PENNSTATE

INSTITUTE FOR CYBERSCIENCE

Lion-X Clusters User Guide

Including Hammer, Anvil, CyberSTAR, and BioSTAR

TABLE OF CONTENTS

GETTING STARTED WITH THE LION-X CLUSTERS

This user guide provides information about the Institute for CyberScience's Lion-X clusters, including instructions on how to perform basic tasks, such as requesting accounts and logging in.

OUR SYSTEMS

The "Lion-X clusters" include Lion-XF, Lion-XG, Lion-XH, Lion-XJ, Lion-GA, Lion-LSP, Hammer, Anvil, CyberSTAR, BioSTAR, and LionX-Hershey. Most of these are used for batch job submissions, while others service particular applications. For example, Hammer is the interactive cluster, used for short jobs that require dedicated GPU (Graphical Processor Unit) processing capacity. Anvil provides a Microsoft Windows computing environment. ICS does not currently operate a Mac cluster.

The clusters and their hostnames are:

Lion-XF	lionxf.rcc.psu.edu
Lion-XG	lionxg.rcc.psu.edu
Lion-XH	lionxh.rcc.psu.edu
Lion-XJ	lionxj.rcc.psu.edu
Lion-XV	lionxv.rcc.psu.edu
Lion-GA	lionga.rcc.psu.edu
Lion-LSP	lion-lsp.rcc.psu.edu
Hammer	hammer.rcc.psu.edu
Anvil	anvil.rcc.psu.edu
CyberSTAR	cyberstar.rcc.psu.edu
BioSTAR	biostar.rcc.psu.edu
LionX-Hershey	lionxhershey.rcc.psu.edu (from outside Hershey network) lionxhershey.hersheymed.net (from inside Hershey network)

HOW TO USE THE SYSTEMS

All users must register for an ICS account, if one does not exist, and connect to the clusters using secure shell (SSH), either in a terminal application command line, PuTTY, or Exceed onDemand. All users must abide by ICS access and data storage policies. ICS provides a wide selection of scientific software applications available through our software stack.

ACCOUNTS

Users that need to register for an account on any one of the clusters listed above may do so by accessing https://ics.psu.edu/advanced-cyberinfrastructure/accounts/ics-accounts/. Users also need to sign up for the University's "Duo" two-factor authentication at http://identity.psu.edu/services/authentication-services/two-factor/self-service-portal/. Users who do not have a Penn State User ID but are collaborators from other institutions also need a "Slim" access account, which must be requested before registering for the other two accounts. Request a

"Slim" access account by following the process outlined here: https://ics.psu.edu/advanced-cyberinfrastructure/accounts/slim-access-accounts/.

Some systems require additional approval to access. These include BioSTAR, Lion-LSP, and LionX-Hershey. CyberSTAR is no longer accepting new account requests. For technical support, please contact the i-ASK Center at iask@ics.psu.edu.

ACCESSING SYSTEMS

Connect to the Lion-X clusters using SSH, a secure protocol that encrypts all data sent between the client computer and server. SSH applications usually allow both interactive terminal sessions on the remote machine and the ability to transfer files securely.

Linux/Unix/Apple OS X
Linux, Unix, and Apple OS X all include the standard command line SSH utilities. Information on using them to open a remote terminal session on one of our systems can be found in the Command Line SSH User Guide. The guide also provides information on transferring files.

Windows
Information on using PuTTY, a freely available SSH client for Windows, can be found in the PuTTY User Guide. We recommend using WinSCP, a freely available SSH file transfer client for Windows. Information on using WinSCP can be found in the WinSCP User Guide.

Using a Remote Graphical User Interface
The ICS interactive cluster is called Hammer. Connect to it via a graphical user interface (GUI) by following the Exceed onDemand User Guide. **Using Exceed onDemand is the preferred method of connecting to Hammer.**

STORAGE POLICIES

Each user has a home directory at /gpfs/home/<userid>, a work directory at /gpfs/work/<userid>, and a scratch directory at /gpfs/scratch/<userid>. GPFS stands for "general parallel file system." The command to check disk quota usage in GPFS is **mmlsquota**.

The /gpfs/home directory is for the most important files, like original source code and paper drafts. It transparently keeps two copies of data on different disk storage targets to increase data availability in the case of a major disaster that would normally require restoring data from tape. The quota for this directory is 8 GB and will not be increased except under extenuating circumstances.

The /gpfs/work directory is where normal data should go for individual users. It does not have data replication like /gpfs/home, but it is backed up to tape and is higher performing than /gpfs/home. The quota for this directory is 64 GB. Research groups can have a shared group directory with policies similar to /gpfs/work.

The /gpfs/scratch directory provides over 130 TB of space for anyone to use. Files here are not backed up and are removed if they have not been used in 30 days, but there are no quotas, so huge datasets or output can be used.

The following table summarizes the directory policies and characteristics:

Directory	Purpose	Quota	Backed Up	Replicated[1]	File Lifetime Limit
/gpfs/home	User Home Directories	8 GB	Yes	Yes	No
/gpfs/work	User Work Directories	64 GB	Yes	No	No
/gpfs/group	Research Group Directories	Variable	Yes	No	No
/gpfs/scratch	Temporary File Area	None	No	No	Yes[2]

1) Replicated means that two copies of every piece of data are stored on disk for extra data protection.
2) 30 days since last use or 60 days since creation.

SOFTWARE

Software on the Lion-X clusters is managed through the Environment Modules package. It allows a user to do such things as: list available software, load software into the environment, and unload software from the environment. Information on using Environment Modules can be found in the Environment Modules User Guide.

Compiling Code

The following compilers are available:

Compiler	C Command	C++ Command	Fortran 77 Command	Fortran 90 Command
GNU	gcc	g++	Gfortran	gfortran
Intel	icc	icpc	Ifort	ifort
Portland Group	pgcc	pgCC	pgf77	pgf90

Compiling MPI Applications

The Lion-X clusters use OpenMPI for their MPI library. Instructions for compiling and running MPI applications are the same as what can be found in the OpenMPI software page.

Running Jobs

The Lion-X clusters use PBS for job queuing and execution. For information on running jobs on the Lion-X clusters, please see the PBS User Guide. Jobs on Hammer are run interactively.

ANVIL (WINDOWS) USER GUIDE

STORAGE

Each user has a windows **home** directory, and their GPFS space mounted as H: drive.
Users may find it convenient to create a desktop link to this space.

ACCESSING ANVIL

System Hostnames
The following is the hostname of the login node of Anvil --> anvil.rcc.psu.edu. When users connect, they will see a normal Windows log in box. Users should use their Access Account **userid@dce.psu.edu** as their user name and their normal Access Account password as their password.

Using RDC to Log In
Connections to Anvil are only through RDC (Remote Desktop Connection), available under accessories. RDC is a secure protocol that encrypts all data sent between the client computer and server.

Linux/Unix/Apple OS X
Linux, Unix, and Apple OS X all support RDC clients using the rdesktop command.

Reconnecting to your open session
Users can use anvil1.rcc.psu.edu through anvil4.rcc.psu.edu as hostnames. To re-access the same host, enter that hostname (e.g., anvil2.rcc.psu.edu).

Disconnecting
The RDC client will keep a session open if the X is used to close the connection, so please use "LOG OFF" instead.

TRANSFERRING FILES

Windows
From a Windows desktop, users may copy and paste to their Anvil window. File space (GPFS) is also mounted on Anvil as the H: drive.

Linux/Unix/Apple OS X
See the Command Line SSH User Guide.

ENVIRONMENT MODULES USER GUIDE

Environment Modules is a framework to manage what software is loaded into a user's environment. Its functionality includes the ability to list all software packages currently available in the Environment Modules system, list all software packages loaded into a user's environment, load a new software package into a user's environment, and unload a software package from a user's environment.

LISTING ALL AVAILABLE PACKAGES

The command to list all available packages is module avail.

The format of the listed packages is <package name>/<package version>. For example, gcc/4.4.2 is version 4.4.2 of gcc.

A slightly more informative listing can be attained by using the module whatis command.

LISTING CURRENTLY LOADED PACKAGES

To see what packages are currently loaded into a user's environment, the command is module list.

LOADING A PACKAGE

The command for loading a package into a user's environment is module load <package name>. This loads the most recent version of <package name>. If a specific version of a package is desired, the command can be expanded to module load <package name>/<package version>.

