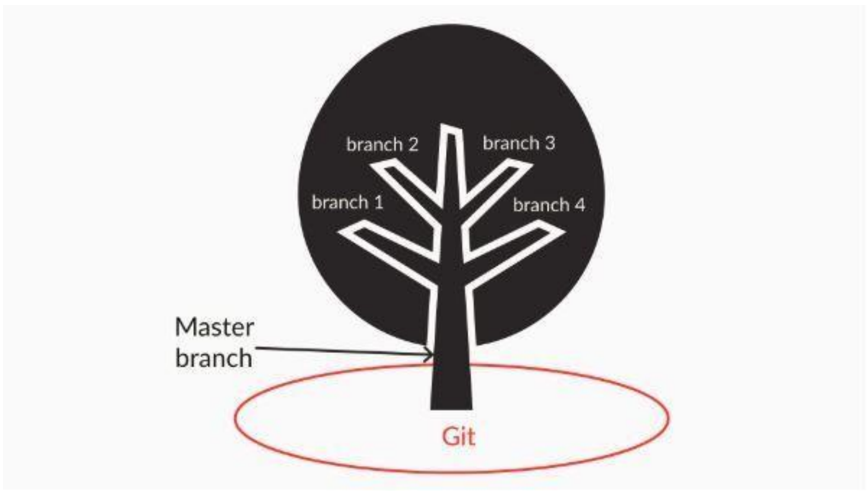
Lecture Notes for Version Control Part - ||

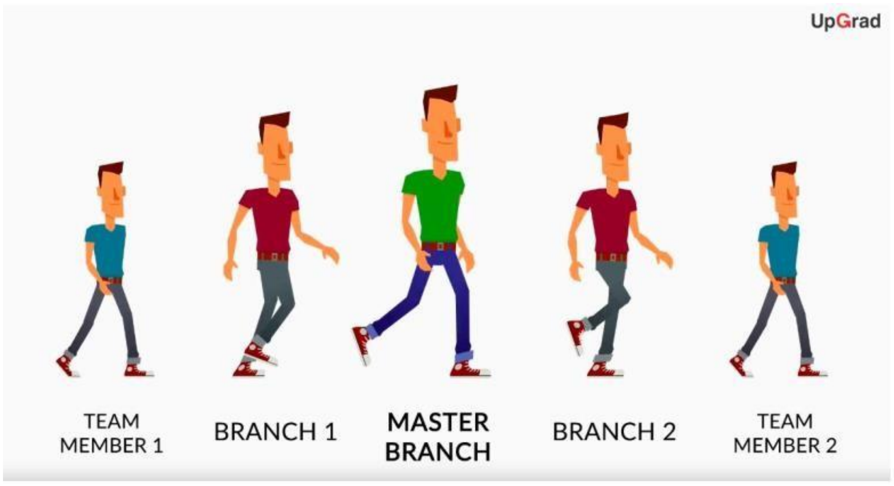
Branching :

When you come across the term branching, you might correlate it to branches of a tree.

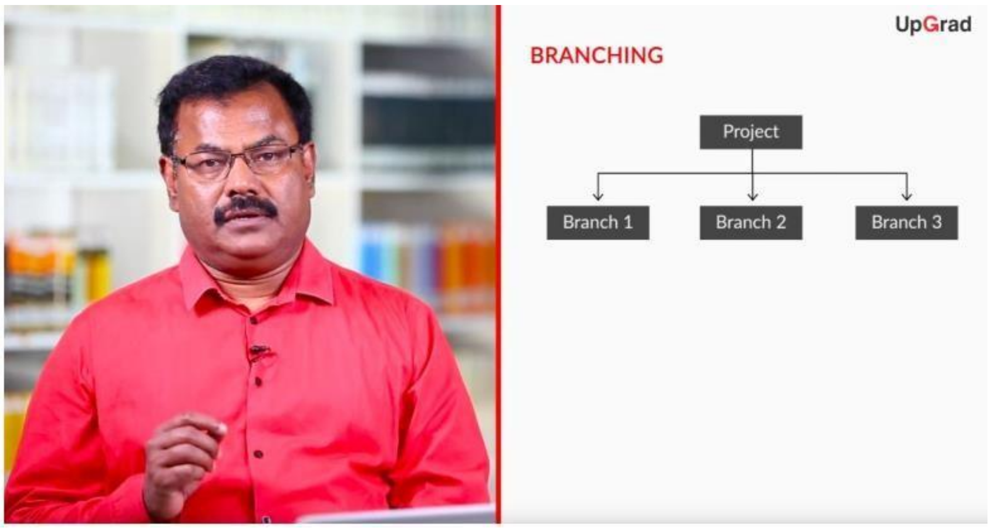
Well, yes! Branching means exactly the same. Imagine the branches growing out of the trunk of a tree.



