GIT & GITHUB

- Interesting tool of Software Development.

git & github	
↓	↓
Version control (software)	Service Web based Hosting.

- Version Control System :→ Track files for changes

C:\Program Files\Git

To check git version (cmd): git --v.

- github hosts the git software.

जिसने game खेला है। उसे Git acche se पता होगा।

It creates a check Points.

VCS (Creates a checkpoints)

Software is Permanent But आज Github famous hai Kal Ko GitLab famous Ho sakta Hai.

1. git -- version.
2. pwd (present working directory)

$clear: To clear the git Bash

PAGE NO.:

Folder/Directories/Repository: Is one and the same

③ git Status.

Configuration Settings

```
git config --global user.email "Your email"
git config --global user.name "Your name"

git config --list
```

It saves the information when you push something.

Always use Double Quotes to prevent mistakes.

Note: `git status` provides information about current state of your working directory

Create Repository

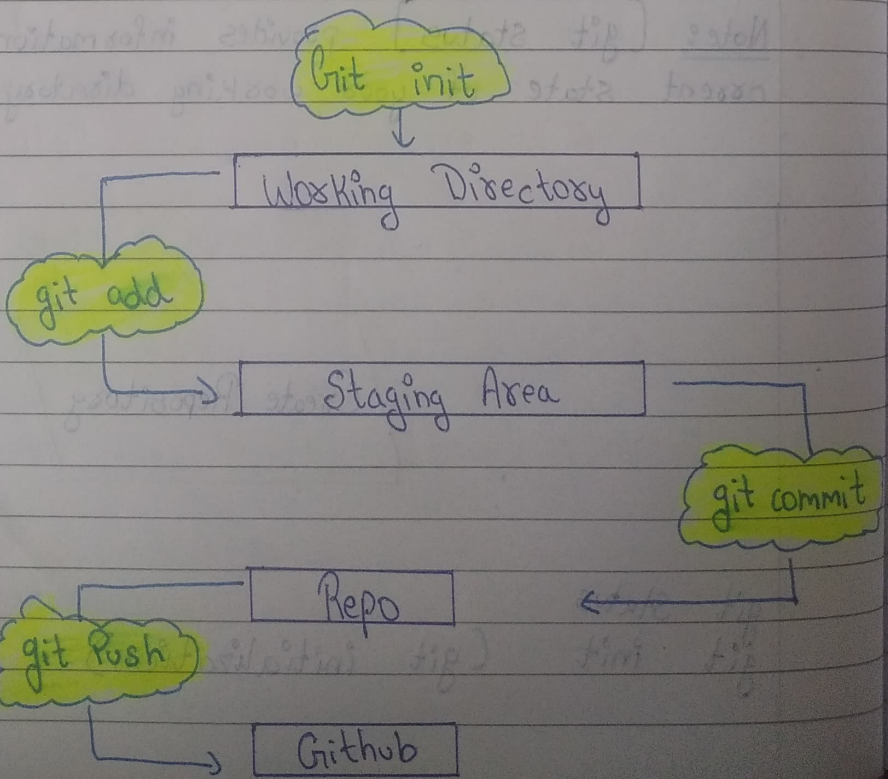
git Status
git init (git initialization)

Study (hint :)

- ls command (git): It will list all the current directory ka contents.
- ls -la (commands)
- (.git) is a hidden folder
- (1) git config --global init.defaultBranch <name>

Summary of Git

1. Mainly used to create checkpoints
2. Collaborative tool.



PAGE NO.:

Step 01:

git add . // will add all the files.

OR

git add <filename> <filename>

<eg:->) git add file1.txt file2.txt.

Step 02:

Nowe we need to pass a commit message

```
git commit -m "message".
```

Q.2 Suppose after commiting we have made certain changes to a file. let's say file1.txt so how to commit the updated file?

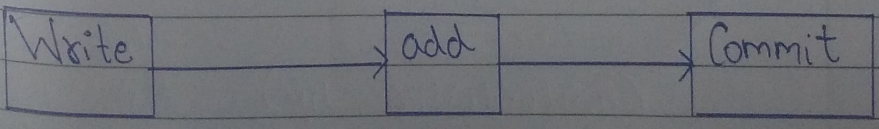
Ans

```
git add file1.txt        // This will add the changes
gitt commit -m "changes done"
```

To check status on commit/files?

