GIT

1. It’s a version control system
2. Used for: easily recover files, keeping tracks of files, who introduced an issue and when, roll back to previously working state

* Version control system(VCS):

Is a software that helps developers to work together and maintain complete history of their work.

* Functions of VCS:

1. Allows developers to work together
2. Does not allow overwriting each other’s changes
3. Maintains a history of every version

* Advantages of git:

1. Free and open source
2. Fast and small
3. Implicit backup:

Chances of losing data are very rare when there are multiple copies of it. Data present on any client side mirrors the repository, hence it can be used in the event of a crash or disk corruption.

1. Security: git uses function 🡪 SHA1(Secure hash function) to name and identify objects within the database
2. No need of powerful hardware:
3. Easier branching:

Branch🡪 copy of a code

Tag🡪 meaningful name of your branch

Clone🡪setup code at your local system

Stage/index🡪snapshot of your changes

Head🡪latest commit in git

* History of vcs:

1. Local vcs🡪

use of DB (own local computer) to keep track of files

Pros🡪 can track files and roll back, no need to use internet for keeping track of files

Cons🡪if u lose your hard disc everything will be lost, no collaboration of data (manually has to be done)

1. Centralized vcs🡪

From a centralized server employees will pull files, will work on them and pushed back to centralized server. So that number of emp s can work on project and there will be no problem if anyone’s laptop get crashed/damaged…

BUT what if centralized server gets crashed…..???

No major problem because there will be some data in emp s laptops but we won’t get old versions

1. Distributed vcs 🡪

There will be a complete backup of a data on centralized server in each and every emps laptop. (*Sabke pass sub kuch)*

* HOW GIT FORMED:

Linux Torvald was using ‘Bitkeeper vcs’ for keeping track of linux kernel updates. One day company who was providing Bitkeeper denied to provide service free of cost. So Linux Torvald decided to designed his own vcs and he did 🡪 GIT

* Difference between GIT & GITHUB:

GITHUB is a hosting service (website) which holds GIT repositories (all version, all updates, Patches)

GIT is a free softwere.

* Features of GIT:

1. fast, reliable

2. it stores snapshots and not differences🡪

It will create a folder (.git) which stores all snapshots i.e total history of our project. We can recover any file any snapshot from this folder easily.

3. almost every operation is local🡪

All work will get done on a local pc and at the end we pushed it into centralized repository.

4. git has integrity->

You can set a checksum to your file. If anyone (hacker) changed anything in your files checksum will get change. While sending data to your friend you tell him that checksum so that he can compare and understand whether it’s a same file or anyone has changed. It might be possible that somebody has put virus in that file.

5. GIT generally only adds data🡪

Generally we go on adding data and do not delete.

* How to install git:

1. open browser-🡪type git install🡪 git scm.com🡪 download for windows🡪

2. next-next-next

* After installing:

1. create a new folder🡪 right click🡪 click on ‘git bash here’🡪git terminal will be open

2. to set username🡪 $ git config --global user.name "Shubham"

3. to set email id🡪

$ git config --global user.email "Shubhamthopate9960@gmail.com"

4. to check username🡪$ git config user.name

5. to check email id 🡪 $ git config user.email

\*note🡪 to zoom in zoom out use 🡪 ‘ctrl –‘ & ‘ ctrl +’

* Git – Three stage architecture:

1. working directory

2. staging area

3. git directory(repository)

Suppose you have a version1(c1) which contains files🡪 index.html, index.css, engine.js

For version2(c2) you made changes in all these files. At the end you came to know that ‘engine.js’ is not ok to commit.

That is what we do in staging area—staging of files which we are going to commit.

Before committing files in repository its best practice ‘to stage files’

* PROJECT 1:

Among multiple files in folder I want to move certain files to the repository.

1. Create a folder “project”🡪 create a text file 🡪 to indicate extensions also 🡪view-options-view- uncheck ‘hide extensions for known files’

2.create multiple files🡪 .txt, excel file, db file

3. right click 🡪git bash here

4. run command🡪 $ git status

Error will come---fatal: not a git repository (or any of the parent directories): .git

5. $ git init-------- to initialize new git repository

6. $ git status---------- check once status, files will be red

7. $ git add --a--------------to add all files in staging area

\*note: instead of ‘git add --a’ u can write ‘git add a .’

