😊 Branches and More

In Git, branches are essentially separate lines of development. When you start working on a project, the default branch is usually called "master" (though some prefer "main" for more inclusive language). This is the primary branch where the main codebase resides.

When you create a new branch, you're essentially creating a copy of the codebase at that point in time. You can then work on new features, fixes, or experiments without affecting the main codebase. Each branch maintains its own commit history, allowing for parallel development.

Branches are incredibly useful because they enable collaboration and experimentation without the risk of breaking the main codebase. Once your changes are ready, you can merge them back into the main branch, incorporating your updates into the project's mainline.

Git makes branching lightweight and efficient, allowing developers to create, switch between, and merge branches with ease. This flexibility is a fundamental aspect of Git's popularity among developers.

😄 Use Cases of Branches in Git:

Here are some common use cases for branches in Git:

1. **Feature Development:** When you're working on a new feature or functionality for your project, you can create a new branch to isolate your changes. This allows you to work independently without interfering with the stable codebase on the main branch. Once your feature is complete, you can merge it back into the main branch.
2. **Bug Fixing:** If you encounter a bug in the main branch of your project, you can create a new branch specifically for fixing that bug. This allows you to focus solely on the fix without disrupting other ongoing development efforts. Once the bug is fixed, you can merge the bug-fix branch back into the main branch.
3. **Experimentation:** Branches provide a safe environment for experimenting with new ideas or approaches. You can create a new branch to test out a new algorithm, library, or design pattern without affecting the stability of the main codebase. If the experiment is successful, you can integrate the changes into the main branch. If not, you can simply discard the branch.
4. **Code Reviews:** Branches facilitate code reviews by allowing developers to create separate branches for code changes. Each developer can work on their feature or fix in isolation, and then create a pull request to merge their branch into the main branch. This makes it easier for team members to review and provide feedback on the changes before they are merged.
5. **Release Management:** Branches are often used to manage releases and versions of a project. For example, you can create a branch for a stable release that contains only bug fixes, while continuing development on new features in separate branches. This allows you to maintain multiple versions of your project simultaneously.

In this guide, we're exploring the usage of Git branches for managing code development. We start by creating branches for different tasks like feature development, bug fixing, or experimentation. Throughout the process, we make changes to these branches independently, ensuring that the main codebase remains stable. The end goal is to merge these changes back into the main branch once they're ready, incorporating the new features or fixes while maintaining the project's integrity. Finally, we demonstrate how to clone repositories and synchronize changes between local and remote repositories. Overall, the aim is to facilitate collaborative and organized software development using Git's branching model.

😄 To begin with the Lab:

1. In this lab we work with the branches in Git.
2. First, we are going to create a new branch in our local system using this command.
3. Below first we created a new branch then using the next command we listed all the branches that are available in our repository.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (main)
$ git branch -c mars1

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (main)
$ git branch -a
* main
  mars1
  remotes/origin/main
```

4. Now first we listed all the objects that are available in our main branch then we switched to our new branch, and when we listed all the objects here then we get to know that the data here is also the same. It is just a copy of our main branch.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (main)
$ ls
gsat9/    moon1.py   moon2.py  moon4.py  moon6.py  moon8.py  scross1/
insat1a/  moon10.py  moon3.py  moon5.py  moon7.py  moon9.py

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (main)
$ git checkout mars1
Switched to branch 'mars1'
Your branch is up to date with 'origin/main'.

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ ls
gsat9/    moon1.py   moon2.py  moon4.py  moon6.py  moon8.py  scross1/
insat1a/  moon10.py  moon3.py  moon5.py  moon7.py  moon9.py
```

5. After that we removed some of our files using git rm command then we listed to check whether they are removed or not. Then we checked the status.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ git rm moon6.py moon8.py moon7.py moon9.py
rm 'moon6.py'
rm 'moon7.py'
rm 'moon8.py'
rm 'moon9.py'

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ ls
gsat9/  insat1a/  moon1.py  moon10.py  moon2.py  moon3.py  moon4.py  moon5.py  scross1/

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ git status
On branch mars1
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    moon6.py
        deleted:    moon7.py
        deleted:    moon8.py
        deleted:    moon9.py
```

6. Then we changed the name of a file using the git mv command.
7. After that we removed some files again.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ ls
gsat9/  insat1a/  moon1.py  moon10.py  moon2.py  moon3.py  moon4.py  moon5.py  scross1/

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ git mv moon1.py moon11.py

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ ls
gsat9/  insat1a/  moon10.py  moon11.py  moon2.py  moon3.py  moon4.py  moon5.py  scross1/

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ git rm moon2.py  moon3.py  moon4.py  moon5.py
rm 'moon2.py'
rm 'moon3.py'
rm 'moon4.py'
rm 'moon5.py'
```

