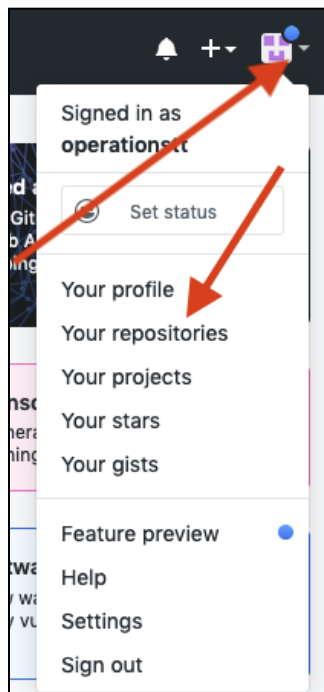


## Configuring SSH for Git

In this lab, you will create and implement SSH keys for Git access

### Part 1: Creating a GitHub Account

1. Go to github.com
2. If you have an existing GitHub account you can use it for this class, or you can sign up for a new account
3. If you choose to sign up for a new account, click 'Sign up' at the top right
  - a. Create a username. You might need to use a personal email address if your work email security policy doesn't allow confirmation emails to verify your account
  - b. When offered to 'Pick your trial plan', don't pick any option. Just click the dropdown under the user icon on the top right, and click 'Your repositories'



- c. You will need to check your email and verify that your email address is real

4. Once you click the email verification, it will take you to create a new repository in your git account. Use the following steps to create your new repo

### Configuring SSH Connections to GitHub

5. Create a private repository in your GitHub account called “repo-lab”
  - a. Name: repo-lab
  - b. Choose ‘Private’
  - c. Do not initialize with a README
  - d. Click the green ‘Create Repository’ button

## Create a new repository

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

---

Owner

quazzle ▾

/

Repository name \*

repo-lab ✓

Great repository names are short and memorable. Need inspiration? How about **fictional-succotash**?

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

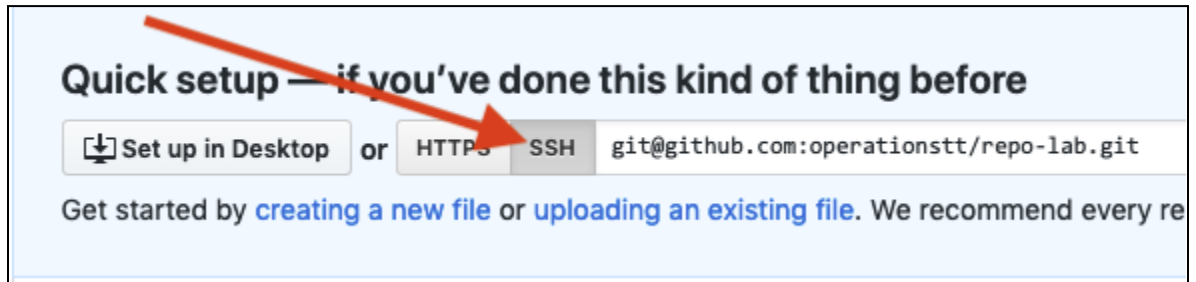
This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾

Add a license: None ▾

ⓘ

6. Copy the SSH URL under 'Quick Setup'



7. On your remote workstation: in the my-repo directory:

**\$ git remote add origin <url-you-copied-from-github>**

```
/home/ubuntu/my-repo --$ git remote add origin git@github.com:operationstt/repo-lab.git
/home/ubuntu/my-repo --$
```

8. Verify this operation worked with:

**\$ git remote -v**

```
/home/ubuntu/my-repo --$ git remote -v
origin  git@github.com:operationstt/repo-lab.git (fetch)
origin  git@github.com:operationstt/repo-lab.git (push)
/home/ubuntu/my-repo --$
```

9. To understand the remote mapping, remove the mapping to origin with:

**\$ git remote rm origin**

10. Try to look at the mappings again, and notice there is no mapping:

**\$ git remote -v**

```
/home/ubuntu/my-repo --> git remote -v
/home/ubuntu/my-repo -->
```

11. Re-create the mapping for origin once again, and verify that it worked:

**\$ git remote add origin <ssh\_url\_you\_copied>**

**\$ git remote -v**

```
/home/ubuntu/my-repo --> git remote add origin git@github.com:op
/home/ubuntu/my-repo --> git remote -v
origin  git@github.com:operationstt/repo-lab.git (fetch)
origin  git@github.com:operationstt/repo-lab.git (push)
```

