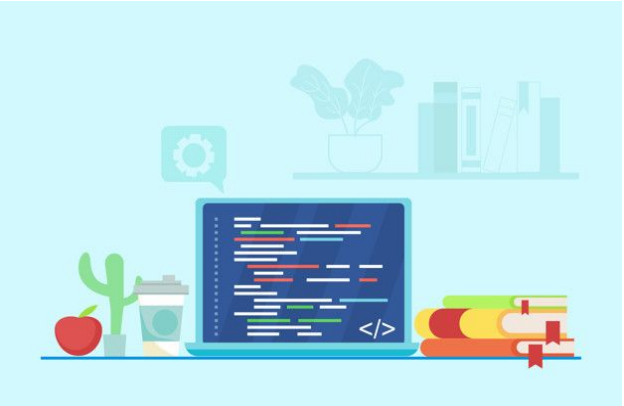
Git and making open source contributions



What is our GOAL for this MODULE?

We learned to practically use advanced features of git versioning tools - staging, branching, merging, comparing, resetting etc. - through writing a story. We also learned to contribute to open source projects on github using git.

What did we ACHIEVE in the class TODAY?

- Practically implemented advanced git commands on a git repository.
- Modified certain features of Monkey-Chunky App.
- Forked the Monkey Chunky Repository and made an open-source contribution to the app.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- **Commit:** Commit adds the work to the HISTORY of the works you are doing. You can move back and forward in your commits.
- **Open Source Projects:** These are projects initiated by a developer or a team of developers and made open so that anyone can contribute to its development and maintenance. Such projects are made by the collaborative efforts of several people.
- **3 Stages of the work area divided by git:**
    - **Working directory:** Where we are creating and making edits to our files.
    - **Staging area**: When the files are ready to be committed, we have to stage them first.
    - **Local GIt Repository:** After the files are committed, they become a part of commit history in a local repository.

What did we REVISE today?

Git commands:

- **git init:** Initialize your repository with git. Git starts tracking changes in your files.
- **git add [file]**: Add a file/file change from your working directory to staging area.
- **git commit-m:** Commit a file to commit history.

- **git remote add [remotename] [remoteURLAddress]:** Connect your local repository to a remote repository (example GitHub).
- **git push:** Push the current local repository to remote.
- **git clone [url]:** Clone the remote repository into a local repository.

How did we DO the activities?

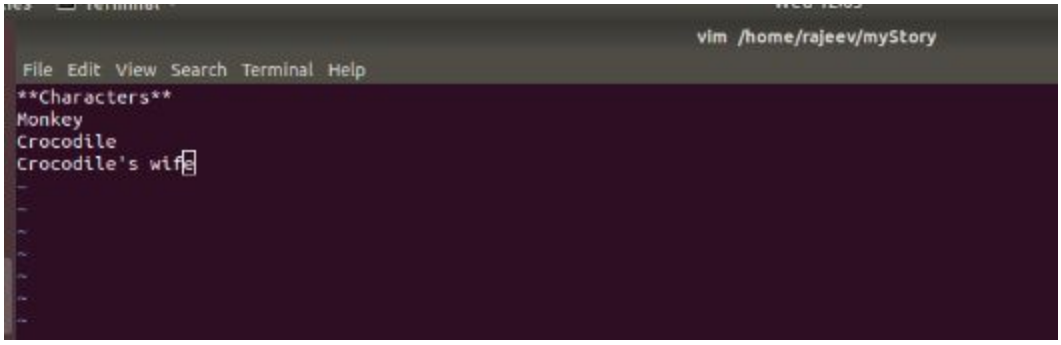
Let's try to write a story using git and see how git helps in keeping control of all the versions of our story (similar to different versions of code).

1. Open a terminal (git bash). Let's make a new directory called "MyStory" using 'mkdir' command and change into the directory (using cd command).

2. Navigate to the said folder and create two empty files called "Characters" and "Story".
    - You can open these files in any editor and start writing.

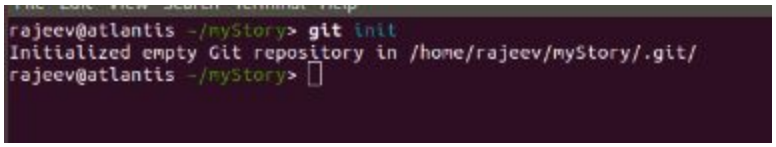
3. Initialize git using 'git init' so that git can start tracking changes in our files.

    - Characters File:



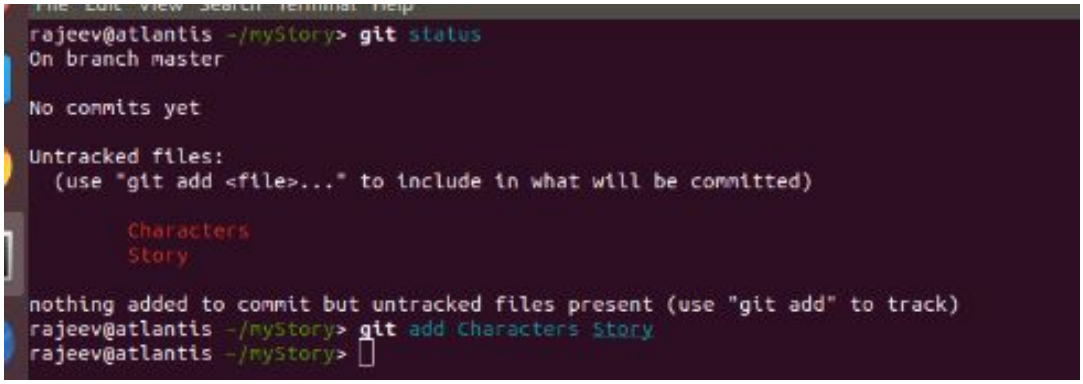
    - Story File:

- Git Init:

