1 Contact

If you have questions about this document you can find our team webpage with contact info at https://up-rs-esp.uni-potsdam.de/. Most questions can be addressed to Dr. Harald Schernthanner (hschernt@uni-potsdam.de).

2 Objectives

This document describes in detail how to access and work remotely using the PC Pool at Potsdam using `ssh` logins. You should already be familiar with your command line, setting up a Python environment, and Jupyter notebooks from the other manual titled *Geological Remote Sensing Python Setup*.

3 Working Remotely: PC Pool in Golm (Building 27)

If you do not or cannot install the required software or Python environment on your personal computer, we have the option of working remotely through the PC Pool at Campus Golm. Heavy computation may also require the more powerful PC Pool on campus, so it's good to familiarize yourself with these steps and generate your own SSH private/public key pair for login to the PC Pool.

3.1 Login Via ssh

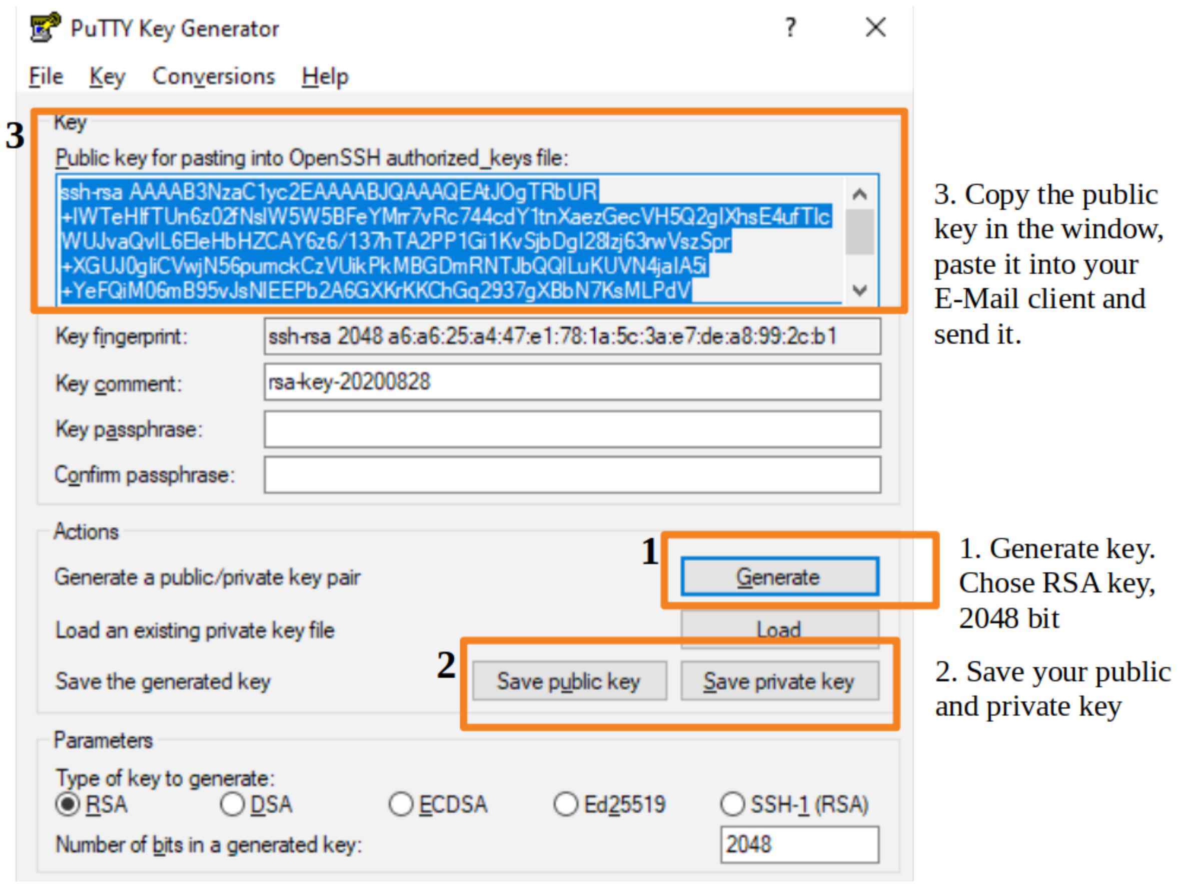
SSH stands for secure shell and is a safe and encrypted way to access computers remotely. We will work with the `ssh` command-line environment. The below steps will walk you through ssh private/public key generation, allowing secure and easy login without passwords. Once logged into the remote PC Pool computer you will be in a Linux (Ubuntu 18.04.5) command-line environment.

