**GIT COMMANDS**

1. git checkout commit\_id - goes to the status of the commit mentioned in commit\_id
2. To get rid of any commit :

git reset --hard commit\_id - all commits **before** commit\_id gets deleted

1. git checkout -- . - To delete or revert the unstaged or uncommited contents from the repository
2. git checkout -b new-feature - Create a new branch called new-feature

(Note : The new branch has all the commits of the master branch)

1. git log - Print all commit and other details
2. Git checkout master - to change to the branch master
3. Git checkout -D new-feature - to delete the branch named new-feature

2 branches - Feature and master

Master m1 m2 m3

Feature F1 F2

git checkout master - **currently in master branch**

1. **git merge feature** - will merge **all** the commits of feature branch to master branch, in order. Eg : the list of commits in master will be f2 , f1, m3, m2, m1.

(Note : incase of any version conflict (2 versions of the same file) , make the change manually , commit the change and that will become the latest commit )

1. **git merge --squash feature** - To just have one commit having the updated content of the feature branch. I.e instead of having all the commits of feature branch, it summarizes it as one commit and stores it in master branch eg: final\_commit, m3,m2,m1

In feature branch, on entering **git rebase master-** It summarizes all the commits of master branch after the common commit between master and feature (here it is M3) into one commit and re initialises the start branch of feature (F1) to the last updated commit of master (m3) eg: f2 f1 m3 m2 m1

Master m1 m2 m3

Feature m1 m2 m3 F1 F2

In master branch, on entering git rebase feature - it does the following : eg: f2 f1 m3 m2 m1

Master m1 m2 m3 F1 F2

Feature m1 m2 m3 F1 F2

<https://www.youtube.com/watch?v=CRlGDDprdOQ>