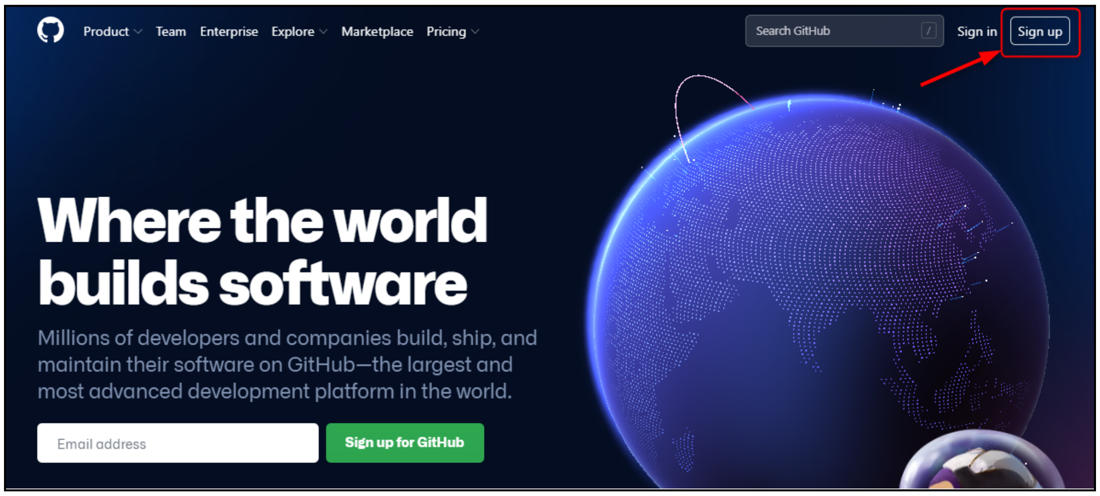
Objective : In this lab, you will create a GitHub account (if you don't already have one), and then create and implement SSH keys for git access.

Creating a GitHub Account:

1. Open your web browser and navigate to **github.com**

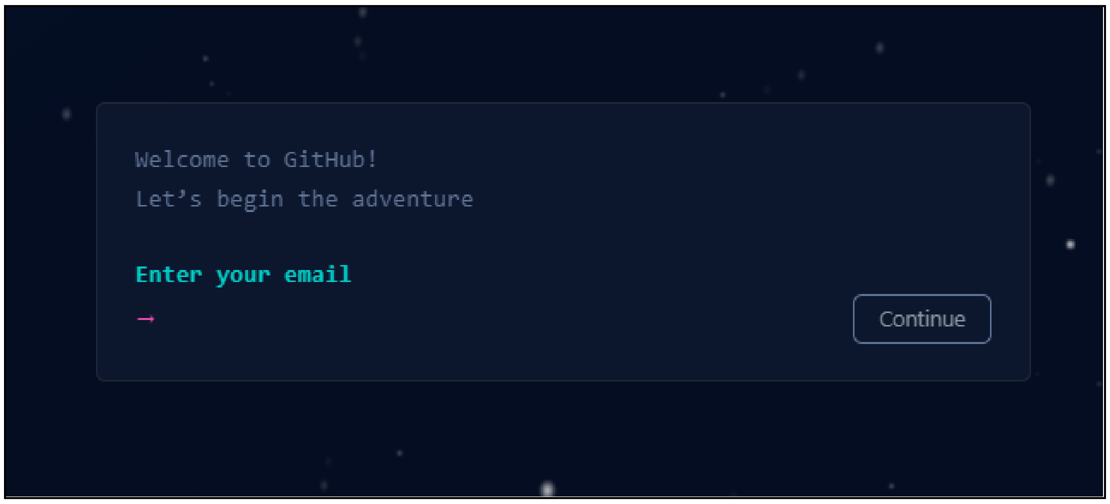
2. Click on **'Sign up'** in the top right corner



Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

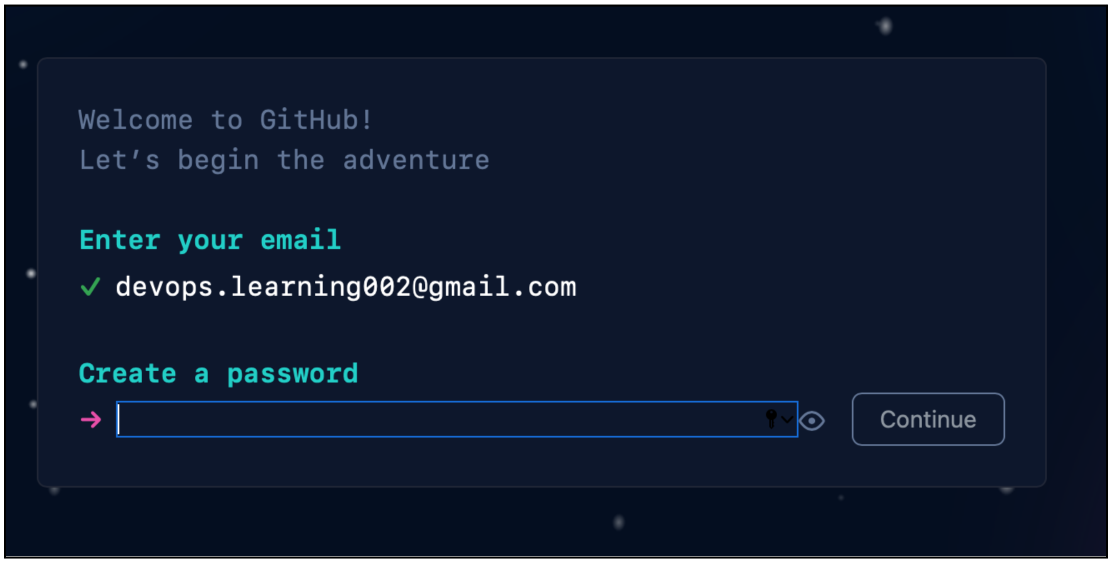
3. Enter the email address that will be used as a credential to login to your GitHub account. Then click on **Continue**



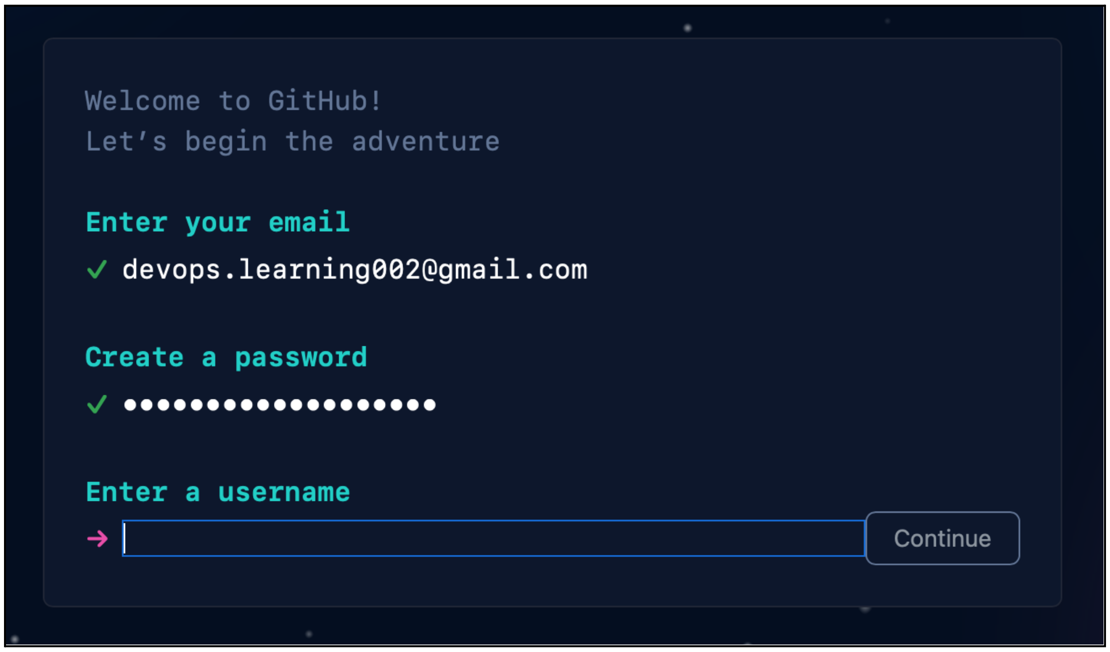
Welcome to GitHub!
Let's begin the adventure