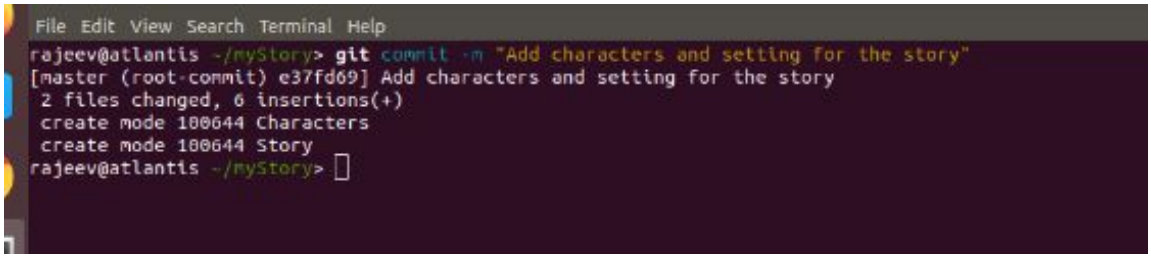


4. The changes we are making in the files are in the working directory. We can make as many changes as we want here.

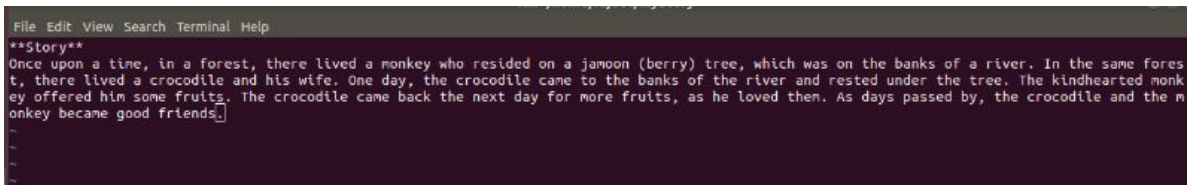
5. If we want to commit these changes to the history in our local repo, we need to stage these changes.
    - We can simply type git status to check the status of our git working directory.
    - You can see that there are two files which need to be added to the staging area.
    - We can do that using git add [filename1] [filename2]

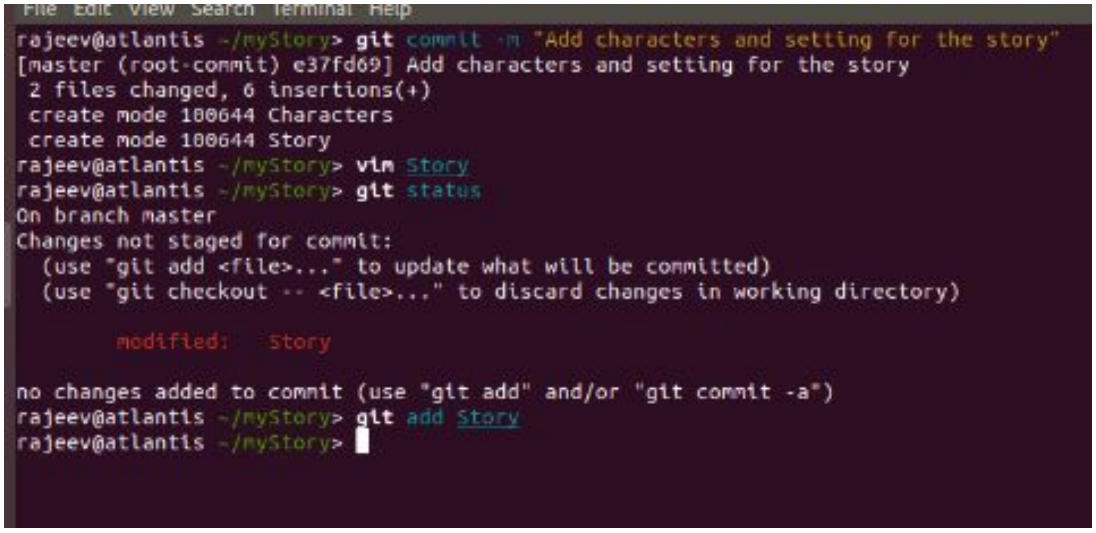


6. We can continue to make more changes in the file.
    - For every change in a file, we will have to change the stages again. Otherwise git will have the snapshot of the older staged file in its memory.
    - At one point of time, we will want to commit the changes to our local repository's history.
    - We can do that using git commit-m "a message"



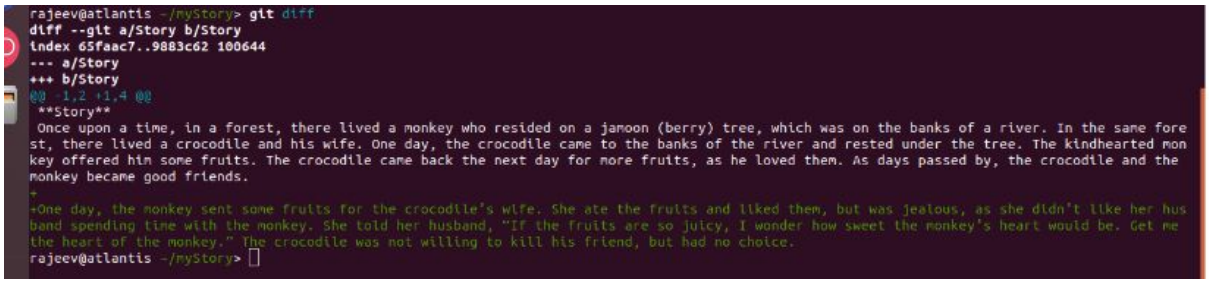
7. Let's make a few more changes in our story and stage the changes again.



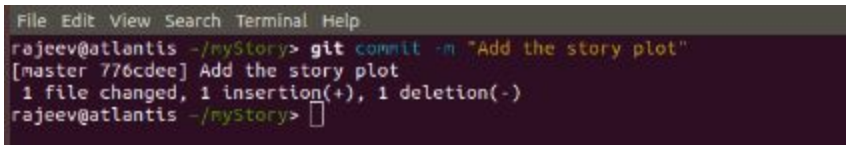


8. There is a beautiful command using which you can see the difference between the two files since the last commit and the current staged file.
    - You can do this using 'git diff --staged'. This will show you the difference between the last committed file and the currently staged file.

