and test it with SOA job submission. The default value is determined at runtime based on your scheduler.

So far, the dialog box should look like the following figure:



HPCtest — Type: HPC Server (How to configure)

Description of this cluster `Description`	For testing installation with HPC Server.
Folder where job data is stored on the client `JobStorageLocation`	\\network\share\jobstorage — Default is determined at runtime
Number of workers available to cluster `NumWorkers`	4 — Default is total number of cores in the cluster
Number of computational threads to use on each worker `NumThreads`	Use default — Default is 1
Root folder of MATLAB installation for workers `ClusterMatlabRoot`	\\network\path\matlab — Default is determined at runtime
HPC Server head node `Host`	scheduler-hostname
HPC Server cluster version running on the cluster `ClusterVersion`	HPCServer — Default is determined at runtime
License number (Optional: Used only if this cluster uses online licensing) `LicenseNumber`	

5 Click **Done** to save your cluster profile.

Step 2: Validate the Configuration

In this step you validate your cluster profile, and thereby your installation. You can specify the number of workers to use when validating your profile. If you do not specify the number of workers in the **Validation** tab, then the validation will attempt to use as many workers as the value specified by the `NumWorkers` property on the **Properties** tab. You can specify a smaller number of workers to validate your configuration without occupying the whole cluster.

1. If it is not already open, start the Cluster Profile Manager from the MATLAB desktop by selecting on the **Home** tab in the **Environment** area **Parallel > Create and Manage Clusters**.
2. Select your cluster profile in the listing.
3. Click **Validation** tab.
4. Use the checkboxes to choose all tests, or a subset of the validation stages, and specify the number of workers to use when validating your profile.
5. Click **Validate**.

The Validation Results tab shows the output. The following figure shows the results of a profile that passed all validation tests.

Note If your validation does not pass, contact the MathWorks install support team.

If your validation passed, you now have a valid profile that you can use in other parallel applications. You can make any modifications to your profile appropriate for your applications, such as `NumWorkersRange`, `AttachedFiles`, `AdditionalPaths`, etc.

To save your profile for other users, select the profile and click **Export**, then save your profile to a file in a convenient location. Later, when running the Cluster Profile Manager, other users can import your profile by clicking **Import**.

Configure for Slurm, PBS Pro, OpenPBS, LSF, TORQUE

Follow these instructions to configure your MATLAB Parallel Server installation to work with Slurm, PBS Pro, OpenPBS, LSF, and TORQUE using built-in cluster types.

You can create a cluster profile using either a built-in cluster type or `Generic`. As a best practice, use built-in cluster types where possible.

You must use a `Generic` cluster profile when:

- You connect to a cluster that does not have a built-in cluster type
- The MATLAB client and the cluster nodes do not have a shared file system
- The MATLAB client machine is unable to directly submit jobs to the third-party scheduler, or
- You need to fully customize how parallel jobs are submitted to the cluster

To configure a cluster using the `Generic` cluster type, see "Configure Using the Generic Scheduler Interface" on page 3-45.

Create Cluster Profile and Validate Installation

This procedure verifies that the parallel computing products are installed and configured correctly on your cluster.

Step 1: Create a Cluster Profile

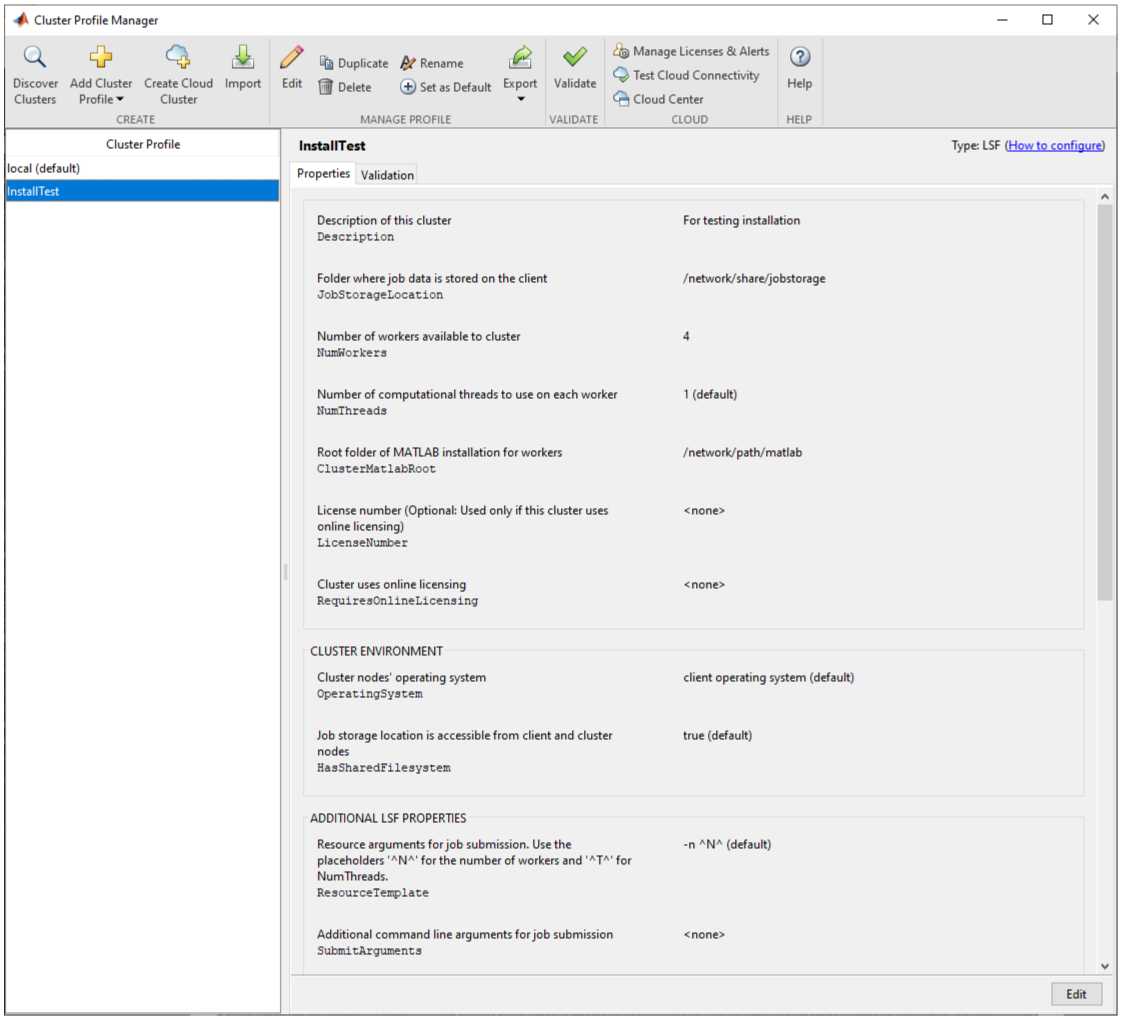
In this step you create a cluster profile to use in subsequent steps.

1. Start the Cluster Profile Manager. On the **Home** tab, in the **Environment** area, select **Parallel > Create and Manage Clusters**.
2. Create a new profile in the Cluster Profile Manager by selecting **Add Cluster Profile > LSF** (or **Slurm**, **PBS Pro** or **Torque**, as appropriate).
3. With the new profile selected in the list, click **Rename** and edit the profile name to be `InstallTest`. Press **Enter**.
4. In the Properties tab, provide settings for the following fields:
   a. Set the **Description** field to `For testing installation`.
   b. Set the **JobStorageLocation** to the location where you want job and task data to be stored (accessible to all the worker machines if you have a shared file system).

      Note `JobStorageLocation` should not be shared by parallel computing products running different versions; each version on your cluster should have its own `JobStorageLocation`.
   c. Set the **NumWorkers** field to the number of workers you want to run the validation tests on, within the limitation of your licensing.
   d. Set the **ClusterMatlabRoot** to the installation location of the MATLAB to be executed by the worker machines.
   e. Set the **SubmitArguments** to include any additional command arguments required by your particular cluster and scheduler.
   f. If you are using LSF, set the **OperatingSystem** to the operating system of your worker machines.

g Set **HasSharedFilesystem** to indicate if client and workers can share the same data location.

The dialog box should look something like this, or slightly different for Slurm, PBS Pro, OpenPBS, or TORQUE schedulers.



InstallTest

Description of this cluster `Description`	For testing installation
Folder where job data is stored on the client `JobStorageLocation`	/network/share/jobstorage
Number of workers available to cluster `NumWorkers`	4
Number of computational threads to use on each worker `NumThreads`	1 (default)
Root folder of MATLAB installation for workers `ClusterMatlabRoot`	/network/path/matlab
License number (Optional: Used only if this cluster uses online licensing) `LicenseNumber`	<none>
Cluster uses online licensing `RequiresOnlineLicensing`	<none>

CLUSTER ENVIRONMENT

Cluster nodes' operating system `OperatingSystem`	client operating system (default)
Job storage location is accessible from client and cluster nodes `HasSharedFilesystem`	true (default)

ADDITIONAL LSF PROPERTIES

Resource arguments for job submission. Use the placeholders '^N^' for the number of workers and '^T^' for NumThreads. `ResourceTemplate`	-n ^N^ (default)
Additional command line arguments for job submission `SubmitArguments`	<none>

5 Click **Done** to save your cluster profile.

Step 2: Validate the Cluster Profile

In this step you verify your cluster profile, and thereby your installation. You can specify the number of workers to use when validating your profile. If you do not specify the number of workers in the **Validation** tab, then the validation will attempt to use as many workers as the value specified by the `NumWorkers` property on the **Properties** tab. You can specify a smaller number of workers to validate your configuration without occupying the whole cluster.

1 If it is not already open, start the Cluster Profile Manager from the MATLAB desktop. On the **Home** tab, in the **Environment** area, select **Parallel** > **Create and Manage Clusters**.

2 Select your cluster profile in the listing.

3 Click **Validation** tab.

4. Use the checkboxes to choose all tests, or a subset of the validation stages, and specify the number of workers to use when validating your profile.
5. Click **Validate**.

The Validation Results tab shows the output. The following figure shows the results of a profile that passed all validation tests.

Note If your validation does not pass, contact the MathWorks install support team.

If your validation passed, you now have a valid profile that you can use in other parallel applications. You can make any modifications to your profile appropriate for your applications, such as `NumWorkersRange`, `AttachedFiles`, `AdditionalPaths`, etc.

To save your profile for other users, select the profile and click **Export**, then save your profile to a file in a convenient location. Later, when running the Cluster Profile Manager, other users can import your profile by clicking **Import**.

Configure LSF Scheduler on Windows Cluster

If your cluster is already set up to use mpiexec and smpd, you can use Parallel Computing Toolbox software with your existing configuration if you are using a compatible MPI implementation library (as defined in *`matlabroot`*`\toolbox\parallel\mpi\mpiLibConf.m`). However, if you do not have mpiexec on your cluster and you want to use it, you can use the mpiexec software shipped with the parallel computing products.

For further information about mpiexec and smpd, see the MPICH home page. For user's guides and installation instructions on that page, select **Documentation > User Docs**.

In the following instructions, *`matlabroot`* refers to the MATLAB installation location.

To use mpiexec to distribute a job, the smpd service must be running on all nodes that will be used for running MATLAB workers.

Note The `smpd` executable does not support running from a mapped drive. Use either a local installation, or the full UNC path name to the executable. Microsoft Windows Vista® does not support the `smpd` executable on network share installations, so with Windows Vista the installation must be local.

---

1. Log in as a user with administrator privileges.
2. Start smpd by typing in a Windows command prompt:

   ```
   matlabroot\bin\win64\smpd -install
   ```

   This command installs the service and starts it. As long as the service remains installed, it will start each time the node boots.
3. If this is a worker machine and you did not run the installer on it to install MATLAB Parallel Server software (for example, if you are running MATLAB Parallel Server software from a shared installation), execute the following command in a Windows command prompt.

```
matlabroot\bin\matlab.bat -install_vcrt
```

   This command installs the Microsoft run-time libraries needed for running jobs with your scheduler.

4. If you are using Windows firewalls on your cluster nodes, execute the following in a Windows command prompt.

   ```
   matlabroot\toolbox\parallel\bin\addMatlabToWindowsFirewall.bat
   ```

   This command adds MATLAB as an allowed program. If you are using other firewalls, you must configure them to make similar accommodation.
5. Log in as the user who will be submitting jobs for execution on this node.
6. Register this user to use mpiexec by typing:

   ```
   matlabroot\bin\win64\mpiexec -register
   ```
7. Repeat steps 5–6 for all users who will run jobs on this machine.
8. Repeat all these steps on all Windows nodes in your cluster.

Configure Windows Firewalls on Server

If you are using Windows firewalls on your cluster nodes,

1. Log in as a user with administrative privileges.
2. Execute the following in a Windows command prompt.

   ```
   matlabroot\toolbox\parallel\bin\addMatlabToWindowsFirewall.bat
   ```

   This command adds MATLAB as an allowed program. If you are using other firewalls, you must configure them for similar accommodation.

Configure Using the Generic Scheduler Interface

The generic scheduler interface provides flexibility to configure the interaction of the MATLAB client, MATLAB workers, and a third-party scheduler. Use the generic scheduler interface when you want complete customization for interfacing MATLAB with your scheduler setup.

You can create a cluster profile using either a built-in cluster type or `Generic`. As a best practice, use built-in cluster types where possible.

You must use a `Generic` cluster profile when:

- You connect to a cluster that does not have a built-in cluster type
- The MATLAB client and the cluster nodes do not have a shared file system
- The MATLAB client machine is unable to directly submit jobs to the third-party scheduler, or
- You need to fully customize how parallel jobs are submitted to the cluster

To configure a cluster using a built-in cluster type, see “Configure for Slurm, PBS Pro, OpenPBS, LSF, TORQUE” on page 3-41, “Configure a Hadoop Cluster” on page 3-64, or “Configure for HPC Pack” on page 3-37.

Interface with Third-Party Schedulers

The generic scheduler interface provides a means of getting tasks from your Parallel Computing Toolbox client session to your scheduler and cluster nodes. To achieve this, you must provide your MATLAB client with a set of plugin scripts. The scripts contain instructions specific to your cluster infrastructure, such as how to communicate with the job scheduler, and how to transfer job and task data to cluster nodes.

Support Scripts

To help you to interface with your scheduler, MathWorks provides add-ons, or plugins for the following third-party schedulers, which you can download from GitHub® repositories or the Add-On Manager.

Plugin	GitHub Repository
Parallel Computing Toolbox plugin for MATLAB Parallel Server with Slurm	https://github.com/mathworks/matlab-parallel-slurm-plugin
Parallel Computing Toolbox plugin for MATLAB Parallel Server with IBM Spectrum LSF	https://github.com/mathworks/matlab-parallel-lsf-plugin
Parallel Computing Toolbox plugin for MATLAB Parallel Server with Grid Engine	https://github.com/mathworks/matlab-parallel-gridengine-plugin
Parallel Computing Toolbox plugin for MATLAB Parallel Server with PBS	https://github.com/mathworks/matlab-parallel-pbs-plugin
Parallel Computing Toolbox plugin for MATLAB Parallel Server with HTCondor	https://github.com/mathworks/matlab-parallel-htcondor-plugin

Use either of the these workflows to get download the appropriate plugin scripts for your scheduler.

- You can download the plugins from a GitHub repository.

  - Clone the GitHub repository from a command windows on your machine. For example, to clone the repository for the Parallel Computing Toolbox plugin for MATLAB Parallel Server with IBM Spectrum LSF, use:

    ```
    git clone https://github.com/mathworks/matlab-parallel-lsf-plugin
    ```
  - Visit the GitHub page in a browser and download the plugin as a ZIP archive.
- Alternatively, to install the add-ons from the MATLAB Add-On manger, go to the **Home** tab and, in the **Environment** section, click the **Add-Ons** icon. In the Add-On Explorer, search for the add-on and install it.
- You can also download the plugins from MATLAB Central™ File Exchange.

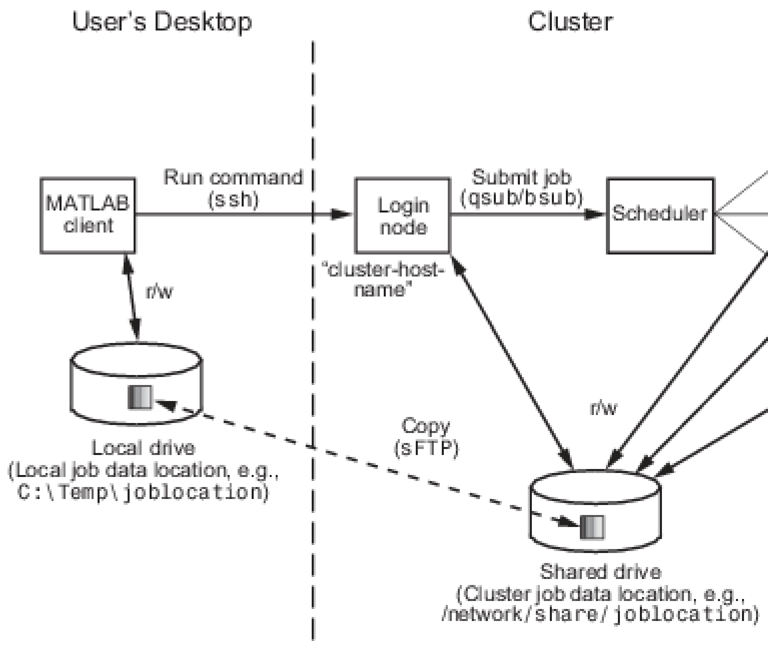
Additional Information

- If the MATLAB client is unable to directly submit jobs to the scheduler, MATLAB supports the use of the `ssh` protocol to submit commands to a remote cluster.
- If the client and the cluster nodes do not have a shared file system, MATLAB supports the use of `sftp` (SSH File Transfer Protocol) to copy job and task files between your computer and the cluster.
- If you want to customize the behavior of the plugin scripts, you can set additional properties, such as `AdditionalSubmitArgs`. For more information, see “Customize Behavior of Sample Plugin Scripts” on page 3-55.
- If your scheduler or cluster configuration is not supported by one of the repositories, it is recommended that you modify the scripts of one of these packages. For more information on how to write a set of plugin scripts for generic schedulers, see “Plugin Scripts for Generic Schedulers” (Parallel Computing Toolbox).