Enter your email
→

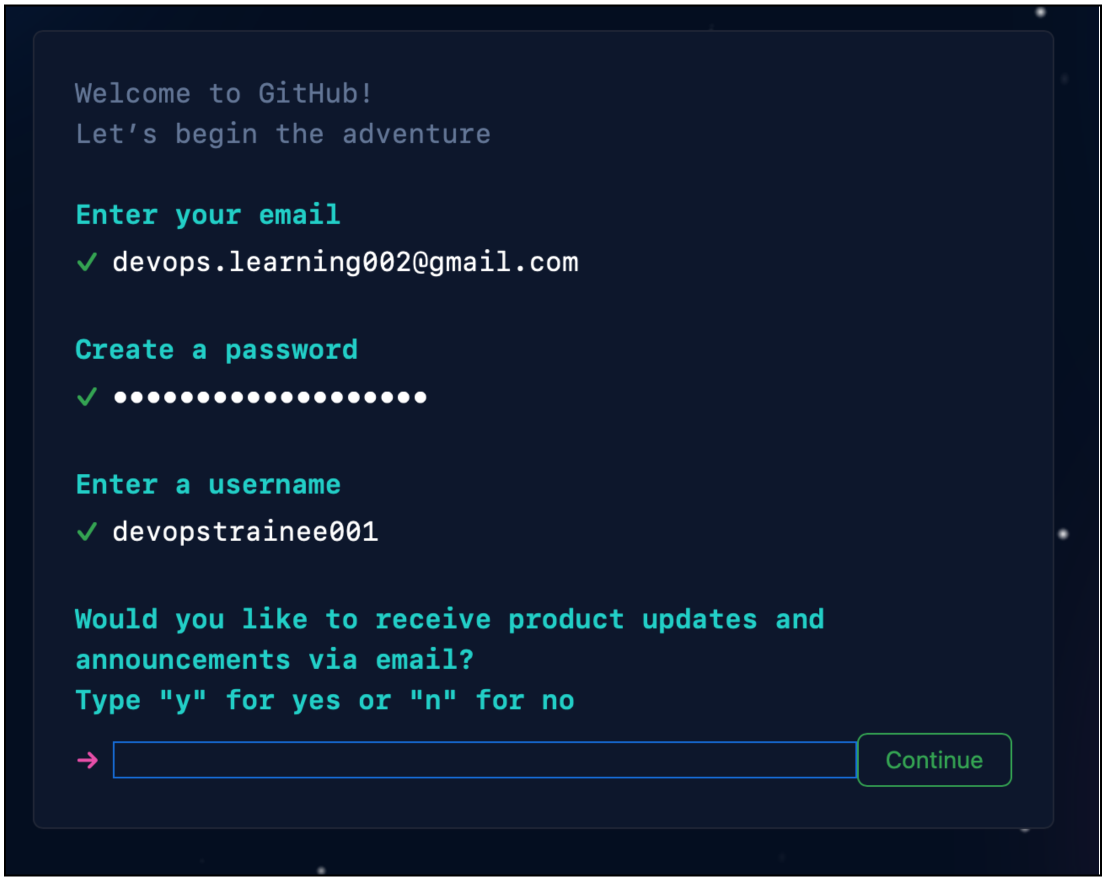
4. Enter your password in **'Create a password'** field and click on **Continue**


```
Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ devops.learning002@gmail.com

Create a password
→
```

5. Enter any content in the **'Enter a username'** field and click on **Continue**


```
Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ devops.learning002@gmail.com

Create a password
✓ ••••••••••••••••••

Enter a username
→
```

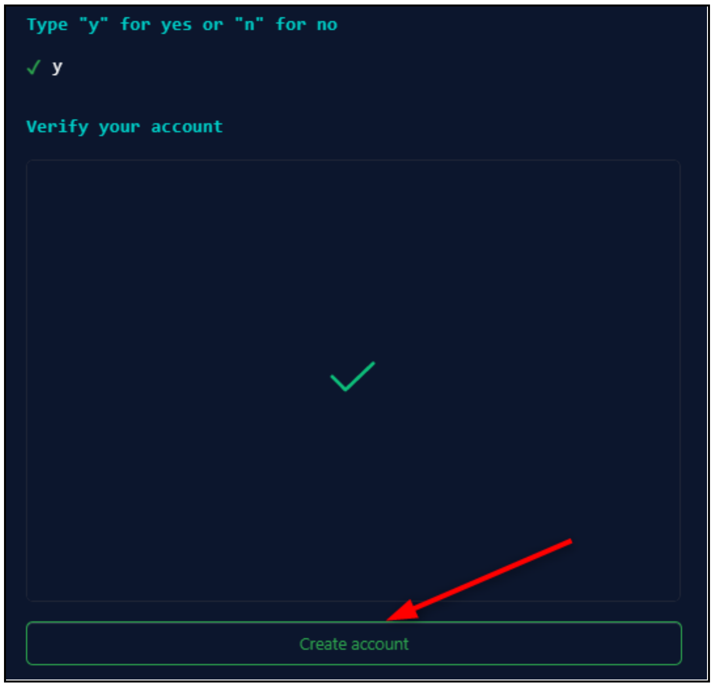
6. Type either **"y"** for **yes** or **"n"** for **no** according to your preference regarding whether you want to **receive product updates and announcements via email**



```
Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ devops.learning002@gmail.com

Create a password
✓ ••••••••••••••••••

Enter a username
✓ devopstrainee001

Would you like to receive product updates and
announcements via email?
Type "y" for yes or "n" for no
→
```

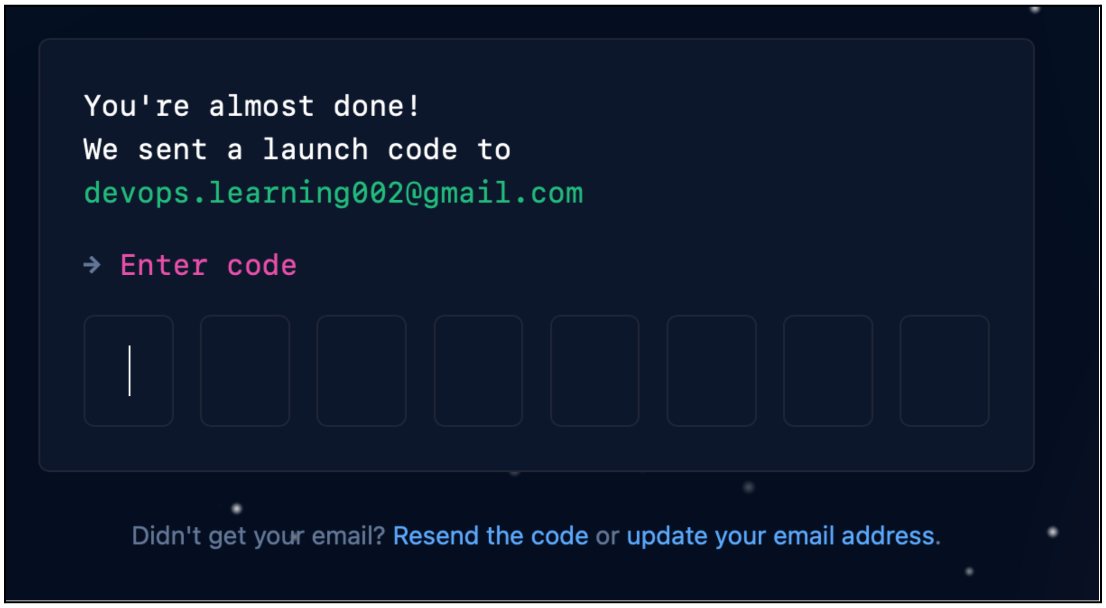
7. It will ask you to solve a puzzle by clicking on **Start Puzzle**. Solve the puzzle accordingly