12. Try pushing to the repo, but this should **fail** due to lack of permissions:

**\$ git push origin master**

```
/home/ubuntu/my-repo --> git push origin master
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
/home/ubuntu/my-repo --> █
```

13. To fix this, we need to follow these **four** steps:

- a. Create an SSH key pair
- b. Add the PUBLIC key to GitHub
- c. Add the PRIVATE key to your workstation keychain
- d. GitHub will use the PUBLIC key that you uploaded to GitHub in step 'B' to decrypt the communication from the PRIVATE key from your workstation, AUTHENTICATING you as an authorized user

**Now we'll go through the 4-steps listed above:**

#### **A) Create an SSH key pair**

14. Create the SSH directory if it doesn't already exist

**\$ mkdir /home/<yourname>/.ssh**

**Note:** You could also use the shorthand notation for your home directory

**\$ mkdir ~/.ssh**

Either command will work

**Note:** In the following command do NOT use the default key filename of:

**/home/ubuntu/.ssh/id\_rsa**

15. Generate a new SSH key named "repo-key". Note that "-t" defines the type of key as an 'RSA Protocol Version 2' key, and the -b flag defines the number of encryption bits as 4K

- a. **\$ ssh-keygen -t rsa -b 4096**

- b. "Enter file in which to save the key" = /home/ubuntu/.ssh/repo-key

This names the file as well as defines the directory

```
/home/ubuntu/.ssh --> ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa): /home/ubuntu/.ssh/repo-key
Enter passphrase (empty for no passphrase):
```

- c. Three VERY important things to follow:

-- Do NOT use the default filename of /home/ubuntu/.ssh/id\_rsa, instead use:

/home/ubuntu/.ssh/repo-key

-- Do not use the ~ shortcut here. Spell out the entire path of:

/home/ubuntu/.ssh/repo-key

-- Do NOT use a passphrase

- d. Hit enter when requested for a passphrase (to leave it empty), confirm with another <enter>

16. Verify the existence of these two files:

/home/ubuntu/.ssh/repo-key (the private key)

/home/ubuntu/.ssh/repo-key.pub (the public key)

```
+-----[SHA256]-----+
/home/ubuntu/.ssh --> ls
authorized_keys  repo-key  repo-key.pub
/home/ubuntu/.ssh --> █
```

