CI/CD Exercise 1

Stefan Penzinger - S2310455002

Table of Contents

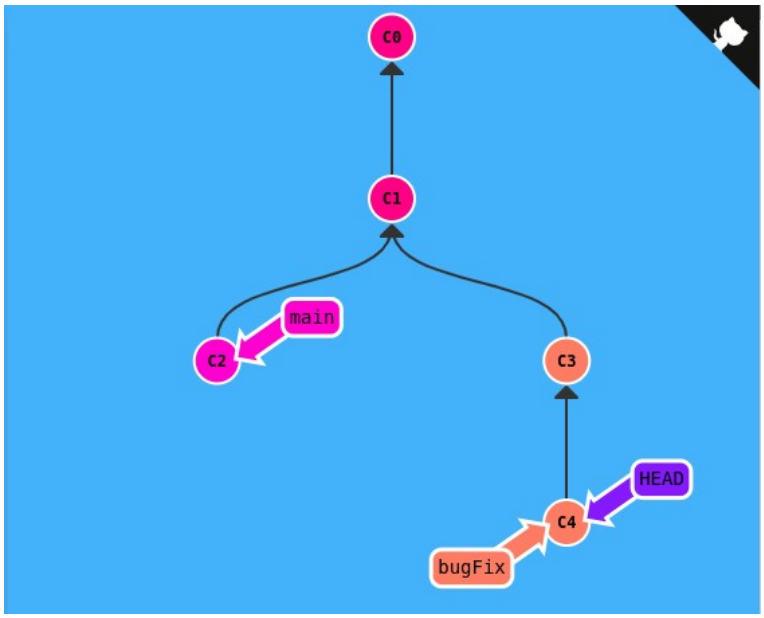
Part 1 - GIT Basics

Ramping Up

1: Detach yo' HEAD

Commands:

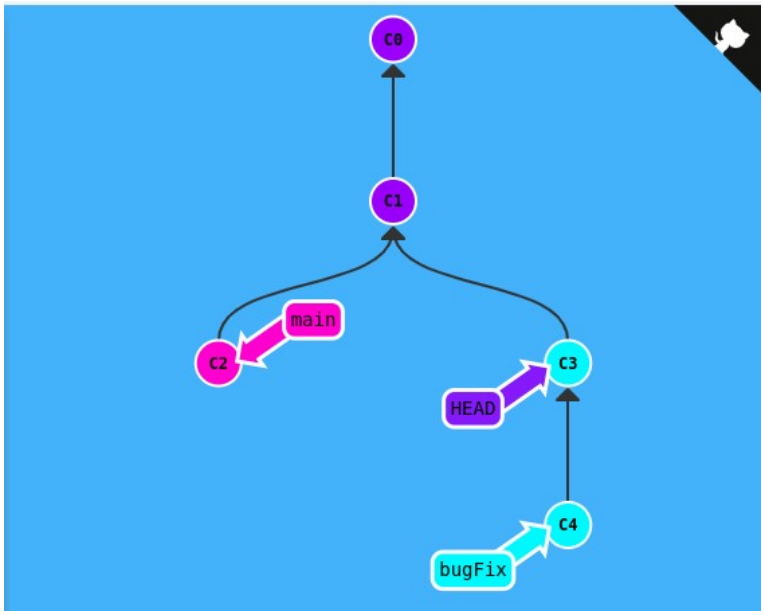
- git checkout C4



2: Relative Refs (^)

Commands:

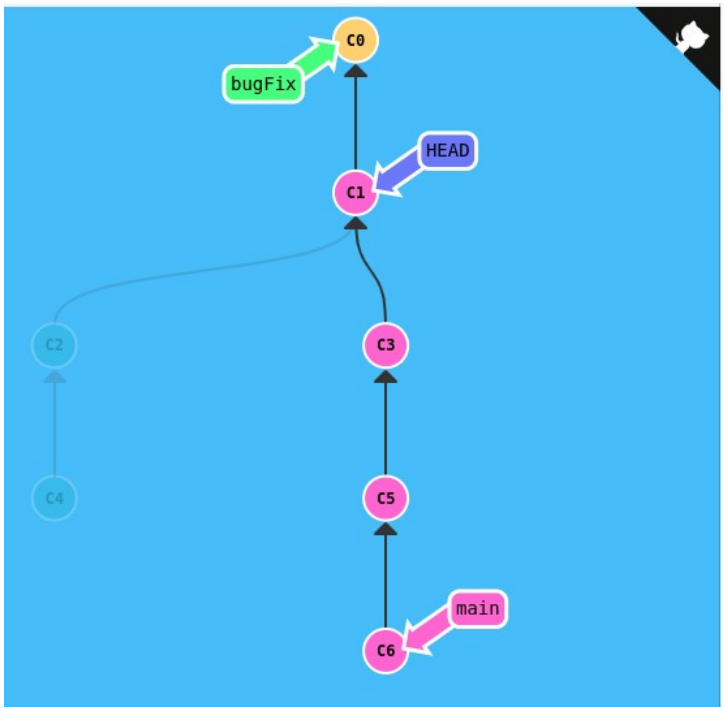
- git checkout bugFix^



3: Relative Refs #2 (~)

Commands:

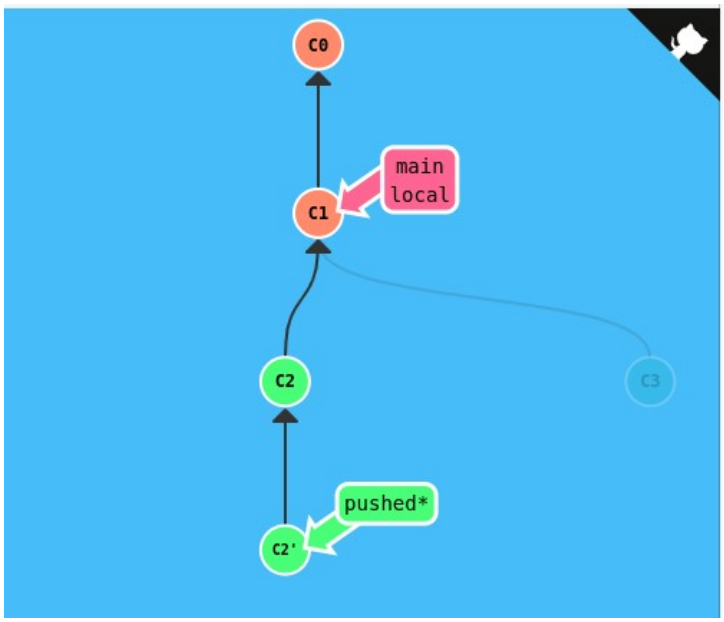
- git branch -f main C6
- git checkout HEAD^
- git branch -f bugFix HEAD^



4: Reversing Changes in Git

Comamnds:

- git reset local^
- git checkout pushed
- git revert pushed

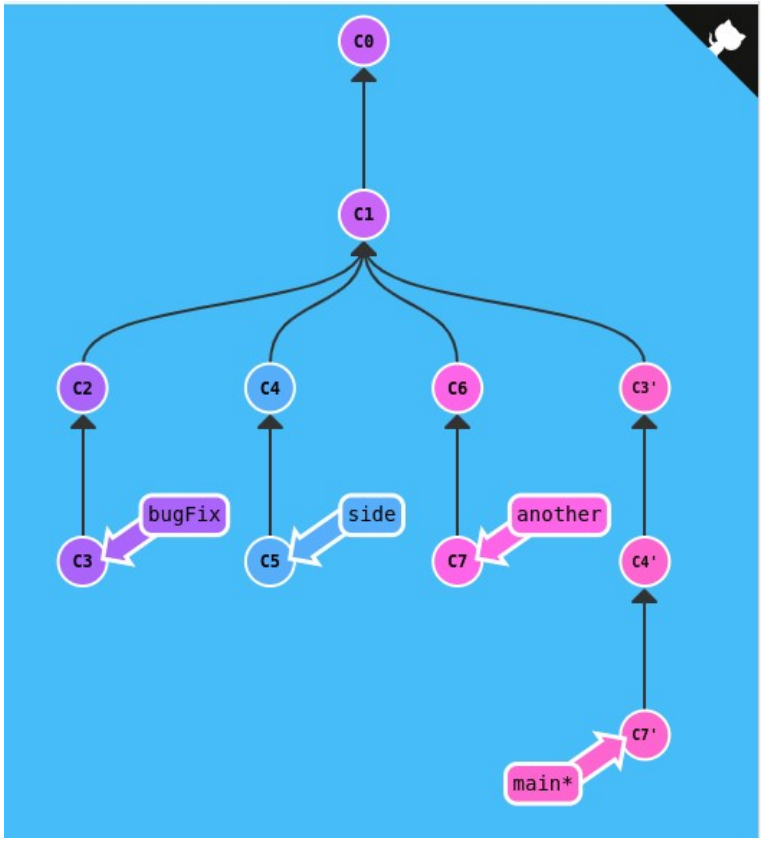


Moving Work Around

1: Cherry-pick Intro

Commands

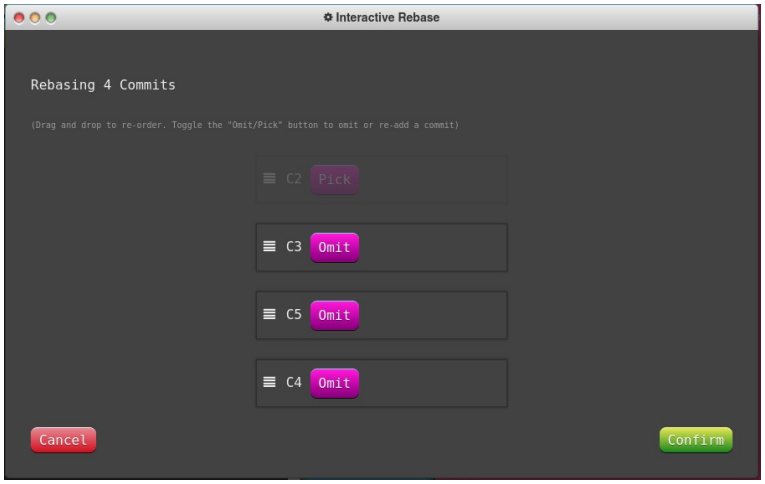
- git cherry-pick bugFix side^ another

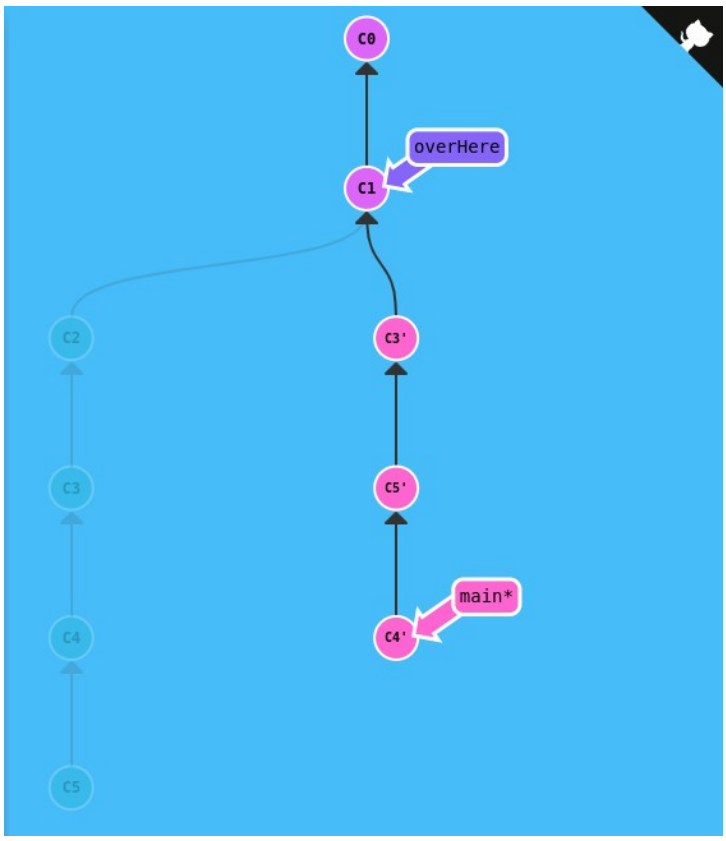


2: Interactive Rebase Intros

Commands:

- git rebase overHere -i



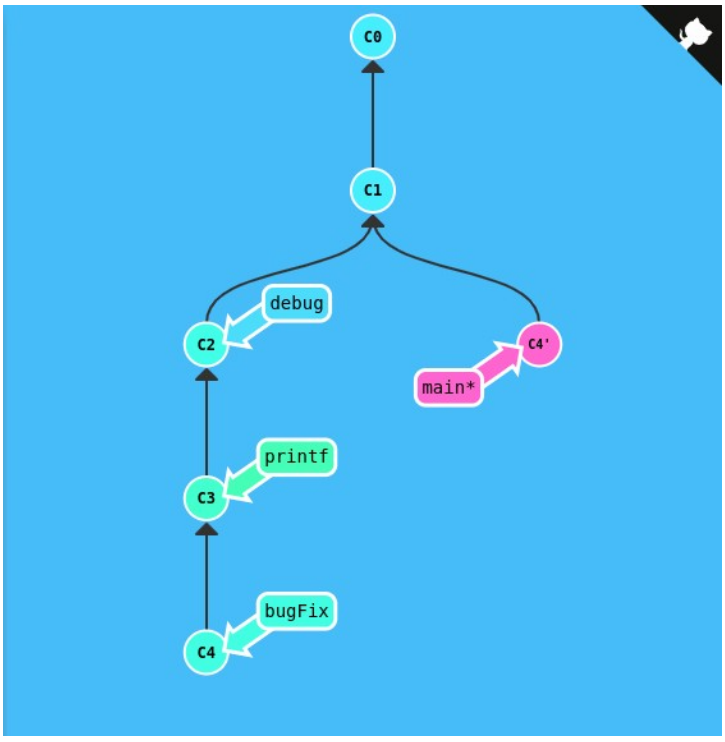


A Mixed Bag

1: Grabbing Just 1 Commit

Commands:

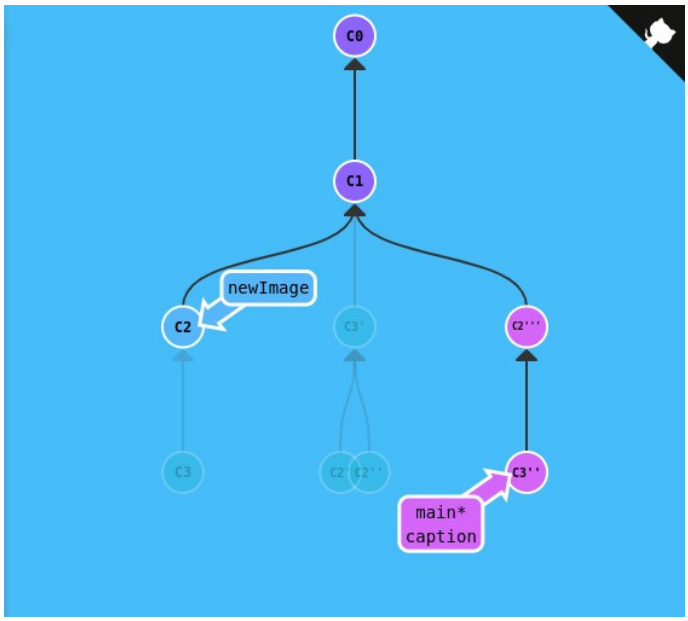
- git checkout main
- git cherry-pick bugFix



2: Juggling Commits

Commands:

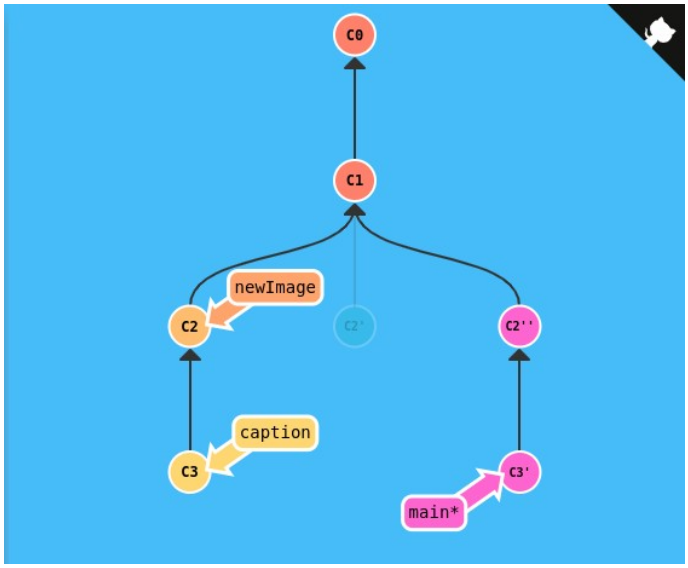
- git rebase main caption -i
- git commit –amend
- git rebase main caption -i
- git checkout main
- git rebase caption



3: Juggling Commits #2

Command:

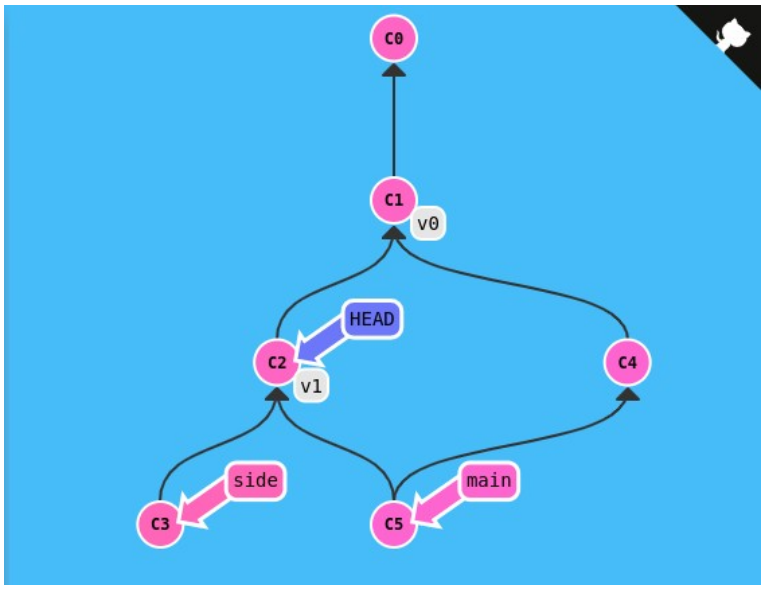
- git checkout main
- git cherry-pick newImage
- git commit –amend
- git cherry-pick caption



4: Git Tags

Commands:

- git tag v0 C1
- git tag v1 C2
- git checkout v1



5: Git Describe

Commands:

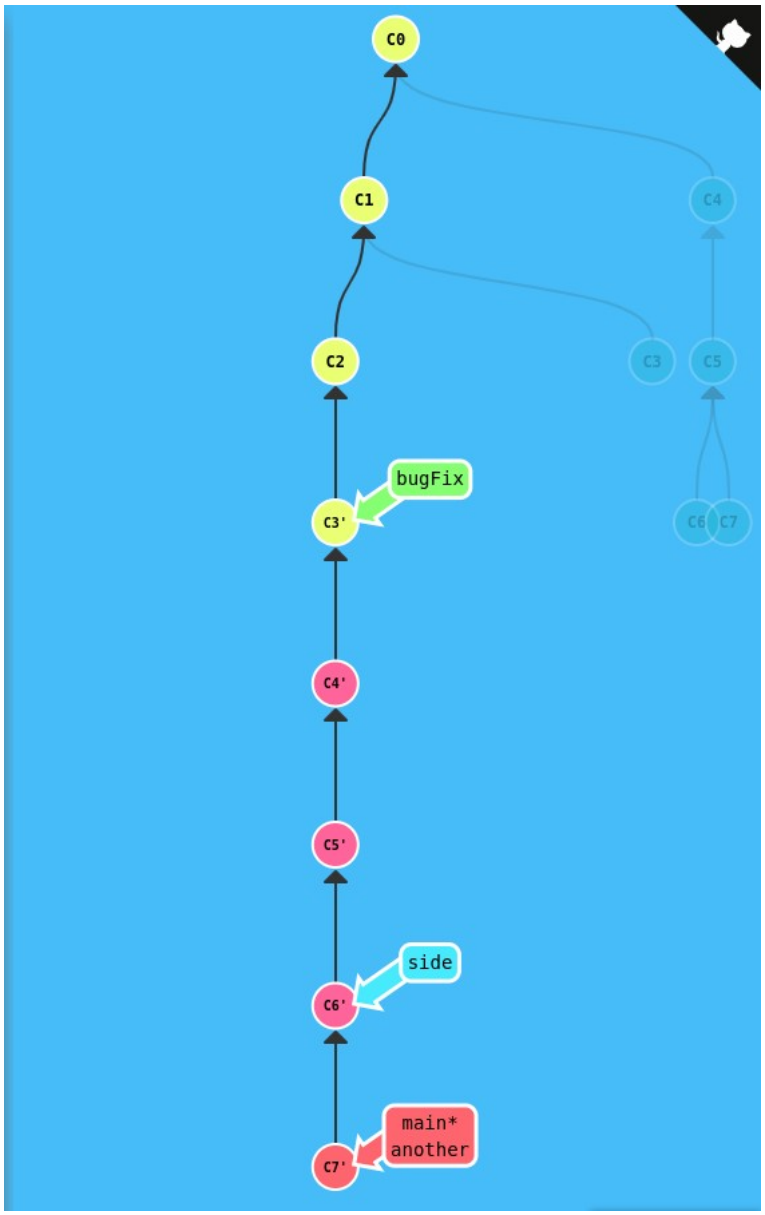
- git describe main
  - v0_2_gC2
- git describe side
  - v1_1_gC4
- git describe bugFix
  - v1_2_gC6

Advanced Topics

1: Rebasing over 9000 times

Commands:

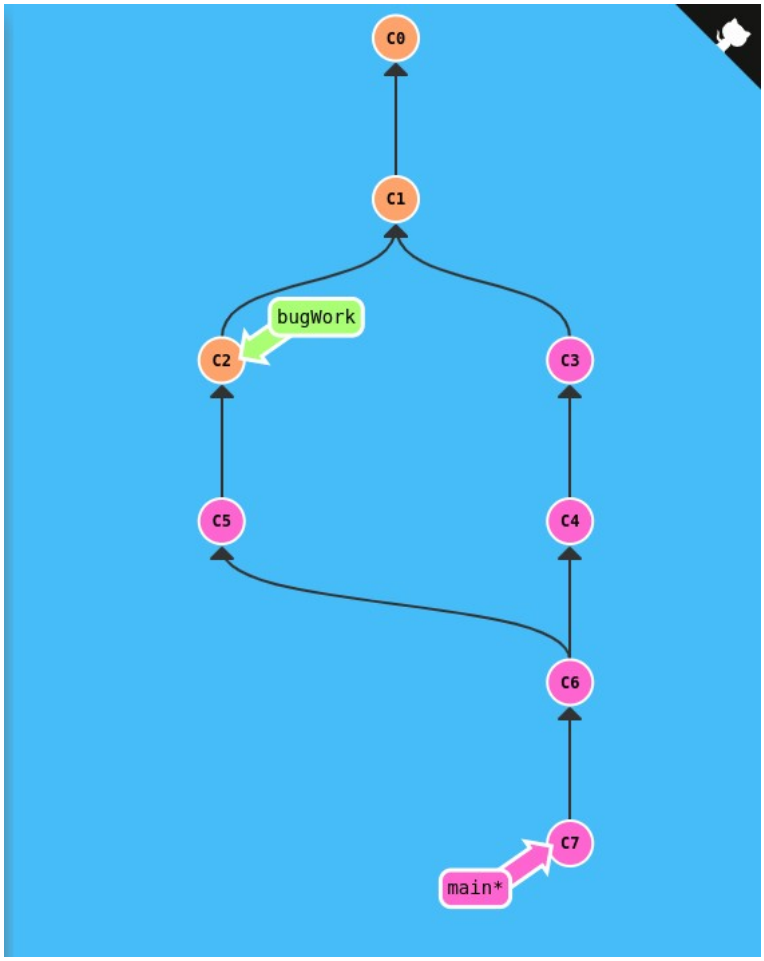
- git rebase main bugFix
- git rebase bugFix another^
- git rebase HEAD side
- git rebase side another
- git rebase another main



