git & github

Git Configurations

Git configurations are settings that allow you to customize how Git works. They consist of variables and their values, and they are stored in a couple of different files. To work with Git, you must set a few configuration variables related to user settings.

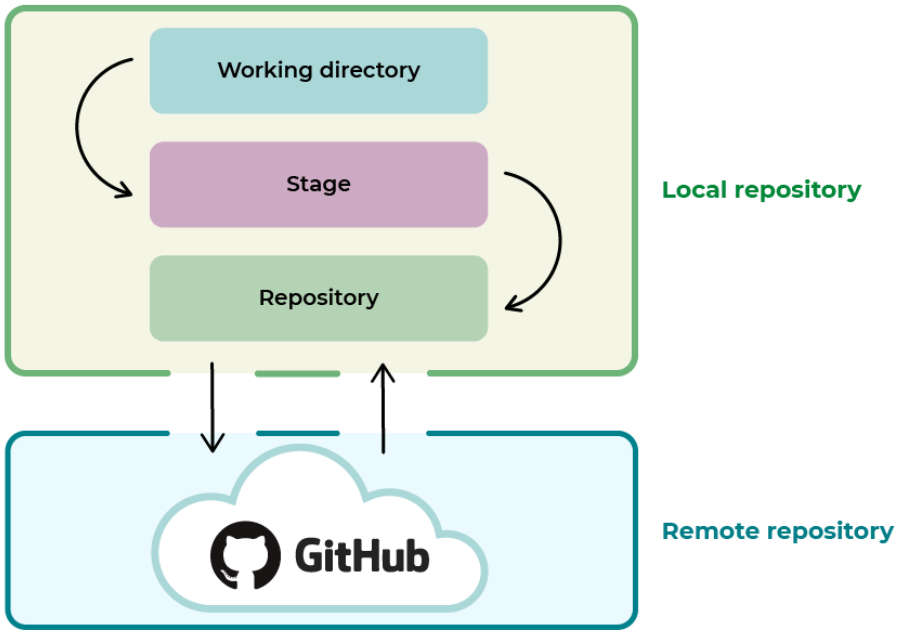
commands	Use	other info
- git version	show_vesion	
- git config --global --list	to see user name and email connected	
- git config --global init.defaultBranch main	to add defult branch	in mac it is by defult is master
- git config --global user.name "Mohansharma13" - git config --global user.email "*msharma13613@gmail.com*"	to config username and email	