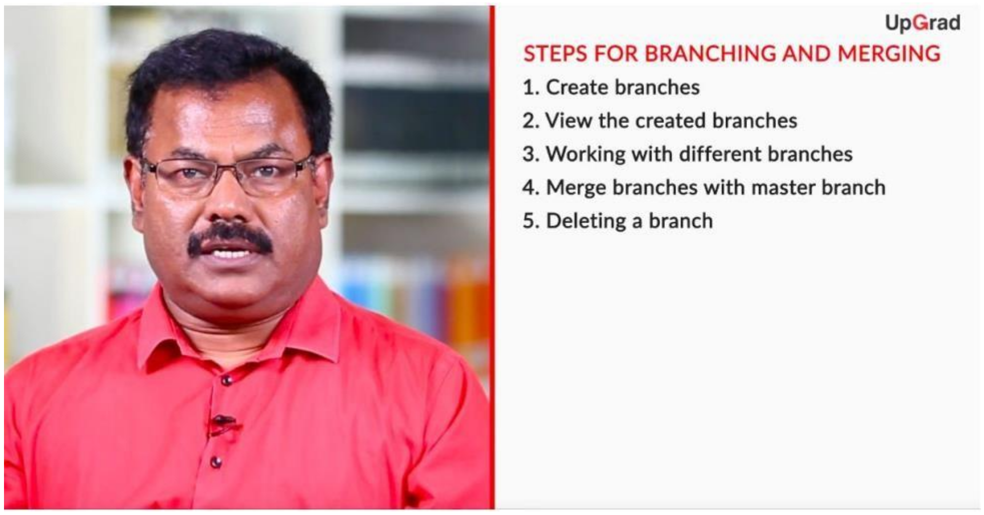
The trunk here plays the role of a master branch, and the branches coming out of the trunk represents the branches in git.



You can have your own copy and work on that, make modifications to it, and if the changes look fine you can merge them back to you master branch



Steps for branching and merging :



UpGrad

STEPS FOR BRANCHING AND MERGING

1. Create branches
2. View the created branches
3. Working with different branches
4. Merge branches with master branch
5. Deleting a branch

Working With Branches :

In this video, you learnt about the following steps:

- Creating branches
- Viewing the created branches
- Working with different branches concurrently

In this video, you learnt about the following commands:

- **git branch <branchname>**: This command will create a branch with the given branch name
- **git branch**: This command will show you all the branches along with the HEAD pointing to the branch you are currently working on
- **git checkout <branchname>**: If you want to move from one branch to another, you can run this command

In-Depth Study of the Concept of Branching

A branch in git is simply a lightweight, movable pointer that points to one of the commits. The default branch name in git is master, which points to the last commit. Every time you make a commit, the master moves forward to that commit.