2: Multiple parents

Commands:

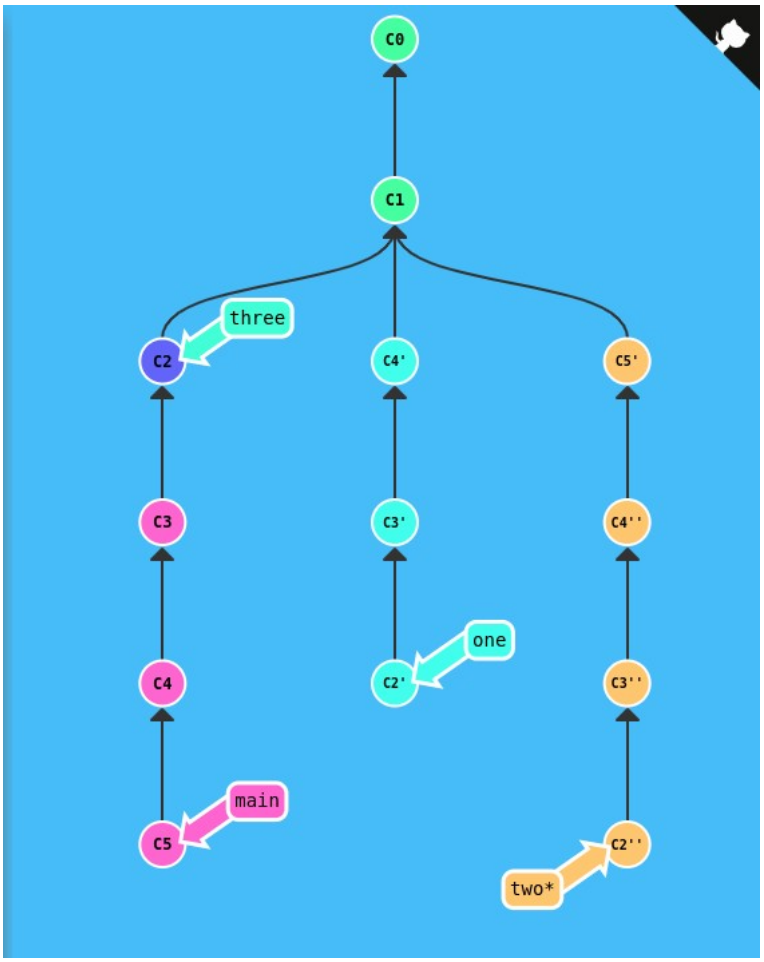
- git branch bugWork HEAD~^2^

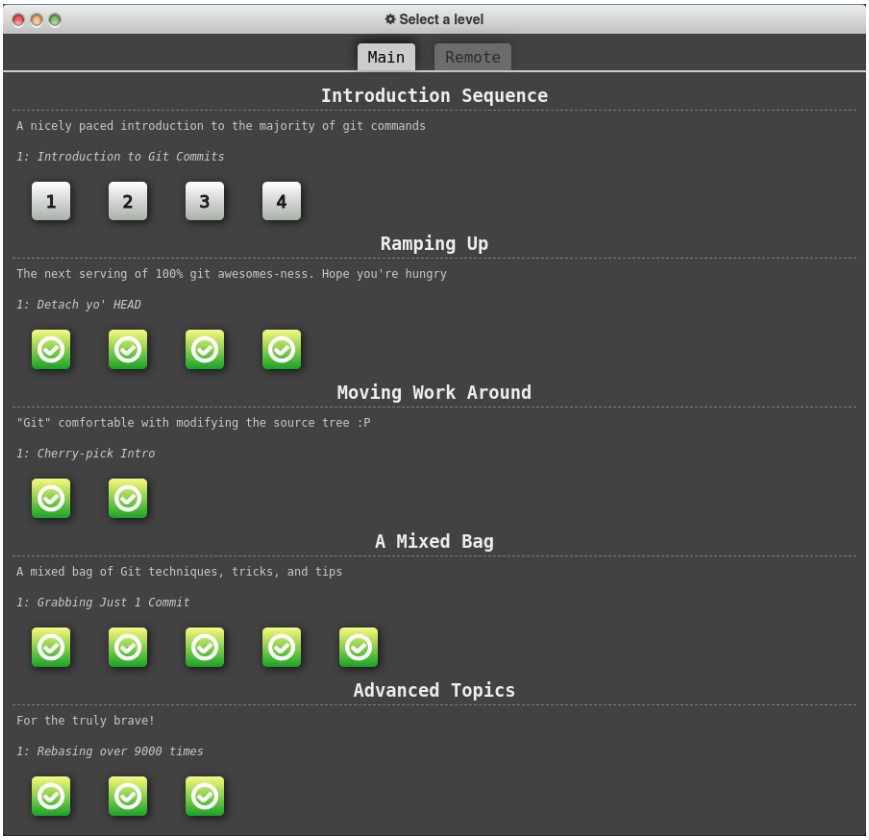


3: Branch Spaghetti

Commands:

- git checkout 3