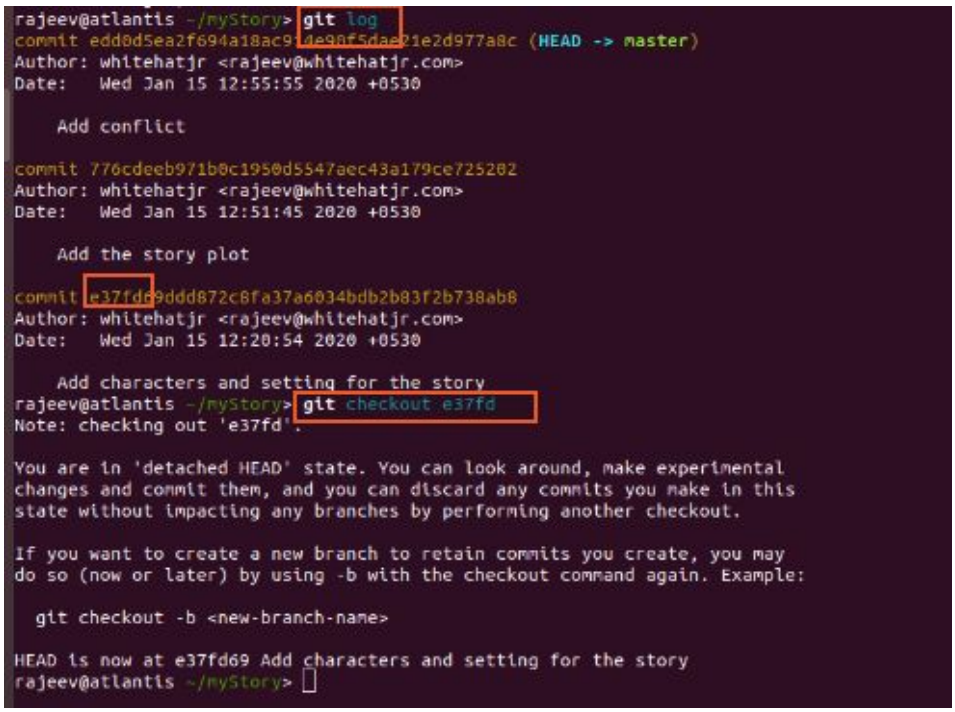
9. You can also make more changes in the working directory and see the difference between your working directory and the currently staged file.
    - We do that using "git diff".
        - git diff is very helpful if a developer is looking at changes made by someone else in their code since they last worked on it.



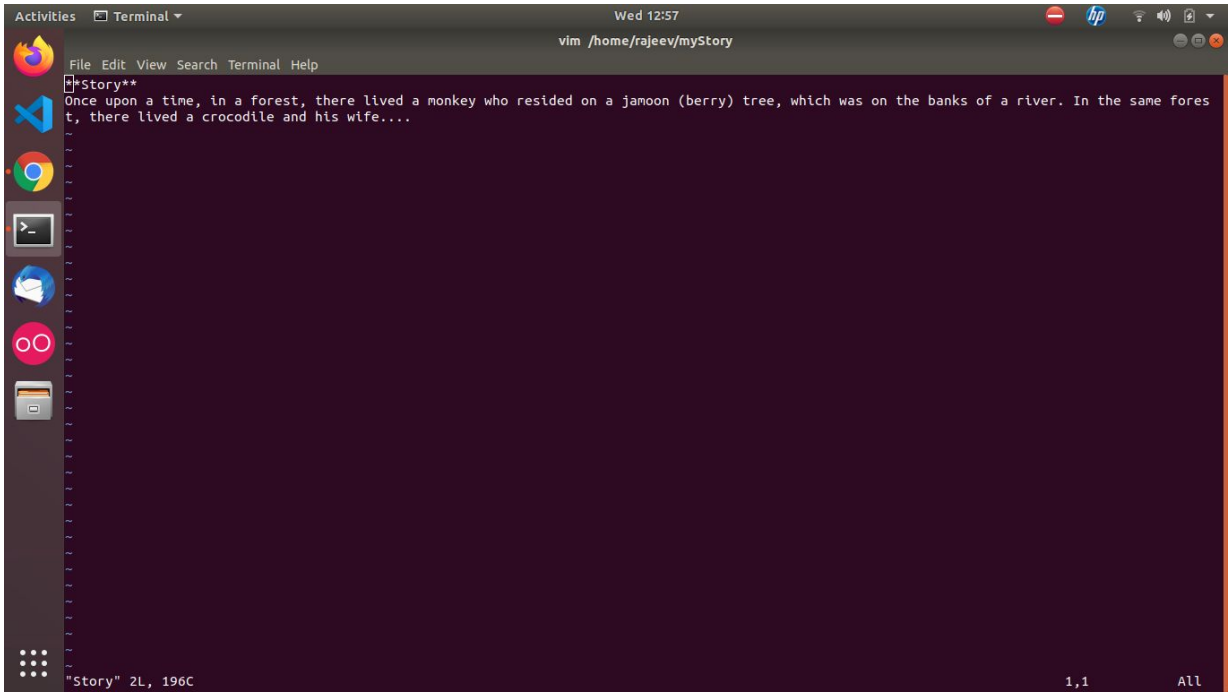
10. We can commit the file to the local repo again with a commit message.



11. Let's say you don't like the current story plot which you have written and want to move back to the older commit.
    - You can see all your commits using the git log command.
    - You can also see the time of the commit, the author and the commit message.
    - There is also a commit id given to each commit. You can use the id or even the first 5 characters of the id to move back to that commit using git checkout <commit id>



- You can open the Story file and see that it contains the text you had written in your first commit!
- This way you can move around to any commit history you have made.
- If you are properly committing your work for every story line (or every feature while writing code), you can fearlessly experiment with your work without having the fear of losing your work.

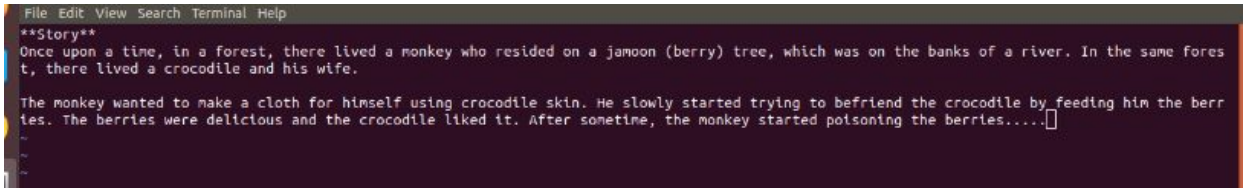


12. To explore another plot in your story or add new characters, create a new branch here.
    - Your current branch was called 'master' by default.
    - We all know this story where the monkey is the hero and crocodile's wife is the villain. Let's flip the story in our new branch. Let's make the crocodile's wife a hero and monkey the villain.
    - You can create a new branch using - git branch <branchName>
    - You can also see all the branches in your local repo using the 'git branch' command.