3.1.1 ssh Key Generation and Login: Windows

Windows does not come with `ssh` in the default command-line environment. The use of PuTTY is recommended. PuTTY is an ssh client, that has the capability to generate private/public ssh keys and login to remote servers. *As an alternative to the below steps, consider installing the Windows Subsystem for Linux available in Windows 10. This will give you a Linux terminal identical to Ubuntu (or whatever Linux flavor you prefer). In that case you could use the instructions in the next section for ssh login on Linux / Mac.*

1. Install PuTTY on your computer.

2. Open the program "PuTTYgen" (Open the start menu > search for "PuTTYgen" > click the icon), a key generator that is installed along with PuTTY.
3. Using PuTTYgen (see below figure): Generate a 2048 bit private/public key pair, save the public and private key to a secure location on your personal computer, and copy-paste the public key to send via email to the system administrator Harald Schernthanner (hschernt@uni-potsdam.de). **Important: Keep your private key somewhere safe, consider adding a key passphrase in PuTTYgen, and *never* handout your private key!**



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAtJOgTRbUR
+IWTeHIfTUn6z02fNsIW5W5BFeYMrr7vRc744cdY1tnXaezGecVH5Q2gIXhsE4ufTIc
WUJvaQvIL6EleHbHZCAY6z6/137hTA2PP1Gi1KvSjbDgI28lzj63rwVszSpr
+XGUJ0gliCVwjN56pumckCzVUikPkMBGDmRNTJbQQILuKUVN4jaIA5i
+YeFQiM06mB95vJsNIEEPb2A6GXKrKKChGq2937gXBbN7KsMLPdV
```

Key fingerprint: ssh-rsa 2048 a6:a6:25:a4:47:e1:78:1a:5c:3a:e7:de:a8:99:2c:b1

Key comment: rsa-key-20200828

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair — Generate

Load an existing private key file — Load

Save the generated key — Save public key — Save private key

Parameters

Type of key to generate:
[✓ RSA] DSA ECDSA Ed25519 SSH-1 (RSA)

Number of bits in a generated key: 2048

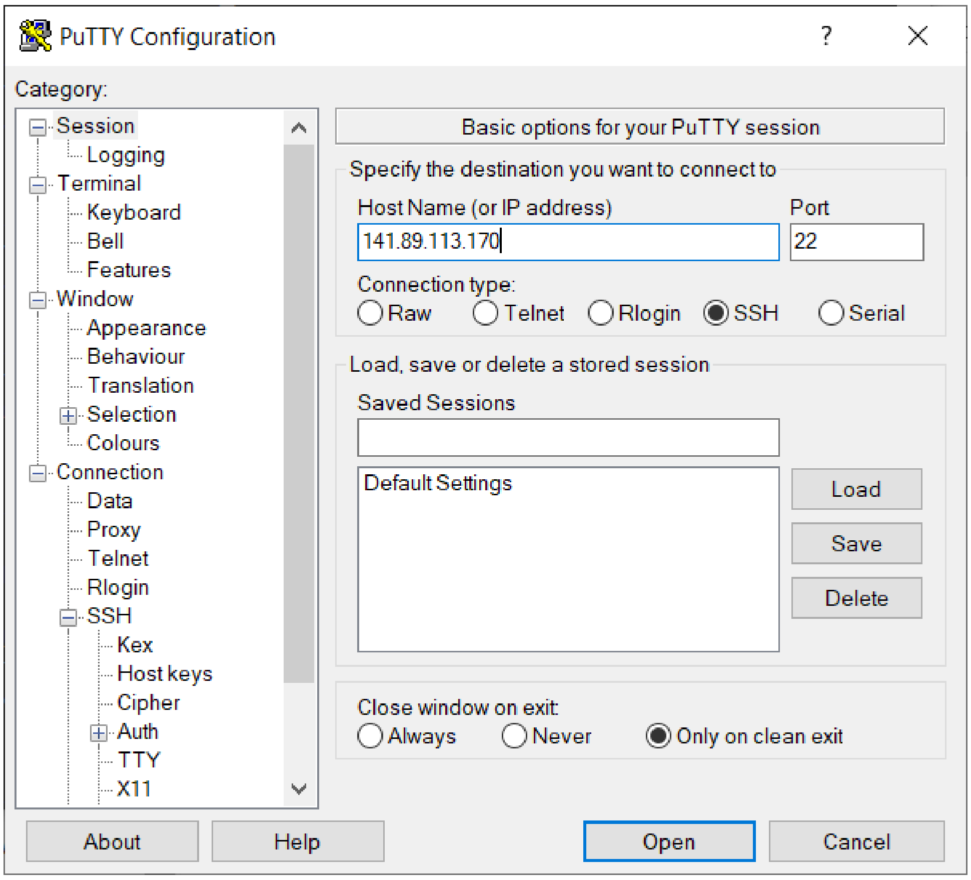
3. Copy the public key in the window, paste it into your E-Mail client and send it.

1. Generate key. Chose RSA key, 2048 bit

2. Save your public and private key

Figure 1: Generate 2048 bit private/public key pair using PuTTYgen

