Exercise 1: Connecting to BW using ssh:

NOTE: $ = command starts here, ␣=means one space between words/characters.

Before you login to the Blue Waters system, make sure you have the following information ready with you:

Username: `tra0xx`

Password: `XXXXXXXXXXX`

Hostname: `bwbay.ncsa.illinois.edu`

Read the manpage for SSH:

```
$ man␣ssh
```

First time login for UNIX (e.g. Mac OS X) / Linux users:

If you use a UNIX (e.g. Mac OS X) or Linux machine, the `ssh` utility is generally available through any command line (Terminal). Follow the steps for logging into the Blue Waters system:

1. On Mac, open the Terminal application.
2. `$ ssh␣username@bwbay.ncsa.illinois.edu` **[press enter]**

To access the system, you will first need to answer a couple of RSA (cryptosystem) or password challenges, accept system usage policies/agreements (on first login only), and then you are logged into the system. A series of system status and MOTD ('Message of the Day') messages may display, after which you are placed in your home directory on a login node. With successful login to Blue Waters, you should see a prompt similar to the following:

```
instr004@h2ologin1:~>
```

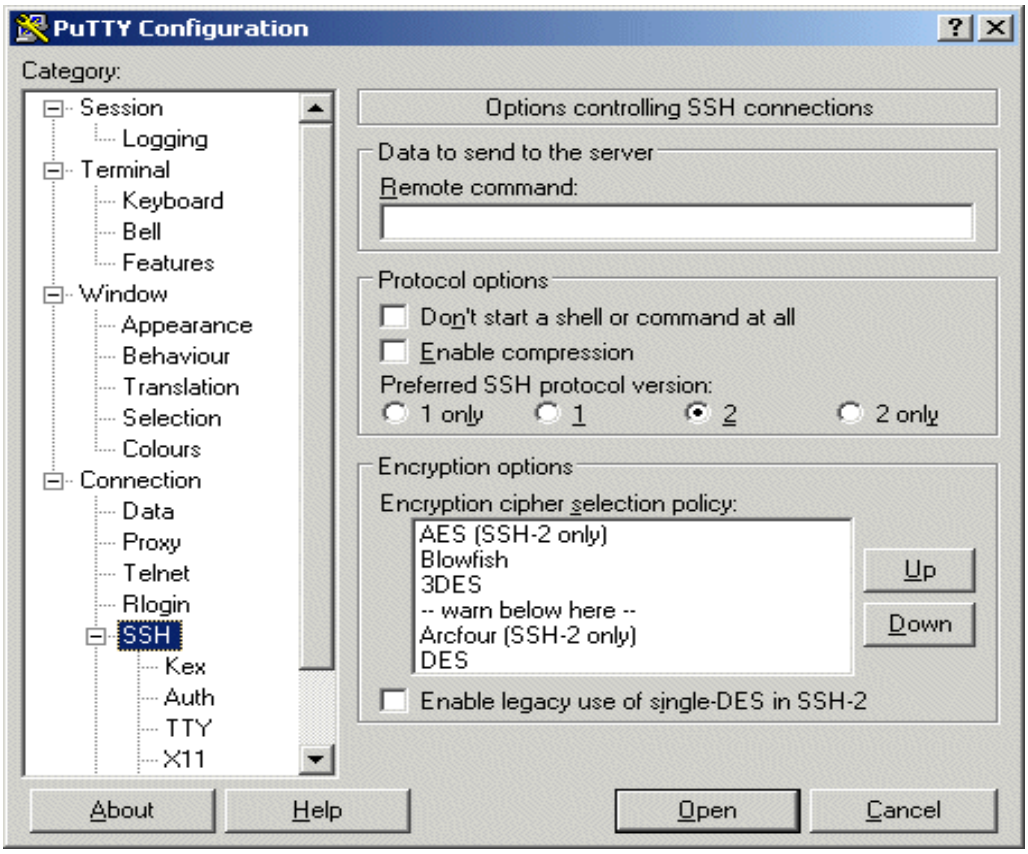
Your username is placed before the @ symbol, and the login node name is placed after the @ symbol. Notice that the login node name is different than the hostname you specified with the `ssh` command (is this what you expected?). This is because Blue Waters has users connect to a gateway node called `bwbay` before routing the user to one of the login nodes.
Each time you log in, you may notice a different number at the end of the name of the login node -- this is because Blue Waters has multiple login nodes. However, each login node will have access to your files, no matter which one you are logged into.