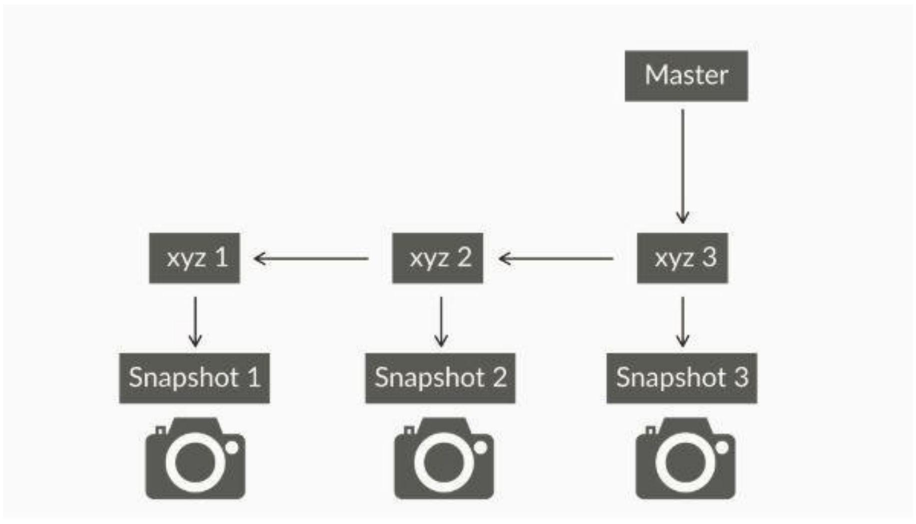


Figure 1. Here the master (branch) is pointing to our last commit

Now if you create another branch, what do you think will happen? When you create a new branch, git will create a new pointer at the same commit you're currently on.

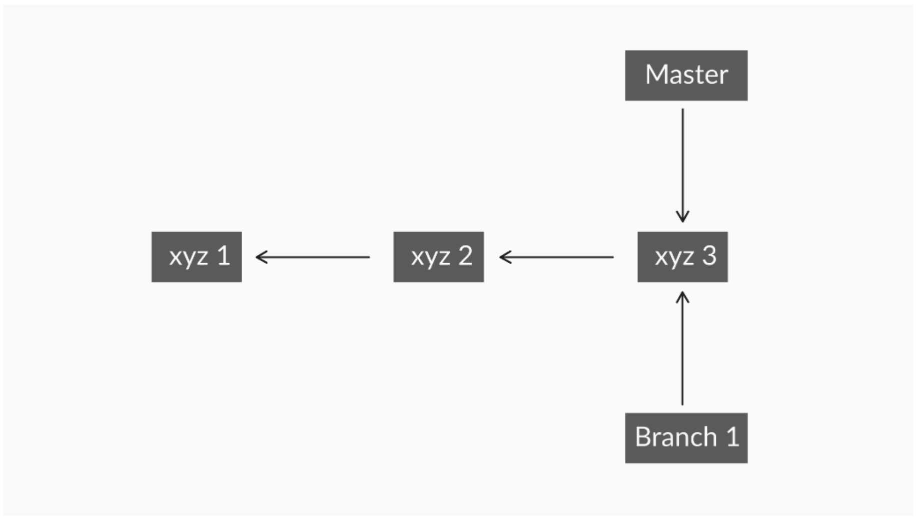


Figure 2: A new pointer at the same commit is created

Now, how do you think Git knows what branch you're currently working on?

Git keeps a special pointer called the HEAD. It points to the branch you are currently working on. To make the HEAD point to the new branch, i.e. Branch1, you will have to switch to that branch — if it didn't switch to that branch automatically.