8. $ git status------------ files became green

9. $ git commit –m “Initial commit”--------to add files to repository

10. $ git status----------- nothing to commit, working tree clean

11. $ git log--------- shubham Initial commit

12. make changes in first.txt

13. $ git status------------------modified first.txt

14. make changes in excel---anything

15.$ git status----------2 files modified

16. now if I want to move only first.txt to repository and not excel file-----

$ git add first.txt

17. $ git status------------only one red file and one green(staged)

18. $ git commit –m “changed first.txt and added better design”

19. $ git status----------only one red file

20. moved excel file also with message “this is modified my excel file”

21. $ git status---------all clean

22. $ git log-----------all messages that we commited

* To remove .git folder along with its content…. To finish tracking of files in that folder………

$ rm –rf .git

* Cloning a remote Git repository from Github

1. Open “GIthub website tensorflow” in browser

2. Go to website of Tensorflow repository

3. Click “clone or download” and copy that URL

4. Go to folder and open git

$ git clone URL-----------Full repository of tensorflow will get cloned

Now you can edit any file you want from that repository,

cd to tensorflow and commit edited file

5. $ git log--------to see whatever changes get done with author

* FILE STATUS LIFECYCLE:

1.Suppose you have a folder(F) which contains files---- f1, f2 ,f3 (UNTRACKED)

2. F is your repository

3. Now when you run a command--------

$ git add –a

f1, f2 ,f3 ----------------(UNMODIFIED)

Here files are not committed yet….. TRACKING started from here

4. now if you change f1,f2 -----------(MODIFIED)

Files became modified

FILE STATUS LIFECYCLE:

UNTRACKED-🡪UNMODIFIED🡪MODIFIED🡪STAGED

DEMO STEPS:

1. $ git status

2. $ git init

3. $ git status--------- UNTRACKED FILES (red)

4. $ git add –a-------------start tracking (UNMODIFIED)

5. $ git status---------------(green)

6. $ do some changes in txt file-------------(MODIFIED)

Now if you run commit command unmodified file will get committed

7. $ git add first.txt-------------this file will be in staging(before modified) and in modified also..----------------------(STAGING)

Now if you run commited command modified file will get committed

* TO IGNORE FILES/FOLDERS IN WORKING DIRECTORY:

Suppose you have files/folders in your working directory that you don’t want to track. To ignore them we have commands🡪

Lets understand this by following example:

STEPS:

1. create error.log file in your directory which you want to avoid.

Add some data to that file

2.$ git status-------- error.log (red),untracked

3.$ touch .gitignore--------- in this file give name of file that you want to ignore

4. open .gitignore file and give name of file that you want to ignore, save and close it

5.$ git status-------- error.log vanished

6.$ git add –a----------to staged .gitignore

7. $ git status

8. $ git commit –m “added gitignore”

9. now if ever you make changes in error.log file it won’t shown in modified files………it gets ignore

10.NOW IF U HAVE MULTIPLE FILES (.log) THAT YOU WANT TO IGNORE THEN…….

11. Suppose u have files errorcopy1.log,2 ,3, 4 , …..

12. just open .gitignore file and add text🡪 ‘\*.log’

This causes any file having extension .log will get ignored

13. Now what if you want to ignore a folder…….???

14. create one folder and add some files to it

15. now to ignore that folder just add name of that folder into file🡪 .gitignore

16. $ git status-------- folder gets ignored

\*note🡪 1.blank folders gets by default ignored.

2.If your folder contents files which gets ignored(.log) then also it will get ignored automatically

3.once you add name of folder(with / next to name) in .gitignore file----- if u create multiple folders with same name they will get ignored

4. if you want to ignore folders with same name which are outside(shubham) only but you want to track folders which are inside(aa/shubham) then in .gitignore file add🡪 /shubham/

* Suppose you have some files in staging area and you modified one of them then that file will be shown at both🡪 staging and modified area(red and green)

1. now if you run command🡪 $ git diff

This will show you comparison between those two files(staged and modified) at staging area and working directory

* $ git diff –staged

It displays your comparison between your last commit and latest staging area

Red was previous and green is latest/updated

* SKIPPING STAGING AREA:

Suppose u have multiple files in .git directory(i.e tracked files)…….. all clean

And you modified one of them…. Then according to regular procedure you have to add it to staging area and then commit

BUT by using following command you can directly commit the file🡪 git commit –a –m “direct commit”

\*note: for newly created/ untracked file this command won’t work… for that you have to add it first

* MOVING/RENAMING/DELETING FILES IN GIT:

1. Deleting:

When you delete any file manually from the folder, you have to add and commit manually in git BUT

When you run command $ rm file.txt-----

File will get deleted automatically and staging in git will also get done automatically.