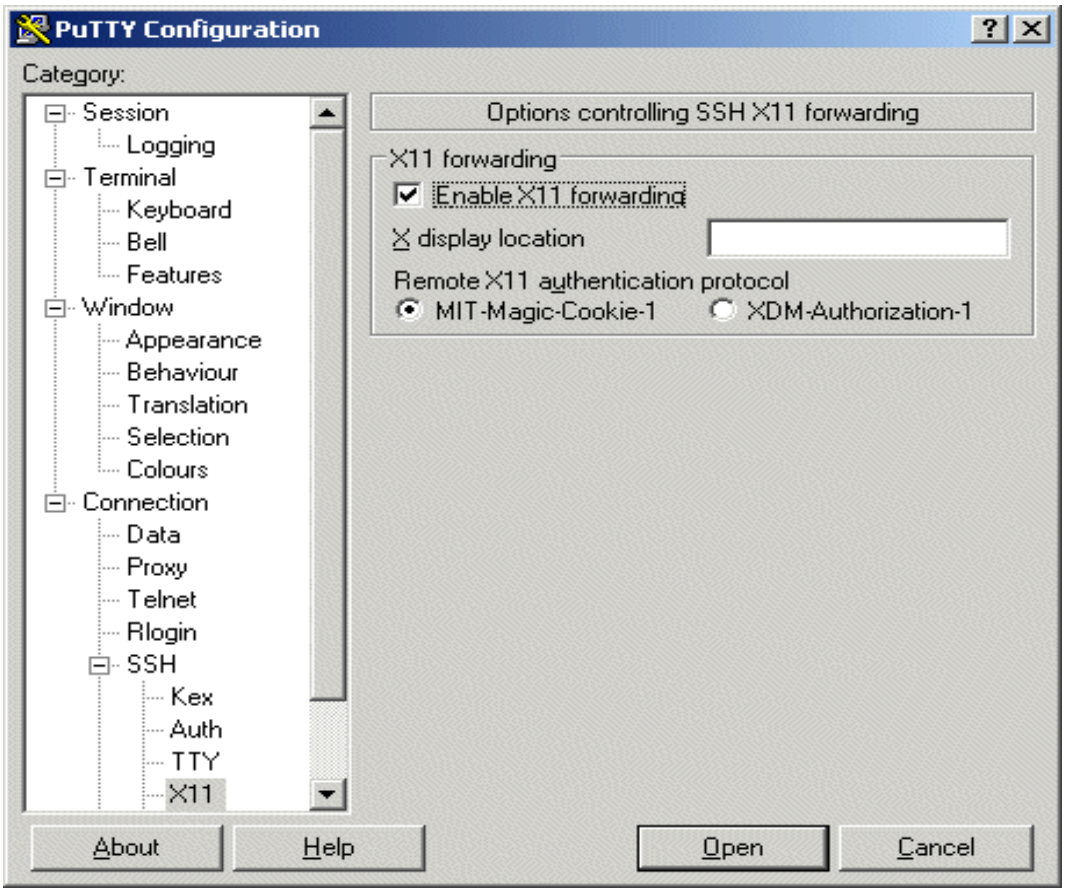
First time login for Windows OS users:

If you use a Windows personal computer, you first need to obtain and install a client program for your system that supports SSH protocol 2, such as PuTTY.

