EXPLORE DIGITAL SKILLS

Introduction to the Cloud - Part II

A Beginner’s Guide to Using AWS
Elastic Cloud Compute (EC2)

Recap of Part 1

In the previous tutorial, you launched a Jupyter Notebook from your very own EC2 instance.

You essentially rented a small computer from AWS (likely in another city or country) and used it to display a notebook on your screen all the way in your home or workspace.

But what's next and how is this useful to us in the course?

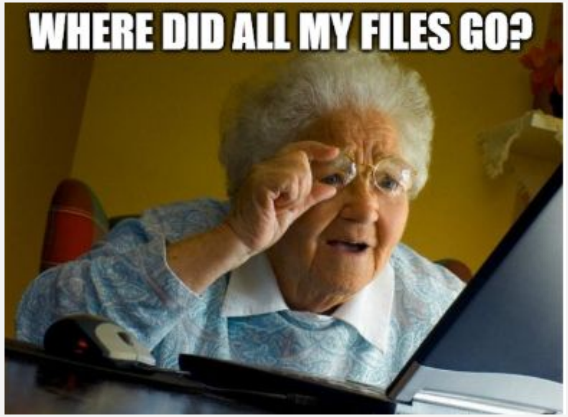
Transferring files to your EC2 Instance

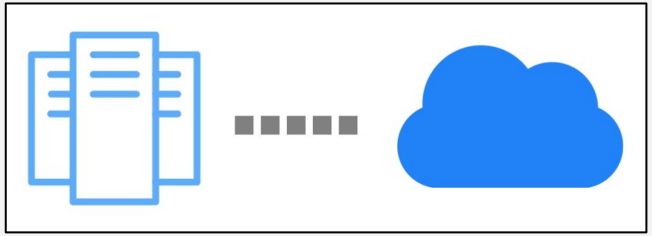
- After launching a notebook from your EC2 instance, you will have noticed that none of your local files were there.
- This is because the notebook is running on a **different computer**, and it will only be able to see files on **that computer**.
- So in order to run EDSA notebooks, or run your own notebooks that are stored locally, we need a way to **transfer files to** our EC2 instance.

In this train we look at two ways to do this:

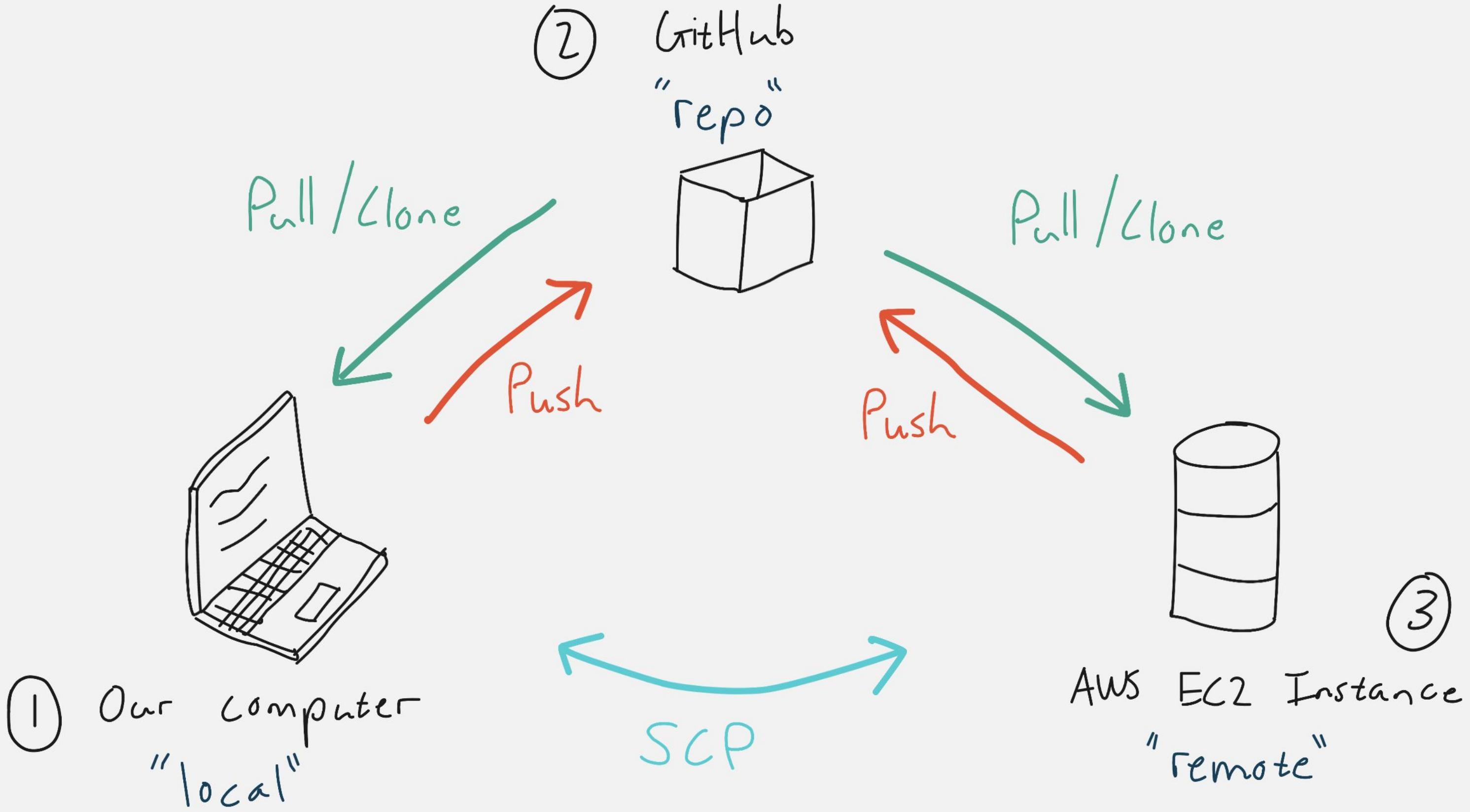
1. Using a **GitHub** remote repository; and
2. Using Secure Copy Protocol (**SCP**).

On the following slide, take a moment to familiarise yourself with the elements of this ecosystem, which involves the machine we're working on (our **local**), the GitHub repository (the **repo**), and the AWS instance (the **remote**).





Understanding the Ecosystem



Using GitHub - Basic Commands

We'll look at the **GitHub method first**. In essence, what we are trying to do is **move a file along the following flow diagram** (refer to the ecosystem diagram to see which is which):

Local -> Repo -> Remote

There are several basic git commands that we'll need to use to achieve this. Briefly:

- **git clone**

  Creates a copy of the GitHub repository to the local machine you're working on.
- **git pull**

  Copies the latest changes (if any) from the GitHub repository to your local repository.
- **git add**

  Adds changes you have made to a file to a staging area, creating a checkpoint.
- **git commit**

  Saves the latest checkpoint that you've made to your local machine.
- **git push**

  Uploads any changes (commits) in your local repository to the remote repository.
- **git status**

  Displays the state of the working directory and the staging area.
- **git fetch**

  Used to download contents from a remote repository.