→ git status.

Usual Workflow of Git



PAGE NO.:

Note: In git this --list,... are known as flags.

git log :

Helps to show the history of repository along with commit id, commit message

We can write it using a flag

→ git log --oneline

↳ It will only print commit message.

CONCEPT of Atomic Commits :

Git does not recommends to work on thousands of file and commiting 1000 features directly Instead it believes on atomic commits. ie (small scale / unit commits).
Because it becomes easier to always go back to previous commit and fix the issues.

PAGE NO.:

Change Default Code Editor.

1. git config --global core.editor "code - wait"

default code editor in git is [vim]

This is used when we simply write git commit (only) without writing -m message.
So in that case code edit file will open in VS code.

2. .gitignore

-> In this the files won't get tracked by git.

eg:->

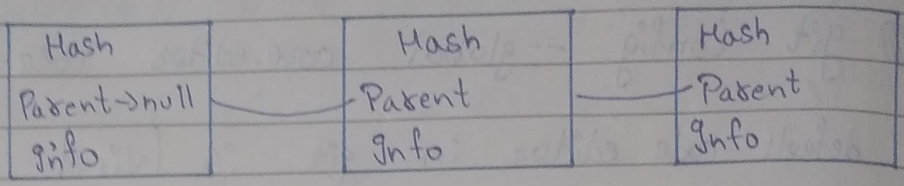
.gitignore
↳ .env
↳ index.js

So changes won't get tracked.

3. .gitKeep

Sometimes we just want to make some empty file for future use. So declaring it inside .gitKeep will not be tracked by ["git"]

GIT BEHIND the SCENES



→ Git infrastructure is totally object based. It considers everything as an object

(II) Git Snapshots

It is a point in time in the history of your code.

(मतलब अभी आपका code कैसा दिखता है। उसका एक Snapshot लेता है Git).

Note: Snapshot doesn't exactly means a photo

3 Major Pillars of git

1. Commit Object
2. Tree Object
3. Blob Object

Blob object is actuall file / code that it has to save
Tree Object is to traverse back by using reference
Commit objects are ones that are being commited. Commit object provides

reference to tree Object and tree Object provides reference to blob object.

(III) Commit Object
-> Each commit in the project is stored in .git folder in the form of commit object.
It contains

(1) Tree Information (Not Present)
(2) File Name (Not Present)
(3) Author
(4) Commit Message
(5) Committer
(6) Tree Object
(7) Parent Commit Object.

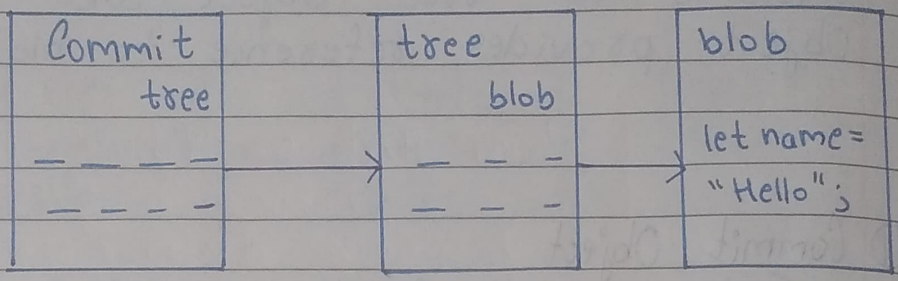
(IV) TREE Object :

Tree object is a container for all the files and folders in the project.
It contains :
(1) File mode
(2) File name
(3) File Hash
(4) Parent tree object.

(V) Blob Object :

Contains Actual File content.

(1)



We can come to Know about this by using internal commands.

