1. nano file/folder name: Edit directly in terminal

2. git - configuration levels:
    _ system : All users + All their repositories
    _ Global : All repositories on the computer for current user
    _ Local : Only current repository

Git steps : create the file/folder -> add to git <=> (modified) -> commit -> push origin/master (to remote directory)

3. git config --global user.name "name" : create name for github global

4. git config --global user.email "email": create email for github global

5. git config user.name: check out name of github

6. git config --list : list all info of the folder

7. git config --global core.editor "link you want to set text editor in github"

8. To create and add git folder/file

    create folder in terminal: mkdir <folderName>

    create file in terminal: touch <fileName>

create github of the file: git init (in the file directory)

add the file to git: git add <fileName>

add all file : git add .

To see the hidden item in mac: command shift .

9.
git status : check the status of the directoty you are in

git status -s: check status quickly
A : already add to the directory
?? : untracked
M : modified

git clean -n -d : to see which files you'll delete

git clean -f -d : delete all the untracked files

10. git add *.java : add all the files have .java in the name

11. git commit -m "What you did/ message" : commit the changes after you add the files to staging area

12. git status --help: get help from git

13. git commit -a -m 'what you did/ message' : Auto commit and track changes to modified file - (commit without step adding to stage)

14. git rm --cached <fileName>: untracked the fileName

15: git rm -r --cached <folderName> : Untracked the folder

16. Create .gitignore
    touch .gitignore
    git add .gitignore

17. git log: see the history using git

    git log - <number> of history you want to see

    git log --all: list all commit history

    git log <branch name> : list all commit history of specific branch

18. git commit --amend: to change the comments/ messages of the commit

19. git restore --stage <fileName> : to unstage the file

20. git restore <fileName> : to discard changes in working directory

21. git remote -v : check whether we have any remote repository

22. git remote add origin <link Https>: add the link of

remote repository to local repository.

22.1 Update git remote origin: git remote set-url origin <link http>

23. git push origin -u master: push and remember all local repository to remote repository on GitHub

23.1 Remove git remote: git remote remove origin (in directory)

24. git clone <link Https> : clone all folders/files from github repository to local computer

USING BRANCH

25. git branch <name> : create new branch

26. git branch : check which branch we are using

27. git branch -r : check which branch we are using for remote repository

28. git branch -a : check all information of branch including remote and local

29. git check out -b (branch) (name of branch): create new branch and move directly to that branch

git switch -c (create) <branch name> : create new

branch and move directly to that branch

30. git check out <branch name> : switch between branches

    git switch <branch name> : switch between branches

    git switch - : switch to master

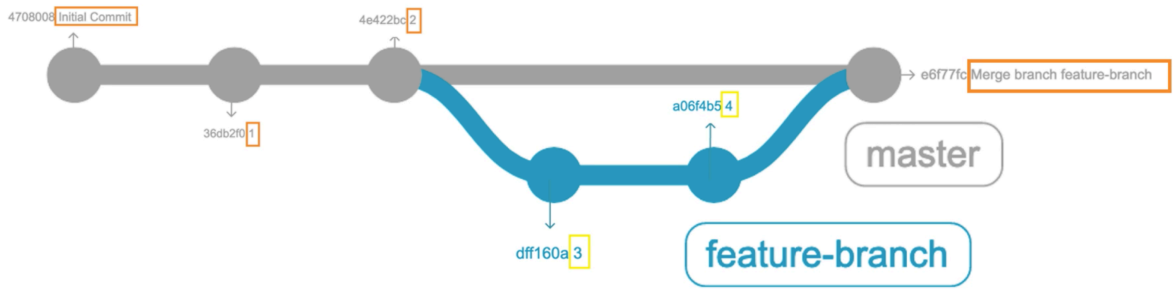
31. git push origin -u <branch name>: push and remember all local repository in specifics branch to remote repository on GitHub