## B) Add the PUBLIC key to GitHub

17. Copy the contents of the PUBLIC key using the command:

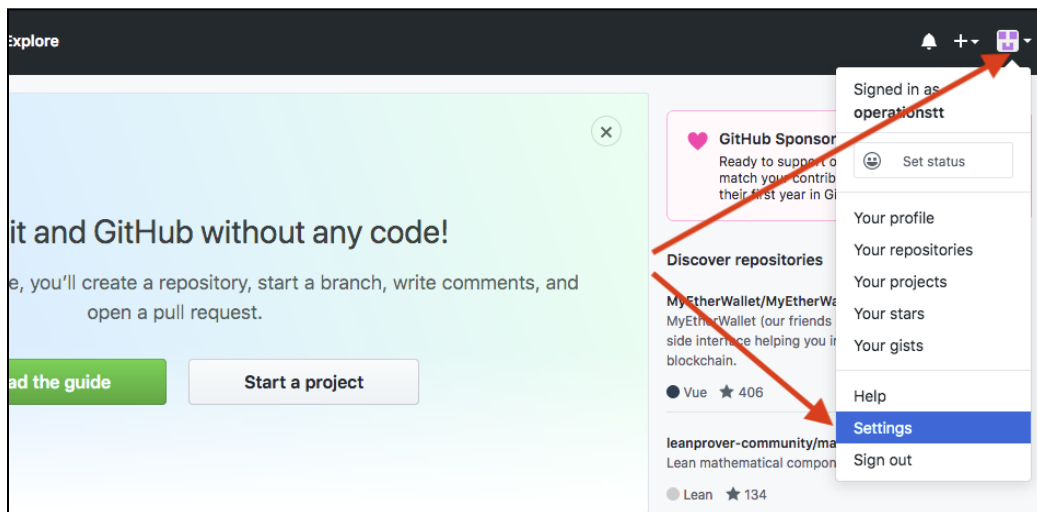
**\$ cat repo-key.pub**

Then copy the contents

```
/home/ubuntu/.ssh --> cat repo-key.pub  
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDcrSzS  
5LtoAv689bvbn0k0Wu3rdzNK0SRXmAd2WhFAWYXwaw/9  
16W/XWQD1UI/2VLwC3AEoAL+A3IjLbUJeOCj1DmH0UGt  
R1V1sc2c+UV8u2WmUdhMVw6k71bcnfsxXvQAYPudSaoE  
D47q+AcS00WNeCtgE5DnS3dc3AsuH+E0G7EIKtVesh73  
T1kTs/Vzkq0ooFcGyujcWSt4X35U00jbGrHBKftpAJZy  
X+PMrL106FFEQ0cgySh0zAMNFMEPSHQAj6/QtGwaUGLc  
/home/ubuntu/.ssh -->
```

18. In GitHub, add the PUBLIC key to the GitHub user “settings” (use the ‘settings’ link under the user avatar at the top right, NOT the ‘settings’ link in your specific repo)

We do this because once GitHub has the PUBLIC KEY, then our workstation can authenticate to it using the PRIVATE KEY



19. Go to: Settings -> SSH & GPG keys -> SSH Keys
20. Click 'New SSH key'
21. Title “Public repo-key”

22. Paste the entire contents of **repo-key.pub** into the 'key' box

SSH keys / Add new

Title

repo-key

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCDcrSzSiAQuqZanXmY3ttjXnjhDQcX9EodrRPXhh/SjGGIDA1LzXC4FcD
/4sIjd9dDxdE15LtoAv689bvcnOk0Wu3rdzNK0SRXmAd2WhFAWYXwaw/9mSwnU/28CO4ZPm7DZ4LPtt5YvL
/68xS0wyUO8TwhH6YDIZ6D2p5JcCF5Z0ia3TY16W/XWQD1UI
/2VLwC3AEoAL+A3IjLbUJeOCj1DmH0UGtQlecCTQEIVEIsHGDE3UC
/z2VMQoVjEfDOM7E0sIA1Qfj2iIC1KZw1EsyvNAIDZJRIVisc2c+UV8u2WmUdhMVw6k71bcnfsxXvQAYPudSaoBt1
yAdAQOVLcIzNIZfZmo7IEImoejKfZKj
/cgUI4TXKKNwwE9epJDO4IbplmNJFkD47q+AcSO0WNeCtgE5DnS3dc3AsuH+E0G7EIKtVesh73Vp
/zHbYI75q9svTGW26NMB+IoN3IQzjjHcP8Lus+iec7guFW2P0YDVntO0n6nQT1kTs
/VzKqOooFcGyujcWSt4X35UOOjbGrHBKftpAJZy1BWIT
/4vgPZESKxyJkujxbwX32jaljR5chuUhmZsYokYAerWwK8WZDbC+L6pyOFX+PMrLIO6FFEQOcgYSh0zAMNFME
PSHQA6jQtGwaUGLdxhC6pjENLFqg9ASetvNHwQ/3IEedZYV8UGw== ubuntu@ip-172-31-28-73
```


Add SSH key

23. Click 'Add SSH key'

24. Enter your GitHub password if requested

SSH keys New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



**repo-key**  
cf:8a:08:99:c9:22:0e:e8:f8:c2:a4:4b:99:7e:35:78  
Added on Dec 10, 2019  
Never used — Read/write

SSH

Delete

25. cd back to your my-repo directory

26. Push the repo with **\$ git push origin master**

**It will fail again!**

### C) Add the PRIVATE key to your workstation keychain

This still fails because you haven't added this new private key to your keychain so that your computer can use it to automatically authenticate on your behalf

```
/home/ubuntu/my-repo --> git push origin master
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
/home/ubuntu/my-repo -->
```

27. Open the SSH agent interface

**\$ eval \$(ssh-agent)**

```
ubuntu@ip-172-31-9-16:~$ eval $(ssh-agent)
Agent pid 20375
```

28. Add the key to the keychain with `$ ssh-add path/to/private_key`

**\$ ssh-add /home/ubuntu/.ssh/repo-key**

