

# Configuring JobServer and TaskServer for Clusters and HPC

## Contents

- [Introduction](#)
- [Before You Start](#)
- [Configuring a TaskServer to Use an External Queue](#)
- [SSH Tunnel access](#)

## 1 Introduction

With the three-tiered architecture of *MedeA* you can submit compute tasks to your local computer, workstations or remote compute clusters. The last is an ideal solution for compute-intensive tasks. As long as you have a working queueing system and a submission script in place, which you can use to run calculations on the remote compute cluster, it can be quite simple to leverage that via the TaskServer.

## 2 Before You Start

1. Install the JobServer and the TaskServer on a shared file system, visible to all compute nodes served by the queuing system. The most robust way is to install the entire *MD* directory tree there, which includes the *MD/Linux-x86\_64* or the *MDWindows-x86\_64* directories which contain libraries and executables for MPI and MKL. In many instances, similar libraries are already installed on your cluster.

**Warning:** *MedeA* GUI, JobServer, and TaskServer must be on the same *MedeA* version.

2. Double-check that the *working directory*, as set in the web interface of the TaskServer, is also on a shared file system.
3. Ensure fast and reliable communication between the nodes running the JobServer and the TaskServer

See [Installing MedeA on Windows with the Windows Installer](#), [Installing MedeA on Windows and Linux from ISO](#), and [JobServer and TaskServer Administration and Configuration](#) sections for more details on installing and configuring *MedeA*, the JobServer, and the TaskServer.

### 2.1 Install the JobServer and the TaskServer on the HPC

Please consult with the HPC IT team whether you are allowed to install the JobServer and the TaskServer as services or daemons running in the background. If not allowed, you can still install the JobServer and the TaskServer as programs that can be started manually. To start the the JobServer and the TaskServer manually run the *debugJobServer* and the *debugTaskServer* scripts in the *MD/JobServer* and *MD/TaskServer* directories, respectively.

**Warning:** The the JobServer and the TaskServer, albeit services/daemons or programs, must be running for the entire duration of the calculation.

## 2.2 Configure Permissions and Databases

Be sure every user running *MedeA* has read/write permissions to the *MD* directory on the shared drive, the Jobs directory, and the Tasks directory. See the JobServer and TaskServer Administration and Configuration page for information about setting these directories.

If the JobServer and TaskServer are running under a single admin account, permissions to the Jobs and Tasks directories need to be only given to the admin account user,

```
chmod -R 744 <jobs_path>/Jobs
chmod -R 744 <tasks_path>/Tasks
```

where *<jobs\_path>* and *<tasks\_path>* are the paths from the JobServer and TaskServer admin pages.

If a user is running the JobServer or TaskServer under their account, a group should be created for them and access to that group allowed for the Jobs and Tasks directories.

For the MD directory, it is recommended that all the users be added to a group, and that read/write access to be given to the group.

```
chmod -R 774 <md_path>/MD
```

where *<md\_path>* is the path to the installed *MD* directory.

**Warning:** Be sure these directories are not restricted by a size quota that could impede *MedeA* writing files to them.

If the *MD* directory is installed on an NFS files system, *MedeA* may not be able to write to SQLite database files. For more information on database files on network shared drives, please see the Databases section.

## 3 Configuring a TaskServer to Use an External Queue

For use with an external queuing system like PBS, LSF, GridEngine, etc, the TaskServer takes the role of a queue filter. Compute tasks are submitted to the queuing system that handles the computation in serial or parallel mode. Concepts and configuration of queuing systems are outside the scope of this document, as settings and procedures can vary between systems. We expect you to know how a specific queuing system works and what settings need to be redefined in the queue submission script.

*MedeA* provides a few template scripts that can be modified to suit your specific settings. Some flags like e.g. the number of processors to use, the queue type, and a project name can be set through the TaskServer administration interface. Other parameters may need to be set directly in the queue specific script by the user. This procedure is feasible for any type of queuing system. Below we give an example for the LSF queue and VASP 5.4.4.