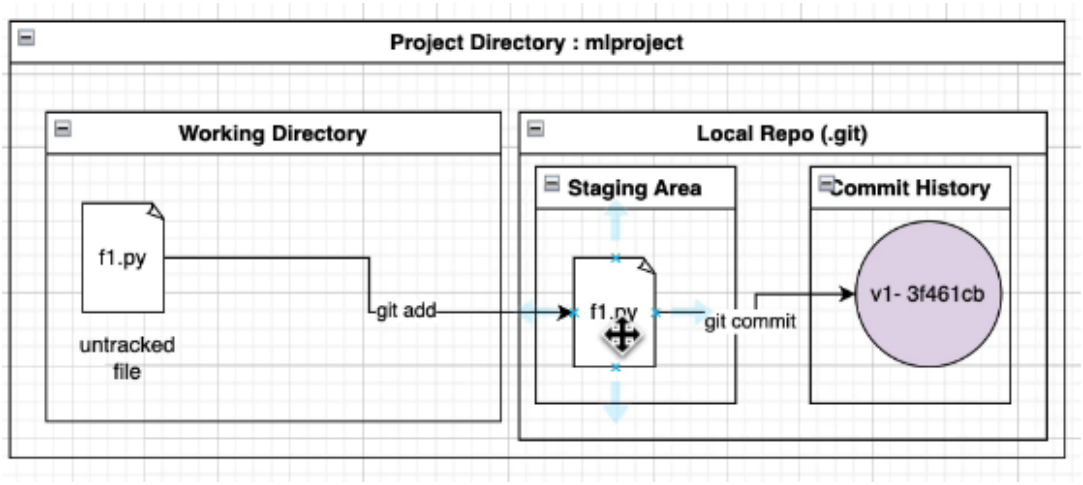
Working with Local Repo

local repo of git is .git file you create by using "git init" command

step follow to convet local directory to local repo

command	use
- git init	create local repo in local directory
- git status	show status
- git add #filename - git add .	-to add fiile to stage area with file name -to add all the file to stage area at once
- git commit -m "myfirst commit"	add stage file to local git repo
- git log	show logs and branch