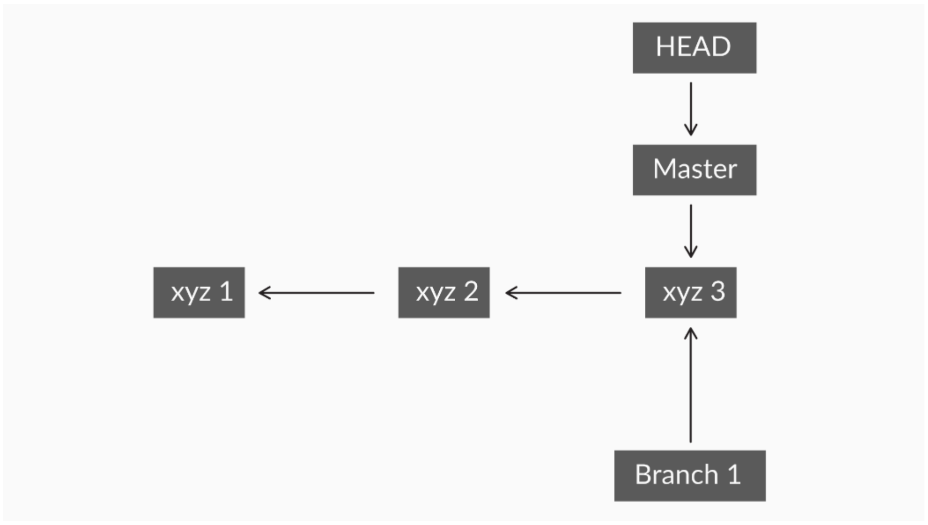


Figure 3: The HEAD points to the branch you're working on

To switch to Branch1, you need to run the command:**git checkout Branch1**
This will now move the HEAD to point to your newly created branch, i.e Branch1

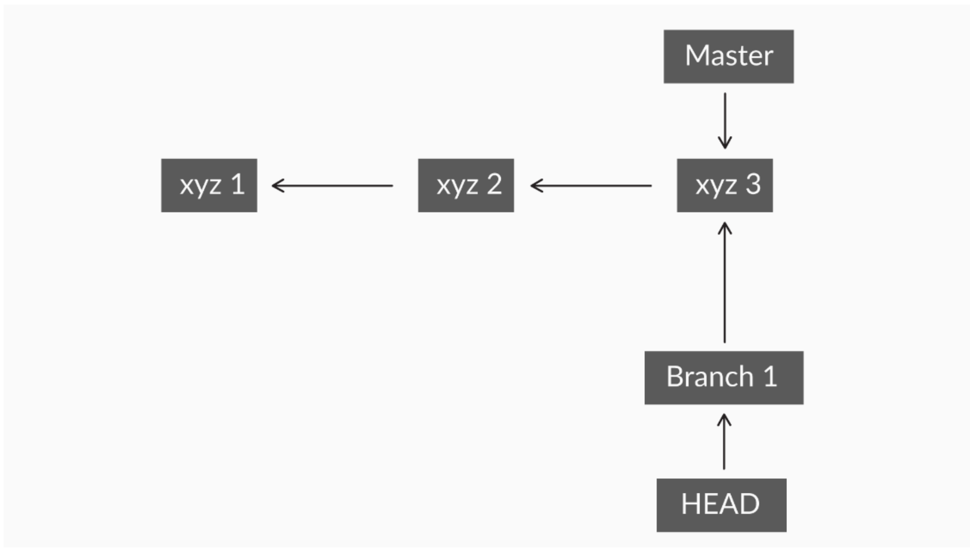


Figure 4: The HEAD points to branch1 when you switch the branches

Working with branches – II

In this video, you learnt about —

- **Merging branches**: For merging, you can use the command **git merge <branchname>**
    - Note: This command will merge the changes in the branch <branchname> with the branch that you are currently working on. Merging can happen between all the branches. Imagine that you have three branches, namely —
        - Master
        - Branch1
        - Branch2

    You can merge any one of the branches above with another one.

- **Deleting branches**: You can use the command **git branch -d <branchname>**

More about the **git diff** command.

The **git diff** command: This command is used to show the changes performed between commits. The main objective of version control is to enable you to work with different versions of the same file. Hence, git provides the command 'diff' to allow you to compare between the different versions of your files. The most common scenario where 'diff' is used would be when you need to see what changes you had made after your last commit. Ways in which we can use the 'git diff' command:

- **git diff commitid1 commitid2**: To see the difference between two commits using their commit IDs
- **git diff branch_name1..branch_name2**: To see the difference between two different branches. Here, 'branch_name1' represents the branch you are currently working on

- **git diff**: This will show you the difference between Git data sources(data sources be commits,branches,files etc).

