Git initialization

**Way 1:**

cd ~/nameOfFolder

git clone <Link> <folderName>

Example: git clone <Link> dev1

**Way 2:**

mkdir dev2

cd dev2

git init

git remote add origin <clone with SSH>

**To get the remote changes locally:**

git pull origin master

**To view all files in directory:**

ls -la

**To view history of commits:**  
git log

git log -p -2 (p is patch 2 means last 2 entries)

git log --stat

**To view configuration:**

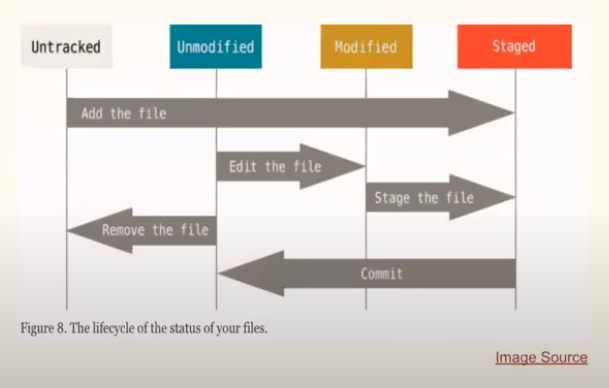
git config --list

**To create a file:**

echo ‘Hello, World’ > hello.md

**To see the status(untracked):**

git status



**To add the file: (Staged condition)**

git add hello.md

git add .

**To commit file:**

git commit -m ‘C1:master Add hello.md’ (-m is message)

**To view the changes occurred in the commit:**

git log -p <hash of the commit from “git log”>

**To commit to server:**

git push origin master(branch name)

**To view local branch:**

git branch

**To view remote or server branch:**

git branch -r

**To inspect a remote server:**

git remote show origin

If we have minor changes which were an accident in the previous commit and we don't want to use another commit then we use the amend command.

git commit --amend -m ‘C1:master Add hello.md’ s

**To ignore certain files:**

nano .gitignore

//file names

node\_modules

.DS\_Store

**Create new local branch:**

git branch b1

**Use b1 branch:**

git checkout b1

**Toggle branch:**

git checkout -

**Shortcut to create a branch and toggle to that branch:**

git branch b11; git checkout b11

These 2 commands become:

git checkout -b b11

**Delete a branch:**

git branch -D b11

**Without committing if I want to save the progress:**

git stash save ‘more change’(give a message like commit)

git status returns a clean working directory.

git stash list

// go to other branch to complete other work.

==>Come back to the node b11 in which the incomplete work was done.

git stash apply

//  if I want to apply other stash then copy from stash list:

git stash apply stash@{1}

**To delete stash:**

git stash drop stash@{1}

**To apply a stash and delete the stash:**

git stash pop

git fetch origin; git merge master -> git pull origin master

**Ignoring Files:**

Nano .gitignore

**Removing Files:**

git rm PROJECTS.md

**Renaming Files:**

git mv file\_from file\_to

**To create Tag:**

git tag -a v1.4 -m "my version 1.4"

**To view tag:**

git tag

git show v1.4

**Sharing Tags**:

git push origin v1.5

Multiple tags:

git push origin --tags

**Deleting Tags:**

git tag -d v1.4

**Show where branch pointers are pointing:**

git log --oneline --decorate

**history of your commits, showing where your branch pointers are and how your history has diverged:**

git log --oneline --decorate --graph --all

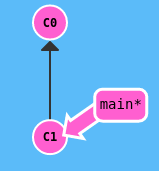
**Graphical merge tool:**

git mergetool

**GAME**

**https://learngitbranching.js.org/?**

1. git commit
2. git checkout -b bugFix



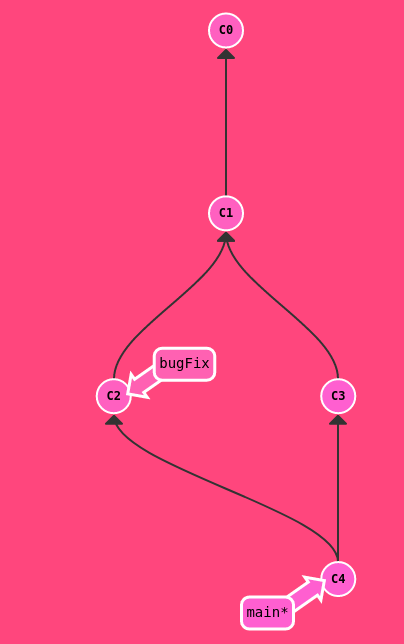
1. git checkout -b bugFix

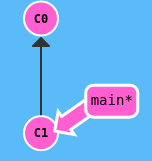
git commit

git checkout main

git commit

git merge bugFix



1. 

