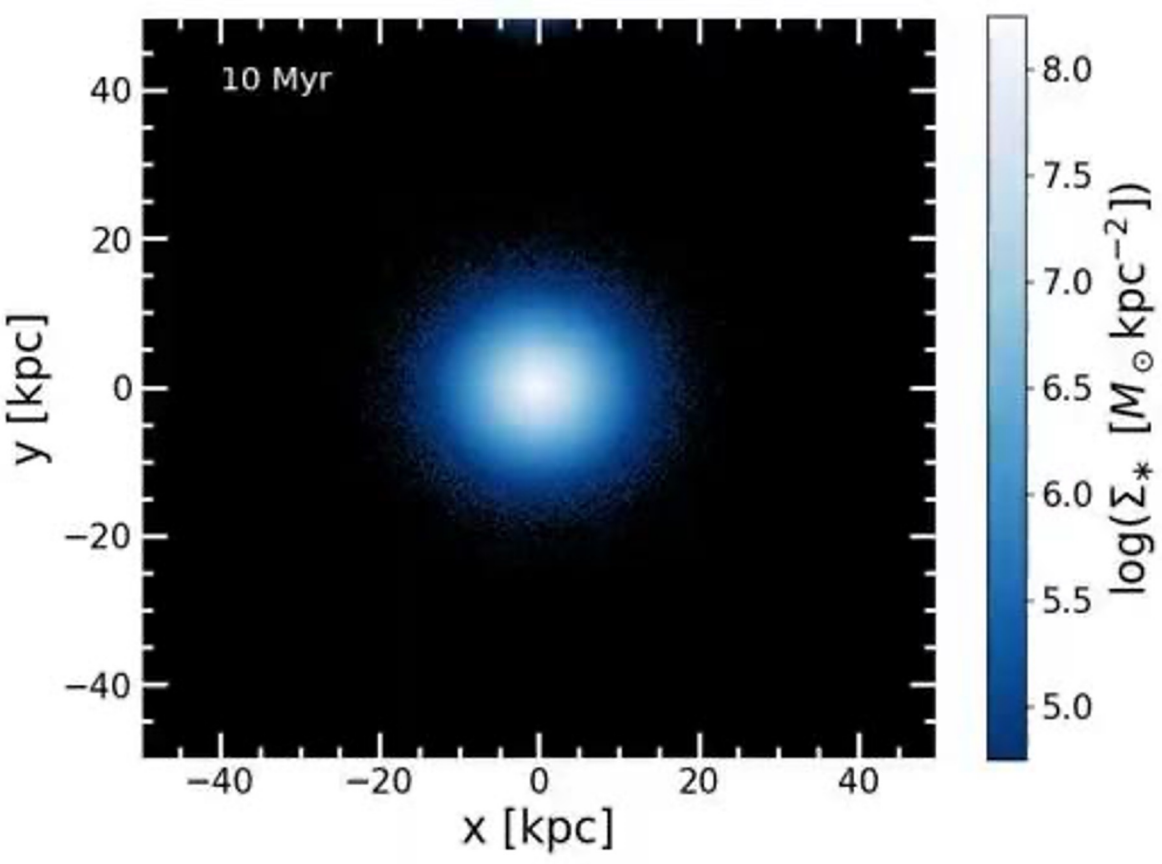
Version Control Using GitHub

Himansh Rathore
ASTR400B: Galaxies & Cosmology
Jan 23, 2025
University of Arizona

What is github ?

- Have you all used drive before ? What does it allow you to do ?
- Github is a code developer platform:
  - Create
  - Maintain
  - Manage
  - **Collaborate**
- Github is based on a version control system called git
  - **Keeps track of versions of files and the modifications therein**
  - Keeps track of version conflicts
  - Allows multiple people to collaborate on one project
- Efforts to make research and code development:
  - Accessible
  - Reproducible
  - Collaborative

What drives scientific research ?



Availability and Accessibility to sophisticated tools and large collaborative projects !

- Gadget-4: publicly available on gitlab (https://gitlab.mpcdf.mpg.de/vrs/gadget4)

Gadget4

- EXP: publicly available on github (https://github.com/EXP-code)

EXP, basis function software for galactic dynamics

The organization hosting EXP: basis function expansion tools for N-body galactic simulations and dynamical discovery

Other examples

A small collaborative project:

- himanshrathore / **pymargay** (https://github.com/himanshrathore/pymargay)

Store and maintain your own private codes:

- himanshrathore / **magclouds** (https://github.com/himanshrathore/magclouds)

Educational materials:

- krittikaiitb / **tutorials** (https://github.com/krittikaiitb/tutorials)

Personal Website:

- Me: (https://himanshrathore.github.io/index.html) - show off your professional skills !

Are there other version control languages ?

Are there other hosting services ?

Our goal for this lecture ...

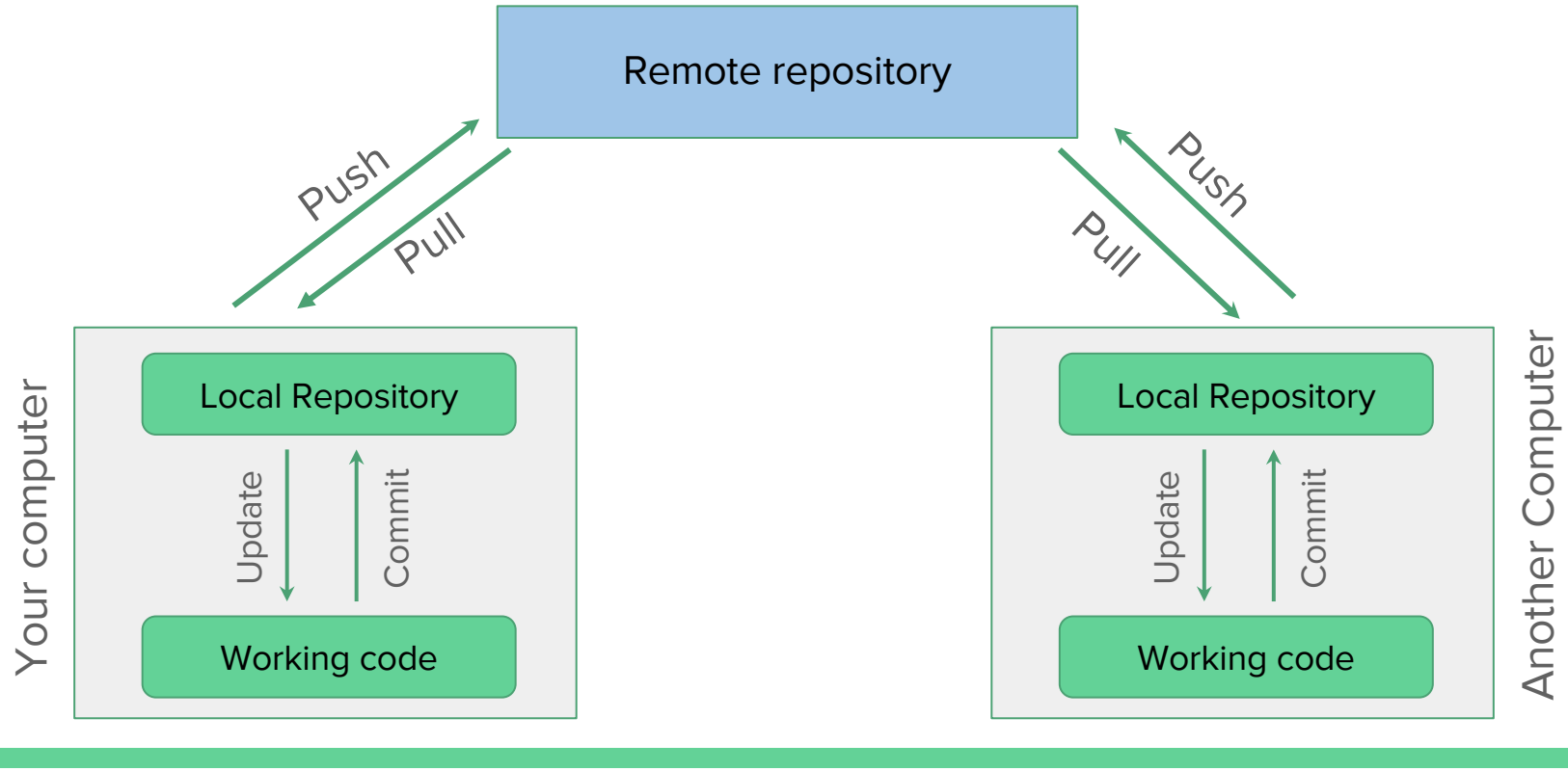
- Appreciate the importance of open-source software development enabled by technologies like git and github.
- Learn different components of github and the associated technical background
- Manage files using git
- Learn how to manage individual projects as well as collaborative projects with github

After this lecture, you will have the required background for accessing the course content and submitting assignments.

