

A guide to using the CS Department Lab Machines by Sam Laberge.

https://www.cs.utexas.edu/~slaberge/docs/lab_machines/

The CS Department Lab Machines

This contains a guide for connecting to and using the [CS department Linux machines](#).

Learning how to connect to and use the lab machines will be extremely important for CS314 and your future CS classes. In 314, the TAs will be compiling and running your code on the lab machines. Therefore, it is *your responsibility* to make sure your code runs as you intended on the lab machines. **We do not grade your code based on how it ran on your personal machine.**

Getting started with using the Lab machines can actually be a tricky process especially if this is your first time using SSH or Linux. Try to set aside some time early in the semester to go through this process and make sure to reach out for help if you get stuck! There are plenty of resources online and you can also ask one of your TAs.

Getting Connected

The UT VPN

Using the UT VPN will make it appear as though you are connected to UT's on-campus internet. This can be very useful even outside of CS. If you are trying to access the UT library, online academic journals, or other university resources, you may need to use the VPN when off campus.

Here, we will need to use the VPN to allow us to connect to the CS Department's Lab Machines.

Foreword on VPNs

When you connect to a VPN, all of your Internet traffic will be routed through that VPN service. This is true of all VPN services, including UT's. Although I doubt UT has any nefarious intents, realize that VPN services could get a lot of information on you based on your internet activity.

Therefore, as a general security principle, only use a VPN when you absolutely have to. In the case of UT's VPN, you should only use it while you are connected to a CS machine, trying to access library documents, etc. Disconnect from the VPN when you are done.

In the case of the CS lab machines, only use the VPN to set up your SSH keys, and then you will not have to connect to the VPN afterwards to connect to the lab machines.

Prerequisites

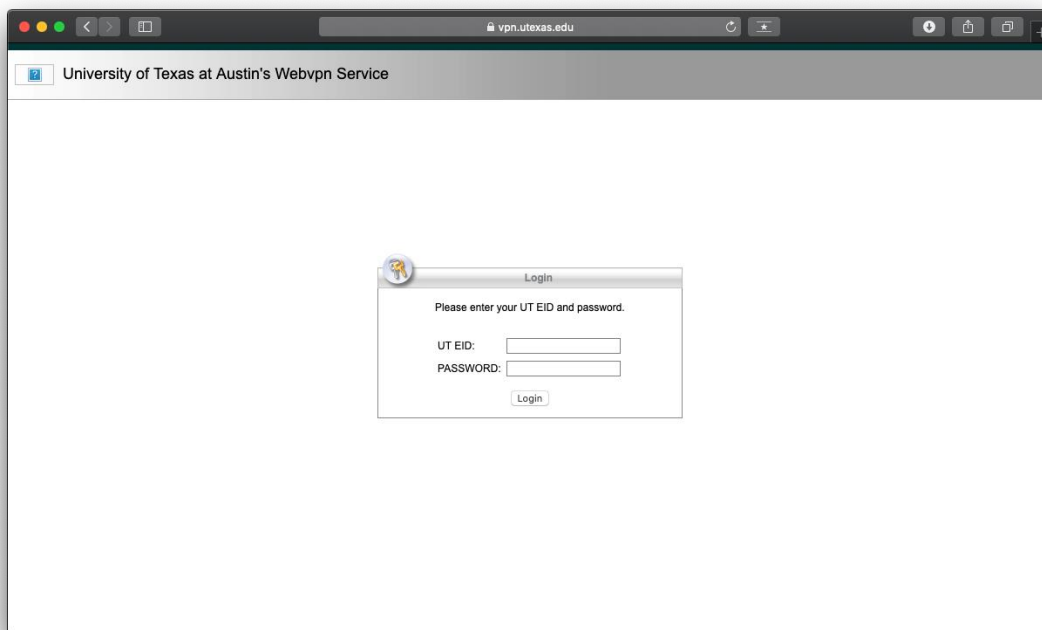
- Make sure you have DUO two-factor authentication set up. You can do this by following [this guide](#).

Setting up the VPN

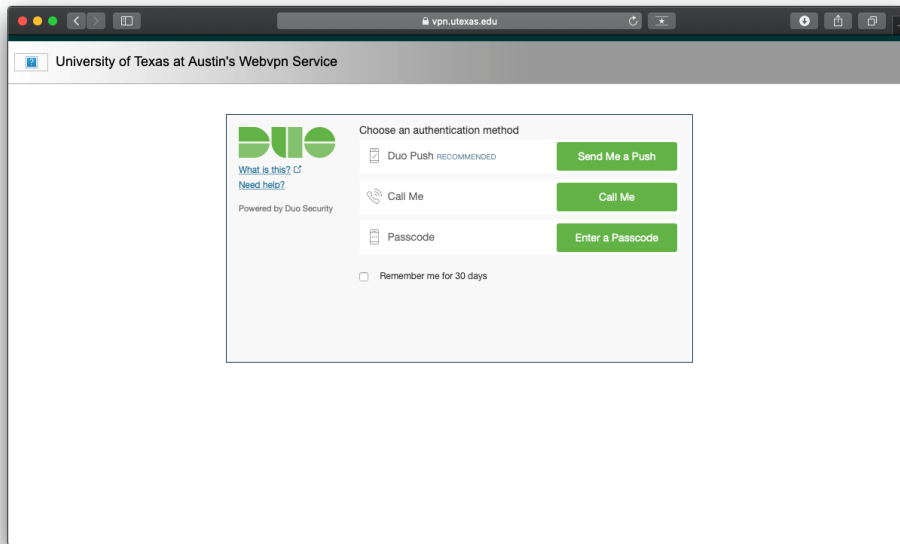
These steps should be the same for Windows, Mac, and Linux.

