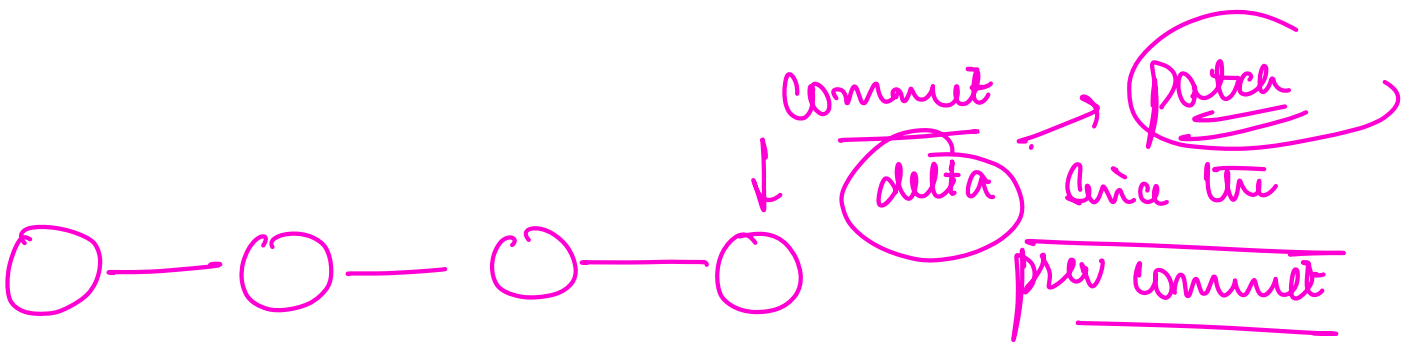
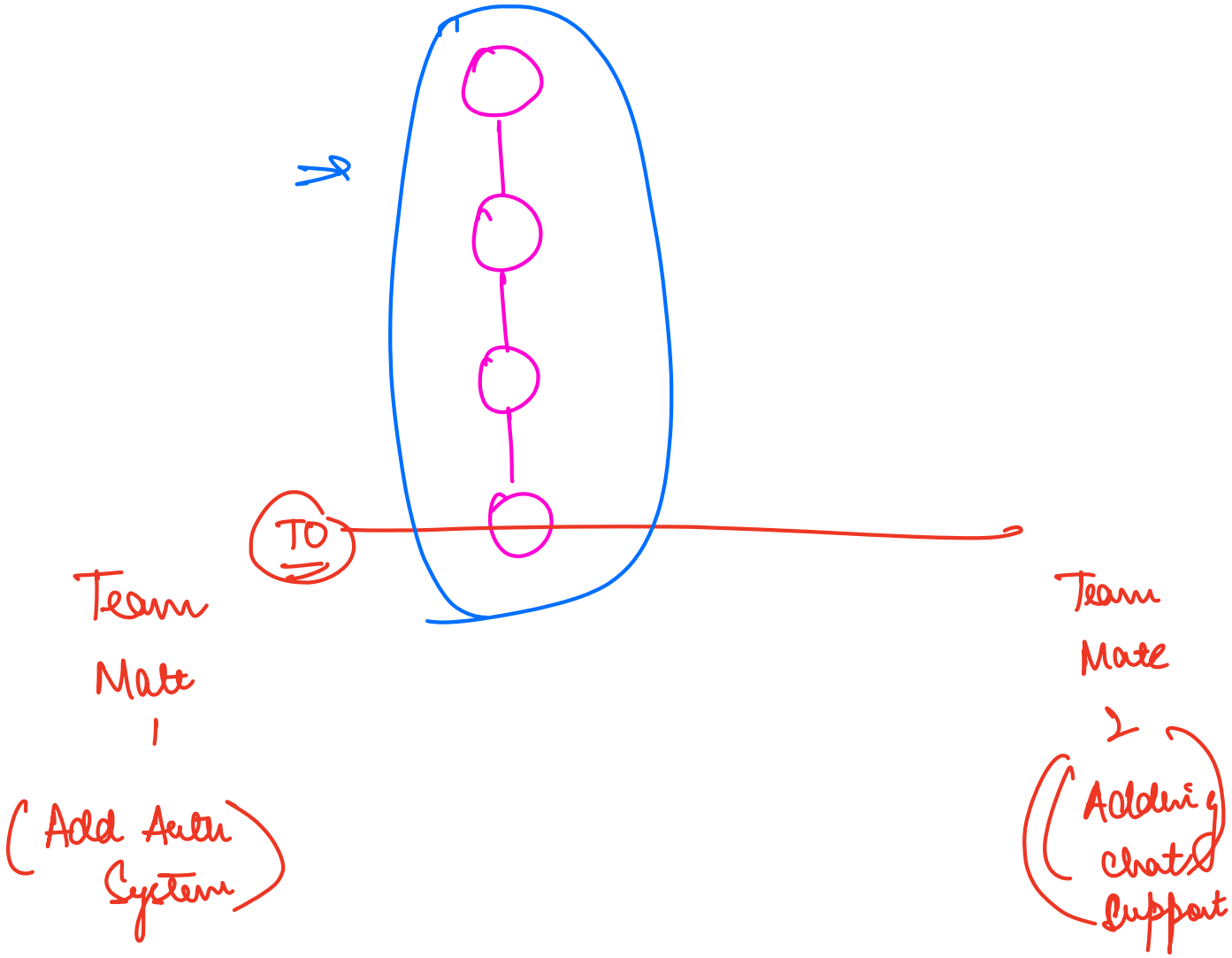
Will start at 9:10