Create a Generic Cluster Profile

Sample Setup for LSF

This example shows how to set up your cluster profile to use the generic scheduler interface. It shows the set up of an LSF scheduler in a network without a shared file system between the client and the cluster machines. The following diagram illustrates the cluster setup:



In this type of configuration, job data is copied from the client host running a Windows operating system to a host on the cluster (cluster login node) running a UNIX® operating system. From the cluster login node, the LSF `bsub` command submits the job to the scheduler. When the job finishes, its output is copied back to the client host.

Requirements

The setup must meet the following conditions:

- The client node and cluster login node must support `ssh` and `sftp`.
- The cluster login node must be able to call the `bsub` command to submit a job to an LSF scheduler. You can find more about this in the `README` file in the `nonshared` subfolder within the installation folder.

Configure a Cluster Profile

Follow these steps to configure the cluster profile. You can modify any of these options depending on your setup.

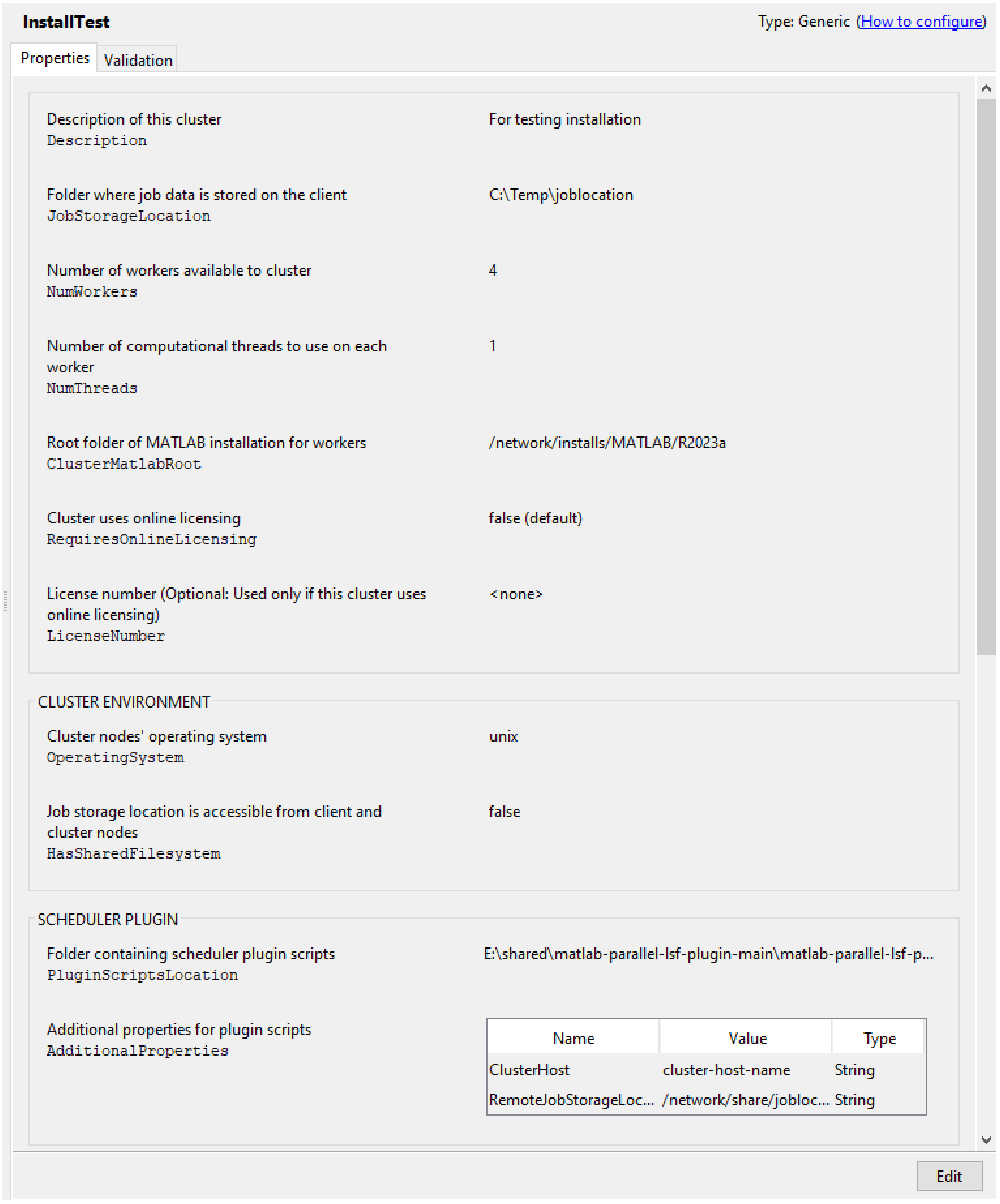
1. Extract the LSF GitHub repository folder and move it to a location that MATLAB clients can access.
2. Start a MATLAB session on the client host.
3. Start the Cluster Profile Manager from the MATLAB desktop. On the **Home** tab, in the **Environment** section, select **Parallel** > **Create and Manage Clusters**.
4. Create a new profile in the Cluster Profile Manager by selecting **Add Cluster Profile** > **Generic**.
5. With the new profile selected in the list, in the **Manage Profile** section, select **Rename** and change the profile name to `InstallTest`. Press **Enter**.
6. In the **Properties** tab, select **Edit** and provide settings for the following fields:

   a. Set the **Description** field to *For testing installation*.

b. Set the **JobStorageLocation** to the location where you want job and task data to be stored on the client machine (not the cluster location), for example, `C:\Temp\joblocation`.

   You must not share **JobStorageLocation** among parallel computing products running different versions. Each version on your cluster must have its own **JobStorageLocation**.
c. Set **NumWorkers** to the number of workers for which you want to test your installation.
d. Set **NumThreads** to the number of threads to use on each worker.
e. Set **ClusterMatlabRoot** to the installation location of MATLAB to run on the worker machines.
f. If the cluster uses online licensing, set **RequiresOnlineLicensing** to true.
g. If you set **RequiresOnlineLicensing** to `true`, enter your **LicenseNumber**.
h. Set **OperatingSystem** to the operating system of your cluster worker machines.
i. Set **HasSharedFilesystem** to `false`. This setting indicates that the client node and worker nodes cannot share the same data location.
j. Set the **PluginScriptsLocation** to the location of your plugin scripts.
k. To connect to a remote cluster, under the **AdditionalProperties** table, select **Add**. Specify a new property with name `ClusterHost`, value `cluster-host-name`, and type `String`.
l. To run jobs on a remote cluster without a shared file system, under the **AdditionalProperties** table, select **Add**. Specify a new property with name `RemoteJobStorageLocation`, value `/network/share/joblocation`, and type `String`.

7. Click **Done** to save your cluster profile changes. The dialog box looks as follows:



InstallTest

Type: Generic (How to configure)

Properties | Validation

Description of this cluster `Description`	For testing installation
Folder where job data is stored on the client `JobStorageLocation`	C:\Temp\joblocation
Number of workers available to cluster `NumWorkers`	4
Number of computational threads to use on each worker `NumThreads`	1
Root folder of MATLAB installation for workers `ClusterMatlabRoot`	/network/installs/MATLAB/R2023a
Cluster uses online licensing `RequiresOnlineLicensing`	false (default)
License number (Optional: Used only if this cluster uses online licensing) `LicenseNumber`	<none>

CLUSTER ENVIRONMENT

Cluster nodes' operating system `OperatingSystem`	unix
Job storage location is accessible from client and cluster nodes `HasSharedFilesystem`	false

SCHEDULER PLUGIN

Folder containing scheduler plugin scripts `PluginScriptsLocation`	E:\shared\matlab-parallel-lsf-plugin-main\matlab-parallel-lsf-p...

Additional properties for plugin scripts `AdditionalProperties`

Name	Value	Type
ClusterHost	cluster-host-name	String
RemoteJobStorageLoc...	/network/share/jobloc...	String

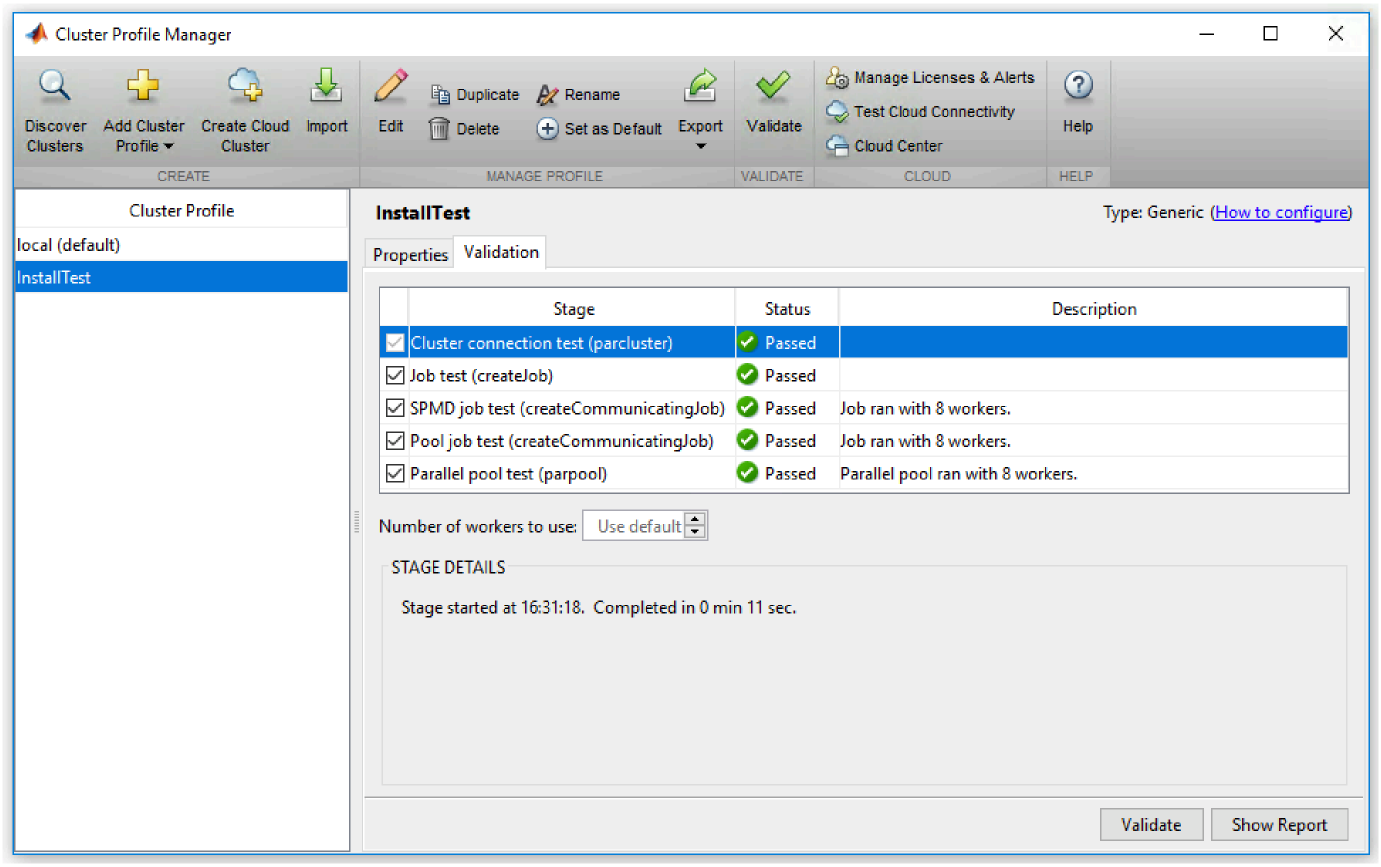
To check that the profile works, perform a validation following the steps in “Validate Cluster Profile and Installation” on page 3-50.

Validate Cluster Profile and Installation

You can specify the number of workers to use when validating your profile. If you do not specify the number of workers in the **Validation** tab, then the validation process attempts to use as many workers as the value specified by the **NumWorkers** property on the **Properties** tab. You can specify a smaller number of workers to validate your configuration without occupying the whole cluster.

1. Start the Cluster Profile Manager from the MATLAB desktop. On the **Home** tab, in the **Environment** area, select **Parallel** > **Create and Manage Clusters**.
2. Select your cluster profile in the listing.
3. Click the **Validation** tab.
4. Use the checkboxes to choose all tests, or a subset of the validation stages, and specify the number of workers to use when validating your profile.
5. Click **Validate**.

The **Validation** results tab shows the output. The following figure shows the results of a profile that passed all validation tests.



InstallTest

Type: Generic (How to configure)

Stage	Status	Description
☑ Cluster connection test (parcluster)	✓ Passed	
☑ Job test (createJob)	✓ Passed	
☑ SPMD job test (createCommunicatingJob)	✓ Passed	Job ran with 8 workers.
☑ Pool job test (createCommunicatingJob)	✓ Passed	Job ran with 8 workers.
☑ Parallel pool test (parpool)	✓ Passed	Parallel pool ran with 8 workers.

Number of workers to use: Use default

STAGE DETAILS

Stage started at 16:31:18. Completed in 0 min 11 sec.

Note If your validation fails any stage, contact the MathWorks install support team.

If your validation passes, you have a valid profile that you can use in other parallel applications. You can make any modifications to your profile that are appropriate for your applications, such as **NumWorkersRange**, **AttachedFiles**, or **AdditionalPaths**.

To save your profile for other users, select the profile, and click **Export**. Then save your profile to a file in a convenient location. When running the Cluster Profile Manager, other users can import your profile by clicking **Import**.

To learn how to distribute a generic cluster profile and plugin scripts for others to use, see “Distribute a Generic Cluster Profile and Plugin Scripts” on page 3-53.

Special Configurations

Depending on your cluster architecture, you might need to perform additional tasks before you connect to your generic scheduler.

Custom MPI builds

You can use an MPI build that differs from the one provided with Parallel Computing Toolbox. For more information about using this option with the generic scheduler interface, see “Use Different MPI Builds on UNIX Systems” on page 2-4.

Run Communicating Jobs with the Grid Engine Family

The sample scripts for Grid Engine family rely on the presence of a `matlab` parallel environment. Parallel environments (PE) are programming environments designed for parallel computing in clusters. To run communicating jobs with MATLAB Parallel Server and a Grid Engine family cluster, you must establish a `matlab` parallel environment.

Create the Parallel Environment

The following steps create the parallel environment, and then make it runnable on all queues. As a best practice, perform these steps on the head node of your cluster. Some steps require administrator access.

1. Download and run the installer for Grid Engine from Grid Engine family.
2. Navigate to the location of the relevant plugin scripts for your submission mode in the installation folder.
3. Modify the contents of `matlabpe.template` to use the number of slots you want and the correct location of the `startmatlabpe.sh` and `stopmatlabpe.sh` files. These files can exist in a shared location accessible by all hosts, or you can copy them to the same location on each host. You can also change other values or add additional values to `matlabpe.template` to suit your cluster. For more information, refer to the `sge_pe` documentation provided with your scheduler.
4. Add the `matlab` parallel environment, using a shell command such as:

   ```
   qconf -Ap matlabpe.template
   ```
5. Make the `matlab` parallel environment runnable on all queues:

   ```
   qconf -mq all.q
   ```

   This command brings up a text editor for you to make changes. Search for the line `pe_list`, and add `matlab`.
6. Ensure you can submit a trivial job to the PE:

```
$ echo "hostname" | qsub -pe matlab 1
```

7. Use `qstat` to check that the job runs correctly, and check that the output file contains the name of the host that ran the job. The default file name for the output file is `~/STDIN.o###`, where ### is the Grid Engine job number.

Note If you change the name of the parallel environment to something other than `matlab`, also change the submit functions.

Configure Firewalls on Windows Cluster

If you are using Windows firewalls on your cluster nodes, you can add MATLAB as an allowed program.

In the following instructions, `matlabroot` refers to the MATLAB installation location.

1. Log in as a user with administrative privileges.
2. Execute the following script in a Windows command prompt:

```
matlabroot\toolbox\parallel\bin\addMatlabToWindowsFirewall.bat
```

If you are using other firewalls, you must configure these separately to add MATLAB as an allowed program.

See Also

Related Examples

- “Plugin Scripts for Generic Schedulers” (Parallel Computing Toolbox)
- “Distribute a Generic Cluster Profile and Plugin Scripts” on page 3-53
- “Use Different MPI Builds on UNIX Systems” on page 2-4
- “Configure for Third-Party Scheduler Cluster Discovery” on page 3-68

Distribute a Generic Cluster Profile and Plugin Scripts

You can distribute a Generic cluster profile and plugin scripts for others to use. If necessary, you can create a Generic cluster profile and plugin scripts as follows:

1. Download the plugin scripts from the GitHub repository appropriate for your third-party scheduler. For more details, see “Support Scripts” on page 3-45.
2. Use the Generic Profile Wizard to create a Generic cluster profile with the default MATLAB plugin scripts.

Decide How Users Access the Plugin Scripts

The **PluginScriptsLocation** property of your Generic cluster profile specifies the folder containing the plugin scripts that your cluster profile uses to submit MATLAB jobs to the cluster. Other users must have access to these plugin scripts or a copy of them in order to submit jobs to the cluster. As the one distributing a Generic cluster profile and plugin scripts, you must decide how other users will access these scripts.

- If you prefer to put the plugin scripts in a read-only shared location, follow the steps in “Shared PluginScriptsLocation Folder” on page 3-53. This option simplifies subsequent steps and allows any changes you make to the plugin scripts to take effect immediately for all users.
- If you prefer to give other users their own copy of your plugin scripts, follow the steps in “Distribute Copies of the PluginScriptsLocation Folder” on page 3-54.