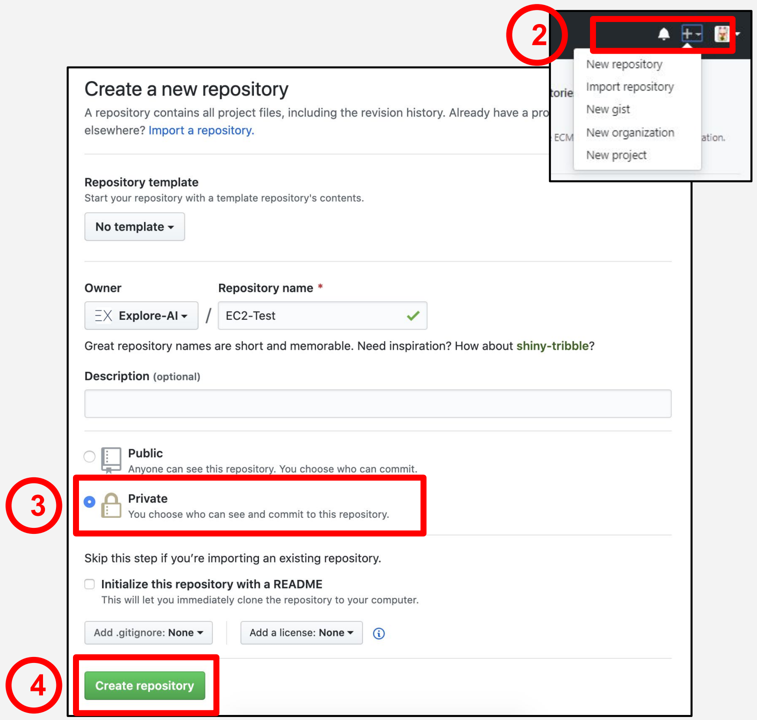
Using GitHub - Step 1: Create a Private Repository

This is where we create the item marked as **repo** on the ecosystem diagram. The EXPLORE course material is private, so it is **very important** that you create a **private repository.**

Tip: This repo is also *remote* by nature, but we won't refer to it as remote, to keep the instructions simpler.

To create a private repository:

1. **Sign into GitHub** - Create an account if you don't have one.
2. At the **top right**, click the **+**, and select ***New repository***.
3. A screen like that shown to the right will appear. Fill in the details. Remember to choose ***Private***.
4. Click ***Create repository***.



Create a new repository

A repository contains all project files, including the revision history. Already have a pro[...] elsewhere? Import a repository.

Repository template
Start your repository with a template repository's contents.
No template ▾

Owner Explore-AI ▾ / **Repository name** * EC2-Test ✓

Great repository names are short and memorable. Need inspiration? How about **shiny-tribble**?

Description (optional)

○ **Public**
Anyone can see this repository. You choose who can commit.

● **Private**
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▾ Add a license: **None** ▾

Create repository

EXPLORE DIGITAL SKILLS

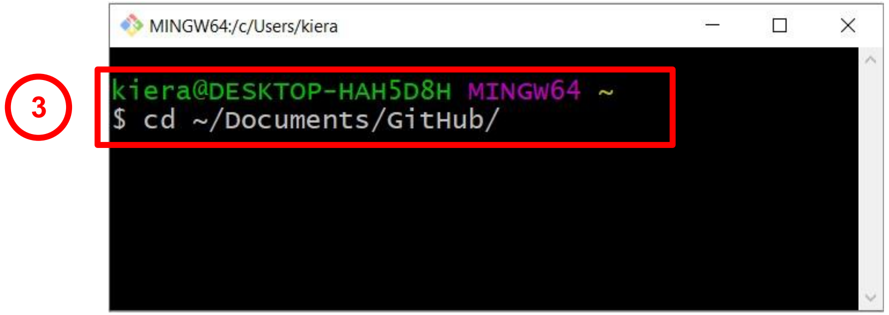
Using GitHub - Step 2: Clone Repo to Local on Windows

For Mac, head to the next slide.

- Right, we have a **repo**. Now, we need to clone it to both the **local** machine and the **remote** machine.
- Once cloned in both places, we can effectively move files between the **local** and **remote** via the **repo** using the **`git push`** and **`git pull`** commands.

Windows

1. We'll be using **git bash**, which should be installed. If not, have a look at the tutorial found here.
2. We'll need a location on our local machine in which to store the cloned repository. We suggest the following: ~/Documents/GitHub*
3. Open git bash, and use the change directory command, `cd`, to plant yourself in the right folder:
   `cd ~/Documents/GitHub`



```
kiera@DESKTOP-HAH5D8H MINGW64 ~
$ cd ~/Documents/GitHub/
```

* ~ Refers to the home directory

Using GitHub - Step 2: Clone Repo to Local on Mac

For Windows, head to the previous slide.