Agenda

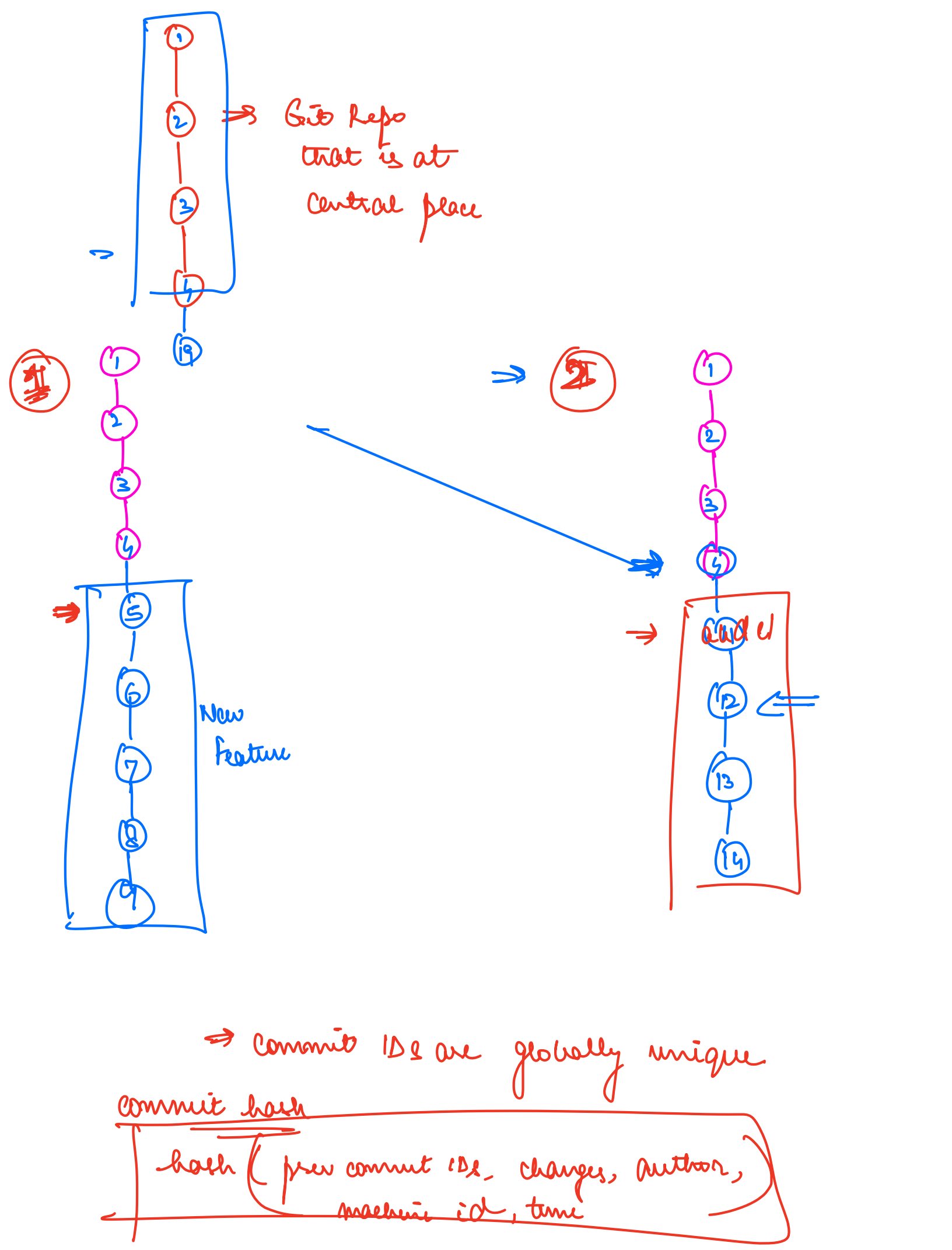
- Complete Git
  - checkout
  - branch
  - merge
  - rebase
  - HEAD
  - pull
    - fetch
    - push
  - clone
    - git remote
  - Submitting first PR
  - Cherry-pick
- Project Overview