Note that Environment Modules will load any prerequisites for a package when that package is loaded. If, for example, the user starts without any packages loaded and then loads gcc, gcc has several supporting libraries that it needs to run, so these packages are automatically loaded.

UNLOADING A PACKAGE

The command to unload a package is module unload <package name>. This command will also remove any prerequisites for the package being removed.

USING ENVIRONMENT MODULES WITH PBS

The module for a software package needs to be loaded within a PBS job as well. A sample PBS script for running the application R with the input file R.in and the output file R.out would be as follows:

#PBS -l nodes=1:ppn=1

```
#PBS -l walltime=24:00:00
#PBS -j oe

cd $PBS_O_WORKDIR

module load R
R --no-save --no-restore < R.in > R.out
```

MAKING MODULES LOAD AUTOMATICALLY ON LOGIN

Modules can be loaded automatically on login by adding the appropriate module load commands to a user's .bashrc file. The .bashrc file is located in the top level of a user's home directory. The following example shows a .bashrc file that automatically loads R, Abinit, and Gaussian:

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions

module load R
module load abinit
module load gaussian
```

PBS USER GUIDE

OVERVIEW

PBS is a job resource manager. A job is defined as a computational task, such as computational simulation or data analysis. PBS provides job queuing and execution services in a batch cluster environment.

In ICS-ACI systems, PBS works with the Moab job scheduler. PBS provides job information to Moab, and Moab tells PBS which jobs to run and on which compute nodes in a cluster to run the jobs.

JOB LIMITS

PBS is configured on each system to have a number of separate job queues. There is a default queue on each system that every user may access. Each funding group has its own queue.

These limits are placed on **funding group queues**:

Maximum Walltime (Hours)	96 (default) to 336+
Maximum Processors in Use Per User	No Limit
Maximum Job Size (Processors)	32

Note that the walltime limits placed on funding group queues are arbitrary and can be adjusted at the request of the group's PI.

If you believe you are part of a funding group of a system, and you do not know what queue you should be using, please email us at iask@ics.psu.edu stating what group you are part of and that you need to know your group queue.

These limits are placed on system **default queues**:

Limit	Lion-X	CyberStar
Maximum Walltime (Hours)	24	96
Maximum Processors in Use Per User	32	No Limit
Maximum Job Size (Processors)	32	256 (nodes=32:ppn=8)

Most queue limits can be checked by running the command qstat -q.

SUBMITTING A JOB

Jobs are submitted to a PBS queue so that PBS can dispatch them to be run on one or more of a cluster's compute nodes. There are two main types of PBS jobs:

Non-interactive Batch Jobs: this is the most common PBS job. A job script is created that contains PBS resource requests and the commands necessary to execute the job. The job script is then submitted to PBS to be run non-interactively.
Interactive Batch Jobs: this is a way to get an interactive terminal on one or more of the compute nodes of a cluster. Commands can then be run interactively through that terminal directly on the compute nodes for the duration of the job. Interactive jobs are helpful for such things as program debugging and running many short jobs.

NON-INTERACTIVE BATCH JOBS

There are two steps to running a non-interactive batch job:

STEP ONE: CREATE A PBS SCRIPT

A PBS script is a standard Unix/Linux shell script that contains a few extra comments at the beginning that specify directives to PBS. These comments all begin with #PBS. The most important PBS directives are:

Definition of Important PBS Directives

PBS Directive	Description
#PBS -l walltime=*HH*:*MM*:*SS*	This directive specifies the maximum walltime (real time, not CPU time) that a job should take. If this limit is exceeded, PBS will stop the job. Keeping this limit close to the actual expected time of a job can allow a job to start more quickly than if the maximum walltime is always requested.
#PBS -l pmem=*SIZE*gb	This directive specifies the maximum amount of physical memory used by any process in the job. For example, if the job would run four processes and each would use up to 2 GB (gigabytes) of memory, then the directive would read #PBS -l pmem=2gb.
#PBS -l nodes=*N*:ppn=*M*	This specifies the number of nodes (nodes=N) and the number of processors per node (ppn=M) that the job should use. PBS treats a processor core as a processor, so a system with eight cores per compute node can have ppn=8 as its maximum ppn request. Note that unless a job has some inherent parallelism of its own through something like MPI or OpenMP, requesting more than a single processor on a single node is usually wasteful and can impact the job start time.
#PBS -q queuename	This specifies what PBS queue a job should be submitted to. *This is only necessary if a user has access to a special queue.* This option can and should be omitted for jobs being submitted to a system's default queue.
#PBS -j oe	Normally when a command runs it prints its output to the screen. This output is often normal output and error output. This directive tells PBS to put both normal output and error output into the same output file.

The following is an example PBS script.

```
# This is a sample PBS script. It will request 1 processor on 1 node
# for 4 hours.
#
#   Request 1 processors on 1 node
#
#PBS -l nodes=1:ppn=1
#
#   Request 4 hours of walltime
#
#PBS -l walltime=4:00:00
#
#   Request 1 gigabyte of memory per process
#
#PBS -l pmem=1gb
#
#   Request that regular output and terminal output go to the same file
#
#PBS -j oe
#
#   The following is the body of the script. By default,
#   PBS scripts execute in your home directory, not the
#   directory from which they were submitted. The following
#   line places you in the directory from which the job
#   was submitted.
#
cd $PBS_O_WORKDIR
#
#   Now we want to run the program "hello".  "hello" is in
#   the directory that this script is being submitted from,
#   $PBS_O_WORKDIR.
#
echo " "
echo " "
echo "Job started on `hostname` at `date`"
./hello
echo " "
echo "Job Ended at `date`"
echo " "
```

Note that the above example script is for a non-MPI job. Information on how to write PBS scripts for MPI jobs can be found here.

STEP TWO: SUBMIT THE PBS SCRIPT TO PBS FOR EXECUTION

Once a PBS script is created, it needs to be submitted to PBS so that it becomes eligible to be run. The command to submit a script to PBS is called qsub, and its syntax is: qsub *scriptfile*.

The following is an example of using qsub to submit a PBS script called myjob.
% qsub myjob
95.lionxj.rcc.psu.edu

The job script myjob has just been submitted to PBS and has been assigned the Job_ID 95.lionxj.rcc.psu.edu. This Job_ID can later be used to control the job.

INTERACTIVE BATCH JOBS

Interactive PBS jobs are similar to non-interactive PBS jobs in that they are submitted to PBS via the command qsub. Submitting an interactive PBS job differs from a non-interactive PBS job in that a PBS script is not necessary. All PBS directives can be specified on the command line. The syntax for submitting an interactive PBS job is: qsub -I ... *pbs directives* ...

The -I flag above tells qsub that this is an interactive job. The following example demonstrates using qsub to submit an interactive job using one processor on one node for four hours:

lionxi:~$ qsub -I -l nodes=1:ppn=1 -l walltime=4:00:00
qsub: waiting for job 1064159.lionxi.rcc.psu.edu to start
qsub: job 1064159.lionxi.rcc.psu.edu ready

lionxi25:~$

There are two things of note here. The first is that the qsub command does not exit when run with the interactive -I flag. Instead, it waits until the job is started and gives a prompt on the first compute node assigned to a job. The second thing of note is the prompt lionxi25:~$ - this shows that commands are now being executed on the compute node lionxi25.

CHECKING JOB STATUS

The command to check job status is qstat, which has **many options**. Some common ones are:

qstat	Shows status of all PBS jobs. Time displayed is the *CPU time* used by the job.
qstat -s	Shows status of all PBS jobs. Time displayed is the *walltime* used by the job.
qstat -u *userid*	Shows status of all PBS jobs submitted by the user *userid*. The time displayed is the *walltime* used by the job.
qstat -n	Shows status of all PBS jobs along with a list of compute nodes that the job is running on.
qstat -f *jobid*	Shows detailed information about the job *jobid*.

A job can be in several **different states**. The most common ones are:

State	Meaning
Q	The job is *queued* and is waiting to start.
R	The job is currently *running*.
E	The job is currently *ending*.
H	The job has a user or system *hold* on it and will not be eligible to run until the *hold* is removed.

DELETING A JOB

The command to delete a job is qdel. Its syntax is "qdel *Job_ID*."

PBS Commands for Deleting Jobs

Command Name	Description of Command Functionality
qdel *Job_ID*	Deletes the job identified by *Job_ID*.
qdel $(qselect -u *username*)	Deletes all jobs belonging to user *username*.

Example: deleting a job with Job_ID 10:
lionxj:~$ qdel 10

Example: deleting all jobs belonging to user abc123:
lionxj:~$ qdel $(qselect -u abc123)

VIEWING JOB OUTPUT

By default, PBS will write screen output from a job to the following files:

PBS Output Files

Output File Name	Contents of Output File
Jobname.o*Job_ID*	This file would contain the non-error output that would normally be written to the screen.
Jobname.e*Job_ID*	This file would contain the error output that would normally be written to the screen.

If the PBS directive #PBS -j oe is used in a PBS script, the non-error and the error output are both written to the *Jobname*.o*Job_ID* file.

COMMAND LINE SSH USER GUIDE

SSH is a secure protocol that encrypts all data sent between the client computer and the computer it is connecting to. SSH applications usually allow both interactive terminal sessions on the remote machine and the ability to transfer files securely.

This guide is intended as a very basic introduction to using command line SSH commands. A lot more information on using SSH can be found online, in books, and in the man pages for each individual SSH command.

OPENING A TERMINAL ON A REMOTE MACHINE THROUGH SSH

The command line SSH command to open a terminal on a remote machine is just called ssh. The syntax to connect to a remote machine is: ssh *username@hostname* where *username* is the name of your account on the remote system and *hostname* is the hostname of the remote system that is being connected to.

The following is an example of the user abc123 connecting to the Lion-X cluster lionxj.rcc.psu.edu:

[abc123@funkmachine ~]$ ssh abc123@lionxj.rcc.psu.edu
abc123@lionxj.rcc.psu.edu's password: password
[abc123@lionxj ~]$
The "[abc123@lionxj ~]$" prompt now shows that user abc123 is logged into lionxj.

TRANSFERRING FILES WITH SSH

There are two main command-line SSH commands to transfer files: **scp** (a non-interactive command that takes a set of files to copy on the command line, copies them, and exits) and **sftp** (an interactive command that opens a persistent connection that multiple copying commands can be performed through).

SCP

To copy one or more local files up to a remote server, the scp syntax would be:
scp *local_file(s) user@hostname:destination_directory*

Example: copy individual files to a remote system
For user *abc123* to copy the local files *foo.c* and *foo.h* into their home directory on the host lionxj.rcc.psu.edu, the following command would be used:
[abc123@funkmachine ~]$ scp foo.c foo.h abc123@lionxj.rcc.psu.edu:.
abc123@lionxj.rcc.psu.edu's password: *password*
foo.c 100% 122 0.1KB/s 0.1KB/s 00:00
foo.h 100% 14 0.0KB/s 0.1KB/s 00:00

Example: copy a whole directory to a remote system
For a user to copy a whole directory called *srcdir* including all of its contents into their home directory on the host hammer.rcc.psu.edu, the -r flag can be added to the scp command. The -r flag tells scp to descend recursively into a directory and copy all of its contents.