Downloading the VPN Software

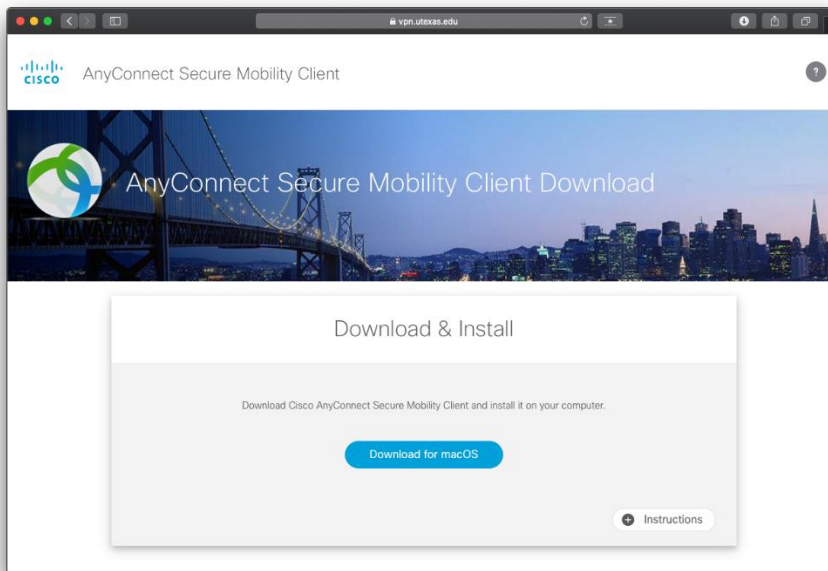
Visit vpn.utexas.edu and log in using your UT EID.



It will then ask you to authenticate using DUO. Choose an authentication method to proceed.



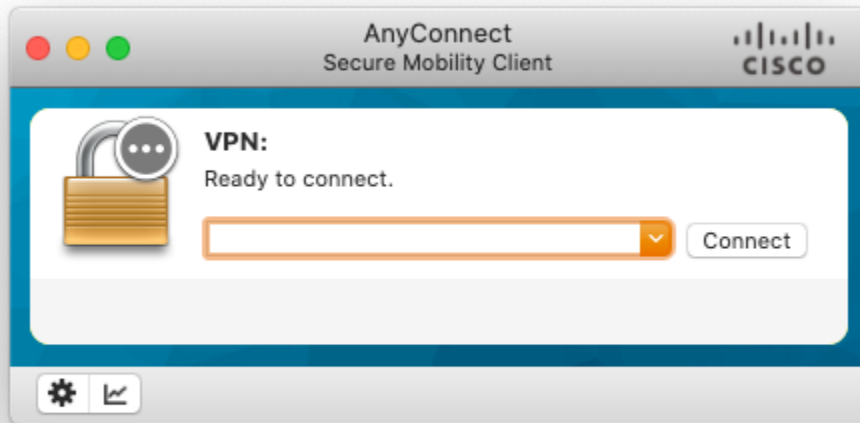
You will then need to install a piece of software by Cisco which will set up the VPN on your device.



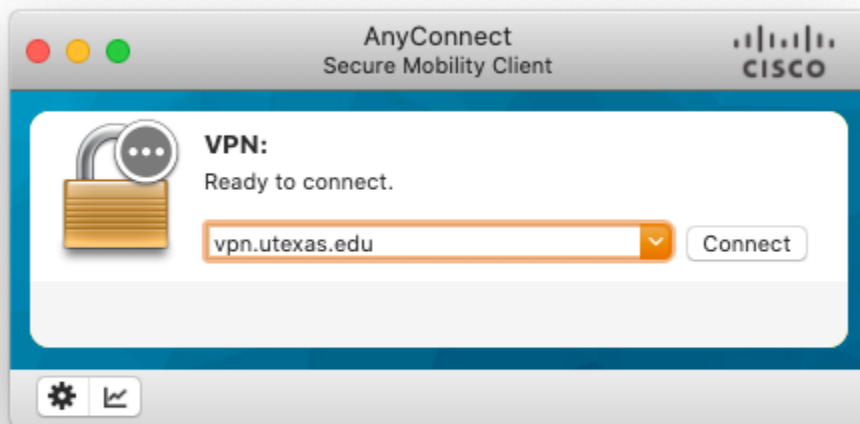
Click on the “Instructions” button if you would like instructions on how to install the software.

Connecting to the VPN

Once you are finished installing, open the “Cisco AnyConnect Secure Mobility Client”. You should see the following window.



Type `vpn.utexas.edu` in the field and click connect.



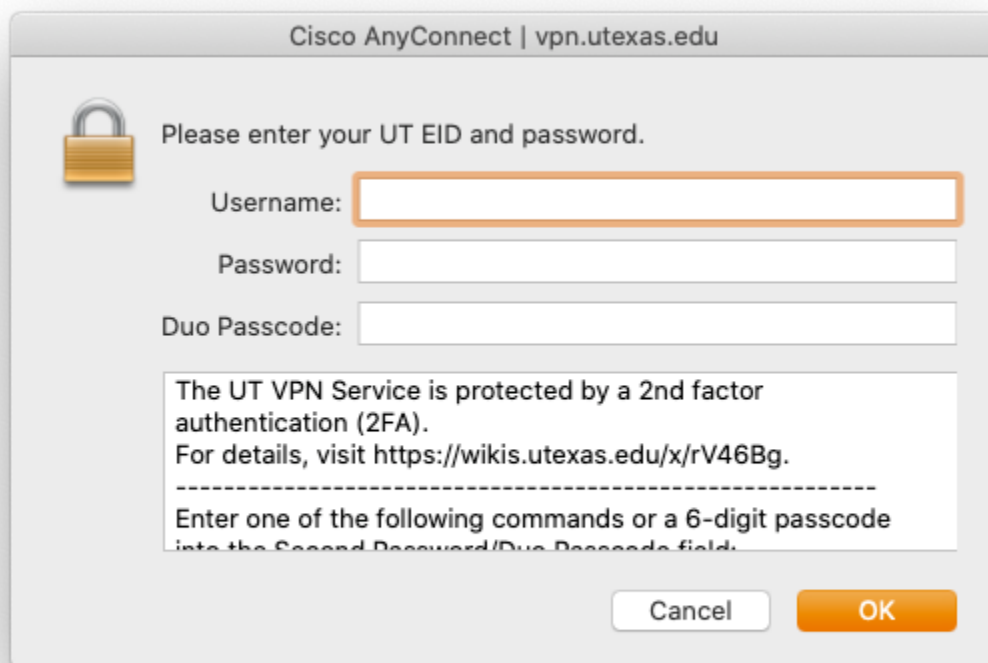
You will then be shown an authentication window.