In the following image, notice that there are two branches connecting multiple boxes:

- The red line represents the master branch
- The grey line represents another branch named 'branch1'
- The rectangular boxes represent the commits and the different commit IDs

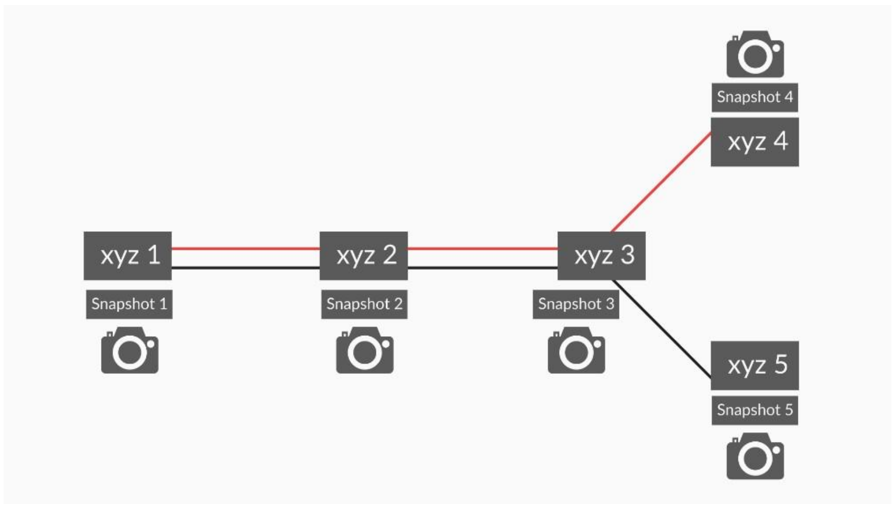


Figure 1: A depiction of git branches and commits

Now, suppose that you need to merge the commit 'xyz 5' on branch1 with the master branch. On merging of the two branches, it creates a new commit. At this point, git will take a snapshot of the changes made. After merging, the new commit would be named 'xyz 6', as shown in the following diagram:

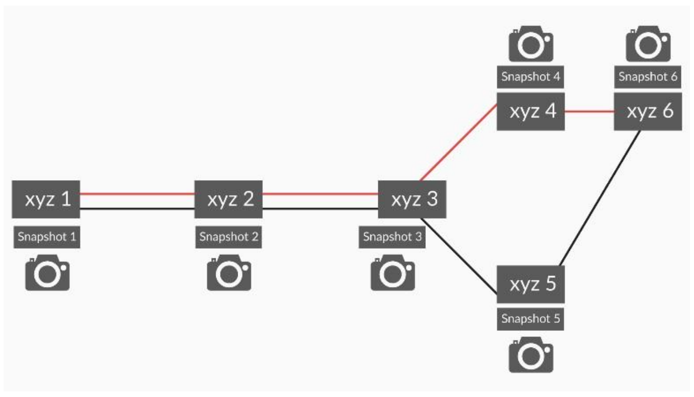


Figure 2: The git branches after merging

Now, once the commit 'xyz 5' is merged, if you want to work on 'branch1', you can keep making commits to it, e.g. commit 'xyz 7'.

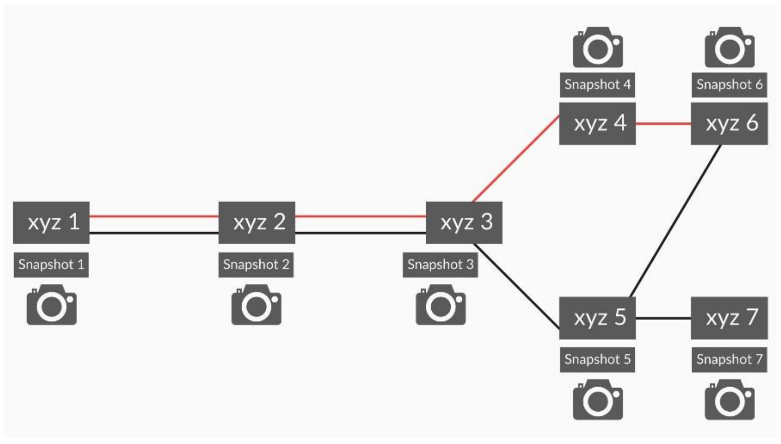
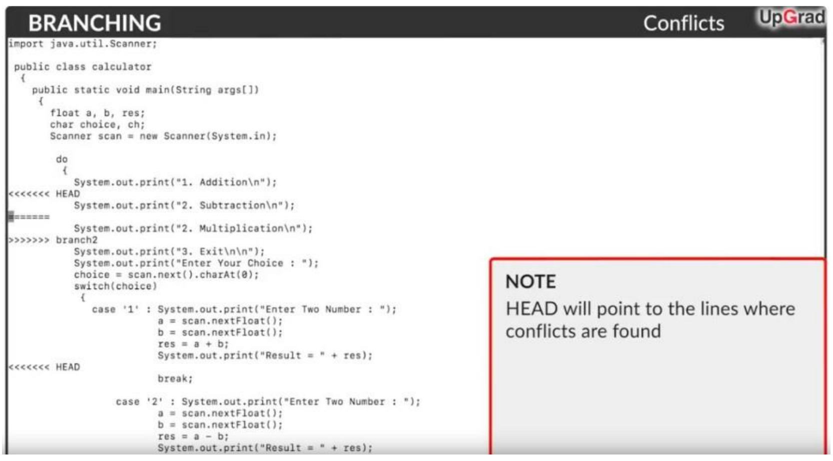