4. Harald Schernthanner (hschernt@uni-potsdam.de) generates an account for you after receiving your public key via email, puts your key on the system, contacts you when your account is ready, and tells you your username and the IP of your assigned machine.
5. When the account is ready, please login using PuTTY (Open the start menu > search for "PuTTY" > click the icon). You need to enter the *IP address of your assigned machine*:



Specify the destination you want to connect to

Host Name (or IP address): 141.89.113.170

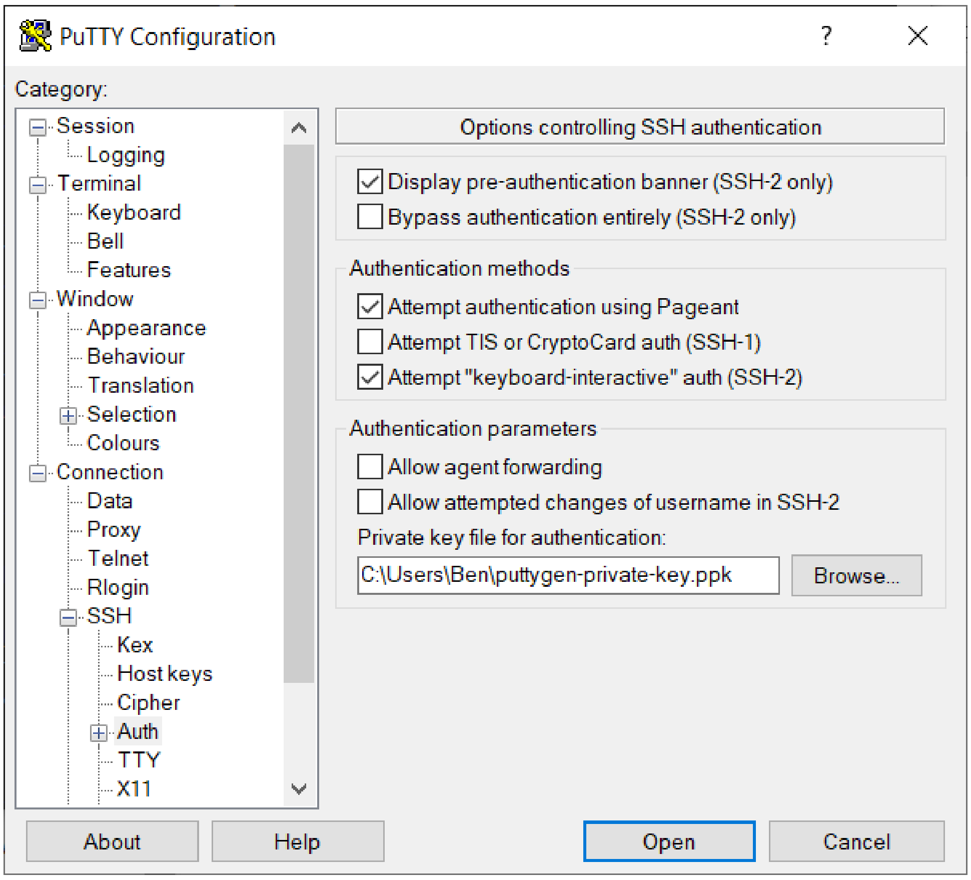
Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Close window on exit: Always Never [✓ Only on clean exit]

Figure 2: Enter IP address of you assigned machine in PuTTY on Windows

6. To complete the login in PuTTY you have to tell the program the path to the private key (Connection > SSH > Auth), then you can press “Open” to login (and “Yes” to the security alert):



Options controlling SSH authentication

[✓] Display pre-authentication banner (SSH-2 only)
[ ] Bypass authentication entirely (SSH-2 only)

Authentication methods

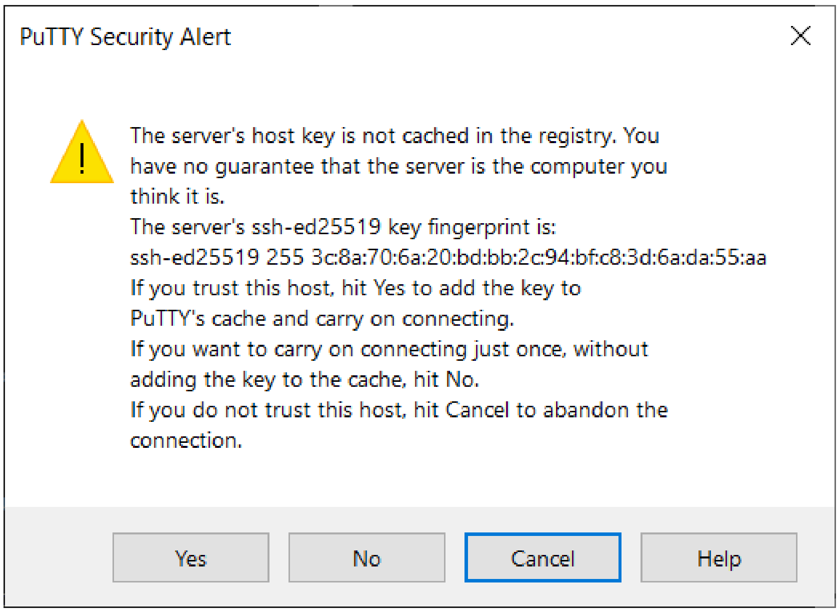
[✓] Attempt authentication using Pageant
[ ] Attempt TIS or CryptoCard auth (SSH-1)
[✓] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

[ ] Allow agent forwarding
[ ] Allow attempted changes of username in SSH-2

Private key file for authentication:
C:\Users\Ben\puttygen-private-key.ppk

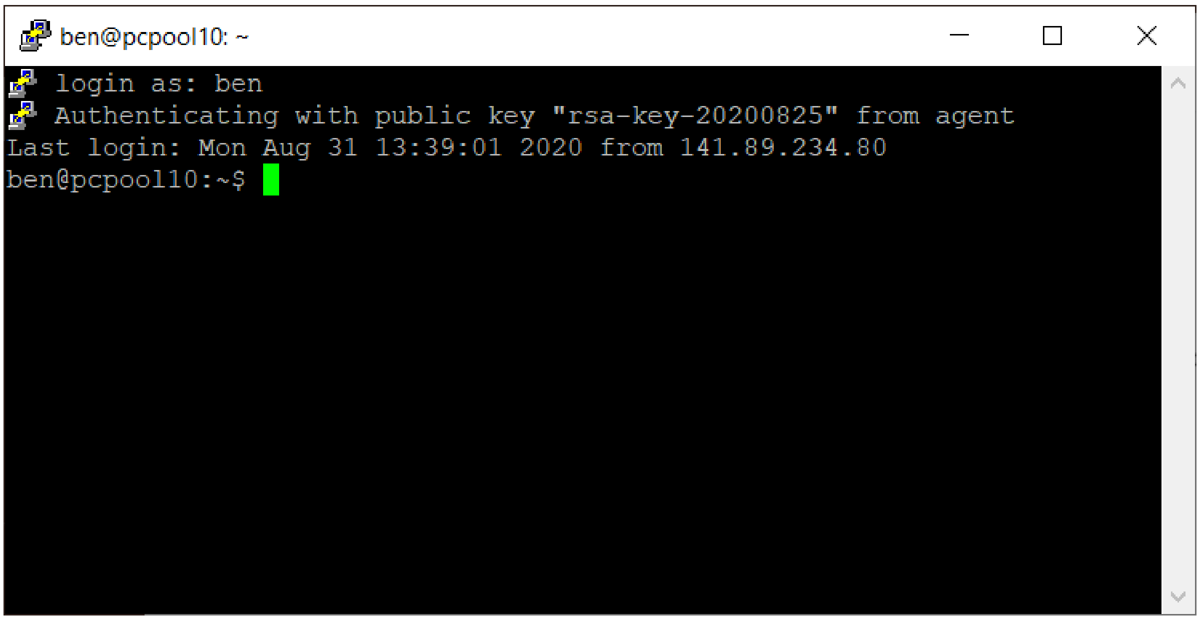
Figure 3: Point PuTTY to the private key you saved in the previous step (Connection > SSH > Auth > "Private key file for authentication"). After the IP address and private key path are entered you can click "Open".


PuTTY Security Alert

The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's ssh-ed25519 key fingerprint is:
ssh-ed25519 255 3c:8a:70:6a:20:bd:bb:2c:94:bf:c8:3d:6a:da:55:aa
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

Figure 4: Say "Yes" to the security alert which should only occur on your very first login.

7. With login complete you will enter your username and end up on a command-line terminal for your assigned computer. Below we see user ben logged into `pcpool10`. You are now remotely logged into the PC Pool!


ben@pcpool10: ~