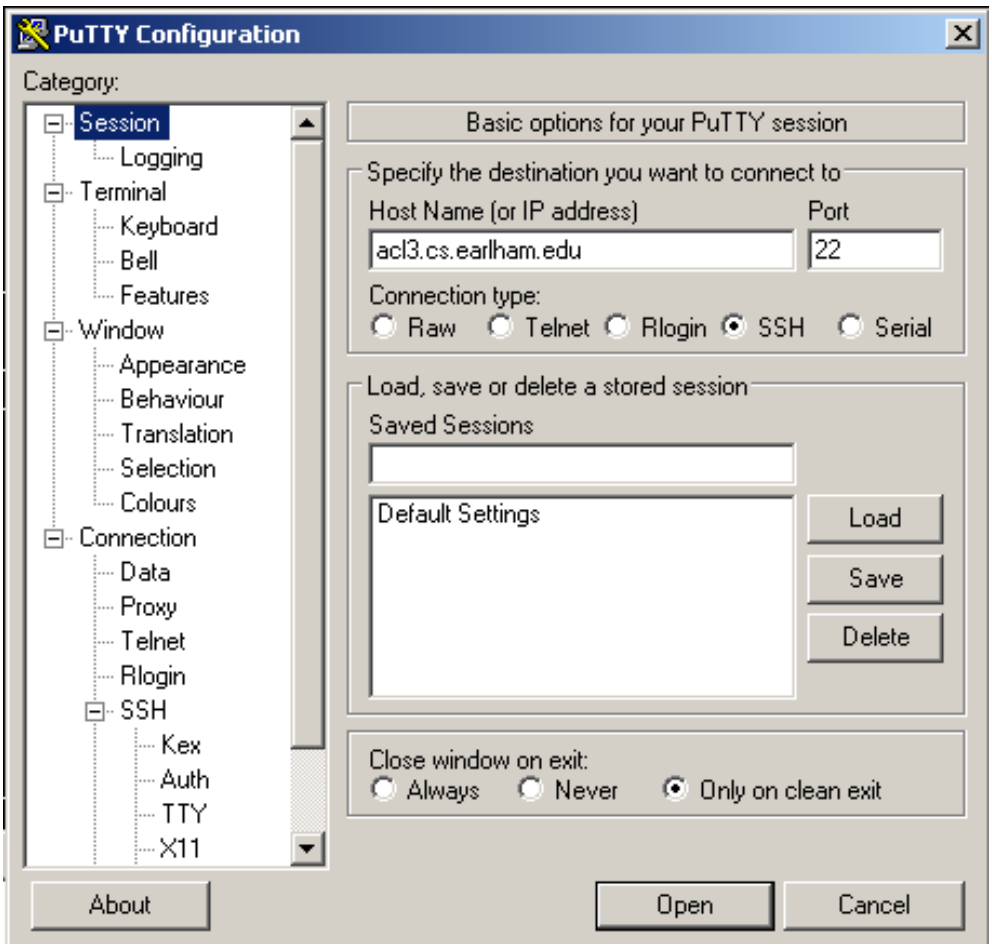
The official PuTTY download page is here:
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

1. You may need to configure your client to support SSH protocol 2 and X11 forwarding. For example, if you are using PuTTY, you may need to click **SSH** in the left pane to see the preferred SSH protocol version:





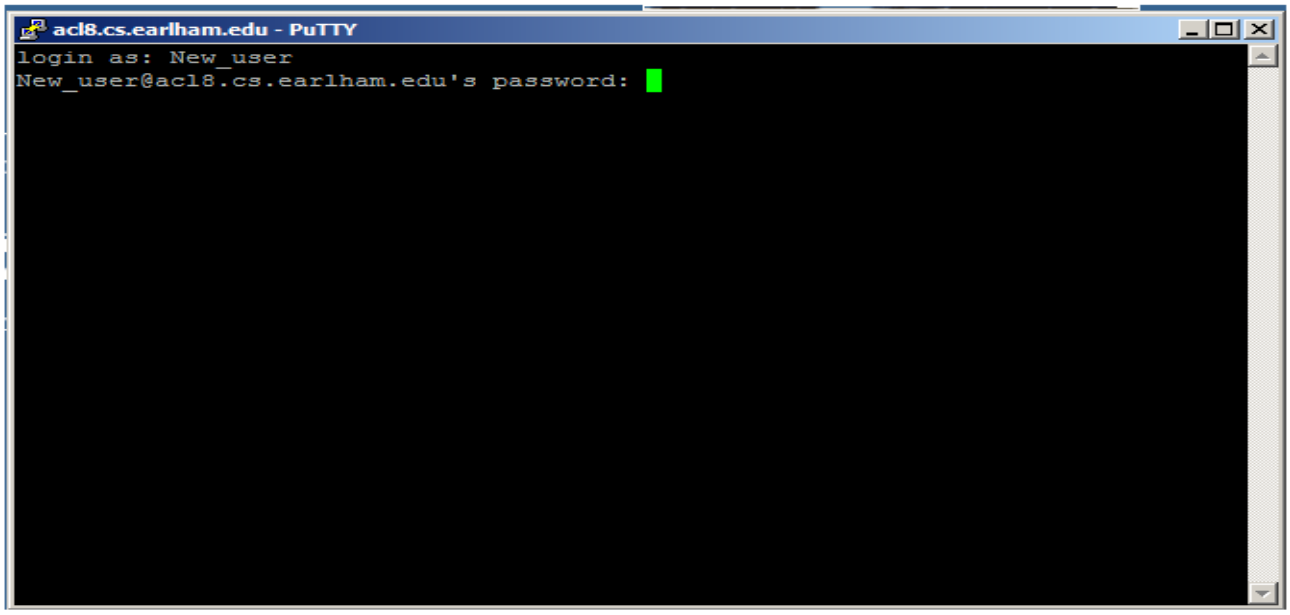
2. Click **X11** in the left pane and **check the box to enable X11 forwarding**:



3. Then click **Session** in the left pane to return to the **Basic options** window.

Enter the hostname (`bwbay.ncsa.illinois.edu`) in the **Host Name** field and click the **Open** button to begin your SSH session.

You need to enter your **username** and answer a couple or RSA or password challenges, system usage policy/agreements (on first login only), and then you are logged into the system.



A series of system status and MOTD (message of the day) type messages may display, after which you are placed in your home directory on a login node.

Read the policy:

Accept the policy agreements:

Answer to questions:

- Q1
- Q2
- Q3

Make sure you see this screen:

Once you have this prompt showing, that means you are connected to Blue Waters.

YEAAAA You just got POWER DIDNT YOU.

```
Last login: Wed Apr 16 15:03:19 2014 from bwedge.ncsa.illinois.edu

   ___  __                _      __      __
  / _ )/ /_ _____        | | /| / /__ _/ /____ _______
 / _  / / // / -_)       | |/ |/ / _ `/ __/ -_) __(_-<
/____/_/\_,_/\__/        |__/|__/\_,_/\__/\__/_/ /___/
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~
Batch and Scheduler configuration.
  Queues:  normal (default), high, low
  Features:  "xe" (default), "xk", "x" (xe or xk non-specific)
  30 min default wall time,
```

```
  -lnodes=X:ppn=Y syntax supported.

All SSH traffic on this system is monitored.

04-15 14:36 Login node resource protection policy enabled. Processes
with excessive cputime consumption will be terminated. Email
notification will be sent to the process owner if this occurs. 01-13
22:46 There will be changes in the default versions for some modules
in the Blue Waters programming environment after the system returns
from the maintenance. No recompilation is required for most cases.
Users of OpenACC applications using PGI 13.6.0 (current default) will
need to either explicitly add the PGI 13.6.0
compiler in their environment or rebuild their application with the
PGI 13.10.0 compiler which will be the default after 1/21. Default
CUDA support for the PGI compiler moves from CUDA4 to CUDA5 with PGI
13.9 or higher.
04-22 10:48 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Blue Waters return to service will be delayed from April
22, 6:00pm to April 23, 06:00am due to some unforeseen technical
issues. Login and file system access will be enabled later today at
7:00pm. Job submission and query will be enabled at 7:00pm as well
(only via qsub and qstat; showq may be unavailable). To help
accommodate for the inconvenience caused by an unexpected delay and
to prime the system for an off-hours return, jobs that will start
within the first 24 hours after return to service (currently at April
23, 06:00am), will be zero charge.
Usage Tip: shorter wallclock jobs will have greater chance of
being backfilled sooner, possibly within the zero charge period.
++++++++++++++++++++++++++++++++++++++++++++++++++
Questions?  Mail help+bw@ncsa.illinois.edu to create a support
ticket.For known issues:  https://bluewaters.ncsa.illinois.edu/known-
issues
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~/usr/bin/xauth:  creating new authority file
/u/training/instr004/.Xauthority
2.0 Learning about Important Linux/Unix Commands:
```

