GIT

(Java code camp
w3schools.com
freecode camp)

Keywords :-

1. VCS → work on project to manage source code in effective manner. by securing
2. versioning → To rollback to previous & other versions easily as per requirements. Also to check modifications.
3. opensource, distributed, fast.
4. Can work locally.
5. updated vcs of SVN, CVS ....
6.

operations for development :-

1. GITHUB (or) BITBUCKET.
   signup, add repositories, remember username
   remote
2. create repository. → name → add README.me
   → create.
3. GIT on local systems :-
   a. yum install -y git
   b. git clone ——— .git

classmate

18/8/20

~~Kubernetes workshop (DEV)~~

(c) cd project ; vim newfile ; echo " "

(4) Stages of GIT :- (ADD, STATUS, COMMIT, PUSH)

working area → creating file

Index / Staging area → git add file

git status

local repository → git commit -m " "
message

remote repository / machine → git push

verify the remote repository

classmate

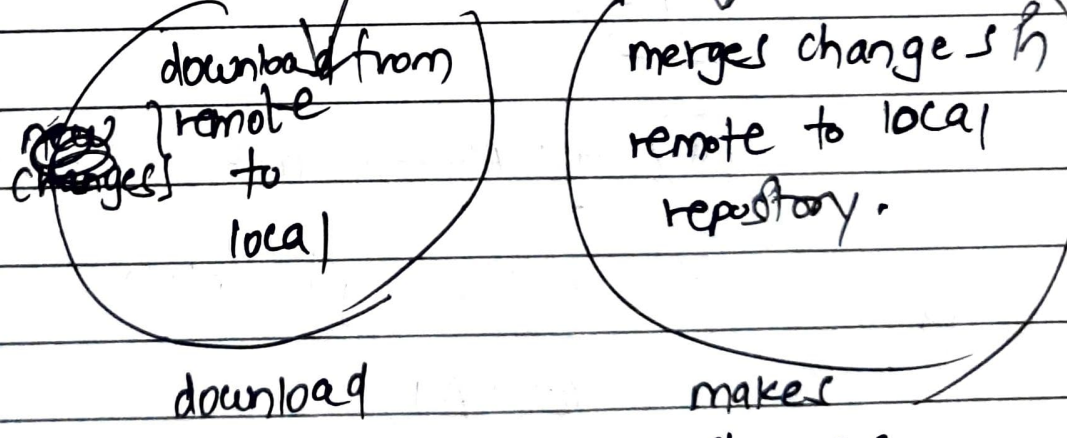
git rm file → commit ✓

(5) GIT PULL → When multiple developers are working on same project & other developers commited changes to the remote repository & we need to get those into the local machine, then we use git pull.

(a) Remote repo → make changes by adding code to file.

(b) git pull origin master → to update the changes made on remote repo on local repo.

(c) git pull = git fetch + git merge



(6) origin → origin points to a remote location.

git commit --ammend

```
git remote -V
git remote add hari ____ .git ____
git remote -V
git pul hari master
```

(7) GIT FETCH :-

git feetch gets changes from remote to local without merging with local repo.

Example :-

(a) GITLAB → make changes to file

(b) Local → git fetch.

CAT

(No changes as fetch only pulls file from remote to local)

(c) git merge

cat file

(changes will be happening)

pull
git fetch
+
git merge

classmate

git branch -av → all branches,

git tag <tagname> → tag to current commit.

(8) GIT HEAD & GIT BRANCH

GIT HEAD → points to the recent commits. (history)

GIT LOG → commit history of current branch.

HEAD → moves to the latest commit

ex :-> (a) git branch (shows all branches)

(b) git branch new-branch

(c) git checkout new-branch (switches branch)

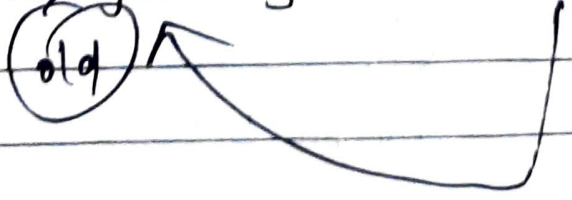
(d) git log (check head)

(e) git checkout old-branch. (now)
git log (check head)

(f) changes from old to new branch

→ git branch

→ git merge new branch.



in local repo

git log --pretty=oneline

git log --stat

git log -p file

(g) remove (or) delete source branch :-

git branch -d new-branch

↓ delete.

git log → to check deletion.

git branch -D new-branch-2

→ force delete even if it contain any file (or) data.

(h) create local branch & push into remote git :-

```
git branch task-1
git checkout
echo "Hi" > index.html
git add .
git commit -m "__"
git push origin task-1
```

↓

pushes branch from local to remote

(9) GIT PULL REQUEST :-

a. git branch
b. git branch task-2
c. git checkout ↓
d. vi " " → git add, git commit git push (origin task-2)
e. Go to github → pull requests
   → soure → task-2 dest → master
   → create pull request (check side by side diff)
   → Merge (any other user code)
   (can also delete source branch).

git pull origin branch-1
↓ remote.
git push origin branch-1

classmate

(11) GIT - Reverting changes:

(a) undo changes in working area (before add)

vim file → git status → git checkout file (or).

→ git status

actual state.