Have short commits

→ Small freq commits



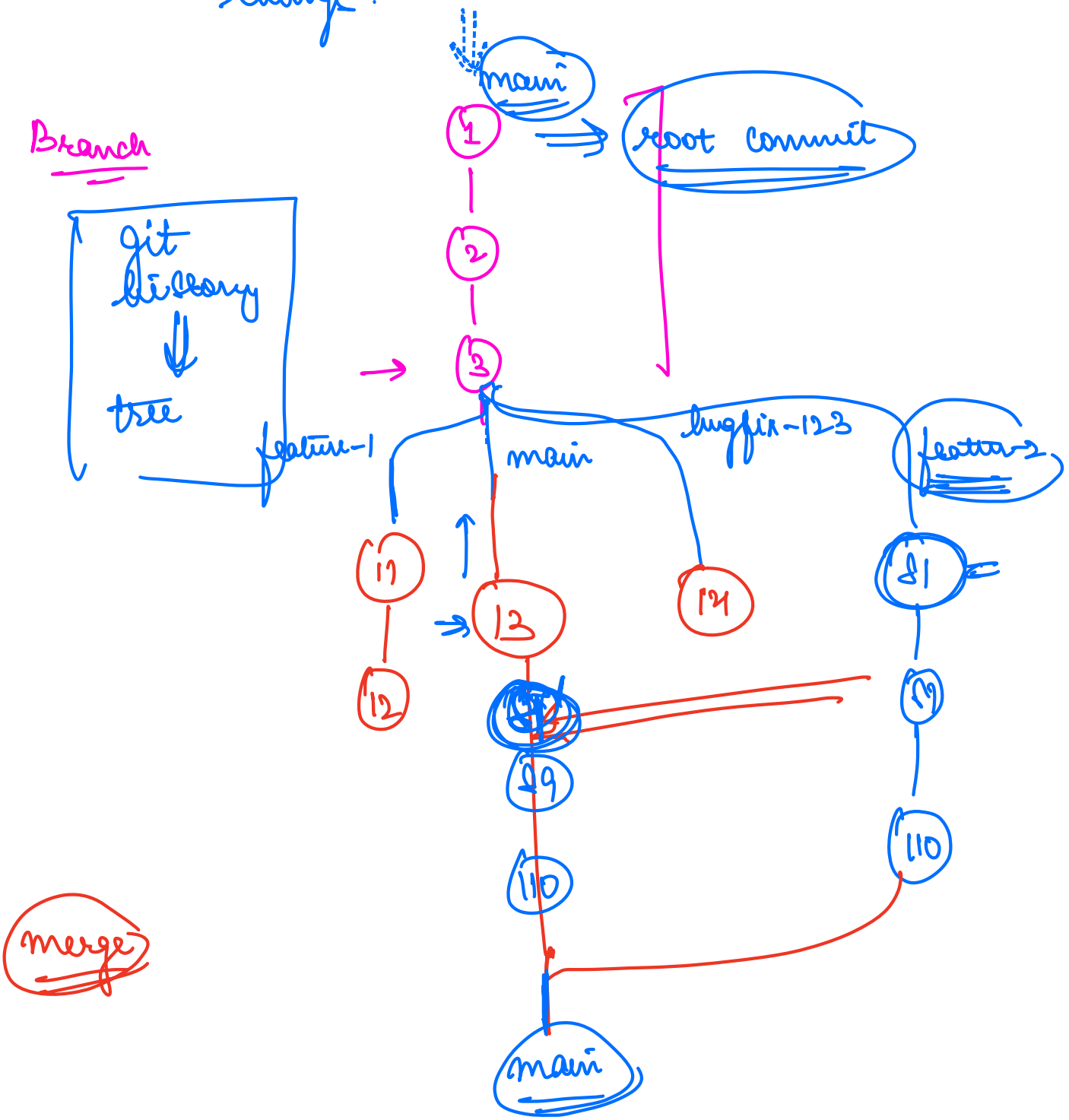
⇒ Commit IDs are globally unique

commit hash

hash (prev commit IDs, changes, author, machine id, time

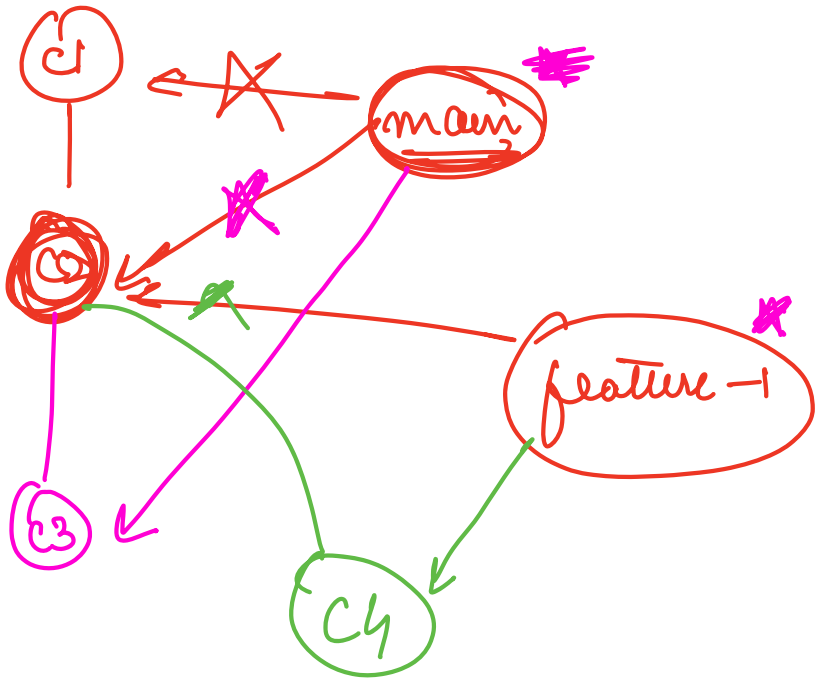
→ Same person may be working on multiple parallel things at same time

→ diff people may be working on diff things.