In the Username field, enter your UT EID.

In the Password field enter the password associated with your UT EID.

In the last field, Duo passcode, you can enter any of the following options:

1. `push` - This will send a push notification to your authorized DUO device.
2. `sms` - This will send an SMS message to your DUO-authorized phone number.
3. `phone` - This will call your DUO-authorized phone number.



The image shows a Cisco AnyConnect login dialog box titled "Cisco AnyConnect | vpn.utexas.edu". It features a lock icon and the text "Please enter your UT EID and password." Below this are three input fields: "Username:", "Password:", and "Duo Passcode:". A text box below the fields contains the following text: "The UT VPN Service is protected by a 2nd factor authentication (2FA). For details, visit <https://wikis.utexas.edu/x/rV46Bg>. ----- Enter one of the following commands or a 6-digit passcode into the Second Password/Duo Passcode field:". At the bottom right are "Cancel" and "OK" buttons.

Click Ok and then, with your selected authentication method, provide two-factor authentication to connect.

If everything was successful, you should see a message indicating you successfully connected to the VPN!



Connecting with SSH

We will connect to the Lab Machines using a protocol called SSH (for **Secure Shell**). SSH allows us to use a computer remotely by sending commands to it and then receiving the resulting output. For CS314, this means you will be able to run and test your assignments on the lab machines.

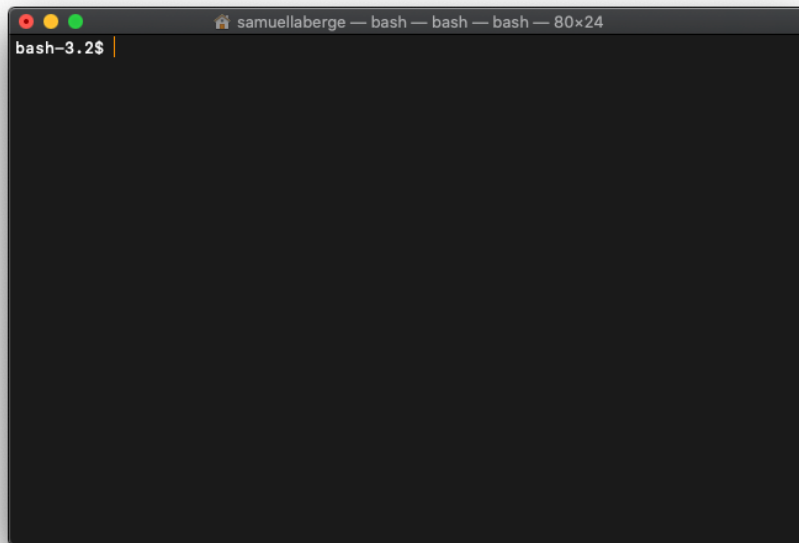
Prerequisites

- Make sure you know your CS account username and password
- Find an available CS machine to connect to with [this link](#)
- If off-campus, make sure you are connected to the [VPN](#) or have set up your [SSH keys](#)

The SSH Command

Firstly, you'll need to open your terminal application of choice. On macOS, the default pre-installed terminal is called `Terminal`. On Windows, the pre-installed `Command Prompt` will work, too.

You should now have a blank prompt which looks something like this:

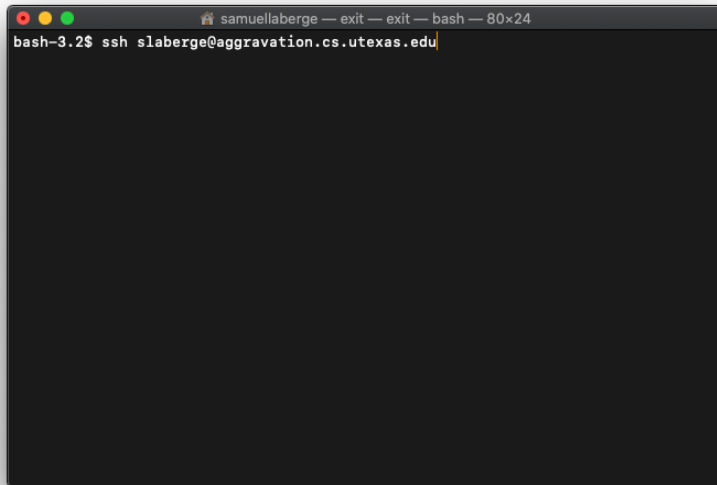


Now, we can issue the SSH command to open a connection with one of the UTCS Lab machines:

```
ssh <CS Username>@<Machine Name>.cs.utexas.edu
```

