→ Git

Git is basically version controll System is a tools that helps to track change in code.

Git is :
- popular
- free & open Source
- fast & scalable

Git : track the history : git store track the history of the project and save the all things.

: colleborate : when so many developer work together at that that time take care beware to not overwrite code.

→ Git hub :

Github is website that allows developers to store and manage their code using git.

→ Git

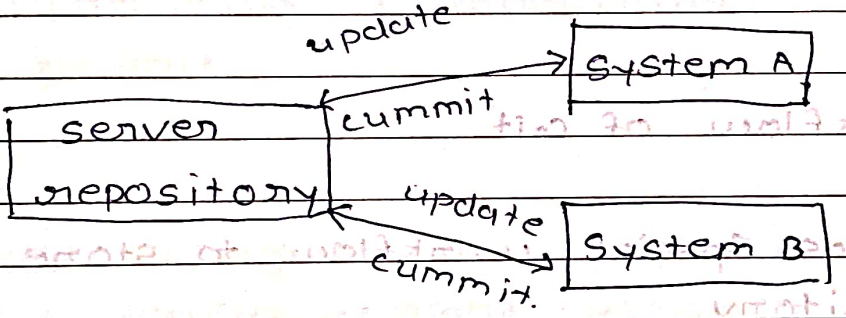
there are two type of vcs :
1. centeralized version control system
2. Distributed version control system

1. centeralized version control system:

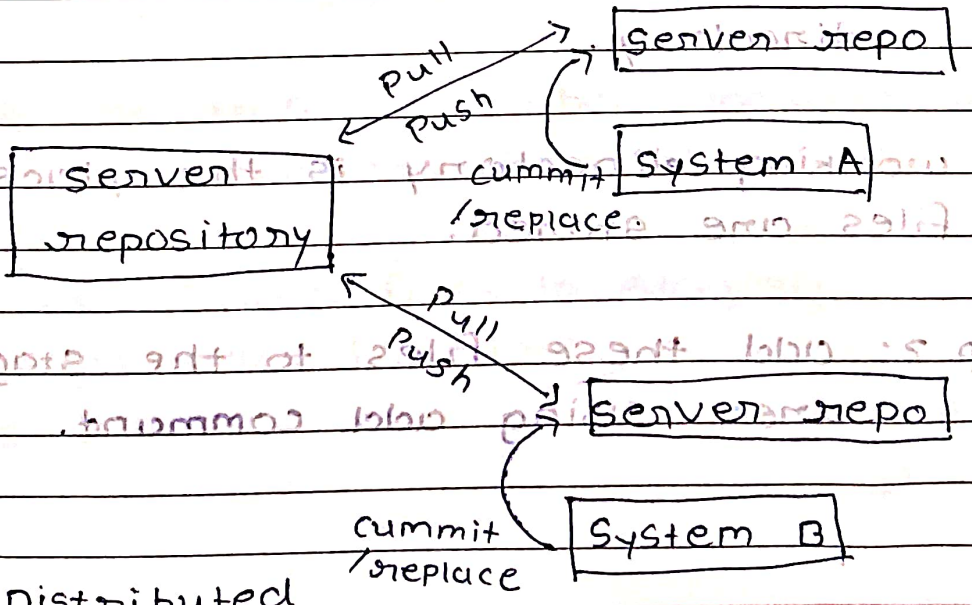
There's one main server that holds all the project's files and their history. It's like a centeralized hub where all the changes and updates to the project are managed and tracked.

2. Distributed version control system.

DVCS is like having your own copy of the project's library on your computer. Instead of just one centeral server.



centralized



Distributed

→ there are two types of local repository and central repository

Local repository:

typically located on local machine
only admin of the machine can work with this repository.

Central / remote repository

located on remote server
meant for team to share and exchange data.

→ workflow of Git

Git has special workflow to store files in repository.

Step 1: modify a file from the working directory.

working directory is the place where files are stored.

Step 2: add these files to the staging area using add command.

-> there are two types of local repository and central repository

local repository:

typically located on local machine.
only admin of the machine can work with this repository

central / remote repository

located on remote server
meant for team to share and exchange data.