git checkout -b bugFix

git commit

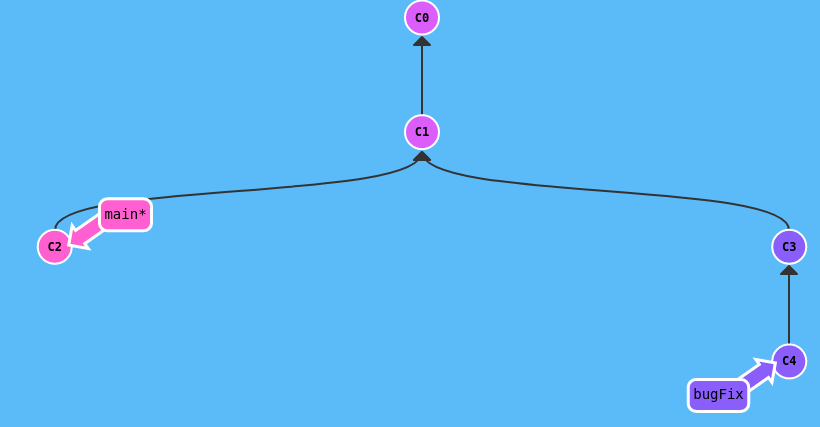
git checkout main

git commit

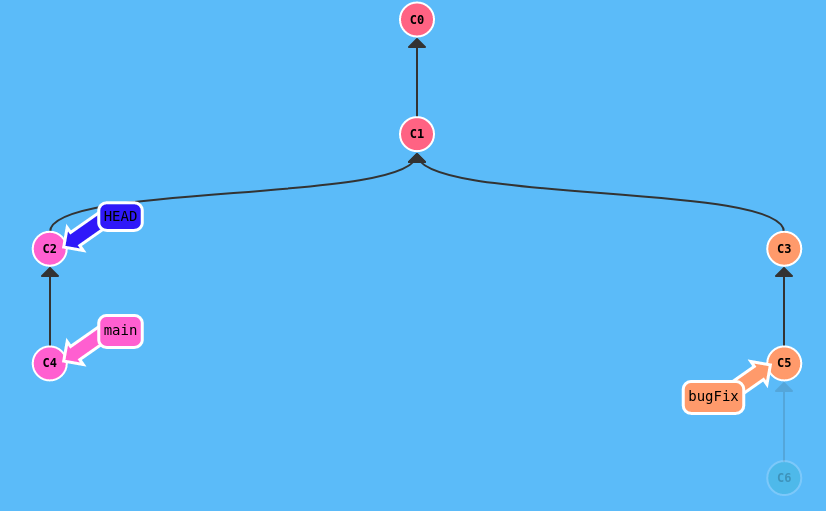
git checkout bugFix

git rebase main

1. git checkout C4



Git checkout C4^

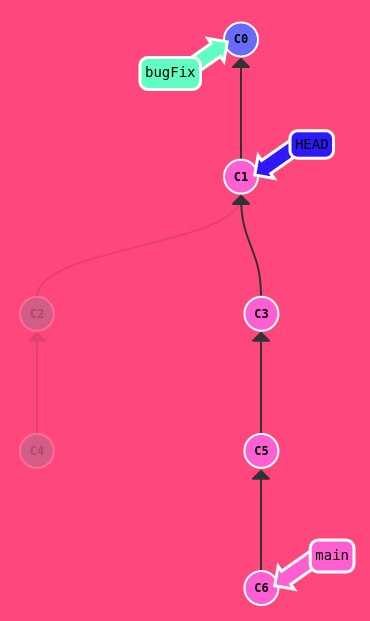


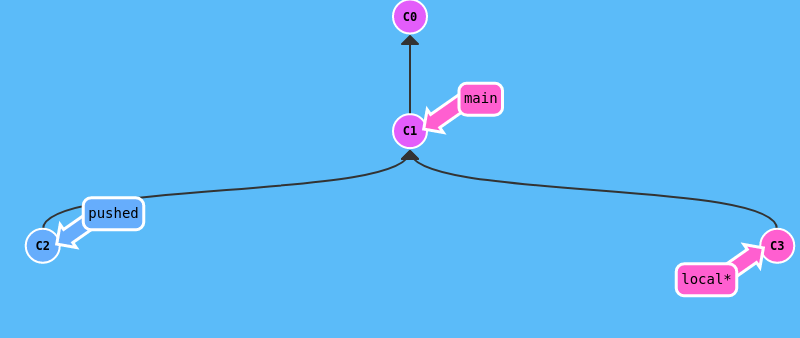
git branch -f bugFix HEAD~3

git checkout C6

git branch -f main HEAD~0

Git checkout C1

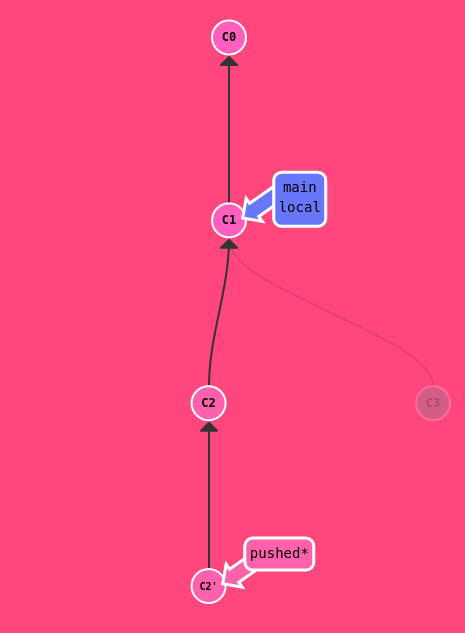