- Right, we have a **repo**. Now, we need to clone it to both the **local** machine and the **remote** machine.
- Once cloned in both places, we can effectively move files between the **local** and **remote** via the **repo** using the `git push` and `git pull` commands.

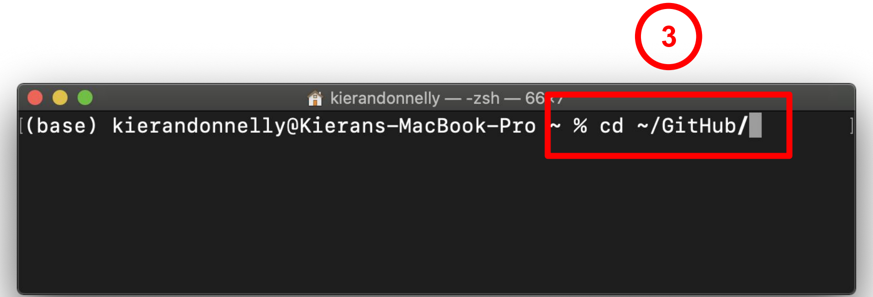
Mac

1. We'll be using **Terminal** and **git**. If git is not installed, revisit the appropriate train here.
2. We'll need a location on our local machine in which to store the cloned repository. We suggest:

   ~/GitHub

3. Open Terminal, and use the change directory command, `cd`, to plant yourself in the right folder:

   ```
   cd ~/GitHub
   ```



```
(base) kierandonnelly@Kierans-MacBook-Pro ~ % cd ~/GitHub/
```

* ~ Refers to the home directory

Using GitHub - Step 2: Clone Repo to Local (Cont.)

From here on out, commands are the same for both Windows and Mac.

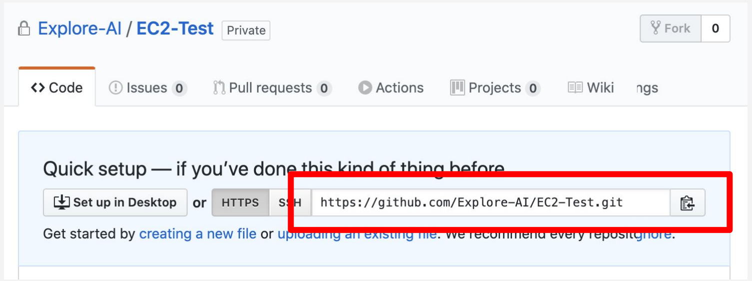
1. We've *changed directory* to the right folder, now we can proceed to clone the GitHub **repo**.

2. On the GitHub website, navigate to your new repo and click the **Copy** button next to the **HTTPS** link for your repo.

3. In your command line tool, use the following command to initiate cloning:

```
git clone <HTTPS URL from GitHub>
```

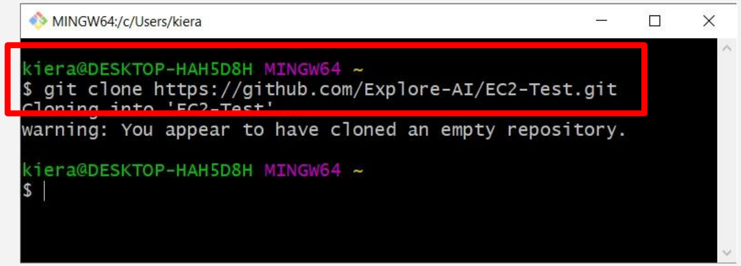
4. If asked to login, ensure you use the same username and password for the GitHub account on which you created the repo.