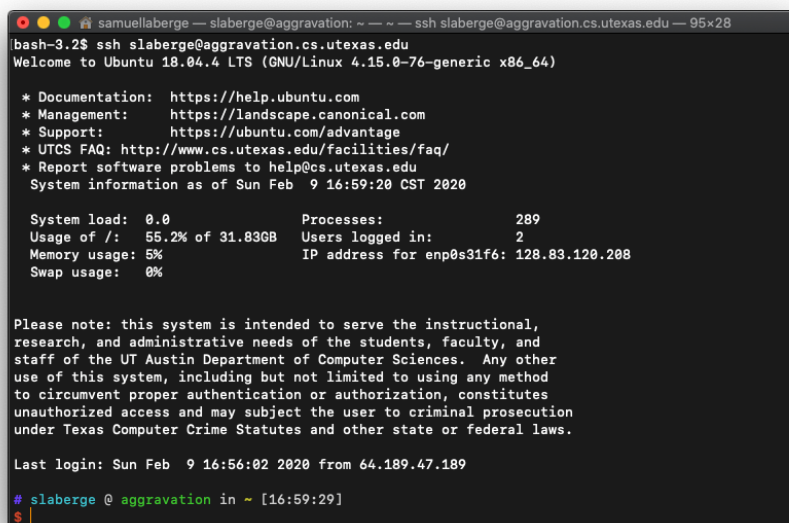
My username is `slaberge` and I'll be connecting to the `aggravation` machine. So the command I'll use looks like:

```
ssh slaberge@aggravation.cs.utexas.edu
```



Press enter to issue the command. You will be asked to enter your CS account password. If you do not remember it, use [this link](#). If you set up SSH keys, you may be asked for your key's passphrase.

If you are on Windows and the `ssh` command could not be found, you may need to enable it. Try following [this tutorial](#).



The prompt you see will not look exactly like the one above, but if you see the name of the machine you connected to in the prompt, then you successfully connected!

Once Connected

All commands that you run from now on are actually being executed on the lab machine, not your personal computer. The output of the commands is then sent back to your computer for you to see them.

To disconnect and end the SSH session, simply enter the following command:

```
logout
```

SSH Keys

Setting up SSH Keys will allow us to connect to the lab machines from off-campus without needing to use the VPN.

SSH Keys are also a more secure form of authentication than just passwords. So, if you ever set up a server which is accessible over the internet (say, for a website) in the future, make sure to use SSH Keys for authentication.

Prerequisites

- Make sure you can connect to a lab machine over SSH. This may require using the UT VPN if off-campus.

Creating a Key Pair

First, we need to create a public/private key pair on our local machine. The public key will be sent to the lab machine whereas the private key should remain on your personal computer and never be shared with anyone.

To create a key, run the following command in your terminal or command prompt:

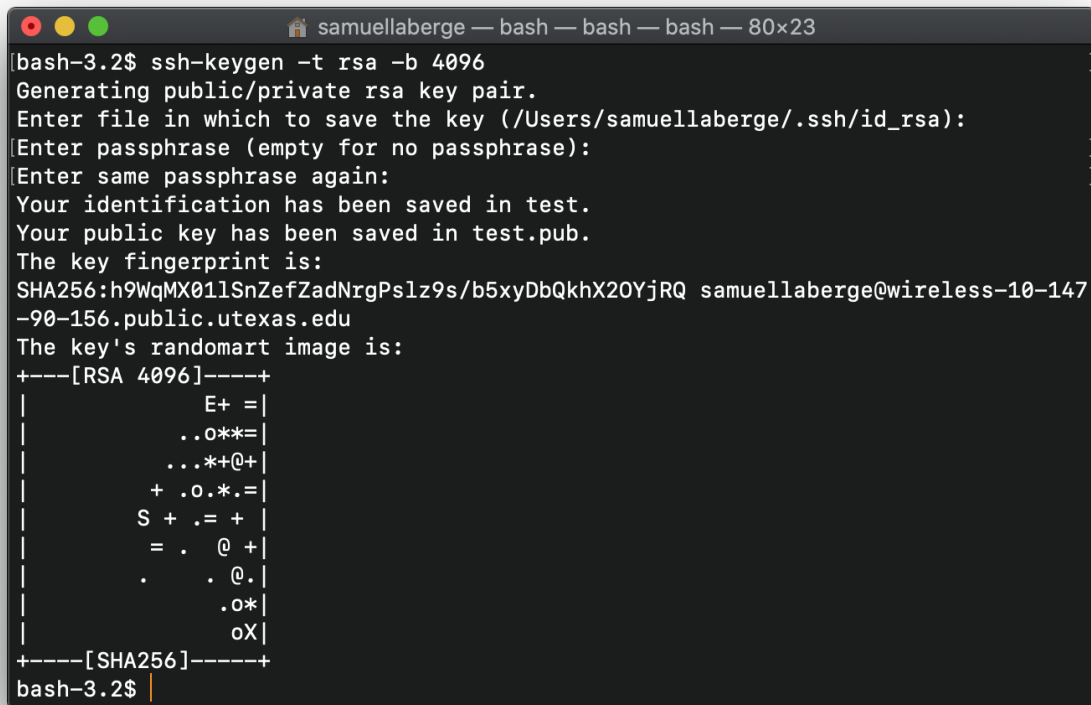
```
ssh-keygen -t rsa -b 4096
```

You will then be prompted to choose a location to store the keys. It is recommended to use the default location, so just press Enter without entering anything at the prompt.

You will then be prompted to enter a passphrase for this key. The CS Department strongly recommends using a passphrase alongside your keys, although it is not necessary. If you choose to enter a passphrase, you will be prompted to enter it anytime you log into the lab machines from off campus. If you do not want to use a passphrase, press Enter without entering one.

You will then be prompted to confirm your passphrase.

The output should look similar to below.