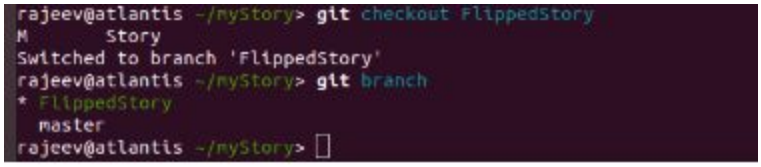
13. After creating the branch, you will also have to checkout to that branch using git checkout command. This will move you to the FlippedStory branch.

14. Let's write our flipped story here, stage the file and write a new commit.

- Flipped Story:



- Creating Git branch:



15. If you currently press git log, it will show you the commits log for only the current branch.

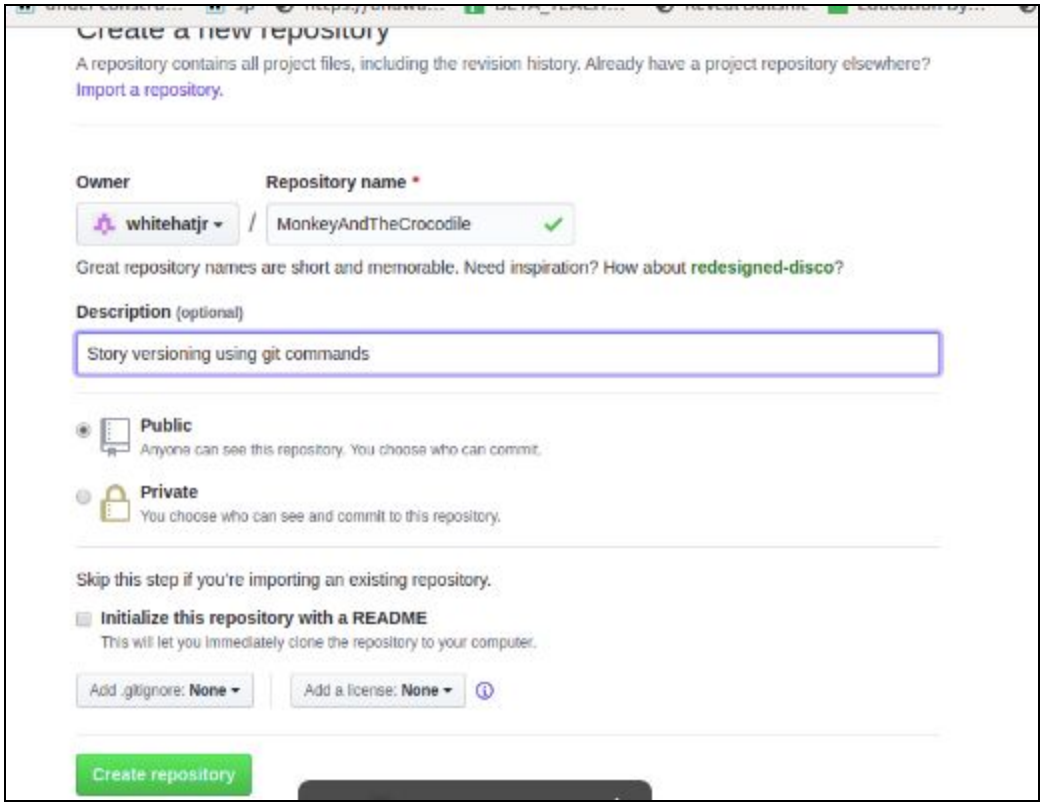
- At any point of time, you can abandon this branch and move to the master branch using 'git checkout master'. It will take you to the last commit you had made in the master branch.
- Later you can switch to FlippedBranch anytime and continue writing the story wherever you have left.

```
rajeev@atlantis ~/myStory> git log
commit edd0d5ea2f694a18ac914e90f5dae21e2d977a8c (HEAD -> master)
Author: whitehatjr <rajeev@whitehatjr.com>
Date:   Wed Jan 15 12:55:55 2020 +0530

    Add conflict

commit 776cdeeb971b0c1950d5547aec43a179ce725202
Author: whitehatjr <rajeev@whitehatjr.com>
Date:   Wed Jan 15 12:51:45 2020 +0530

    Add the story plot

commit e37fd69ddd872c8fa37a6034bdb2b83f2b738ab8
Author: whitehatjr <rajeev@whitehatjr.com>
Date:   Wed Jan 15 12:20:54 2020 +0530

    Add characters and setting for the story
rajeev@atlantis ~/myStory>
```

16. You can check the difference between the lines in the HEAD of the two branches using - git diff branchA branchB
    - HEAD here refers to the latest commit of the two branches.
    - Sometimes two branches can work towards the same feature and then later they can be merged using git merge command.

```
rajeev@atlantis ~/myStory> git branch
  FlippedStory
* master
rajeev@atlantis ~/myStory> git diff master FlippedStory
diff --git a/Story b/Story
index 9883c62..8955f00 100644
--- a/Story
+++ b/Story
@@ -1,4 +1,4 @@
 **Story**
-Once upon a time, in a forest, there lived a monkey who resided on a jamoon (berry) tree, which was on the banks of a river. In the same for
st, there lived a crocodile and his wife. One day, the crocodile came to the banks of the river and rested under the tree. The kindhearted mo
key offered him some fruits. The crocodile came back the next day for more fruits, as he loved them. As days passed by, the crocodile and the
monkey became good friends.
+Once upon a time, in a forest, there lived a monkey who resided on a jamoon (berry) tree, which was on the banks of a river. In the same for
st, there lived a crocodile and his wife.

-One day, the monkey sent some fruits for the crocodile's wife. She ate the fruits and liked them, but was jealous, as she didn't like her hu
band spending time with the monkey. She told her husband, "If the fruits are so juicy, I wonder how sweet the monkey's heart would be. Get me
the heart of the monkey." The crocodile was not willing to kill his friend, but had no choice.
+The monkey wanted to make a cloth for himself using crocodile skin. He slowly started trying to befriend the crocodile by feeding him the be
ries. The berries were delicious and the crocodile liked it. After sometime, the monkey started poisoning the berries.....
rajeev@atlantis ~/myStory>
```

17. Now, let's upload all our commits to a remote repository (also called upstream repository).
    - We will first need to create an empty Github repository.