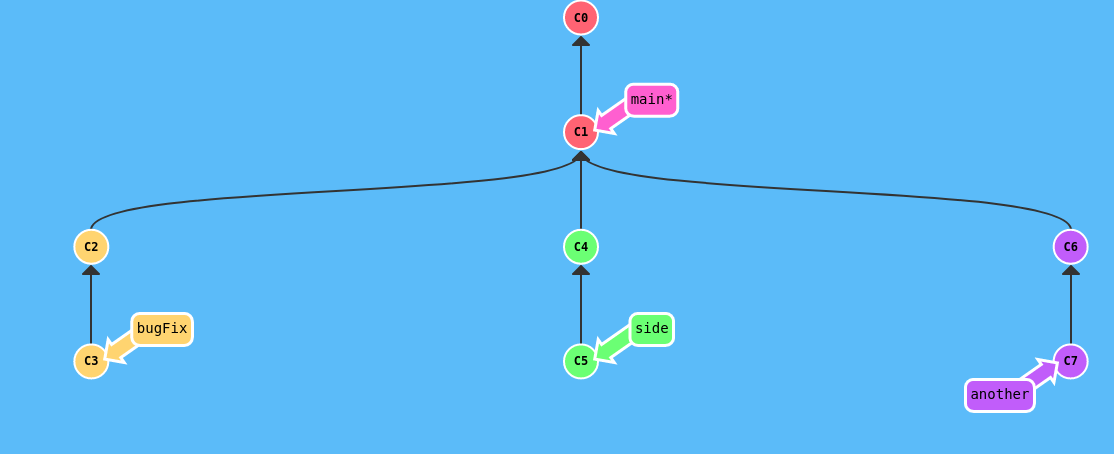
Git reset HEAD~1

Git checkout pushed

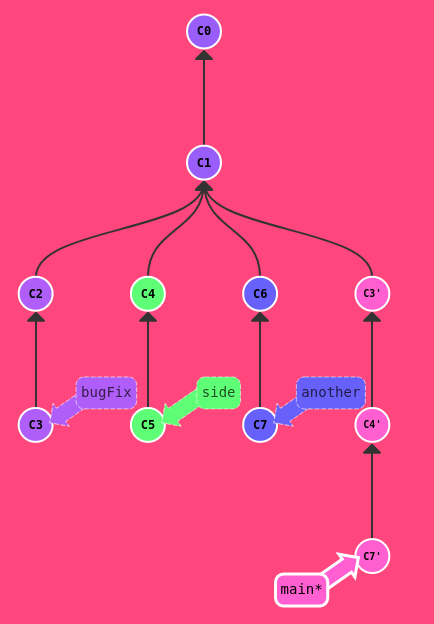
Git revert HEAD



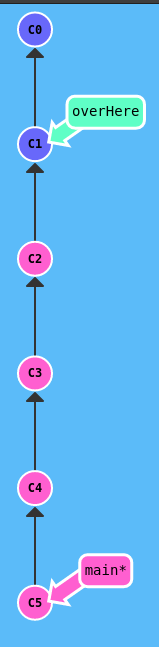
9)



Git cherry-pick C3 C4 C7

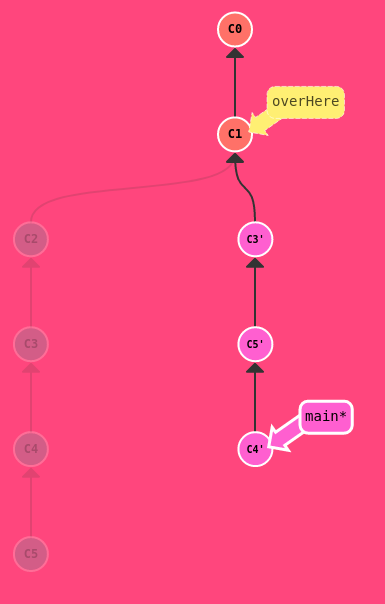


10)

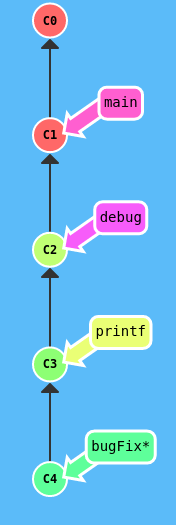


git rebase -i HEAD~4

Then select and arrange according to 3,5,4

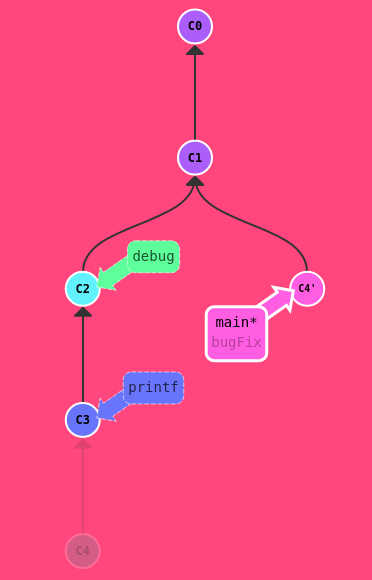


11)