```
login as: ben
Authenticating with public key "rsa-key-20200825" from agent
Last login: Mon Aug 31 13:39:01 2020 from 141.89.234.80
ben@pcpool10:~$
```

Figure 5: Logged on to the PC Pool as user ben on computer `pcpool10`

3.1.2 ssh Key Generation and Login: Linux / Mac

On Linux or Mac, you should be able to use `ssh` right away from the terminal without installing any additional software.

1. Open a terminal and execute:

```
ssh-keygen
```

2. You will be asked to "Enter file in which to save the key (/home/ben/.ssh/id_rsa):". Here, define a folder and filename (e.g., `/home/yourusername/yourkeyname`). When you are asked to "Enter a passphrase:" leave it blank by pressing enter (we don't necessarily need a passphrase for the public key).
3. This generates two files: First the private key named `yourkeyname` (**keep it in a safe place and do not hand it out**) and second the public key named `yourkeyname.pub`. Send the public key to the administrator Harald Schernthanner (hschernt@uni-potsdam.de).
4. Harald Schernthanner (hschernt@uni-potsdam.de) generates an account for you after receiving your public key via email, puts your key on the system, contacts you when your account is ready, and tells you your username and the IP of your assigned machine.
5. Login with `ssh` from your terminal. You need to enter the *IP address of your assigned machine*. For username `ben` logging onto the assigned computer with IP address `141.89.113.170` (After the `-i` flag you have to define **the path to your private key**):