### 3.1 Test a Queue Submission Script to be Used with the *MedeA* TaskServer

The first step is to let the TaskServer automatically create a queue submission script for a given task. To achieve this, do the following:

1. Create the queue setup script in *<Install\_dir>/MD/TaskServer/Tools/* by **copying** the file *<queue>\_template.tcl* to *<queue>.tcl*. Take the LSF queue as an example: *cp templateLSF.tcl LSF.tcl*

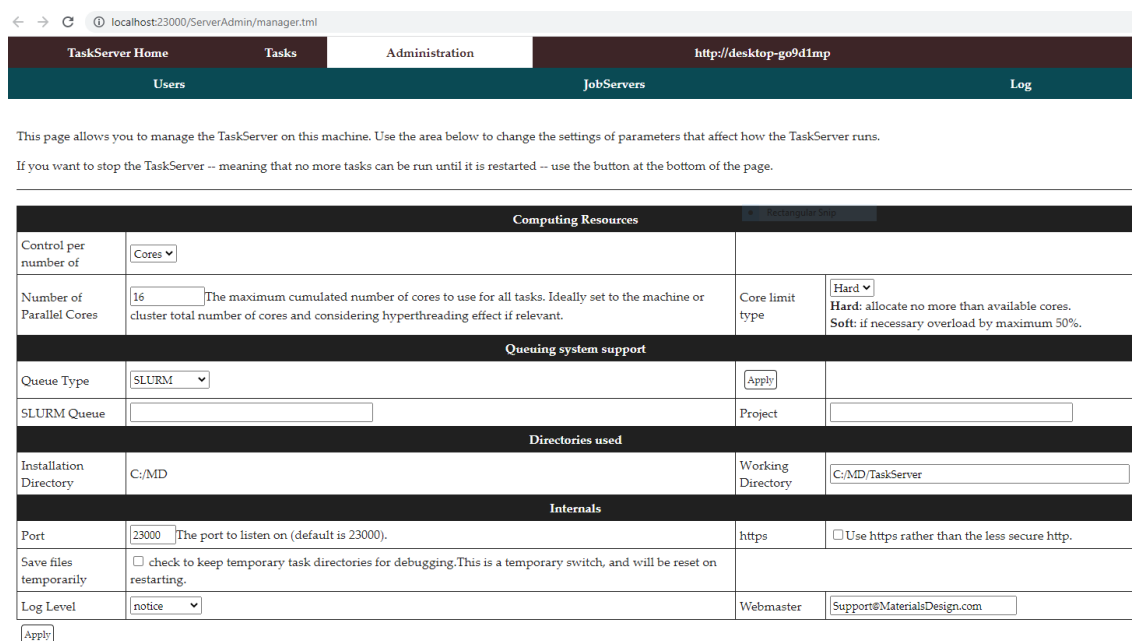
If you are working with a queuing system other than LSF, GridEngine, SLURM, or PBS (Torque/MOAB), you need to create a file `<your_queuing_system_name>.tcl` based on `template<your_queuing_system_name>.tcl`. You will need to modify this script to match the relevant queue submission commands on your machine. Note that the queue type is case sensitive, so the names in the two previous steps need to match exactly, `LSF.tcl` is not `lsf.tcl`.

2. Login to the TaskServer machine and on the TaskServer admin page

`http://localhost:23000/ServerAdmin/manager.tml`

**Note:** `localhost` above can be replaced with the hostname or IP address of your TaskServer

Set the Queue Type to e.g. LSF and confirm with **Apply**



The screenshot shows the TaskServer Administration interface. The 'Queuing system support' section is highlighted, showing the Queue Type set to 'SLURM' and an 'Apply' button. Other sections include 'Computing Resources' (Control per number of Cores, Number of Parallel Cores, Core limit type), 'Directories used' (Installation Directory, Working Directory), and 'Internals' (Port, Save files temporarily, Log Level, Webmaster).

With these settings, the TaskServer will look for a file `LSF.tcl` in the directory `<Install_dir>/TaskServer/Tools/>` where `<Install_dir>` is the MedeA install directory (default is `C:/MD` under Windows or `/home/<username>/MD` under Linux).

