**Git DOC**

1. Take back-up of code

$ cp -R project project.bak

1. git init : Initialize directory as git repository

$ cd dir\_name

$ git init

1. git clone: Copy git repository so u can add to it.

$ git clone git://github.com/schacon/simplegit.git

1. git add and git status: Add file contents to staging area
2. first check the status of project folder

$ git status OR

$ git status –s (for short information)

1. Then add files in staging area

$ git add file1 file2 OR

$ git add \* (Will add all files)

1. git diff : Shows diff of what is staged and what is modified but unstaged

$ git diff

$ git diff – - cached (will show you what contents have been staged)

$ git diff Head ( both staged and unstaged changes together)

$ git diff –- stat (don't want full diff o/p, but want more than the git status o/p)

1. git commit : Record snap shot of staging

$ git commit –m ‘ur message’

$ git commit –a (Add file and commit)

1. git reset : Undo changes and commit

$ git reset HEAD (will undo the staged files)

$ git reset – -soft HEAD~ (moves HEAD to specified commit reference, index and staging are untouched) doing same as ***$ git reset - -amend***

$ git reset – hard HEAD ( unstage files AND undo any changes in the working directory since last commit )

1. git rm : Remove files from staging area

$ git rm file\_name (It removes file from staging and from local too)

$ git rm file\_name - – cached (Remove file from staging only).

1. git stash : Add current changes to stack

*Stashing takes the current state of the working directory and index, puts it on a stack for later*

$ git stash (Your modification will lost and will get clean copy)

$ git stash list (will display a queue of current stash items)

$ git stash apply (grab the item from the stash list and apply to current working directory)

$ git stash drop (remove last item from the stash list)

$ git stash clear (remove all items from the stash list)

1. git branch: list, create and manage working context

$ git branch (shows list of branches)

$ git branch testing (Creates new branch)

$ git branch –v (See last commit on each branch)

$ git branch –d (to delete the branch)

1. git checkout: switch between branches

$ git checkout testing (Switched branch from master to testing)

*If we create a branch e.g testing and switch to main branch and add a new file in main branch then the file will be in master branch only. It will not available in testing branch even after switch to the testing branch*

$ git checkout –b branch\_name (create new branch and switch to the branch)

1. git merge : merge branch context in current branch

If we are on master branch and want to merge testing branch in which files has been deleted.

$ git merge testing (so that the file will be deleted from master branch too.)

1. git log: to check what you have done

$ git log

$ git log - - online (will show information in short)

1. git tag : tag a point in history as important

$ git tag –a v1.0

1. git remote : list of git aliases on remote

$ git remote (will return list)

$ git remote add alilas\_name url (will create remote alias)

where url e.g. [*git@github.com:pravinsonawane/GitPractice.git*](mailto:git@github.com:pravinsonawane/GitPractice.git)

$ git remote –v (will show actual url of each alias)

$ git remote rm alias\_name (will remove existing remote alias)

$ git remote rename old\_alias new\_alias

$ git remote set-url alias\_name new\_url (will change remote alias url)

1. git fetch : download new branches and data from a remote repository

$ git fetch (It synchronize local code with remote branches )

1. git pull : fetch from remote repo and merge into current branch

$ git pull

1. git push: Push code changes to the remote rpository

$ git push alias\_name branch\_name

Before push u will have to commit the changes.