This lecture will be interactive. Please try to work along with me and help each other !

Some basic technical background behind git

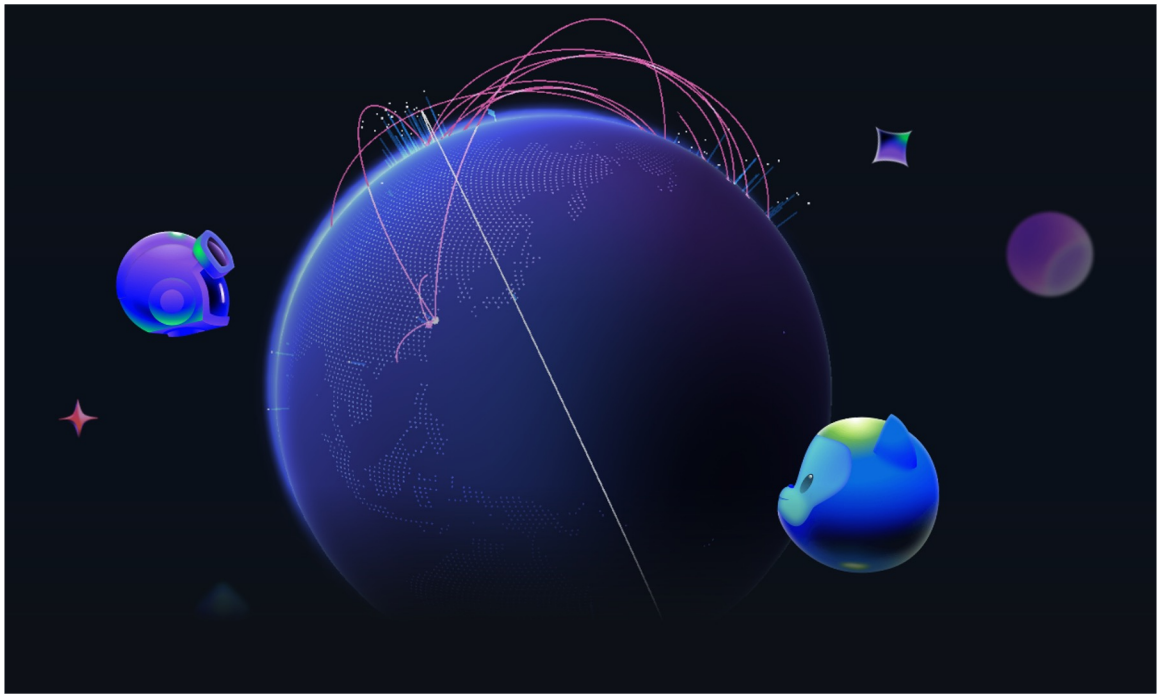
Repository - the directory that hosts your project



Slide adapted from Hayden Foote

Using git and github

- Github can be used through the web-interface (github.com) as well as the terminal.
- However, using the terminal offers several advantages:
  - More control
  - More functionality
  - More speed
  - More security
  - More cool !
  - Sometimes the only way !!
- In this lecture, we will perform most operations using the terminal.



Using github with the terminal

Basic idea

- Connect the local files in your computer to your remote repository
- So that you can sync files between your computer and the remote repository
- You can modify and create code on your computer, and then automatically sync it to the remote repository, while keeping track of versions
- Your collaborators can create or modify code on their computers and automatically sync it to the remote repository, while keeping track of versions
- This requires a communication and authentication protocol between the remote repository and your computer

Communication and Authentication Protocols

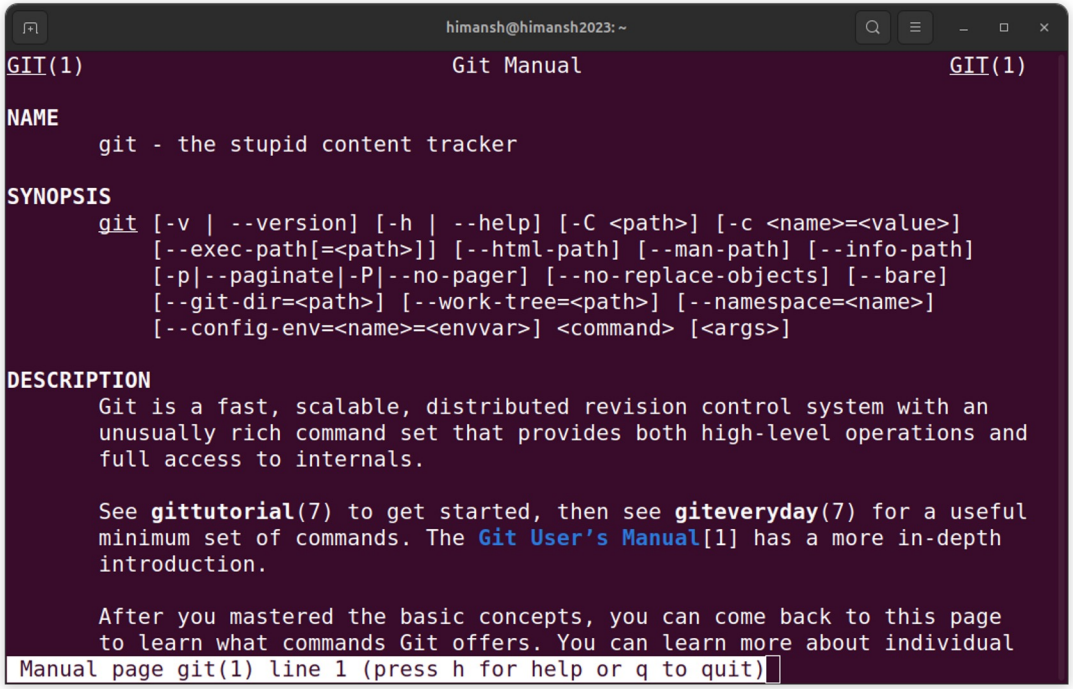
- Github offers two communication protocols:
  - SSH (Secure Shell)
  - HTTPS (Hypertext Transfer Protocol)
- SSH is more secure, however it needs more setup.
- We will use HTTPS in this lecture. You are free to use either.
- For authentication, there are two main options:
  - Personal access token (works like a password), see HW1 instructions
  - Public key encryption (for SSH)
- Public key encryption is more secure, but is harder to setup.
  - For this lecture, we will focus on the personal access token. You are free to use either.
- Talk to me during office hours if you would like to know more about SSH authentication and public key encryption !

Creating a personal access token

- The personal access token can be obtained from the web-interface. Steps are clearly outlined in:
  - https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens#creating-a-personal-access-token-classic
- There are two types of personal access tokens:
  - Classic
  - Fine grained (offers more functionality and security)
- We will create a classic token. Make sure you store the token in a secure location.
- Git will often ask your token. You can store your tokens in the git credential manager so that you do not have to type it everytime (https://docs.github.com/en/get-started/getting-started-with-git/caching-your-github-credentials-in-git#git-credential-manager)