A terminal window titled 'samuellaberge — bash — bash — bash — 80x23' showing the execution of 'ssh-keygen -t rsa -b 4096'. The output includes prompts for a passphrase and confirmation, followed by the generation of a public/private key pair. It displays the key fingerprint as 'SHA256:h9WqMX01lSnZefZadNrgPslz9s/b5xyDbQkhX20YjRQ samuellaberge@wireless-10-147-90-156.public.utexas.edu' and a randomart image for the RSA 4096 key. The terminal ends with the prompt 'bash-3.2\$' and a cursor.

```
bash-3.2$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/samuellaberge/.ssh/id_rsa):
[Enter passphrase (empty for no passphrase):
[Enter same passphrase again:
Your identification has been saved in test.
Your public key has been saved in test.pub.
The key fingerprint is:
SHA256:h9WqMX01lSnZefZadNrgPslz9s/b5xyDbQkhX20YjRQ samuellaberge@wireless-10-147-90-156.public.utexas.edu
The key's randomart image is:
+---[RSA 4096]---+
|                 E+  =|
|                ..O**=|
|               ...*+@+|
|              + .O.*.=|
|             S + . = + |
|            = . @ + |
|           . . @ . |
|          .O*|
|         oX|
+-----[SHA256]-----+
bash-3.2$
```

By default, the public key is stored in a file called `id_rsa.pub`. On macOS/Linux this file is in the hidden directory called `.ssh` in your home directory. On Windows it is located in `C:\Users\<YOUR_USERNAME>\.ssh\` by default.

The private key is stored in the same directory but named `id_rsa`. **This file should never be moved off of your computer or shared with anyone.**

[Sending your key to the Lab Machine](#)

Now, we need to send a copy of our public key to a lab machine. Once your key has been copied to one lab machine, it can be used to log in to *any other* lab machine. Follow the instructions for your operating system:

macOS/Linux

Type the following command into your terminal. Be sure to replace `CS_USER` with your CS username.

```
ssh-copy-id CS_USER@linux.cs.utexas.edu
```

If prompted “Are you sure you want to continue connecting?” type `yes` and press Enter. You may need to enter your CS password.

Windows

If everything was done successfully, you should now be able to log into the lab machines from off-campus (without using the VPN)!

Sources

This guide is based on UTCS’s guide found [here](#).

Using the Lab Machines

Transferring Files

To transfer files to and from the CS department’s lab machines, we will use a file transfer protocol named SFTP (**S**SH **F**ile **T**ransfer **P**rotocol).

On macOS and Linux, you can transfer files over SFTP using the `scp` command. However, the command can be quite cumbersome and having a GUI application makes transferring files much easier.

Below are some graphical applications for all operating systems which support SFTP.

Windows

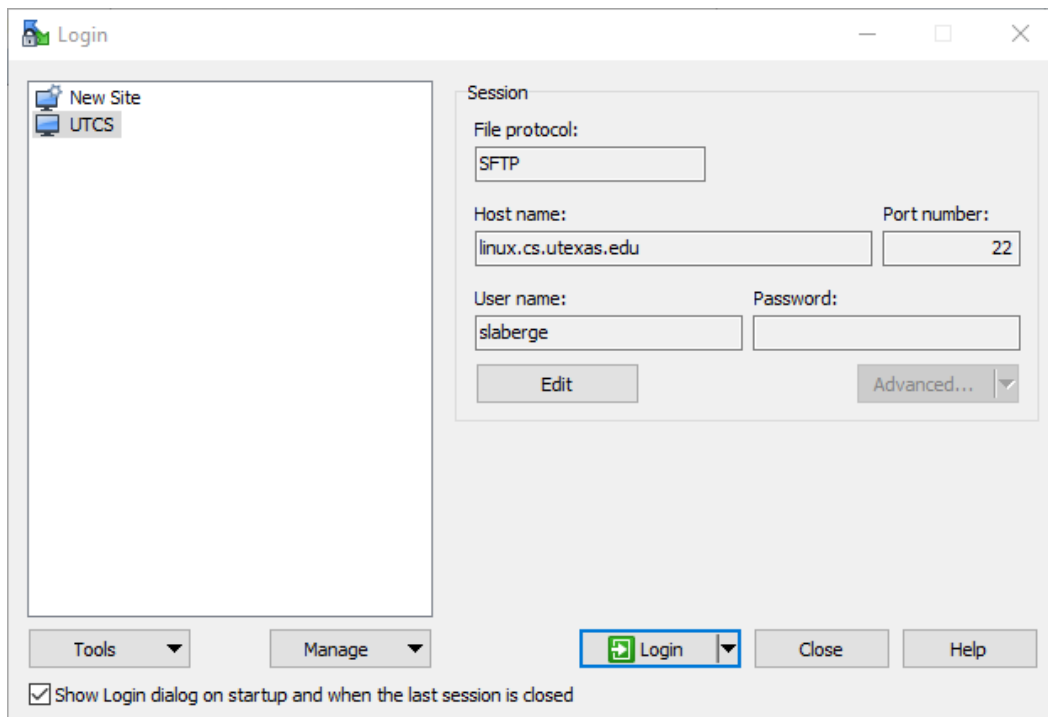
[WinSCP](#)

This is a free and open-source SFTP client for Windows. It is available for download [here](#). (Be careful of the advertisements at the top of the page)