Branch Name

→ variable pointing to a particular commit



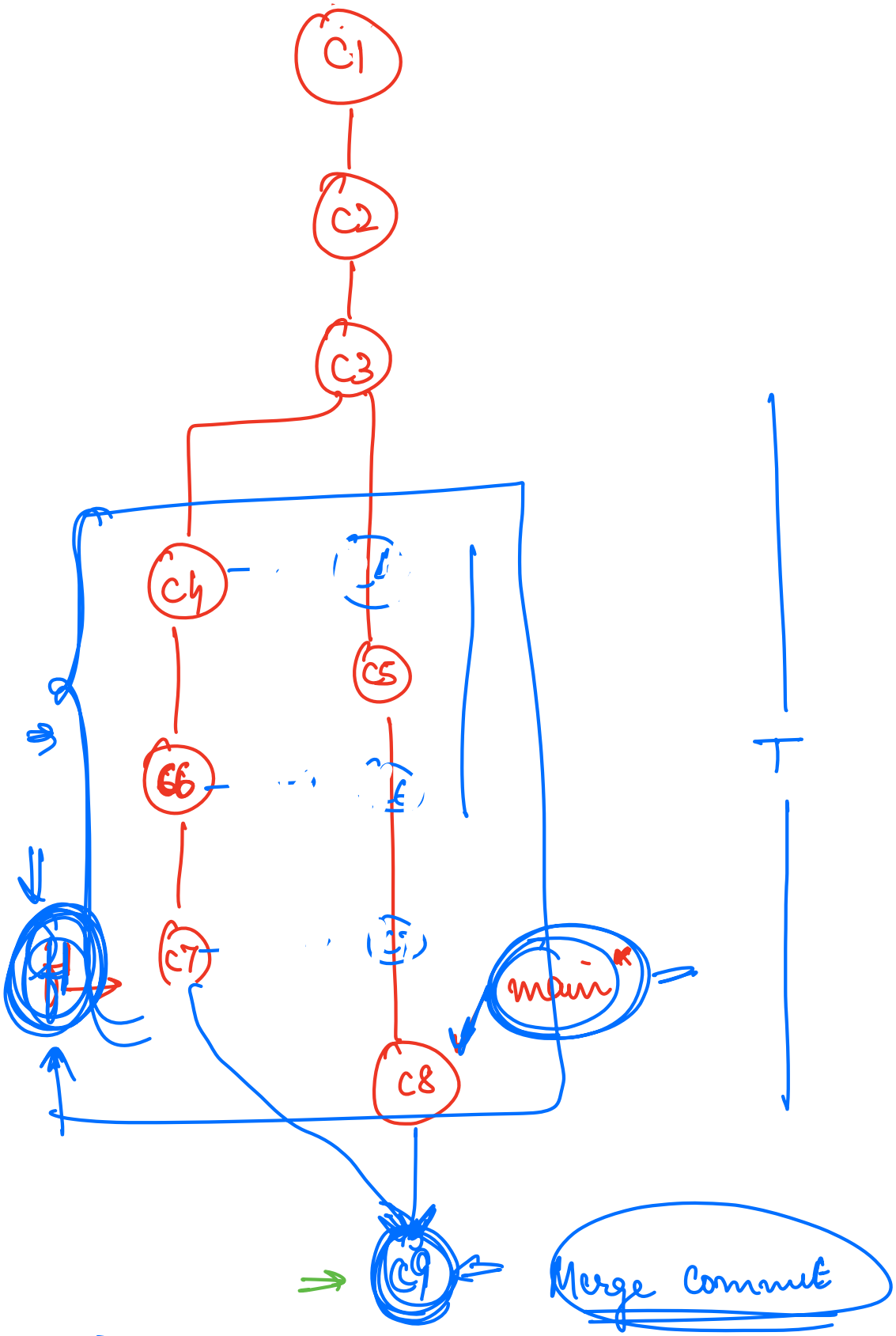
git checkout -b new_branch_name

↓

git branch new_branch_name

+

git checkout new_branch_name.