```
ssh -i /home/ben/yourkeyname ben@141.89.113.170
```

3.2 Data Storage on PC Pool Computers

Do not store your data in your home directory (`/home/student`). Instead use `/data` for fast, local storage and `/NAS` for network storage. You should create your own sub-directory. For the user `ben`, I would write:

```
mkdir /data/ben
```

Note: We periodically wipe the `/home`, `/data`, and `/NAS` directories (don't worry, not during courses!), so don't plan on long-term storage on the PC Pool

3.3 File Transfer Between Local and Remote Computers

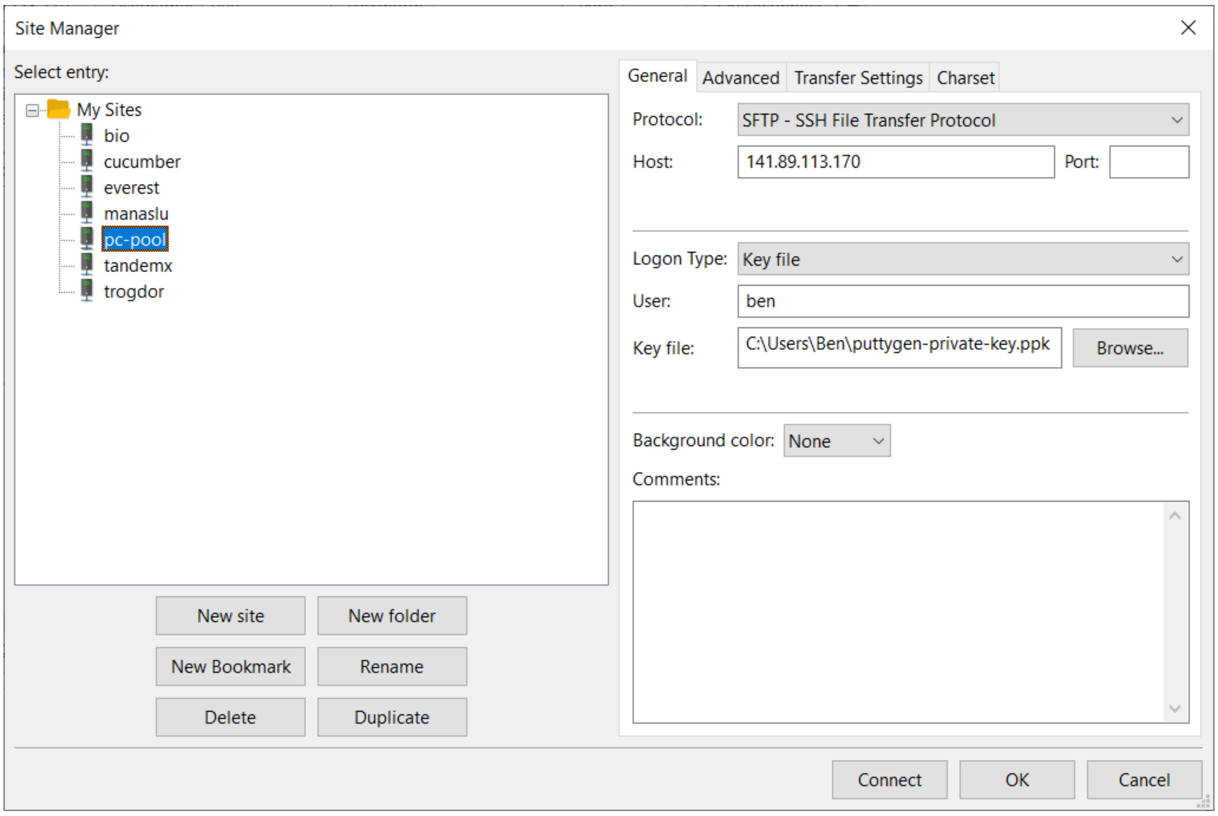
At times you will need to move files between your local computer and the remote machines in the PC Pool. For instance, after you finish processing a large dataset on the remote computer you may want to transfer the output to your local system to look at the results and create figures. Or, for an `.ipynb` Jupyter notebook that you ran on the server, you may want to transfer it to your local computer after running. This is a common task and there are many options for file transfer between local and remote systems via SSH File Transfer Protocol (SFTP).

Keep in mind: Likely, your download speed will be very fast (i.e., it will be fast to pull files from the PC Pool), but your upload speed will be slow (i.e., you have to be patient if you put data on the PC Pool).