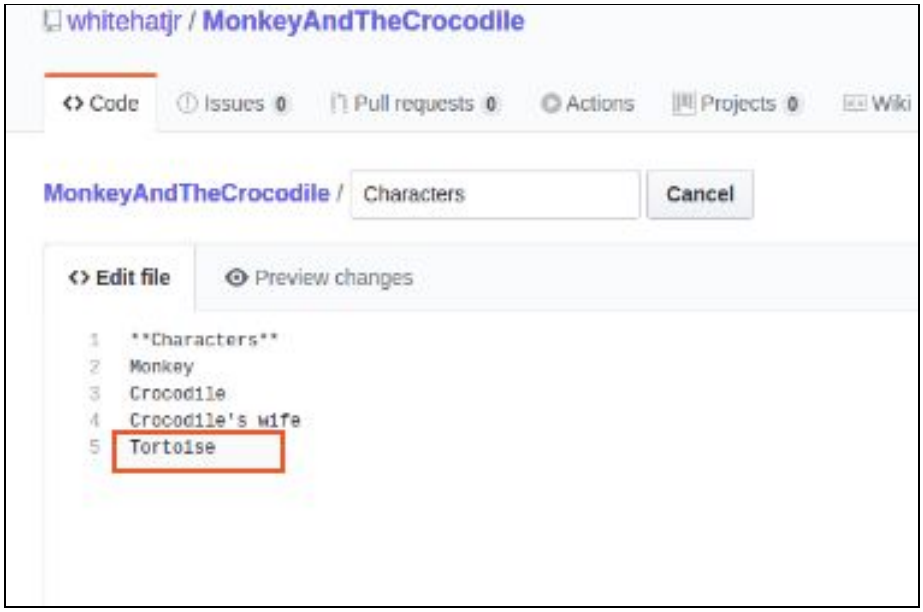


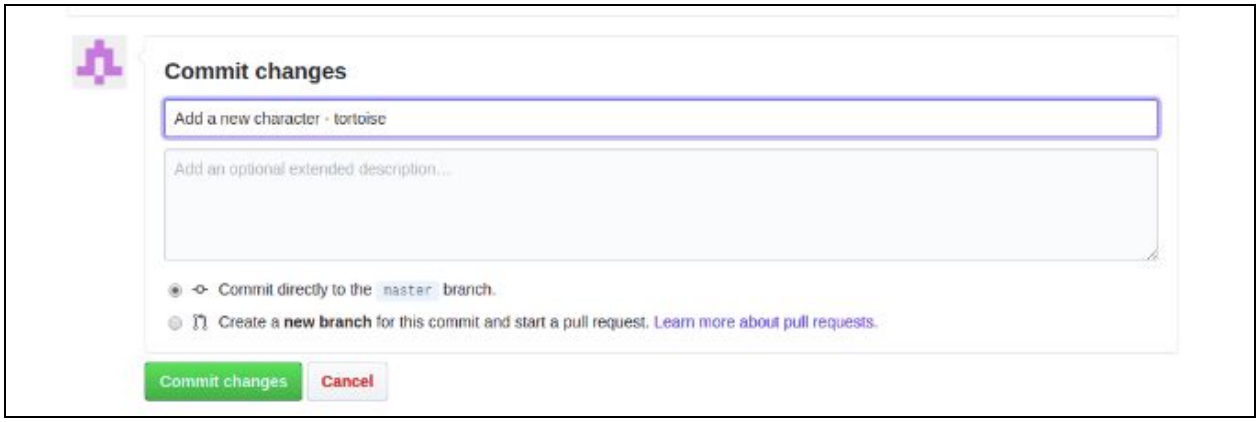
- Now we need to set the remote (upstream) repository for our local repo.

```
rajeev@atlantis ~/myStory> git remote add origin https://github.com/whitehatjr/MonkeyAndTheCrocodile.git
rajeev@atlantis ~/myStory>
```

```
rajeev@atlantis ~/myStory> git push origin master
Password for 'https://whitehatjr@github.com':
Counting objects: 10, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 1.19 KiB | 609.00 KiB/s, done.
Total 10 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/whitehatjr/MonkeyAndTheCrocodile.git
 * [new branch]      master -> master
rajeev@atlantis ~/myStory>
```

- Change a file on the remote repo and make a commit.





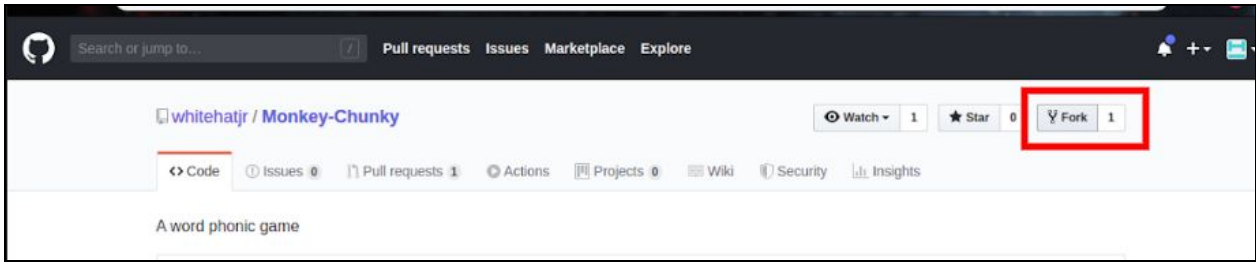
- We will not be able to make any new pushes to the remote repo since our local repo is out of sync. We can use git pull to update our repo with the upstream repo.

*Note: If there are too many conflicts in the current working directory vs the remote repo, you will be asked to make a new commit. Make a new commit and then apply the git push.

- Now you can open the working directory to see that the current repo is updated with the new name of the character.