Creating a github repository

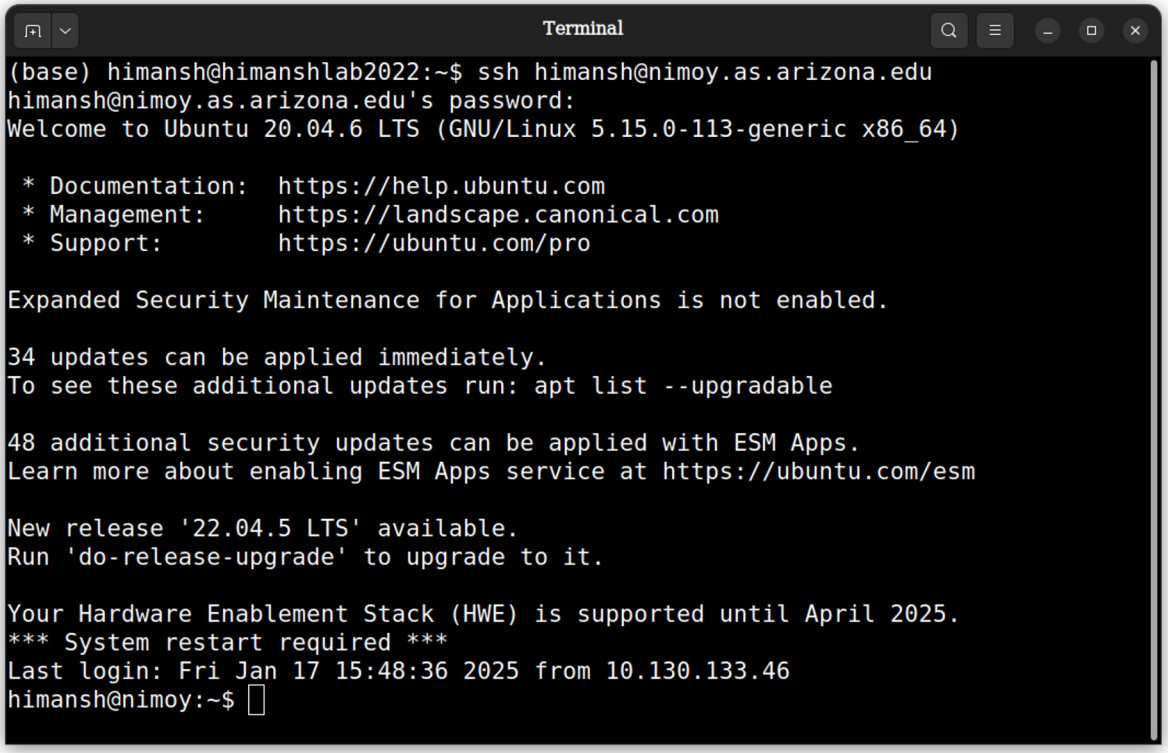
- Log in to github and go to “Dashboard”
- Select the “New” button
- Give a short repository name. I am giving himansh400B.
- Give a short description
- Choose repository access. Can change it later.
    - **If you have given private access to your HW repository, please give me access through my username “himanshrathore” if you have not already done so.**
- Initialize the repository with a README file
    - Give a longer description to your project
    - Can write a documentation/installation instructions if you have code
- License - for our purposes choose NONE
- Click on “Create repository” !



Writing git scripts

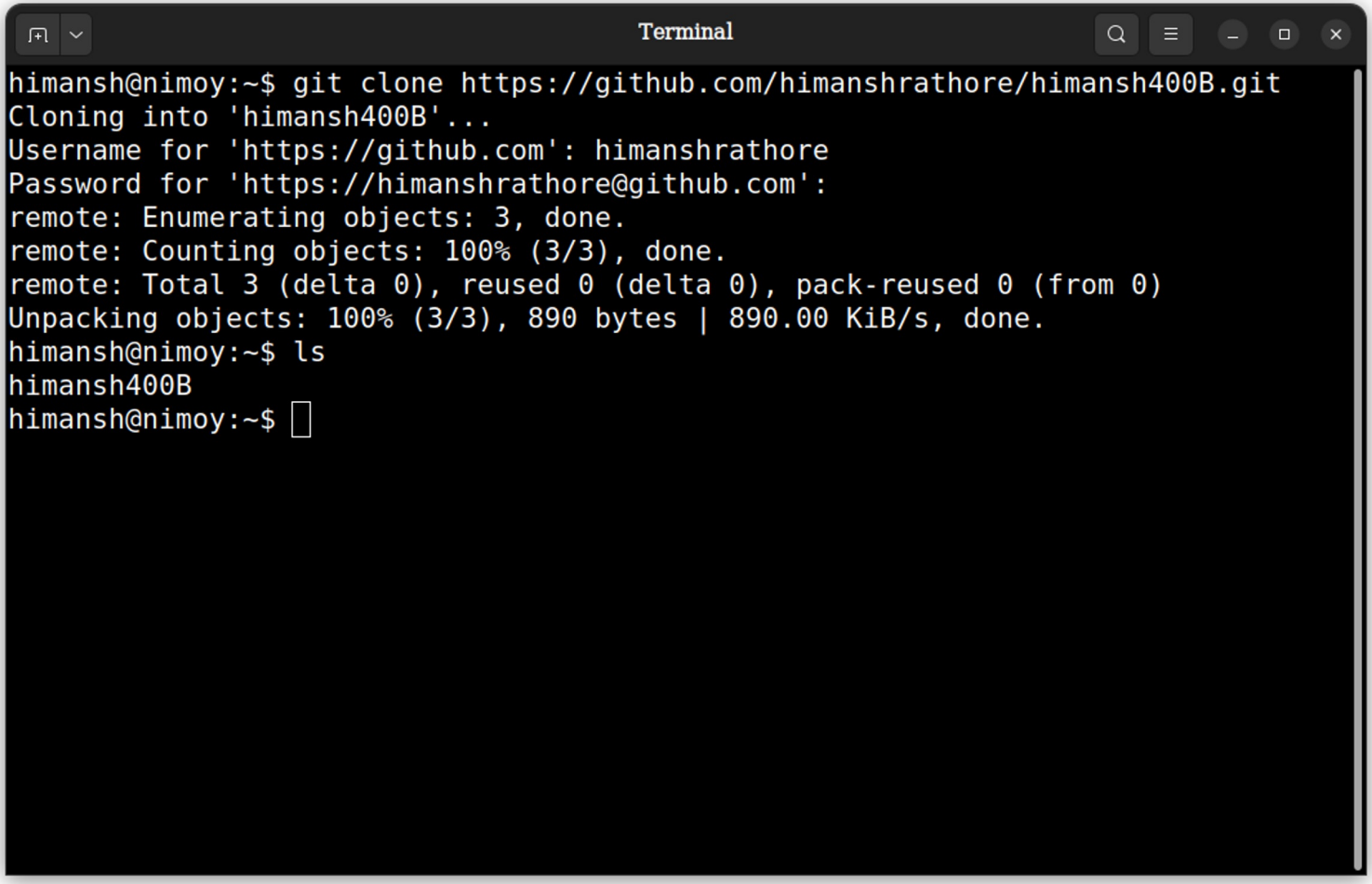
Access nimoy

- Alternatively, you can open a terminal in your computer.
- git is already installed in nimoy.
- Can check by typing git and pressing enter.



