Git Tut →

setting up →

① sudo apt-get install git.

② git --version

③ git config --global username "uname"

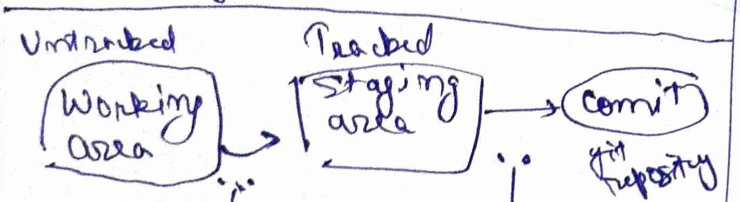
④ git config --global email "email"

⑤ show ~~[illegible]~~ config →

git config -l

Initializing git repository

: git init



[: git add .]

[: git commit -m "message"]

~~Imp~~ →

Working area :→ where you add code and files.

Staging area :→ When you are done adding files (save it as draft).

commit (save) :→ Final Screenshot of version of control.

[: git status] { [show tracked and untracked files].

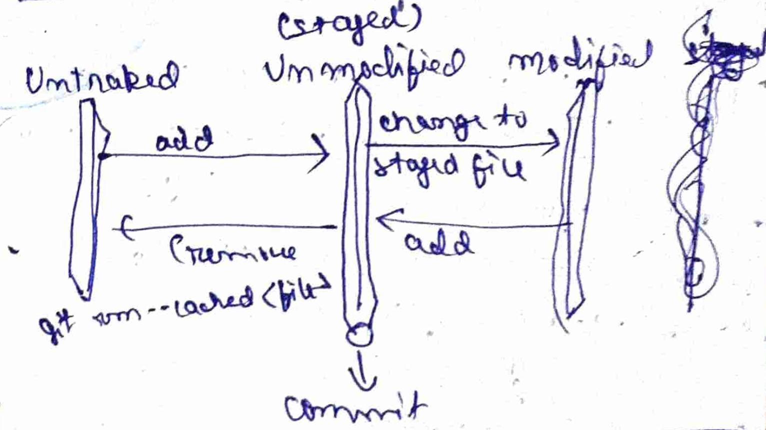
: [git log] [show commit history].

⑤ [git clone url] < git name

[will clone the URL, automatically call 'git init', so, need to call git init.

→ It will also show previous commits.]

⑥ File status lifecycle :→



⑦ [.gitignore] :→

files ~~inside~~ name inside it will ~~be~~ ~~not~~ be ignore.

Uses → Error log generated should be ignored so, place the name of its in .gitignore

⑧ git diff

→ Compare staging area and working directory.

```
git diff --staged
```

compare staging area with last commit

```
git diff --cached <filename>
```

within staging area

⑨ Adding file directly to staging area

```
git commit -a -m "Direct commit"
```

-a for adding all Tracked ~~staged~~ filed to commit.

For untracked file ⇒ Track it first.

⑩ Renaming file →

```
: git mv <present-name> <new-name>
```

~~git~~ Removing a file

```
: git rm <present-file>
```

These two will directly ~~affect~~ shows in staging area

Putting file from Tracked to untracked

~~put those file in .gitignore~~

```
: git rm ~~cached~~
      --cached <file-name>
```

⑪ git log →

```
git log -p
```

:→ commits with diff

```
git log -p -n
```

:→ last n commits with diff

```
git ~~diff~~ --stat
```

[give short description of each commit

```
git log --pretty=oneline
```

[all commits with one line description

```
git log --pretty=short
```

[short with only Author name.

```
git log --pretty= ~~short~~ long
```

[commit with Author and commiter]

```
git log --since=2.days
                  " months
                  " years
```

```
git log --pretty=format:
        "%h -- %an"
```

hash

author name

```
git commit --ammend
```

add change to previous commit.

⑫ Unstage a file

```
→ git rm --cached <file>
→ git restore --staged <file>
  git reset HEAD -- <file>
```

rm vs restore

↓

```
git rm --cached <File>
```

It will delete the file from staging area and make it untracked.

⇓

if commit ⇒ file will not be contained

staging area contains file which should be commited

⇓

Clear

```
git restore --staged <File>
```

⇓

It will match the file from the last commit

⇓

and it does not delete the file from staging area

⇓

However, if last commit does not have the file, ⇒ It will delete the file ⇒ equivalent to git rm in this case.

If the last commit had the file ⇒ it will make it modified file.

⇓

if commit ⇒ file will contain its earlier version.

→ similar to

```
git reset HEAD -- <file>
```

modify the file to previous commit →

```
: git checkout -- <File>
```

modify all files to previous commit

```
: git checkout -- .
```

13 Pushing a file :-

After git clone →

```
: git remote add origin <link>
```

origin is the name of link.

Now press →

```
: git remote -v
```

for seeing origin.

Now,

```
: git push -u origin master
```

for pushing the changes.