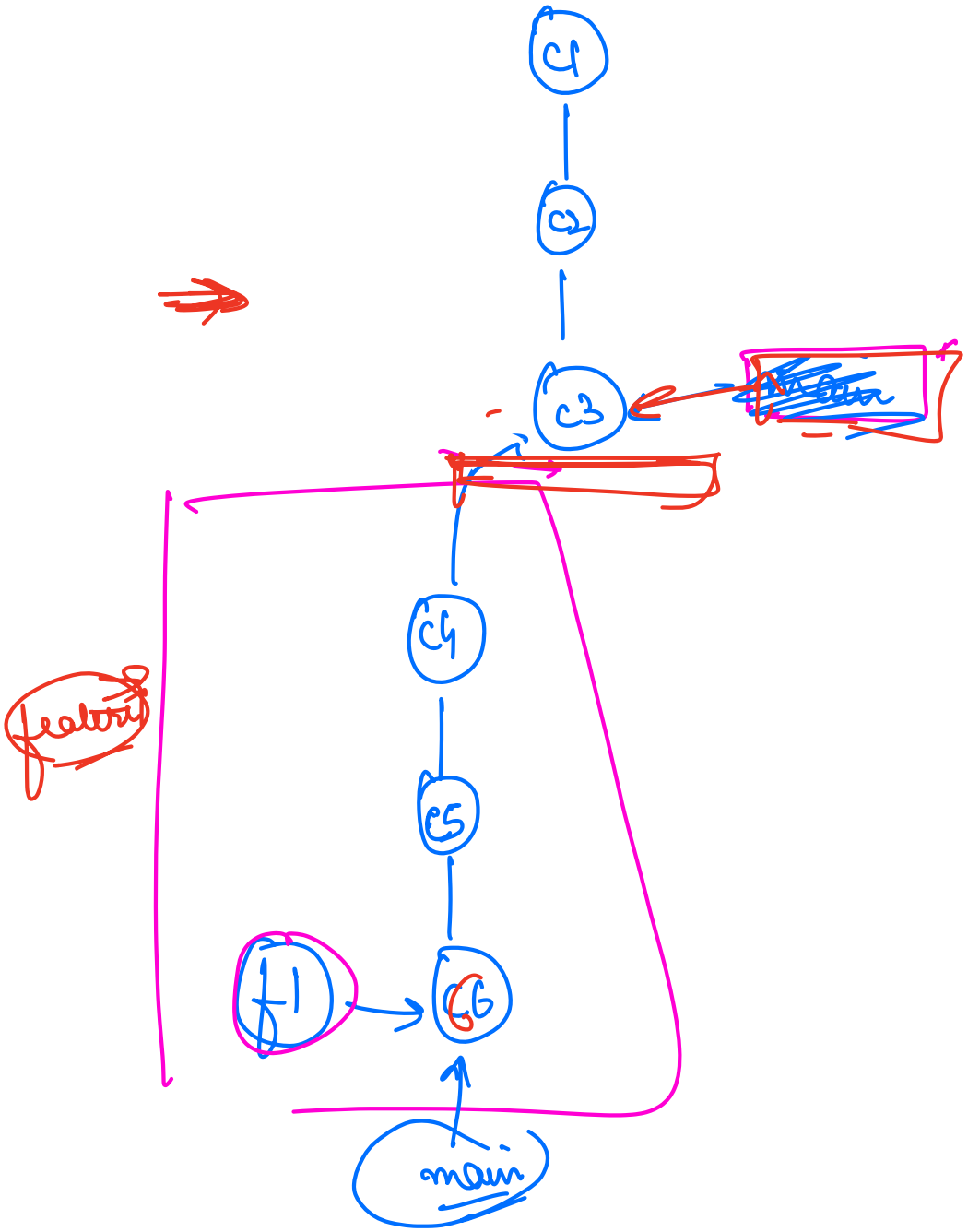
Merge Commit

During merge

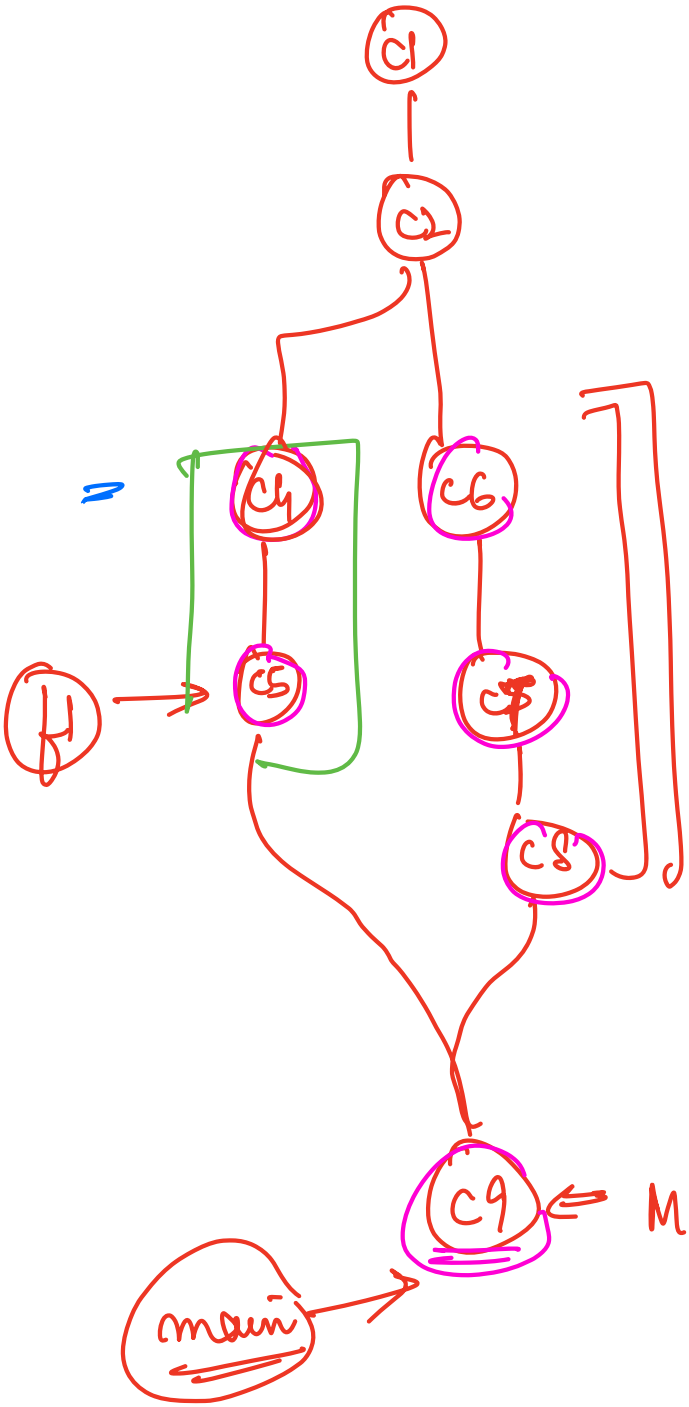
→ When git see changes have been made in both branches, it creates a new commit which is called