Using SSH Keys with WinSCP

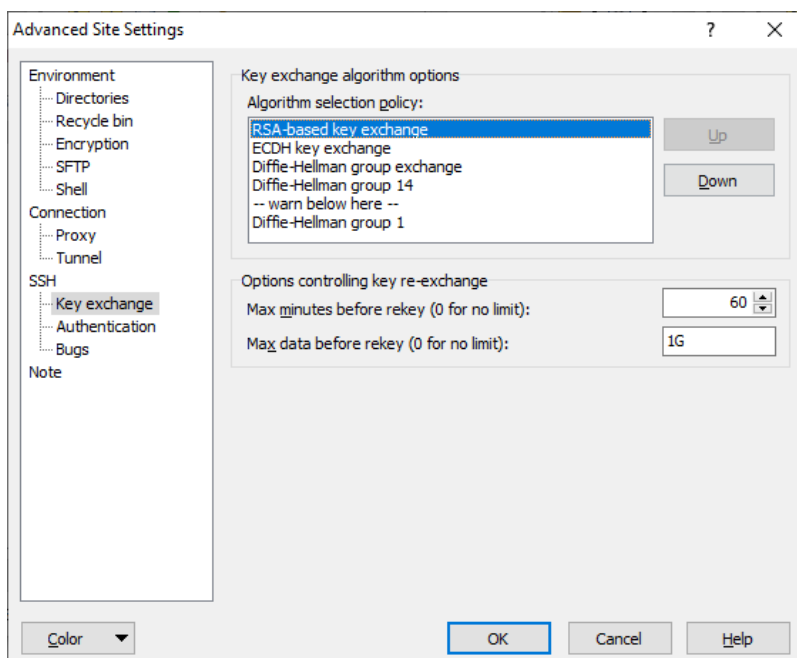
Make sure you have followed the guide for setting up SSH keys on windows [\[here\]](https://www.cs.utexas.edu/facilities-documentation/ssh-keys-cs-windows-10)(<https://www.cs.utexas.edu/facilities-documentation/ssh-keys-cs-windows-10>).

Next, open the dialog to create a new connection.



Enter the name of a lab machine (`linux.cs.utexas.edu` works for this purpose) and your CS username. Make sure the File protocol is set to SFTP.

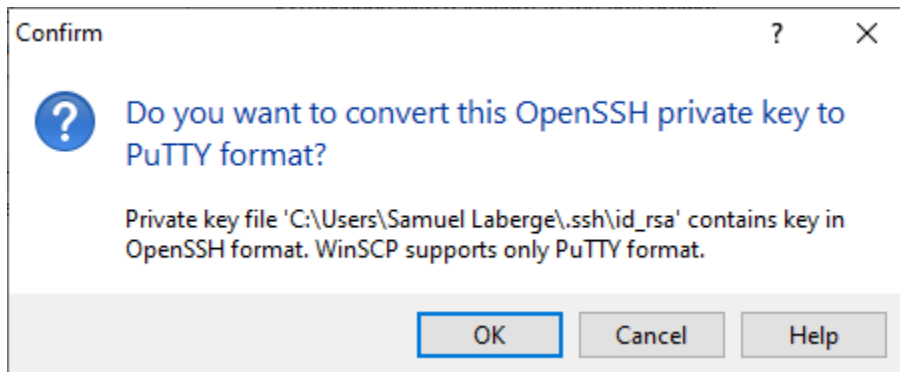
Now, enter the Advanced menu. Under SSH > Key Exchange, move RSA-based key exchange to the top of the Algorithm Selection Policy list. This can be done by selecting it and clicking the Up button.



After that, go to `SSH > Authentication`. Here, we have to tell WinSCP where our Private key is located. If you left it in the default location with the default file name, it will be called `id_rsa` and will be in a directory named `.ssh` in your `Users` directory. Select that file or provide the full path to the file.

Then, you may get an error that WinSCP only supports PuTTY style keys. If you are given the option to convert it to a PuTTY key, then do so.

If that option is not presented to you, you can sometimes make WinSCP show it to you by clicking the `Display public key` button. This is the dialog you should see:



Once the key has been converted and selected, you are done in the Advanced settings and you can connect to the lab machine. You will then be prompted for the passphrase associated with your key, if you have one.

Once connected to the remote machine, you will be provided with a split view. One side will be the files on your personal machine and the other side will be the files on the CS machine. You can simply drag-and-drop files between the two sides of the window to start transferring files.

Linux Basics

You are about to see a lot of potentially unfamiliar commands below. **Don't worry about memorizing all of these right away .**

The only way of getting familiar with these commands is by using them. So if that means you need to look up a Linux cheat-sheet every time or you print out a list of these on your desk, that's perfectly normal.

Some of these commands have strange, un-intuitive names which still cause the most experienced Linux users to forget or confuse them.

Syntax Used

When a command is shown in a code block, it may be followed by *optional* or *required* arguments:

- Optional arguments will be enclosed with square brackets: [optional].
- Required arguments will be enclosed with angle brackets: <required>.

When you actually enter these commands at a prompt, do not include any brackets.

File System Commands

`pwd` - Print the **current working directory**.

```
pwd
```

`ls` - List all of the files in the current directory.

```
ls [-a Show hidden files] [-l Show as list with details]
```

`rm` - Remove (delete) on or more files

```
rm <File 1> [File 2] [File 3] ...
```

Once a file is removed, it typically cannot be recovered!

If you accidentally remove a file on a lab machine, the CS department occasionally runs backups of everyone's files and you may be able to recover it by contacting the department.

`cd` - Change directory

```
cd <Directory name>
```

Use `cd ..` to go up to the parent directory.

`cp` - Copy a file

Usage:

```
cp <Source file> <Destination>
```

`mkdir` - Create a new directory

```
mkdir <New directory name>
```

`rmdir` - Delete an empty directory

```
rmdir <Directory Name>
```

Other Commands

`cat` - Output the contents of a file to the screen

```
cat <File name>
```

`more` - Output one screenful of a file. Press Enter to scroll down. Press q to quit.

```
more <File name>
```

`nano` - Open the nano text editor. Use Control+X to exit (you may be prompted to save, type y or n).

```
nano [File name]
```

`vim` - Open the vim text editor. To quit, first press Escape then type :q! to quit without saving, or :wq to save and quit.

```
vim [File name]
```

How to Learn More About a Command

Most commands have many arguments you can specify to them. If you want to see all the possible arguments you can provide to a command and what they do, you can check the “man”, or manual, page for a specific command.