```
[abc123@funkmachine ~]$ scp -r srcdir abc123@hammer.rcc.psu.edu:
abc123@lionxj.rcc.psu.edu's password: password
bork.c          100%  388     0.4KB/s  22.0KB/s   00:00
swap2.c         100%  199     0.2KB/s  22.0KB/s   00:00
list.c          100%  1029    1.0KB/s  22.0KB/s   00:00
array.c         100%  536     0.5KB/s  22.0KB/s   00:00
foo.c           100%  122     0.1KB/s  22.0KB/s   00:00
acid.c          100%  446     0.4KB/s  22.0KB/s   00:00
blah.c          100%  112     0.1KB/s  22.0KB/s   00:00
```

Example: copy files down from a remote system
To copy files from a remote system down to a local machine, the order of the scp command just needs to be reversed so that the first argument (the source argument) is the remote system, and the second argument (the destination argument) is a directory on the local system.

For user *abc123* to copy the remote file *results.txt* on the server lionxi.rcc.psu.edu into their current directory on their local machine, the following command would be used:

```
scp jwh128@lionxi.rcc.psu.edu:results.txt .
abc123@lionxj.rcc.psu.edu's password: password
results.txt                   100%    8     0.0KB/s   0.0KB/s   00:00
```

SFTP

The interactive command sftp uses the same syntax as a standard command-line ftp client. It differs from a standard ftp client in that the authentication and the data transfer happen through the SSH protocol rather than the FTP protocol. The SSH protocol is encrypted whereas the FTP protocol is not.

There are a number of basic commands that are used inside of stfp:
put *filename*: uploads the file *filename*
get *filename*: downloads the file *filename*
ls: lists the contents of the current remote directory
lls: lists the contents of the current local directory
pwd: returns the current remote directory
lpwd: returns the current local directory
cd *directory*: changes the current remote directory to *directory*
lcd *directory*: changes the current local directory to *directory*

The syntax for calling sftp is:
sftp *username@hostname*

WINSCP INSTALLATION HOW-TO

OVERVIEW

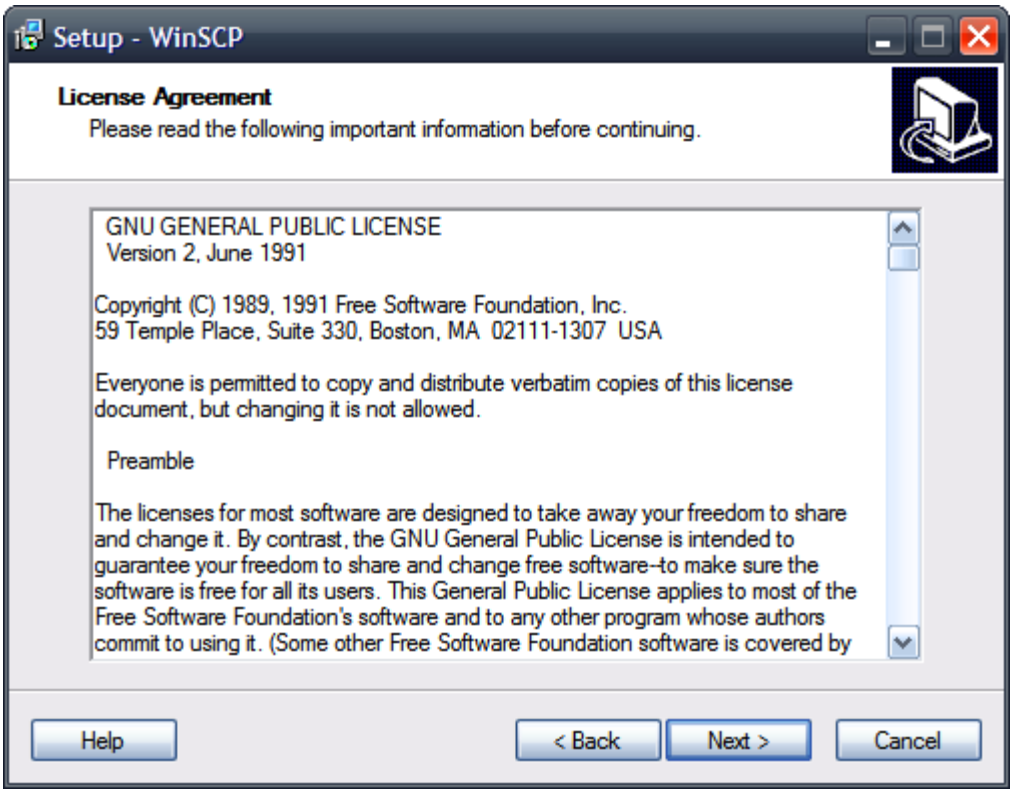
WinSCP provides a free secure FTP (SFTP) and secure copy (SCP) client for Windows using SSH, allowing users to transfer files to and from our cluster file system.

For more information, please visit the WinSCP homepage at http://winscp.net/eng/index.php.

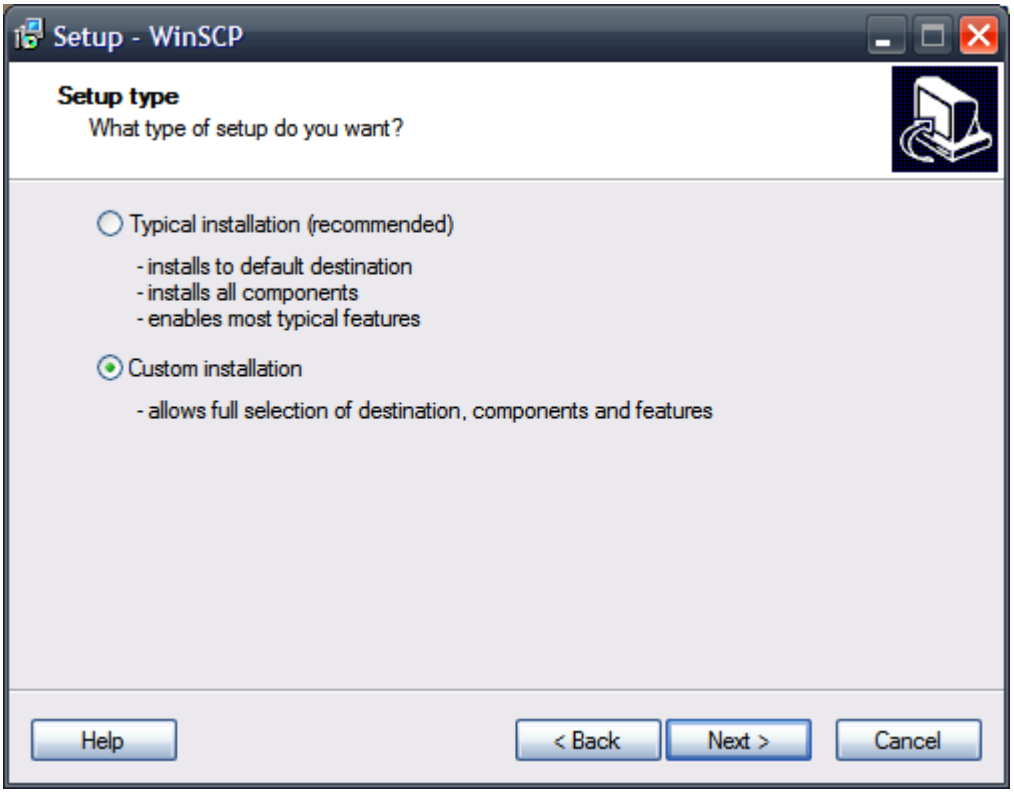
INSTALLATION

Download and run the current setup executable available at http://winscp.net/download/winscp381setup.exe. You will see the following intro screen. Click [Next].