Shared PluginScriptsLocation Folder

If other users have read access to the **PluginScriptsLocation** folder specified in your cluster profile, they need only a copy of your profile to submit MATLAB jobs to the cluster.

Note If you have moved your `PluginScriptsLocation` folder to a shared location, remember to update the **PluginScriptsLocation** property of your cluster profile before continuing.

- To distribute a copy of your profile, you must:
  1. Open MATLAB and navigate to **Home** > **Parallel** > **Create and Manage Clusters** to open the Cluster Profile Manager.
  2. Select your profile in the list and click **Export**.
  3. Choose a name for the `.settings` file, which contains your profile, and click **Save**.
  4. Send a copy of the `.settings` file to other users.
- To import your profile, other users must:
  1. Save the `.settings` file to a location of their choice.
  2. Open MATLAB and navigate to **Home** > **Parallel** > **Create and Manage Clusters**.
  3. Open the Cluster Profile Manager and click **Import**.
  4. Select the `.settings` file and click **Open**. A copy of the profile appears in their profile list.
  5. Check that the profile works by selecting the profile in the Cluster Profile Manager and clicking **Validate**.

Distribute Copies of the PluginScriptsLocation Folder

If you cannot or prefer not to put your plugin scripts in a shared location, you can give users a copy of your plugin scripts in addition to your profile.

Note If you make changes to your plugin scripts, you will have to distribute copies of your updated plugin scripts for the changes to take effect for other users.

- To distribute a copy of your profile and plugin scripts, you must:

  1. Open MATLAB and navigate to **Home** > **Parallel** > **Create and Manage Clusters** to open the Cluster Profile Manager.
  2. Select your profile in the list and click **Export**.
  3. Choose a name for the `.settings` file, which contains the exported profile, and click **Save**.
  4. Send other users a copy of

     - The `.settings` file.
     - The **PluginScriptsLocation** folder and all files therein.
- To import your profile and plugin scripts, other users must:

  1. Save all files to a location of their choice.
  2. Open MATLAB and navigate to **Home** > **Parallel** > **Create and Manage Clusters**.
  3. Open the Cluster Profile Manager and click **Import**.
  4. Select the `.settings` file and click **Open**. A copy of the profile appears in their profile list.
  5. Select the profile in the Cluster Profile Manager and click **Edit**. Scroll down to the Scheduler Plugin section of the profile and change the **PluginScriptsLocation** property to point to their copy of the **PluginScriptsLocation** folder.
  6. Check that the profile works by selecting the profile in the Cluster Profile Manager and clicking **Validate**.

Further Considerations for Clusters with Shared File Systems

If you are distributing a Generic cluster profile with the **HasSharedFileSystem** property set to true (for example, if your cluster shares a filesystem with MATLAB clients, see "Interface with Third-Party Schedulers" on page 3-45), the cluster machines must have read and write access to the folder specified in the **JobStorageLocation** property of the profile. Remind other users receiving the profile that they must either:

1. Set the **JobStorageLocation** property of their profile to a shared location, preferably one that is unique to their user name and MATLAB version.
2. Only create or submit jobs to the cluster when the current working folder is a shared location.

See Also

"Interface with Third-Party Schedulers" on page 3-45 | "Support Scripts" on page 3-45

Customize Behavior of Sample Plugin Scripts

When using the generic scheduler interface, you can modify the behavior of the plugin scripts by setting additional properties for a generic cluster profile or object using `AdditionalProperties`. For more information on the generic scheduler interface, see “Configure Using the Generic Scheduler Interface” on page 3-45.

The sample plugin scripts allow you to set these properties.

Properties for All Schedulers

Property	Description	Type
`AdditionalSubmitArgs`	Additional scheduler arguments for job submission. The sample plugin scripts add the value of this property to the scheduler submission string.	String
`AuthenticationMode`	Option to indicate how you are authenticated when you connect to the cluster, specified as one of the following: • `"Agent"` – the client interfaces with an SSH agent running on the client machine. • `"IdentityFile"` – the client uses the identity file specified by the `IdentityFile` additional property. • `"Multifactor"` – the client to prompts you for input one or more times. For example, if two-factor authentication (2FA) is enabled on the client, the client requests your password and a response for the second authentication factor. • `"Password"` – the client prompts you for your SSH password. Your user name is specified by the `Username` additional property.	String String array
`ClusterHost`	Host name of the cluster machine that has the scheduler utilities to submit jobs. Use this if your cluster is unable to directly submit jobs to the scheduler. The cluster machine must run Linux.	String
`IdentityFile`	Location on the client machine of the SSH identity file that identifies you in `ClusterHost`.	String String array

Property	**Description**	**Type**
`IdentityFileHasPassphrase`	Set this property to `true` if `IdentityFile` requires a passphrase.	Logical
`RemoteJobStorageLocation`	Location to store job files on the cluster. Use this property if your client and the cluster nodes do not have a shared file system.	String
`UseIdentityFile`	Option to use an identity file. Set this property to `true` if you want to use an SSH identity file to log in to `ClusterHost`. If you set this property, then also set `IdentityFile` and `IdentityFileHasPassphrase`.	Logical
`Username`	User name to log in to `ClusterHost` with.	String
`UseUniqueSubfolders`	Option to use unique subfolders. Set this property to `true` if you want MATLAB to store job files under different subfolders based on the user name and MATLAB version. Doing so helps to prevent conflicts between jobs submitted from different users and MATLAB versions.	Logical

Properties for Slurm Only

For more information about these properties, see the Slurm documentation.

Property	**Description**	**Type**
`AccountName`	Provide an account name to which the scheduler can charge for the resources used.	String
`Constraint`	Specify the node features required to run the job.	String
`MemPerCPU`	Specify the minimum memory required per CPU.	String
`Partition`	Specify a partition for the job's resource allocation.	String
`RequireExclusiveNode`	Option to share nodes with other running jobs. Set this property to `true` if the job cannot share nodes with other running jobs.	Logical

Property	Description	Type
`Reservation`	Specify a reservation name from which the scheduler can allocate resources for the job.	String
`WallTime`	Specify a limit on the total run time of the job.	String
`EmailAddress`	Provide an email address to receive notifications from the scheduler.	String

Properties for LSF Only

For more information about these properties, see the LSF documentation.

Property	Description	Type
`MemPerCPU`	Specify the minimum memory required per CPU.	String
`Project`	Specify a project to which the scheduler must assign the job.	String
`QueueName`	Specify a cluster queue name on which the scheduler can run the job.	String
`RequireExclusiveNode`	Option to share nodes with other running jobs. Set this property to `true` if the job cannot share nodes with other running jobs.	Logical
`WallTime`	Specify a limit on the total run time of the job.	String
`EmailAddress`	Provide an email address to receive notifications from the scheduler.	String

Properties for Grid Engine Family Only

Property	Description	Type
`AccountName`	Specify an account name under which the scheduler must run for the job.	String
`MemPerCPU`	Specify the minimum memory required per CPU.	String
`Project`	Specify a project to which the scheduler must assign the job.	String
`QueueName`	Specify a cluster queue name to run the job.	String

Property	Description	Type
`WallTime`	Specify a limit on the total run time of the job.	String
`EmailAddress`	Provide an email address to receive notifications from the scheduler.	String

Properties for PBS Only

Property	Description	Type
`AccountName`	Provide an account name to which the scheduler can charge for the resources used.	String
`Priority`	Specify the priority of the job relative to other jobs in the cluster.	Double

Property	Description	Type
`ProcsPerNode`	Number of processors per node, specified as a finite positive integer scalar. When you submit a job to the cluster, the number of cores per node that MATLAB requests is guaranteed to be less than or equal to `ProcsPerNode`. Set `ProcsPerNode` equal to the maximum number of processors you want MATLAB to request from each cluster node. MATLAB requests the smallest number of cores per node required to run the job. • If the `NumThreads` property of the cluster is less than or equal to `ProcsPerNode`, MATLAB requests `NumThreads` processors per worker, then maximizes the number of workers per node. For example if `ProcsPerNode` is `16` and `NumThreads` is `5`, MATLAB requests `15` cores, the largest multiple of `5` that is less than `ProcsPerNode`. • If `NumThreads` of the cluster is greater than `ProcsPerNode`, MATLAB requests `ProcsPerNode` processors per node. When `NumThreads` is greater than `ProcsPerNode`, you might encounter performance issues. As a best practice, set `NumThreads` less than or equal to `ProcsPerNode`. For more information, see “Edit Number of Workers and Cluster Settings” (Parallel Computing Toolbox).	Positive integer scalar
`QueueName`	Specify a cluster queue or server name to run the job.	String

Property	Description	Type
`WallTime`	Specify a limit on the total run time of the job.	String
`EmailAddress`	Provide an email address to receive notifications from the scheduler.	String

Properties for HTCondor Only

For more information about these properties, see the HTCondor documentation.

Properties	Description	Type
`AccountingGroup`	Specify an accounting group name under which the scheduler must negotiate the job.	String
`AccountingGroupUsername`	Provide a username for resource usage accounting.	String
`RequestDisk`	Specify the disk space requirements for the job.	String
`RequestMemory`	Specify the memory requirements for the job.	String
`Priority`	Specify the priority of the job relative to other jobs in the cluster.	Double
`Requirements`	Specify the node features required to run the job.	String
`EmailAddress`	Provide an email address to receive notifications from the scheduler.	String

When you create a generic cluster profile for the first time with the Generic Profile Wizard, the installer guides you through setting some of these properties. For more information on the installer, see “Interface with Third-Party Schedulers” on page 3-45.

You can also define your own properties to customize the behavior of the plugin scripts. To do so, you must modify the plugin scripts. For more information, see “Adding User Customization” (Parallel Computing Toolbox).

Set Additional Properties

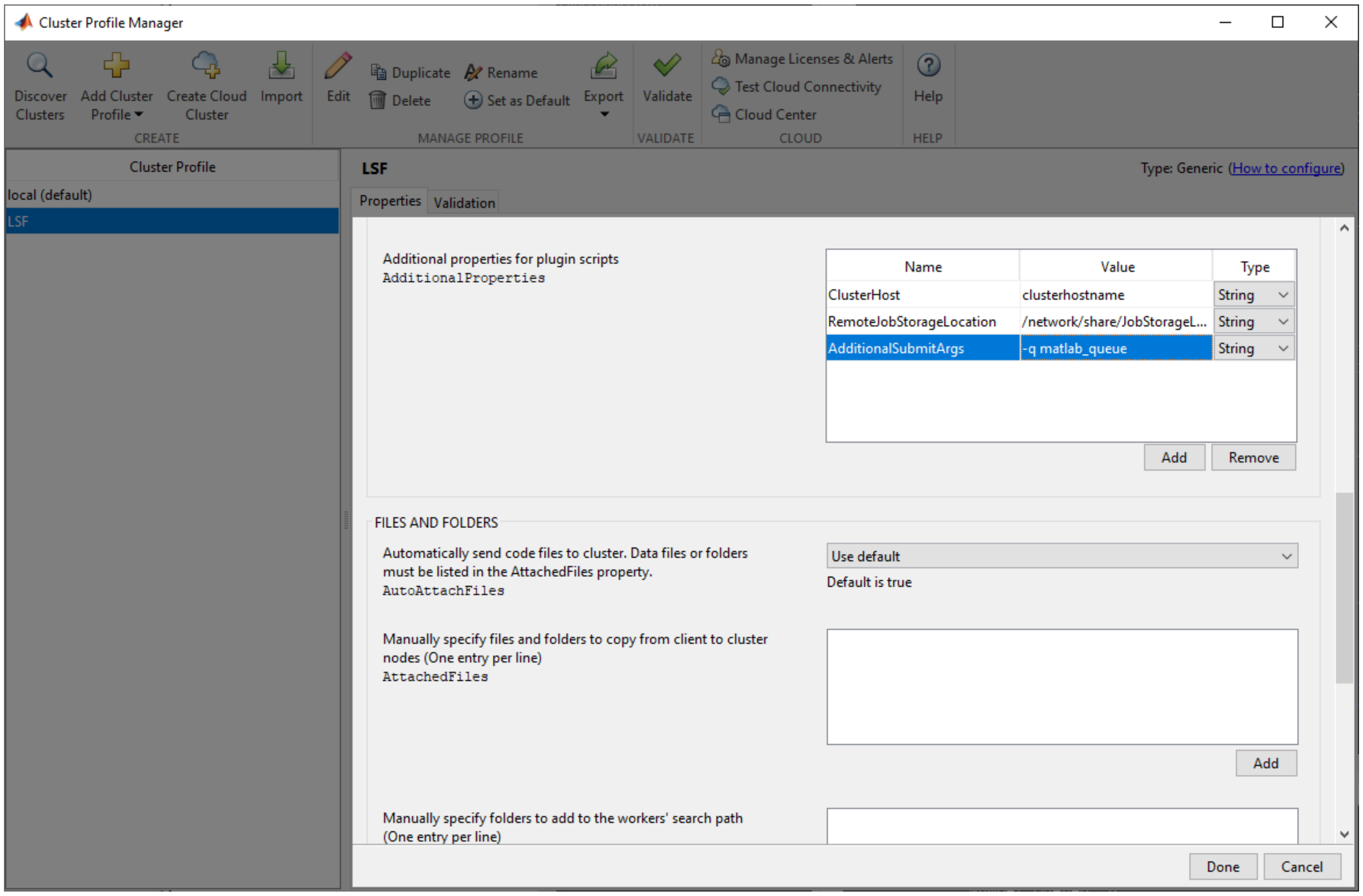
You can set additional properties in the cluster profile, with the Cluster Profile Manager, or programmatically.

Set Properties in Cluster Profile

When you set the additional properties in the cluster profile, the properties apply every time you use the cluster.

- In the MATLAB toolstrip, on the **Home** tab, in the **Environment** area, select **Parallel** > **Create and Manage Clusters**.
- In the Cluster Profile Manager, click on the generic profile that you want to modify.
- Click **Edit** at the bottom-right.
- Go to the `AdditionalProperties` table.
- To add a new property, click **Add**. To modify an existing property, click on the property.

The following image shows an example of setting `AdditionalSubmitArgs` for an LSF cluster profile.



Additional properties for plugin scripts
AdditionalProperties

Name	Value	Type
ClusterHost	clusterhostname	String
RemoteJobStorageLocation	/network/share/JobStorageL...	String
AdditionalSubmitArgs	-q matlab_queue	String

FILES AND FOLDERS

Automatically send code files to cluster. Data files or folders must be listed in the AttachedFiles property.
AutoAttachFiles: Use default
Default is true

Manually specify files and folders to copy from client to cluster nodes (One entry per line)
AttachedFiles

Manually specify folders to add to the workers' search path (One entry per line)

Set Properties Programmatically

You can set additional properties programmatically by accessing the `AdditionalProperties` of a generic cluster object. Note that this action does not update the cluster profile and the properties only apply to that particular cluster object. The following is an example of how to set `AdditionalSubmitArgs` for an LSF cluster.

First, create a cluster object by using the `parcluster` function. In the following code, change `MyLSFCluster` to the name of your cluster profile.

```
c = parcluster("MyLSFCluster");
```

Next, set `AdditionalSubmitArgs` so that the plugin scripts use a different job queue.

```
c.AdditionalProperties.AdditionalSubmitArgs = '-q matlab_queue';
```

With this change, MATLAB passes the additional arguments to the scheduler when you submit a job. For example, submit a batch job.

```
job = batch(c,"myScript");
```

See Also

Related Examples

- "Configure Using the Generic Scheduler Interface" on page 3-45
- "Plugin Scripts for Generic Schedulers" (Parallel Computing Toolbox)

Configure a Hadoop Cluster

In this section...

"Cluster Configuration" on page 3-64

"Client Configuration" on page 3-64

"Kerberos Authentication" on page 3-64

"Hadoop Version Support" on page 3-65

Parallel MATLAB code that contains `tall` (MATLAB) arrays and `mapreduce` (MATLAB) functions can be submitted to the Hadoop cluster from suitably configured MATLAB clients.

To configure the client to run MATLAB code on the cluster, you must already be able to submit to the cluster from the intended client machine. The client machine must have a Hadoop installation that can access the cluster outside of MATLAB.

Many Hadoop distributions do not support direct access of Linux based clusters from Windows clients. Users of Windows clients typically need to set up a Linux gateway node that can be accessed from the Windows client via SSH or VNC. The cluster can then be accessed from this gateway node.

Cluster Configuration

1. Integrate MATLAB Parallel Server with your cluster infrastructure. For instructions, see "Install and Configure MATLAB Parallel Server for Third-Party Schedulers" on page 3-12.
2. If your cluster requires Kerberos authentication, ensure your MATLAB Parallel Server installations have been configured correctly. For instructions, see "Kerberos Authentication" on page 3-64.

Client Configuration

1. Ensure your client can access the Hadoop cluster outside MATLAB.
2. Ensure your client MATLAB installation has been configured for Kerberos authentication if your cluster requires it. For instructions, see "Kerberos Authentication" on page 3-64.

To access the cluster from within MATLAB, set up a `parallel.cluster.Hadoop` (Parallel Computing Toolbox) object using the following statements.

```
setenv('HADOOP_HOME', '/path/to/hadoop/install')
cluster = parallel.cluster.Hadoop;
```

Use `mapreducer` (MATLAB) to specify `mapreduce` to run on the Hadoop cluster object.

For examples of how to run parallel MATLAB code on your Hadoop cluster, see "Run mapreduce on a Hadoop Cluster" (Parallel Computing Toolbox) and "Use Tall Arrays on a Spark Cluster" (Parallel Computing Toolbox).

Kerberos Authentication

If the cluster uses Kerberos authentication that requires the Oracle® Java Cryptography Extension, you must configure all installations of MATLAB and MATLAB Parallel Server. If you are using

Hortonworks® or Cloudera® distributions, it is likely that you need to complete these configuration steps.

The configuration instructions are the same for client and worker MATLAB installations.