Exercise 2: Learning Linux Command Line

Here is what we want you to learn from this exercise:

1. Get comfortable with using command line Linux
2. Navigate the file systems and see what files and folders (from now on we will always call folders directories, the Linux way) you own
    a. How to change ownership and permissions on files/directories
3. Different ways to see content of a file
4. How to copy/move files around in the file system
5. Who else is on the system, (this might give you an idea why not to run programs on login nodes, because you are not the only one using it)
6. And of course getting comfortable with mighty man (manual) pages!!

Using man pages, write 1-3 sentences for each of the commands below and important options you can pass to each command.

Command Name:	**Description + Options**
1. `ls`	:
2. `cd`	:
3. `cp`	:
4. `pwd`	:
5. `cat`	:
6. `make`	:
7. `who`	:
8. `quota`	:
9. `less`	:
10. `more`	:
11. `touch`	:
12. `rm`	:
13. `mkdir`	:
14. `module`	:
15. `wget`	:
16. `man`	:
17. `grep`	:
18. `chmod`	:
19. `chown`	:
20. `ssh`	:

Other sources for Linux command line:
https://newton.utk.edu/bin/view/Main/LinuxCommandLineBasics

Some important Terms:

What is Path/PATH?

Ever computer operating system comes with a “File System”. The file system manages the files and directories in hierarchical/tree like structure. Basically you can have a one big folder inside it you can have many other subfolders and files. Inside the subfolders you can have many other sub-sub-folders and files. For example, I have a folder call mobeen1, inside it I have another folder call mobeen2, inside that I have another folder call mobeen3 and inside that I have a file call holly_cow.txt.

If I want to share this file with Aaron, I will need to tell him where the file is located. That location is know as path. So the path to the files will look something like this:

```
$ /home/mobeen1/mobeen2/mobeen3/holy_cow.txt
```

Exercise 3: Transferring files to/from Blue Waters

We will teach you three ways to transfer your files from one place to another, one user to another, and so on.

1. Local system transferring:

You have already learned in the manpage exercise how to copy files from one directory to another or one place to another under the same user using the `cp` command. You could also use the `cp` command to copy files/directories from another user's home directory.

Example:

```
$ cp -r ~instr004/BW_Institute/Examples .
```

This command will copy the `Examples` directory and, with the `-r` option, recursively all the files within it.

2. Copy files from a remote machine:

If you have access to your a server machine at home, office, or school, you can copy files from/to it using the `scp` command utility.

The syntax for the scp command is a bit different depending on whether you are copying to or from the system:

To the system:

```
$ scp -r File_name username@hostname:dest_path
```

Example:

```
$ scp -r Examples instr004@bwbay.ncsa.illinois.edu:~/public
```

This command will copy the `Examples` directory and all its contents to the `instr004` user's `public` directory on the `bwbay.ncsa.illinois.edu` machine.

Note that if you want to leave out the `dest_path`, you still need to include the colon (`:`).

From the system:

```
$ scp -r username@hostname:path_source/File_name dest_path
```

Example:

```
$ scp -r instr004@bwbay.ncsa.illinois.edu:~/Examples \
~/BW_Projects .
```

This command will copy the `Examples` directory with all its files from the `bwbay` host to the `BW_Projects` directory on my computer under my home directory.

NOTES: break up into groups, each will run program for some number of processors

Exercise 4: Running Scientific models on Blue Waters

Most of the modules we have here are part of the BCCD (Bootable Cluster CD) operating system, LittleFe and Shodor initiative for learning parallel programming/computing through scientific examples. The modules are from different areas of science such as: physics (n-body simulation of galaxies), biology (disease model, molecular dynamics), mathematics (Sieve, Area Under-Curve, Life), etc...