3.3.1 GUI: FileZilla - *Your most likely choice*

You can install FileZilla Client on any OS. This is a straight forward way to copy files back and forth (you see progress and expected up/download times). If you plan to move files back-and-forth continuously, you may want to look into `rsync` or a similar command-line option.

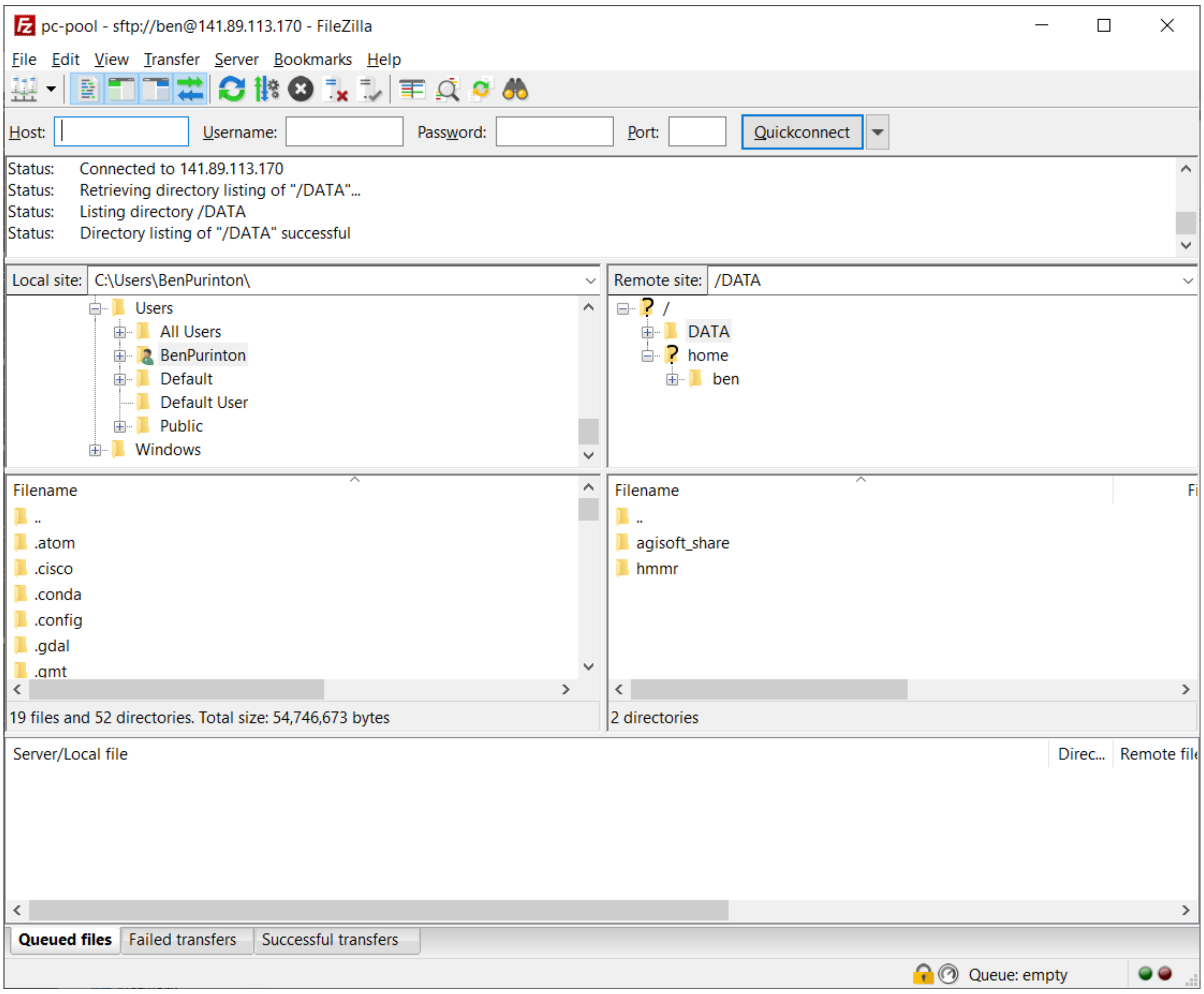
When you open FileZilla for the first time you will need to click the top left icon ("Open the Site Manager" with the little server icons underneath "File"). In the Site Manager window create a new site, call it "pc-pool", and fill out the "General" tab exactly as below (changing the fields to your host IP, username, and SSH private key file location):



Protocol: SFTP - SSH File Transfer Protocol
Host: 141.89.113.170 Port:
Logon Type: Key file
User: ben
Key file: C:\Users\Ben\puttygen-private-key.ppk Browse...
Background color: None

Figure 6: Open a FileZilla session and create a new site to the remote pc-pool host with your username and SSH private key.

After you click "Connect" you can see all of the files on the local and remote systems in two side-by-side panels, navigate around the system folders, and copy data back and forth:


pc-pool - sftp://ben@141.89.113.170 - FileZilla
File Edit View Transfer Server Bookmarks Help
Host: Username: Password: Port: Quickconnect

