Git heads

- The head points out the last commit in the current checkout branch. It is like a pointer to any reference. The HEAD can be understood as the “current branch” When you switch branches with 'checkout,' the HEAD is transferred to the new branch.
- By using below command is used to check the status of head
- git show HEAD (OR) git show commit id

```
lenovo@akhil MINGW64 ~/Desktop/meena (chinni|CHERRY-PICKING)
$ git show HEAD
commit 08beca199c5733c72931ee8dfad873f669cf279b (HEAD -> chinni, origin/master, master)
Author: akhil <akhil13.gandla@gmail.com>
Date:   Tue Oct 5 09:02:18 2021 +0530

    changes are done by gunndumalla chinni alias prathyusha

diff --git a/text2.txt b/text2.txt
new file mode 100644
index 0000000..22b4f0c
--- /dev/null
+++ b/text2.txt
@@ -0,0 +1 @@
+meena and chinni both are sisters
\ No newline at end of file
```

- In the above output, you can see that the commit id for the Head is given. It means the Head is on the given commit
- Now, check the commit history of the project. You can use the git log command to check the commit history. See the below output:

```
lenovo@akhil MINGW64 ~/Desktop/meena (master)
$ git log
commit 08beca199c5733c72931ee8dfad873f669cf279b (HEAD -> master, origin/master, chinni)
Author: akhil <akhil13.gandla@gmail.com>
Date:   Tue Oct 5 09:02:18 2021 +0530

    changes are done by gunndumalla chinni alias prathyusha

commit 2c603889f9be40911659e656108312cf546fb247
Author: akhil <akhil13.gandla@gmail.com>
Date:   Tue Oct 5 08:58:01 2021 +0530

    chnages are done by gundumalla meena maadhuri
```

- As we can see in the above output, the commit id for most recent commit and Head is the same. So, it is clear that the last commit has the Head.
- We can also check the status of the Head by the commit id. Copy the commit id from the above output and paste it with the **git show** command. Its result is same as **git show head** command if the commit id is last commit's id. See the below output:

```
lenovo@akhil MINGW64 ~/Desktop/meena (chinni|CHERRY-PICKING)
$ git show 08beca199c5733c72931ee8dfad873f669cf279b
commit 08beca199c5733c72931ee8dfad873f669cf279b (HEAD -> chinni, origin/master, master)
Author: akhil <akhil13.gandla@gmail.com>
Date:   Tue Oct 5 09:02:18 2021 +0530

    changes are done by gunndumalla chinni alias prathyusha

diff --git a/text2.txt b/text2.txt
new file mode 100644
index 0000000..22b4f0c
--- /dev/null
+++ b/text2.txt
@@ -0,0 +1 @@
+meena and chinni both are sisters
\ No newline at end of file
```

Git Detached Head

- When **Head doesn't point to most recent commit, such state is called detached Head**. If you checkout with an older commit, it will stand the detached head condition.

```
lenovo@akhil MINGW64 ~/Desktop/meena (master)
$ git checkout  2c603889f9be40911659e656108312cf546fb247
Note: switching to '2c603889f9be40911659e656108312cf546fb247'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 2c60388 chnages are done by gundumalla meena maadhuri
```

Number of heads created in repository

- For instance, “master” is a “head” because it is a reference to a branch. If we are viewing the master branch, then “master” is also our HEAD. If we are not viewing the master branch, then whatever branch or commit we are viewing is our HEAD. A repository can contain a number of heads but only **one HEAD**

tags

- Tags make a point as a specific point in Git history. Tags are used to mark a commit stage as relevant. We can tag a commit for future reference. Primarily, it is used to mark a project's initial point like v1.1.
- Tags are much like branches, and they do not change once initiated. We can have any number of tags on a branch or different branches.
- For create tag use below command
- git tag <tag name>