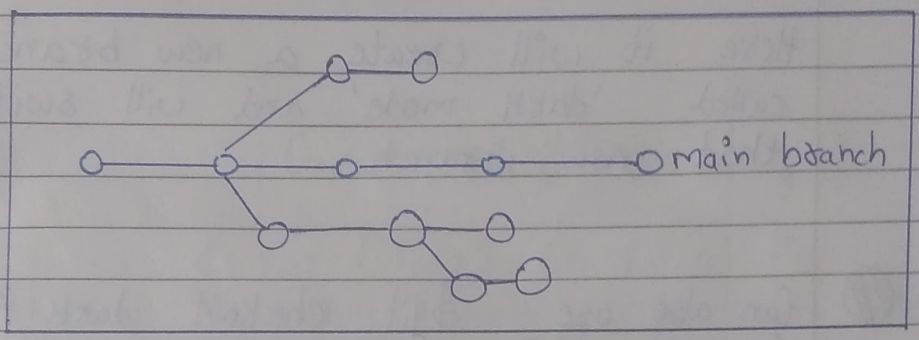
Can refer "chai n docs"

- Open Source Contribution :
  - ↳ fork repository
  - ↳ clone in codebase : git clone " "
  - ↳ git add , git push
  - ↳ Contribute (click)
  - ↳ Create Pull Request.

Porselline Commands (fanciness) fancy layer.	Guardning/Tooling Commands Manual
git init git add git commit git push.	Not Recommended Sha1= $(git hash-object -w example.txt) $git update-index --add --cacheinfo 100644 $ Sha1 example.txt

BRANCHES in Git

Git is like different timeline
Branches are a way to work on different versions of a project at the same time.



Head in Git: Pointer to the current branch.

(i) command: `$ git branch` // branching.

(ii) Create a new branch

↳ `$ git branch bug-fix`

(iii) switch to bug-fix branch

↳ `$ git switch bug-fix`

Now the moment suppose we switched to bug-fix branch and commited one file called suppose `"file3.txt"`

So if we switch back to master branch we won't se "file3.txt" {This is called as different timeline}

Q.2 How we can switch and create a branch simultaneously?

Ans `git switch -c dark mode.`

Note: If branch is alread created it will only switch.

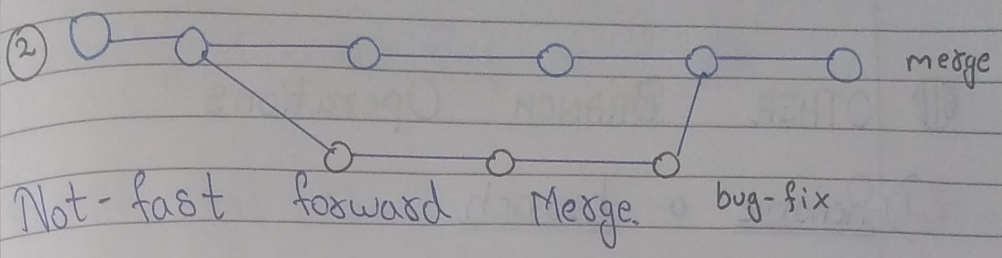
Here it will create a new branch called 'dark mode' and will switch to that new branch.

- Can also use $git checkout dark mode (To switch Branch)
- Can also check for git graph for branches and git commits.

Q.2 How to Merge our changes to Main/Master Branch?

Ans
1. First move to the branch where you have to merge
2. $git checkout master
3. $git merge bug-fix.

(2) and (3) — This technique is called Fast-Forward



CONFLICTING Situation

Different branches working on same file.
eg:→ master , bug-fix

Conflict Resolver in Vs Code is comparatively easier.
Vs Code gives us multiple options.

→ ① Accept Current change
② Accept Both changes
③ Accept incoming change (Merge)

(II) Methods to merge conflict branches.

① Abort the merge: $git merge --abort

② Manually edit : Removing unnecessary elements from structure.

```
<<<< head (current)
====
>>>> Bug-fix (incoming)
```

(II) OTHER BRANCH Operations

1) Rename a branch

git branch -m <old-name> <new-name>

2) Delete a branch

git branch -d <branch-name>

Git Diff

- It is just an informational command similiar to `$git log`.
- It is used to show the difference between the two commits.

Comparing Staging Areas.

(i) git diff --staged.

Stages are given unique index because everything in git is an Object.

@@ indicates line Numbers

PAGE NO.:

Comparing different Branches.