Starting in R2018b, configure your MATLAB installation by enabling the appropriate security policy in the Java installation.

1. In the MATLAB Editor, open the file `${MATLAB_ROOT}/sys/java/jre/${ARCH}/jre/lib/security/java.security`.
2. Change the line

   ```
   #crypto.policy=unlimited
   ```

   to

   ```
   crypto.policy=unlimited
   ```

For previous releases, you must download additional security files from Oracle.

1. Download the Oracle Java Cryptography Extension zip file from the Oracle Java SE page.
2. Unzip the downloaded zip file into a temporary folder.
3. Replace the files `local_policy.jar` and `US_export_policy.jar` in the folder `${MATLABROOT}/sys/java/jre/${ARCH}/jre/lib/security` with the downloaded versions.

Hadoop Version Support

- MATLAB `mapreduce` is supported on Hadoop 2.x clusters. Note that support for Hadoop 1.x clusters has been removed.
- MATLAB tall arrays are supported on Spark™ enabled Hadoop 2.x clusters. You can use tall arrays on Spark enabled Hadoop clusters supporting all architectures for the client, while supporting Linux and Mac architectures for the cluster. This includes cross-platform support.

Functionality	Result	Use Instead	Compatibility Considerations
Support for running MATLAB `mapreduce` on Hadoop 1.x clusters has been removed.	Errors	Use clusters that have Hadoop 2.x installed to run MATLAB `mapreduce`.	Migrate MATLAB `mapreduce` code that runs on Hadoop 1.x to Hadoop 2.x.

See Also

`parallel.cluster.Hadoop`

Related Examples

- "Install and Configure MATLAB Parallel Server for Third-Party Schedulers" on page 3-12
- "Use Tall Arrays on a Spark Cluster" (Parallel Computing Toolbox)
- "Run mapreduce on a Hadoop Cluster" (Parallel Computing Toolbox)
- "Read and Analyze Hadoop Sequence File" (MATLAB)

Configure a Spark Cluster

Submit parallel MATLAB code that contains `tall` (MATLAB) arrays and `mapreduce` (MATLAB) functions to a Spark cluster from suitably configured MATLAB clients.

To configure the client to run MATLAB code on the cluster, you must already be able to submit to the cluster from the intended client machine. The client machine must have a Spark installation that can access the cluster outside of MATLAB.

Many Spark distributions do not support direct access of Linux based clusters from Windows clients. Users of Windows clients typically need to set up a Linux gateway node that can be accessed from the Windows client via SSH or VNC. The cluster can then be accessed from this gateway node.

Cluster Configuration

1. Integrate MATLAB Parallel Server with your cluster infrastructure. For instructions, see “Install and Configure MATLAB Parallel Server for Third-Party Schedulers” on page 3-12.

   MATLAB Parallel Server supports Spark clusters running in different environments such as Spark Standalone or Databricks.
2. If your cluster requires Kerberos authentication, ensure your MATLAB Parallel Server installation have been configured correctly. For instructions, see “Kerberos Authentication” on page 3-66.

Client Configuration

1. Ensure your client can access the Spark cluster outside MATLAB.
2. Ensure your client MATLAB installation has been configured for Kerberos authentication if your cluster requires it. For instructions, see “Kerberos Authentication” on page 3-66.

To access the cluster from within MATLAB, set up a `parallel.cluster.Spark` object using the following statements.

```
cluster = parallel.cluster.Spark('/path/to/spark/install');
```

Use `mapreducer` (MATLAB) to specify `mapreduce` to run on the Spark cluster object.

For examples of how to run parallel MATLAB code on your Spark cluster, see “Use Tall Arrays on a Spark Cluster” (Parallel Computing Toolbox).

Kerberos Authentication

If the cluster uses Kerberos authentication that requires the Oracle Java Cryptography Extension, you must configure all installations of MATLAB and MATLAB Parallel Server. If you are using Spark on Hadoop, for example Cloudera distributions, it is likely that you need to complete these configuration steps.

The configuration instructions are the same for client and worker MATLAB installations.

Starting in R2018b, configure your MATLAB installation by enabling the appropriate security policy in the Java installation.

1. In the MATLAB Editor, open the file `${MATLAB_ROOT}/sys/java/jre/${ARCH}/jre/lib/security/java.security`.
2. Change the line

   ```
   #crypto.policy=unlimited
   ```

   to

   ```
   crypto.policy=unlimited
   ```

Spark Version Support

Spark 2.2 or later supports MATLAB `mapreduce`, tall arrays and parallel usage of datastores. For the client, you can use tall arrays on Spark clusters supporting all architectures, while supporting Linux and Mac architectures for the cluster. This includes cross-platform support.

See Also
`parallel.cluster.Spark`

Related Examples
- "Install and Configure MATLAB Parallel Server for Third-Party Schedulers" on page 3-12
- "Use Tall Arrays on a Spark Cluster" (Parallel Computing Toolbox)

Configure for Third-Party Scheduler Cluster Discovery

You can let MATLAB locate your third-party scheduler cluster configured with the Generic scheduler interface. To discover clusters using the Parallel Computing Toolbox cluster discovery functionality, you must create a cluster configuration file and store it at a location accessible to MATLAB users.

Prerequisites

To run jobs on your cluster, you must provide the MATLAB client with a set of plugin scripts. The scripts contain instructions specific to your cluster infrastructure, such as how to interface with the job scheduler, and how to transfer job and task data to cluster nodes.

Download the plugin scripts from the GitHub repository appropriate for your third-party scheduler. You must save the plugin repository at a location MATLAB clients can access. For more details, see “Support Scripts” on page 3-45.

Cluster Configuration File Format

The cluster configuration file is a plain text file with the extension `.conf` containing key-value pairs that describe the cluster configuration information. The MATLAB client will use the cluster configuration file to construct a cluster profile for the user who discovers the cluster. Cluster profiles let you define certain properties for your cluster, then have these properties applied when you create cluster, job, and task objects in the MATLAB client.

The cluster configuration file must conform to these specifications:

- The configuration file must have the extension `.conf`
- You must specify each cluster property name and value in the *`Name = Value`* format.

  ```
  Name = InstallTest
  ```
- You must specify values for the cluster `Name`, `Numworkers`, `JobStorageLocation`, and `PluginScriptsLocation` properties.
- To add comments to the configuration file, you must start the line with the hash (`#`) symbol.
- You can provide properties as structures. The structure field name must be the property name and the field value must specify the property value.
  - You can specify structures using dot notation of the form *`structName.fieldName = fieldValue`*. For example, you can specify a value for the `Username` property in the `AdditionalProperties` structure.

    ```
    AdditionalProperties.Username = myUserName
    ```
  - Alternatively, you can use square brackets to indicate that the subsequent name-value pairs belong to the same structure.

    ```
    [AdditionalProperties]
    Username = myUserName
    ClusterHost = cluster-host-name
    ```
  - If you add the `AdditionalProperties` structure to the configuration file, MATLAB converts the structure to the `parallel.cluster` property `parallel.cluster.AdditionalProperties`.

- You can specify which operating system applies to a property value. Add one or more of the specifiers `"Windows"`, `"Mac"`, `"Linux"`, or `"Unix"` in parenthesis to the property value. MATLAB will only execute the property value on the specified platform. For example, you can set different job storage locations for the client computer for Windows or UNIX operating systems.

  ```
  JobStorageLocation (Windows) = C:\Temp\joblocation
  JobStorageLocation (Unix) = /home/Temp/joblocation
  ```
- Specify operating system environmental variable names as `"$VARIABLE"`. MATLAB uses `getenv("VARIABLE")` to find the values of the environmental variable. For example, you can use environmental variable to set different usernames for Windows, macOS, or Linux operating systems.

  ```
  [AdditionalProperties]
  Username (Windows) = "$USERNAME"
  Username (Mac, Linux) = "$USER"
  ClusterHost = cluster-host-name
  ```

  Note The variable `"$MATLAB_VERSION_STRING"` returns the release number of the MATLAB client, e.g. 2022a. Add an "R" or "r" as appropriate to complete the MATLAB version.

  ```
  ClusterMATLABRoot = /network/installs/MATLAB/R"$MATLAB_VERSION_STRING"
  ```
- MATLAB converts textual representations of numbers to type double and true and false to type logical.
- MATLAB supports cell arrays and string arrays, for example: `{'hello', 'world'} or ["hello", "world"]`.

Store Configuration File

Store the cluster configuration file at a location the MATLAB client of users who want to discover the cluster can find. MATLAB will search for the configuration file in these locations:

- *matlabroot*`/toolbox/parallel/user/clusterprofiles` – You can use this location if users share a MATLAB installation on the cluster.
- The folder defined by the environment variable `$MATLAB_CLUSTER_PROFILES_LOCATION` – This can be useful if the cluster administrator can set environment variables on the organisation's machines when they are imaged.
- The user's home folder – `%USERPROFILE%` on Windows and `$HOME` on UNIX.
- The user's **Downloads** folder in their home folder – This can be useful if users can download the configuration file from their organisation's intranet.
- Any locations defined by Domain Name System (DNS) text records listed under `_mdcs._tcp.<domain>` – This can be useful if none of the above options are possible, and is a convenient workflow of discovering a cluster because no changes are required to the user's machine.

  A DNS text (TXT) record associates a text string with a particular domain. Your system administrator creates DNS TXT records in your organization's DNS infrastructure.

  The DNS TXT record must have the form `"discover_folder=<`*`folder`*`>"` where `<`*`folder`*`>` is an accessible location on the filesystem.

MATLAB will discover configuration files if you store it in `<folder>`. You can store multiple configuration files in `<folder>` and multiple DNS TXT records can be registered `under_mdcs._tcp`. For a description of the required record, and validation information, see "DNS TXT Record" on page 2-42.

Discover Clusters on Users Machine

To locate a cluster from a MATLAB client, use **Discover Clusters**. For more information, see "Discover Clusters" (Parallel Computing Toolbox).

See Also

Related Examples

- "Configure Using the Generic Scheduler Interface" on page 3-45
- "Plugin Scripts for Generic Schedulers" (Parallel Computing Toolbox)
- "Discover Clusters and Use Cluster Profiles" (Parallel Computing Toolbox)

Admin Center

- "Start Admin Center" on page 4-2
- "Set Up Resources" on page 4-3
- "Test Connectivity" on page 4-9
- "Export and Import Sessions" on page 4-12
- "Prepare for Cluster Profiles" on page 4-13

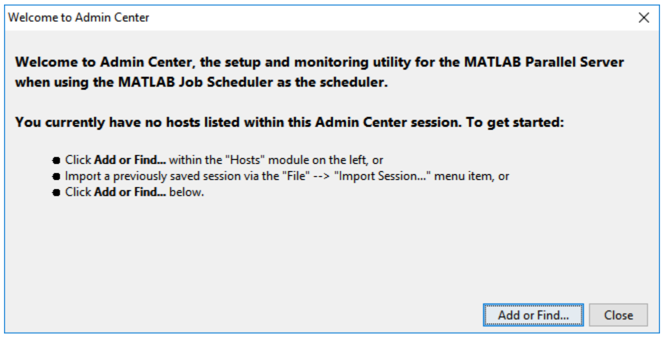
Start Admin Center

Admin Center is a graphical user interface with which you can control and monitor the MATLAB Parallel Server processes of a MATLAB Job Scheduler cluster. Admin center does not support any common job scheduler (CJS) clusters or third-party schedulers.

You start Admin Center outside a MATLAB session by executing the following:

- `*matlabroot*/toolbox/parallel/bin/admincenter` (on UNIX operating systems)
- `*matlabroot*\toolbox\parallel\bin\admincenter.bat` (on Microsoft Windows operating systems)

The first time you start Admin Center, you see the following welcome dialog box.



Welcome to Admin Center

Welcome to Admin Center, the setup and monitoring utility for the MATLAB Parallel Server when using the MATLAB Job Scheduler as the scheduler.

You currently have no hosts listed within this Admin Center session. To get started:

- Click **Add or Find...** within the "Hosts" module on the left, or
- Import a previously saved session via the "File" --> "Import Session..." menu item, or
- Click **Add or Find...** below.

A new session of Admin Center has no cluster hosts listed, so the usual first step is to identify the hosts you want to include in your listing. To do this, click **Add or Find**. Further information continues in the next section, “Set Up Resources” on page 4-3.

If you start Admin Center again on the same host, your previous session for that machine is loaded; and unless the update rate is set to `never`, Admin Center performs an update immediately for the listed hosts and processes. To clear this information and start a new session, select the pull-down **File > New Session**.

See Also

More About

- “Set Up Resources” on page 4-3
- “Export and Import Sessions” on page 4-12

Set Up Resources

In this section…

“Add Hosts” on page 4-3
“Start mjs Service” on page 4-4
“Start a MATLAB Job Scheduler” on page 4-5
“Start Workers” on page 4-6
“Stop, Destroy, Resume, Restart Processes” on page 4-7
“Move a Worker” on page 4-7
“Update the Display” on page 4-8

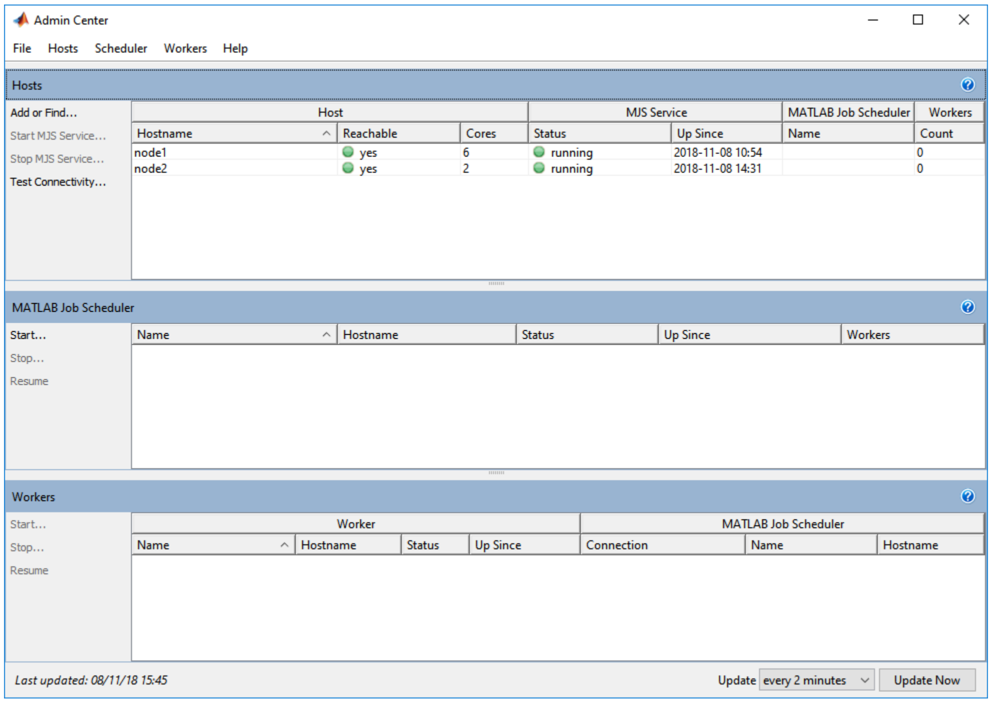
Add Hosts

To specify the hosts you want listed in Admin Center, click **Add or Find** in the Welcome dialog box, or if this is not a new session, click **Add or Find** in the Hosts module.

In the Add or Find Hosts dialog box, identify the hosts you want to add to the listing, by one of the following methods:

- Select **Enter Hostnames** and provide short host names, fully qualified domain names, or individual IP addresses for the hosts.
- Select **Enter IP Range** and provide the range of IP addresses for your hosts.

If one of the hosts you have specified is running a MATLAB Job Scheduler, Admin Center automatically finds and lists all the hosts running workers registered with that MATLAB Job Scheduler. Similarly, if you specify a host that is running a worker, Admin Center finds and lists the host running that worker’s MATLAB Job Scheduler, and then also all hosts running other workers under that MATLAB Job Scheduler.



Host			MJS Service		MATLAB Job Scheduler	Workers
Hostname	Reachable	Cores	Status	Up Since	Name	Count
node1	● yes	6	● running	2018-11-08 10:54		0
node2	● yes	2	● running	2018-11-08 14:31		0

Name	Hostname	Status	Up Since	Workers

Worker				MATLAB Job Scheduler		
Name	Hostname	Status	Up Since	Connection	Name	Hostname

Last updated: 08/11/18 15:45

Start mjs Service

A host must be running the mjs service if a MATLAB Job Scheduler or worker is to run on that host. Normally, you set this up with Admin Center or command-line scripts during the installation of MATLAB Parallel Server on your cluster, as described in the installation instructions available at "Install and Configure MATLAB Parallel Server for MATLAB Job Scheduler and Network License Manager" on page 3-16.

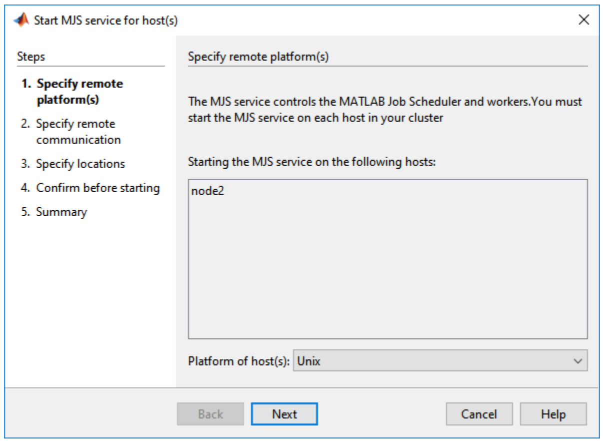
If you want to add or remove hosts to your cluster, Admin Center allows you to start and stop the mjs service on those hosts. To start the mjs service on a group of hosts with the same platform, select all those hosts in the Hosts module, and click **Start MJS Service** in the left column of the panel.