```
ubuntu@ip-172-31-9-16:~$ ssh-add ~/.ssh/repo-key
Identity added: /home/ubuntu/.ssh/repo-key (/home/ubuntu/.ssh/repo-key)
```

(D) GitHub will use the PUBLIC key that you uploaded to GitHub in step (B) to decrypt the communication from the PRIVATE key that you added to your workstation keychain in step (C), AUTHENTICATING you as an authorized user. **Please note: if you close and RE-OPEN the terminal, you will need to do the eval and ssh-add again to properly authenticate**

29. Return to the 'my-repo' directory

30. Push to the repo again, which should succeed now that everything is in place:

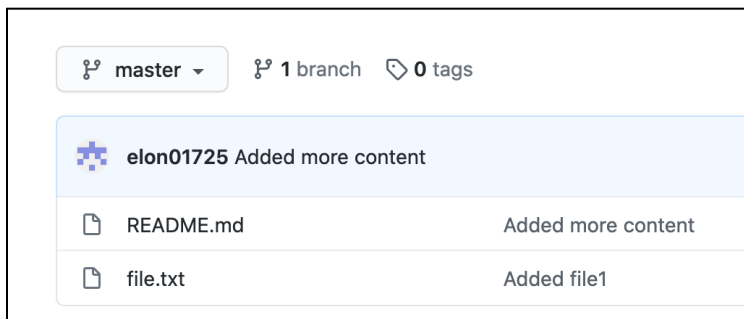
**\$ git push origin master**

**Note: you may need to agree to continue the connection. Type 'yes' to continue**