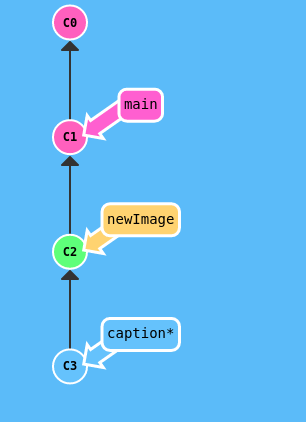


Git checkout main

Git cherry-pick c4



12)



Git rebase -i HEAD~2

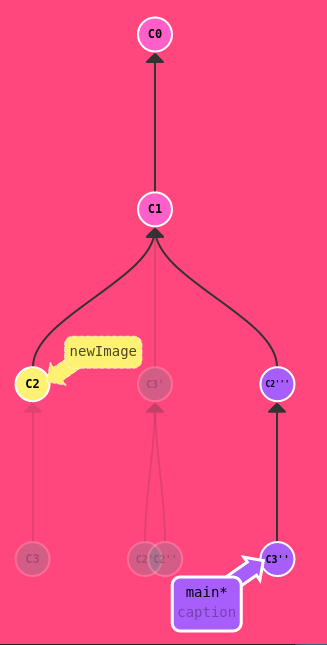
== rearrange 3,2

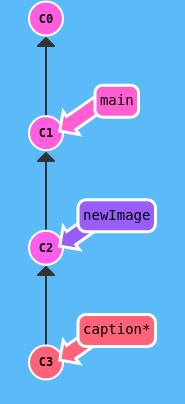
Git commit --amend

Git rebase -i HEAD~2

==rearrange 2,3

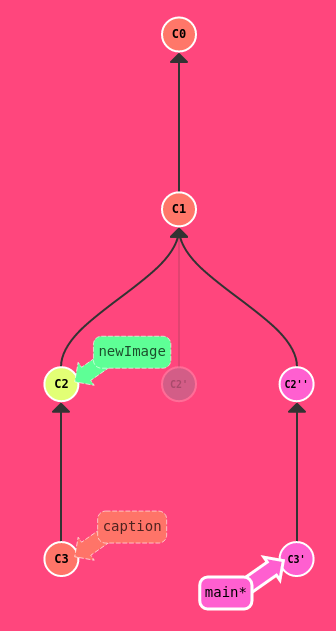
Git branch -f main HEAD

13)

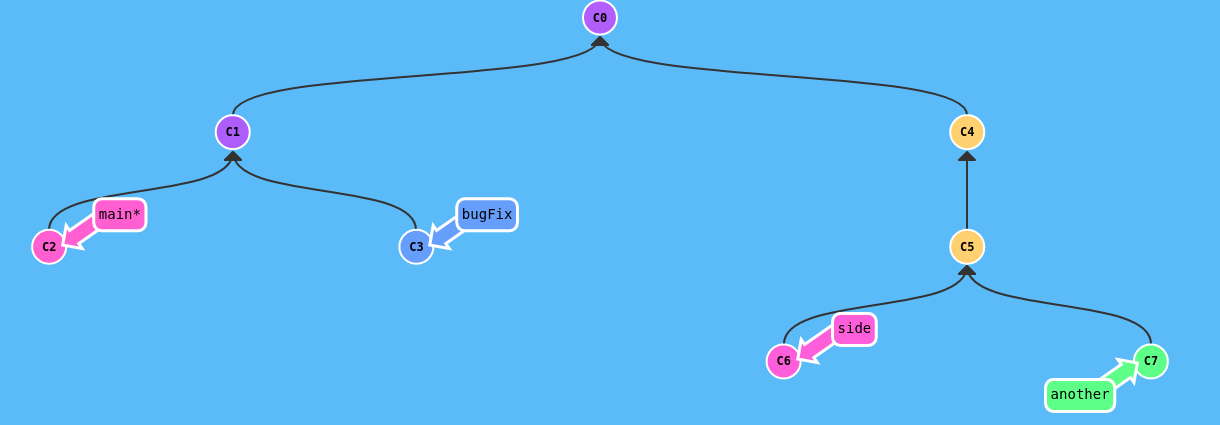


Git checkout main

Git cherry-pick c2 c3



14)