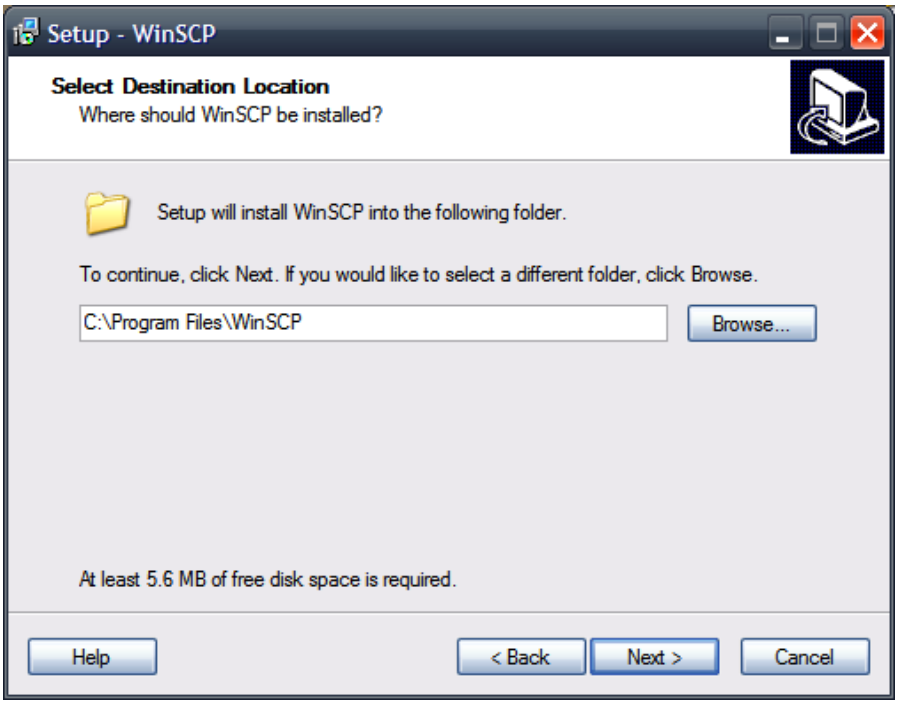
The installer will display the GPL license for review. Once you review the license, click [Next].



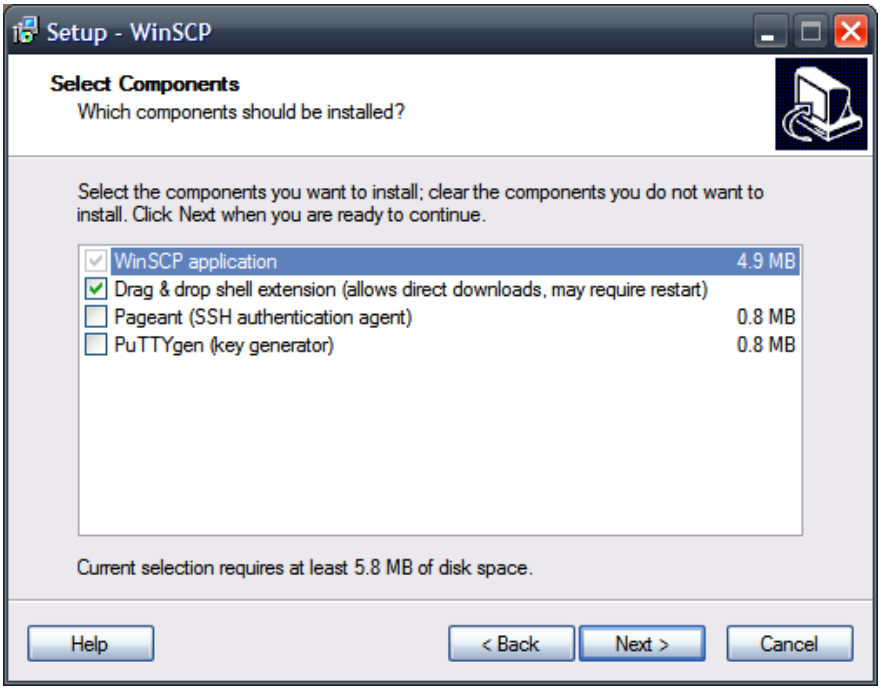
The custom installation type is recommended. Click [Next].



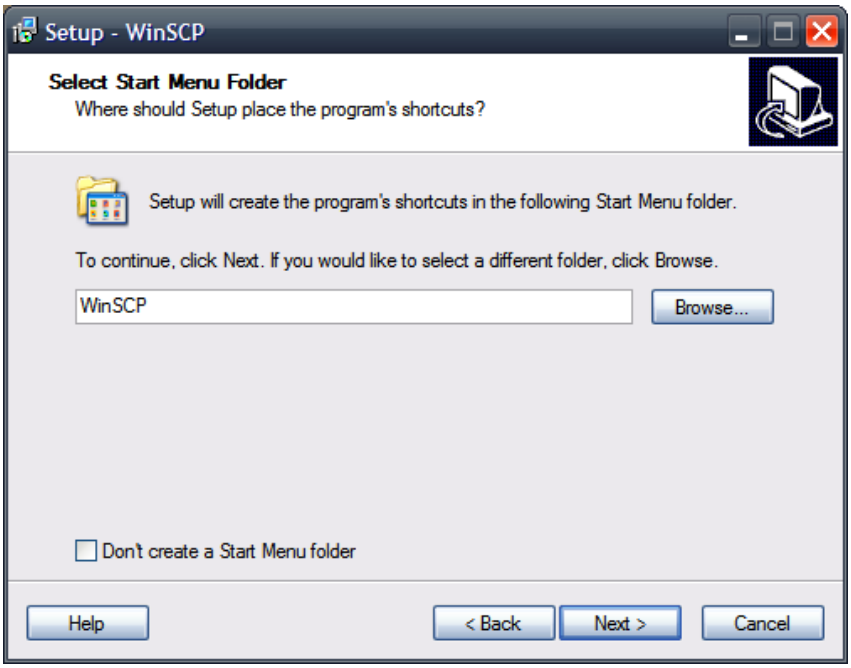
The default installation location is recommended. Click [Next].



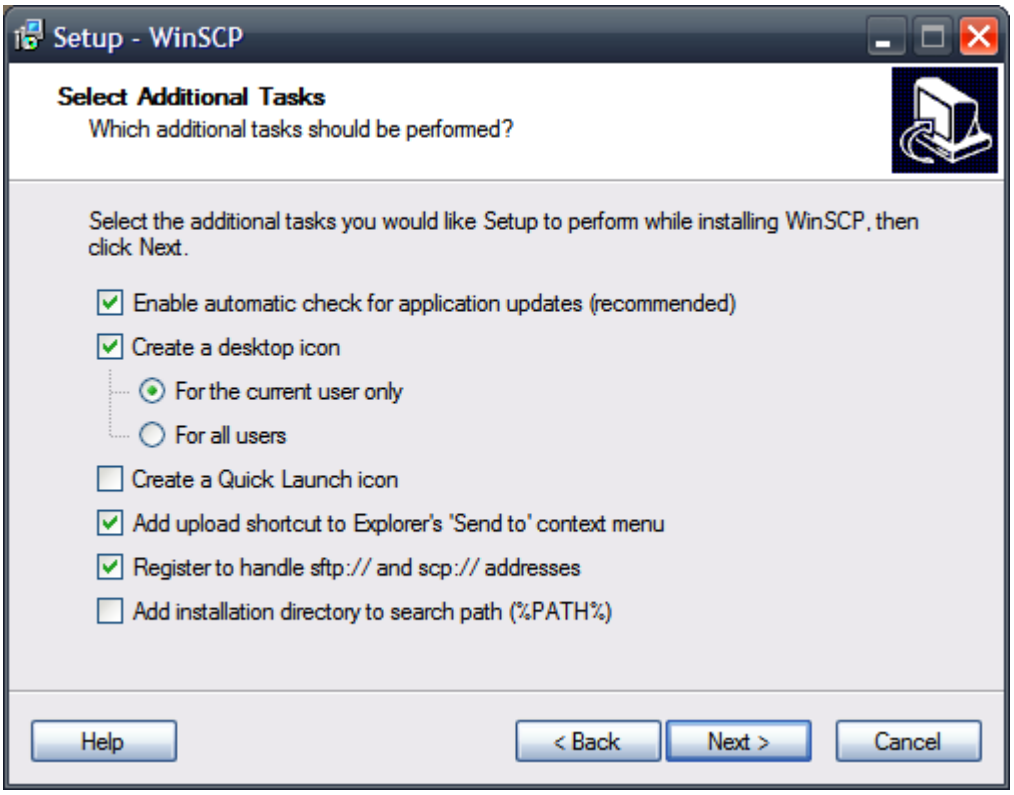
The default component selection is fine, but if you have PuTTY installed, you already have Pageant and PuTTYgen, so these items can be deselected. Leaving them selected will not cause any harm. We recommend leaving the “Drag & Drop shell extension” checked as this makes for very easy file management. Once you have selected your desired installation components, click [Next].



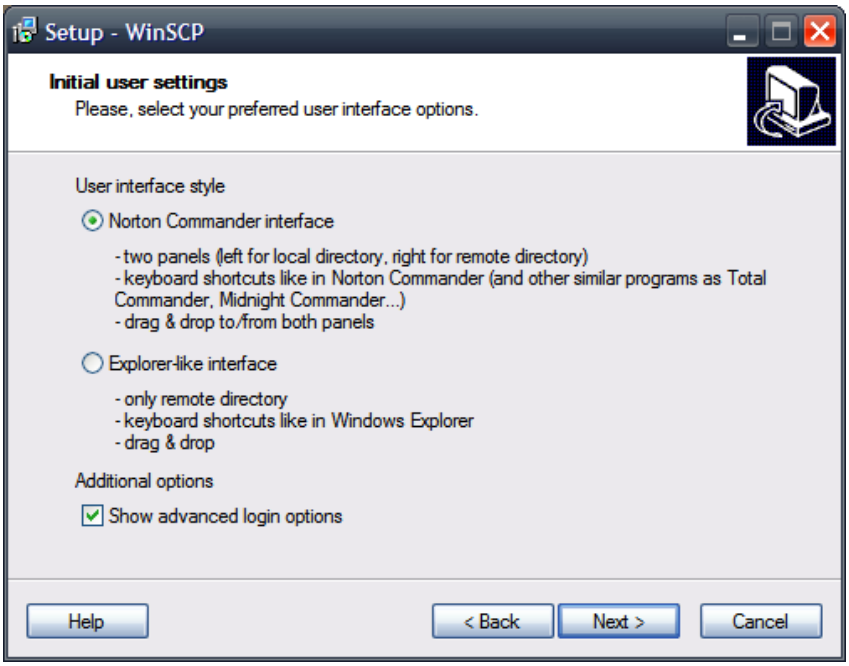
The Start Menu folder is simply the name of the sub directory under which your WinSCP shortcuts will be located. The default value should be fine. Click [Next].



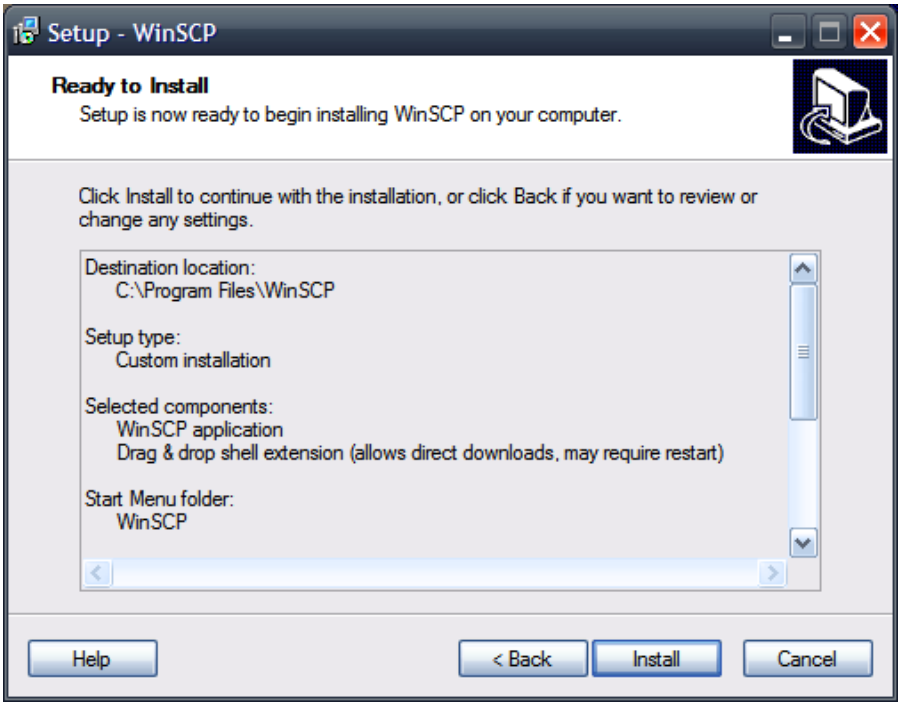
Additional tasks include creating shortcuts in various locations to make WinSCP easier to access. The following option set is reasonable. Click [Next].



We recommend using “Norton Commander interface” style, since it provides a dual-pane, easy-to-use look. “Show advanced login options” adds some options to the connection pane and may remain checked or be unchecked since most users will not need the additional settings. These settings can be changed later if desired. Click [Next].



The installer will verify the settings. If everything looks as you set it in previous steps, click [Install].



Following a brief installation status screen, you should see the following dialog indicating successful installation of WinSCP. You are now equipped to transfer and manage files remotely. Due to the shell extensions WinSCP provides, a reboot is necessary. Select whether you wish to reboot immediately. Click [Finish].

INSTALLING PUTTY SSH ON WINDOWS

OVERVIEW

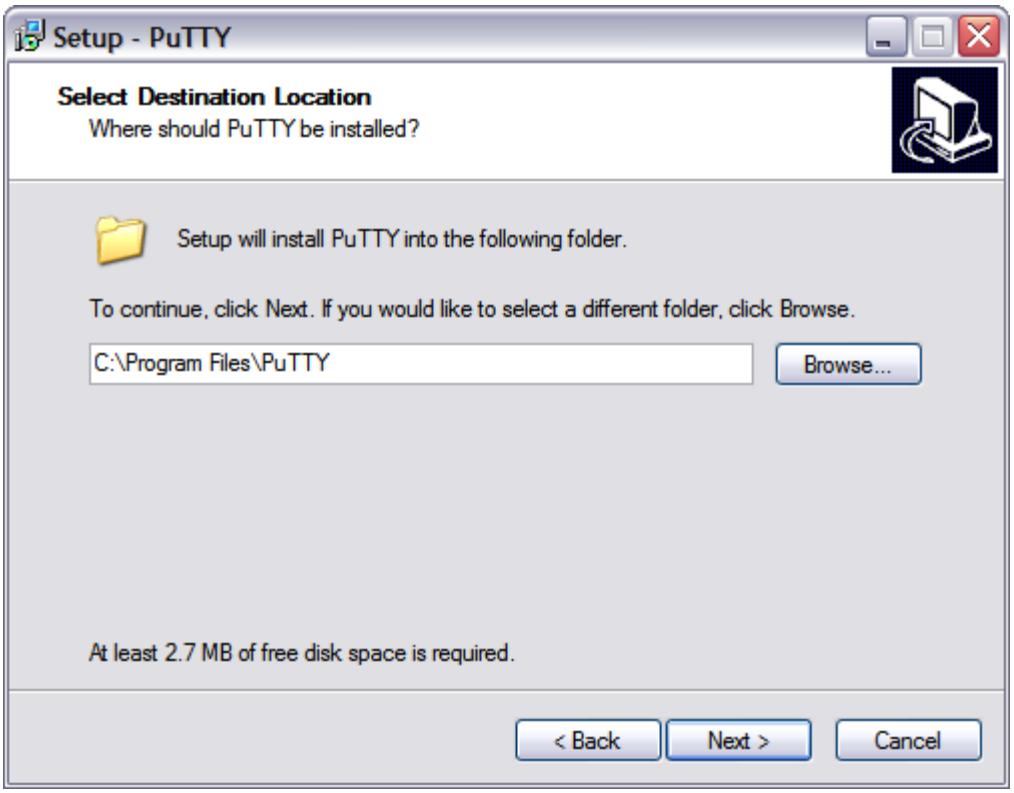
PuTTY provides SSH connectivity for Windows and emulates an xterm terminal. This allows users to connect to our Linux machines remotely from a Windows environment.

For more information, please visit the PuTTY homepage.

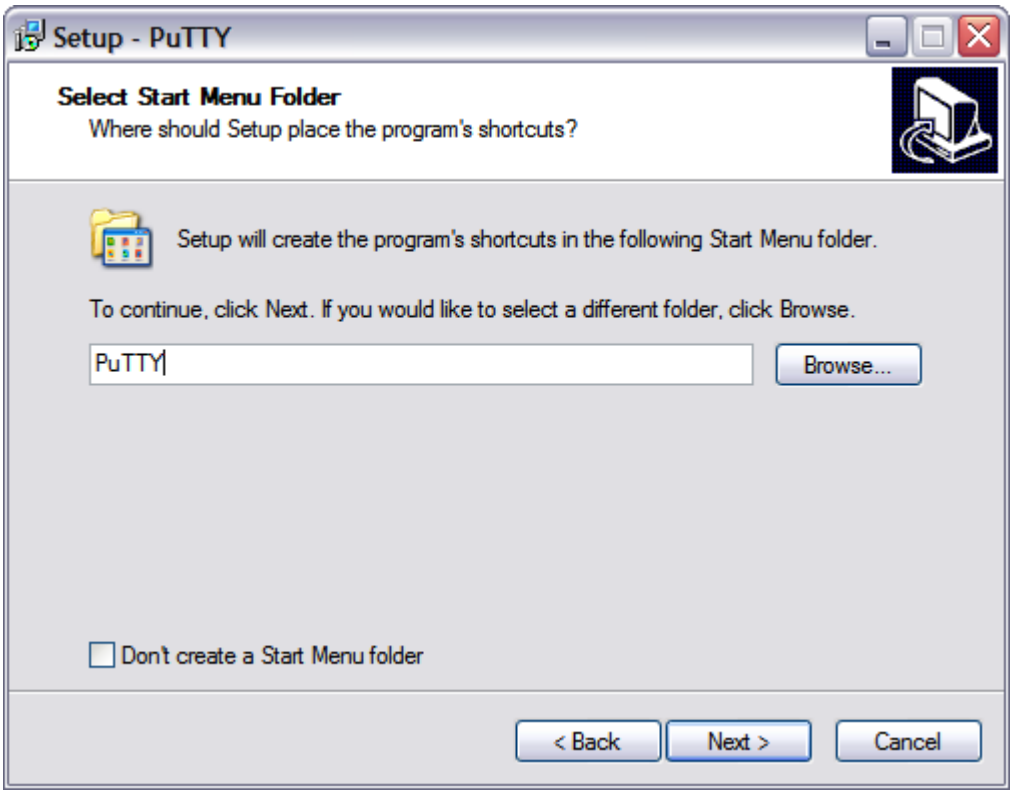
INSTALLATION

Download and run the current setup executable available at http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe. You will see the following intro screen. Click [Next].

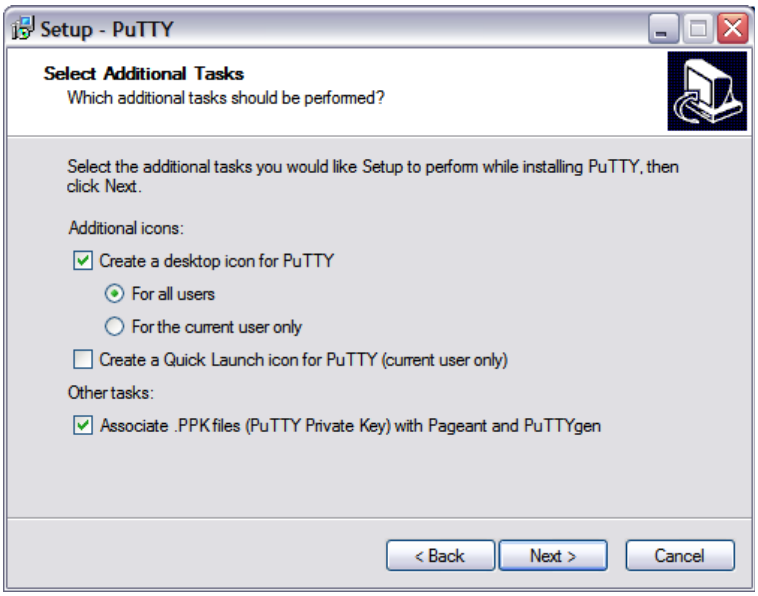
The default installation location is recommended. Click [Next].



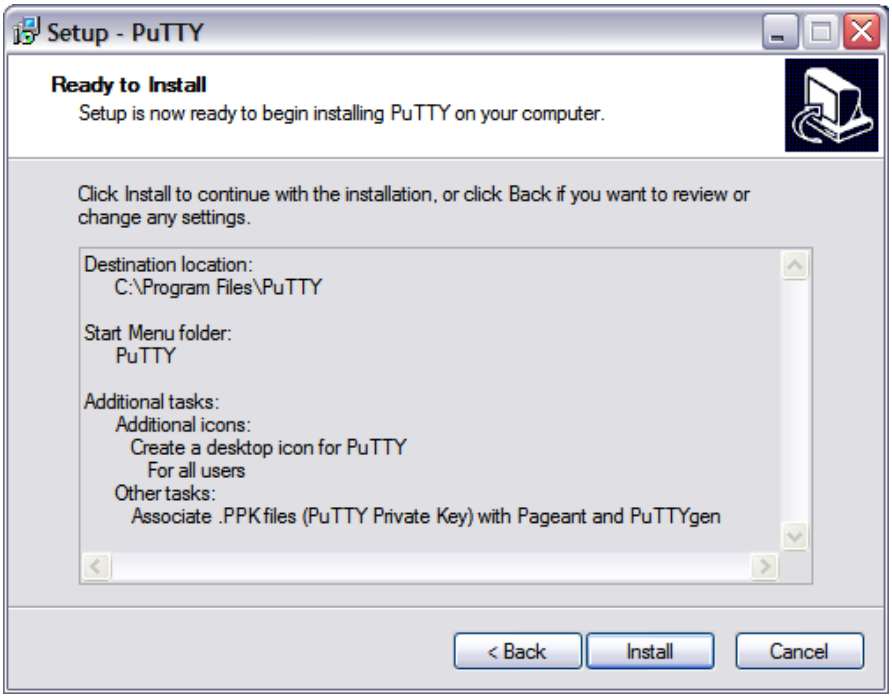