32. git branch -d (delete) <branch name> : delete specific branch locally

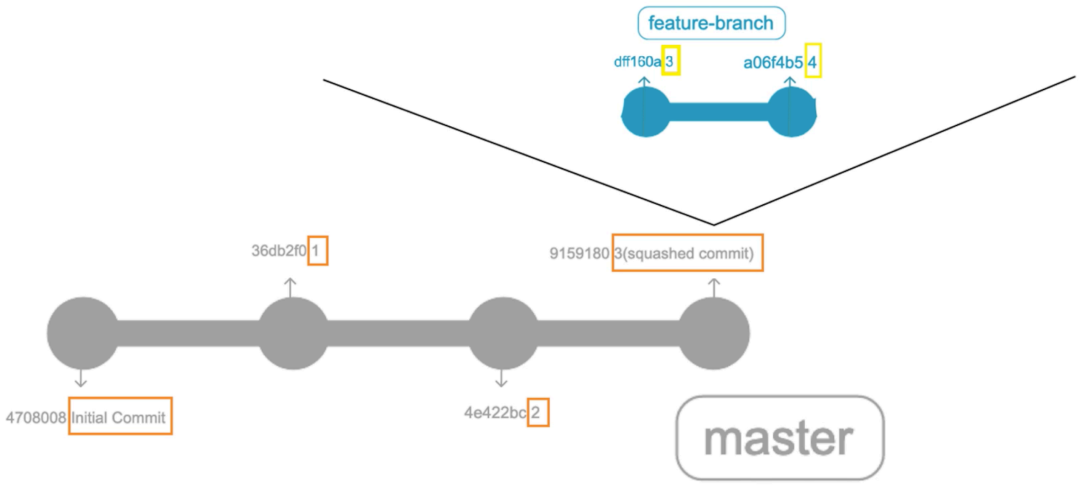
33. git push origin --delete <branch name> : delete specific remote branch

34 . PULL OPTIONS From another branch to master (local to remote) on GitHub website

Merge commit



Squash and Merge



Rebase and Merge

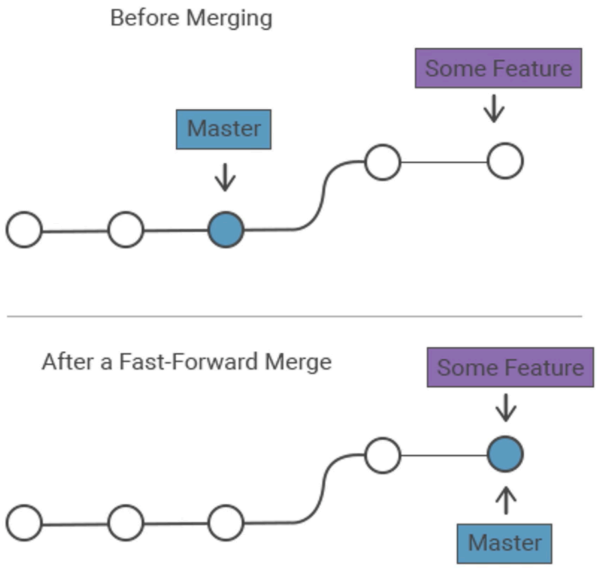
Rebase and Merge

35. git fetch : check if is there any change in remote repository / want to merge to local master repository later at home