Alternative methods for starting mjs include selecting the pull-down **Hosts** > **Start MJS Service**, or right-clicking a listed host and selecting, **Start MJS Service**.

A dialog box leads you through the procedure of starting the mjs service on the selected hosts. There are five steps to the procedure in which you provide or confirm information for the service:

1. Specify remote platform — Windows or UNIX. You can start mjs on multiple hosts at the same time, but they all must be the same platform. If you have a mixed platform cluster, run the mjs startup separately for each type of platform.
2. Specify remote communication — Choose the protocol for communication with the hosts.
3. Specify locations — Specify the location of the MATLAB installation and the `mjs_def` file for the hosts.
4. Confirm before starting — Review information before proceeding.
5. Summary — Status about the startup attempt.

The dialog box looks like this for the first step:


Start MJS service for host(s)
Steps
1. Specify remote platform(s)
2. Specify remote communication
3. Specify locations
4. Confirm before starting
5. Summary
Back
Next
Cancel
Help

Specify remote platform(s)

The MJS service controls the MATLAB Job Scheduler and workers.You must start the MJS service on each host in your cluster

Starting the MJS service on the following hosts:

node2

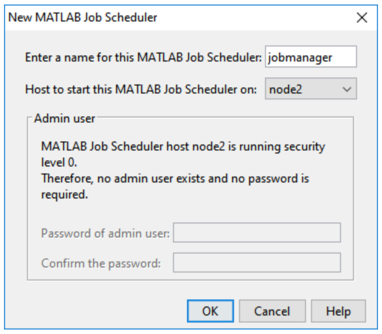
Platform of host(s): Unix

At each step, you can click **Help** to read detailed information about that step.

Start a MATLAB Job Scheduler

To start a MATLAB Job Scheduler, click **Start** in the MATLAB Job Scheduler module.

In the New MATLAB Job Scheduler dialog box, provide a name for the MATLAB Job Scheduler, and select a host to run it on.



Enter a name for this MATLAB Job Scheduler: jobmanager

Host to start this MATLAB Job Scheduler on: node2

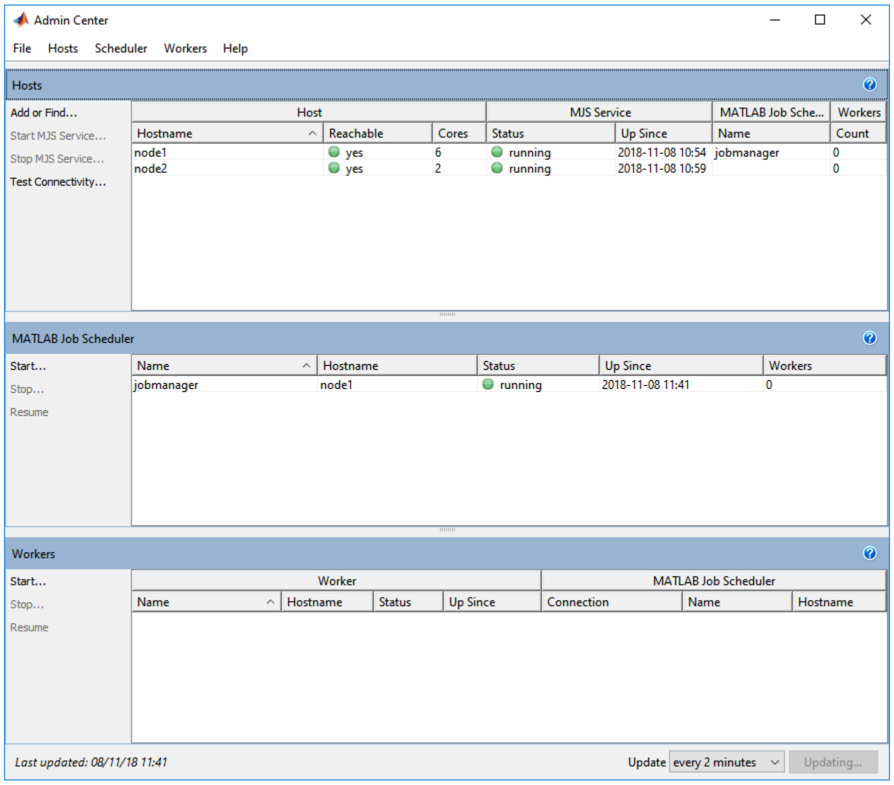
MATLAB Job Scheduler host node2 is running security level 0.
Therefore, no admin user exists and no password is required.

Password of admin user:

Confirm the password:

Alternative methods for starting a MATLAB Job Scheduler include selecting the pull-down **Scheduler** > **Start**, or right-clicking a listed host and selecting, **Start Scheduler**.

With a MATLAB Job Scheduler running on your cluster, Admin Center might look like the following figure, with the MATLAB Job Scheduler listed in the MATLAB Job Scheduler module, as well as being listed by name in the Hosts module in the line for the host on which it is running.



Hosts

Host			MJS Service		MATLAB Job Sche...	Workers
Hostname	Reachable	Cores	Status	Up Since	Name	Count
node1	● yes	6	● running	2018-11-08 10:54	jobmanager	0
node2	● yes	2	● running	2018-11-08 10:59		0

MATLAB Job Scheduler

Name	Hostname	Status	Up Since	Workers
jobmanager	node1	● running	2018-11-08 11:41	0

Workers

Worker				MATLAB Job Scheduler		
Name	Hostname	Status	Up Since	Connection	Name	Hostname

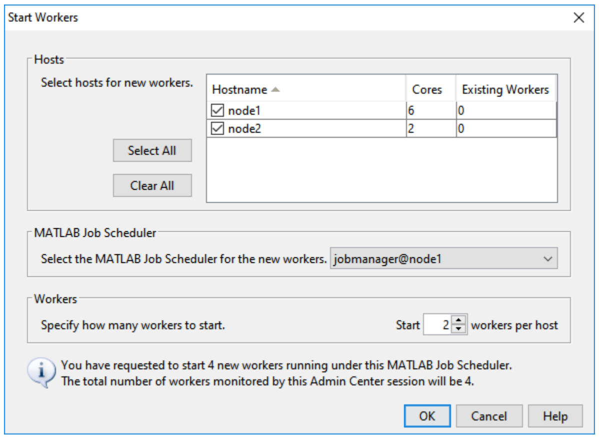
Last updated: 08/11/18 11:41

Update every 2 minutes

Start Workers

To start MATLAB workers, click **Start** in the Workers module.

In the Start Workers dialog box, specify the numbers of workers to start on each host, and select the hosts to run them. From the list, select the MATLAB Job Scheduler for these workers. Click **OK** to start the workers. Admin center automatically provides names for the workers, based on the hosts running them.



Hosts

Select hosts for new workers.

Hostname	Cores	Existing Workers
☑ node1	6	0
☑ node2	2	0

MATLAB Job Scheduler

Select the MATLAB Job Scheduler for the new workers. jobmanager@node1

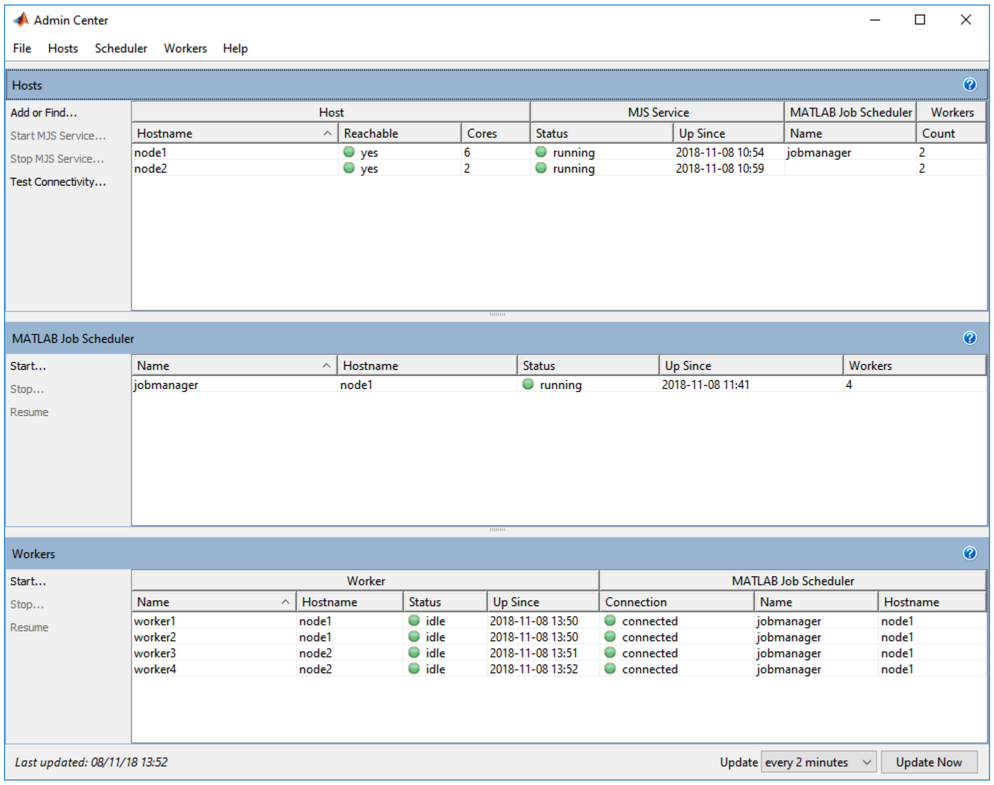
Workers

Specify how many workers to start. Start 2 workers per host

You have requested to start 4 new workers running under this MATLAB Job Scheduler.
The total number of workers monitored by this Admin Center session will be 4.

Alternative methods for starting workers include selecting the pull-down **Workers** > **Start**, or right-clicking a listed host or MATLAB Job Scheduler and selecting **Start Workers**.

With workers running on your cluster, Admin Center might look like the following figure, which shows the workers listed in the Workers module. Also, the number of workers running under the MATLAB Job Scheduler is listed in the MATLAB Job Scheduler module, and the number of workers for each MATLAB Job Scheduler is listed in the Hosts module.



Host			MJS Service		MATLAB Job Scheduler	Workers
Hostname	Reachable	Cores	Status	Up Since	Name	Count
node1	● yes	6	● running	2018-11-08 10:54	jobmanager	2
node2	● yes	2	● running	2018-11-08 10:59		2

Name	Hostname	Status	Up Since	Workers
jobmanager	node1	● running	2018-11-08 11:41	4

Worker				MATLAB Job Scheduler		
Name	Hostname	Status	Up Since	Connection	Name	Hostname
worker1	node1	● idle	2018-11-08 13:50	● connected	jobmanager	node1
worker2	node1	● idle	2018-11-08 13:50	● connected	jobmanager	node1
worker3	node2	● idle	2018-11-08 13:51	● connected	jobmanager	node1
worker4	node2	● idle	2018-11-08 13:52	● connected	jobmanager	node1

To get more information on any host, MATLAB Job Scheduler, or worker listed in Admin Center, right-click its name in the display and select **Properties**. Alternatively, you can find the **Properties** option under the **Hosts**, **Scheduler**, and **Workers** drop-down menus.

Stop, Destroy, Resume, Restart Processes

You can **Stop** or **Destroy** the mjs service, MATLAB Job Schedulers, and workers. The primary difference is that stopping a process shuts it down but retains its data; destroying a process shuts it down and clears its data. Use **Start MJS Service** to have mjs continue with existing data. Use **Resume** to have a MATLAB Job Scheduler or worker continue with its existing data. When you use **Restart**, a dialog box requires you to confirm your intention of starting a new process while keeping or discarding data.

Move a Worker

To move a worker from one host to another, you must completely shut it down, than start a new worker on the desired host:

1. Right-click the worker in the Workers module list.
2. Select **Destroy**. This shuts down the worker process and removes all its data.

3. If the old worker host is not running any other MATLAB Parallel Server processes (mjs service, MATLAB Job Scheduler, or workers), you might want to remove it from the Admin Center listing.
4. If necessary, add the new host to the Admin Center host listing.
5. In the Workers module, click **Start**. Select the desired host in the Start Workers dialog box, along with the appropriate number and MATLAB Job Scheduler name.

Use a similar process to move a MATLAB Job Scheduler from one host to another. Note, however, that all workers registered with the MATLAB Job Scheduler must be destroyed and started again, registering them with the new instance of the MATLAB Job Scheduler.

Update the Display

Admin Center updates its data automatically at regular intervals. To set the update rate, select an option from the **Update** list. Click **Update Now** to immediately update the display data.

See Also

More About

- "Start Admin Center" on page 4-2
- "Test Connectivity" on page 4-9

Test Connectivity

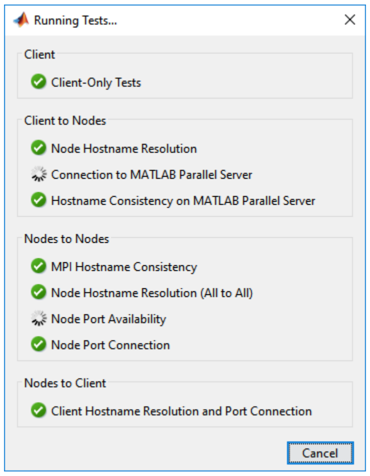
Admin Center lets you test communications between your MATLAB Job Scheduler node, worker nodes, and the node where Admin Center is running.

The tests are divided into four categories:

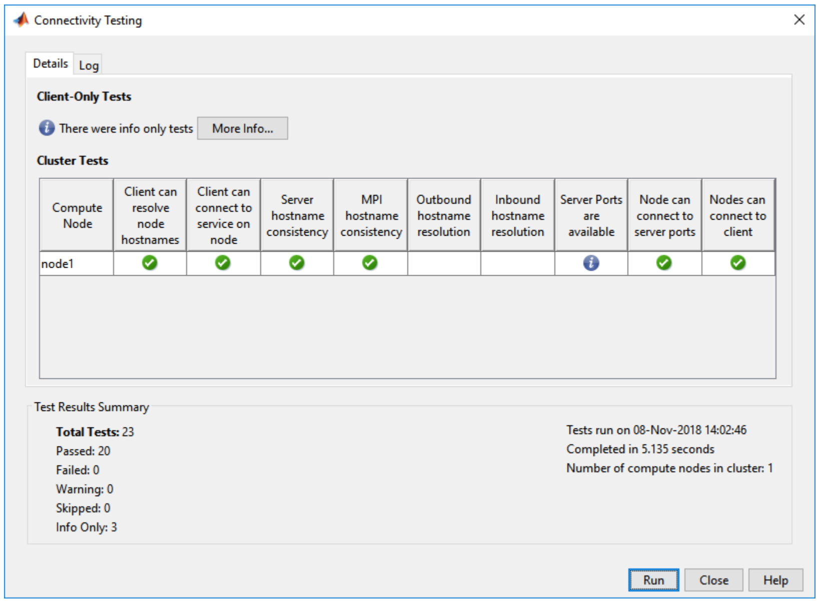
- **Client** — Verifies that the node running Admin Center is properly configured so that further cluster testing can proceed.
- **Client to Nodes** — Verifies that the node running Admin Center can identify and communicate with the other nodes in the cluster.
- **Nodes to Nodes** — Verifies that the other nodes in the cluster can identify each other, and that each node allows its mjs service to communicate with the mjs service on the other cluster nodes.
- **Nodes to Client** — Verifies that other cluster nodes can identify and communicate with the node running Admin Center.

First click **Test Connectivity** to open the Connectivity Testing dialog box. By default, the dialog box displays the results of the last test. To run new tests and update the display, click **Run**.

During test execution, Admin Center displays this progress dialog box.



When the tests are complete, the Running Tests dialog box automatically closes, and Admin Center displays the test results in the Connectivity Testing dialog box.


Connectivity Testing

Details | Log

Client-Only Tests

There were info only tests | More Info...

Cluster Tests

Compute Node	Client can resolve node hostnames	Client can connect to service on node	Server hostname consistency	MPI hostname consistency	Outbound hostname resolution	Inbound hostname resolution	Server Ports are available	Node can connect to server ports	Nodes can connect to client
node1	✓	✓	✓	✓			ⓘ	✓	✓

Test Results Summary

Total Tests: 23
Passed: 20
Failed: 0
Warning: 0
Skipped: 0
Info Only: 3

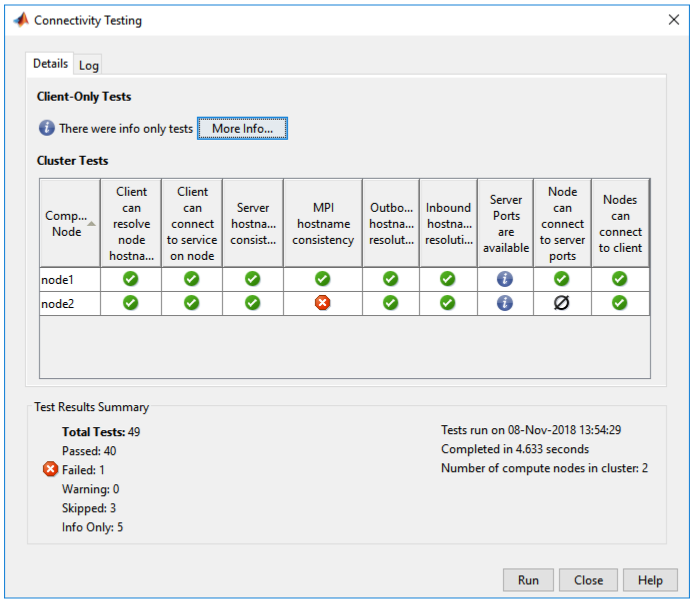
Tests run on 08-Nov-2018 14:02:46
Completed in 5.135 seconds
Number of compute nodes in cluster: 1

Run | Close | Help

The possible test result symbols are described in the following table.

Test Result	Description
✓	Test passed.
ⓘ	Test passed, extra information is available.
⚠	Test passed, but generated a warning.
✖	Test failed.
⌀	Test was skipped, possibly because prerequisite tests did not pass.

Test that include failures or other results might look like the following figure.