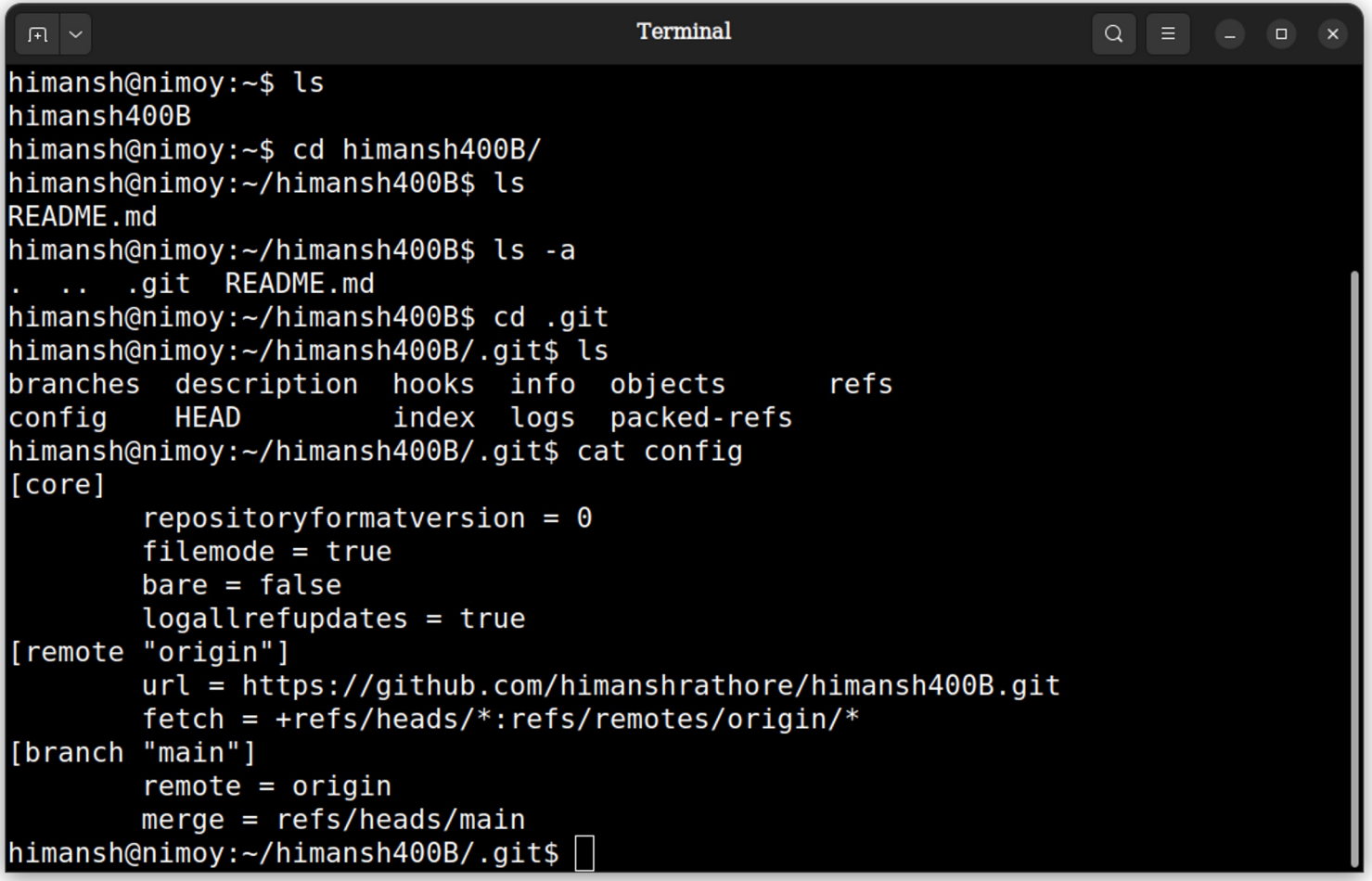
Cloning a repository using git

- Navigate to the home directory
  - $ cd
- Now, on the web interface, go to the repository you created
  - Click the green “Code” button and access the drop down list
  - Choose the HTTPS option
  - Copy the contents of the textbox
    - “https://github.com/himanshrathore/himansh400B.git”
- In the terminal, run the following:
  - $ git clone https://github.com/himanshrathore/himansh400B.git
- Enter your github username. When it asks for your password, use your personal access token.
- You should have a clone of your repository as a new directory. We will refer to this as the local repository. Check using:
  - $ ls
- This is how you will clone the class repository as well !


Terminal
himansh@nimoy:~$ git clone https://github.com/himanshrathore/himansh400B.git
Cloning into 'himansh400B'...
Username for 'https://github.com': himanshrathore
Password for 'https://himanshrathore@github.com':
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 890 bytes | 890.00 KiB/s, done.
himansh@nimoy:~$ ls
himansh400B
himansh@nimoy:~$

What is special about this cloned directory ?

- What is the difference between this cloned directory and any regular directory that already exists on the computer ?
- Access the directory
  - $ cd himansh400B
  - $ ls
- You will see the contents of your repository as it is
- But, now access the hidden files
  - $ ls -a
- What do you see ? Access the .git directory
  - $ cd .git
  - $ ls
- Thats the magic of github ! This directory is somehow connected to your repository on the web server. The connection is accessed using git !
  - $ cat config


Terminal
himansh@nimoy:~$ ls
himansh400B
himansh@nimoy:~$ cd himansh400B/
himansh@nimoy:~/himansh400B$ ls
README.md
himansh@nimoy:~/himansh400B$ ls -a
. .. .git README.md
himansh@nimoy:~/himansh400B$ cd .git
himansh@nimoy:~/himansh400B/.git$ ls
branches description hooks info objects   refs
config   HEAD   index logs packed-refs
himansh@nimoy:~/himansh400B/.git$ cat config
[core]
repositoryformatversion = 0
filemode = true
bare = false
logallrefupdates = true
[remote "origin"]
url = https://github.com/himanshrathore/himansh400B.git
fetch = +refs/heads/*:refs/remotes/origin/*
[branch "main"]
remote = origin
merge = refs/heads/main
himansh@nimoy:~/himansh400B/.git$

Lets make some edits to the files

- Go one level up in directory
  - $ cd ..
- Run the command git fetch. This fetches the version history of your repository from the web interface.
  - $ git fetch
- Run the command git status. This tells you whether your local repository is in sync or not with the remote repository. This is very important for avoiding merge conflicts - more on this later. **Make sure your local repository is in sync.**
  - $ git status

Lets make some edits to the files

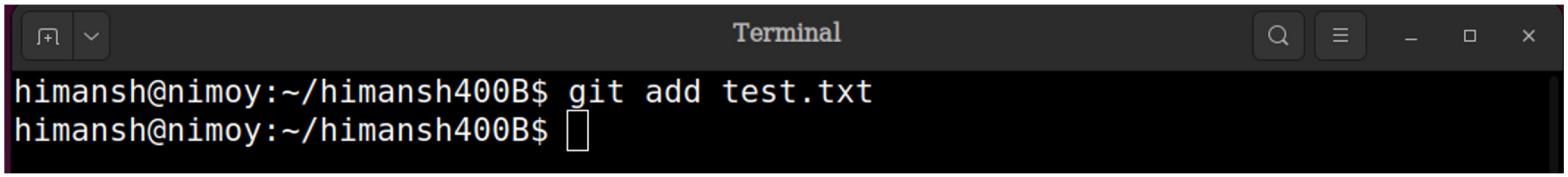
- Open test.txt file with a text editor. I like to use gnu nano.
  - https://www.nano-editor.org/
  - $ nano test.txt
- Other popular terminal based text editors:
  - emacs - https://www.gnu.org/software/emacs/
  - vi - https://www.geeksforgeeks.org/vi-editor-unix/
- Modify the file. Example - insert a line
  - “This is my first modification”
  - Press CTRL+O to write and then CTRL+X to quit
- Now, how do we sync this to the github remote repository ?
  - First, we need to tell git that we have made some modification to a file

Before that, let’s make sure git knows who you are

- Run the following:
  - $ git config --global user.name "<your username>"
  - $ git config --global user.email "<your email>"

Using git add and git commit

- Use git add to tell git you have made a modification
  - $ git add test.txt



- Note: if you modified multiple files, or created some additional files, you can use '.' to add all the files or all the modifications to git at once
  - $ git add .
- Lets commit our changes. For this, do:
  - $ git commit -m "first modification to test file"
  - "-m" is for a message - the commit message should be short and informative

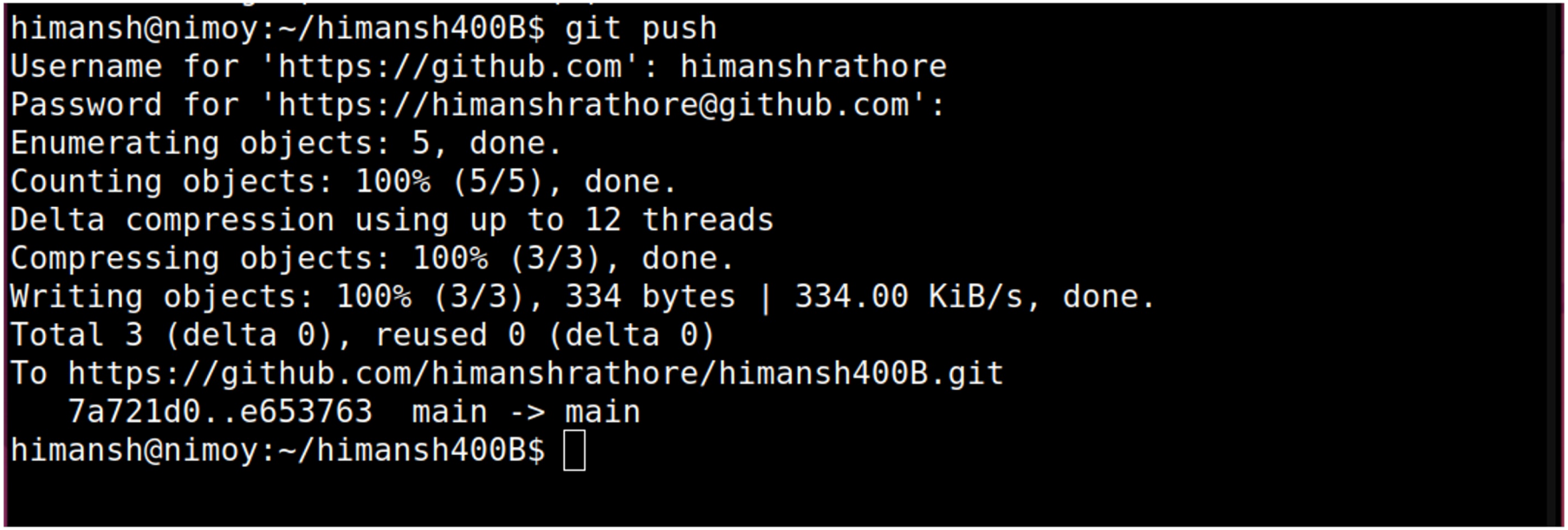
Using git push

Push the changes to your remote repository by.