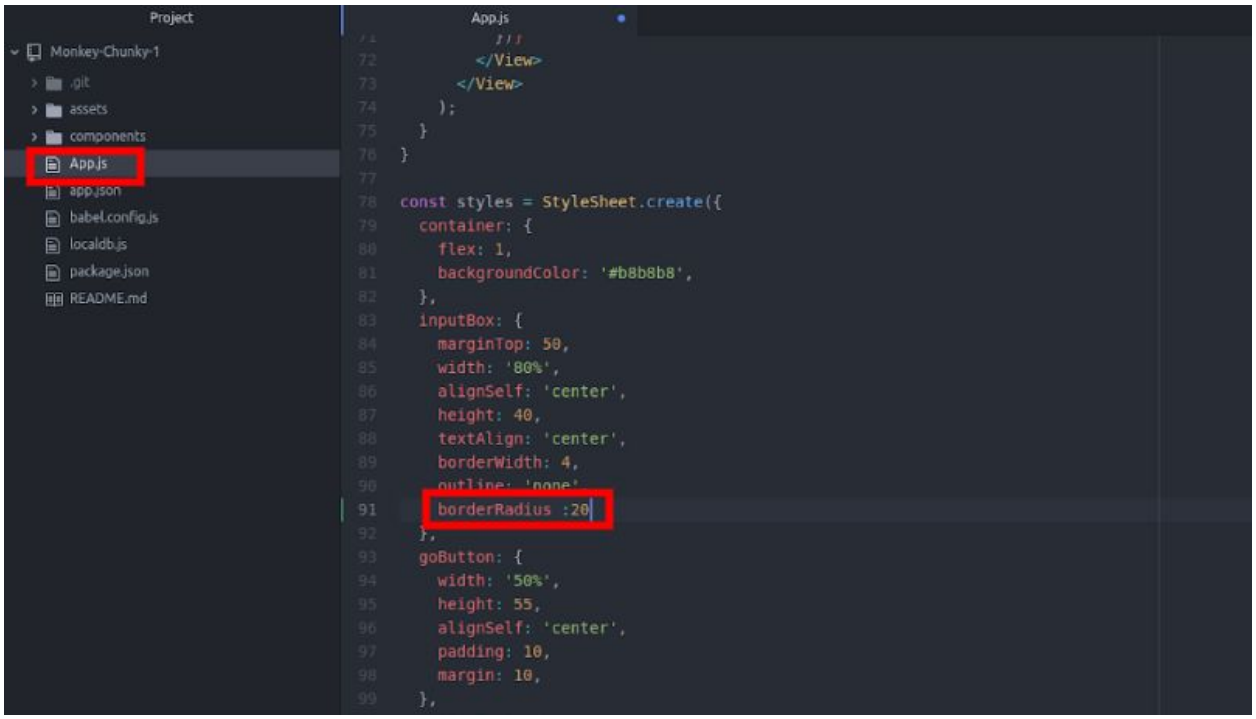
To fork the project:

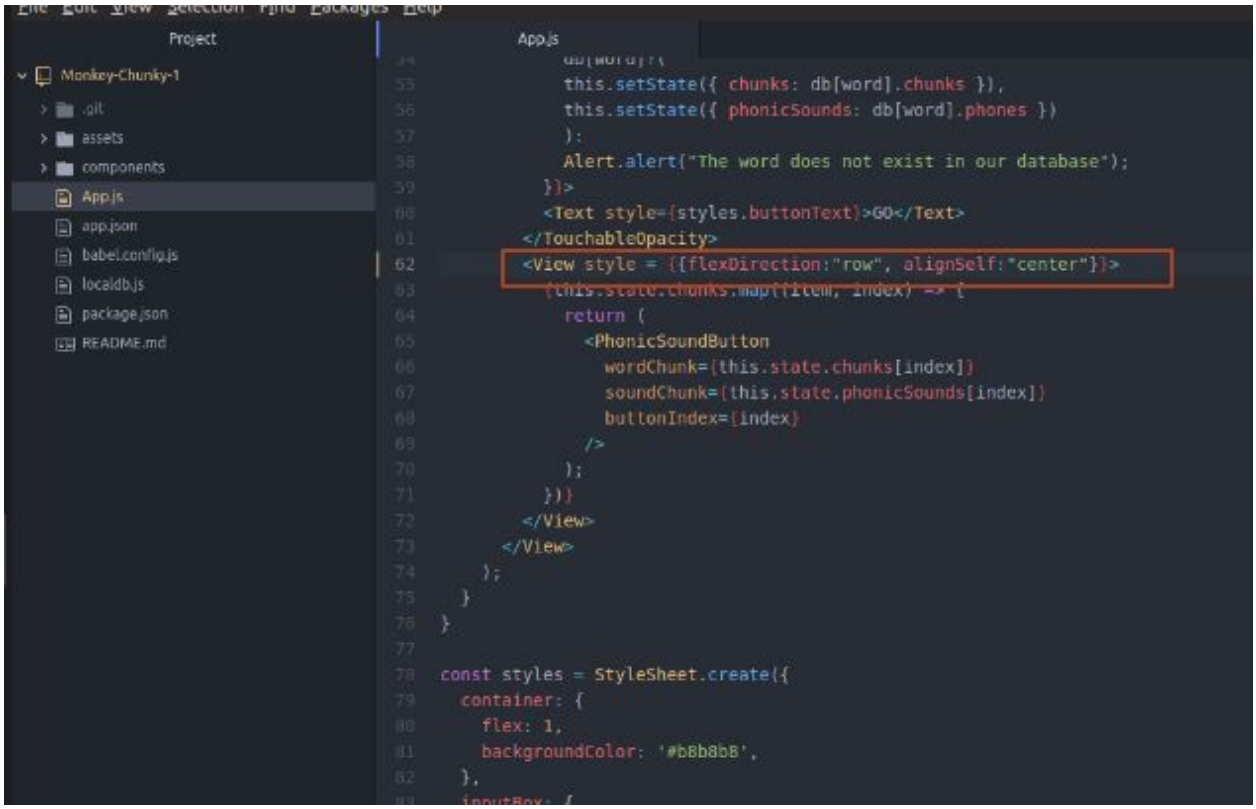
Forking creates a duplicate of the repository in your own account where you can work and modify the contents of the project.