Figure 3: The git branches after the commit 'xyz 7’

Managing Conflicts :

Merging branches in git isn't always as easy as it may look. If two people change the same line of a code at the same time, don’t you think git will get confused?

Definitely! Let’s look at how you can help git in such a situation.



```
import java.util.Scanner;

 public class calculator
  {
    public static void main(String args[])
     {
       float a, b, res;
       char choice, ch;
       Scanner scan = new Scanner(System.in);

        do
         {
           System.out.print("1. Addition\n");
<<<<<<< HEAD
           System.out.print("2. Subtraction\n");
=======
           System.out.print("2. Multiplication\n");
>>>>>>> branch2
           System.out.print("3. Exit\n\n");
           System.out.print("Enter Your Choice : ");
           choice = scan.next().charAt(0);
           switch(choice)
            {
              case '1' : System.out.print("Enter Two Number : ");
                         a = scan.nextFloat();
                         b = scan.nextFloat();
                         res = a + b;
                         System.out.print("Result = " + res);
<<<<<<< HEAD
                         break;

                 case '2' : System.out.print("Enter Two Number : ");
                         a = scan.nextFloat();
                         b = scan.nextFloat();
                         res = a - b;
                         System.out.print("Result = " + res);
```

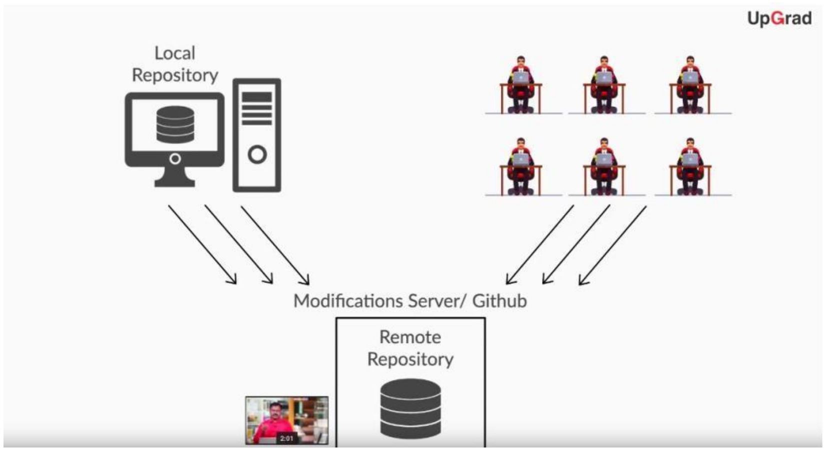
NOTE
HEAD will point to the lines where conflicts are found

The best workaround to deal with a merge conflict is to use your best decision, that is —

- You can keep all the changes by making necessary modifications to the files where the merge conflict happens
- Or, simply discard some changes that cause the merge conflict