$ git push

It will again ask for your username. Paste your personal access token when it prompts for the password.

Now go to the web interface and check the file (you may need to refresh the browser page)!

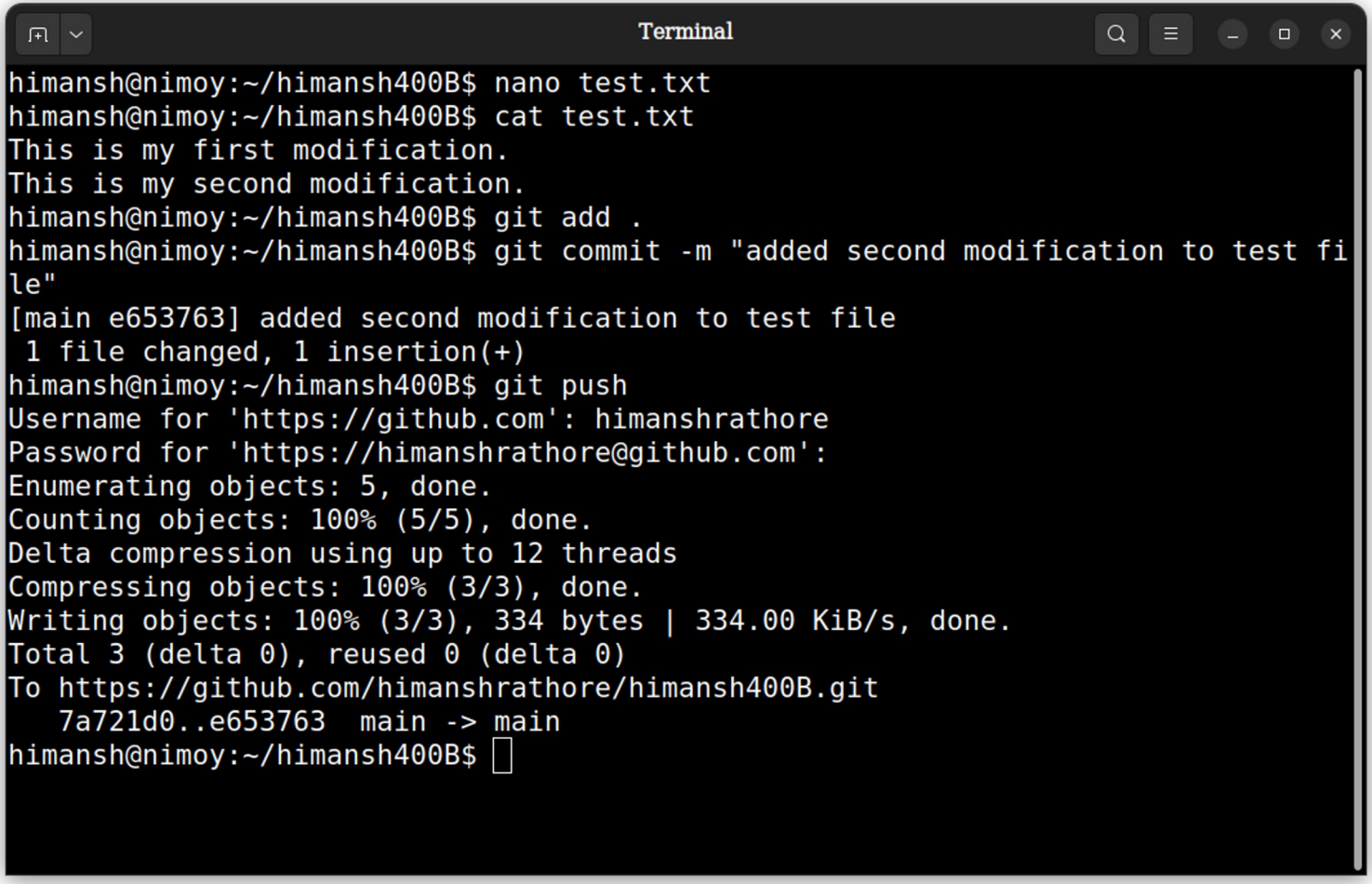

himansh@nimoy:~/himansh400B$ git push
Username for 'https://github.com': himanshrathore
Password for 'https://himanshrathore@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 334 bytes | 334.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/himanshrathore/himansh400B.git
7a721d0..e653763 main -> main
himansh@nimoy:~/himansh400B$

Uploading your HWs

- You can work on your HW independently of the github repository on your local computer. You can also choose to work on your personal computer or nimoy.
- When you are finished, copy your HW directory to your github repository. Make sure the name of your HW directory and the file/folder structure inside is according to what is asked.
- Then, just do the following:
  - $ git add .
  - $ git push
- Make sure to double check in the web-interface that the HW has been uploaded.
- Worst case - upload directly through the web-interface, using the “add file” button in your repository.
- Ask me during office hours if you face any issue !

Lets make another modification to the file

- Open the file again with a text editor and add another line.
- If you are using nano, use CTRL+O to write and CTRL+X to exit.
- Follow the same steps:
  - git add .
  - git commit -m “added second modification to test file”
  - git push
- Go to the web interface and refresh the page. You will see the modified file.
- What if you wanted to access the previous version of this file ?