Quick setup — if you've done this kind of thing before

or https://github.com/Explore-AI/EC2-Test.git

Get started by creating a new file or uploading an existing file. We recommend every repositgnore.



```
kiera@DESKTOP-HAH5D8H MINGW64 ~
$ git clone https://github.com/Explore-AI/EC2-Test.git
Cloning into 'EC2-Test'...
warning: You appear to have cloned an empty repository.

kiera@DESKTOP-HAH5D8H MINGW64 ~
$ |
```

Using GitHub - Step 3: Clone Repo to Remote

1. We've cloned the **repo** to our **local**. Now, we'll clone it to our **remote** machine.

2. Connect to your remote machine with the following command. Revisit Part 1 here if you've forgotten:

   `ssh explore-student@<EC2 Instance IPv4 address>*`

3. Use the following command to initiate cloning:

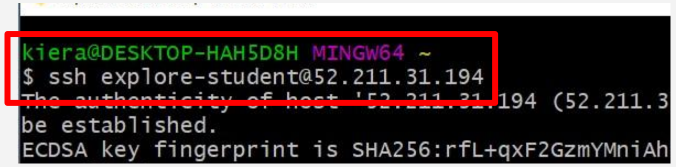
   `git clone <HTTPS URL from GitHub>`

4. Login. If you use your email address as your username and get an error, you may need to get your actual username from GitHub. Note: If you have 2FA enabled, see this post.

5. To check the **repo** has been cloned, use the list command:

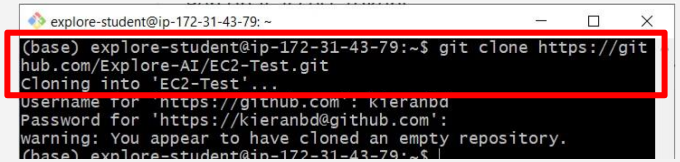
   `ls`

2



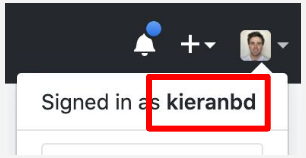
```
kiera@DESKTOP-HAH5D8H MINGW64 ~
$ ssh explore-student@52.211.31.194
The authenticity of host '52.211.31.194 (52.211.3
be established.
ECDSA key fingerprint is SHA256:rfL+qxF2GzmYMniAh
```

3



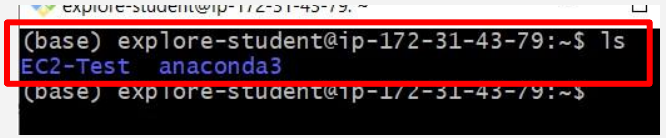
```
explore-student@ip-172-31-43-79: ~
(base) explore-student@ip-172-31-43-79:~$ git clone https://git
hub.com/Explore-AI/EC2-Test.git
Cloning into 'EC2-Test'...
Username for 'https://github.com': kieranbd
Password for 'https://kieranbd@github.com':
warning: You appear to have cloned an empty repository.
(base) explore-student@ip-172-31-43-79:~$
```

4



Signed in as kieranbd

5



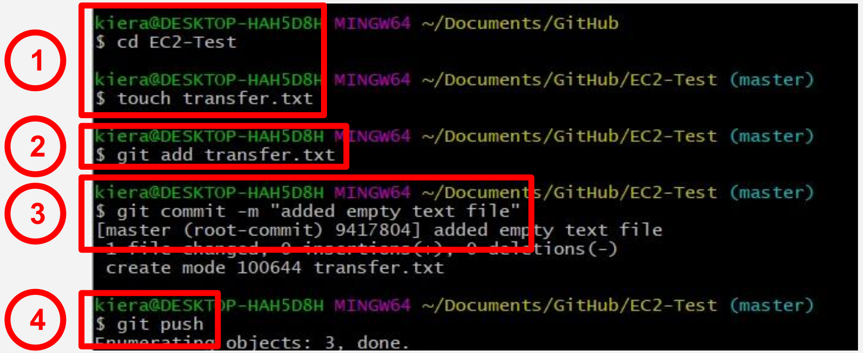
```
explore-student@ip-172-31-43-79: ~
(base) explore-student@ip-172-31-43-79:~$ ls
EC2-Test  anaconda3
(base) explore-student@ip-172-31-43-79:~$
```

* This should be the public IP address

Using GitHub - Step 4: Add File to Repo from Local

Our **repo** is now cloned to both our **local** and **remote** machines. We are now able to transfer files! We'll create a text file on our **local**, push that file to the **repo**, and pull the file to the **remote** machine.