3. On the TaskServer admin page, under `http://localhost:23000/ServerAdmin/manager.tml` check the box **Save files temporarily** to keep temporary files on the TaskServer machine during testing.

**Note:** When restarting the TaskServer during testing, this option will be changed back to the default. Click **Apply** or hit **Return**

4. The updated page will show entries **LSF Queue** and **Project**. You may have to set an **LSF Queue** name and/or a **Project** variable depending on your LSF queue settings
5. Select an option for **Control per number of**. This option determines how many *Cores* or *Tasks* you would like to run in the LSF queue at the same time. The LSF queue will handle this parameter depending on your resources.
6. Test the configuration by running a simple job from *MedeA* (make sure you submit to the TaskServer you have just modified) See the JobServer and TaskServer Administration and Configuration section for more details on the JobServer and the TaskServer configurations.

You can look at the output on your TaskServer machine by browsing to `http://<taskserver>:23000/Tasks/` and clicking on the new task directory created by your task (it is not deleted after task completion as we have set the corresponding flag on the TaskServer admin

page). *stdout* contains message and errors from submitting to the queuing system, *VASP.out* contains the output from VASP, while *LAMMPS.out*, *Gibbs.out* and *Mopac.out* are used with other codes.

7. Open the shell script *T\*.VASP* or *T\*.LAMMPS* (which is the LSF queuing script) and check the commands that have been submitted to the LSF queue. A good way to optimize the testing procedure is to use this script to submit test jobs manually on the command line. If you need to set additional options for the queuing system, you can try them out using this file. Once you are sure what parameters to use you can add them to the section of *LSF.tcl* where the script is written, so next time you submit a task these values will be set.

---

**Note:** The setup of LSF, PBS, GridEngine, SLURM, HPC depends on your system and is not controlled by the MedeA TaskServer. The TaskServer simply passes on these values to the queuing system. If your queuing system is not configured correctly, e.g. to run in parallel, these values will have no effect.

---

### 3.2 Basic Workflow in template<Queue>.tcl

template<Queue>.tcl file is a tcl code file that executes when the TaskServer is trigger by the JobServer. This file contains string variables that contain placeholders % %. These place holders are replaced using the *regsub* command to produce the submission script that submitted to the queue.

1. Creation of a basic script to run VASP, LAMMPS, GIBBS, or MOPAC.
2. Find the right executable (if there are different versions as for VASP)
3. Replace placeholders like *%LIB\_PATH%* in the basic script with
4. Full paths, number of nodes, actual scratch directory.
5. Specification of submission command (such as *qsub*, *job*, *bsub*) and
6. Submission.
7. Waiting for results and transfer back to JobServer.

Here is an example section for the LSF queue:

```
set script {#!/bin/sh
#BSUB -q %QUEUE%
#BSUB -P %PROJECT%
#BSUB -n %NPROC%
#BSUB -o "%DIR%/OUTPUT%"
#BSUB -e "%DIR%/ERROR%"
#BSUB -R "span[ptile=%NPROC%]"
#BSUB -J %CODE%-TASKID%
cd "%DIR%"
echo "Running %CODE% on %NPROC% cores on $NNODES nodes:"
echo "  Using executable %EXE%"
echo "On nodes"
cat $TMPDIR/machines
echo ""
export PATH="%PATH%"
export LD_LIBRARY_PATH="%LD_LIBRARY_PATH%"
%PREAMBLE%
trap 'touch finished; touch interrupted' 2 9 15
%RUN%
touch finished
}
```

This file contains only LSF arguments, changing to the actual task directory, copying two input files, and running the proper executable with the required libraries.

The TaskServer defines the respective placeholders.

