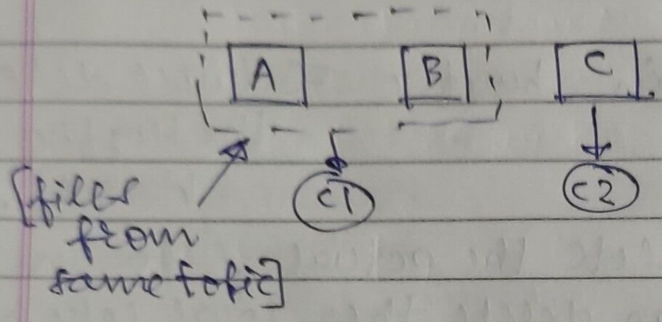
Mastering Version Control

Page No.:____
Date: __/__/__

Perfect commit :-

- Commits must be made to individual files, highlighting the actual changes done to each of them.



# Only combine commit for same topic files.

# Staging area will help : select files to be commited in the next commit.

- Commit messages must be very understandable: contains two parts :
  - (a) subject : concise summary of what happened
  - (b) Body : more detailed explanation. (Reason for change, what change).

git commit ; a window will open:
(type the subject
leave one line
and then write body)

# Git provides a technique called "branch", but it doesn't state its use.

Team members must agree on how to deal with branches and releases and updates.

Page No.:_____
Date:__/__/__

Branches enhance structures and workflows
It helps to manage state, release and feature branches in git.

- Two example of branching strategies one may adopt :-
  (a) Github flow : contains a single main branch and others are short lived (very simple/clean)
  (b) Git flow : contains many long-run branches.
    - more structure, more rules
    - long-running; "main" + "develop"
    - short-lived : features, hotfixes and releases.

The best branching model will depend on your project, release cycle and team.

Pull requests

(1.) When you finish the code on one branch and want to integrate it to the master/main,
- in some cases, your changes are bit complicated.
- Hence, pull requests will invite reviewers to provide feedback before merging.
- and then merge.

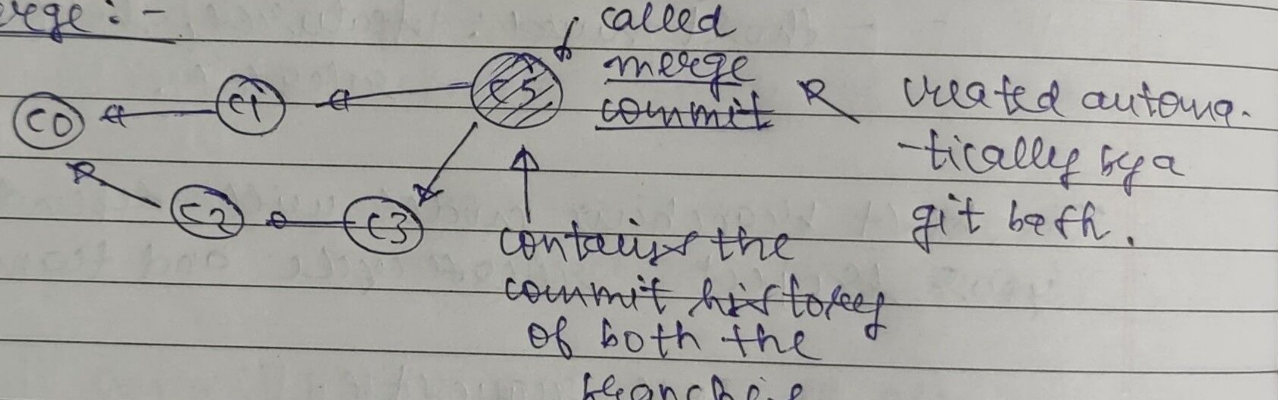
(2.) Second use-case is "contributing code to other repositories".

- you decide to improve the repository but you don't have access to it.
- Fork it ⇒ Fork is the personal copy of git repository.
- Make changes and then sent a pull request to the main contributor to include the changes if he/she wishes to include.

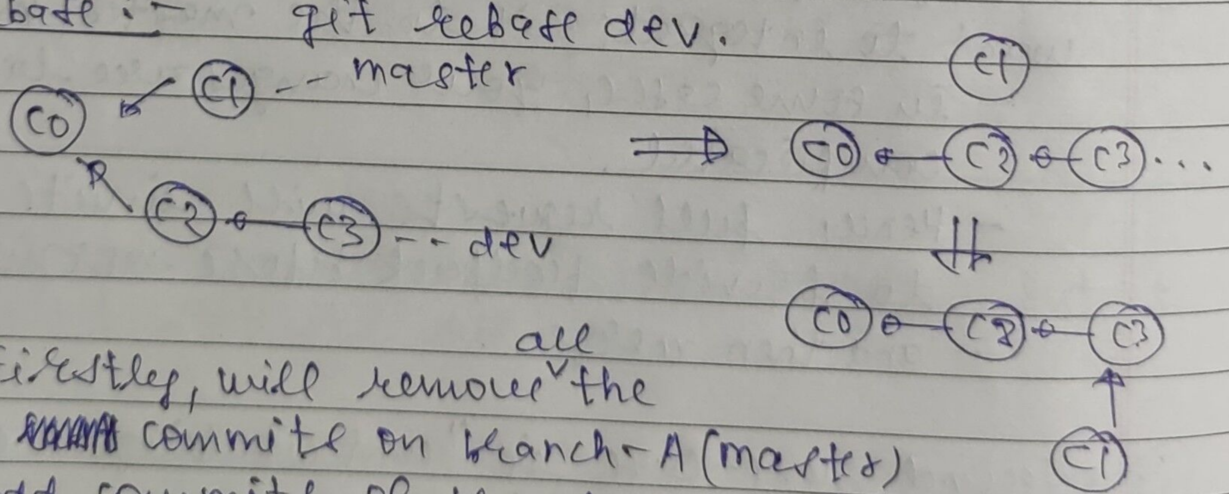
In fork repo, we can make changes.

Merge and rebase

· Merge :-


C0
C1
C2
C3
C5
called merge ~~commit~~
created automatically by a git bash.
contains the ~~commit history~~ of both the branches.

· Rebase :- git rebase dev.



#Firstly, will remove all the commits on branch-A (master) add commits of branch-B and then add the commits of branch-A then.

- Rebase will rewrite the commit - history.

- Parent commit of C1 will also change.

Git rebase creates a new set of commits applied on top of the target branch, while git merge creates a new merge commit that commit that combines the changes from both branches.