8. Then we created some more files now we can say that we have made some drastic changes in our repository.

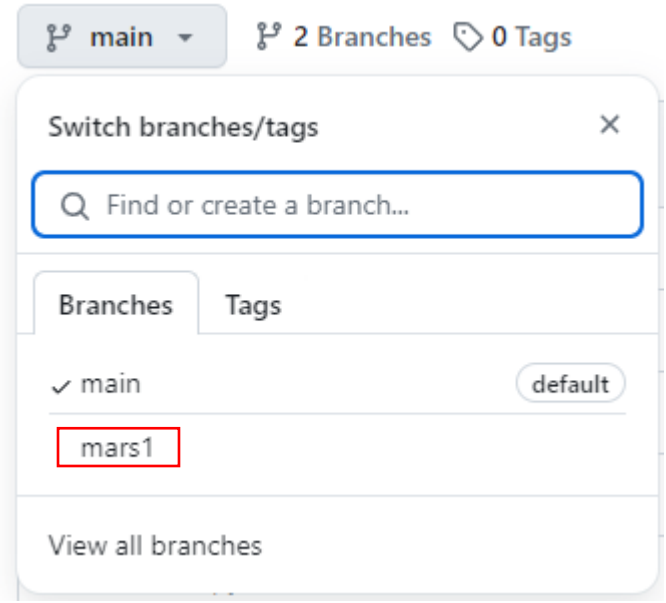
```
$ touch tesla{1..4}

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ ls
gsat9/  insat1a/  moon10.py  moon11.py  scross1/  tesla1  tesla2  tesla3  tesla4
```

9. After that we performed the commands to push the changes to our GitHub repository. First we gave the git add . command then we use git commit to commit our changes. Then we use the push command.

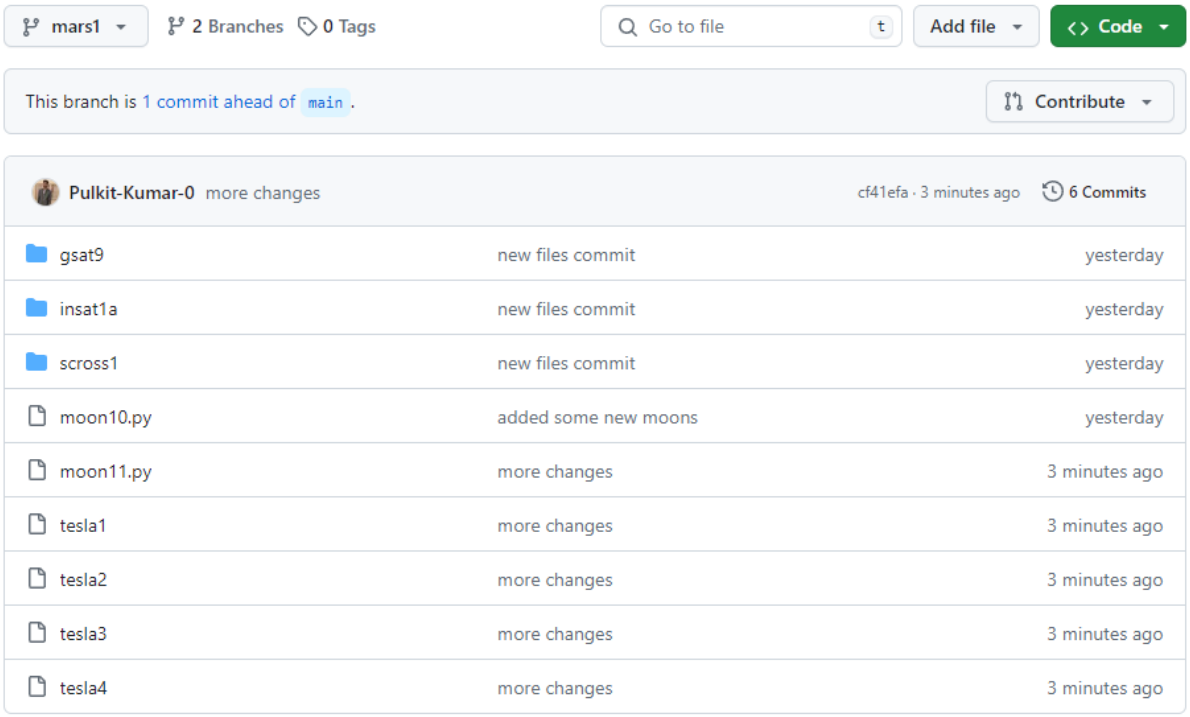
10. Now while using the push command we are on the mars1 branch. So, for now this is our main branch.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ git add .