%DIR%	the actual task directory
%RUN%	the executable with full path (vasp, vasp_parallel)-including mpirun if needed
%EXE%	the executables with full paths
%LD_LIBRARY_PATH%	the LD_LIBRARY_PATH: includes vasp, MKL, and MPI
%CODE%	as short description for the code (like VASP)
%MPIOPTION%	options for pinning or selecting a specific fabric
%TASKID%	the task number
%JOB%	the job number
%PREAMBLE%	any commands required before starting the executable
%OUTPUT%	the output file
%ERROR%	the error file
%NPROC%	number of processors
%QUEUE%	queue as defined on the web interface of the TaskServer
%PROJECT%	and optional project name (for some queuing systems)

### Debugging the Submission Script Template

When debugging the `<Queue>.tcl` file it is best to get the submission script working first, then make changes to the `<Queue>.tcl` file.

1. Point your browser to the TaskServer admin page (<http://localhost:32000/TaskServer/ServerAdmin/manager.tml?server=> and check the box “Save files temporarily” and click **Apply**
2. Submit a new job from MedeA to the queue
3. On a terminal go to the task directory
4. Submit the submission script “TX.medeA. . .” from the command line to the queue using `sbatch` or `srunch`
5. Study the error message and examine its responses on the queue and modify the TX. . . script accordingly
6. Make any necessary changes and reflect the changes to `<Queue>.tcl` when you have a successful TX. . . script.

### 3.3 Path to Submission Command:

If the TaskServer is on a shared file system (throughout the entire cluster) there is not much work required. Just check that queue commands such as `qsub`, `bsub`, `sbatch`, `qdel`, `bdel`, `scancel`, etc, are in the path - or set with absolute path inline. E.g.

```
set "bsub /full_path_to_bsub"
```

### 3.4 Use a Local Scratch Directory with VASP from Global Directory

To use a node-local scratch directory (`/scratch`) add some lines to the script

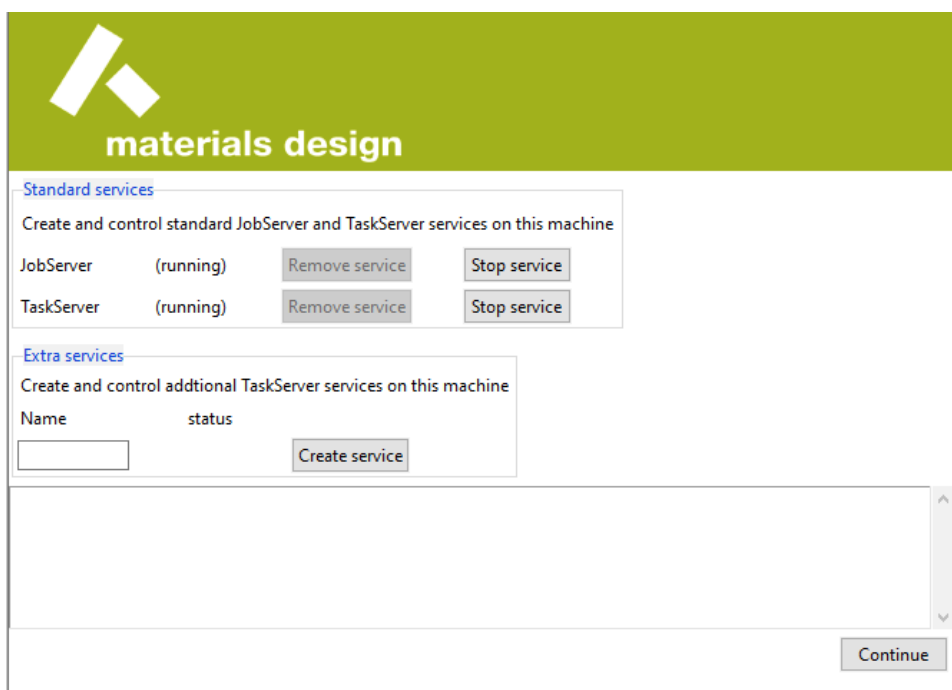
```
cd %DIR%
mkdir /scratch/task%TASKID%
cp * /scratch/task%TASKID%/
cd /scratch/task%TASKID%
...
%RUN%
cp * %DIR%/
touch finished
```

### 3.5 Creating a Manual TaskServer and Queue

If neither of the above two installation scenarios is an option, an alternative is to manually transfer files to and from a remote system, doing the work of the TaskServer by hand, hence a “manual” type. This is a good option for a limited number of large calculations.