Client-Only Tests

ⓘ There were info only tests More Info...

Cluster Tests

Comp... Node	Client can resolve node hostna...	Client can connect to service on node	Server hostna... consist...	MPI hostname consistency	Outbo... hostna... resolut...	Inbound hostna... resoluti...	Server Ports are available	Node can connect to server ports	Nodes can connect to client
node1	✓	✓	✓	✓	✓	✓	ⓘ	✓	✓
node2	✓	✓	✓	✗	✓	✓	ⓘ	∅	✓

Test Results Summary

Total Tests: 49
Passed: 40
✗ Failed: 1
Warning: 0
Skipped: 3
Info Only: 5

Tests run on 08-Nov-2018 13:54:29
Completed in 4.633 seconds
Number of compute nodes in cluster: 2

Double-click any of the symbols in the test results to drill down for more detail. Use the **Log** tab to see the raw data from the tests.

The results of the tests that run on only the client are displayed in the lower-left corner of the dialog box. To drill into client-only test results, click **More Info**.

See Also

More About

- "Start Admin Center" on page 4-2
- "Export and Import Sessions" on page 4-12

Export and Import Sessions

By default, Admin Center saves the cluster definition, process status, and test results, so the next time the same user runs Admin Center on the same machine, that saved information is available and displayed by default. You can export session data so that a different user or a different host can access it, by selecting the pull-down **File** > **Export Session**. Browse to the location where you want to store the session data and provide a name for the file. Admin Center applies the extension `.mjs` to the file name.

You can import that saved session data into a subsequent session of Admin Center by selecting the pull-down **File** > **Import Session**. The imported data includes cluster definition and test results.

Note When importing a session file, Admin Center automatically sets its update rate to never (i.e., disabled), so that you can statically examine a cluster setup from the time the session was saved for evaluation or diagnostic purposes.

See Also

More About

- “Start Admin Center” on page 4-2
- “Prepare for Cluster Profiles” on page 4-13

Prepare for Cluster Profiles

Admin Center does not create cluster profiles, but the information displayed in Admin Center is of vital importance when you create your cluster profiles — information such as MATLAB Job Scheduler name, MATLAB Job Scheduler host, and number of workers. For more information about creating and using profiles, see “Discover Clusters and Use Cluster Profiles” (Parallel Computing Toolbox).

See Also

More About

- “Start Admin Center” on page 4-2
- “Discover Clusters and Use Cluster Profiles” (Parallel Computing Toolbox)
- “Benchmark Your Cluster with the HPC Challenge” (Parallel Computing Toolbox)
- “Specify Your Parallel Preferences” (Parallel Computing Toolbox)

Control Scripts

admincenter

Start Admin Center GUI

Syntax

```
admincenter
```

Description

`admincenter` opens the MATLAB Parallel Server Admin Center. When setting up or using a MATLAB job scheduler cluster, Admin Center allows you to establish and verify your cluster, and to diagnose possible problems.

For details about using Admin Center, see:

- “Start Admin Center” on page 4-2
- “Set Up Resources” on page 4-3
- “Test Connectivity” on page 4-9

See Also

`mjs` | `nodestatus`

createSharedSecret

Create shared secret for secure communication

Syntax

```
createSharedSecret
createSharedSecret -file <filename>
```

Description

`createSharedSecret` creates a shared secret file used for secure communication between job managers and workers. The file is named `secret` in the current folder.

`createSharedSecret -file <filename>` create a shared secret file as the given filename.

Before passing sensitive data from one service to another (e.g., between job manager and workers), these services need to establish a trust relationship using a shared secret. This script creates a file that serves as a shared secret between the services. Each service is trusted that has access to that secret file.

Create the secret file only once per cluster on one machine, then copy it into the location specified by `SHARED_SECRET_FILE` in the `mjs_def` file on each machine before starting any job managers or workers. In a shared file system, all nodes can point to the same file. Shared secrets can be reused in subsequent sessions.

Examples

Create a shared secret file in a central location for all the nodes of the cluster:

```
cd matlabInstallDir/toolbox/parallel/bin
createSharedSecret -file /share/secret
```

Then make sure that the nodes' shared or copied `mjs_def` files set the parameter `SHARED_SECRET_FILE` to `/share/secret` before starting the mjs service on each.

See Also

`mjs`

mjs

Install, start, stop, or uninstall mjs service

Syntax

```
mjs install
mjs uninstall
mjs start
mjs stop
mjs console
mjs restart
mjs ... -mjsdef <mjs_defaults_file>
mjs ... -clean
mjs status
mjs -version
mjs -useonlinelicensing
```

Description

The mjs service ensures that all other processes are running and that it is possible to communicate with them. Once the mjs service is running, you can use the `nodestatus` command to obtain information about the mjs service and all the processes it maintains.

The `mjs` executable resides in the folder *`matlabroot`*`\toolbox\parallel\bin` (Windows operating system) or *`matlabroot`*`/toolbox/parallel/bin` (UNIX operating system). Enter the following commands at a Windows or UNIX command-line prompt, respectively.

`mjs install` installs the mjs service in the Microsoft Windows Service Control Manager. This causes the service to automatically start when the Windows operating system boots up. The service must be installed before it is started.

`mjs uninstall` uninstalls the mjs service from the Windows Service Control Manager. Note that if you wish to install mjs service as a different user, you must first uninstall the service and then reinstall as the new user.

`mjs start` starts the mjs service. This creates the required logging and checkpointing directories, and then starts the service as specified in the mjs defaults file.

`mjs stop` stops running the mjs service. This automatically stops all job managers and workers on the computer, but leaves their checkpoint information intact so that they will start again when the mjs service is started again.

`mjs console` starts the mjs service as a process in the current terminal or command window rather than as a service running in the background.

`mjs restart` performs the equivalent of `mjs stop` followed by `mjs start`. This command is available only on UNIX operating systems.

`mjs ... -mjsdef <mjs_defaults_file>` uses the specified alternative mjs defaults file instead of the one found in *`matlabroot`*`/toolbox/parallel/bin`. This file is provided for Linux (`mjs_def.sh`) and Windows (`mjs_def.bat`). For information on backwards compatibility, see "Run Multiple MATLAB Parallel Server Versions" on page 3-29.

`mjs ... -clean` performs a complete cleanup of all service checkpoint and log files before installing or starting the service, or after stopping or uninstalling it. This deletes all information about any job managers or workers this service has ever maintained.

`mjs status` reports the status of the mjs service, indicating whether it is running and with what PID. Use `nodestatus` to obtain more detailed information about the mjs service. The `mjs status` command is available only on UNIX operating systems.

`mjs -version` prints version information of the mjs process to standard output, then exits.

`mjs -useonlinelicensing` ensures that workers use online licensing. Unless you specify –`useonlinelicensing`, `mjs` uses the network license manager.

See Also

`nodestatus` | `startjobmanager` | `startworker` | `stopjobmanager` | `stopworker`

nodestatus

Status of mjs processes running on node

Syntax

```
nodestatus
nodestatus -flags
```

Description

`nodestatus` displays the status of the mjs service and the processes which it maintains. The mjs service must already be running on the specified computer.

The `nodestatus` executable resides in the folder *`matlabroot`*`\toolbox\parallel\bin` (Windows operating system) or *`matlabroot`*`/toolbox/parallel/bin` (UNIX operating system). Enter the following command syntax at a Windows or UNIX command-line prompt, respectively.

`nodestatus -`*`flags`* accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
`-remotehost <hostname>`	Displays the status of the mjs service and the processes it maintains on the specified host. The default value is the local host.
`-infolevel <level>`	Specifies how much status information to report, using a level of 1-3. 1 means only the basic information, 3 means all information available. The default value is 1.
`-baseport <port_number>`	Specifies the base port that the mjs service on the remote host is using. You need to specify this only if the value of `BASE_PORT` in the local `mjs_def` file does not match the base port being used by the mjs service on the remote host.
`-v`	Verbose mode displays the progress of the command execution.
`-json`	View the output in JavaScript Object Notation (JSON) format. Output in `json` format is easy to parse.

Examples

Display basic information about the mjs processes on the local host.

```
nodestatus
```

Display detailed information about the status of the mjs processes on host `node27`.

```
nodestatus -remotehost node27 -infolevel 2
```

See Also

`mjs` | `startjobmanager` | `startworker` | `stopjobmanager` | `stopworker`

remotecopy

(Removed) Copy file or folder to or from one or more remote hosts using transport protocol

Note `remotecopy` has been removed. Use `scp` or `sftp` instead. For more information, see Version History.

Syntax

`remotecopy <flags> <protocol options>`

Description

`remotecopy <flags> <protocol options>` copies a file or folder to or from one or more remote hosts by using a transport protocol (such as ssh). Copying from multiple hosts creates a separate file per host, appending the hostname to the specified filename.

The following table describes the supported flags and options. You can combined multiple flags in the same command, preceding each flag by a dash (`-`).

Flags and Options	Operation
`-local <file-or-foldername>`	Specify the name of the file or folder on the local host.
`-remote <file-or-foldername>`	Specify the name of the file or folder on the remote host.
`-from`	Specify to copy from the remote hosts to the local host. You must use either the `-from` flag, or the `-to` flag.
`-to`	Specify to copy to the remote hosts from the local host. You must use either the `-from` flag, or the `-to` flag.
`-remotehost host1[,host2[,...]`	Specify the names of the hosts where you want to copy to or from. Separate the host names by commas without any white spaces. This is a mandatory argument.
`-remoteplatform { unix | windows }`	Specify the platform of the remote hosts. This option is required only if different from the local platform.
`-quiet`	Prevent `remotecopy` from prompting for missing information. The command fails if all required information is not specified.
`-help`	Print the help information for this command.

Flags and Options	Operation

`-protocol <type>`	Force the usage of a particular protocol type. Specifying a protocol type with all its required parameters also avoids interactive prompting and allows for use in scripts.
	
	The supported protocol types are `scp` and `sftp`.
	
	To get more information about one particular protocol type, enter
	
	`remotecopy -protocol <type> -help`
	
	For example:
	
	`remotecopy -protocol sftp -help`

`<protocol options>`	Specify particular options for the protocol type being used.

Note The file permissions on the copy might not be the same as the permissions on the original file.

Examples

Copy the local file `mjs_def.sh` to two other machines. (Enter this command on a single line.)

```
remotecopy -local mjs_def.sh -to
  -remote /matlab/toolbox/parallel/bin -remotehost hostA,hostB
```

Retrieve folders of the same name from two hosts to the local machine. (Enter command on a single line.)

```
remotecopy -local C:\temp\log -from -remote C:\temp\mjs\log
  -remotehost winHost1,winHost2
```

Version History

R2022b: remotecopy has been removed

Errors starting in R2022b

`remotecopy` is no longer supported.

- Previously you used `remotecopy` and `-protocol scp` to copy files to and from a remote host using the `scp` protocol.

  The following table details how to use `scp` instead.

Errors	Recommended
`remotecopy -remotehost host1 -local /my/file/path -to -remote /remote/file/path -protocol s`	`scp /my/file/path host1:/remote/file/path`

`remotecopy -remotehost host1,host2 -local /my/file/path -to -remote /remote/file/path -prot`	`scp /my/file/path host1:/remote/file/path`
	`scp /my/file/path host2:/remote/file/path`

Errors	Recommended
`remotecopy -remotehost host1 -local /my/file/path -from -remote /remote/file/path -protocol`	`scp host1:/remote/file/path /my/file/path`

- Previously you used `remotecopy` and `-protocol sftp` to copy files to and from a remote host using the `sftp` protocol.

  The following table describes how to use `sftp` instead.

Errors	Recommended
`remotecopy -remotehost host1 -local /my/file/path -to -remote /remote/file/path -protocol s`	`sftp /my/file/path host1:/remote/file/path`

`remotecopy -remotehost host1,host2 -local /my/file/path -to -remote /remote/file/path -prot`	`sftp /my/file/path host1:/remote/file/path`
	`sftp /my/file/path host2:/remote/file/path`

`remotecopy -remotehost host1 -local /my/file/path -from -remote /remote/file/path -protocol`	`sftp host1:/remote/file/path /my/file/path`

R2021b: `remotecopy` will be removed
Warns starting in R2021b

The `remotecopy` function issues a warning that it will be removed in a future release.

R2018b: `remotecopy` function will no longer support the `rcp` protocol in a future release
Warns starting in R2018b

The `rcp` protocol is insecure. The `remotecopy` function will no longer support `rcp` as an option to the `-protocol` flag in a future release. Instead, specify any other of the supported protocols, such as `scp`.

See Also
`mjs`

remotemjs

(Removed) Execute mjs command on one or more remote hosts by transport protocol

Note `remotemjs` has been removed. Use `ssh` instead. For more information, see Version History.

Syntax

`remotemjs` *`<mjs options> <flags> <protocol options>`*

Description

`remotemjs` *`<mjs options> <flags> <protocol options>`* allows you to execute the mjs service on one or more remote hosts.

For a description of the mjs service, see the `mjs` reference page.

The following table describes the supported flags and options. You can combined multiple flags in the same command, preceding each flag by a dash (`-`).

Flags and Options	**Operation**
`<mjs options>`	Options and arguments of the mjs command, such as `start`, `stop`, etc. See the `mjs` reference page for a full list.
`-matlabroot <installfoldername>`	The MATLAB installation folder on the remote hosts, required only if the remote installation folder differs from the one on the local machine.
`-remotehost host1[,host2[,...]`	The names of the hosts where you want to run the mjs command. Separate the host names by commas without any white spaces. This is a mandatory argument.
`-remoteplatform { unix | windows }`	The platform of the remote hosts. This option is required only if different from the local platform.
`-quiet`	Prevent mjs from prompting the user for missing information. The command fails if all required information is not specified.
`-help`	Print help information.

Flags and Options	Operation

`-protocol <type>`	Force the usage of a particular protocol type. Specifying a protocol type with all its required parameters also avoids interactive prompting and allows for use in scripts.
	
	The supported protocol types are `ssh` and `winsc`.
	
	To get more information about one particular protocol type, enter
	
	`remotemjs -protocol <type> -help`
	
	For example:
	
	`remotemjs -protocol winsc -help`
	
	Using the `winsc` protocol requires that you log in as a user with admin privileges on the remote host.

`<protocol options>`	Specify particular options for the protocol type being used.

Note If you are using OpenSSH on a Microsoft Windows operating system, you can encounter a problem when using backslashes in path names for your command options. In most cases, you can work around this problem by using forward slashes instead. For example, to specify the file `C:\temp\mjs_def.bat`, you should identify it as `C:/temp/mjs_def.bat`.

Examples

Start mjs on three remote machines of the same platform as the client:

```
remotemjs start -remotehost hostA,hostB,hostC
```

Start mjs in a clean state on two UNIX operating system machines from a Windows operating system machine, using the ssh protocol. Enter the following command on a single line:

```
remotemjs start -clean -matlabroot /usr/local/matlab
 -remotehost unixHost1,unixHost2 -remoteplatform UNIX
 -protocol ssh
```

Version History

R2022b: `remotemjs` has been removed
Errors starting in R2022b

`remotemjs` is no longer supported.

Previously you used `remotemjs` to run MATLAB Job Scheduler commands on a remote host using `-protocol ssh` or `-protocol winsc`.

The following table details how to use `ssh` instead.

Errors	Recommended
`remotemjs <mjs options> -matlabroot <installfoldername> -remotehost host1`	`ssh host1 <installfoldername>/toolbox/parallel/bin/mj`
`remotemjs <mjs options> -matlabroot <installfoldername> -remotehost host1,host2`	`ssh host1 <installfoldername>/toolbox/parallel/bin/mj` `ssh host2 <installfoldername>/toolbox/parallel/bin/mj`

R2021b: `remotemjs` will be removed
Warns starting in R2021b

`remotemjs` function issues a warning that it will be removed in a future release.

R2018b: `remotemjs` function will no longer support the `rsh` protocol in a future release
Warns starting in R2018b

The `rsh` protocol is insecure. The `remotemjs` function will no longer support `rsh` as an option to the `-protocol` flag in a future release. Instead, specify any other of the supported protocols, such as `ssh`.

See Also
`mjs`

pausejobmanager

Pause job manager process

Syntax

```
pausejobmanager
pausejobmanager -flags
```

Description

`pausejobmanager` pauses a job manager that is running under the mjs service.

The `pausejobmanager` executable resides in the folder *`matlabroot`*`\toolbox\parallel\bin` (Windows operating system) or *`matlabroot`*`/toolbox/parallel/bin` (UNIX operating system). Enter the following command syntax at a DOS or UNIX command-line prompt, respectively.

`pausejobmanager -`*`flags`* accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
`-name <job_manager_name>`	Specifies the name of the job manager to pause. The default is the value of `DEFAULT_JOB_MANAGER_NAME` parameter the `mjs_def` file.
`-remotehost <hostname>`	Specifies the name of the host where you want to pause the job manager. The default value is the local host.
`-baseport <port_number>`	Specifies the base port that the mjs service on the remote host is using. You need to specify this only if the value of `BASE_PORT` in the local `mjs_def` file does not match the base port being used by the mjs service on the remote host.
`-secretfile <path_to_shared_secret_file>`	Specifies the path to the shared secret file to use to authenticate the command. Use this flag to override the value of `SHARED_SECRET_FILE` in the local `mjs_def` file. If not specified, the command will attempt to use the path in the local `mjs_def` file or the default location: `$CHECKPOINTBASE/security/secret`
`-v`	Verbose mode displays the progress of the command execution.

Examples

Pause the job manager `MyJobManager` on the local host.

```
pausejobmanager -name MyJobManager
```

Pause the job manager `MyJobManager` on the host `JMHost`.

```
pausejobmanager -name MyJobManager -remotehost JMHost
```

See Also

`mjs` | `nodestatus` | `startjobmanager` | `stopjobmanager` | `resumejobmanager`

resumejobmanager

Resume job manager process

Syntax

```
resumejobmanager
resumejobmanager -flags
```

Description

`resumejobmanager` resumes a job manager that is running under the mjs service.