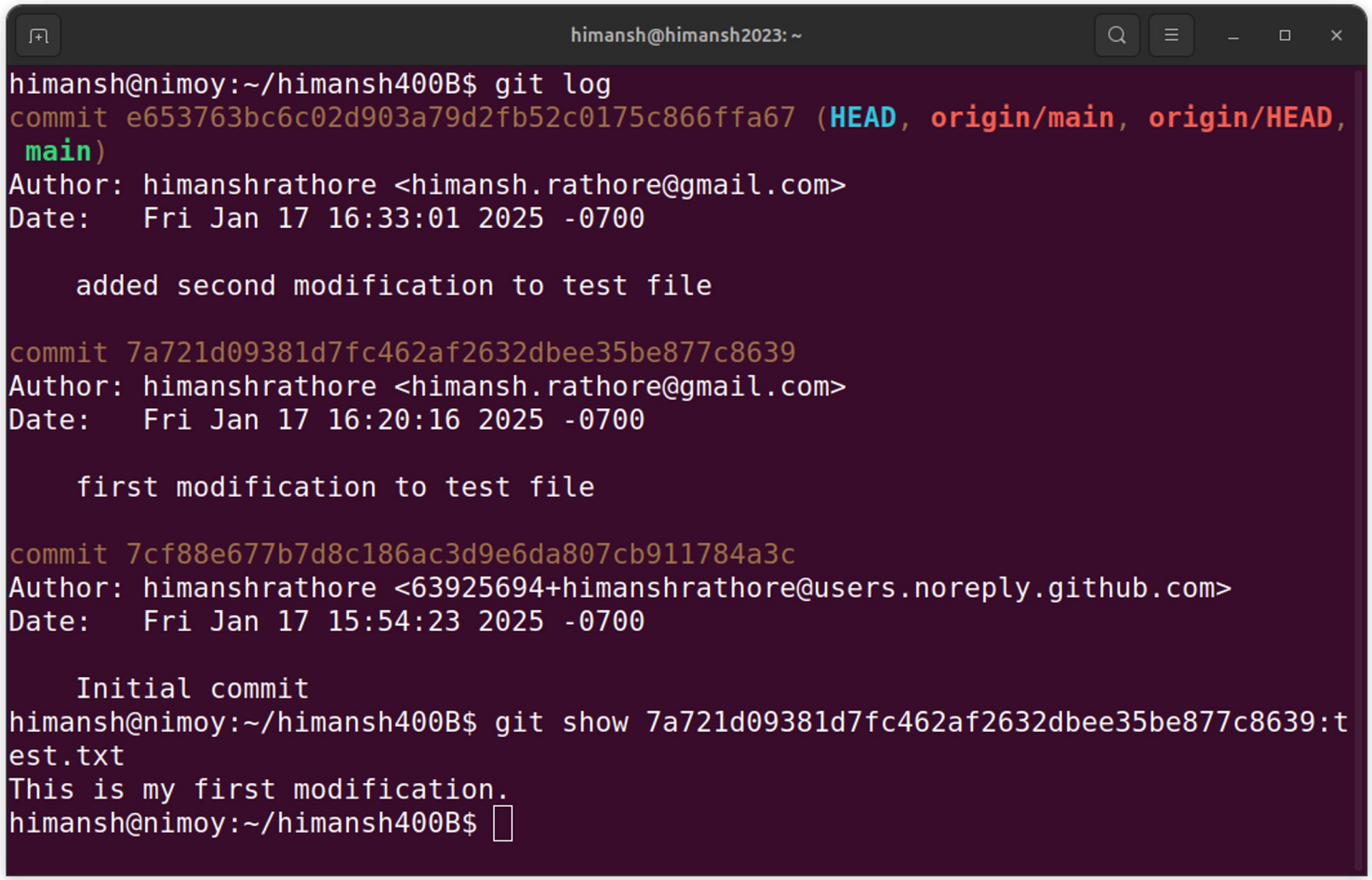

Terminal
himansh@nimoy:~/himansh400B$ nano test.txt
himansh@nimoy:~/himansh400B$ cat test.txt
This is my first modification.
This is my second modification.
himansh@nimoy:~/himansh400B$ git add .
himansh@nimoy:~/himansh400B$ git commit -m "added second modification to test fi
le"
[main e653763] added second modification to test file
1 file changed, 1 insertion(+)
himansh@nimoy:~/himansh400B$ git push
Username for 'https://github.com': himanshrathore
Password for 'https://himanshrathore@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 334 bytes | 334.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/himanshrathore/himansh400B.git
7a721d0..e653763 main -> main
himansh@nimoy:~/himansh400B$

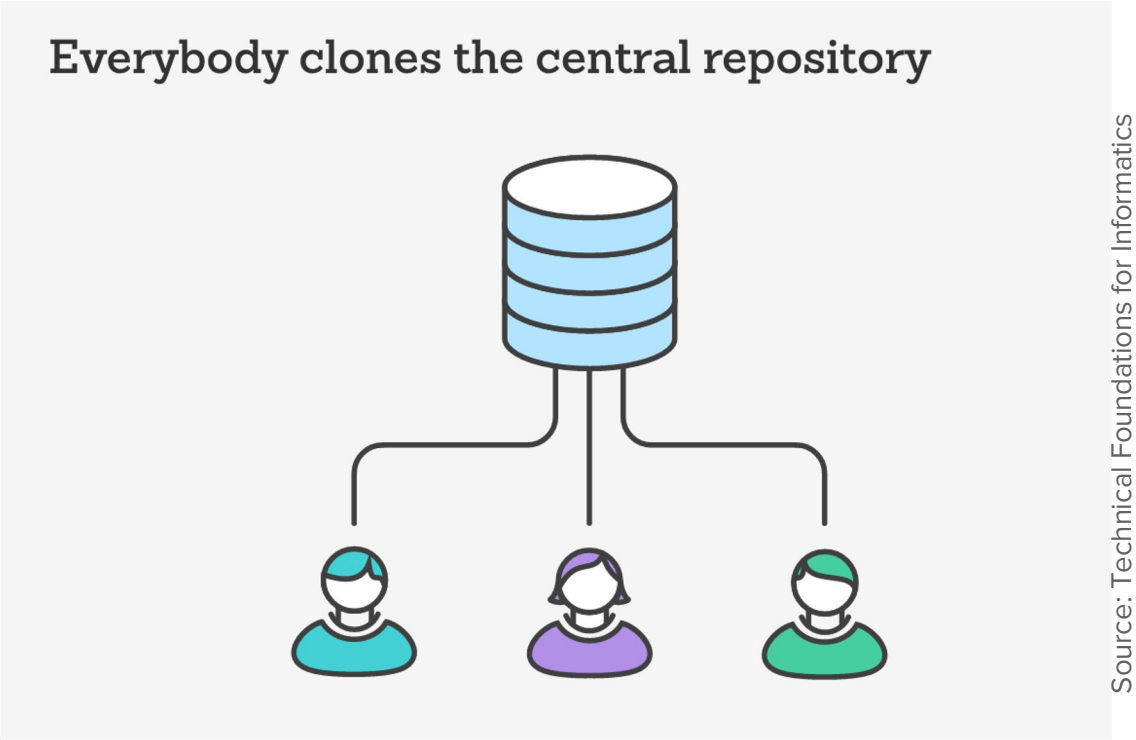
How to access the previous version of a repository ?

- Suppose for some reason you realized you need a previous version of a file or perhaps the entire repository
  - Its going to happen way too often in research !
- In the main page of the repository, click on the file
- Click on the “History” button at the right hand side
- You will see the history of this file with the corresponding commit messages along with date/time
  - Thats why the commit messages are important !
- In each history tab, on the left you have a variety of options
  - You can view the file at a specific instance of the past
  - You can view the entire repository at that instance
- This is a very powerful feature of github and version control !

Using the terminal ...

- Use the git log command:
  - $ git log
- This will give you the commit history, along with an identifier for each commit. The identifier looks like a complex sequence of letters and numbers.
- You can view a file at a previous commit using git show.
  - $ git show <commit_identifier>:<path_to_file_you_want_to_view>
- You can also explore the entire repository at a specific commit. See:
  - https://betterstack.com/community/questions/how-can-i-view-old-version-of-file-with-git/


himansh@himansh2023: ~
himansh@nimoy:~/himansh400B$ git log
commit e653763bc6c02d903a79d2fb52c0175c866ffa67 (HEAD, origin/main, origin/HEAD, main)
Author: himanshrathore <himansh.rathore@gmail.com>
Date: Fri Jan 17 16:33:01 2025 -0700
added second modification to test file
commit 7a721d09381d7fc462af2632dbee35be877c8639
Author: himanshrathore <himansh.rathore@gmail.com>
Date: Fri Jan 17 16:20:16 2025 -0700
first modification to test file
commit 7cf88e677b7d8c186ac3d9e6da807cb911784a3c
Author: himanshrathore <63925694+himanshrathore@users.noreply.github.com>
Date: Fri Jan 17 15:54:23 2025 -0700
Initial commit
himansh@nimoy:~/himansh400B$ git show 7a721d09381d7fc462af2632dbee35be877c8639:test.txt
This is my first modification.
himansh@nimoy:~/himansh400B$