8. Once the puzzle verification succeeded, click on **Create account**



```
Type "y" for yes or "n" for no
✓ y

Verify your account
```

9. Enter the verification code (sent to the email account that you used to create your GitHub account)

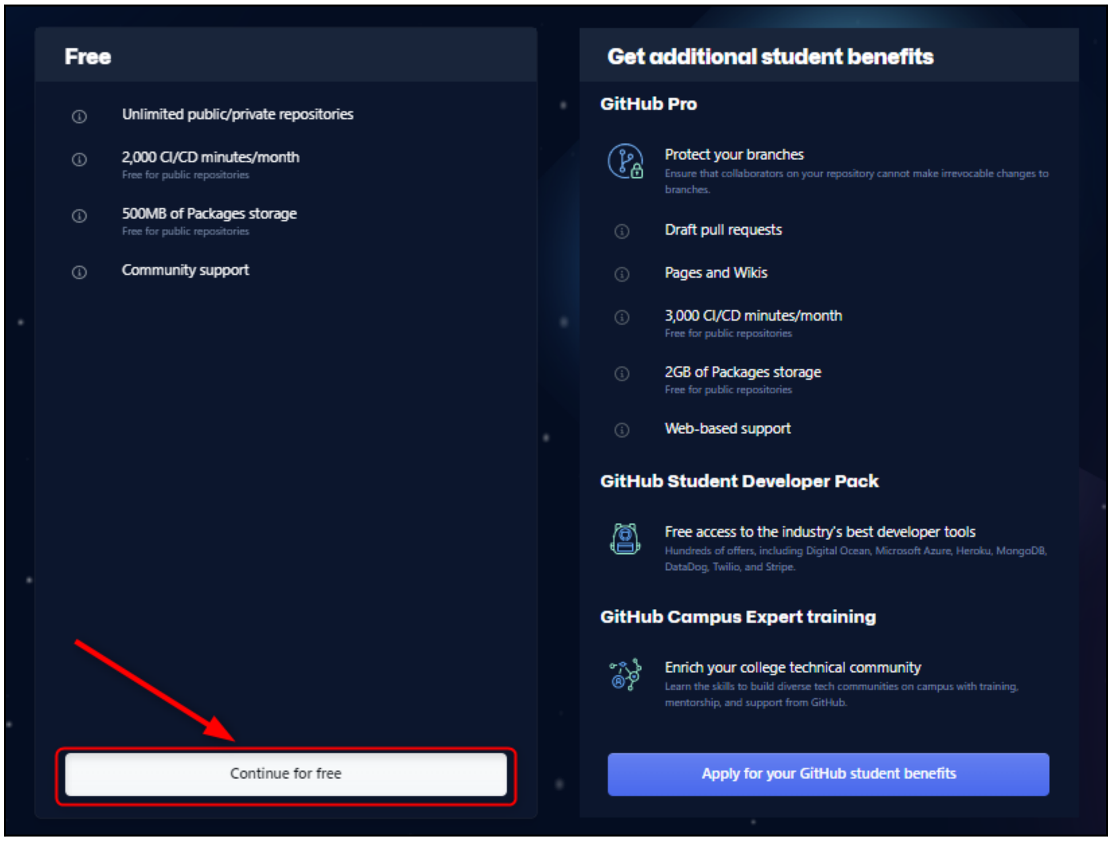


```
You're almost done!
We sent a launch code to
devops.learning002@gmail.com

→ Enter code
```

Didn't get your email? Resend the code or update your email address.

10. In this step, it will ask you some questions. Provide inputs as per your need

11. Click on **Continue for free**



Free

- Unlimited public/private repositories
- 2,000 CI/CD minutes/month
  Free for public repositories
- 500MB of Packages storage
  Free for public repositories
- Community support

Get additional student benefits

GitHub Pro

- Protect your branches
  Ensure that collaborators on your repository cannot make irrevocable changes to branches.
- Draft pull requests
- Pages and Wikis
- 3,000 CI/CD minutes/month
  Free for public repositories
- 2GB of Packages storage
  Free for public repositories
- Web-based support

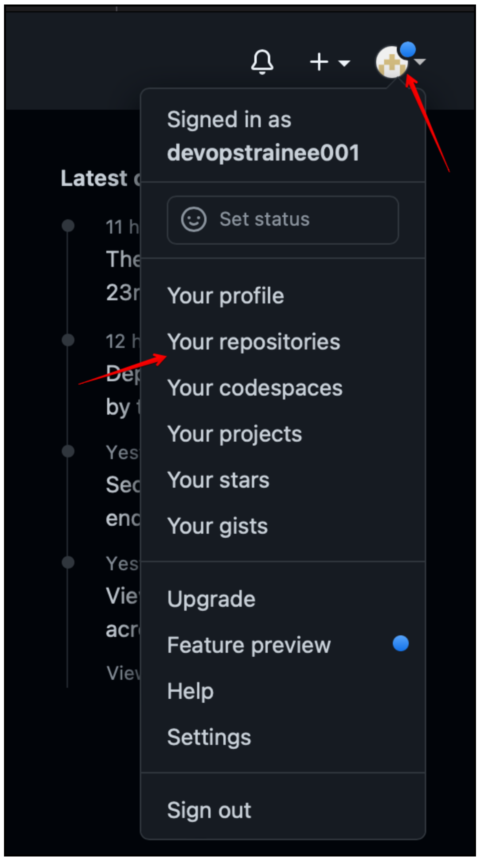
GitHub Student Developer Pack

- Free access to the industry's best developer tools
  Hundreds of offers, including Digital Ocean, Microsoft Azure, Heroku, MongoDB, DataDog, Twilio, and Stripe.

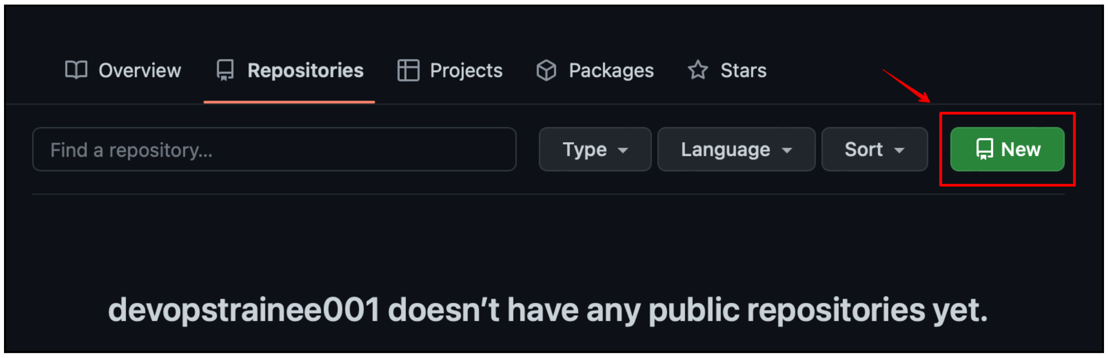
GitHub Campus Expert training

- Enrich your college technical community
  Learn the skills to build diverse tech communities on campus with training, mentorship, and support from GitHub.