**Hint:** Manual TaskServer is not an ideal solution for MT, TSS, Phonon, and HT jobs as these jobs usually contains tens to hundreds of computing tasks. It would be laborious to manually transmit hundreds of tasks to the remote computing resource.

1. Start the Maintenance program
  - On Windows right-click >> **More** >> **Run as Administrator**
  - On Linux, start the MDMaintenance program **without** sudo, `./MDMaintenance.x`
2. Check **Manage MD services** and **Start**



- On Linux, enter your “Sudo password” and click **Check**



**materials design**

**Admin rights**  
 You do not have admin rights, a sudo password is required in order to install, remove, start or stop services  
 Sudo password:

**Standard services**  
 Create and control standard JobServer and TaskServer services on this machine as user rshan and group rshan

JobServer	(running)	<input type="button" value="Remove service"/>	<input type="button" value="Stop service"/>
TaskServer	(running)	<input type="button" value="Remove service"/>	<input type="button" value="Stop service"/>

**Extra services**  
 Create and control additional TaskServer services on this machine

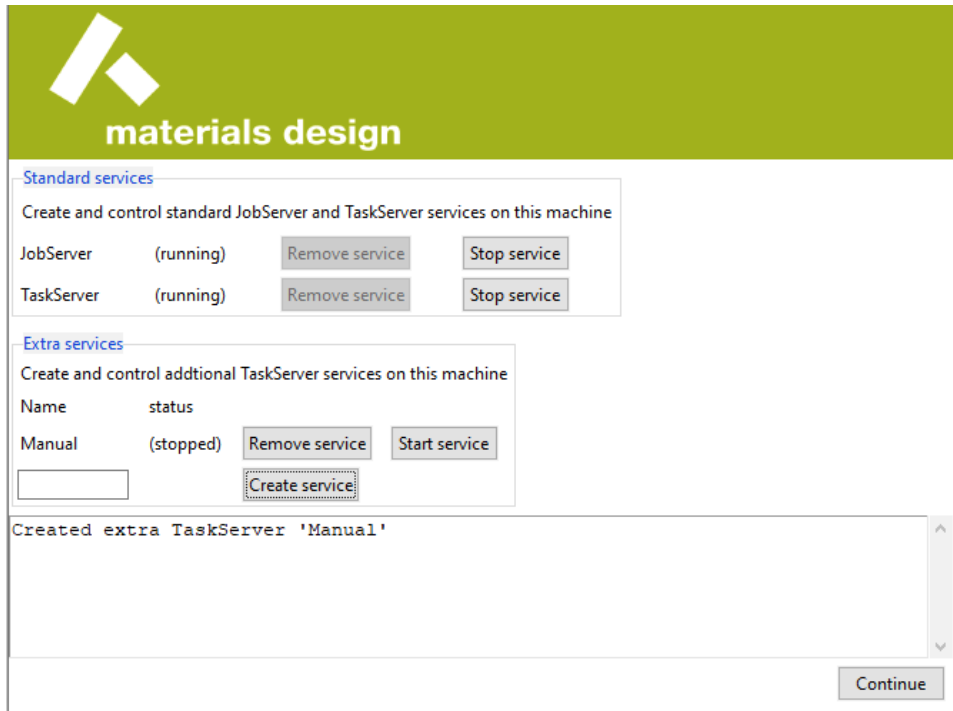
Name	status		
Manual	(running)	<input type="button" value="Remove service"/>	<input type="button" value="Stop service"/>
<input type="text"/>		<input type="button" value="Create service"/>	

- In the “Extra services” section, enter **Manual** for queue name and click **Create service** to generate a new TaskServer, a new queue named *Manual*, and assign the TaskServer to this queue.