Source: Technical Foundations for Informatics

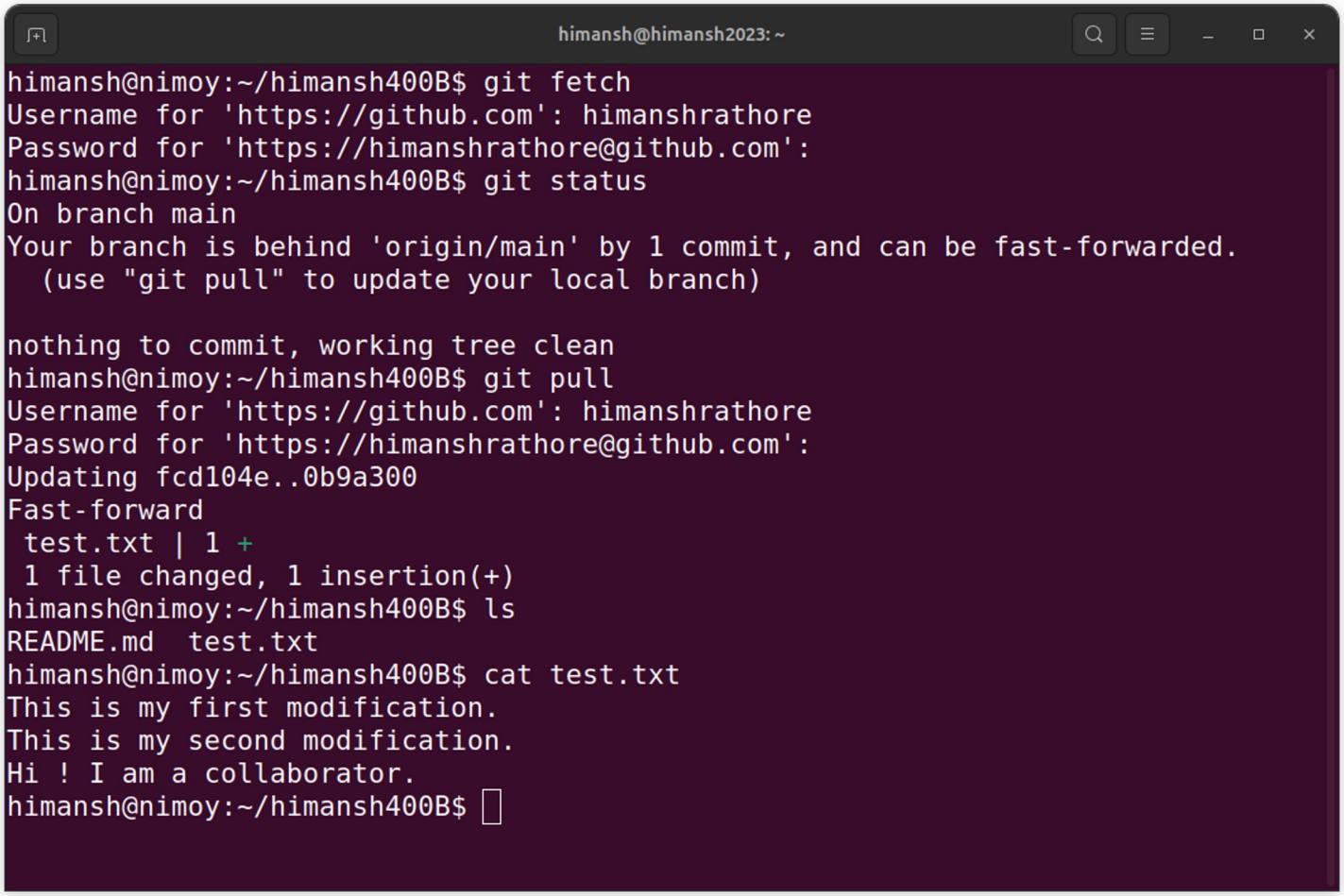
Collaboration with github

Mock collaboration setting

- Suppose you have a collaborator who makes some modification to your files.
  - Go to the web interface and edit the text file you just created. Important - edit it from the web interface.
  - Click on the file and then click on the pencil icon on the right.
  - Insert another line, e.g. “Hi ! I am a collaborator”.
  - Click on the green “Commit changes” button. Add a commit message like “collaborator adds a new line”.
- Note that in reality, the “collaborator” will have their own clone of the repository on their computer. They will make modifications there and push them to the remote repository.
- Now, your remote repository has been updated by the “collaborator”. How do you access those modifications onto your local computer ?

Using git fetch, git status and git pull

- First, run git fetch. It will fetch the latest commit history from the remote repository.
  - $ git fetch
- Now, run git status. It will tell you if the version of the repository on your computer is behind, ahead or in-sync with the remote repository.
  - $ git status
  - In this case, it tells us that we are behind. This is true, since the “collaborator” has just made a modification that we do not have.
- Use git pull to update the repository in your local computer.
  - $ git pull
- Check the file !
  - $ cat test.txt
- This is how you will update the class repository !


himansh@himansh2023: ~
himansh@nimoy:~/himansh400B$ git fetch
Username for 'https://github.com': himanshrathore
Password for 'https://himanshrathore@github.com':
himansh@nimoy:~/himansh400B$ git status
On branch main
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
(use "git pull" to update your local branch)
nothing to commit, working tree clean
himansh@nimoy:~/himansh400B$ git pull
Username for 'https://github.com': himanshrathore
Password for 'https://himanshrathore@github.com':
Updating fcd104e..0b9a300
Fast-forward
test.txt | 1 +
1 file changed, 1 insertion(+)
himansh@nimoy:~/himansh400B$ ls
README.md test.txt
himansh@nimoy:~/himansh400B$ cat test.txt
This is my first modification.
This is my second modification.
Hi ! I am a collaborator.
himansh@nimoy:~/himansh400B$

Merge conflicts

A merge conflict commonly occurs in the following situation:

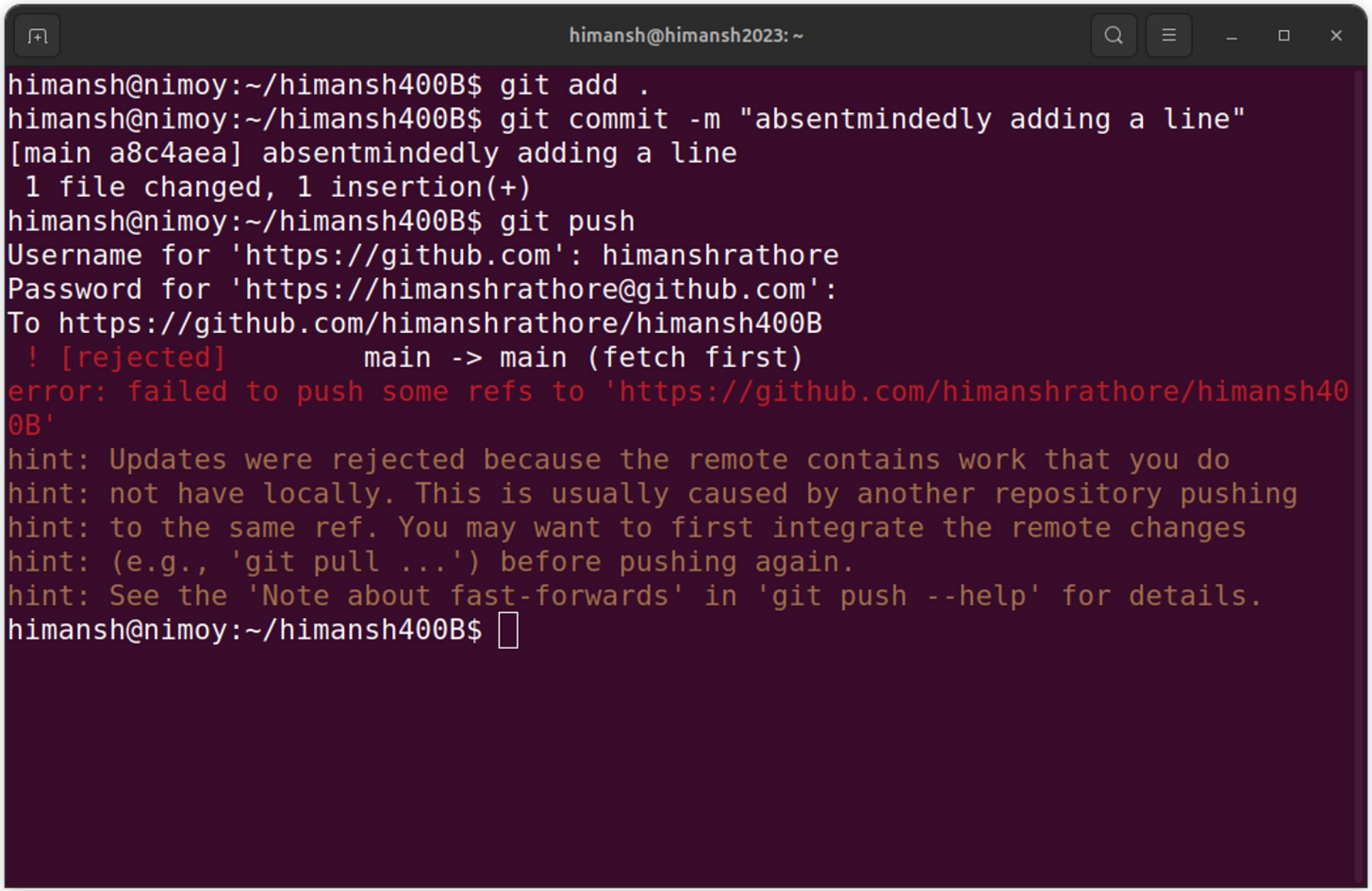
- Your local repository is behind the remote repository
- You forget to update your local repository using git pull
- Then, you make changes to your local repository
- And then, you try to push those changes to the remote repository

Lets try to deliberately perform a merge conflict. On the web interface, suppose the “collaborator” adds another line to the test.txt file. E.g. “Hi again, I am adding another line”. Do this and commit.

Now, without updating your local repository, make some changes to the test.txt file. Like “Absentmindedly added a line”.

Now add and commit, and try to push the changes.