36. git merge : merge all change from remote to local repository

Fast - forward

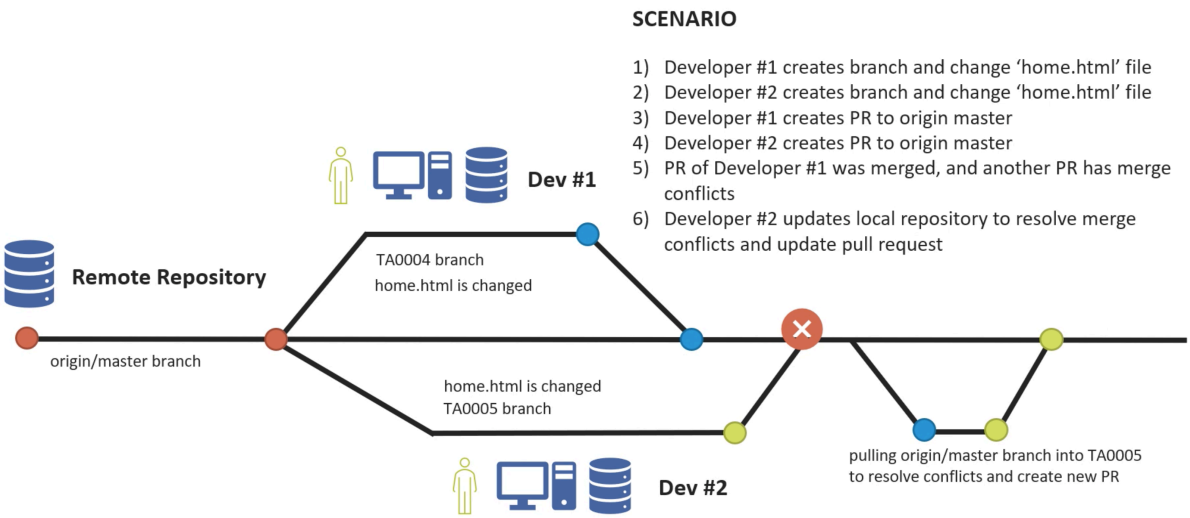
Fast-forwarding in Git



37. GIT PULL = GIT FETCH + GIT MERGE

38. When merge conflict can appear?

When merge conflict can appear?



WAY TO SOLVE GIT MERGE CONFLICT

Option 1:

Go to your branch , then use “git pull origin master” to update all the changing from remote repository to your branch

Edit the file conflict by using nano

Use “git commit”

Then use “git push” to push all changes to remote repository

go to GitHub and accept Merge pull request (Note: you can’t use rebase and merge option)

Option 2:

Git reset —hard HEAD~1 : remove the nearest commit (ONLY USE IF YOU DID OPTION 1 BUT WANT TO USE REBASE AND MERGE OPTION ON GITHUB)

Use “git pull —rebase origin master”

Edit the file conflict by using nano

Then use “git add .” And “git rebase —continued”

Use “git push —force-with-lease” : force to update the

local repository to remote repository

——

FOUR RULES OF HAPPY WORK WITH GIT

1. Always create branches from the master

2. Force update and change commit history only on your branches

3. Use —force-with-lease instead of -f

4. Always rebase on the origin master branch before creating a commit

——

CHANGE COMMIT MESSAGE

git rebase -i HEAD~3 : open the 3 nearest commits and customize commit message (you should edit the commit message from your branch, not on the master branch)

————————————————————————————————————

SQUASH COMMIT MESSAGE

git rebase -i HEAD~3 : use ‘ s ‘ in front of the message you want to use to squash (you don’t need to use ’ s ‘ for the first commit in the list)

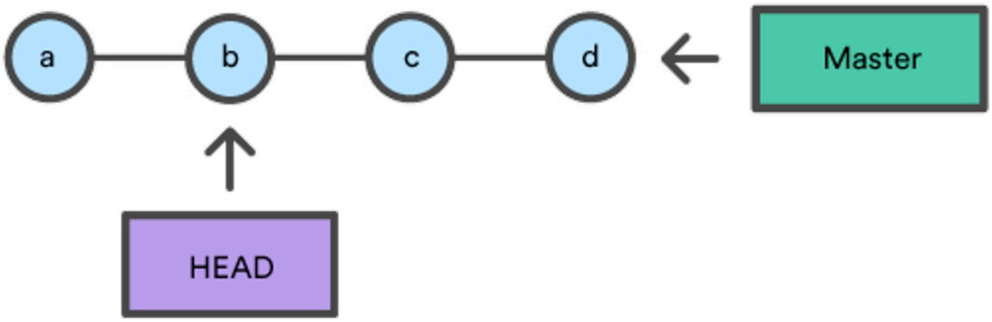
Then force update to the remote repository