```
/home/ubuntu/my-repo --> git push origin master
The authenticity of host 'github.com (192.30.253.113)' can't be
RSA key fingerprint is SHA256:nThbg6kXUpJWG17E1IGOCspRomTxdCAR
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.253.113' (RSA) to
Counting objects: 12, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (12/12), 972 bytes | 0 bytes/s, done.
Total 12 (delta 0), reused 0 (delta 0)
To git@github.com:operationstt/repo-lab.git
 * [new branch]      master -> master
/home/ubuntu/my-repo -->
```

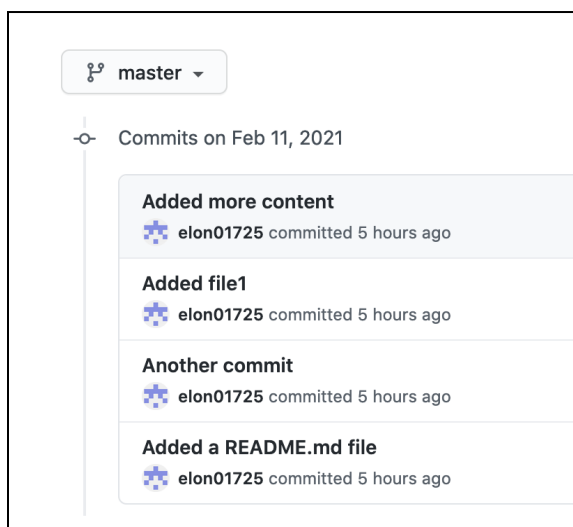
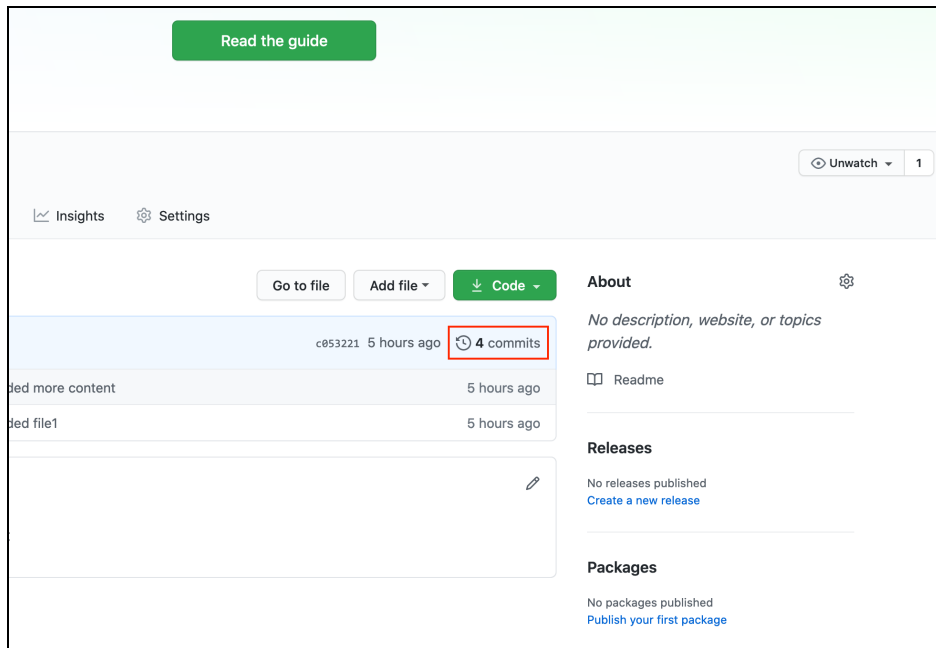
31. Refresh the GitHub web interface and you should see your files in the repo



32. You may need to click the dropdown at the top right of the GitHub interface and click 'Your repositories' to find your repo

Note there is only 1 branch. While we have more branches on our **local system**, we haven't pushed them to the GitHub repo yet

33. Click 'commits'. Note this is the same info you saw with '\$ git log'. This shows all of the commits done on this repo, even when it was only local to your workstation before you pushed to the remote repo



Note the user, showing who made the commit

## A shortcut for git push

34. To make our lives so much easier, let's shortcut "git push origin master" to the infinitely easier command of "git push"
35. Open the README file in the master branch
36. Make a change to the content of the file. Save, add and commit the file
37. Save the shortcut to the current remote repo and branch using the --set-upstream (or -u) flag. **Either option below works**

**\$ git push --set-upstream origin master**

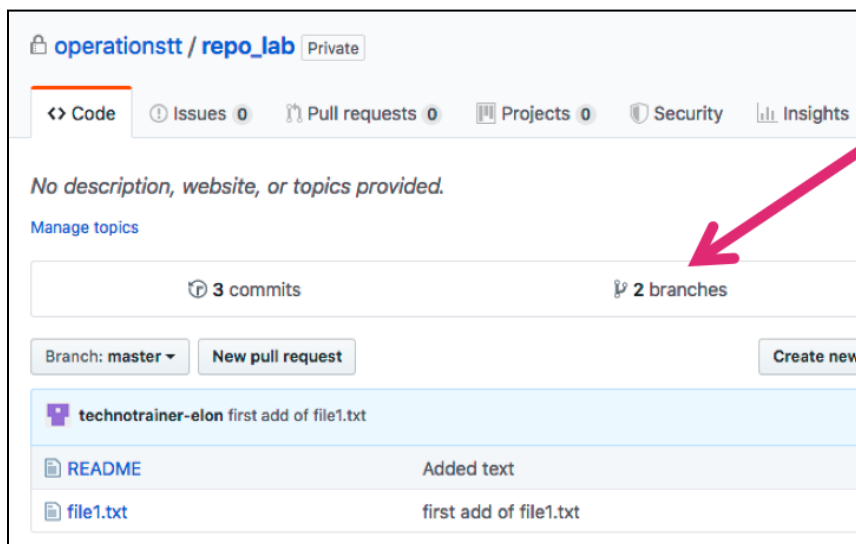
or use the shortcut of -u

**\$ git push -u origin master**

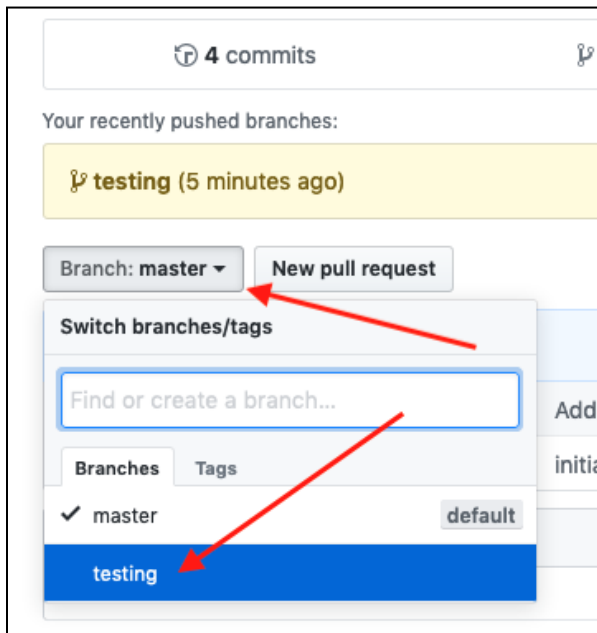
38. Test that your shortcut is working
  - a. Make another change to the README file on the master branch
  - b. Save, add and commit the file
  - c. Push the file to GitHub with:  
**\$ git push**
  - d. Your push should go through. From now on, when pushing to this specific repo and this specific branch, you can just use  
**\$ git push**  
instead of  
**\$ git push origin master**

## Create a New Branch on GitHub

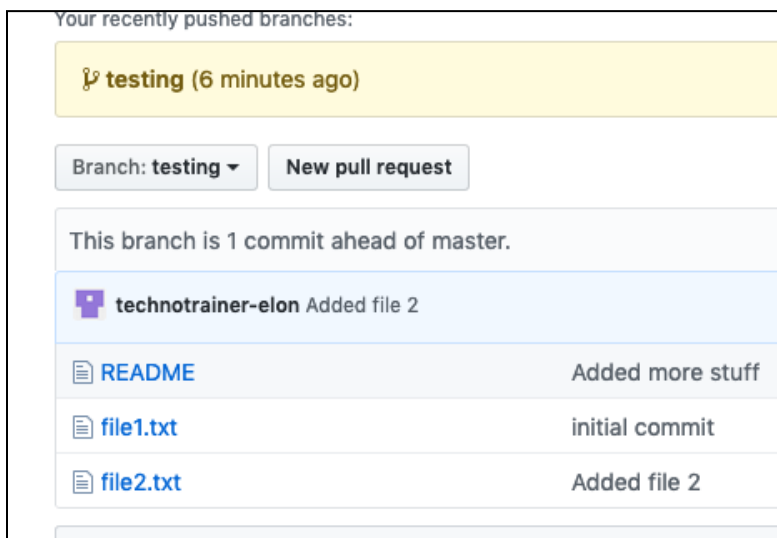
39. On your remote workstation, checkout the testing branch. Refer to the 'Basic Git Commands' lab if you don't remember how to do this