12. Create a new GitHub repository by clicking on the **drop down** menu in the top right corner. Then click on '**Your repositories**'

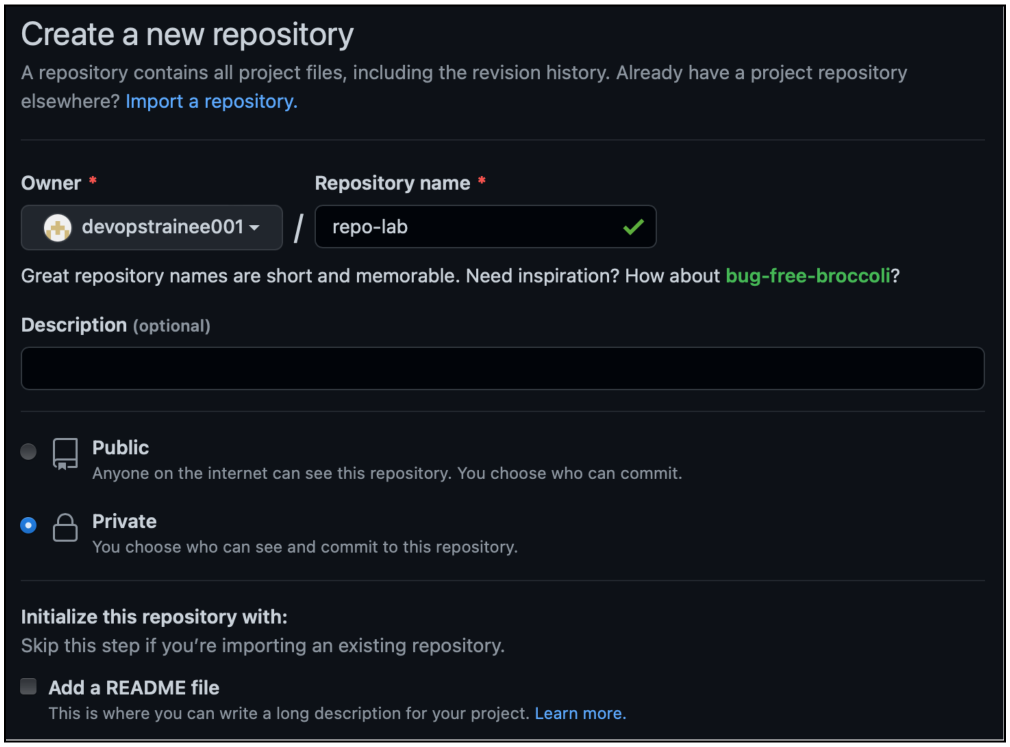


13. Next click on **New** in the top right corner



14. Provide the name '**repo-lab**' in the **Repository name** field

- Select **Private** instead of **Public**


Create a new repository

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

Owner * devopstrainee001 / Repository name * repo-lab

Great repository names are short and memorable. Need inspiration? How about bug-free-broccoli?

Description (optional)

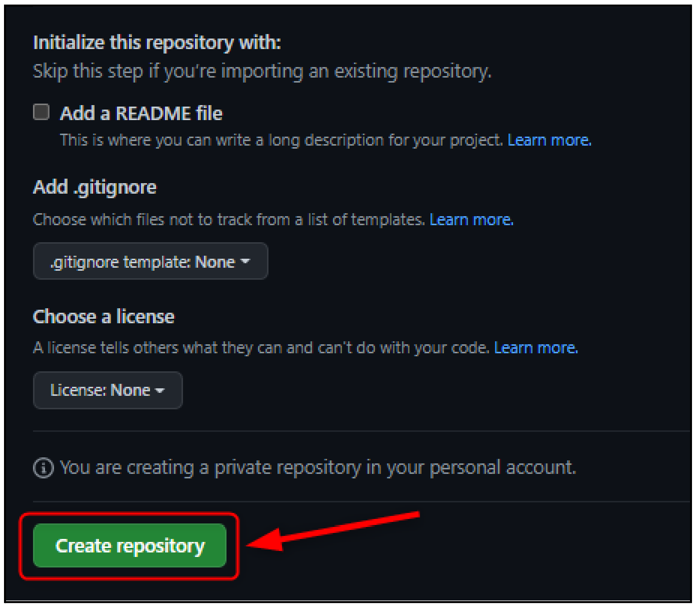
Public
Anyone on the internet can see this repository. You choose who can commit.

Private [✓]
You choose who can see and commit to this repository.

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

Add a README file
This is where you can write a long description for your project. Learn more.

- Scroll down and click on **Create repository**


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

Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

.gitignore template: None

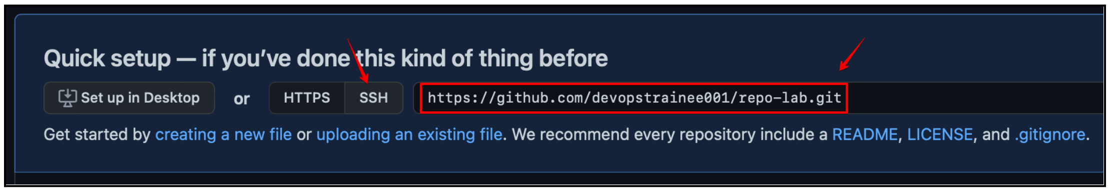
Choose a license
A license tells others what they can and can't do with your code. Learn more.

License: None

You are creating a private repository in your personal account.

Configuring SSH Connections to GitHub

15. Copy the SSH URL under **'Quick Setup'**



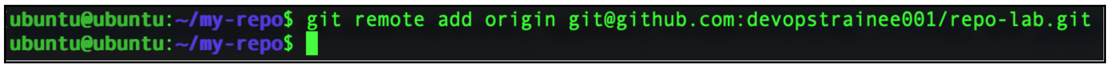
Quick setup — if you've done this kind of thing before

```
https://github.com/devopstrainee001/repo-lab.git
```

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

16. On your ubuntu remote workstation, in the **my-repo** directory:

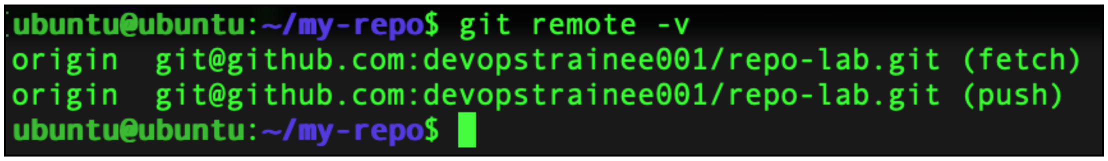
$ git remote add origin <url-you-copied-from-github>



```
ubuntu@ubuntu:~/my-repo$ git remote add origin git@github.com:devopstrainee001/repo-lab.git
ubuntu@ubuntu:~/my-repo$
```

17. To verify whether the origin is added:

$ git remote -v



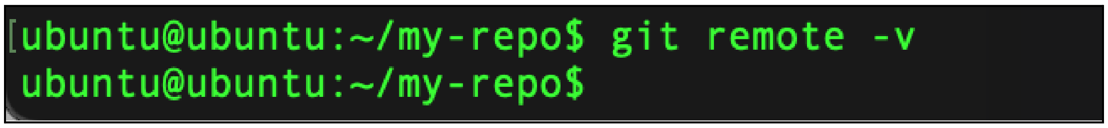
```
ubuntu@ubuntu:~/my-repo$ git remote -v
origin  git@github.com:devopstrainee001/repo-lab.git (fetch)
origin  git@github.com:devopstrainee001/repo-lab.git (push)
ubuntu@ubuntu:~/my-repo$
```