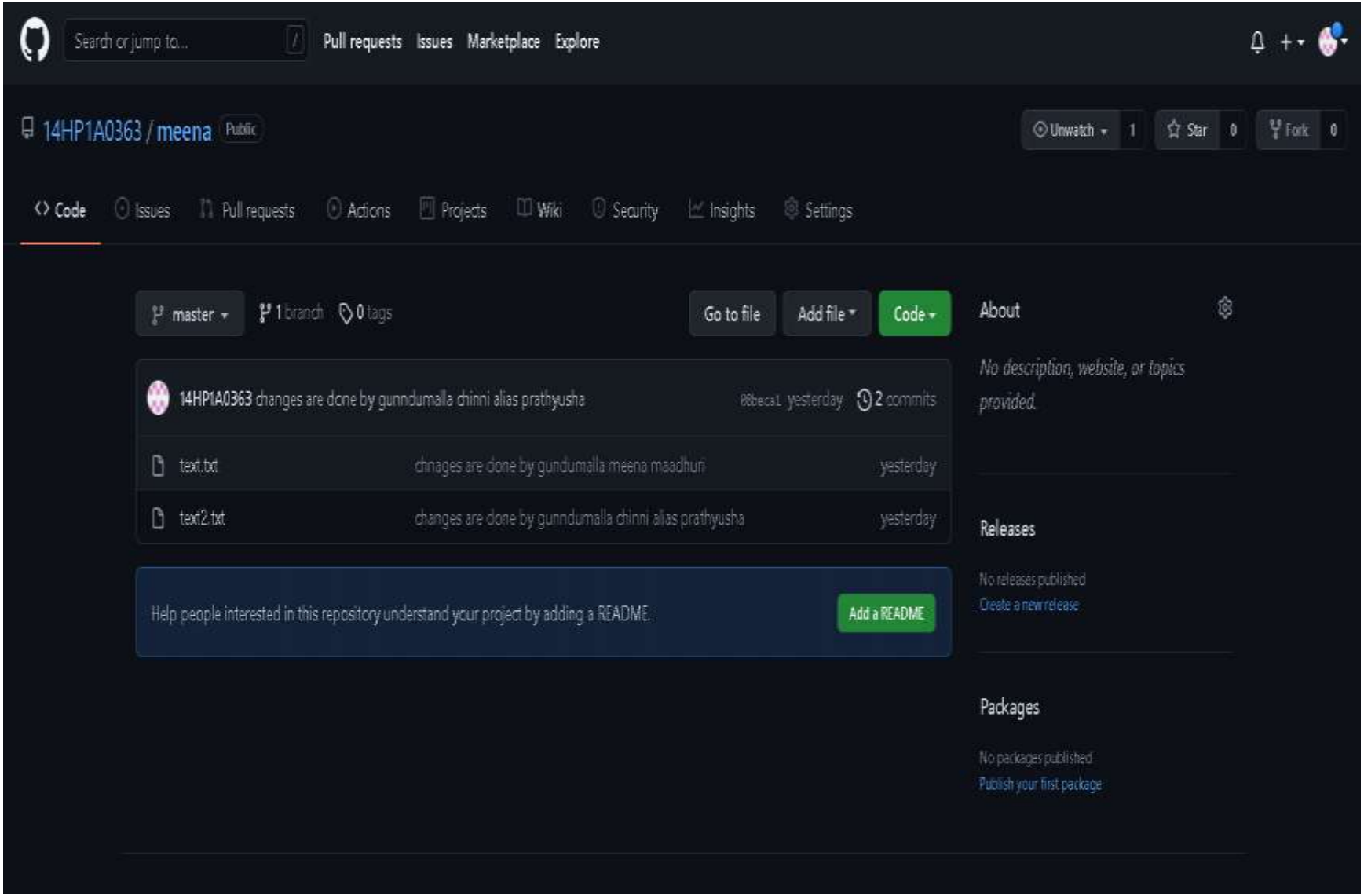
- We can list the available tags in our repository. There are three options that are available to list the tags in the repository. They are as follows:
- git tag
- git show
- git tag -l ".*“
- Types of Git tags
- There are two types of tags in git. They are as:
- Annotated tag
- Light-weighted tag
- Let's understand both of these tags in detail.
- Annotated Tags
- Annotated tags are tags that store extra Metadata like developer name, email, date, and more. They are stored as a bundle of objects in the Git database.
- If you are pointing and saving a final version of any project, then it is recommended to create an annotated tag. But if you want to make a temporary mark point or don't want to share information, then you can create a light-weight tag.

- The data provided in annotated tags are essential for a public release of the project. There are more options available to annotate, like you can add a message for annotation of the project.

- Light-Weighted Tag:
- Git supports one more type of tag; it is called as Light-weighted tag. The motive of both tags is the same as marking a point in the repository. Usually, it is a commit stored in a file. It does not store unnecessary information to keep it light-weight. No command-line option such as **-a,-s** or **-m** are supplied in light-weighted tag, pass a tag name.

- Git Fork
- A fork is a rough copy of a repository. Forking a repository allows you to freely test and debug with changes without affecting the original project. One of the excessive use of forking is to propose changes for bug fixing. To resolve an issue for a bug that you found, you can:
- Fork the repository.
- Make the fix.
- Forward a pull request to the project owner.
- Forking is not a Git function; it is a feature of Git service like GitHub.
- **When to Use Git Fork**
- Generally, forking a repository allows us to experiment on the project without affecting the original project. Following are the reasons for forking the repository:
- Propose changes to someone else's project.
- Use an existing project as a starting point.

- Fork a Repository
- The forking and branching are excellent ways to contribute to an open-source project. These two features of Git allows the enhanced collaboration on the projects.
- Forking is a safe way to contribute. It allows us to make a rough copy of the project. We can freely experiment on the project. After the final version of the project, we can create a pull request for merging.
- It is a straight-forward process. Steps for forking the repository are as follows:
- Login to the GitHub account.
- Find the GitHub repository which you want to fork.
- Click the Fork button on the upper right side of the repository's page.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
14HP1A0363 / meena
Public
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
1 branch
0 tags
Go to file
Add file
Code
14HP1A0363 changes are done by gunndumalla chinni alias prathyusha
88beca1 yesterday
2 commits
text.txt
chnages are done by gundumalla meena maadhuri
yesterday
text2.txt
changes are done by gunndumalla chinni alias prathyusha
yesterday
Help people interested in this repository understand your project by adding a README.
Add a README
About
No description, website, or topics provided.
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package