The Start Menu folder is simply the name of the sub directory under which your PuTTY shortcuts will be located. The default value should be fine. Click [Next].



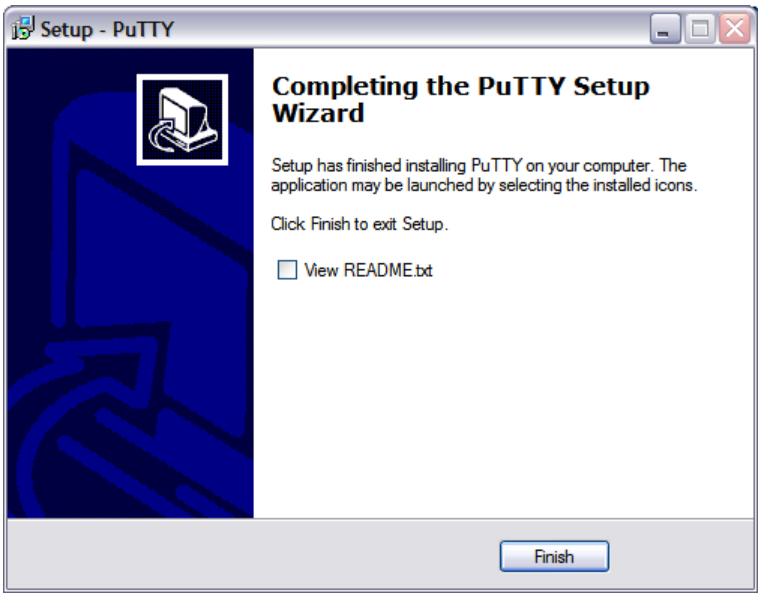
Additional tasks include creating shortcuts in various locations to make PuTTY SSH easier to access. The following option set is reasonable. Click [Next].



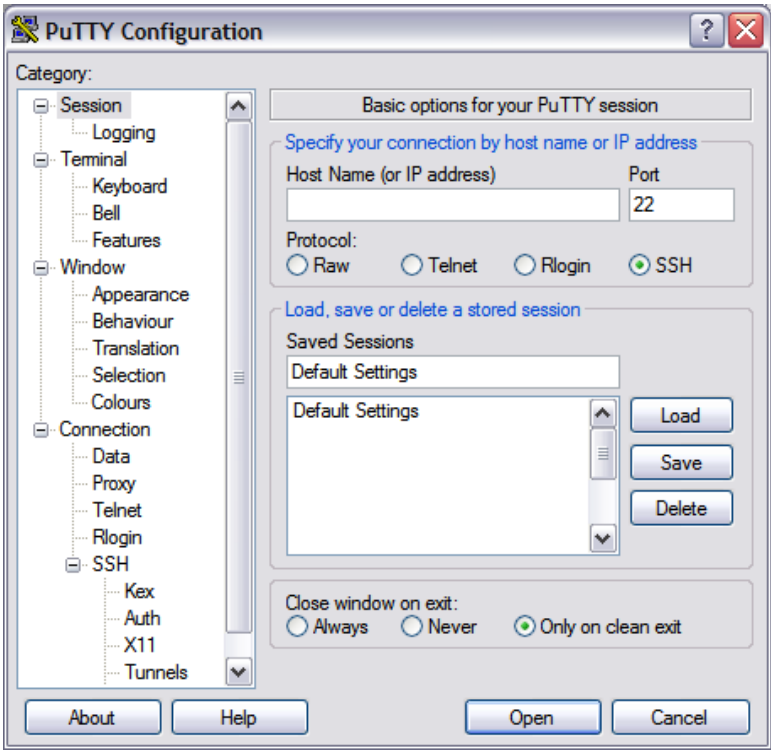
The installer will verify the settings. If everything looks as you set it in previous steps, click [Install].



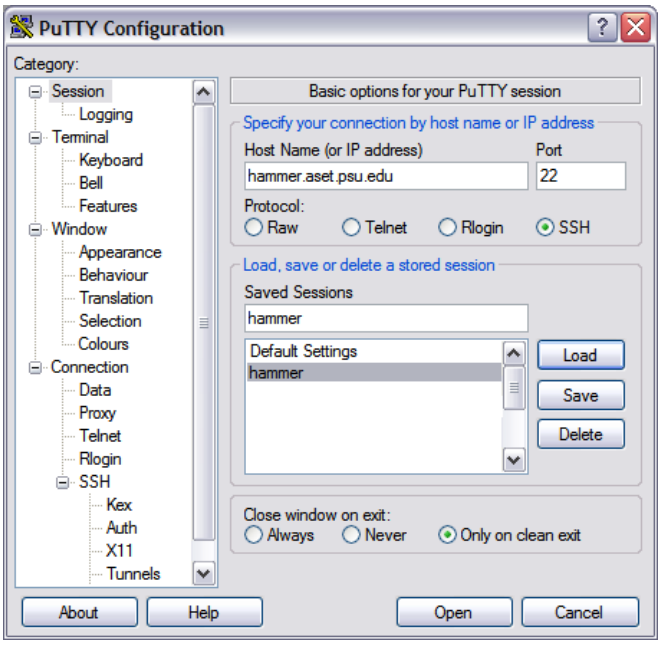
Following a brief installation status screen, you should see the following dialog indicating successful installation of PuTTY. You are now equipped to connect to our clusters remotely. Uncheck "View README.txt" and click [Finish].



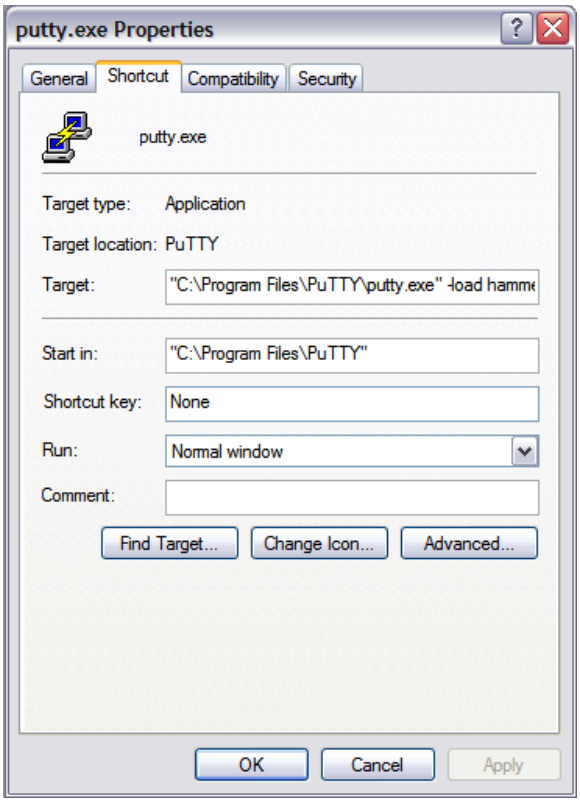
TIP 1: To avoid re-entering your preferences each time you connect to a machine, you can save your settings in a specific session, or to have the settings available at all times and applied to all future sessions by default, save your settings to "Default Settings" session. To do this, in the "Session" configuration window, type "Default Settings" into the "Saved Sessions" field, and click [Save]:



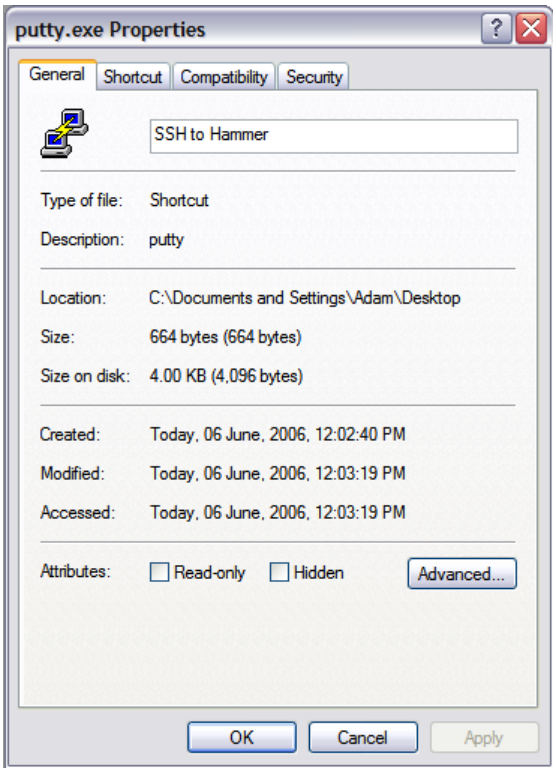
TIP 2: To save a session to a specific host, i.e. hammer.rcc.psu.edu, enter the host name in the "Host Name" field, then name the session using the "Saved Sessions" field, and click [Save]:



To access this session in the future, you can simply double-click the session name in the list-box or create a shortcut in your Start-Menu or on your desktop pointing to your PuTTY binary ("C:\Program Files\PuTTY\putty.exe" by default) using the flag "-load hammer" where you replace "hammer" with your selected session name:



You can change the shortcut's name by selecting the “General” tab and entering your desired name into the text field:



EXCEED ONDEMAND HOW-TO

OVERVIEW

Exceed onDemand is a commercial software product that allows a high-fidelity graphical connection to be made to a remote system by means of a dedicated display server. Each Hammer system has such a server running, thus allowing a connection to be made, and **Exceed onDemand is the preferred way to connect to Hammer**.

WINDOWS

1. Download the appropriate Exceed onDemand client for your Windows version, either a 36-bit (http://ics.psu.edu/wp-content/uploads/2015/08/x86.zip) or 64-bit (http://ics.psu.edu/wp-content/uploads/2015/08/x64.zip) version.
2. Navigate to your downloads folder, double-click on the downloaded zip file, and double-click on Setup.exe.