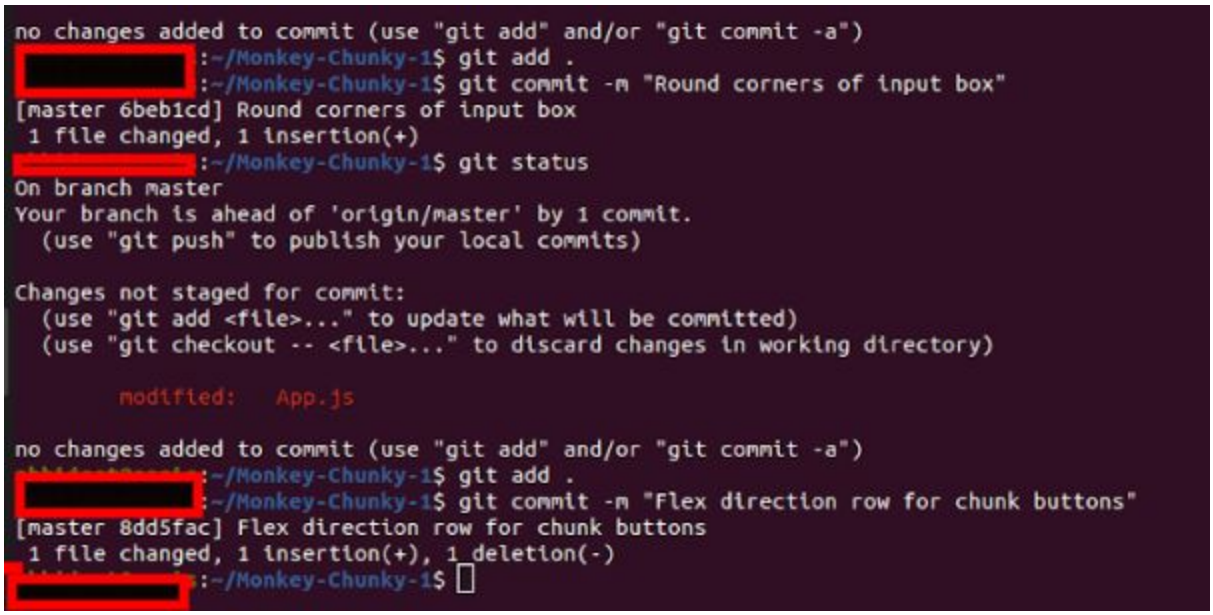
1. This is the forked repository. You can clone this repository into your system.

2. Now cd into your forked repo. Let's make some changes to this project.
    - Some example changes you can make are:
        - You can make the input box rounded.
        - You can make the phonic chunk buttons in a horizontal row instead of column by changing the flex-direction.

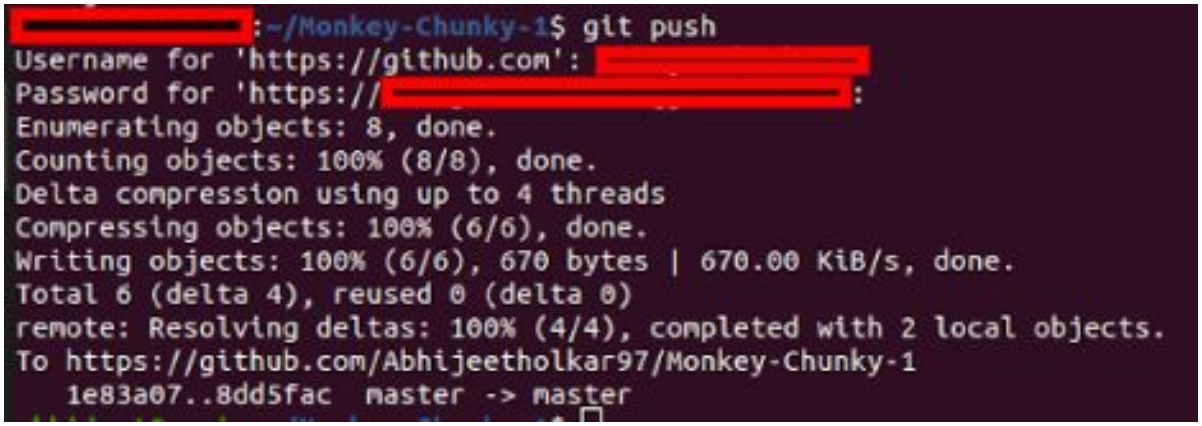




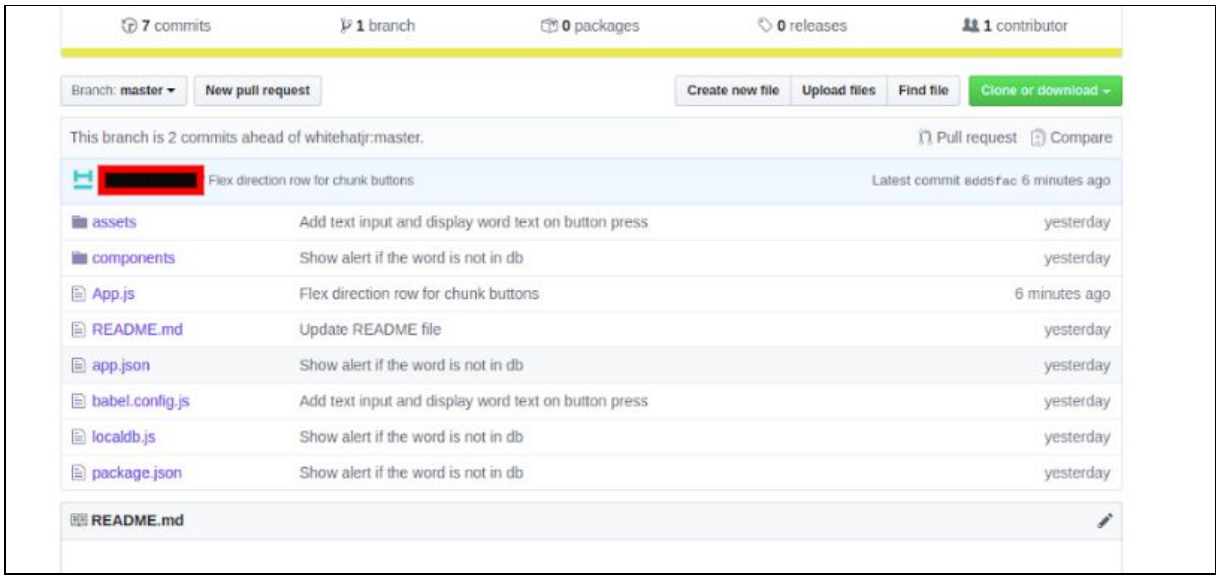
3. Finally, you can stage and commit these changes.