```
Status:    Connected to 141.89.113.170
Status:    Retrieving directory listing of "/DATA"...
Status:    Listing directory /DATA
Status:    Directory listing of "/DATA" successful
```

Figure 7: FileZilla connected to remote host showing the file browser for both the local and remote computers.

Note: The new site "pc-pool" will stay in your site manager list and you can easily reconnect in your next FileZilla session as long as you don't move your SSH private key to a new location.

3.3.2 Command Line: `rsync`

If you like to work on the command line (because it is more efficient), rsync is the best option. It should come standard on Linux / Mac. For Windows: it is included in the Cygwin Linux-like terminal; can be added to the Git shell via these instructions; and/or is included in the Windows Subsystem for Linux available in Windows 10.

The basic syntax is:

```
rsync <options> <local filename> <remote filename>
```

For example, if I (user: `ben` on computer `141.89.113.160`) want to copy the file `my_jupyter_notebook.ipynb` from my local computer to the PC Pool location `/data/ben`, use:

```
rsync -avzh --progress my_jupyter_notebook.ipynb ben@141.89.113.160:/data/ben/
```

Or to transfer a remote file on the PC pool to my current directory (**Note:** the period (`.`) below refers to ***current directory***, so wherever you are currently `cd`'d to):

```
rsync -avzh --progress ben@141.89.113.160:/data/ben/my_jupyter_notebook.ipynb .
```

The `rsync` options used here include: `-a` (archive), `-v` (verbose), `-z` (compress), `-h` (human readable), and `--progress` (progress bar). You can read about the `rsync` options online via the manual or in cheat sheets.

4 Using Jupyter Remotely Via SSH

You can run Jupyter on the remote PCs to harness more computing power and to use datasets that you may not want (or be able) to download locally.

4.1 Using `screen` Sessions

When logged in to the remote PC via the ssh command (Mac / Linux) or via PuTTY (Windows), you should *always* start a `screen` session, which keeps your Jupyter notebook or other processes alive even if you lose your local internet connection. Simply enter the command:

```
screen
```

You can familiarize yourself with a few important `screen` commands here. The most important are:

To give the screen session a useful name for finding it later:

```
screen -S screen_name
```

To detach from a screen session without killing the screen (and thus killing the processing), use the keyboard shortcut: `Ctrl-a + d`

And to re-join a detached screen:

```
screen -r screen_name
```

To end a screen, re-join it using the above command and just type `exit`. You can also end a screen without first rejoining it by entering:

```
screen -S screen_name -X quit
```

Note: You should become very familiar with the above screen commands. Especially detaching, re-joining, and quitting screens, which you should be doing in every SSH session! If you keep starting new screens without ending old ones we eventually have to kill all the processes

4.2 Opening Remote Notebook Via Port Forwarding: Linux / Mac

On Mac / Linux login to the remote PC using ssh:

```
ssh -i /home/my_username/yourkeyname my_username@my_pc_ip_address
```

Once you are in a **screen session** on the remote PC, we can setup the Jupyter notebook via port forwarding. First do:

```
conda activate <Class Environment Name>
jupyter notebook --generate-config
jupyter notebook password
```

This will first activate the Python environment for the class (e.g., `DataAnalysis` in place of `<Class \ Environment Name>`), then generate a configuration file for Jupyter, and then allow you to create a password to access the notebook.

Following this, you need to `cd` into the remote directory where your data is stored, e.g., `cd \ /data/my_username/DataAnalysis_Labs/`. Make sure that you have your own directory set up on the `/data` drive! Otherwise you will overwrite each other's work!

Now start the Jupyter Notebook session on the PC Pool:

```
jupyter notebook --no-browser --port=XXXX
```

Replace "XXXX" with your assigned port number. ***Note: Use the four-digit port number (e.g., 8888) that was assigned to you along with your PC IP address and username. If you accidentally use the same port as someone else on the same PC, you would be trying to edit each other's notebooks!***

Now you are all set on the server side and you can open a new command line / terminal on your local computer.

The next step is to tell your local computer how to access the remotely running Jupyter Notebook you started on the server. We do this via port forwarding. Once again, this is a simple one-line command **run in a new command-line prompt on your local computer**:

```
ssh -N -f -L localhost:YYYY:localhost:XXXX -i /home/my_username/yourkeyname  \
    my_username@my_pc_ip_address
```

`-N` tells SSH that no remote commands will be executed, `-f` tells SSH to go to background so the local terminal remains usable, and `-L` lists the port forwarding configuration (remote port XXXX to local port YYYY). Here, “XXXX” is the same port you used in the remote PC command above, and “YYYY” is the port you want your local computer to listen on. You can use the default 8888 in place of “YYYY”, or whatever port you prefer. This command won’t show any output – you can check it is working by opening up your web browser, and navigating to the url “http://localhost:YYYY/”.

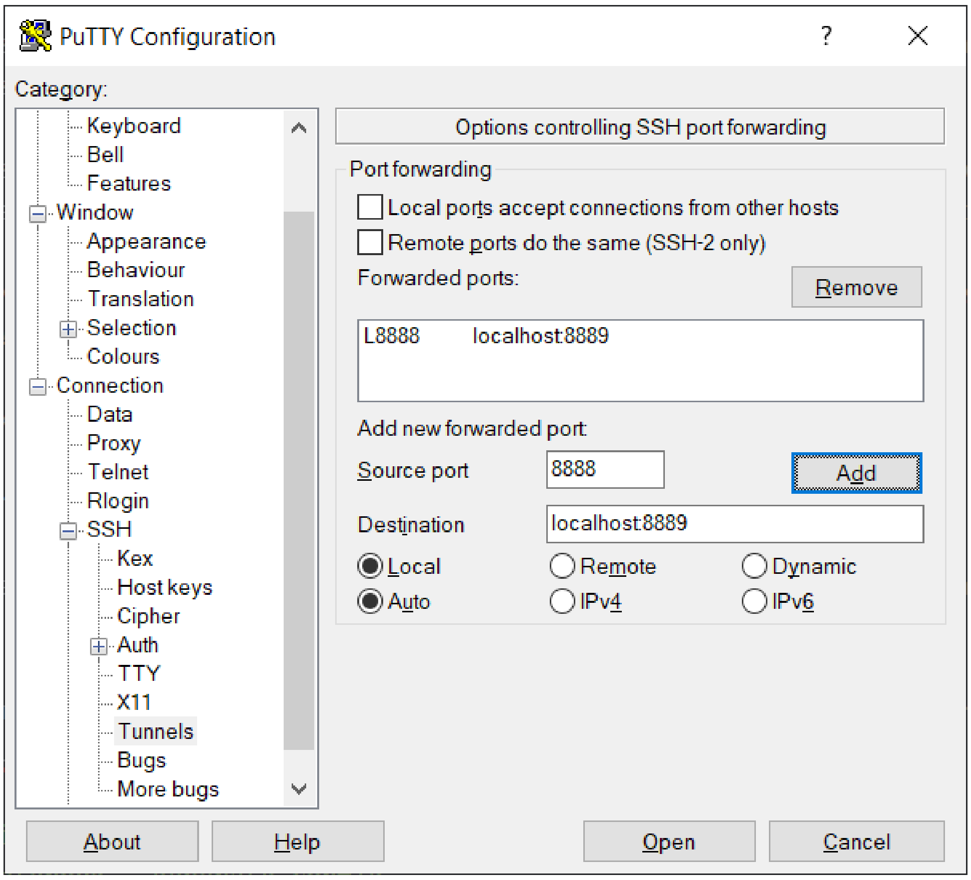
You should now be able to access the remote notebooks directly from your browser, while using the computing power of the PC Pool. File transfers between the local and remote machines for datasets, notebooks, and outputs from running scripts can be accomplished using FileZilla or `rsync`, as described above.

4.3 Opening Remote Notebook Via Port Forwarding: Windows

On Windows first open PuTTY and enter the IP address of your assigned machine (**Note:** You can save the IP address by entering it in the “Saved Sessions” box and then pressing “Save” so you don’t have to keep entering it when you open PuTTY).

After this, navigate to Connection > SSH > Tunnels and put in “Source port” the local port that you want to listen on. You can use 8888 here, or any other port greater than 8000 (e.g., 8881). In the “Destination” put “localhost:XXXX” and then press “Add”. Replace “XXXX” with your assigned port number. ***Note: Use the four-digit port number (e.g., 8888) that was assigned to you along with your PC IP address and username. If you accidentally use the same port as someone else on the same PC, you would be trying to edit each other’s notebooks!***

In this example the user `ben` has been assigned port 8889:



Options controlling SSH port forwarding

Port forwarding

☐ Local ports accept connections from other hosts

☐ Remote ports do the same (SSH-2 only)

Forwarded ports:

```
L8888        localhost:8889
```

Add new forwarded port:

Source port: 8888

Destination: localhost:8889

[✓ Local] Remote Dynamic

[✓ Auto] IPv4 IPv6

Figure 8: Port forwarding in PuTTY.

Now you can press "Open" to start the remote session with port forwarding setup. Your computer will "listen" on port 8888 and the remote PC will broadcast on port 8889, or whatever port combination you use.

In the remote terminal start a **screen session**. Once you are in a screen on the remote PC, we can setup the Jupyter notebook. First do:

```
conda activate <Class Environment Name>
jupyter notebook --generate-config
jupyter notebook password
```

This will first activate the Python environment for the class (e.g., `DataAnalysis` in place of `<Class \ Environment Name>`), then generate a configuration file for Jupyter, and then allow you to create a password to access the notebook.

Following this, you need to `cd` into the remote directory where your data is stored, e.g., `cd \ /data/my_username/DataAnalysis_Labs/`. Make sure that you have your own directory set up on the `/data` drive! Otherwise you will overwrite each other's work!

Now start the Jupyter Notebook session on the PC Pool:

```
jupyter notebook --no-browser --port=XXXX
```

Replacing “XXXX” with your assigned port number.

Now on you local computer open a browser and navigate to the url “http://localhost:8888/”, replacing 8888 with whichever port you put in the “Source port” in PuTTY.

You should now be able to access the remote notebooks directly from your browser, while using the computing power of the PC Pool. File transfers between the local and remote machines for datasets, notebooks, and outputs from running scripts can be accomplished using FileZilla or `rsync`, as described above.