$ git add .
$ git commit -m “absentmindedly adding a line”
$ git push

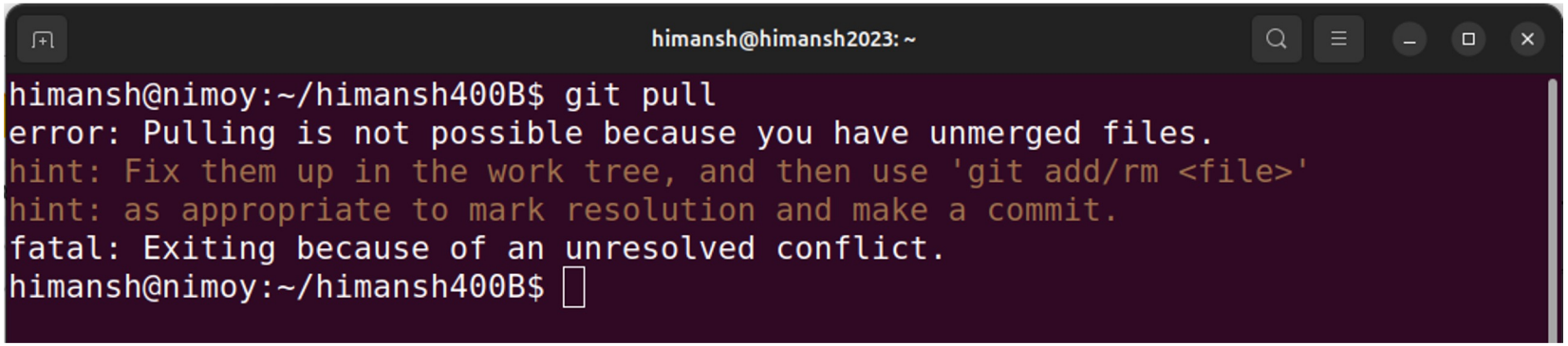

himansh@himansh2023: ~
himansh@nimoy:~/himansh400B$ git add .
himansh@nimoy:~/himansh400B$ git commit -m "absentmindedly adding a line"
[main a8c4aea] absentmindedly adding a line
1 file changed, 1 insertion(+)
himansh@nimoy:~/himansh400B$ git push
Username for 'https://github.com': himanshrathore
Password for 'https://himanshrathore@github.com':
To https://github.com/himanshrathore/himansh400B
! [rejected] main -> main (fetch first)
error: failed to push some refs to 'https://github.com/himanshrathore/himansh400B'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
himansh@nimoy:~/himansh400B$

Merge conflicts

Your attempt will be rejected. At this point, if you try to update your local repository using git pull:

$ git pull

Then, it will give an error message:



Resolving merge conflicts

- Resolving merge conflicts is not trivial and requires manual intervention, particularly if it is a large project.
  - Care should be taken that they do not happen.
- It is recommended to check the status of your local repository using the “git status” command before you start modifying it.
- Some ways in which you can manually intervene to resolve a merge conflict:
  - https://www.atlassian.com/git/tutorials/using-branches/merge-conflicts
- For now, we will just undo the change we made in the remote repository using git reset.
- If you made a lot of modifications absentmindedly, you can save a copy of the modifications somewhere. You can also use git stash (see the above link).

Using git reset

- Run git fetch and then access the git log
- Take note of the last commit that you made before the “collaborator”. This was the state of your local repository before you absentmindedly modified it.
- Copy the commit identifier and run:
  - $ git reset --hard <commit_identifier>
- This will bring your repository to the stage before you absentmindedly modified it.
- Now, git pull to update the repository.
  - $ git pull

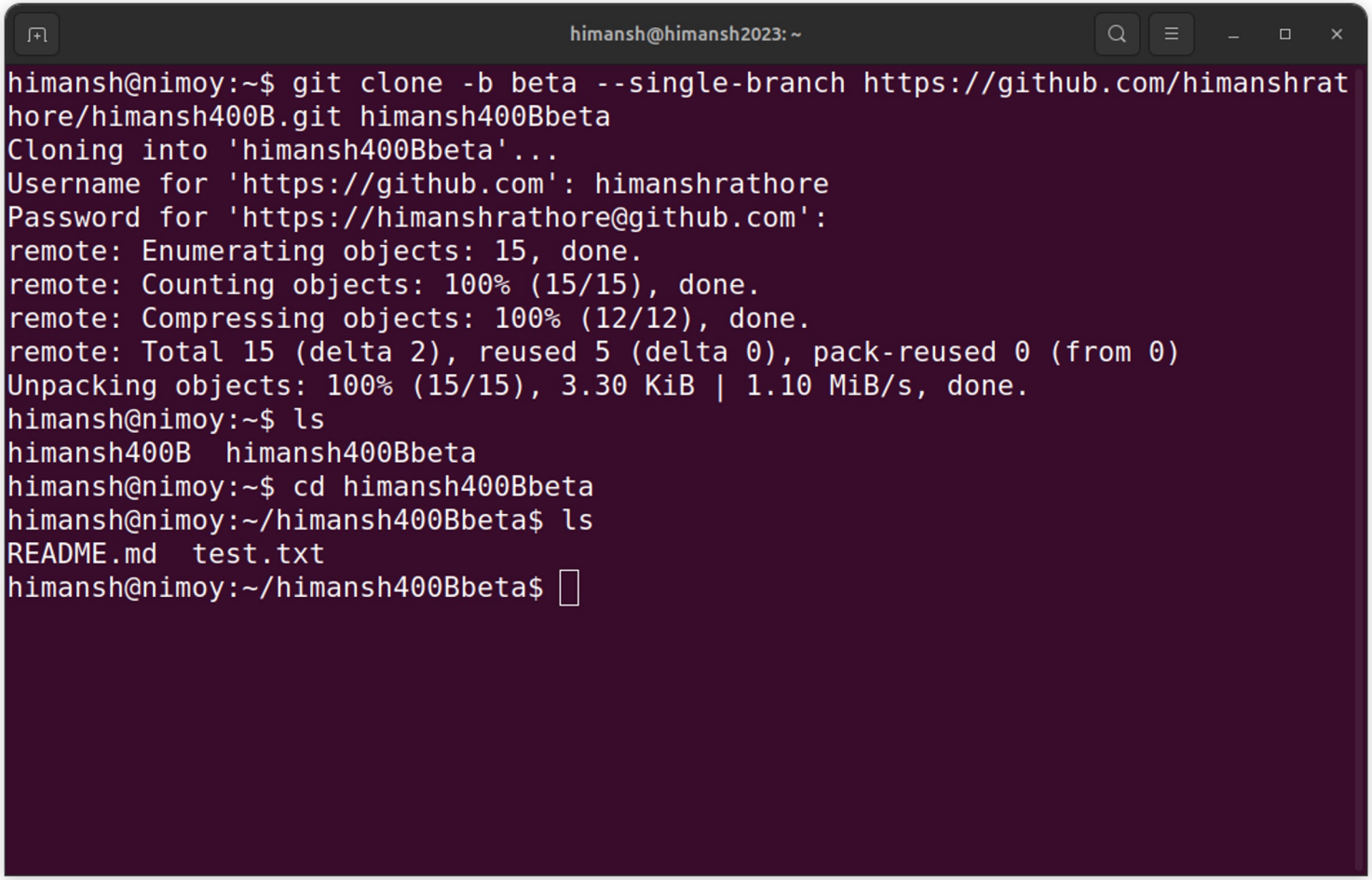
Using branches in github

Branches is an effective way to collaborate and also helps avoid merge conflicts. Branches are like a repository within a repository.

- Go to the “Branch” button on the repository webpage.
- Click on the “New Branch” button and create a new branch. Give it a name, e.g. beta.
- Specify the source of the branch. Everything will be copied from the source branch at this instance. Choose the source to be the main branch.
- Now, you can work on beta independent of the main branch.
    - Github also allows you to merge your branches.

Cloning a specific branch in github

- $ git clone -b <branch_name> --single-branch <repo_url> <directory_name>
  - By default, github will assigned the repository name to all cloned branches.
  - Since you already have your local repository with that name, it can cause a naming conflict.
  - So, give a different name to the cloned branch, like “himansh400Bbeta”. This will not affect the actual name of your branch in github.
- Now,
  - $ ls
  - You will see a new branch. Go into it and you will see your files and folders within that branch.
  - Now, you can work on this branch and perform most of the git operations independently of your main branch.


himansh@himansh2023: ~
himansh@nimoy:~$ git clone -b beta --single-branch https://github.com/himanshrat
hore/himansh400B.git himansh400Bbeta
Cloning into 'himansh400Bbeta'...
Username for 'https://github.com': himanshrathore
Password for 'https://himanshrathore@github.com':
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 15 (delta 2), reused 5 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (15/15), 3.30 KiB | 1.10 MiB/s, done.
himansh@nimoy:~$ ls
himansh400B himansh400Bbeta
himansh@nimoy:~$ cd himansh400Bbeta
himansh@nimoy:~/himansh400Bbeta$ ls
README.md test.txt
himansh@nimoy:~/himansh400Bbeta$

Additional resources and references

- GitHub docs - official resource for github documentation
  - https://docs.github.com/en
- My repository (if you want to explore):
  - https://github.com/himanshrathore/
- A nice reference to the command line and shell programming:
  - https://www.tutorialspoint.com/unix/index.htm
- More about branches:
  - https://www.w3schools.com/git/git_branch.asp?remote=github