---

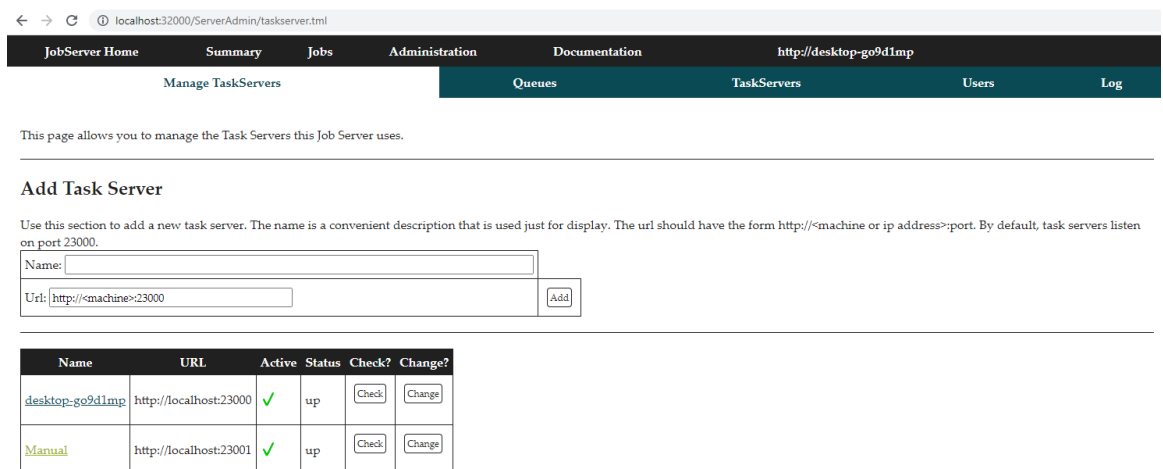
**Note:** This requires a successful installation with both JobServer and TaskServer running.

---



4. Check the newly created Manual TaskServer and Queue is up:

- Go to the JobServer >> Administration >> Manage TaskServers . Check the communication between the JobServer and the TaskServer is **up**. Click the **Check** button if it is not **up**.



- Check the Manual Queue is associated with the Manual TaskServer. Click **Queues** , and click the **Change** button of the *Manual* Queue:



← → ↻ localhost:32000/ServerAdmin/queues.html

JobServer Home Summary Jobs Administration Documentation http://desktop-go9d1mp

Manage TaskServers Queues TaskServers Users Log

Edit the 'Manual' queue

Item	Value
Queue	Manual
Description	On local Task-Server Manual
Default Priority	5
Number of Cores	8
Number of Jobs	1
Is Active	<input checked="" type="checkbox"/>
Change?	
<input type="button" value="Reset"/> <input type="button" value="Update"/>	

#### TaskServers used by the queue

The following table shows the TaskServers connected to this queue. You may remove any TaskServer by pressing the *Remove* button. If there are other TaskServers available, you can select one or more to add in the last row of the table and then press the *Add* button.

TaskServer	
Manual	<input type="button" value="Remove"/>
desktop-go9d1mp	<input type="button" value="Add"/>

Set the *Number of Cores* to 1 and the *Number of Jobs* to a large number such as 20. This will allow more manual task folders to be created at a time

- Go to the Manual TaskServer page, click **Administration**. Verify the Queue Type is set to **manual** and the *Number of Parallel Cores* is large enough

← → ↻ localhost:32000/TaskServer/ServerAdmin/manager.html?server=4

JobServer Home Summary Jobs Administration Documentation http://desktop-go9d1mp

TaskServer Home Tasks Administration http://desktop-go9d1mp

Users JobServers Log

This page allows you to manage the TaskServer on this machine. Use the area below to change the settings of parameters that affect how the TaskServer runs.

If you want to stop the TaskServer -- meaning that no more tasks can be run until it is restarted -- use the button at the bottom of the page.