3. The OpenText Exceed onDemand Client Setup Wizard will appear. Click the Next button.
4. In the License Agreement step, select the "I accept" radio button and click Next.
5. Click Next several times to proceed through the Customer Information, Destination Folder, Custom Setup, Additional Install Options steps, accepting the default options.
6. In the Ready to Install the Product step, click Install. When the product has installed click Finish.
7. An Exceed onDemand Client shortcut should now appear on your desktop and Start Menu. Double-click shortcut to launch the Exceed onDemand client. In the window that appears, enter **hammer.rcc.psu.edu** into the Host box, your Penn State Access Account ID into the User ID box, and your Penn State Access Account password into the password box and click Login.
8. Once you have logged in, a new dialog will appear asking you to choose the Xconfig and Xstart that you would like to use. To have the Hammer desktop appear in a separate window, choose one of the **Desktop_Mode** Xconfig options and the **Gnome_Desktop.xs** Xstart option. To have the Hammer application windows seamlessly integrate with your normal Windows windows, choose the **Seamless_Mode.cfg** Xconfig option and the **Terminal.xs** Xstart option.
9. Once you have chosen your desired Xconfig and Xstart options, click the Run button. You will then be connected to Hammer.

MAC OS X

Note: If you are running OS X Mountain Lion (10.8) or newer, you must first install XQuartz, log out, and log back in so that you have a working X-Server on your machine. Previous versions of OS X included an X-Server, but versions 10.8 and above do not. XQuartz can be downloaded from the XQuartz SourceForge page at http://xquartz.macosforge.org/landing/.

1. Download the Exceed onDemand client for Mac. We host an installer at http://ics.psu.edu/wp-content/uploads/2015/08/EoDClient8-13.8.6.dmg_.zip.

2. Navigate to your downloads folder, unzip the file, double-click on the downloaded dmg file, and then the Exceed OnDemand setup package will appear. Double-click on the Exceed onDemand Client.pkg
3. The Open Text Exceed OnDemand Client Setup Installer will appear. Click the Continue button.
4. In the License step, click Continue then Agree.
5. In the Installation Type step, click Install. if the installer asks for your password, enter your local password that you use to log into your Mac
6. In the Summary step, click Close.
7. An Exceed onDemand Client application will now appear in your Applications folder. Navigate there in Finder and double-click on the application.