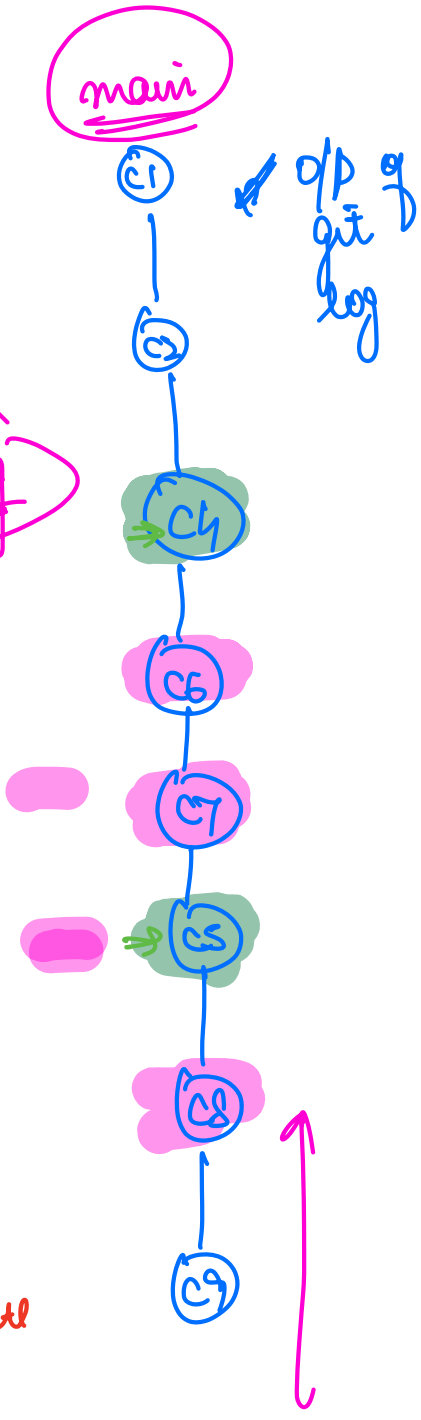
a merge commit

→ if no changes, no new commit



Problems with merge

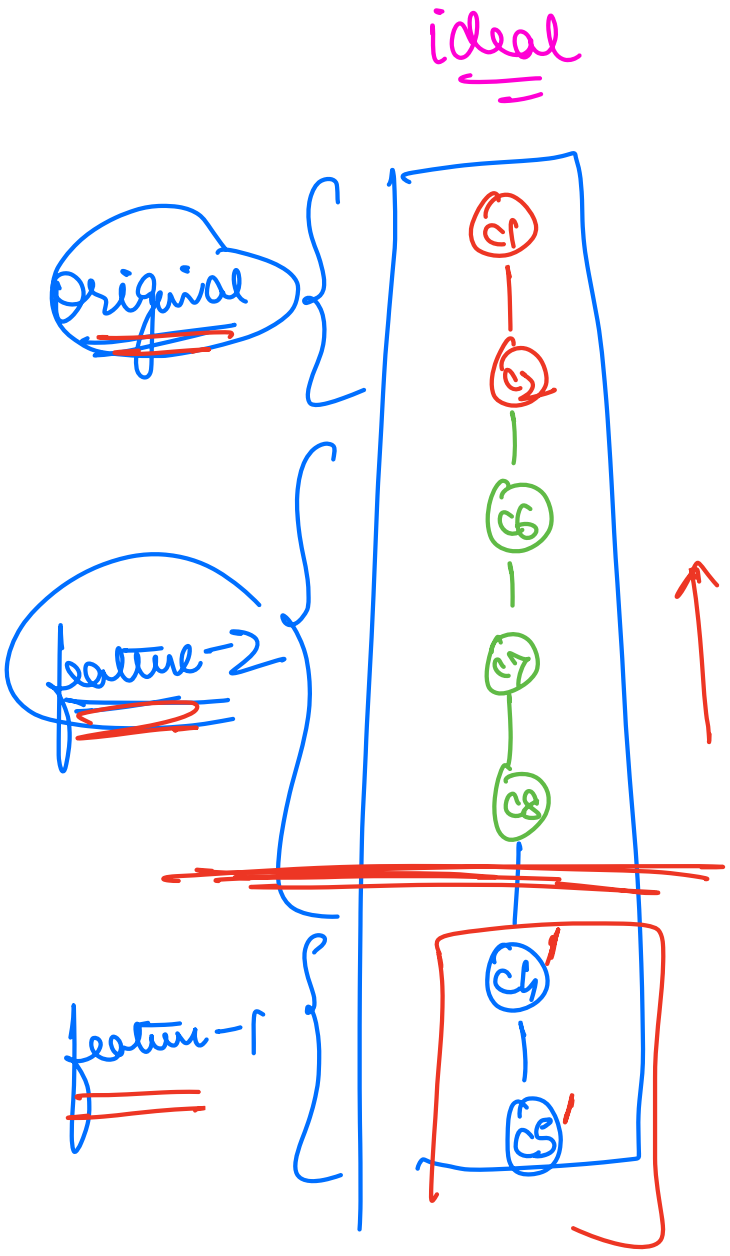




git log

Case

1. Code was working
2. merged H1 =
3. prod is down



When we did merge, we didn't actually copy commits of 1 branch into other instead we created new commit with 2 parents

git rebase

(Break till 10:40)