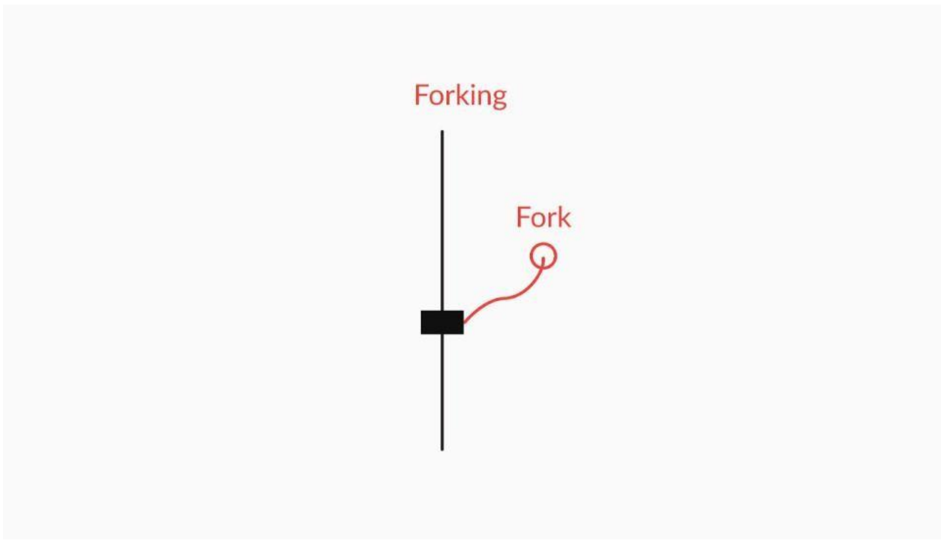
Collaboration :

Collaboration helps you work with the people around you and produce something even better than before. You can add features or improvise some of the features in someone else's project who is sitting miles away from you.

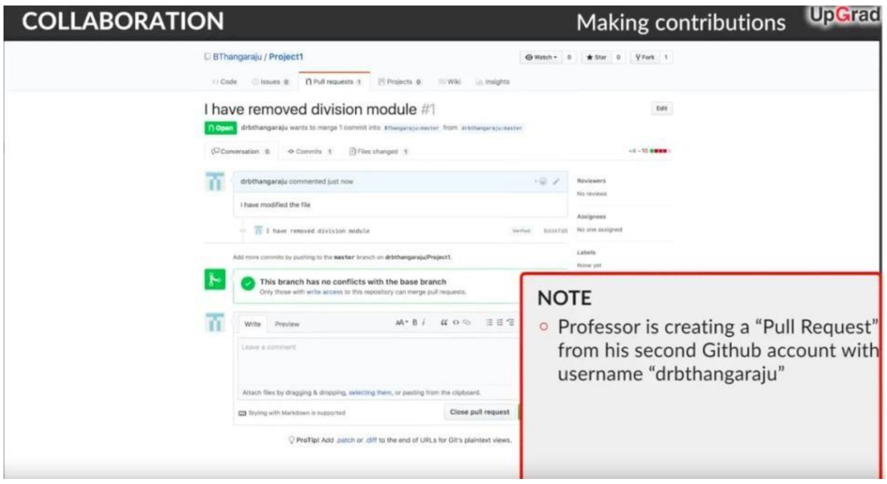


To contribute to someone's project we need to follow the steps below-

- To contribute to someone's project we first need to **fork** it. Forking will get you that project on your github account.



- Now to bring that project to your local system you need to **clone** it.
- Next you will make modifications and push the changes back to your Github account.
- Now to inform the owner of the project about new changes that you made you will create a **pull request**



COLLABORATION

Making contributions

BThangaraju / Project1

I have removed division module #1

Open drbthangaraju wants to merge 1 commit into BThangaraju:master from drbthangaraju:master

drbthangaraju commented just now

I have modified the file

I have removed division module

Add more commits by pushing to the master branch on drbthangaraju/Project1.

This branch has no conflicts with the base branch

Only those with write access to this repository can merge pull requests.