The `resumejobmanager` executable resides in the folder *`matlabroot`*`\toolbox\parallel\bin` (Windows operating system) or *`matlabroot`*`/toolbox/parallel/bin` (UNIX operating system). Enter the following command syntax at a Windows or UNIX command-line prompt, respectively.

`resumejobmanager -`*`flags`* accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
`-name <job_manager_name>`	Specifies the name of the job manager to resume. The default is the value of `DEFAULT_JOB_MANAGER_NAME` parameter the `mjs_def` file.
`-remotehost <hostname>`	Specifies the name of the host where you want to resume the job manager. The default value is the local host.
`-baseport <port_number>`	Specifies the base port that the mjs service on the remote host is using. You need to specify this only if the value of `BASE_PORT` in the local `mjs_def` file does not match the base port being used by the mjs service on the remote host.
`-secretfile <path_to_shared_secret_file>`	Specifies the path to the shared secret file to use to authenticate the command. Use this flag to override the value of `SHARED_SECRET_FILE` in the local `mjs_def` file. If not specified, the command will attempt to use the path in the local `mjs_def` file or the default location: `$CHECKPOINTBASE/security/secret`
`-v`	Verbose mode displays the progress of the command execution.

Examples

Resume the job manager `MyJobManager` on the local host.

```
resumejobmanager -name MyJobManager
```

Resume the job manager `MyJobManager` on the host `JMHost`.

```
resumejobmanager -name MyJobManager -remotehost JMHost
```

See Also

`mjs` | `nodestatus` | `startjobmanager` | `stopjobmanager` | `pausejobmanager`

startjobmanager

Start job manager process

Syntax

```
startjobmanager
startjobmanager -flags
```

Description

`startjobmanager` starts a job manager process and the associated job manager lookup process under the mjs service, which maintains them after that. The job manager handles the storage of jobs and the distribution of tasks contained in jobs to MATLAB workers that are registered with it. The mjs service must already be running on the specified computer.

The `startjobmanager` executable resides in the folder *`matlabroot`*`\toolbox\parallel\bin` (Windows operating system) or *`matlabroot`*`/toolbox/parallel/bin` (UNIX operating system). Enter the following command syntax at a Windows or UNIX command-line prompt, respectively.

`startjobmanager -`*`flags`* accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
`-name <job_manager_name>`	Specifies the name of the job manager. This identifies the job manager to MATLAB worker sessions and MATLAB clients. The default is the value of the `DEFAULT_JOB_MANAGER_NAME` parameter in the `mjs_def` file.
`-remotehost <hostname>`	Specifies the name of the host where you want to start the job manager and the job manager lookup process. If omitted, they start on the local host.
`-clean`	Deletes all checkpoint information stored on disk from previous instances of this job manager before starting. This cleans the job manager so that it initializes with no existing jobs or tasks.
`-baseport <port_number>`	Specifies the base port that the mjs service on the remote host is using. You need to specify this only if the value of `BASE_PORT` in the local `mjs_def` file does not match the base port being used by the mjs service on the remote host.
`-certificate <path_to_certificate_file>`	Specifies the path to the certificate file to use to connect to the job manager. You must use this flag when the `mjs_def` file of the job manager sets `REQUIRE_CLIENT_CERTIFICATE` to `true`.
`-useMSMPI`	Use Microsoft MPI (MS-MPI) for clusters on Windows platforms.

Flag	Operation
`-secretfile <path_to_shared_secret_file>`	Specifies the path to the shared secret file to use to authenticate the command. Use this flag to override the value of `SHARED_SECRET_FILE` in the local `mjs_def` file. If not specified, the command will attempt to use the path in the local `mjs_def` file or the default location: `$CHECKPOINTBASE/security/secret`
`-v`	Verbose mode displays the progress of the command execution.

Examples

Start the job manager `MyJobManager` on the local host.

```
startjobmanager -name MyJobManager
```

Start the job manager `MyJobManager` on the host `JMHost`.

```
startjobmanager -name MyJobManager -remotehost JMHost
```

See Also

`mjs` | `nodestatus` | `pausejobmanager` | `resumejobmanager` | `startworker` | `stopjobmanager` | `stopworker`

startworker

Start MATLAB worker session

Syntax

```
startworker
startworker -flags
```

Description

`startworker` starts a MATLAB worker process under the mjs service, which maintains it after that. The worker registers with the specified job manager, from which it will get tasks for evaluation. The mjs service must already be running on the specified computer.

The `startworker` executable resides in the folder *`matlabroot`*`\toolbox\parallel\bin` (Windows operating system) or *`matlabroot`*`/toolbox/parallel/bin` (UNIX operating system). Enter the following command syntax at a Windows or UNIX command-line prompt, respectively.

`startworker -`*`flags`* accepts the following input flags. Multiple flags can be used together on the same command, except where noted.

Flag	Operation
`-name <worker_name>`	Specifies the name of the MATLAB worker. The default is the value of the `DEFAULT_WORKER_NAME` parameter in the `mjs_def` file.
`-remotehost <hostname>`	Specifies the name of the computer where you want to start the MATLAB worker. If omitted, the worker is started on the local computer.
`-jobmanager <job_manager_name>`	Specifies the name of the job manager this MATLAB worker will receive tasks from. The default is the value of the `DEFAULT_JOB_MANAGER_NAME` parameter in the `mjs_def` file.
`-jobmanagerhost <job_manager_hostname>`	Specifies the host on which the job manager is running. The worker contacts the job manager lookup process on that host to register with the job manager. This overrides the setting of `JOB_MANAGER_HOST` in the `mjs_def` file on the worker computer. You must specify the job manager host by one of these means.
`-clean`	Deletes all checkpoint information associated with this worker name before starting.
`-num <num_workers>`	Specifies the number of workers to start. The default is one. If starting more than one worker, worker names will have '_1', '_2', and so on, appended to them.

Flag	Operation
`-baseport <port_number>`	Specifies the base port that the mjs service on the remote host is using. You only need to specify this if the value of `BASE_PORT` in the local `mjs_def` file does not match the base port being used by the mjs service on the remote host.
`-secretfile <path_to_shared_secret_file>`	Specifies the path to the shared secret file to use to authenticate the command. Use this flag to override the value of `SHARED_SECRET_FILE` in the local `mjs_def` file. If not specified, the command will attempt to use the path in the local `mjs_def` file or the default location: `$CHECKPOINTBASE/security/secret`
`-v`	Verbose mode displays the progress of the command execution.

Examples

Start a worker on the local host, using the default worker name, registering with the job manager `MyJobManager` on the host `JMHost`.

```
startworker -jobmanager MyJobManager -jobmanagerhost JMHost
```

Start a worker on the host `WorkerHost`, using the default worker name, and registering with the job manager `MyJobManager` on the host `JMHost`. (The following command should be entered on a single line.)

```
startworker -jobmanager MyJobManager -jobmanagerhost JMHost
               -remotehost WorkerHost
```

Start two workers, named `worker_1` and `worker_2`, on the host `WorkerHost`, registering with the job manager `MyJobManager` that is running on the host `JMHost`. (The following command should be entered on a single line.)

```
startworker -num 2 -name worker -remotehost WorkerHost
               -jobmanager MyJobManager -jobmanagerhost JMHost
```

See Also

`mjs` | `nodestatus` | `startjobmanager` | `stopjobmanager` | `stopworker`

stopjobmanager

Stop job manager process

Syntax

```
stopjobmanager
stopjobmanager -flags
```

Description

`stopjobmanager` stops a job manager that is running under the mjs service.

The `stopjobmanager` executable resides in the folder *`matlabroot`*`\toolbox\parallel\bin` (Windows operating system) or *`matlabroot`*`/toolbox/parallel/bin` (UNIX operating system). Enter the following command syntax at a Windows or UNIX command-line prompt, respectively.

`stopjobmanager -`*`flags`* accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
`-name <job_manager_name>`	Specifies the name of the job manager to stop. The default is the value of `DEFAULT_JOB_MANAGER_NAME` parameter the `mjs_def` file.
`-remotehost <hostname>`	Specifies the name of the host where you want to stop the job manager and the associated job manager lookup process. The default value is the local host.
`-clean`	Deletes all checkpoint information stored on disk for the current instance of this job manager after stopping it. This cleans the job manager of all its job and task data.
`-baseport <port_number>`	Specifies the base port that the mjs service on the remote host is using. You need to specify this only if the value of `BASE_PORT` in the local `mjs_def` file does not match the base port being used by the mjs service on the remote host.
`-secretfile <path_to_shared_secret_file>`	Specifies the path to the shared secret file to use to authenticate the command. Use this flag to override the value of `SHARED_SECRET_FILE` in the local `mjs_def` file. If not specified, the command will attempt to use the path in the local `mjs_def` file or the default location: `$CHECKPOINTBASE/security/secret`
`-v`	Verbose mode displays the progress of the command execution.

Examples

Stop the job manager `MyJobManager` on the local host.

```
stopjobmanager -name MyJobManager
```

Stop the job manager `MyJobManager` on the host `JMHost`.

```
stopjobmanager -name MyJobManager -remotehost JMHost
```

See Also

`mjs` | `nodestatus` | `startjobmanager` | `startworker` | `stopworker`

stopworker

Stop MATLAB worker session

Syntax

```
stopworker
stopworker -flags
```

Description

`stopworker` stops a MATLAB worker process that is running under the mjs service.

The `stopworker` executable resides in the folder *`matlabroot`*`\toolbox\parallel\bin` (Windows operating system) or *`matlabroot`*`/toolbox/parallel/bin` (UNIX operating system). Enter the following command syntax at a Windows or UNIX command-line prompt, respectively.

`stopworker -`*`flags`* accepts the following input flags. Multiple flags can be used together on the same command.

Flag	Operation
`-name <worker_name>`	Specifies the name of the MATLAB worker to stop. The default is the value of the `DEFAULT_WORKER_NAME` parameter in the `mjs_def` file.
`-remotehost <hostname>`	Specifies the name of the host where you want to stop the MATLAB worker. The default value is the local host.
`-clean`	Deletes all checkpoint information associated with this worker name after stopping it.
`-baseport <port_number>`	Specifies the base port that the mjs service on the remote host is using. You need to specify this only if the value of `BASE_PORT` in the local `mjs_def` file does not match the base port being used by the mjs service on the remote host.
`-onidle`	Stops the MATLAB worker after it has finished its current task. If the MATLAB worker is idle, stops the worker immediately.
`-secretfile <path_to_shared_secret_file>`	Specifies the path to the shared secret file to use to authenticate the command. Use this flag to override the value of `SHARED_SECRET_FILE` in the local `mjs_def` file. If not specified, the command will attempt to use the path in the local `mjs_def` file or the default location: `$CHECKPOINTBASE/security/secret`

Flag	Operation
`-v`	Verbose mode displays the progress of the command execution.

Examples

Stop the worker with the default name on the local host.

```
stopworker
```

Stop the worker with the default name, running on the computer `WorkerHost`.

```
stopworker -remotehost WorkerHost
```

Stop the workers named `worker1` and `worker2`, running on the computer `WorkerHost`.

```
stopworker -name worker1 -remotehost WorkerHost
stopworker -name worker2 -remotehost WorkerHost
```

See Also

`mjs` | `nodestatus` | `startjobmanager` | `startworker` | `stopjobmanager`

Reference Architectures

- "Run MATLAB Parallel Server on Amazon Web Services" on page 6-2
- "Run MATLAB Parallel Server on AWS Batch" on page 6-4
- "Run MATLAB Parallel Server on Microsoft Azure Using Reference Architecture" on page 6-5
- "Run MATLAB Parallel Server from Microsoft Azure Marketplace" on page 6-7
- "Configure MATLAB Parallel Server Licensing for Cloud Platforms" on page 6-17

Run MATLAB Parallel Server on Amazon Web Services

In this section...

"Requirements" on page 6-2
"Run from GitHub" on page 6-2

Use a customizable reference architecture to run MATLAB Parallel Server on Linux or Windows virtual machines in Amazon Web Services (AWS®).

Use this reference architecture when you want to launch a cluster of MATLAB workers in a specific region, combine your MATLAB resources with your existing cloud resources, or automate deployment.

For a simpler but less customizable method of launching a MATLAB Parallel Server cluster in AWS, try MathWorks Cloud Center.

Requirements

To use this reference architecture, you need:

- An AWS account. You are responsible for the costs of all AWS services.
- An SSH key pair for your AWS account in your chosen region. For more information, see Amazon EC2 Key Pairs.
- Parallel Computing Toolbox installed on a client machine. The version must match the MATLAB version of the reference architecture.
- A MATLAB Parallel Server license that meets the following conditions:
  - Linked to a MathWorks Account.
  - Configured for cloud use. Individual and Campus-Wide licenses are already configured. For other license types, contact your administrator. You can identify your license type and administrator by viewing your MathWorks Account. Administrators can consult "Configure MATLAB Parallel Server Licensing for Cloud Platforms" on page 6-17.

Run from GitHub

To launch MATLAB in AWS, use the reference architecture templates provided in the following GitHub repositories:

- MATLAB Parallel Server on AWS Linux VM (GitHub)
- MATLAB Parallel Server on AWS Windows VM (GitHub)

You can run the template directly from the link in the GitHub repositories. Choose your MATLAB release, and then click the **Launch Stack** icon next to the region in which you want to deploy your resources.

See Also

Related Examples

- “Run MATLAB Parallel Server on Microsoft Azure Using Reference Architecture” on page 6-5

External Websites

- MATLAB Parallel Server on AWS Linux VM (GitHub)
- MATLAB Parallel Server on AWS Windows VM (GitHub)

Run MATLAB Parallel Server on AWS Batch

Deploy MATLAB Parallel Server in Amazon Web Services (AWS), using AWS Batch, with your Amazon Web Services account.

Use this reference architecture when you want to batch process MATLAB jobs using AWS Batch. AWS Batch is a service that dynamically provisions, manages, monitors, and terminates Amazon EC2 instances based on the volume and resource requirements of the submitted jobs.

For a simpler but less customizable method of launching a MATLAB Parallel Server cluster in AWS, try MathWorks Cloud Center.

Requirements

To use this reference architecture, you need:

- An AWS account. You are responsible for the costs of all AWS services.
- Parallel Computing Toolbox installed on a client machine. The version must match the MATLAB version of the reference architecture.
- A MATLAB Parallel Server license that meets the following conditions:
  - Linked to a MathWorks Account.
  - Configured for cloud use. Individual and Campus-Wide licenses are already configured. For other license types, contact your administrator. You can identify your license type and administrator by viewing your MathWorks Account. Administrators can consult “Configure MATLAB Parallel Server Licensing for Cloud Platforms” on page 6-17.

Run from GitHub

To launch MATLAB Parallel Server in AWS using AWS Batch, use the reference architecture templates provided in the following GitHub repository:

- MATLAB Parallel Server with AWS Batch (GitHub)

You can run the template directly from the link in the GitHub repository. Choose your MATLAB release, and then click the **Launch Stack** icon next to the region in which you want to deploy your resources.

See Also

Related Examples

- “Run MATLAB Parallel Server on Amazon Web Services” on page 6-2

External Websites

- MATLAB Parallel Server with AWS Batch (GitHub)

Run MATLAB Parallel Server on Microsoft Azure Using Reference Architecture

In this section...

“Requirements” on page 6-5
“Run from GitHub” on page 6-5

Use a customizable reference architecture to run MATLAB Parallel Server on Windows virtual machines in Microsoft Azure.

Use this reference architecture when you want to launch a cluster of MATLAB workers in Microsoft Azure, combine your MATLAB resources with your existing cloud resources, or automate deployment.

For a simpler but less customizable method of launching a MATLAB Parallel Server cluster in Microsoft Azure, see “Run MATLAB Parallel Server from Microsoft Azure Marketplace” on page 6-7.

Requirements

To use this reference architecture, you need:

- A Microsoft Azure account. You are responsible for the costs of all Microsoft Azure services.
- Parallel Computing Toolbox installed on a client machine. The version must match the MATLAB version of the reference architecture.
- A MATLAB Parallel Server license that meets the following conditions:
  - Linked to a MathWorks Account.
  - Configured for cloud use. Individual and Campus-Wide licenses are already configured. For other license types, contact your administrator. You can identify your license type and administrator by viewing your MathWorks Account. Administrators can consult “Configure MATLAB Parallel Server Licensing for Cloud Platforms” on page 6-17.

Run from GitHub

To launch MATLAB in Azure, use the reference architecture templates provided in the following GitHub repository:

- MATLAB Parallel Server on Microsoft Azure (GitHub)

You can run the template directly from the link in the GitHub repository. Choose your MATLAB release, and then click the **Deploy to Azure** icon.

See Also

Related Examples

- “Run MATLAB Parallel Server on Amazon Web Services” on page 6-2
- “Run MATLAB Parallel Server from Microsoft Azure Marketplace” on page 6-7

External Websites

- MATLAB Parallel Server on Microsoft Azure (GitHub)

Run MATLAB Parallel Server from Microsoft Azure Marketplace

This topic describes how to use the Azure Marketplace to quickly deploy and run MATLAB Parallel Server in Azure. The elasticity of the cloud infrastructure combined with MATLAB Parallel Server enables you to leverage greater computing resources and keep your calculations close to your data.

Anyone with a valid MATLAB license that includes MATLAB Parallel Server and an Azure account can run MATLAB Parallel Server in the cloud. This offering is called *MATLAB Parallel Server (BYOL)* because you "Bring Your Own License."

Requirements

To complete these instructions, you need:

- A MATLAB License on page 6-7 that includes MATLAB Parallel Server
- A Microsoft Azure Account
- A working knowledge of Azure Resource Manager

Licensing

By default, the MATLAB Parallel Server (BYOL) offering in the Azure Marketplace uses online licensing. The following table summarizes which licenses support running MATLAB Parallel Server on the cloud using online licensing.