① git diff <branch-name-one> <branch-name-two>

② git diff branch-name-one .. branch-name-two

common Syntax.

Comparing different Commits

① git diff <commit-name/id> <commit-name/id>

STASHING

Stashing is a way to save your changes in a temporary location
Useful to make changes to file without commiting it so that we can come back to the file later and apply the changes.

Because we cannot checkout to different branches without commit or stash.

Simply write

① `git stash`

② We can also name our stash

↳ `git stash save "working later"`

③ We can also see our stash list

↳ `git stash list`

★ To Apply Stash

→ git stash apply

To Apply specific stash

→ git stash apply stash@{0}

PAGE NO.:

Applying and Droping the Stash Simultaneously.

-> git stash drop // only drops the stash

-> git stash pop // Applies and drops the stash

Clearing the Stash

git stash clear.

Applying stash to a specific branch

git stash apply stash@{0} <branch-name>

Suppose we have saved a stash at one branch but wants to apply on another branch.

{Git Tagging}

→ Can be thought of as Sticky notes.

- Creating a tag

→ git tag <tag-name>

① Creating Annoted Tag. (ie Tag with message)

→ `git tag -a <tag-name> -m "Release 1.0"`

② Delete a tag

→ `git tag -d <tag-name>`

③ tag a Specific commit

→ `git tag <tag-name> <commit-hash>`

④ List all tags

→ git tag.

{REBASE in Git}

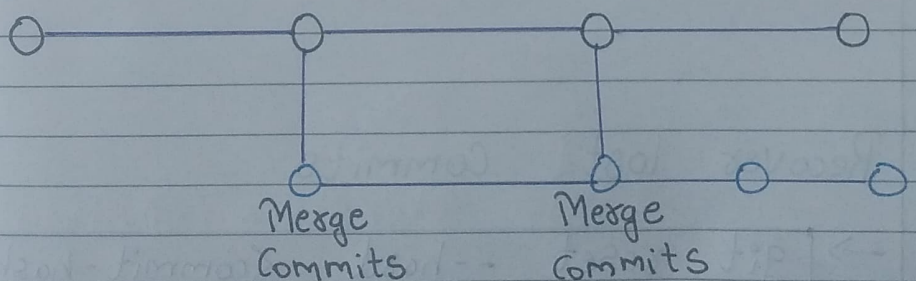
→ Very Powerfull command
→ If not used properly then it can mess up the entire code base
→ Usually not recommended to use untill fully sure about outputs.

PURPOSE

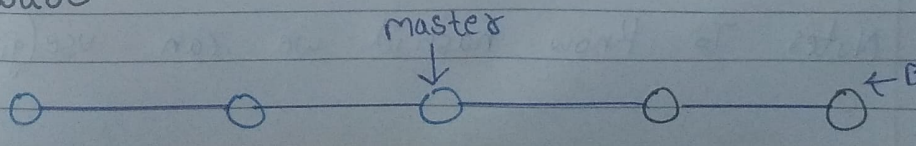
① Powerful Git feature used to change the base of a branch.

② Allows to move a branch to a new starting point

③ Some developers use this rebase over merge commands as it allows you to make changes to the code without affecting the original branch.



Rebase



PAGE NO.:

Command

→ `git rebase master`

In this scenario there would not be any extra commits / commits message for that particular changes made.

{ GIT Reflog }

Command that shows you the history of your commits. Allows us to see the changes that we have made over time.

(i) Can also manage for specific commits

→ `git reflog <commit-hash>`

(ii) Recover lost Commits

→ `git reset --hard <commit-hash>`

Note: To know hash id we can use (git log)

OR we can also apply it in another method.

» `git reset --hard Head@{1}`

In this way the head will move by one commit back.

Alternatives to Github

Gitlab ; Bitbucket ; Azure Repos

Some companies prefer to save code to their servers.

{ Git Push }

- git remote -v [Tells about remote server connections]

- git remote add origin " "
  - ↓
  - This name can be anything (origin is standard)

With this we are establishing a connection called origin.

- git push -u origin
  - ↓
  - { Adds Mainstream (upstream) }