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

$ git remote rm origin

19. Try to look at the mappings again:

$ git remote -v



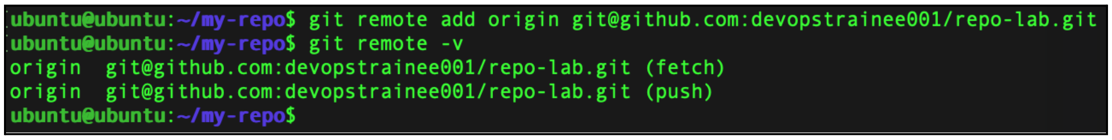
```
ubuntu@ubuntu:~/my-repo$ git remote -v
ubuntu@ubuntu:~/my-repo$
```

Note: You will notice that there is no mapping to any origin

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

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

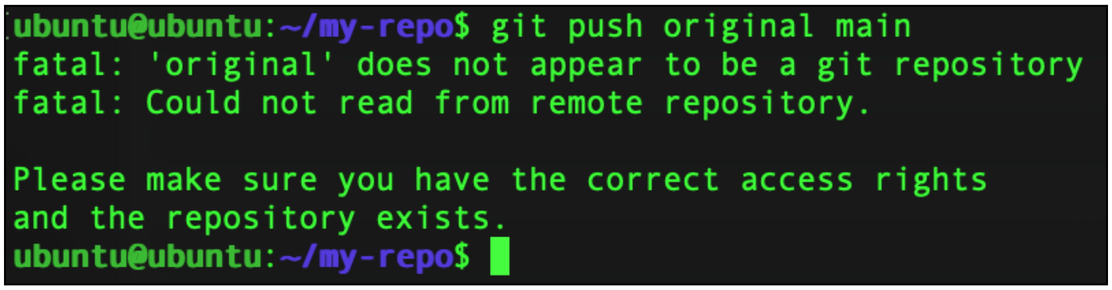
$ git remote -v



```
ubuntu@ubuntu:~/my-repo$ git remote add origin git@github.com:devopstrainee001/repo-lab.git
ubuntu@ubuntu:~/my-repo$ git remote -v
origin  git@github.com:devopstrainee001/repo-lab.git (fetch)
origin  git@github.com:devopstrainee001/repo-lab.git (push)
ubuntu@ubuntu:~/my-repo$
```

21. Try pushing to the repo. This should **fail** due to lack of permissions:

$ git push origin main



```
ubuntu@ubuntu:~/my-repo$ git push original main
fatal: 'original' 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.
ubuntu@ubuntu:~/my-repo$
```

Hint: To fix this, we need to follow these four steps:

- Create an SSH key pair (step A)
- Add the PUBLIC key to GitHub (step B)
- Add the PRIVATE key to your workstation keychain (step C)
- 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 (step D)

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

Create an SSH key pair (A)

22. If it doesn't already exist, create the SSH directory:

$ mkdir /home/ubuntu/.ssh

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

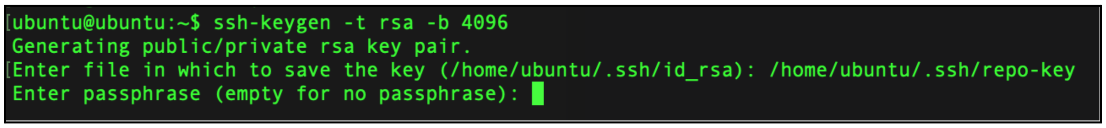
$ mkdir ~/.ssh

Note: Either command will work

23. Next, generate a new SSH key with the name "**repo-key**":

    a. cd ~
    b. **$ ssh-keygen -t rsa -b 4096**
    c. "Enter file in which to save the key" = **/home/ubuntu/.ssh/repo-key** (use the full path)

       Note: This will provide the name of the file as well as define the directory



```
[ubuntu@ubuntu:~$ 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): █
```

    d. 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:

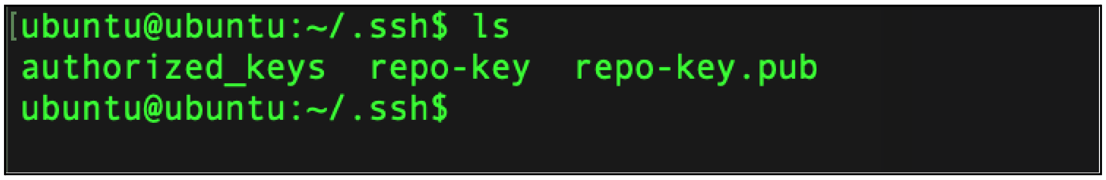
         /home/ubuntu/.ssh/repo-key
       - Do NOT use a passphrase

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

24. Verify the existence of these two files:

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

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

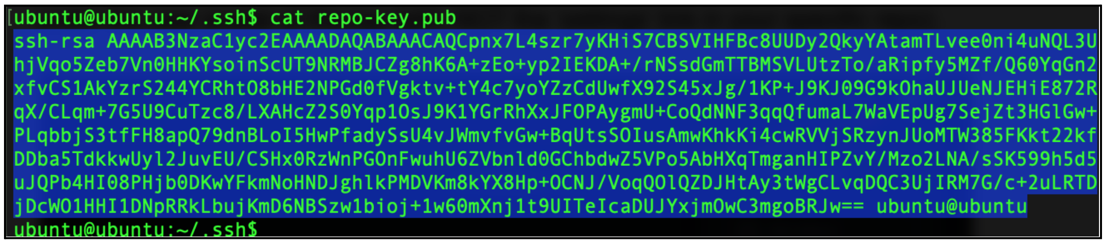


```
[ubuntu@ubuntu:~/.ssh$ ls
 authorized_keys  repo-key  repo-key.pub
 ubuntu@ubuntu:~/.ssh$
```

Add the PUBLIC key to GitHub (B)

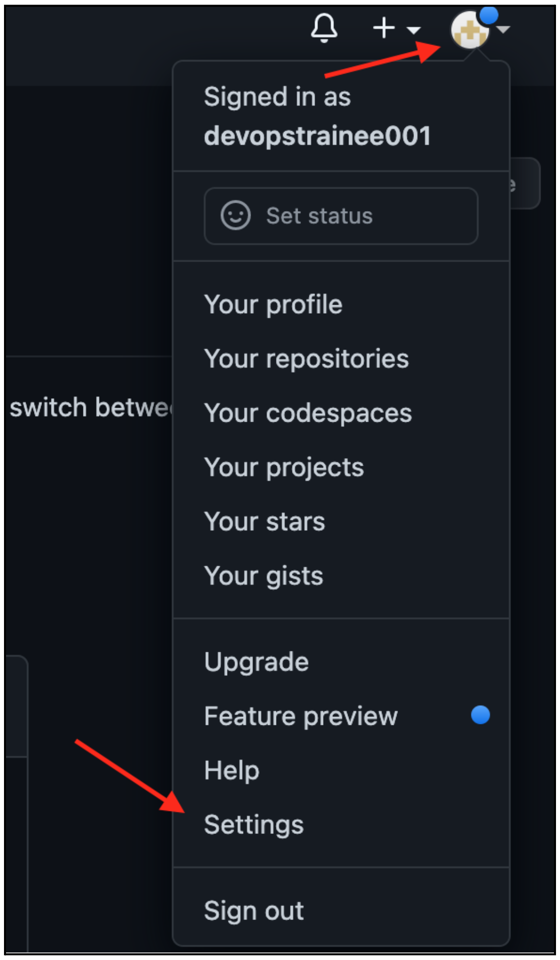
25. Copy the contents of the PUBLIC key using the following command:

    $ cat repo-key.pub, then copy the contents



```
[ubuntu@ubuntu:~/.ssh$ cat repo-key.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCpnx7L4szr7yKHiS7CBSVIHFBc8UUDy2QkyYAtamTLvee0ni4uNQL3U
hjVqo5Zeb7Vn0HHKYsoinScUT9NRMBJCZg8hK6A+zEo+yp2IEKDA+/rNSsdGmTTBMSVLUtzTo/aRipfy5MZf/Q60YqGn2
xfvCS1AkYzrS244YCRht08bHE2NPGd0fVgktv+tY4c7yoYZzCdUwfX92S45xJg/1KP+J9KJ09G9kOhaUJUeNJEHiE872R
qX/CLqm+7G5U9CuTzc8/LXAHcZ2S0Yqp1OsJ9K1YGrRhXxJFOPAygmU+CoQdNNF3qqQfumaL7WaVEpUg7SejZt3HGlGw+
PLqbbjS3tfFH8apQ79dnBLoI5HwPfadySsU4vJWmvfvGw+BqUtsSOIusAmwKhkKi4cwRVVjSRzynJUoMTW385FKkt22kf
DDba5TdkkwUyl2JuvEU/CSHx0RzWnPGOnFwuhU6ZVbnld0GChbdwZ5VPo5AbHXqTmganHIPZvY/Mzo2LNA/sSK599h5d5
uJQPb4HI08PHjb0DKwYFkmNoHNDJghlkPMDVKm8kYX8Hp+OCNJ/VoqQOlQZDJHtAy3tWgCLvqDQC3UjIRM7G/c+2uLRTD
jDcWO1HHI1DNpRRkLbujKmD6NBSzw1bioj+1w60mXnj1t9UITeIcaDUJYxjmOwC3mgoBRJw== ubuntu@ubuntu
ubuntu@ubuntu:~/.ssh$
```

26. 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)



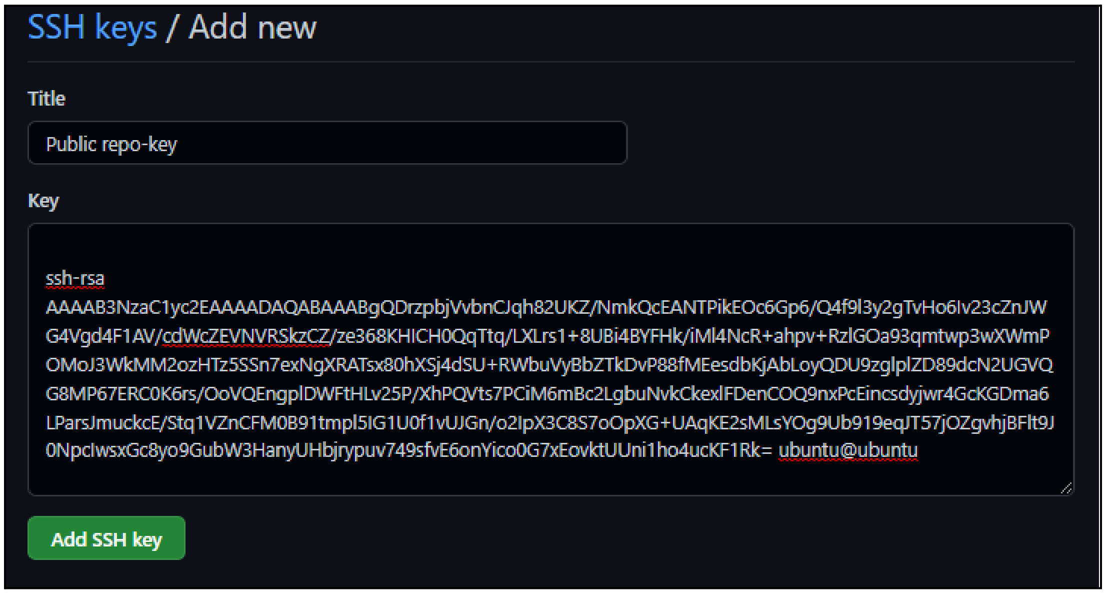
Note: This is done because once GitHub has the PUBLIC KEY, then our workstation can authenticate to it using the PRIVATE KEY

27. Go to **Settings -> SSH & GPG keys -> SSH Keys**

28. Click '**New SSH key**'

29. Title “**Public repo-key**”

30. Paste the entire contents of repo-key.pub into the ‘key’ textbox



SSH keys / Add new

Title

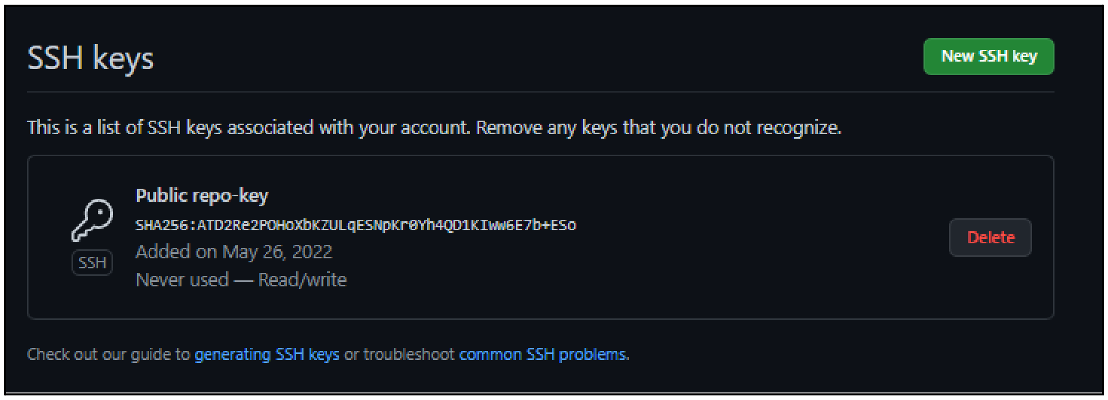
Public repo-key

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDrzpbjVvbnCJqh82UKZ/NmkQcEANTPikEOc6Gp6/Q4f9l3y2gTvHo6lv23cZnJW
G4Vgd4F1AV/cdWcZEVNVRSkzCZ/ze368KHICH0QqTtq/LXLrs1+8UBi4BYFHk/iMl4NcR+ahpv+RzlGOa93qmtwp3wXWmP
OMoJ3WkMM2ozHTz5SSn7exNgXRATsx80hXSj4dSU+RWbuVyBbZTkDvP88fMEesdbKjAbLoyQDU9zglplZD89dcN2UGVQ
G8MP67ERC0K6rs/OoVQEngplDWFtHLv25P/XhPQVts7PCiM6mBc2LgbuNvkCkexlFDenCOQ9nxPcEincsdyjwr4GcKGDma6
LParsJmuckcE/Stq1VZnCFM0B91tmpl5IG1U0f1vUJGn/o2IpX3C8S7oOpXG+UAqKE2sMLsYOg9Ub919eqJT57jOZgvhjBFlt9J
0NpclwsxGc8yo9GubW3HanyUHbjrypuv749sfvE6onYico0G7xEovktUUni1ho4ucKF1Rk= ubuntu@ubuntu
```

31. Click ‘**Add SSH key**’

32. Enter your GitHub password if requested



SSH keys

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

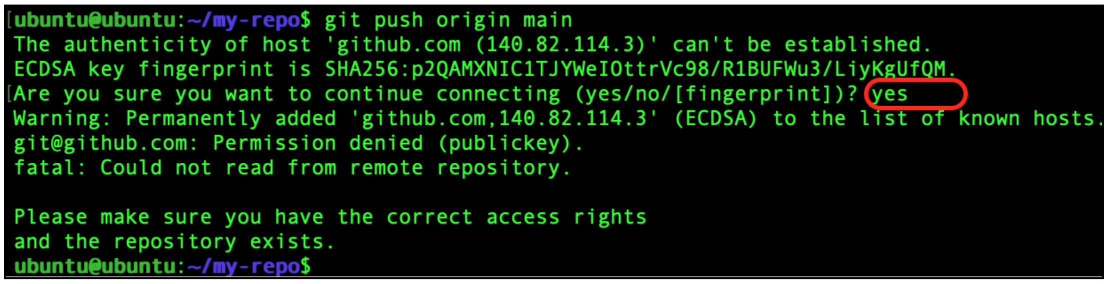
Public repo-key
`SHA256:ATD2Re2POHoXbKZULqESNpKr0Yh4QD1KIww6E7b+ESo`
Added on May 26, 2022
Never used — Read/write

Check out our guide to generating SSH keys or troubleshoot common SSH problems.

33. **Push to the repo again**

- Move back to your **my-repo** directory

- Push the repo with **$ git push origin main**
  Note: It will fail again