Pulling the file →

```
: git pull origin master
```

How to get to previous commit

→

```
: git revert -m <hash>
```

git ~~reset~~ reset →

```
git reset --hard <hash>
```

⇒ destructive

(14) Branching

~~Create~~ Create a branch →

: git branch <Branch Name>

List all branch: →

: git branch , git branch -v
for all branch + last commit

git branch --merged
all merged Brand

Switch Branch →

: git checkout <hash code>

Create and switch branch

: git branch -b <Branch-Name>

--no-merged
not merged Brand

git branch -m <Branch>
change name of current Branch

Merging Brach

Switch to parent branch and →

: git merge <Branch-Name> → Adds commit

Note :→ ① gitignore files will unaffected By switching branch.

② Any uncommit changes [from untracked files and staging area] will be transfered across branch as it is.

③ Merging can lead to conflicts [VScode].

Resolving merge issue (Not supported ~~in linux~~)

: git mergetool --tool=~~emerge~~ emerge

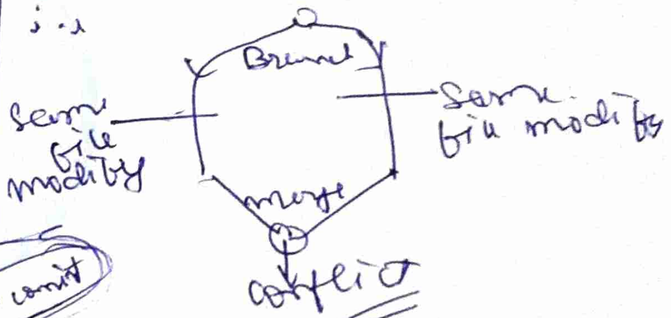
⇒ git mergetool --tool=~~vim~~ vimdiff

emerge →

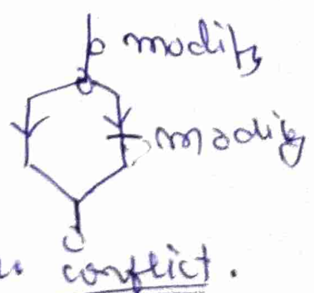
n ~~xx~~ → next
a :→ version a
b :→ version b
q → quit

Conflict only arises ~~as~~ if any two Branch trys to modify same line in code.

i.e



● In situation like this

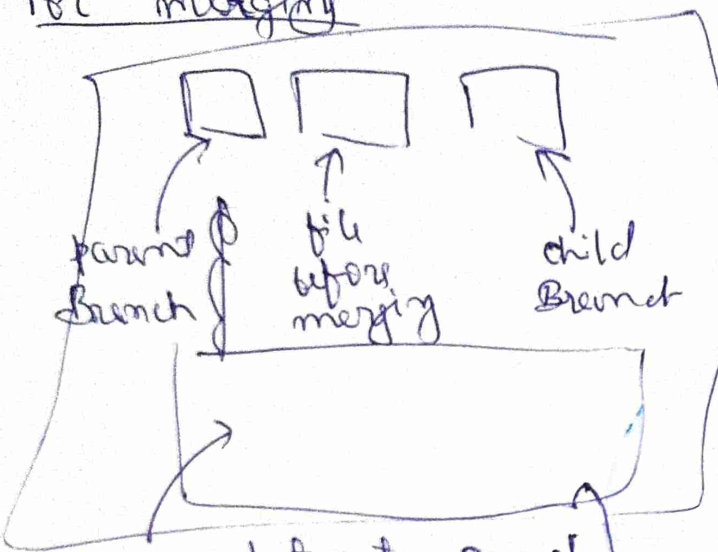


git merge --abort
to abort merging.

vimdiff

vimdiff file1 file2
⇒ Very nice diff.

Vimdiff →
Should be used inplace of diff for merging



This is what gets saved. Edit this.

→ Delete .orig file after done merging.

Deleting Branch → (merged)

```
: git branch -d <branch-name>
```

Deleting (unmerged branch)

```
: git branch -D <Branch-name>
```

Branching workflow
↙ Long Running Branch ⇒ Will remain infinitely
↘ Topic Branch ⇒ for specific topic

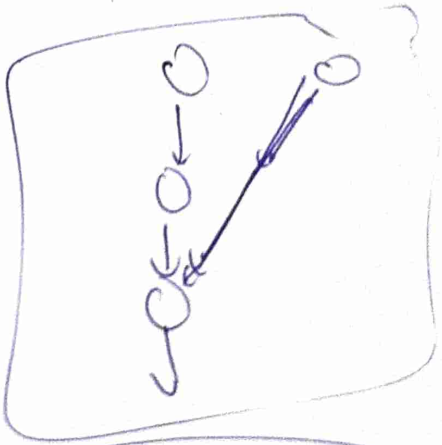
Note :→
① While Pushing, remain in that Branch which you need to push.

② While switching → commit first.

Deleting Branch on Remote

```
: git push -d origin <Branch>
```

Commits in git are Linked List



git branch --merged will give all accessible branches from that Branch.

: git clone → will bring all Branches you have to do checkout in that Branch.