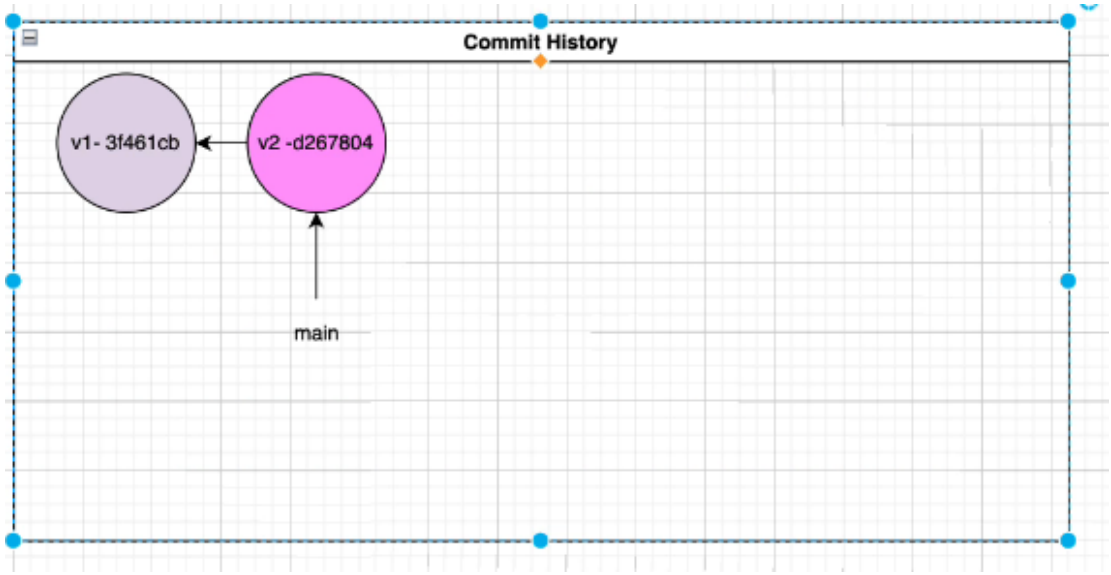
Working with Branches

branches in git are moveable pointer to commits that we perform in repo

you can see the branch using

```
git log
```

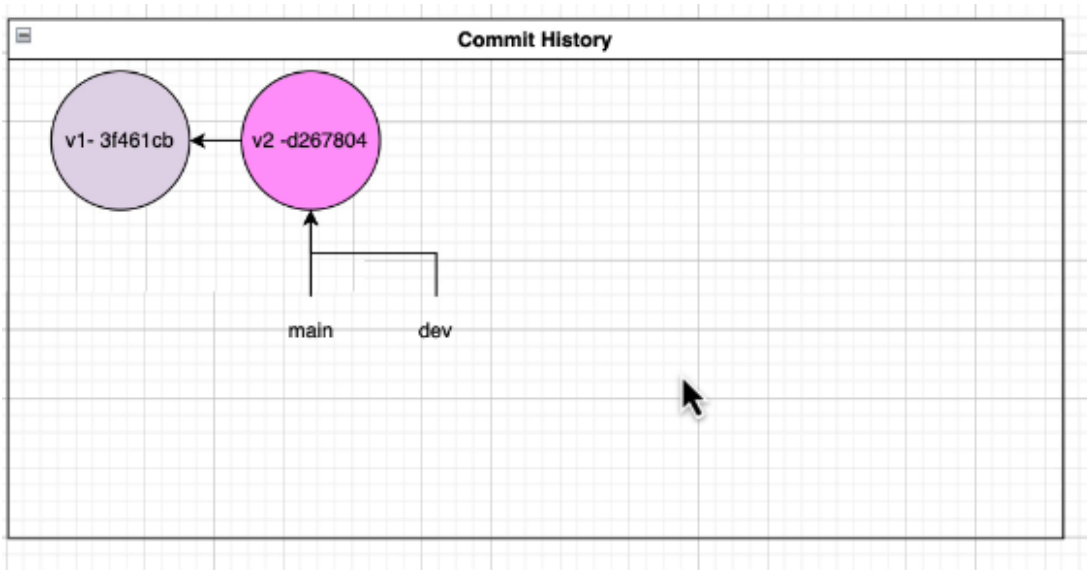
By defult you are on the main branch ,every time you make change (i.e commit to local repo) new version of that directory is form in repo and main brach automatic point to new version



To make new branch

```
git branch #branchname
```

```
# git branch dev
```



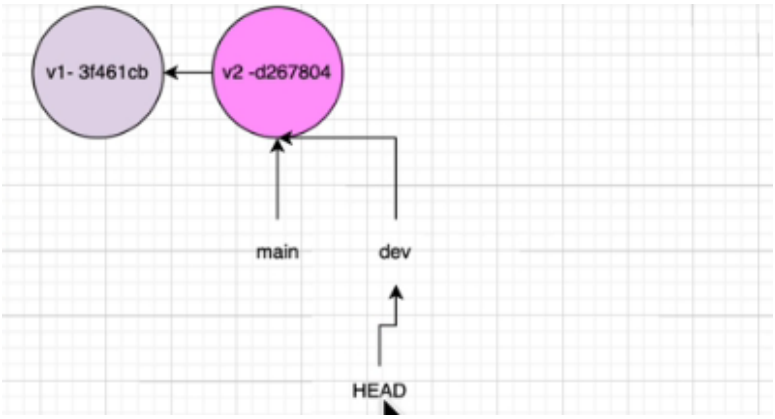
-> to check wich brach you are in you can use "git branch"

Switching the Branches

when you create branch you dont automatically switch branch you have to manualy change the branch

->there are two way to switch the branch

- git switch #branch_name
- git `checkout #branch_name`



now, any change that are create in branch do not reflected in main branch

now that you have created you have creted new branch

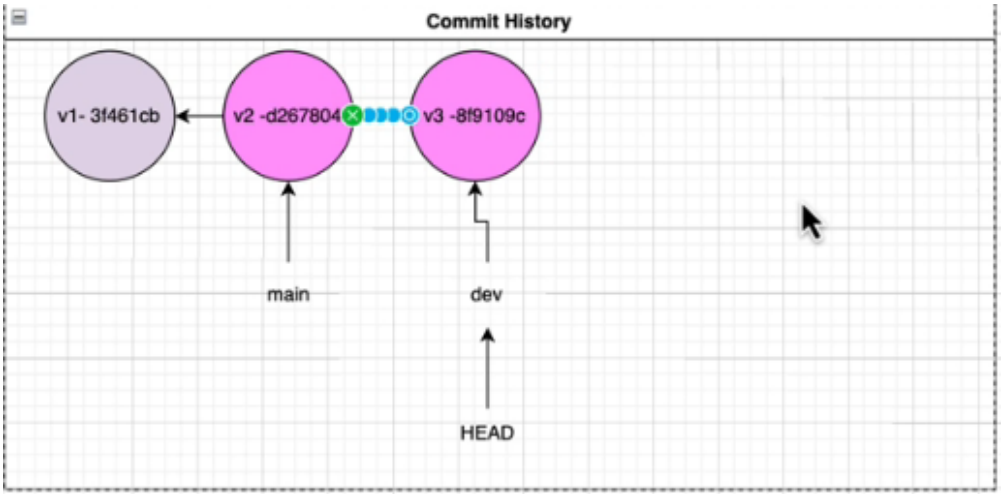
example

```
git branch dev
git switch dev
```