```
[ubuntu@ubuntu:~/my-repo$ git push origin main
The authenticity of host 'github.com (140.82.114.3)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
[Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com,140.82.114.3' (ECDSA) to the list of known hosts.
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
ubuntu@ubuntu:~/my-repo$
```

Note: 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 to GitHub

Add the PRIVATE key to your workstation keychain (C)

34. Open the SSH agent interface

    $ eval `ssh-agent`

    Note: use backticks (not single-quotes), typically at the top left of the keyboard

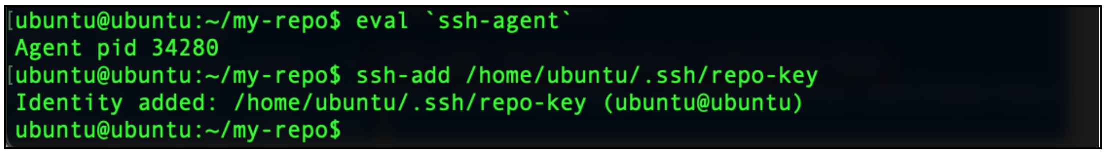
35. Add the key to the keychain with

    $ ssh-add **path_to_private_key**

    Example:

    $ ssh-add ~/.ssh/repo-key

    (it is OK to use the ~ shortcut in the ssh-add command)



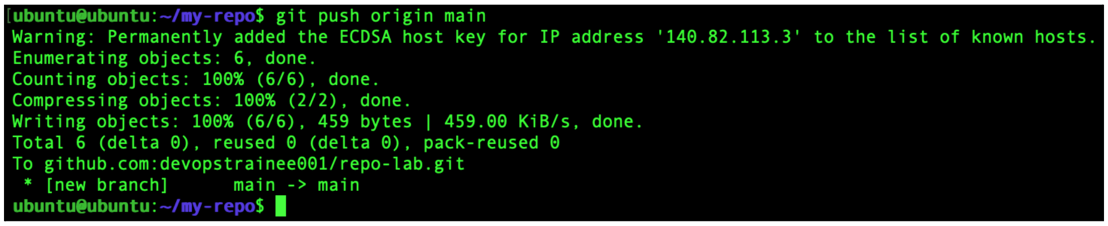
```
[ubuntu@ubuntu:~/my-repo$ eval `ssh-agent`
Agent pid 34280
[ubuntu@ubuntu:~/my-repo$ ssh-add /home/ubuntu/.ssh/repo-key
Identity added: /home/ubuntu/.ssh/repo-key (ubuntu@ubuntu)
ubuntu@ubuntu:~/my-repo$
```

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 (D)

36. Push to the GitHub repository again, which should succeed now that everything is in place:

    $ git push origin main

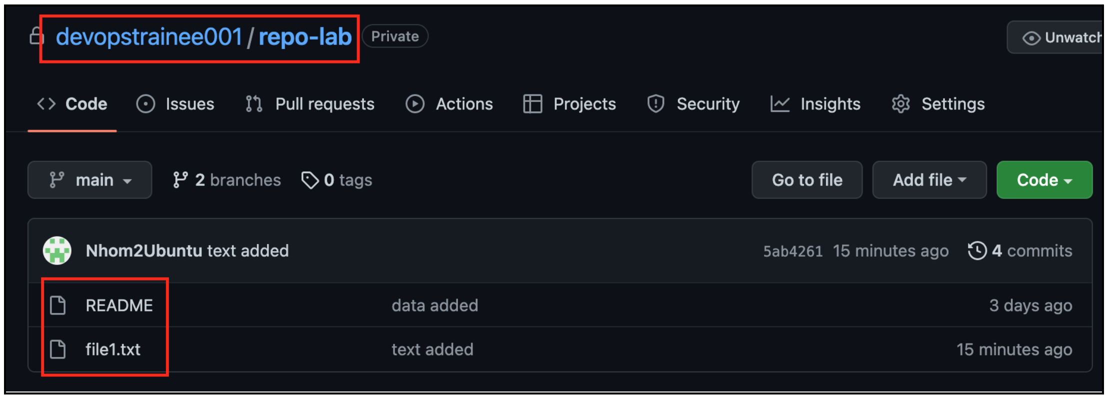
    Note: you may need to agree to continue the connection. Type 'yes' to continue



```
ubuntu@ubuntu:~/my-repo$ git push origin main
Warning: Permanently added the ECDSA host key for IP address '140.82.113.3' to the list of known hosts.
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 459 bytes | 459.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:devopstrainee001/repo-lab.git
 * [new branch]      main -> main
ubuntu@ubuntu:~/my-repo$ 
```

37. Refresh the GitHub web interface for your repository. You should see your files in the repo

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

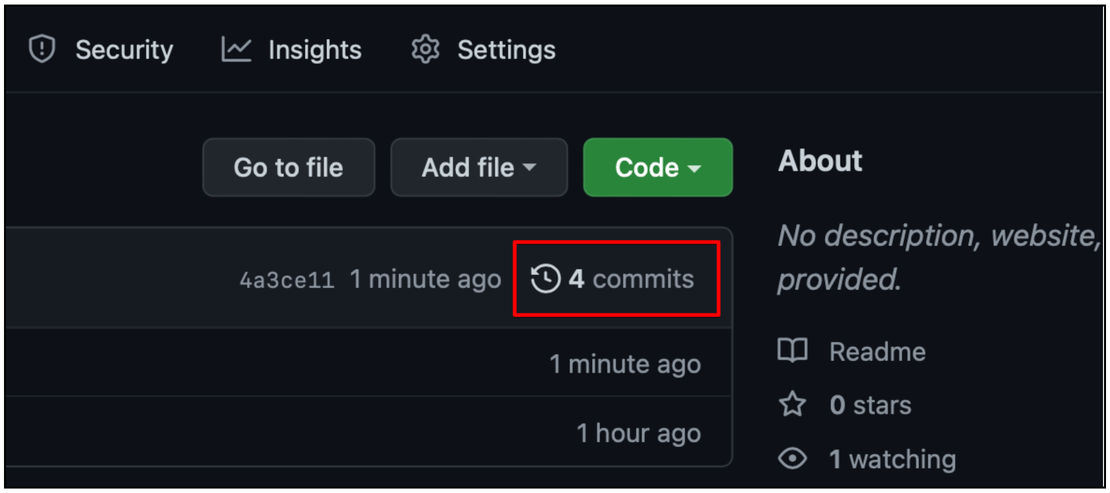


Nhom2Ubuntu text added 5ab4261 15 minutes ago 4 commits

```
README      data added     3 days ago
file1.txt   text added     15 minutes ago
```

Note: There is only 1 branch because we haven't pushed other branches in our workstation to the GitHub repository

39. Click '**commits**'. Notice that 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 repository



40. Verify the user showing who made the commit

A shortcut for git push

41. To make our lives easier, let's shortcut "git push origin main" to the infinitely easier command of "**git push**"

42. Open the **README** file in the main branch

43. Make a change to the content of the file. Save, add, and commit the file

44. Save the shortcut to the current remote repo and branch using the --set-upstream (or -u) flag. **Either option works**

    $ git push --set-upstream origin main
    or use the shortcut of -u
    $ git push -u origin main

45. Test that your shortcut is working

46. Make another change to the README file on the main branch

47. Save, add, and commit the file

48. Push the file to GitHub with:

    $ git push

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 main
```