To do so:

```
man <command name>
```

For example, to learn more about the `ls` command, you could type:

```
man ls
```

If you are still unsure about a command, the next step is to look online for more documentation and examples.

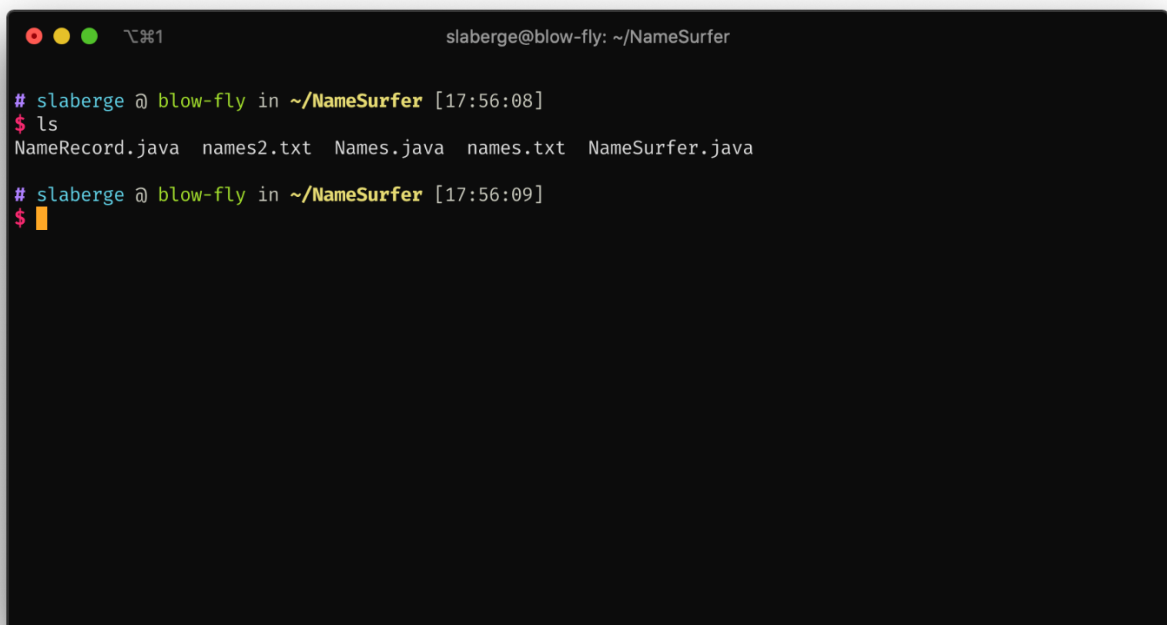
Running Java Programs on Lab Machines

Transferring your Program to the Lab Machines

Firstly, transfer all the required `.java` and input files to the lab machines using an SFTP application of your choice. Put all of these files together in the same directory. I'd recommend making a directory just for these files because once we compile your program, it may generate a lot of `.class` files, and we wouldn't want to clutter up another directory. Now, connect to a lab machine using `ssh` and navigate to this directory using `cd`.

Compiling Your Program

Once we are in the right directory, issue an `ls` command. You should be able to see all of your `.java` and input files.

A terminal window with a dark background. The title bar shows three colored circles (red, yellow, green) and the text "slaberge@blow-fly: ~/NameSurfer". The terminal content shows a prompt "# slaberge @ blow-fly in ~/NameSurfer [17:56:08]" followed by a "\$ ls" command. The output of the command is "NameRecord.java names2.txt Names.java names.txt NameSurfer.java". Below this, another prompt "# slaberge @ blow-fly in ~/NameSurfer [17:56:09]" is shown with a "\$" prompt character and a cursor.

```
slaberge@blow-fly: ~/NameSurfer
# slaberge @ blow-fly in ~/NameSurfer [17:56:08]
$ ls
NameRecord.java names2.txt Names.java names.txt NameSurfer.java
# slaberge @ blow-fly in ~/NameSurfer [17:56:09]
$
```

Now, make sure you know the name of the class which has your `main` method. For this example, my main method is in the `NameSurfer` class, which is in the `NameSurfer.java` file. So, to compile all of the classes the main method in the `NameSurfer` class will require, run this command:

```
javac <main Method Class Name>.java
```

If your code is free of errors and warnings, you shouldn't see any output.

```
slaberge@blow-fly: ~/NameSurfer

# slaberge @ blow-fly in ~/NameSurfer [17:59:23]
$ ls
NameRecord.java  names2.txt  Names.java  names.txt  NameSurfer.java

# slaberge @ blow-fly in ~/NameSurfer [17:59:24]
$ javac NameSurfer.java

# slaberge @ blow-fly in ~/NameSurfer [17:59:27]
$
```

Now, if you issue another ls command, you will be able to see all of the `.class` files as a result of the compilation.

```
slaberge@blow-fly: ~/NameSurfer

# slaberge @ blow-fly in ~/NameSurfer [18:02:16]
$ ls
NameRecord.java  names2.txt  Names.java  names.txt  NameSurfer.java

# slaberge @ blow-fly in ~/NameSurfer [18:02:17]
$ javac NameSurfer.java

# slaberge @ blow-fly in ~/NameSurfer [18:02:21]
$ ls
NameRecord.class  names2.txt  Names.java  NameSurfer.class
NameRecord.java  Names.class  names.txt  NameSurfer.java

# slaberge @ blow-fly in ~/NameSurfer [18:02:21]
$
```

Running Your Java Program

To run your program, you will once again need to know the name of the class which contains your `main` method. In this example, it is once again `NameSurfer`. To run your program with this `main` method, issue the `java` command:

```
java <main Method Class Name>
```

Notice that this command's name, `java` does not end with a "c" and you should not include any file extensions to the class name.

Running Graphical Programs Over SSH

Although SSH connections are done through a command line interface, it is actually possible to run graphical applications over SSH using X-forwarding.