——

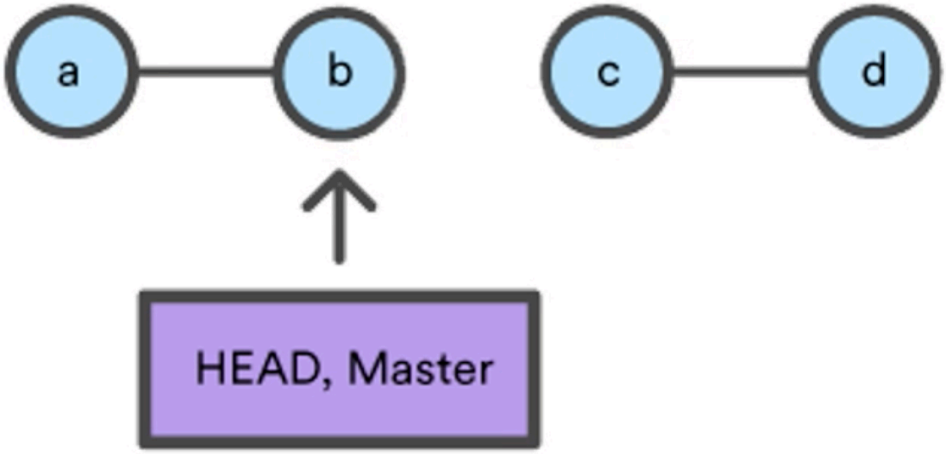
WHEN TO USE RESET:

* Undo last changes
* Change commit history
* Restore state of the remote branch
* Restore state of the branch after unsuccessful rebase

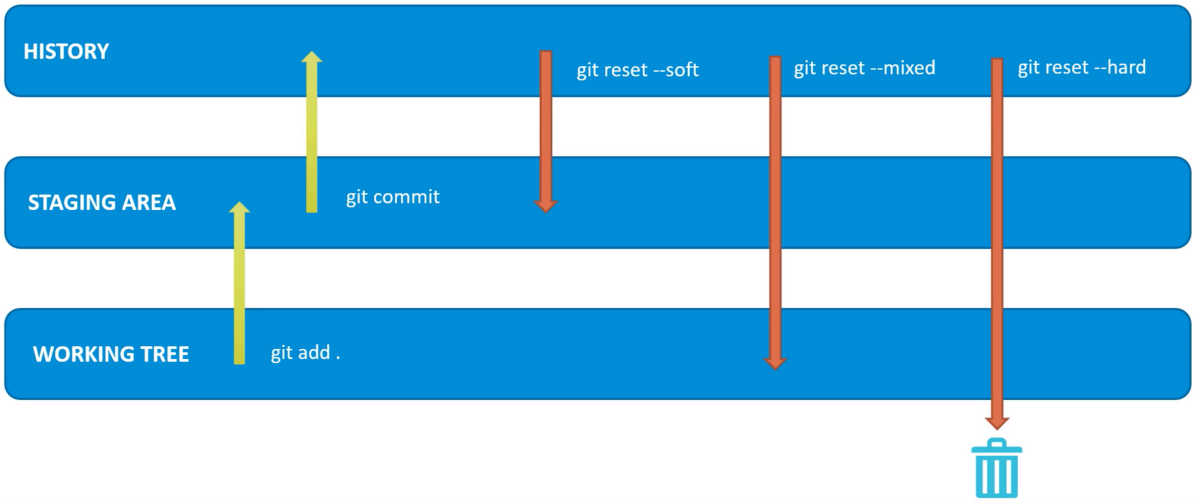
git checkout b



git reset b



reset soft VS mixed VS hard



git reset —soft < first 5 number of commit > : move head/ master to that commit you entered, move all the commits that above the commits you entered to staging area (ready to create new commit)

git reset (—mixed) < first 5 number of commit > : move head/master to that commit you entered, move all the commits that above the commits you entered to working tree (ready to add)