Create a New Branch on GitHub

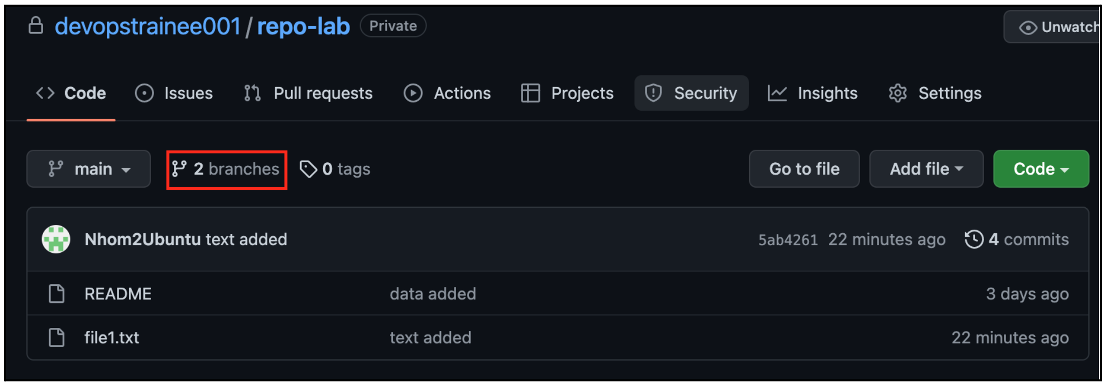
49. On your linux instance, checkout the testing branch. Refer to the lab '**Basic Git Commands**' if you don't remember how to do this

50. Create a new file with the name **"file2.txt"** with some content, add and commit this file in the **testing** branch

51. Push this to a testing branch on the remote repo:

    ```
    $ git push origin testing
    ```

    Note that the "$ git push" shortcut only points to the 'main' branch. Because you are pushing to the 'testing' branch you need to specify the branch here



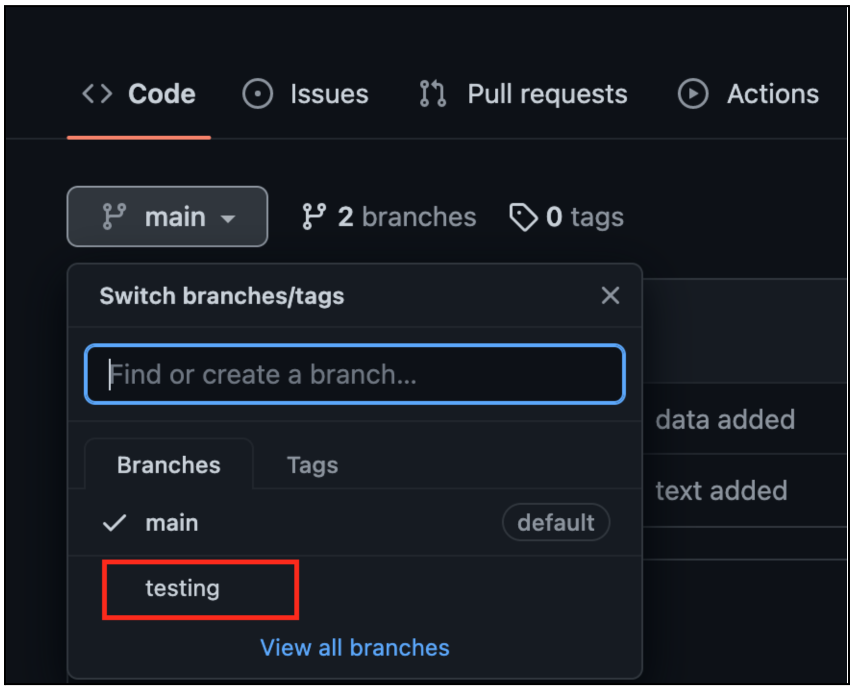
Nhom2Ubuntu text added 5ab4261 22 minutes ago 4 commits

README	data added	3 days ago
file1.txt	text added	22 minutes ago

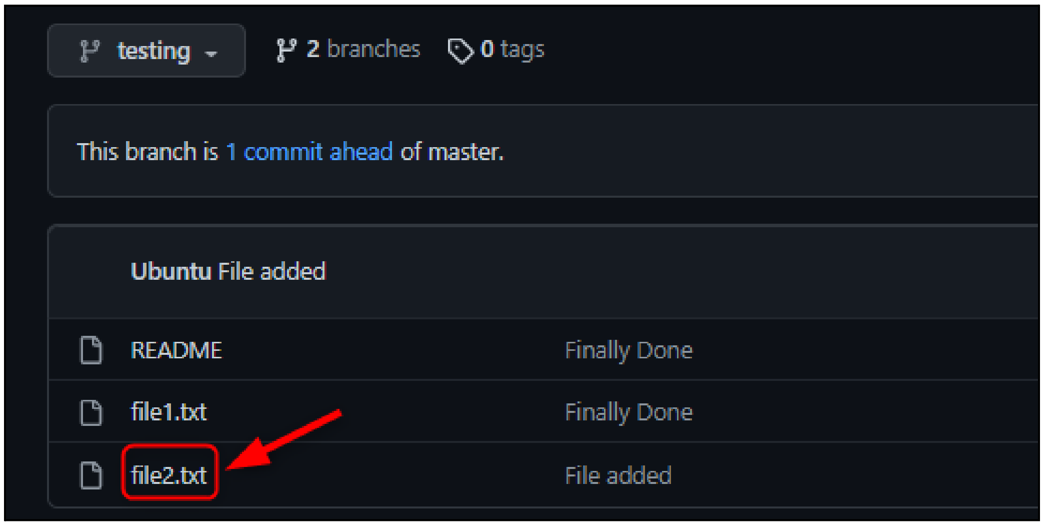
52. Refresh the GitHub interface to verify that the new **'testing'** branch exists on GitHub

    Note: You may need to click on the repository name at the top of the page to see the branches

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



54. Verify your new **file2.txt** is in the branch on GitHub repository



This branch is 1 commit ahead of master.

Ubuntu File added

README	Finally Done
file1.txt	Finally Done
file2.txt	File added

Merge Files On GitHub

55. In GitHub, go back to the main branch (using the dropdown you used in the last step)

56. Look at the files in the main branch. You should have **‘README’** and **‘file1.txt’**. You don't see **‘file2.txt’** because this file belongs to the 'testing' branch

57. Checkout to the **main** branch **on your workstation**

58. Pull the testing branch. This will copy the files from the **‘testing’** branch to the local **‘main’** branch

```
$ git pull origin testing
```

59. Look at your current branch and look at your local files. Note that **file2.txt** has been added to the list of files in the **main** branch on your workstation

60. Commit this to main. 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
```

61. Refresh **GitHub** to verify the **main** branch now contains **file2.txt**

Git End-To-End

62. Create and checkout a new ‘**staging’** branch

63. 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)

64. Create a file in this branch with the name **‘staging1.txt’** with some content

65. **Add**, **commit** and **push** this to the **‘staging**’ branch on GitHub

66. Checkout the **main** branch on your workstation

67. Next, merge ~~this file from~~ the 'staging' branch to the 'main' branch by pulling the **staging** branch down

68. Push the new resulting local main branch to the GitHub main branch

69. Verify the files you have created for the staging branch is now in the **main** branch on GitHub repository

Notify your instructor that you are done with the Lab

END OF LAB