-> workflow of Git

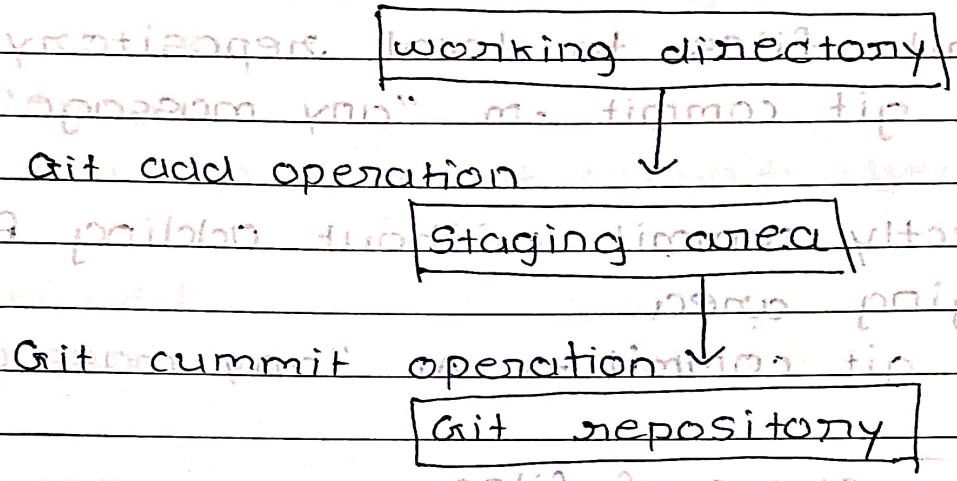
Git has special workflow to store files in repository.

Step 1: modify a file from the working directory.

working directory is the place where files are stored.

Step 2: add these files to the staging area using add command.

Step: 3 perform commit operation that moves the files the staging area and saved in local repository.



→ Way initializing a Repository
git init

→ configuring Git
git config --global user.name "My name"
git config --global user.email "someone@gmail.com"
git config --list

Note: ~ this sign show location of root directory (folder)

→ To add single files to staging area
git add filename

-> To add more files to staging area
git add --all OR
git add .

-> Commit files to local repository
git commit -m "any massage"

-> Directly commit without adding files in Staging area.
git commit -a -m "any massage"

-> Check status of files:
git status

-> To check log history of committed changes
git log

-> apply given bellow commant for add file to git repo (step by step)

git init
git add --all
git cummit -m "version massage"
git branch -M main
git remote add origin url
git push -u origin main

-> after modify file then you follow this step.

-> there are some type of git status

Untracked
new file that git doesn't yet track

modified
changed

Steged
file is ready to be committed

Unmodified
unchange

-> with out cummit git cannot trake file or new added file.

Add & cummit (after modification)

add - adds new or changed file in your working directory to the git staging area.

git add file.name OR
git add .

-> cummit - it is the record of change

git cummit -m "Some message"

here, -m show message

-> Push command:

Push - upload local repo content to remote repo

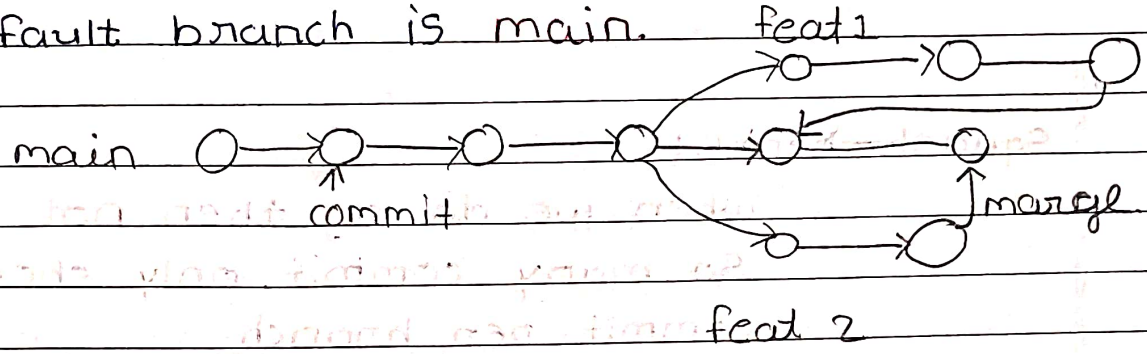
git push origin main

-> Simpley,
after modify file then you follow this command

git add.
git cummit -m "massege"
git push origin main

Git branching

default branch is main.



main branch lock. main branch only access by high authority

every develop we can work on own branch.