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ git commit -m "more changes"
[mars1 cf41efa] more changes
 9 files changed, 6 deletions(-)
 rename moon1.py => moon11.py (100%)
 delete mode 100644 moon2.py
 delete mode 100644 moon7.py
 delete mode 100644 moon8.py
 delete mode 100644 moon9.py
 rename moon3.py => tesla1 (100%)
 rename moon4.py => tesla2 (100%)
 rename moon5.py => tesla3 (100%)
 rename moon6.py => tesla4 (100%)

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ git push origin mars1
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 292 bytes | 292.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'mars1' on GitHub by visiting:
remote:      https://github.com/Pulkit-Kumar-0/sattelitework/pull/new/mars1
remote:
To github.com:Pulkit-Kumar-0/sattelitework.git
 * [new branch]      mars1 -> mars1
```

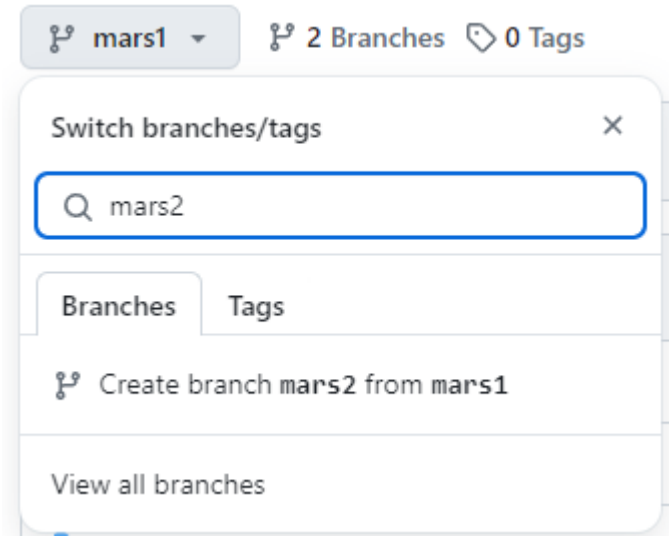
11. Then open your GitHub, here if you will click on the main then you can see your branches.



12. And also you can see the data in your mars1 branch.



13. Also, you can create a new branch from GitHub itself. But this new branch will be created using your previous branch so it will have the same data.



14. Then go your git bash and perform the git pull command and you will see the changes in your local system too.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ git pull
From github.com:Pulkit-Kumar-0/sattelitework
 * [new branch]      mars2      -> origin/mars2
Already up to date.
```

15. Now we ran the git checkout command to change our branch to mars2 and then list our objects.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$  git checkout mars2
Switched to a new branch 'mars2'
branch 'mars2' set up to track 'origin/mars2'.

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars2)
$ ls
gsat9/  insat1a/  moon10.py  moon11.py  scross1/  tesla1  tesla2  tesla3  tesla4
```

16. After that some more files.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars2)
$ touch lambo range merc bmw

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars2)
$ ls
bmw  gsat9/  insat1a/  lambo  merc  moon10.py  moon11.py  range  scross1/  tesla1  tesla2  tesla3  tesla4
```

17. After that again we performed the same steps to add commit and push the changes.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars2)
$ git add .

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars2)
$ git commit -m "changes"
[mars2 8b588c0] changes
 4 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 bmw
 create mode 100644 lambo
 create mode 100644 merc
 create mode 100644 range

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars2)
$ git push origin mars2
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 273 bytes | 273.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Pulkit-Kumar-0/sattelitework.git
   cf41efa..8b588c0  mars2 -> mars2
```

18. So, all of our branches having the different data.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars2)
$ git branch -a
  main
  mars1
* mars2
  remotes/origin/main
  remotes/origin/mars1
  remotes/origin/mars2
```

19. So, below you can see that we switch back to mars1 branch and we can see that data is different then we switch to main branch there also the data is different.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars2)
$ git switch mars1
Switched to branch 'mars1'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ ls
gsat9/  insat1a/  moon10.py  moon11.py  scross1/  tesla1  tesla2  tesla3  tesla4