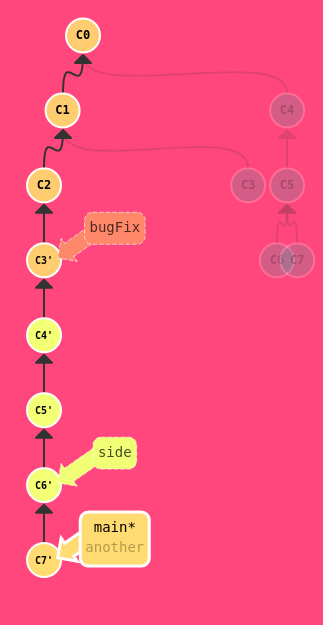


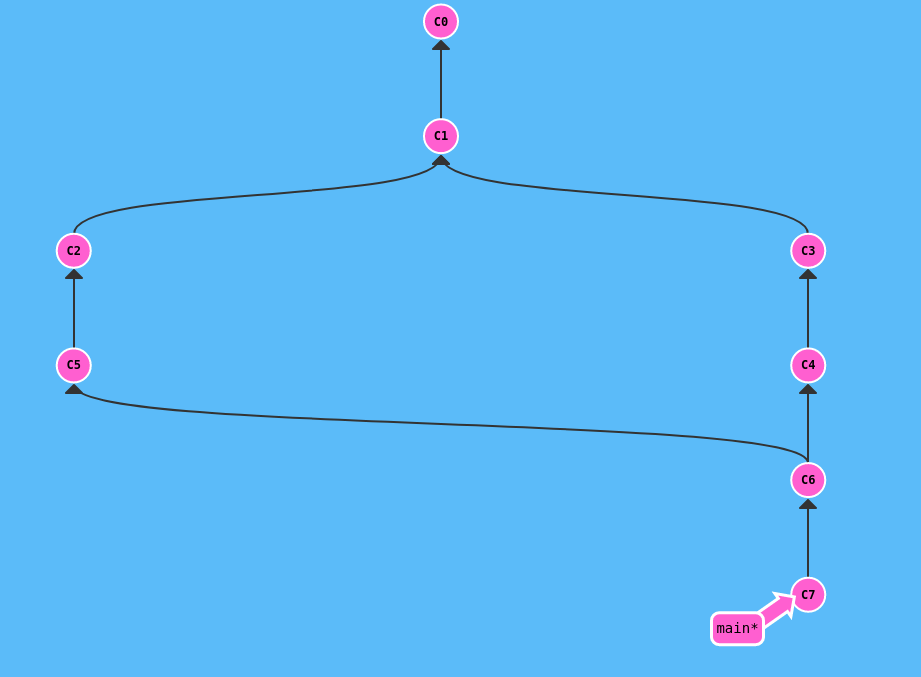
Git rebase main bugFix

Git rebase bugFix side

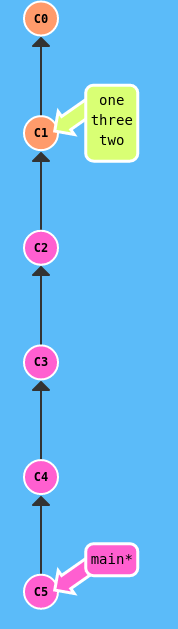
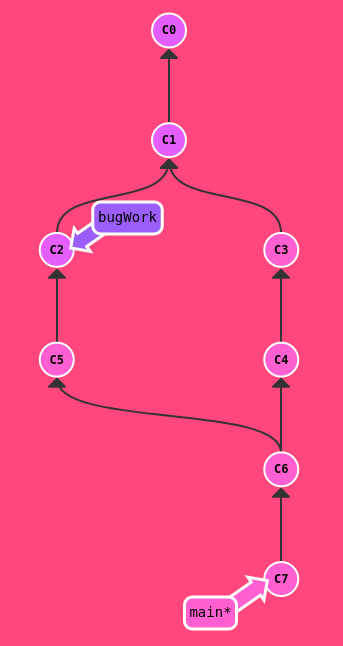
Git rebase side another