To create branch:

```
git branch <branch-name>
```

To go specific branch

```
git checkout <branch-name>
```

To marge branch

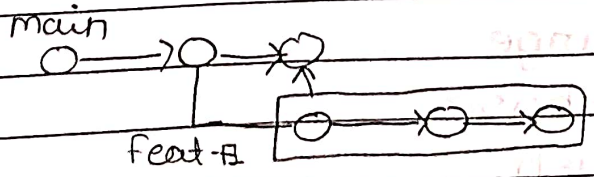
```
git marge
```

To go specific branch if not exist then creat

```
git checkout -b <branch-name>
```

main branch is single source of truth

Squash:



create only one commity and that commit marge in main branch that is called Squash

To squash marge.

git marge --squash <branch-name>

//That is called squash marge

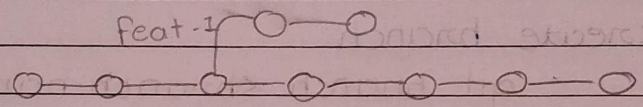
Squash benefit:

when we do log then not show so many commit, only show one commit per branch.

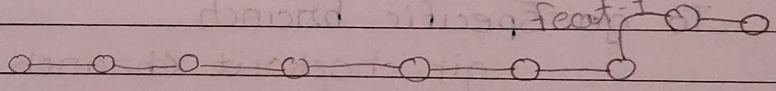
Squash cons:

It's loss history we cannot revert.

Rebase rewrite timeline.



Rewrite time line and make like this



how to keep your branch in Sync with MAIN? - Rebase main

git init
git add .
git commit
git branch
git checkout
git marge
git rebase
git push
git pull

This is 98%.

To squash merge

git merge --squash <branch-name>

(That is called Squash merge

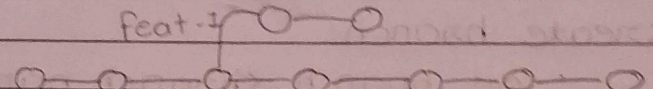
Squash benefit:

when we do log then not show so many commit. only show one commit per branch

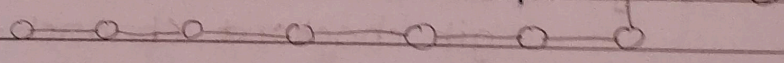
Squash cons:

It's loss history we cannot revent.

Rebase rewrite timeline.



Rewrite time line and make like this



how to keep your branch in sync with MAIN? - Rebase main

git init
git add .
git commit
git branch
git checkout
git merge
git rebase
git push
git pull

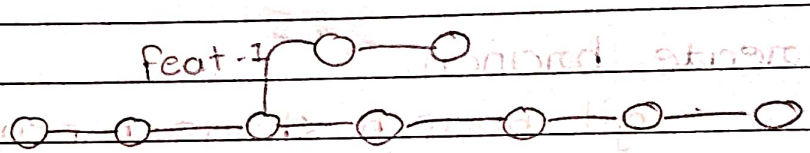
This is 98%.

To squash marge
git marge --squash <branch-name>
//That is called squash marge

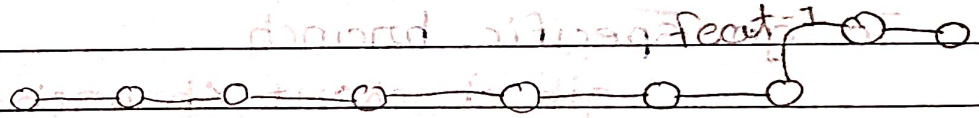
Squash benefit:
when we do log then not show so many commit, only show one commit per branch.

Squash cons:
It's loss history we cannot revert.

Rebase rewrite timeline.



Rewrite time line and make like this



how to keep your branch in sync with MAIN? - Rebase main

git init
git add .
git commit
git branch
git checkout
git marge
git rebase
git push
git pull

This is 98%.