This trick can be useful for CS314 because some of our assignments use a GUI window. If you want to test one of these graphical programs on the lab machines, you'll need to use X-forwarding. If you do not have X-forwarding enabled, you'll get an exception which looks similar to this:

```
Exception in thread "main" java.awt.HeadlessException:
No X11 DISPLAY variable was set, but this program performed an operation
which requires it.
    at
java.desktop/java.awt.GraphicsEnvironment.checkHeadless (GraphicsEnvironment.j
ava:208)
    at java.desktop/java.awt.Window.<init> (Window.java:548)
    at java.desktop/java.awt.Frame.<init> (Frame.java:423)
    at java.desktop/javafx.swing.JFrame.<init> (JFrame.java:224)
    at DrawingPanel.<init> (DrawingPanel.java:87)
    at Recursive.drawCarpet (Recursive.java:188)
    at RecursiveTester.doCarpetTest (RecursiveTester.java:33)
    at RecursiveTester.main (RecursiveTester.java:27)
```

Windows

One-Time Setup

Windows does not have a built-in X-Windows server so we'll need to install one. Xming is a free and popular choice. You can download the installer [here](#).

Open a command prompt and type the following commands:

```
mkdir \dev
echo x > \dev\tty
```

Every-Time Setup

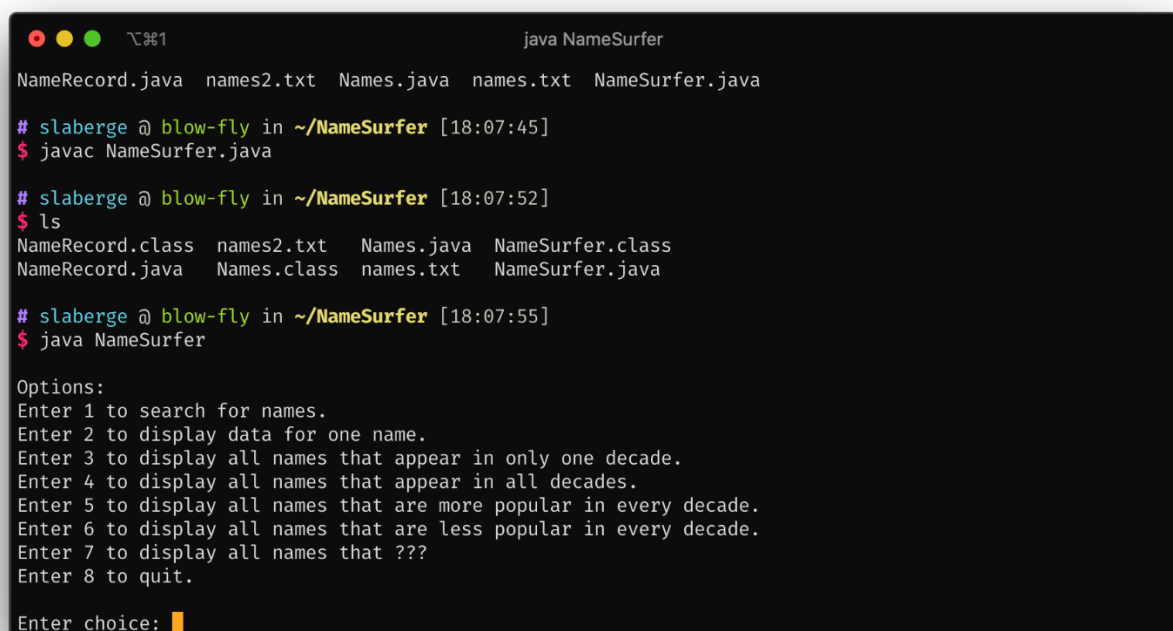
Make sure that Xming is running (it will be in the system tray when running) and run the following three commands in the command prompt. When connecting to the lab machine, you'll need to add an extra argument to the `ssh` command:

```
set DISPLAY=localhost:0
echo x > \dev\tty
ssh -Y CS_USER@CS_MACHINE.cs.utexas.edu
macOS Linux
```

To test if enabling X-forwarding worked, try running this command on the lab machine over SSH:

```
xcowsay "Hello, World\!"
```

If you see the message on your screen, it worked! You should now be able to run your graphical Java programs

A screenshot of a terminal window titled "java NameSurfer". The window shows a series of commands and their outputs. The user is in a directory named ~/NameSurfer. They run 'javac NameSurfer.java' which compiles the files. Then they run 'ls' which lists the files: NameRecord.class, names2.txt, Names.java, NameSurfer.class, NameRecord.java, Names.class, names.txt, and NameSurfer.java. Finally, they run 'java NameSurfer' which starts the program. The program displays a menu of options for searching and displaying names, and prompts the user to enter a choice. The cursor is currently on the line "Enter choice: ".

```

NameRecord.java  names2.txt  Names.java  names.txt  NameSurfer.java

# slaberge @ blow-fly in ~/NameSurfer [18:07:45]
$ javac NameSurfer.java

# slaberge @ blow-fly in ~/NameSurfer [18:07:52]
$ ls
NameRecord.class  names2.txt  Names.java  NameSurfer.class
NameRecord.java   Names.class  names.txt   NameSurfer.java

# slaberge @ blow-fly in ~/NameSurfer [18:07:55]
$ java NameSurfer

Options:
Enter 1 to search for names.
Enter 2 to display data for one name.
Enter 3 to display all names that appear in only one decade.
Enter 4 to display all names that appear in all decades.
Enter 5 to display all names that are more popular in every decade.
Enter 6 to display all names that are less popular in every decade.
Enter 7 to display all names that ???
Enter 8 to quit.

Enter choice: █
```

If everything was successful, you should be running your Java program on a lab machine!

Make sure that your program works as intended in this environment since this is how TAs will be grading your assignments.