Computing Resources			
Control per number of	Cores		
Number of Parallel Cores	16384	The maximum cumulated number of cores to use for all tasks. Ideally set to the machine or cluster total number of cores and considering hyperthreading effect if relevant.	Core limit type <input type="button" value="Hard"/> <b>Hard:</b> allocate no more than available cores. <b>Soft:</b> if necessary overload by maximum 50%.
Queue Type	manual	<input type="button" value="Apply"/>	
Directories used			
Installation Directory	C:/MD	Working Directory	C:/Users/ThomasNelson/Tasks
Internals			
Port	23001	The port to listen on (default is 23000).	https <input type="checkbox"/> Use https rather than the less secure http.
Log Level	notice	Webmaster	Support@MaterialsDesign.com
<input type="button" value="Apply"/>			

**Hint:** If the Manual TaskServer does not come up, check all sub-directories in MD/TaskServer are owned by the user

See the JobServer and TaskServer Administration and Configuration section for more details on the Job-Server and the TaskServer configurations.

### 3.6 Using the TaskServer in Manual Mode

1. Select queue *Manual* when submitting a job
2. The JobServer creates all required input files on the local TaskServer in `~MD/TaskServer/Tasks/task00XXX`, but instead of launching the calculation waits patiently till the file *DeleteWhenDone.txt* is removed.

3. Find the task number *XXX* of the newly created Job *YYY* on its status page <http://localhost:32000/jobStatus.tml?id=YYY>
4. Manually transfer the input files from `~/MD/TaskServer/Tasks/task00XXX` to the target machine
5. Run the calculation on the remote machine
6. Transfer all output files back to the task folder
7. Delete *DeleteWhenDone.txt*
8. The JobServer processes the results and, depending on the type of calculation, creates Trajectory, BandStructure, Density of State to be visualized in *MedeA*.

---

**Note:** Run some test calculations to understand how the JobServer and TaskServer work. Many VASP jobs require 2 or 3 tasks to generate all the result files needed.

It is necessary to keep track of ongoing calculations and copy results back to the originating task directory.

For VASP tasks, you can only modify input parameters specifying parallelization options (namely NPAR and NCORE) but not change the structure or type of job. There are no limitations for other codes.

---

## 4 SSH Tunnel access

When connecting to the JobServer and the TaskServer that does not have a public/external IP address, you can use the port forwarding method to create SSH tunnels.

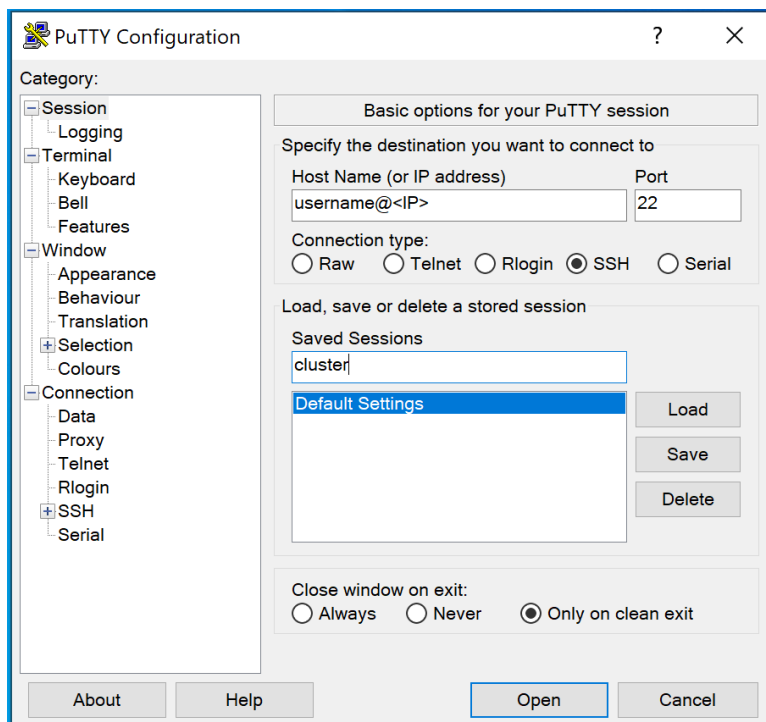
### 4.1 Windows

Instructions given below are for the **Putty** program. A similar procedure can be applied to other graphical ssh programs such as MobaXterm.