when you change in branch and make some changes and commit the changes it will create new version of local repo

eg-

```
git add .
git commit -m "new version created v3"
```



note->

if we switch to main branch the all the file in our directory will to converted to file commited to "v2" as main is still pointing to old version

Merging in git

merging is of two type

1) Fast-forward merge:

This happen when branch been merged has no new commits i is cause git just move the main pointer to branch pointer place

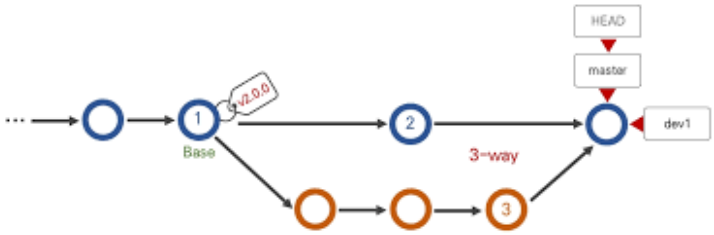
```
#create v4 in dev & commit to the repo

git switch main
git merge dev
```

2) Three way merge:

This occurs when there are divergent changes in the both the source and target branches

- **Algorithm that Git uses to combine changes from two different branches.**
- **This algorithm is also known as a "three-way merge" because it involves three commits:**
  - **Common Ancestor Commit (Base Commit): The commit where the branches diverged. It represents the last commit that both branches had in common before they started to diverge.**
  - **Branch A Commit: The commit representing the tip (latest commit) of one branch.**
  - **Branch B Commit: The commit representing the tip of the other branch.**



when you have to diffent branch the have grown indepedently when you try to combine them it cause the conflict

```
git switch main
git merge new_branch
```

it is on the developer to resolve the branch conflict it can be done in github or vscode manually

after the conflict resolve you can commit the changes

Git Checkouts

->checking out commits reffers to the process of switching your codebase to specific commit in a vesion control system

->when you are checkout a commit you are telling the vesion control system to set your working directory and codebase to the state it was in at the time of that specific commit

> NOTE -> if you are checking out the commit git wants you to deal with uncommited changes i.e commit them or remove them or stash them

command-

```
git checkout #commit-hash(you can see hash code using git log)
this will move your head to to commit it is recommend not to make any changes in this sate if you want the make branch
```

To switch back to current vesrion

```
git log --all
git checkout #commit-hash
```

Remote Git Repositories (github)

setting up the repo credential

```
#create remote repogit first

remote add origin <url>
#making new branch and moving to it
git branch -M main
git push -u origin main
```

some time while pushing to will counter the error that it is not be able to access the origin it is because it is not be able to ascess your github password
to remove this issue remove your github password from window Credential Manager
->now that you have delete the password for githhub from window if you try to push now git will ask you about the usernme and password

->enter username of your github and in password enter the personal ascess token

Cloning and Delete Branches

cloning the remote repo

```
git clone <url> <dir-name>
```

-> working with clone repo

```
#cheaking al the branch
```

```
git branch --all
```

-> making another branch to work

```
git branch unit_test
```

```
git switch unit_test
```

now to push unit_test branch to remote repo after some changes in the branch

```
git push -u origin unit-test
```

Remove branches

to remove the branch you first have to switch the branch the delete it

```
git checkout main
```

```
git branch -d <branch-name> #delete local branch
```

-> to remove the branch from remote repo use

```
git branch -d unit-test
```

```
git push origin -d unit-test
```