Git rebase another main





Git branch bugWork main ^^2^



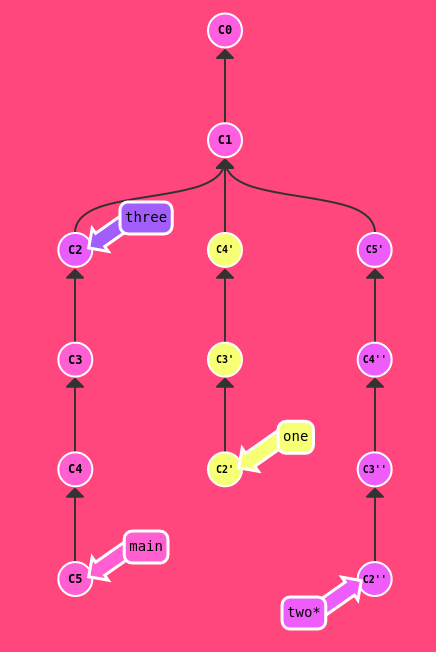
Git checkout one

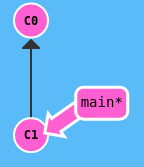
Git cherry-pick C4 C3 C2

Git  checkout two

Git cherry-pick C5 C4 C3 C2

Git branch -f three C2//git rebase C2 three



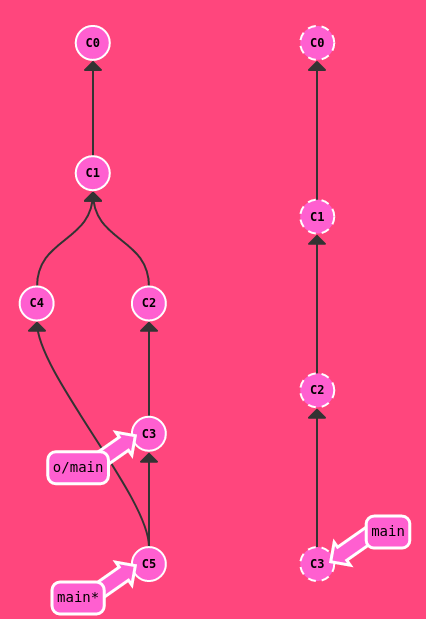


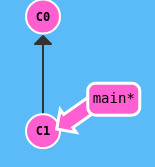
Git clone

Git fakeTeamwork 2

Git commit

Git pull





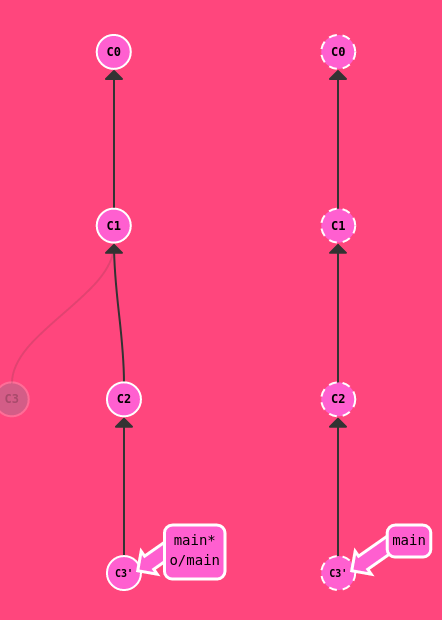
Git clone