1. First download and install [Putty](#) [1].
2. Setup a session with your HPC machine, as shown below:

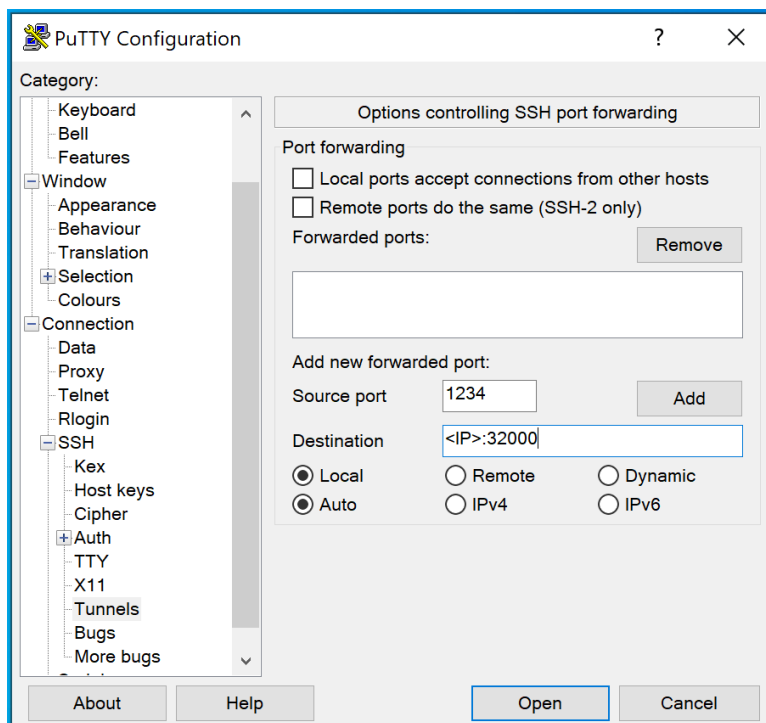
---

[1] <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>



where  $\langle IP \rangle$  is the IP address of the HPC machine. Then save the session with the name of the HPC machine. This will allow you ssh access to the machine.

Then expand the *SSH* menu and select *Tunnels*. Enter a source port not currently being used on your machine, such as 32001. Then set the Destination as the  $\langle IP \rangle.32000$  for the JobServer running on your HPC machine. Then click *Add*.



Go back to sessions and click *Save* on more time. Then click *Open* and type in your password. You see an open terminal to the HPC machine in the Putty window. You can then access the JobServer page from your browser from <http://localhost:32001>. You can also add <http://localhost:32001> to your JobServers in the Job Servers tab of the MedeA Preferences menu by selecting **File** then **Preferences**.

If you are familiar with Windows command prompt, you can use the following command:

```
ssh -L *:<local_port_number>:localhost:<remote_port_number> username@IP
```

So if you would like to create an SSH tunnel between your local port 32001 and the remote port 32000 (which is the JobServer on the remote computer), you can use:

```
ssh -L *:32001:localhost:32000 username@IP
```

This process can be simplified by setting up a Desktop shortcut. Right-click on your desktop and select >> **New** >> **Shortcut** from the drop-down list to generate a shortcut. Enter for **Type the location of the item:**

```
powershell.exe -noexit -ExecutionPolicy Bypass -command "ssh -L  
↩*:32001:localhost:32000 username@IP"
```

Click on **Next** and name your shortcut. By double-clicking on this shortcut icon you can now start your SSH tunnel.

## 4.2 Linux

On Linux you can open a Terminal and enter the following command:

```
ssh -L *:<local_port_number>:localhost:<remote_port_number> username@IP
```

So if you would like to create an SSH tunnel between your local port 32001 and the remote port 32000 (which is the JobServer on the remote computer), you can use:

```
ssh -L *:32001:localhost:32000 username@IP
```