8. In the window that appears, enter hammer.rcc.psu.edu into the Host box, your Penn State Access Account ID into the User ID box, and your Penn State Access Account password into the password box and click Login.
9. Once you have logged in, a new dialog will appear asking you to choose the Xconfig and Xstart that you would like to use. To have the Hammer desktop appear in a separate window, choose one of the **Desktop_Mode** Xconfig options and the **Gnome_Desktop.xs** Xstart option. To have the Hammer application windows seamlessly integrate with your normal Windows windows, choose the **Seamless_Mode.cfg** Xconfig option and the **Terminal.xs** Xstart option.

   Once you have chosen your desired Xconfig and Xstart options, click the Run button. You will then be connected to Hammer.

LINUX

1. Download the Exceed onDemand client for Linux, either a 36-bit (http://ics.psu.edu/wp-content/uploads/2015/08/eodclient8-13.8.6.787-linux-i586.tar.gz) or 64-bit (http://ics.psu.edu/wp-content/uploads/2015/08/eodclient8-13.8.6.787-linux-x64.tar.gz) version.
2. Open a terminal and follow these steps:
3. tar xzf <your_downloads_folder>/eodclient-latest-linux-x64.tar.gz
4. cd Exceed_onDemand_Client_8
5. ./eodxc
6. In the window that appears, enter hammer.rcc.psu.edu into the Host box, your Penn State Access Account ID into the User ID box, and your Penn State Access Account password into the password box.
7. Next you must choose the Xconfig and Xstart that you would like to use. To have the Hammer desktop appear in a separate window, choose one of the **Desktop_Mode** Xconfig options and the **Gnome_Desktop.xs** Xstart option. To have the Hammer application windows seamlessly integrate with your normal Windows windows, choose the **Seamless_Mode.cfg** Xconfig option and the **Terminal.xs** Xstart option.

   Once you have chosen your desired Xconfig and Xstart options, click the Run button (green arrow). You will then be connected to Hammer.

If you have trouble with any of the above steps, please contact iask@ics.psu.edu.

OPENMPI USAGE INSTRUCTIONS

OpenMPI is an MPI-1 and MPI-2 compliant MPI (Message Passing Interface) library. MPI is a standard API used for distributed memory parallel computing.

LISTING AVAILABLE OPENMPI VERSIONS

There are different versions of OpenMPI for different compiler families. You can find the available versions by running **module avail openmpi**. In the following example, there are openmpi versions for three different compiler families: GNU, Intel, and PGI (Portland Group). The versions listed are all 1.4.0.