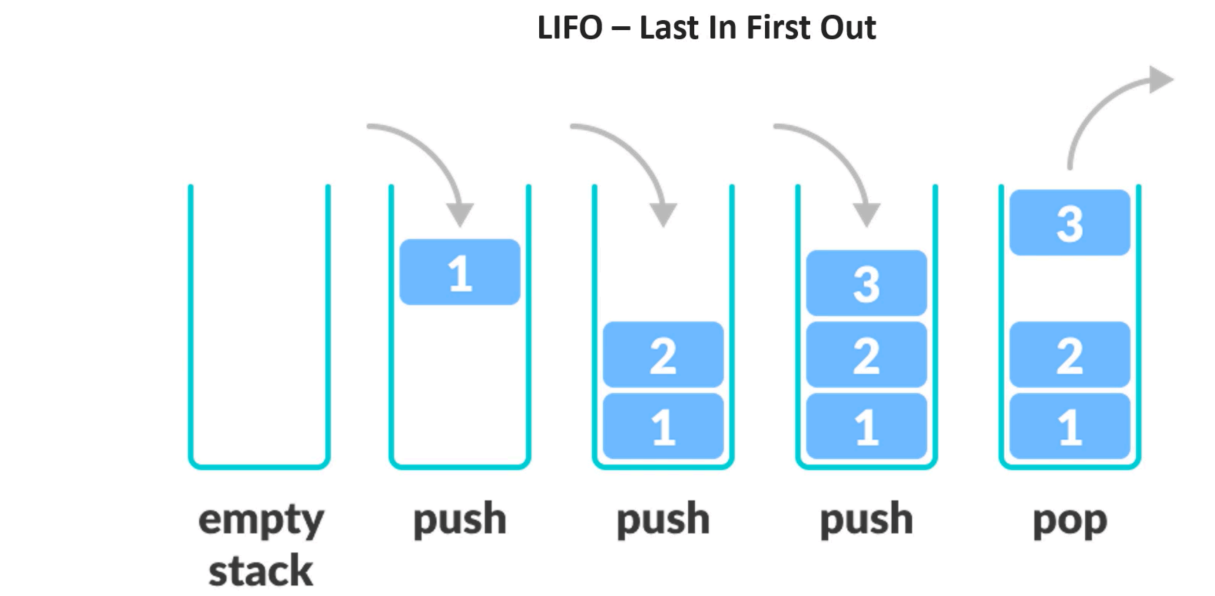
git reset —hard < first 5 number of commit > : move head/master to that commit you entered, delete all the commits that above the commits you entered

———

GIT STASH - use when you want to save all the changes into Stash Stack and apply it later

git stash apply : apply/get back - all the last change you save to stach stack into current branch

Stack (LIFO principle)



git stash list: list all the changes you saved

git stash apply stash@{<number>}: apply the specific change you want into the current branch

git stash save “comments/messages” : save the changes with the specific messages/comments

git stash pop: apply and remove the last change (the last Stash Stack)

git stash -u: save all the changes including (untracked/

not add yet) file by git

git stash branch <name branch> : create new branch using the last stash stack

git stash drop stash@{<number>} : not apply , only remove the last change (the last Stash Stack)

git stash clear : clear all the changes were saved in stash stack

———

. LifeCycle of git:

Untracked -> (add the file) -> Unmodified

<- (Remove the file) <-

Unmodified -> (Edit the file) -> Modified

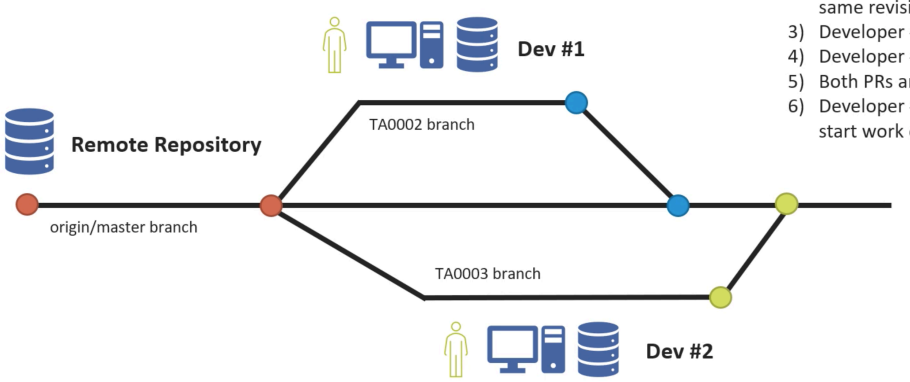
Modified -> (Stage the file) -> Staged

Staged -> (Commit) Modified / Unmodified

———

.Scenario

Development in a team with Git



SCENARIO

1) Developer #1 creates branch
2) Developer #2 creates branch from the same revision
3) Developer #1 creates PR to origin master
4) Developer #2 creates PR to origin master
5) Both PRs are merged
6) Developer #1 updates local repository to start work on new task

---

.vi editor :

press i key for insert/editing

press esc key for get out of insert mode

write :wq for write and quit