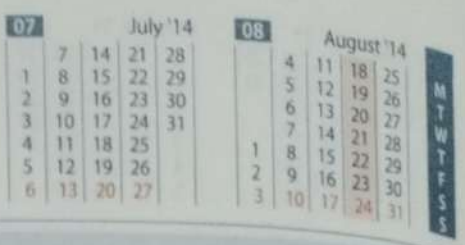
Git

August 2014

23 235-130 Saturday



Important

→ git init

↳ to initialize .git folder which is responsible for storing the history of your project / repository.

→ ls

↳ it is used to list all the files and directories which is present in the current working directory.

→ ls -a

↳ it is used to list all the files and directories along with they are hidden

→ ls .git/

↳ used to open the .git folder/ directory.

→ touch name.txt

↳ used to create empty files.

24 236-129 Sunday

→ git status

↳ used to check the status of all the files/directories or changes in the repository which helps us to know the about the working tree

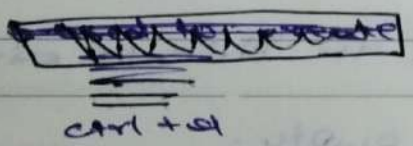
→ git add name.txt →

↳ single file added to the staging area from the unstaging area

Important

→ git add .
↳ used to move all the files/directories or changes to the staging area.

→ git commit -m "message"
↳ used to commit the changes with a message.

→ cat > name.txt


↳ used to create a file and you can write in it, for the first time.

→ cat name.txt
↳ now this will print what the content present inside name.txt file.

→ git restore --staged name.txt
↳ used to re-store all the changes which was moved to the staging area.

→ git clean -f
↳ used to delete the untrack files.

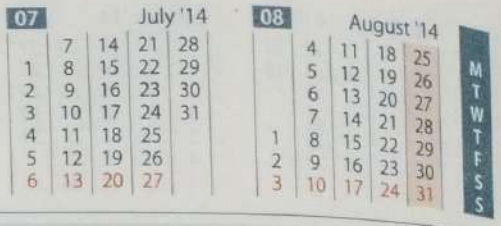
→ git clean -fd
↳ used to delete all files as well as directories.

August 2014

26

238-127

Tuesday



Important

→ git log
↳ used to see all the commit history.

→ rm name.txt
↳ remove single file only and it is remove not delete any directories.

→ rm -rf .
↳ it is used to delete all the files or directory even if they are not empty.

→ git reset hash_commit
↳ used to undo changes in the Git repository. It allows you to reset commits.

→ Git stash
↳ used to cleaning the working tree clean and keep all the track in stash.
↳ used to save temporary uncommit changes.

→ git stash pop
↳ used to restore changes and it will remove from the stash

→ git stash list
↳ used to know the list of stash changes.

Important

→ git stash clear

↳ used to remove all the stashed changes from the stash changes.

→ git remote add origin url

working with urls — adding new url — name of the url, you are adding to be add'

→ git remote -v

↳ list all the urls which are added to a github repository.

→ git push origin main

↳ push all the changes to the origin of branch main. after committing to your working tree.

Open Source Basics

August 2014

28 Thursday

240-125

07 July '14

	7	14	21	28		
1	8	15	22	29		
2	9	16	23	30		
3	10	17	24	31		
4	11	18	25			
5	12	19	26			
6	13	20	27			

08 August '14

	4	11	18	25		
	5	12	19	26		
	6	13	20	27		
	7	14	21	28		
1	8	15	22	29		
2	9	16	23	30		
3	10	17	24	31		

M T W T F S S

Important

1→ Fork to your github.

2→ Clone it from your GitHub

git clone url of repo

3→ git remote add upstream upstream url.

4→ git remote -v

↳ list all the urls which are attached to your repository.

→ git branch lalit

↳ creating new branch

→ git branch

↳ list all the branches which is there in your repository.

→ git checkout lalit

↳ you are changing the branch to lalit

→ git status

↳ return the status of tracked and untracked files

→ git add .

↳ move all the changes of untracked changes to the staging area.

Important

→ git push origin lalit -f

is it will push of your code to your github repositary to the lalit branch forcefully.

Fetch upstream

→ git checkout main
↳ go to the main branch

→ git status
↳ checking the working tree is clean or not

→ git log
↳ it will return all the logs / commit history

→ git fetch --all --prune
Does not merge changes in to your local working branch.

download the latest changes from the origin without merging them into your local branches

if you have multiple remotes eg (origin, upstream) Fetches updates from all remotes instead of just one

if someone delete branch from remote it will exist to your local-remote tracking branch This will remove such stall remote-tracking branch.

August 2014

30 242-123 Saturday

07 July '14

	7	14	21	28		
1	8	15	22	29		
2	9	16	23	30		
3	10	17	24	31		
4	11	18	25			
5	12	19	26			
6	13	20	27			

08 August '14

	4	11	18	25		
	5	12	19	26		
	6	13	20	27		
	7	14	21	28		
1	8	15	22	29		
2	9	16	23	30		
3	10	17	24	31		

M T W T F S S

Important

: → git reset

↳ It is commonly used in undo changes in a git repository.

9

10

→ git pull upstream main →

↓

12 Fetch the changes from remote branch 1 then into the current branch

↳ name of the remote repository from whose the changes are pulled

The branch from which the changes are fetched and merged

2

3

→ git push origin main

↳ It will push your changes to your github repository and you can make pull request from your GitHub to meging your changes from where you forked it.

31 243-122 Sunday

Important

squashing commits

→ git log ↵

≡ ↵ lots of commits as o/p.

→ git rebase -i <u>hashcode of commit log</u>

o/p:

pick ——— File1
s → pick ——— File2
s → pick ——— File3
s → pick ——— File4

<u>Note</u>, change "pick" to "s" which means squash.
It will squash the previous commits eg. in File 1 commit.

How you have to exit from it.

<u>escape</u> : x
Button

→ git reset --hard <u>hashcode of commit log</u>

It delete the commits ~~which~~ before this hashcode log.