Here is what we want you to learn from this exercise:

1. How to run programs/applications on Blue Waters supercomputer:
    a. Each time run the program on different number of nodes/cores
    b. Before you run the program make an educated prediction how the program will behave if you run it each time on different number of nodes/cores
    c. When running the program pay attention to the details and observe what's happening.
    d. Was the final outcome similar to what you predicted or something very unexpected, and why?
2. How to ask for resources that you can use to run your programs?
    a. Don't ask for what you don't need
3. Learn how different scientific models behave when you allocate them different sources

Asking for Resources:

Example:
Interactive jobs:

```
$ qsub -I -l nodes=1

$ qsub -I -l nodes=2:ppn=32:xe -l walltime=00:30:00
```

Compiling all the modules:

You should have copied by now the BW_Institue directory to your accounts. Make sure it has the Examples directory with all the modules in it.

1. Go to BW_Institute/Examples directory:
   ```
   $ cd Examples
   $ ls -al
   ```
2. Compile all the modules. Make sure there is no error during compilation:
   ```
   $ make build-all NO_X11=1
   ```
3. Running GalaxSee Model:
   ```
   $ time aprun -n 1 ./GalaxSee.cxx-mpi 1000 100 1000 0
   $ time aprun -n 2 ./GalaxSee.cxx-mpi 1000 100 1000 0
   $ time aprun -n 4 ./GalaxSee.cxx-mpi 1000 100 1000 0
   $ time aprun -n 5 ./GalaxSee.cxx-mpi 1000 100 1000 0
   ```

```
$ time aprun -n 8 ./GalaxSee.cxx-mpi 1000 100 1000 0
$ time aprun -n 9 ./GalaxSee.cxx-mpi 1000 100 1000 0
$ time aprun -n 16 ./GalaxSee.cxx-mpi 1000 100 1000 0
$ time aprun -n 32 ./GalaxSee.cxx-mpi 1000 100 1000 0
```

4. Running Life Model

"Life" is a nearest-neighbors model. Simple rules give rise to complex, chaotic behavior.

a. Running Life OpenMP Version:

```
$ export OMP_NUM_THREADS=2
$ time ./Life.c-openmp -w 10 -n 200 -i 5
$ export OMP_NUM_THREADS=4
$ time ./Life.c-openmp -w 10 -n 200 -i 5
$ export OMP_NUM_THREADS=8
$ time ./Life.c-openmp -w 10 -n 200 -i 5
$ export OMP_NUM_THREADS=16
$ time ./Life.c-openmp -w 10 -n 200 -i 5
$ export OMP_NUM_THREADS=32
$ time ./Life.c-openmp -w 10 -n 200 -i 5
```

b. Running Life MPI Version:

```
$ time aprun -n 2 ./Life.c-mpi -w 10 -n 200 -i 5
$ time aprun -n 4 ./Life.c-mpi -w 10 -n 200 -i 5
$ time aprun -n 6 ./Life.c-mpi -w 10 -n 200 -i 5
$ time aprun -n 8 ./Life.c-mpi -w 10 -n 200 -i 5
$ time aprun -n 10 ./Life.c-mpi -w 10 -n 200 -i 5
$ time aprun -n 16 ./Life.c-mpi -w 10 -n 200 -i 5
$ time aprun -n 32 ./Life.c-mpi -w 10 -n 200 -i 5
```

5. Running GalaxSee-v2:

This module is bit different than the previous version of GalaxSee. The executable/ program in this model can take large number of arguments that changes the behavior of the model. Therefore, to make it easier we will make use of an input-file (simple.gal). If you wanted to increase the problem size, or make other changes to the behavior of the model please make those changes in the input file called simple.gal.

```
$ time aprun -n 1 ./GalaxSee-v2.cxx-mpi simple.gal
$ time aprun -n 2 ./GalaxSee-v2.cxx-mpi simple.gal
$ time aprun -n 4 ./GalaxSee-v2.cxx-mpi simple.gal
$ time aprun -n 8 ./GalaxSee-v2.cxx-mpi simple.gal
$ time aprun -n 16 ./GalaxSee-v2.cxx-mpi simple.gal
$ time aprun -n 24 ./GalaxSee-v2.cxx-mpi simple.gal
$ time aprun -n 32 ./GalaxSee-v2.cxx-mpi simple.gal
```