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (mars1)
$ git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (main)
$ ls
gsat9/  insat1a/  moon1.py  moon10.py  moon2.py  moon3.py  moon4.py  moon5.py  moon6.py  moon7.py  moon8.py  moon9.py  scross1/
```

20. Then we used git merge command to merge all the data of mars1 to our main branch and you can see that it all went successfully.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (main)
$ git merge mars1
Updating ccc886c..cf41efa
Fast-forward
 moon1.py => moon11.py | 0
 moon2.py              | 6 ------
 moon7.py              | 0
 moon8.py              | 0
 moon9.py              | 0
 moon3.py => tesla1    | 0
 moon4.py => tesla2    | 0
 moon5.py => tesla3    | 0
 moon6.py => tesla4    | 0
 9 files changed, 6 deletions(-)
 rename moon1.py => moon11.py (100%)
 delete mode 100644 moon2.py
 delete mode 100644 moon7.py
 delete mode 100644 moon8.py
 delete mode 100644 moon9.py
 rename moon3.py => tesla1 (100%)
 rename moon4.py => tesla2 (100%)
 rename moon5.py => tesla3 (100%)
 rename moon6.py => tesla4 (100%)

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (main)
$ ls
gsat9/  insat1a/  moon10.py  moon11.py  scross1/  tesla1  tesla2  tesla3  tesla4
```

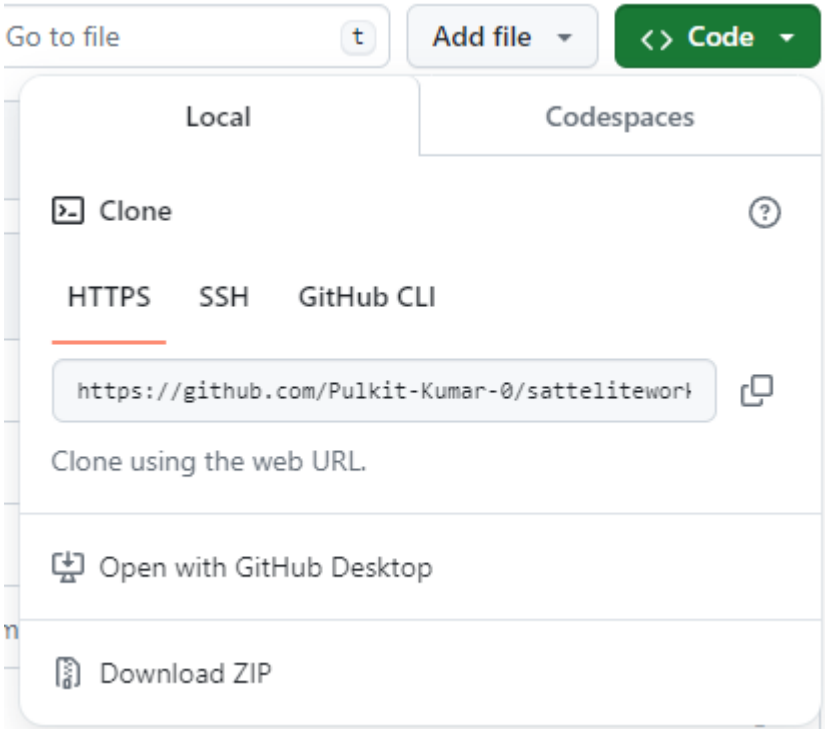
21. Now by using the git push --all origin command we push all the data to our repository.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (main)
$ git push --all origin
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Pulkit-Kumar-0/sattelitework.git
   ccc886c..cf41efa  main -> main
```

22. Now you can remove your repository from your local machine by deleting it or using the command.

    rm -rf sattelitework

23. Now we going to use a new command which is git clone. This command will help to clone everything from our repository on GitHub or from any other repository.
24. For that open GitHub, then click on Code the choose HTTPS and copy the URL.



25. After that open your git bash where you want to clone the repository.
26. Then write git clone then paste the URL after that. It will clone everything.
27. After that go into your repository and do the listing of objects. You will see the exact same objects there.

```
PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository
$ git clone https://github.com/Pulkit-Kumar-0/sattelitework.git
Cloning into 'sattelitework'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 22 (delta 6), reused 16 (delta 2), pack-reused 0
Receiving objects: 100% (22/22), done.
Resolving deltas: 100% (6/6), done.

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository
$ ls
sattelitework/

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository
$ cd sattelitework/

PULKIT@LAPTOP-G2CAKBK8 MINGW64 ~/Desktop/gitrepository/sattelitework (main)
$ ls
gsat9/  insat1a/  moon10.py  moon11.py  scross1/  tesla1  tesla2  tesla3  tesla4
```