2. Renaming:

Suppose you have a file named🡪 shubham.txt

To rename it to amol.txt run following command🡪

$ git mv shubham.txt amol.txt

\*note:

If u add a name of file in ‘.gitignore’ that you are already tracking

Then if you modify that file it shall not be shown in git status….BUT

In reality it gets shown in git status…….

In such a condition you have to tell git not to track those files explicitly by following commands🡺

$ git rm –cached filename.txt-----------it won’t remove file, just stops tracking



Suppose you clone any(mypanda) repository in existing repository by cloning command then it becomes separate git repository

Now just close existing git repository and open new (mypanda) folder and git bash there….

1.$ git status------------clear

2.$ git log---------------all commits

3.$ git log –p-------------diff(whats removed and whats added)

4.$ q-------------------------quit

5.$ git log –p -3----------------only 3 commits will shown

6.$ git log –stat-----------whats actually happened in commit in short

7.$ git log --pretty=oneline-------- whats actually happened in commit in single line

8.$ q-------------------quit

9.$ git log --pretty=short------------ whats actually happened in commit in short(author,message,commit)

10.$ git log --pretty=full---------

11.$ git log --since=2.days---------last 2 days commits only

12.$ git log --since=2.months

13. $ git log --since=2.weeks

14. $ git log --since=2.years

Open🡺’git scm useful options for git log format’ in browser

Open🡪git log documentation

Go down to pretty options------

15. $ git log --pretty=format: “%h -- %an”----- an stands for author name

* UNSTAGING OF FILE:

Suppose u have file that u have staged by git add command and now u want to unstage it. For that just run following command:

$ git restore --staged FILEname.ext

* RECOVERY OF LOST DATA:

Suppose u have a file containing some information. By mistake u delete that information and update it with some other information. But now u want ur previous information back. For that just run following command:

$ git checkout – FILEname

\*note: This command works only if files are not staged

For multiple files to get previous data:

$ git checkout -f

* WORKING WITH REMOTE REPOSITORIES:

Suppose u have a repository stored on a server. You can push or pull data from that servers repository by following steps:

1. go to browser and open🡪 github and create an account

2.create repository over there

3.go to your local folder where u have created ur repository

4.open gitbash by clicking right click

\*remote: website/repository that we have created on internet

5. since u already have created ur repository run command:

$ git remote add origin [git@github.com:---------------------------](mailto:git@github.com:---------------------------)

The website in which we have stored our repository is called here:

Origin-----------which is remote for our repository

6. $ git remote

Origin

7.$ git remote –v--------------2 url s will be shown for fetching and pushing

8. $ git push –u origin master

Error: no permission

Then how to get permission----???

9.account—> settings🡪ssh and gpg keys🡪new ssh key🡪 title: my pc🡪

10.open browser: ssh keys github🡪 open website—generating new ssh key….--> copy a command to generate ssh key and paste it in ur gitbash and substitute that with urr own email id🡪

11. go to ur gitbash and run second command🡪

$ eval $(ssh-agent -s)------------ just copy paste

o/p shall be same(nearly) as shown over there in browser

12. second command to add ssh private key to ssh agent

$ ssh-add ~/.ssh/id\_rsa

Identity added---------

13.$ tail ~/.ssh/id\_rsa.pub

14. copy key and paste in 9 🡪add ssh key

15. $ git push –u origin master

It shall push

16. check 🡪 go to github tab🡪home🡪click on ur repository

17. add any new file into ur repository and commit

18.$ git push –u origin master

Newly added file shall be visible into github tab

* SETTING ALIAS IN GIT :

Instead of writing bigger commands u can write short one. Lets c how to do it:

A) Suppose u want to write ‘git st’ instead of ‘git status’

1.$ git config --global alias.st status

2.$ git st

B) Suppose u want to write ‘git ci’ instead of ‘git commit’

1.$ git config --global alias.ci commit

2.$ git ci

c) Suppose u want to write ‘git unstage ’ instead of ‘git restore --staged-- ’

1.$ git config --global alias.unstage ‘restore --staged --’

2.$git unstage filename.ext

D) to see last commit

$ git log -p -1

1.$ git config --global alias.last ‘log -p -1’

2.$ git last