4. Push the new commits to your forked repo.

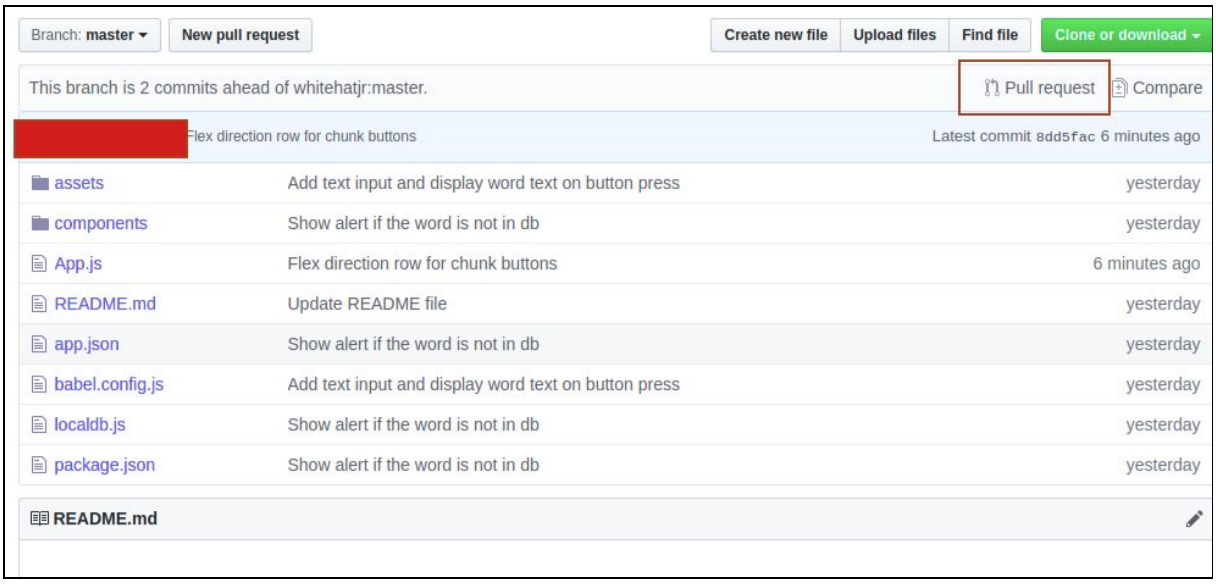


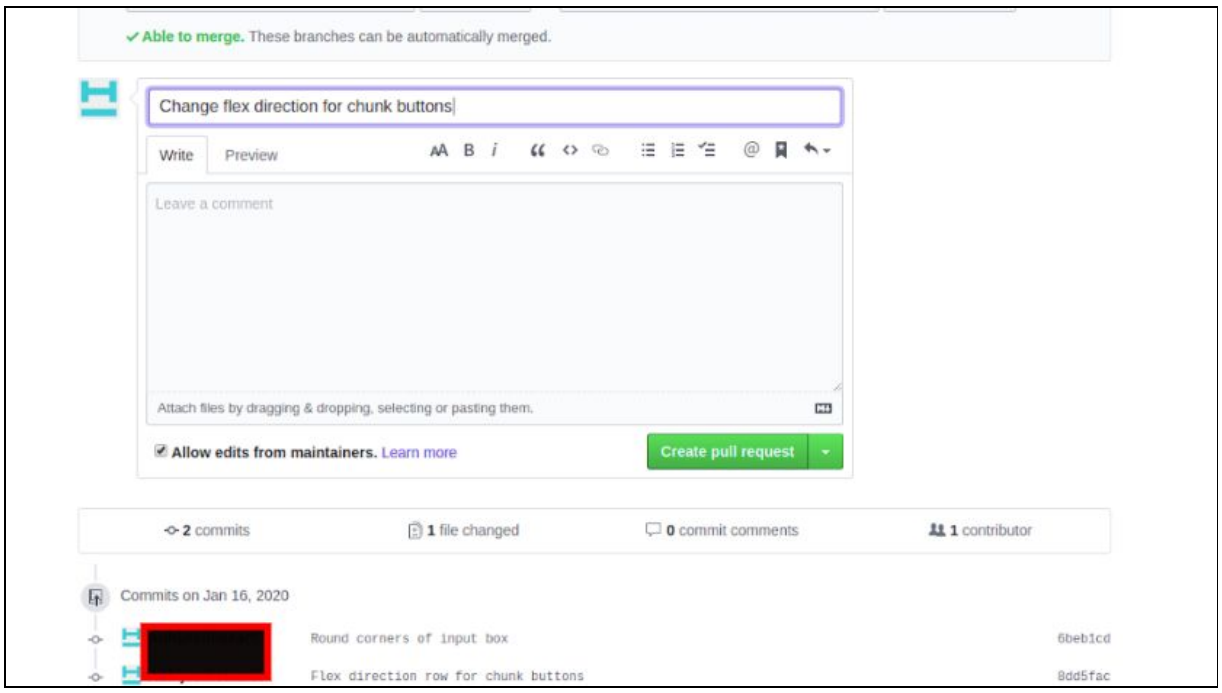
5. Let's see if the new commits are added in the remote forked repo.



Now your forked repo is different from the original repo.

6. You can make a pull request on the original repo.
    - This will send a request to the original developer of the project to pull the changes you have made into their current project.
    - You will have to write a message to tell the developer what change you made.
    - The developer can accept the pull request and merge the changes into their current project.
    - The pull request will remain open till it is merged with the original project or closed by the developer.
    - Once merged, you will then be added as a contributor to the project.





What's NEXT?

In the next class, we will be starting on a new project to build a simple library management system.

EXTEND YOUR KNOWLEDGE

1. Git Command Line: https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html