```
[jwh128@cyberstar ~]$ module avail openmpi

------------------- /usr/global//Modules/3.2.6/modulefiles -------------------
openmpi/gnu/1.4.0   openmpi/intel/1.4.0   openmpi/pgi/1.4.0
```

LOADING AN OPENMPI VERSION INTO YOUR ENVIRONMENT

The latest version of OpenMPI for a particular compiler family can be loaded with the command **module load openmpi/<compiler family>**. In the following example, the latest version for the GNU compilers is loaded. Note that the example module command loads all prerequisites for the GNU version of OpenMPI. This can be seen in the **module list** commands in the example below.

```
[jwh128@cyberstar ~]$ module list
No Modulefiles Currently Loaded.
[jwh128@cyberstar ~]$ module load openmpi/gnu
[jwh128@cyberstar ~]$ module list
Currently Loaded Modulefiles:
  1) gmp/4.3.1          3) ppl/0.10.2         5) gcc/4.4.2
  2) mpfr/2.4.1         4) cloog-ppl/0.15.7   6) openmpi/gnu/1.4.0
```

COMPILING MPI SOURCE CODE WITH OPENMPI

OpenMPI contains compiler wrappers that add all of the necessary include and linking flags for compiling MPI code. The compiler wrappers are listed in the following table.

MPI Compiler Commands

Programming Language	Compiler Wrapper
C	mpicc

C++	mpiCC
Fortran77	mpif77
Fortran90	mpif90

The following example shows how to compile the C source code *hellompi.c* into the MPI executable *hellompi* using the mpicc command.

```
[jwh128@cyberstar mpi]$ mpicc hellompi.c -o hellompi
```

RUNNING OPENMPI EXECUTABLES THROUGH PBS

MPI executables need to be launched with the commmand **mpirun**. The mpirun command takes care of starting up all of the parallel processes in the MPI job. The following is an example PBS script for running an OpenMPI job across four processors. Note that the proper OpenMPI version needs to be loaded before running the job.

```
#PBS -l nodes=4:ppn=1
#PBS -l walltime=2:00:00
#PBS -j oe

cd $PBS_O_WORKDIR

module load openmpi/gnu
mpirun ./hellompi
```