Git fakeTeamwork

Git commit

Git pull --rebase

Git push

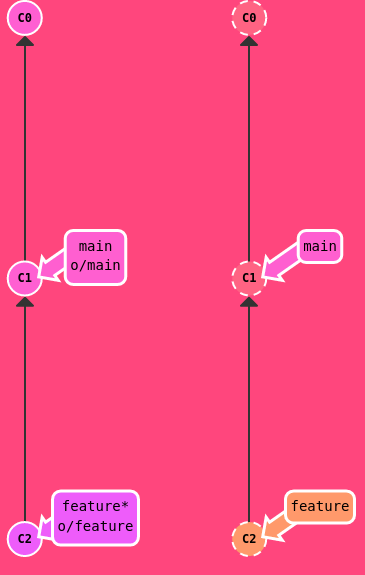


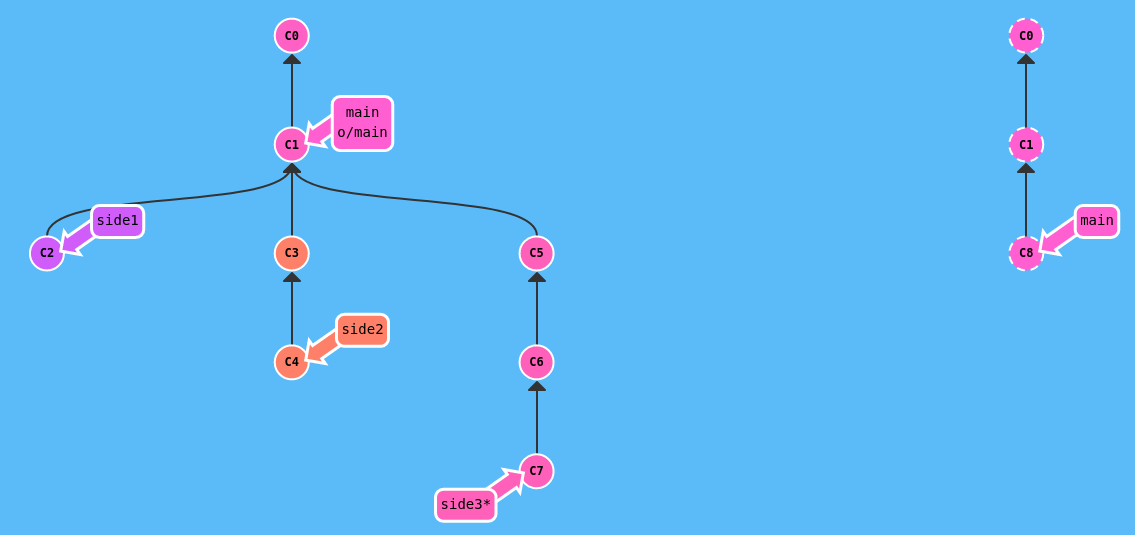


Git reset o/main

Git checkout -b feature

Git push origin feature





Git fetch

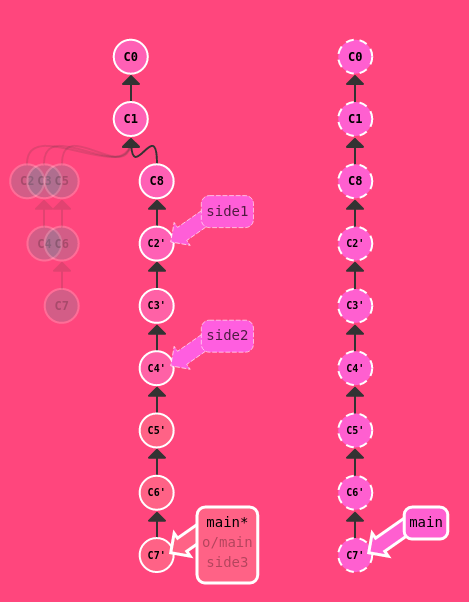
Git rebase o/main side1

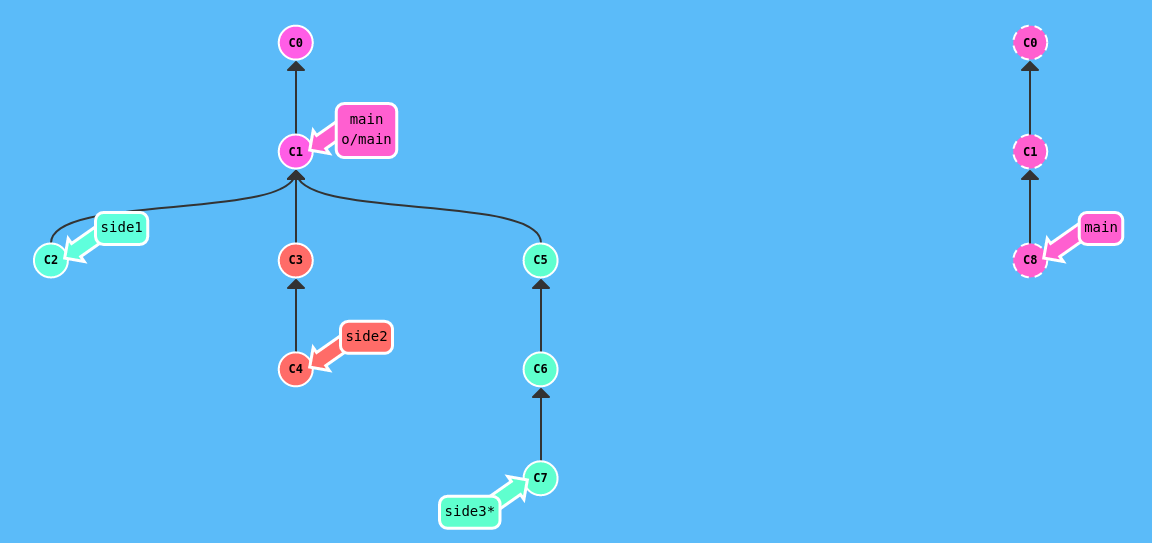
Git rebase side1 side2

Git rebase side2 side3

Git rebase side3 main

Git push





Git checkout main

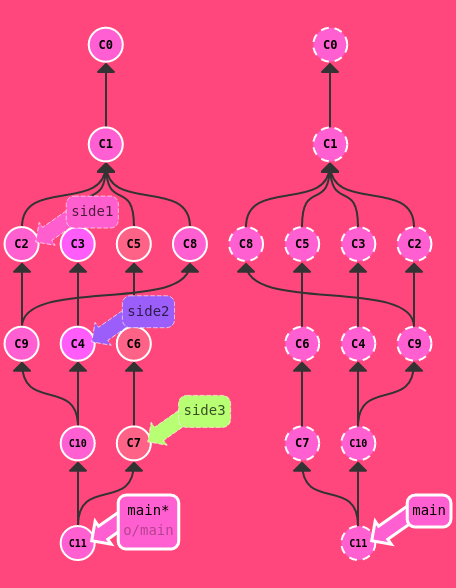
