**Git**

**git init :** to initialize git repository which will create .git file in current folder.  
**git remote add origin <Repo path> :** adding central Repo to local Repo  
**git config --global user.name “user name” :** to configure user name  
**git config --global user.email “email” :** to configure user email  
**git config --list :** list the config  
**git config --global alias.s “status ” :** create alias for status (alias name is ‘s’)  
**git config -- global --unset alias.s :** delete the alias   
**git status :** provide status of the file   
**git add . :** add files to working directory to staging area  
**git add ./\*/-A :** to move more files to staging are  
**git commit –m “commit message” <file anme> :** used to move file from staging area to local repo  
**git commit --amend -m “New message”:** used to rename the commit message  
**git log :** show the commit log , If we commit multiple files at same time but for each commit it will provide one commit ID  
**git log –n :** display only last n number of commit ID  
**git log --since =yy-mm-dd –until=yy-mm-dd :** gives log between given days  **git reset HEAD <filename>** or **git reset <filename>:** used to move file from staging are to working directory  
**git reset HEAD~3 :** Remove the recent 3 commits in the history/ but we cant remove particular commit using reset  
**git reset –soft HEAD~1:** move last 1 commit file from local repo to staging are  
**git reset --mixed HEAD!2 :** move last 2 commit file from local repo to working directory  
 when we reset commit id assigned to that commit will be removed   
**git reset –hard HEAD~1 :**  This wont keep last 2 commit in any working or staged area   
**git clone <gitpath> :** used to clone the remote repo to local repo  
**git branch :** used to list the branches   
**git branch <branchName> :** used to create new branches  
**git branch –m <oldname><newName> :** used to change branch name   
**git checkout <TargetbranchName> :** used to switch to target branch  
**git checkout –b <branchName> :** This will create a branch n control will move to that branch  
**git merge<TargetBranch> :** If data is modified in branch (master1)and need same data in other target branch(master) then go to target branch and use below command .(Video 31434)  
 **git merge master1**// user should be in master branch  
**git branch –d <BranchName> :** used to delete git branch, without checking out from current we can delete the current branch.  
**git push <path of repository> <branchName> :** this will create a branch in given repository and push the committed data  
**git push orgin <branch name> :** similar to above command

When 2 different branch having same file name and different data in file . At this time when we try to merge data between branches conflict will occur.  
Then follow manual procedure to overcome conflict  
Open conflict file keep required data commit and push to repository.   
auto merging will not be preferred because it may erase data in any branch file. (V437)

**Stash Memory :** This is temporary storage area , it will in local area where we will initialize git, We can move data in staging area to stash and we can retrieve it back.

this will be useful when we need to commit files except few files(instead of reset and add we can use stash)  
Stash memory name will create automatically but label name we can give  
**git stash save <lable name> :** used to create stash memory  
**gir show –p <stashName> :** list the data of commit ID or stash memory  
 **git stash –list :** list the stash memory   
**git stash pop :** copy stash{0} data to staging area and delete stash  
**git stash pop stash{x} :** copy stash {x } data to staging area and delete stash   
**git stash apply :** copy stash{0} data to staging area and won’t delete stash  
**git stash apply stash{x}:** copy stash{x} data to staging area and won’t delete stash  
**git stash drop :** delete stash{0} data   
**git stash drop stash{x}:** delete stash{x} data  
**git tag <tagname> :** create tag for new commit (useful during versions release)  
**git tag <tagName> <oldCommit Id>:** create tag for older commit. **git push origin --tag :**  tags will be updated to central Repo   
**git tag -d :** delete tag in local Repo   
**git push origin -d <tagName> :** delete tag in Remote Repo   
**git pull :** this will check the data in local Repo and Central Repo , if any extra data in central reop then it will just sync/ update the local Repo  
**git rebase <branchName> :** This is similar to git merge but it wont create a commitId when will merge the data (copy data from branch to other branch)

**Origin master :** is a remote branch   
**git push –u origin <branchName> :** used to create branch in remote Repo.  
**git branch -r :** list the branches in remote repo.  
**Git fetch <branch> :** this will fetch data from central repo and store it in remote repo(origin master).  
**git pull = git fetch+ git merge**

**Git ignore :** This will hide specified file in working directory  
 create .gitignore file and add the files needs to be ignored  
**git checkout <filename> :**  discard changes in working area   
**git revert <CID> :**  remove all commit of CID and that will create new CID for reverting