40. Create a new file named "file2.txt" with some content, **add** and **commit** this file **in the testing branch**
41. Push this to a testing branch on the remote repo:  
**\$ git push origin testing**  
  
*Note that our "\$ git push" shortcut only points to the 'master' branch. Since you are pushing to the 'testing' branch you need to specify the branch here*
42. Refresh the GitHub interface to verify that the new 'testing' branch exists on GitHub. You may need to click on the repo name at the top of the page to see the branches



43. Click the Branch dropdown, and choose 'testing'



44. Verify your new file2.txt is in the branch on GitHub



## Merge Files On GitHub

45. On **GitHub**, go back to the master branch (using the dropdown you used in the last step)
46. Look at the files in the master branch. You should have 'README' and 'file1.txt'. You don't see file2.txt because this file belongs to the 'testing' branch
47. Checkout the master branch **on your workstation**
48. Pull the testing branch. This will copy the files from the 'testing' branch to the local 'master' branch  
**\$ git pull origin testing**
49. You will be presented with an editor window where you can write a commit message. Accept the default commit message by saving the file and exiting the editor
50. Look at your local files. Note that **file2.txt** has been added to the list of files in the master branch on your workstation
51. Commit this to master. Note that your commit might show as "nothing to commit". This is OK because you did a 'git pull'  
**\$ git add .**  
**\$ git commit**  
**\$ git push**
52. Refresh GitHub to verify the master branch now contains file2.txt

## Git End-To-End

53. Create and checkout a new 'staging' branch
54. Verify that you are on the 'staging' branch, and note the files in the branch (they are the same as the branch you were on when you created the new 'staging' branch)
55. Create a file in this branch named 'staging1.txt' with some content
56. Add, commit and push this to the 'staging' branch on GitHub
57. Checkout the master branch on your workstation
58. Pull the master branch down

59. Merge the staging branch into the master branch

60. Push the new resulting local master branch to the GitHub master branch

61. Verify the file you created for the staging branch is now in the master branch on GitHub

**Notify your instructor that you are done with the lab**

**END OF LAB**