License Type	Cloud Availability
Individual (a license that only you use that is in your name; not a Home or Student license)	Your license is already configured for MATLAB Parallel Server in the cloud.
Campus-Wide License (a license that you use that belongs to your academic institution)	Your license is already configured for MATLAB Parallel Server in the cloud.
Home and Student (that only you use and is in your name)	These license types are not eligible to use MATLAB Parallel Server in the cloud.
All other license types, including Concurrent and Network Named User	Contact your license administrator.* The license administrator might need to make some changes to the license to enable you to run MATLAB Parallel Server in the cloud. **Administrators**: For more information on this process, see “Configure MATLAB Parallel Server Licensing for Cloud Platforms” on page 6-17.
*Not sure who the license administrator is? Sign in to your MathWorks Account, click the license you are using, then click the tab marked “Contact Administrators.”	

To use a license that is managed using a network license manager, you can specify the port and host name or IP address of the network license manager when you create the cloud resources in Azure. You can either specify the location of an existing license manager, or deploy a new license manager in the cloud. You must make sure that the license manager can communicate with your cloud resources. To deploy a Network Manager in Azure using an Azure Marketplace software plan developed by MathWorks, follow the steps at Run Network License Manager from Azure Marketplace.

Deploy MATLAB Parallel Server Resources in Azure

Configure and Deploy Template from Marketplace

Use the following steps to configure and deploy MATLAB Parallel Server (BYOL) resources in a resource group.

1 Navigate to https://portal.azure.com and log in to your Azure account.

2 From the Portal, click **Create a Resource**.

3 Search for “matlab parallel server” in the marketplace and select the **MATLAB Parallel Server (BYOL)** offering.

4 Click **Create** on the offering page to begin setup.

5 The setup process uses a Resource Manager template to help you configure the virtual machine (VM) and networking settings. To make setup easy, many of the fields come prepopulated with suitable values. The following table describes how to set the various options in each menu of the template. Click **Next** at the end of each step to proceed to the next menu.

Menu	Option	Description
Basics	Subscription	Select an Azure subscription to use.
	Resource group	You can select an existing resource group from the drop-down menu, or click **Create New** to create a new resource group. If you select an existing resource group, then it must not have any currently deployed resources.
	Region	Select a region from the drop-down list.
Cluster Settings	Cluster name	Select a name for the cluster. The default name is `myCluster`.
	Num worker nodes	The number of Azure instances to start for the workers to run on.

Menu	Option	Description
	Num workers per node	The number of MATLAB workers to start on each instance. Specify one worker for every two vCPUs to assign one worker per physical core. For example, a Standard_D64s_v3 instance has 64 vCPUs, so can support 32 MATLAB workers. For details on the number of vCPUs for each instance type, see https://learn.microsoft.com/azure/virtual-machines/sizes.
	Headnode instance type	The Azure instance type to use for the head node, which runs the job manager. No workers are started on this node, so this node can be a smaller instance type than the worker nodes. For a list of instance types and sizes, see https://learn.microsoft.com/azure/virtual-machines/sizes.
	Worker instance type	The Azure instance type to use for the workers. For a list of instance types and sizes, see https://learn.microsoft.com/azure/virtual-machines/sizes.
	Database volume size	The size of the volume in gigabytes used to store the database files. If you set this value to 0, a separate volume is not created and the root volume is used for the database.
	Client IP address	The IP address range that can be used to access the cluster from MATLAB. The range must be a valid IP CIDR range of the form x.x.x.x/x. Use a value of the form x.x.x.x/32 to restrict access to only your computer.

Menu	Option	Description
	Password	Choose the admin password for the user "matlab". This password is required when you log into any instance (headnode or worker) using remote desktop protocol. Your password must have at least 12 characters and meet the Azure password requirements. For information on the password requirements, see What are the password requirements when creating a VM?
	Confirm password	Retype your chosen password.
	Network license manager port@server	If you are using a network license manager, enter the port and hostname or IP address. Make sure that the network license manager can communicate with your Azure resources. To deploy a network license manager in Azure, see Run Network License Manager from Azure Marketplace.
Networking	Virtual network	The **Virtual network** field is prepopulated with a new virtual network resource named `vnet01`. • You can configure the name, address space, or subnets of the new virtual network resource by clicking **Create new**. • You can select an existing virtual network resource in the drop-down menu. If you choose to use an existing resource, then the template does not create any new virtual network resources.

Menu	Option	Description
	Subnets	The **Subnet** field is prepopulated with a new subnet named `subnet-1` and an associated subnet address. • You can select a different subnet of the selected virtual network in the drop-down menu. • If you selected an existing virtual network rather than creating a new one, then you can click **Manage subnet configuration** to configure the subnets for the network.

Menu	Option	Description
Review and Create	—	When you advance to the **Review and Create** menu, Azure automatically runs some final validation checks on the information entered on previous screens. If Azure finds any errors, then you must fix them before proceeding. After the validation completes successfully, review the MathWorks terms of use and privacy policy. You are responsible for all associated costs once you deploy MATLAB Parallel Server in Azure. Once you are satisfied with the values you have entered, click **Create** to finalize setup and begin deploying the selected resources, or click **Download a template for automation** to get a copy of the completed template. • The deployment can take several minutes. You get a notification in the Azure Portal once the deployment is complete. • When the deployment is complete, follow the instructions in “Connect to your Cluster from MATLAB” on page 6-12 to use the cluster you set up.

Connect to your Cluster from MATLAB

Once you have successfully configured and deployed the MATLAB Parallel Server (BYOL) resources, use the following steps to connect to the cluster you set up.

1. From the Azure Portal, navigate to the resource group with the resources you deployed.
2. Select the Storage Account ending with storage.
3. Select the Files container type.
4. Select the File Share named "shared."

5. Download the `.settings` file.
6. Open MATLAB on your client machine.
7. In the Parallel drop-down menu in the MATLAB toolstrip, select Create and Manage Clusters....
8. Click Import.
9. Select the downloaded profile and click open.
10. Click Set as Default.
11. (Optional) Validate your cluster by clicking the Validate button.

After you set the cloud cluster as the default, the next time you run a parallel language command (such as `parfor`, `spmd`, `parfeval`, or `batch`), MATLAB connects to the cluster. The first time you connect, you are prompted for your MathWorks account login. The first time you run a task on a worker, it can take several minutes for the worker MATLAB to start. This delay is due to provisioning the instance disk. This is a one-time operation, and subsequent tasks begin much faster.

Your cluster is now ready to use. The cluster remains running after you close MATLAB on the client machine.

Caution Use the profile and client IP address range to control access to your cloud resources. Anyone with the .settings file can connect to your resources from a machine within the specified IP address range and run jobs on it.

Port Requirements for Accessing MATLAB Parallel Server

To access a MATLAB Parallel Server cluster from your client MATLAB, your client machine must be able to communicate on specific ports. Make sure that the network firewall allows the following outgoing connections.

Required Ports	Description
TCP 27350 to 27358 + 4*N	Ports 27350 to 27358 + 4*N, where N is the maximum number of workers on a single node
TCP 443	HTTPS access to (at least) *.mathworks and *.microsoft.com
TCP 3389	Remote Desktop Protocol for access to cluster nodes

Delete MATLAB Parallel Server Resources from Azure

You can remove the resource group and all associated resources when you are done with them to help save on costs.

1. Log in to the Azure Portal.
2. Select the resource group containing the MATLAB Parallel Server (BYOL) resources you deployed.
3. Click the **Delete resource group** icon to delete all resources deployed in the group.
4. You are prompted to confirm the deletion.

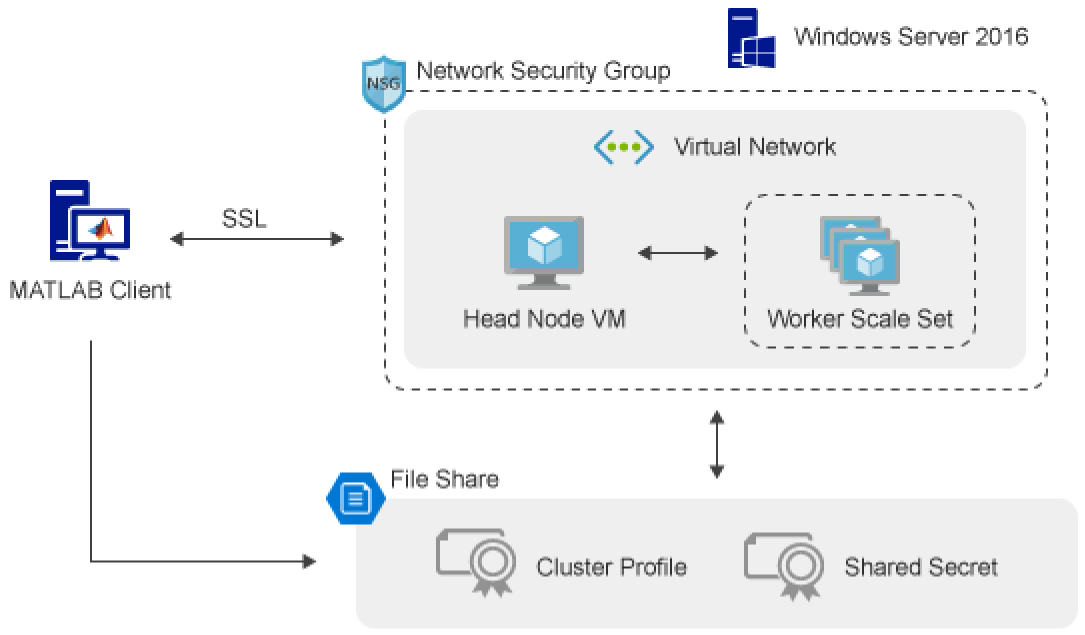
The deletion is final, and files do not persist between deployments. You must create the resources to make them available again.

Architecture and Resources for MATLAB Parallel Server in Azure

Deploying MATLAB Parallel Server in Azure creates several resources in your resource group. The deployment sets up a single Azure VM for the headnode, an Azure virtual machine scaling set (VMSS) for the workers, a network interface with a public IP address to connect to the VM, a network security group that controls network traffic, and a virtual network for communication between resources. The following table summarizes the resources that are created.

Resource Name	Default Resource Name in Azure	Description
Virtual machine	`myclus-headnode`	A compute instance for the cluster headnode. The MATLAB install is part of the VM image and the job database is stored either locally on the root volume. Optionally, a separate data disk can be used. Communication between clients and the headnode is secured using SSL.
Virtual machine scaling set	`mycl-vmss`	A scaling set for worker instances to be launched into. The scaling features are not currently used. The scaling set is configured to attach an extension to each instance that is configured at deployment time as a worker node of the cluster. Communication between clients and workers is secured using SSL.
Network interface	`myclus-headnodenic`	Enables the headnode and workers to communicate with the internet, Azure, and on-premises resources.
Network security group	`mycluster-nsg`	Allows or denies traffic to and from sources and destinations.
Virtual network	`vnet01`	Enables resources to communicate with each other. If you select an existing virtual network resource, then the template does not create this resource.

The architecture of these resources is summarized in the following diagram.



VM Software

A preconfigured Windows VM is provided to make deployment easy. The VM contains the following software:

- MATLAB, Simulink, toolboxes, and support for GPUs.

  The license used to run MATLAB determines which products and toolboxes are available for you to use. However, all products are installed on the VM, so the `ver` command returns a list of all products. To add new products to your license, see Products and Services.
- Add-Ons: Deep Learning Toolbox™ Model for AlexNet Network, Deep Learning Toolbox Model for GoogLeNet Network, and Deep Learning Toolbox Model for ResNet-50 Network.

Marketplace vs. Reference Architecture

The Azure Marketplace offering runs only the latest release of MATLAB Parallel Server. If you want to run older releases of MATLAB Parallel Server, or customize the templates and automation scripts more extensively, then you can use the MATLAB Parallel Server Reference Architecture for Microsoft Azure instead.

Support

If you encounter an unexpected problem, search MATLAB Answers for solutions first. Most likely, other people have run into the same problem and resolved it already.

If the problem persists, or there are not any relevant posts on MATLAB Answers for the issue, contact Technical Support.

See Also

External Websites

- Use MATLAB in the Cloud
- MATLAB Parallel Server Reference Architecture for Microsoft Azure

Configure MATLAB Parallel Server Licensing for Cloud Platforms

Before You Begin

This document shows administrators how to set up a license for use with the MATLAB Parallel Server cloud reference architectures. If you are an end user, contact your administrator.

To use the default MATLAB Parallel Server installation provided as part of a MathWorks reference architecture (Azure or AWS), you must be using release R2018a or later. The installation is configured to use online licensing by default. This license manager might differ from your current license configuration. Follow these instructions for setting up your license to use this installation.

If you have questions about any part of this license configuration process, contact MathWorks Cloud Support.

License Configuration Requirements

To set up your license to use MATLAB Parallel Server running in the cloud:

- You must be an administrator of a MATLAB Parallel Server license. If you do not have a server license, you can request a free trial or contact your MathWorks sales representative. If you are not sure that you are an administrator, or if you are not sure if you are associated with a MATLAB Parallel Server license, sign in to your MathWorks Account. Click the MATLAB Parallel Server license. If you see a “Manage Users” tab, you are an administrator.
- All licensed end users must be associated with the server license. (Instructions are included in this procedure.)
- The end user’s MATLAB license must include Parallel Computing Toolbox . The toolbox is required so that jobs can be submitted to the cluster running the server. End users must be using the same release as the cluster.

Select Licensing Configuration Option

From the following scenarios, select a procedure for setting up the license manager with online licensing. Not all procedures are required for all scenarios.

If you have a license setup scenario that is not listed here, Contact Support.

Scenario	Procedure
• You have a license for MATLAB Parallel Server, and • The license is already configured for online licensing.	“Add End Users to License” on page 6-18
You want to get a new on-demand license for MATLAB Parallel Server.	**1** Get an On-Demand License for MATLAB Parallel Server **2** “Add End Users to License” on page 6-18

Scenario	Procedure
• You just received a trial or an annual or perpetual license for MATLAB Parallel Server that has not yet been activated, and • The license or trial is configured for the network license manager.	1 "Change License Manager Type" on page 6-18 2 "Add End Users to License" on page 6-18
• You have an active annual or perpetual license for MATLAB Parallel Server, and • The license is configured for the network license manager, and • You are currently actively using that license in a cluster, and • You want to use online licensing	1 "Change License Manager Type" on page 6-18 2 "Add End Users to License" on page 6-18 3 "Disable Network License Manager for MATLAB Parallel Server" on page 6-19
• You have an active annual or perpetual license for MATLAB Parallel Server, and • The license is configured for the network license manager, and • You are currently actively using that license in a cluster, and • You want to use the network license manager running on the cloud.	1 Select one of the following: • Network License Manager for MATLAB on AWS • Network License Manager for MATLAB on Azure 2 "Add End Users to License" on page 6-18
• You are currently using the network license manager, and • You want to switch to the network license manager running on the cloud.	Select one of the following: • Network License Manager for MATLAB on AWS • Network License Manager for MATLAB on Azure

Change License Manager Type

1. Sign in to your MathWorks Account.
2. In your MathWorks Account, click the MATLAB Parallel Server license that you plan to use in the cloud.
3. Go to **Install and Activate**.
4. In the text box for **License Manager**, click the **edit** icon (a pencil).
5. Follow the steps shown and then click **Change to Online Licensing**.

To use the Network License Manager instead, repeat these steps to see instructions on switching back.

Add End Users to License

For online licensing to know which users have permission to check out a license, you must define a list of allowed users for that license. Allowed users must have a MathWorks Account associated with the on-demand license.

You can associate end users yourself or you can generate emails so that end users can associate themselves. Both actions are performed in License Center.

1. Sign in to the MathWorks Account that is associated with the license you want to use.
2. In your MathWorks Account, click on the MATLAB Parallel Server license.
3. Go to **Manage Users**.

   - Associate end users manually:

     a. Click **Add User**.
     b. Add an end user. Repeat this step for as many end users as you need to associate.
   - Invite end users to associate themselves:

   a. Click **Invite End Users to Add Themselves**.
   b. Enter email addresses in the template.

When you are finished, go to “Disable Network License Manager for MATLAB Parallel Server” on page 6-19.

Disable Network License Manager for MATLAB Parallel Server

To change your installed network license manager to online licensing, you must shut down the service and either delete the license (if the license manager was only for MATLAB Parallel Server) or modify the license (if license manager is managing other products in addition to the server).

If Network License Manager Manages Only MATLAB Parallel Server Cluster

Disable the network license manager, and delete the license.

Platform	Procedure
Windows	1 Start your current `LMtools.exe`. 2 Navigate to the **Start/Stop/Reread** tab and choose **Stop Server**. 3 Navigate to the **Config Services** tab and choose **Remove Service**. 4 Close the current `LMtools.exe`.
Linux	1 Run `lmdown`. 2 Delete the existing network license manager by deleting the entire folder where you previously placed the license manager binaries.

If Network License Manager Manages Multiple Products

Modify the license to remove MATLAB Parallel Server.

1. Shut down the network license manager.

Platform	Procedure
Windows	In your current LMtools.exe, navigate to the **Start/Stop/Reread** tab and choose **Stop Server**.
Linux	Run `lmdown`.

2 Open the MathWorks license file for editing:

```
matlabroot/etc/license.dat
```

The exact name of your license file and its location might vary. If you have trouble finding the license file, see Where are the license files for MATLAB located?

3 In the license file, locate and remove the entry that begins with the text:

```
INCREMENT MATLAB_Distrib_Comp_Engine ...
```

The entry might span more than one line in the file. Remove the entire entry and save the file.

4 Restart the network license manager for the license file change to take effect.

Network license manager utilities, such as `lmstat` – can verify that only the proper products are licensed. To use the `lmstat` function, open Terminal and run the following commands:

```
cd $MATLABROOT/etc/glnxa64
./lmutil lmstat -a
```

See Also

External Websites

- MATLAB Parallel Server on Amazon Web Services (GitHub)
- MATLAB Parallel Server on Microsoft Azure (GitHub)