*It may help to have two command line prompts open: one connected to your **local** machine and one connected to your **remote** machine.*



```
kiera@DESKTOP-HAH5D8H MINGW64 ~/Documents/GitHub
$ cd EC2-Test

kiera@DESKTOP-HAH5D8H MINGW64 ~/Documents/GitHub/EC2-Test (master)
$ touch transfer.txt

kiera@DESKTOP-HAH5D8H MINGW64 ~/Documents/GitHub/EC2-Test (master)
$ git add transfer.txt

kiera@DESKTOP-HAH5D8H MINGW64 ~/Documents/GitHub/EC2-Test (master)
$ git commit -m "added empty text file"
[master (root-commit) 9417804] added empty text file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 transfer.txt

kiera@DESKTOP-HAH5D8H MINGW64 ~/Documents/GitHub/EC2-Test (master)
$ git push
Enumerating objects: 3, done.
```

1. In your command line, ensuring you're in the repository directory on your local, create an empty text file:

   `touch transfer.txt`

2. Next, we'll add this change (the new file), to the staging area:

   `git add transfer.txt`

3. We can now commit this change to our local repository, creating a checkpoint[1]:

   `git commit -m "<message>"`

4. Lastly, we'll push the changes on the local to our GitHub repo[2]:

   `git push`

1. *Replace `<message>` with a short message describing the changes you're committing, but leave the quotation marks.. E.g.: `"added text file"`*
2. *If you haven't pushed using this account before, you may be asked to enter a user.name. This name will appear on GitHub next to your commit messages.*

Using GitHub - Step 5: Pull from Repo to Remote

We have pushed our text file to the GitHub **repo**. Now, we simply need to pull (sync changes) from the repo to our **remote** machine.

1. In your command line, ensure you're connected to your EC2 instance via SSH. *cd* into the repository directory:

   `ssh explore-student@<IPv4 address>`

2. Jump into your repository, and check what's in it. It should be empty - unless you initialised it with a Readme when you created it.

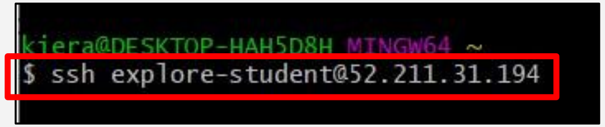
   `cd <repository name>`, and then `ls`