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

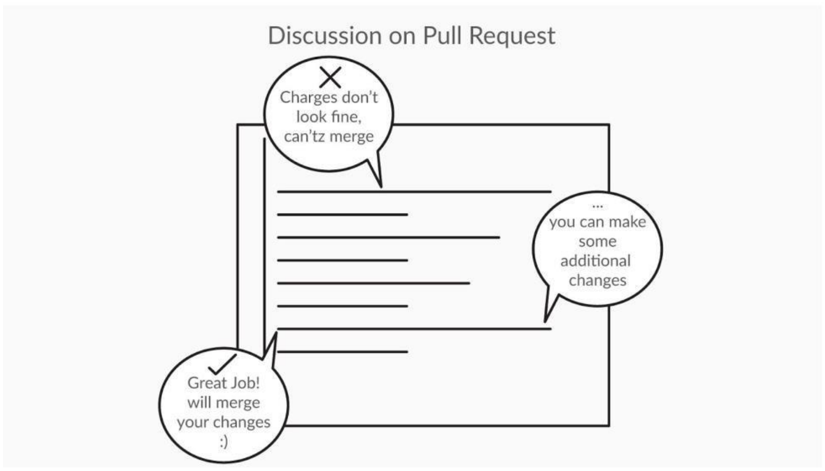
Close pull request

ProTip! Add .patch or .diff to the end of URLs for Git's plaintext views.

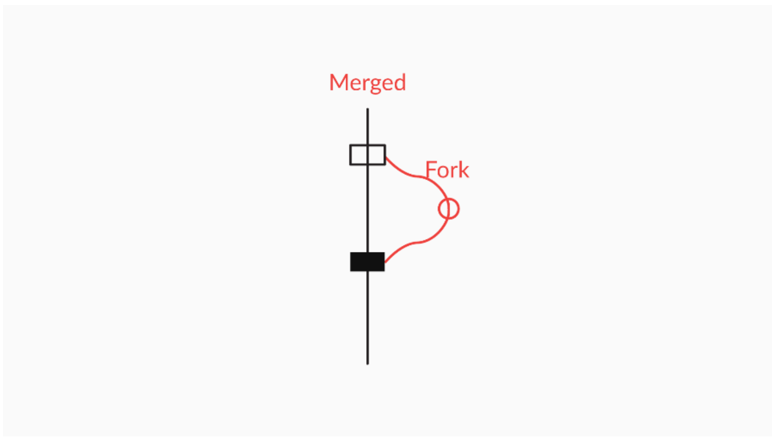
NOTE

- Professor is creating a "Pull Request" from his second Github account with username "drbthangaraju"

- There will be discussion on your changes

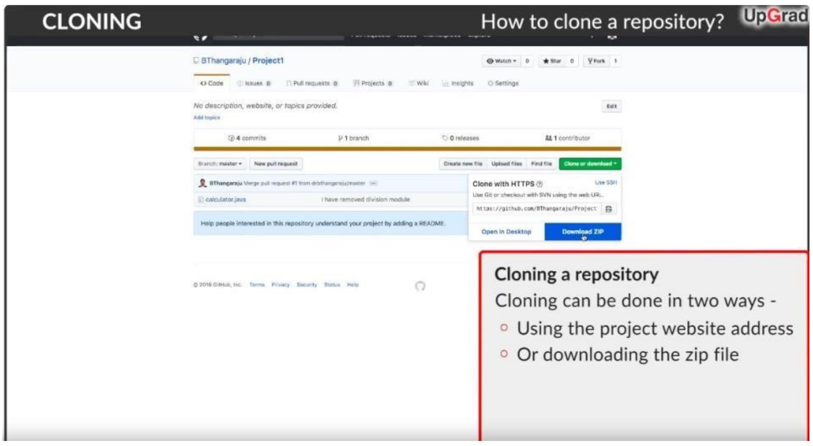

Discussion on Pull Request
Charges don't look fine, can'tz merge
... you can make some additional changes
Great Job! will merge your changes :)

▪



Cloning :

Cloning means making an identical copy. This video will help you clone a github project and bring it to you local machine



Cloning a repository

Cloning can be done in two ways -

- Using the project website address
- Or downloading the zip file

The two ways of cloning are:

- Cloning from your GitHub account
- Cloning from the command line using commands such as —

'git clone url' (of the git repository)