(b) unstage (remove changes made with git add).

```
git reset HEAD <file>
```
→ (staged file)

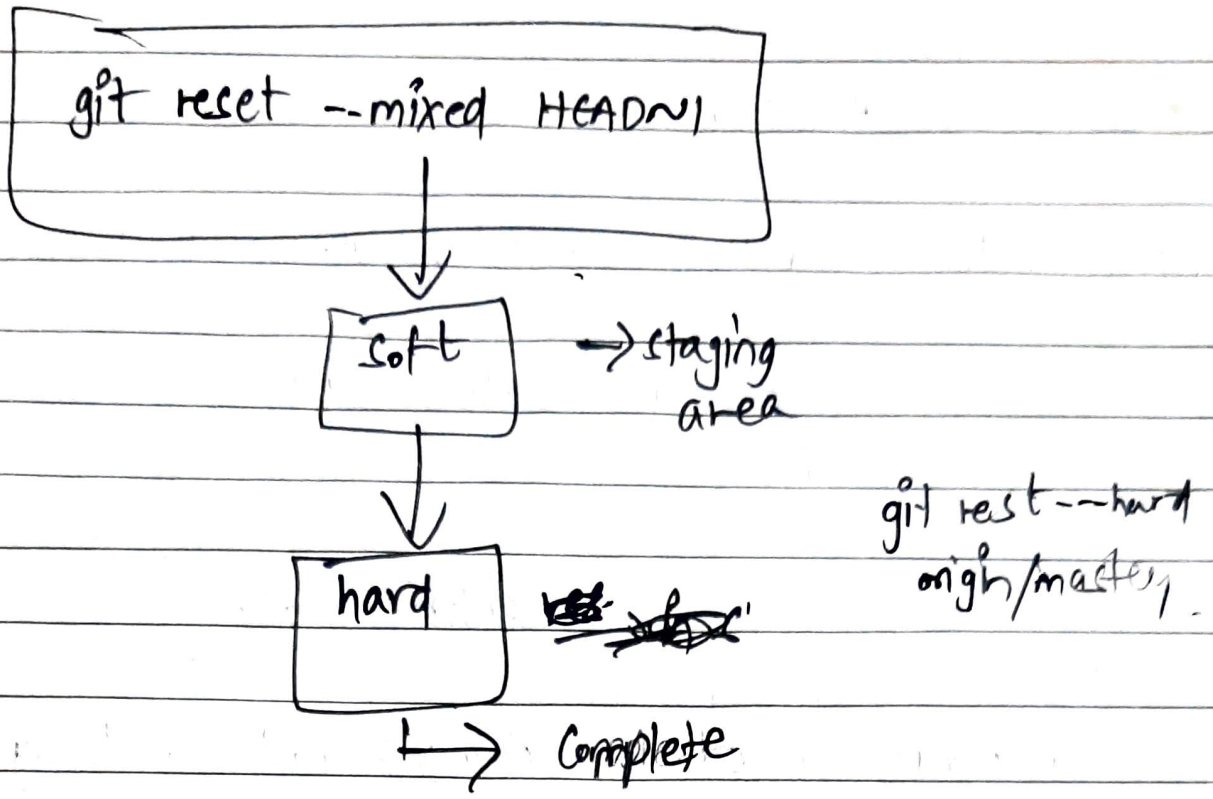
(c) After commit:

```
git reset HEAD~1
```
→ use when commit is present in local but not repo.

```
git reset HEAD~3
```
→ to use after changes made in remote repo

```
git reset --help
```

classmate



(9) git revert :-

git revert id

reverts back to previous stage by changing the commit.

(2) GIT CONFICTS (imp)

check laptop files for pdf

(13) GIT Cherry :-

```
git cherry-pic commit-id
```

(a) create two branches
→ master & feature.

(b) git checkout master
echo "one" > one.sh
git add.
git commit -m "one"

(c) git branch feature.
echo "two" > two.sh
git add., git commit.
(similarly create another "three.sh")

(d) Now if we want to move commit from one branch to another :-

git checkout master.
git log. --stat (in feature branch)
(copy the commit id we want)

```
git cherry-pick commit id's
```

CLASSMATE

git show

(14) How to revert to previous versions of file in GIT.

(a) git checkout master.

(b) echo "Hi" > one.sh.
git add, commit.

(c) Again ~~add~~ ~~three~~ lines for three time & git add along with commit.

(d) Now, git log --stat

git show commit-1 : one.sh
git show commit-2 : one.sh
git show commit-3 : one.sh

commit; 1, 2, 3 are commit ids (check by commit message)

(e) git checkout commit-id to change the file content.

But don't use frequently (causes issues)

classmate

(15) **GIT BLAME**

show the details of the file.

(a) suppose we created a file teams and commited three times with,

sachin
dhoni
zaheer

as batsman
wicket-keeper
bowler

as commit -m " "

then,
to check the details of the file, we use git blame.

(2) git blame teams

gives all the details of the file.

git show commit numb : teams

gives the changes made

classmate

(15) # GIT BLAME

show the details of the file.

(a) suppose we created a file teams and commited three times with,

sachin, dhoni, zaheer } as batsman, wicket-keeper, bowler

↓

as commit -m " "

then,
to check the details of the file, we use git blame.

(2)

```
git blame teams
```

↳ gives all the details of the file.

```
git show commitnumb : teams
```

↳ (gives the changes made)

(3) `git blame -L 1,+3 team`

↓

to see three changes.

`--shee=1.week`

↓

all history past week.

Botogoboto

(1) Hard / soft reset

(2) fast-forward merge with commit.

`git merge --no-ff branch-2`

`git log --graph--onefine`

GIT Freecodecamp :-

`git pull -u origin branch`

↓

primary (makes changes remote)

→ add (skipping add module)

git -am " "
commit

git diff branch (or) git diff

git reset

git reset HEAD~1

commit

staging

git reset commit id

--hard → removes completely///

(14) GIT STASH (Academind).

git stash

git stash apply

git stash list & git stash apply @2

git stash push -m "comment" & git stash list

git stash drop 3

git stash pop 4 → will be added (apply) & stash will be deleted,

git stash clear → clear all stash.