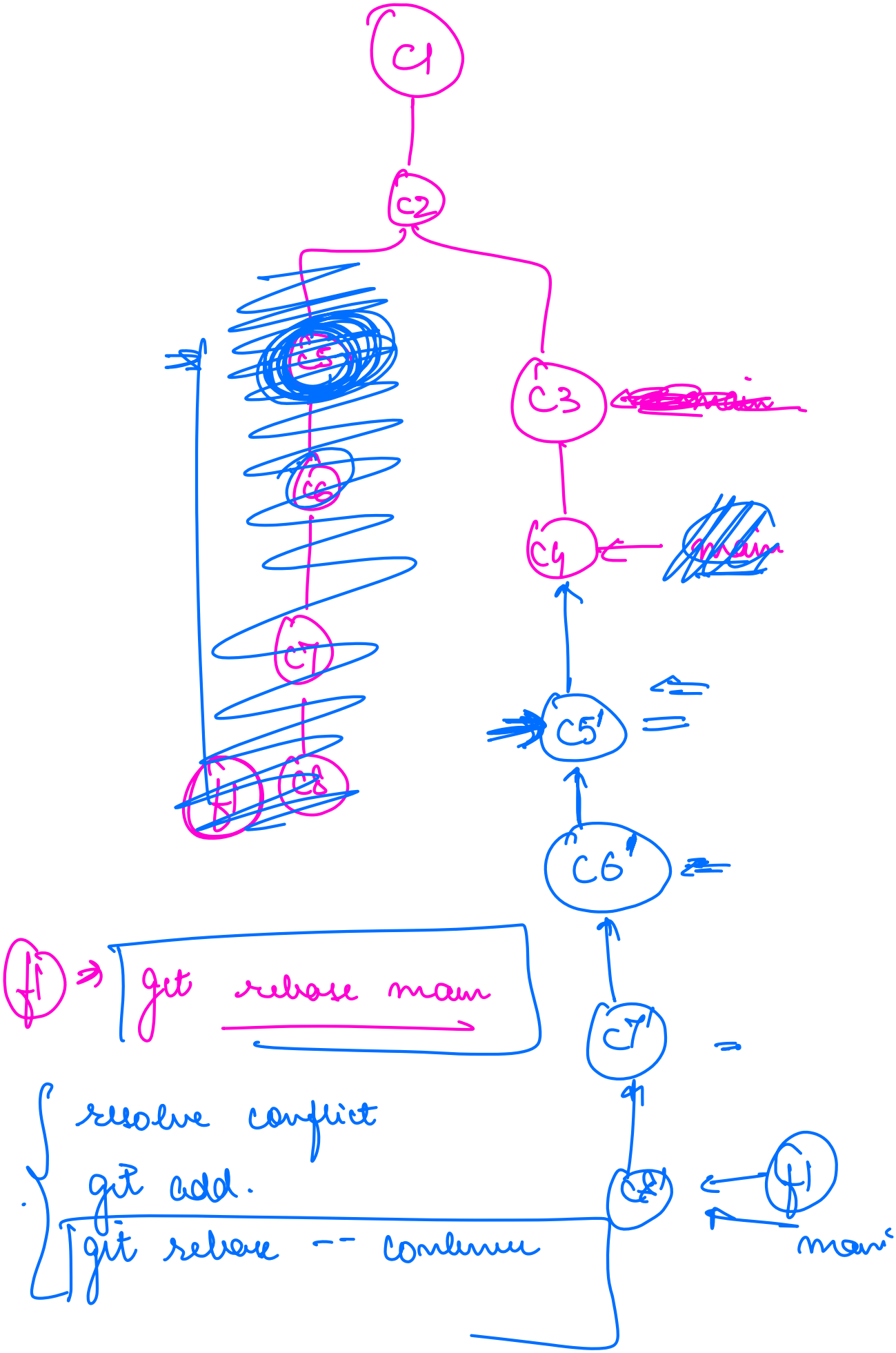
When working in prodⁿ, always rebase instead of merge

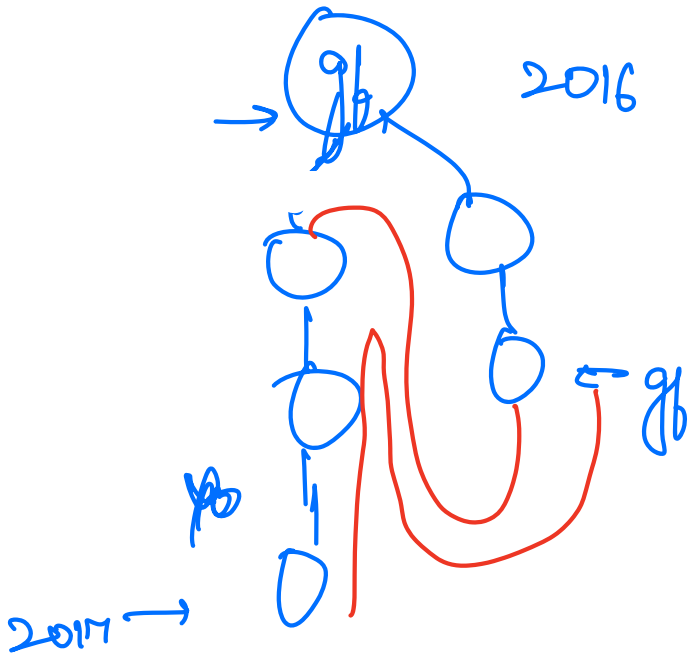
⇓

Ensures history is clean

When working on a feature branch, keep rebasing the feature branch frequently on main

⇒ Because when you will merge, merge will be fast-forward.