Git pull

Git merge side1

Git merge side2

Git merge side3

Git push



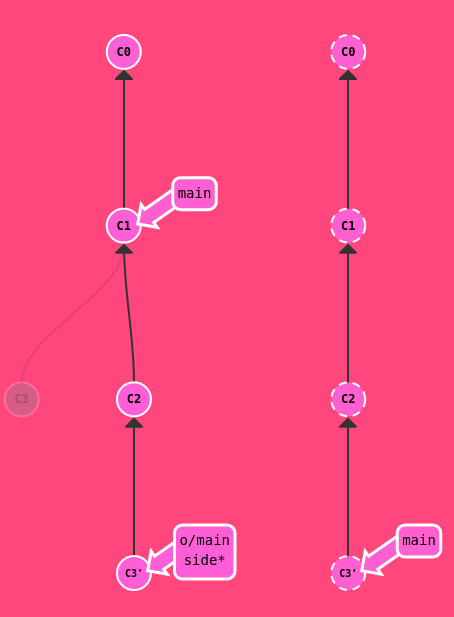


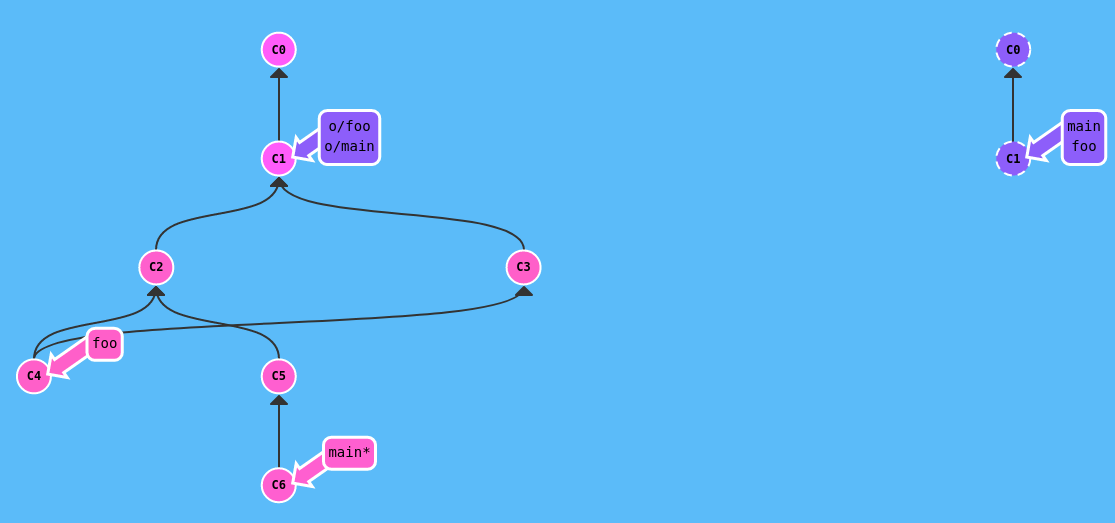
Git checkout -b side o/main

Git commit

Git pull-rebase

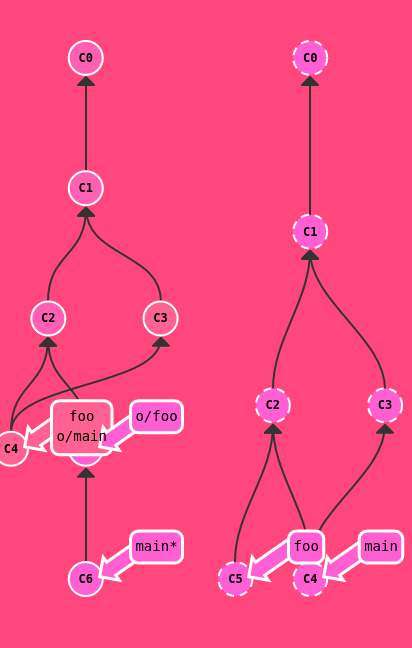
Git push

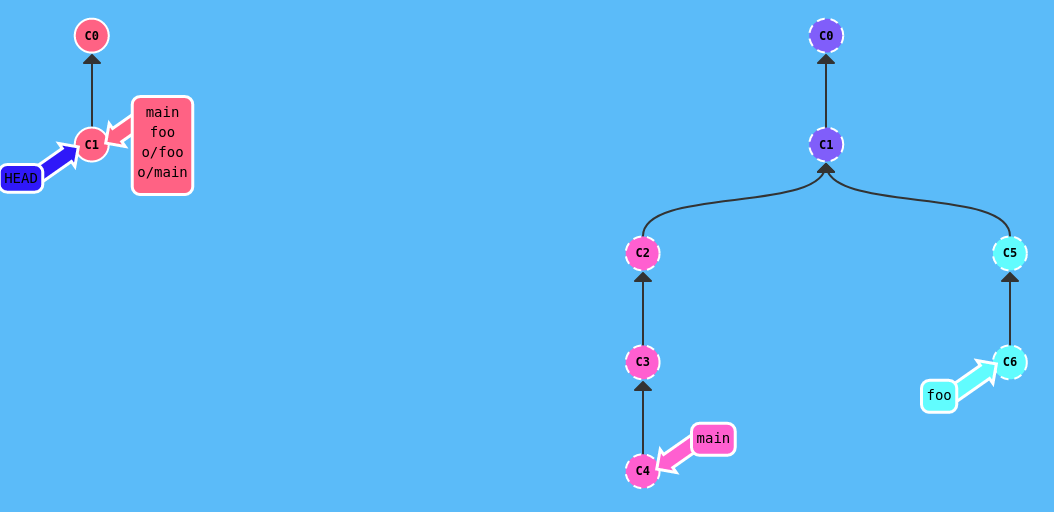




Git push origin main^:foo

Git push origin foo:main





Git fetch origin foo:main

Git fetch origin main~1:foo

Git checkout foo

Git merge main