3. It's good practice to status check a repository before pulling/pushing, to ensure there are no pending changes:

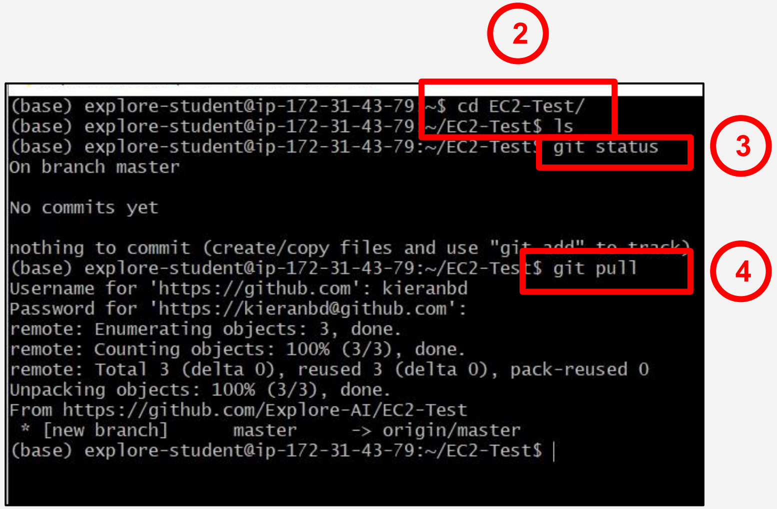
   `git status`

4. Pull the changes from the GitHub **repo** to your **remote**:

   `git pull`



```
kiera@DESKTOP-HAH5D8H MINGW64 ~
$ ssh explore-student@52.211.31.194
```



```
(base) explore-student@ip-172-31-43-79:~$ cd EC2-Test/
(base) explore-student@ip-172-31-43-79:~/EC2-Test$ ls
(base) explore-student@ip-172-31-43-79:~/EC2-Test$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
(base) explore-student@ip-172-31-43-79:~/EC2-Test$ git pull
Username for 'https://github.com': kieranbd
Password for 'https://kieranbd@github.com':
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/Explore-AI/EC2-Test
 * [new branch]      master     -> origin/master
(base) explore-student@ip-172-31-43-79:~/EC2-Test$ |
```

Transferring files to your EC2 Instance using Github

- Nice! You have just transferred a file from you local machine to your remote EC2 instance.
- To check that the transfer worked, simply run the `ls` command on your remote machine again to print the contents of the repository.
- Using this process, you can transfer notebooks to your remote EC2 instance. After launching Jupyter Notebook from your instance, you can navigate to the appropriate directory to find the files you transferred. Each instance has only ~10GB of storage space - please use and manage your space wisely.
- As you may have worked out, this can be an onerous process - especially with multiple or constantly-changing files. GitHub is primarily for code storage and version control, and may not always be the best option for transferring from **local** to **remote**.
    - This is where **SCP** comes in.

Using SCP - Step 1: Bash Command

1. The Secure Copy Protocol (SCP) is a network protocol that uses the Secure Shell (SSH) protocol to transfer files securely between local and remote hosts.

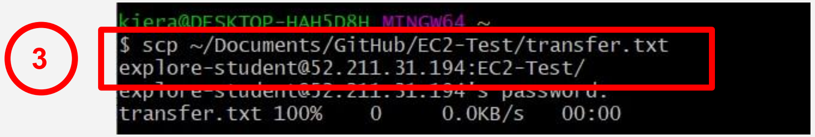
2. The commands to transfer files using SCP are very simple: one for local to remote, and another for remote to local. The commands are to be issued in your command prompt.

3. Copying a file from local to remote:

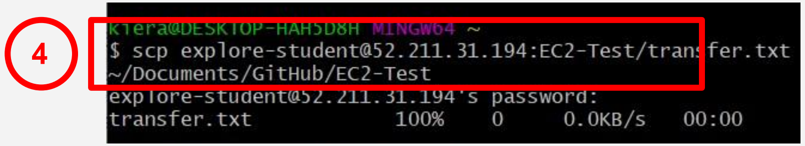
   `scp <path_to_file> explore@<IPv4-address>:<location_to_transfer_to>`

4. Copying a file from remote to local:

   `scp explore@<IPv4-address>:<path_to_file> <location_to_transfer_to>`



```
kiera@DESKTOP-HAH5D8H MINGW64 ~
$ scp ~/Documents/GitHub/EC2-Test/transfer.txt
explore-student@52.211.31.194:EC2-Test/
explore-student@52.211.31.194's password:
transfer.txt 100%     0      0.0KB/s    00:00
```



```
kiera@DESKTOP-HAH5D8H MINGW64 ~
$ scp explore-student@52.211.31.194:EC2-Test/transfer.txt
~/Documents/GitHub/EC2-Test
explore-student@52.211.31.194's password:
transfer.txt                100%    0     0.0KB/s    00:00
```

Conclusion

In this train we've covered the following:

File Transfer Ecosystem

At a high level we discussed the local -> Repo -> Remote file transfer ecosystem

GitHub File Transfer

By means of a worked example, we looked at how we can use GitHub to transfer files to an EC2 instance

SCP File Transfer

We discussed how SCP can be used to:

- Copy files from a local to a remote machine and
- Copy files from a remote to a local machine

EXPLORE DIGITAL SKILLS