C1
C2
C3
C4
C5
C6
C7
C8
C5'
C6'
C7'
C8'
f1
git rebase main
resolve conflict
git add.
git rebase --continue
f1
main



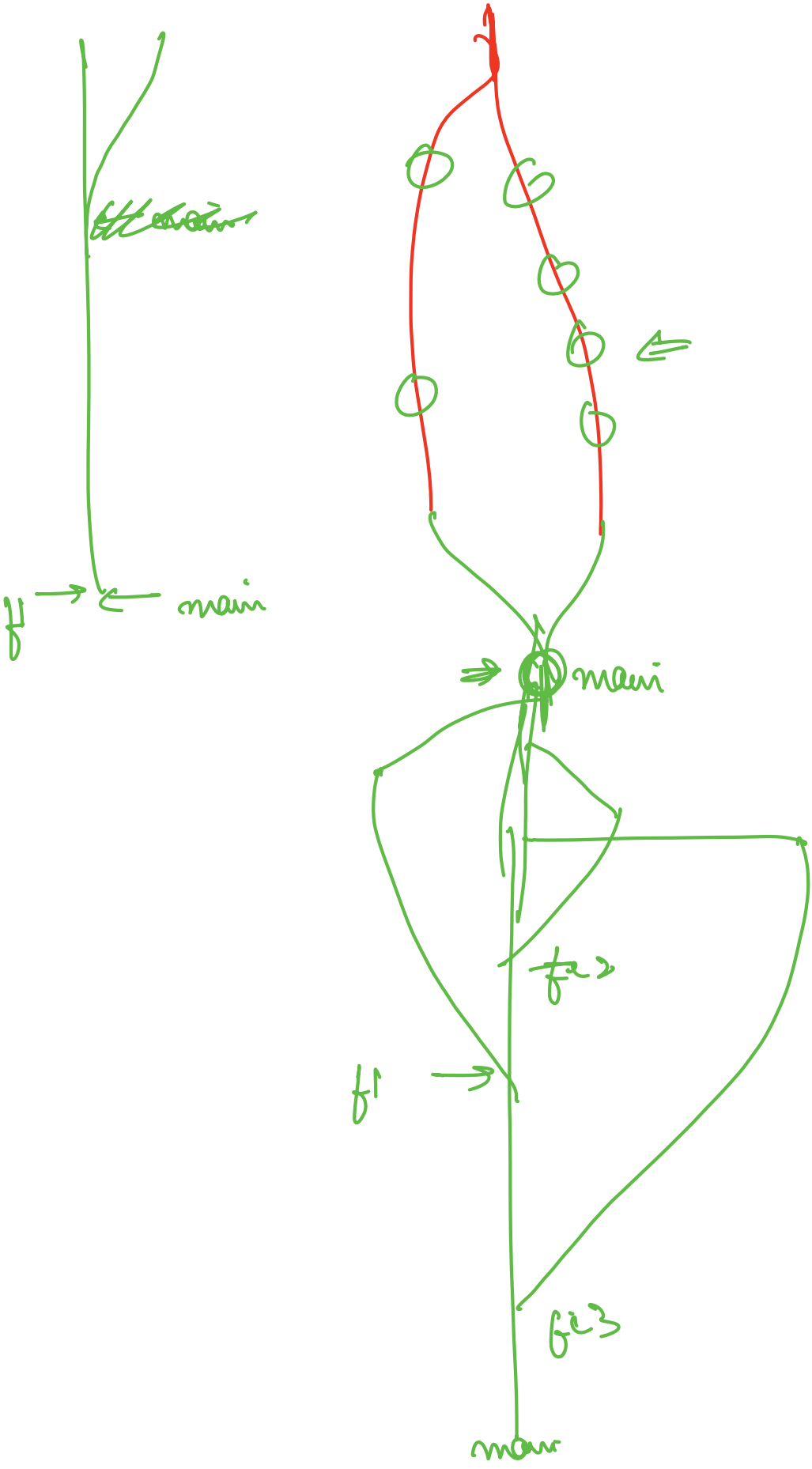
→ Project Overview

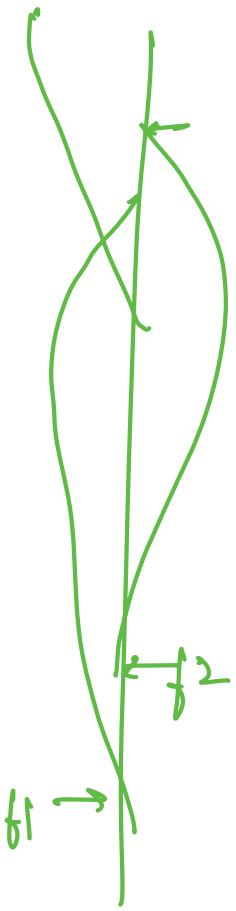
H/W 1

→ Go through project overview document

→ Complete "Main" section of learn.git branching.js.org

→ Git will be completed in next class


f
main
main
f2
f1
f3
main


f2
f1