- git reset main~3
- git checkout one
- git cherry-pick main~ main~2 three
- git checkout two
- git cherry-pick main one~2 one~ one





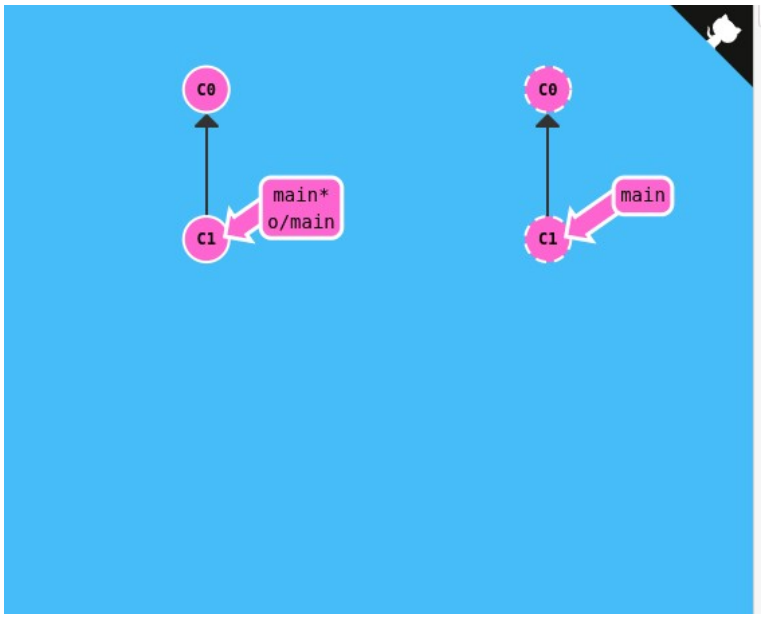
Part 2 - GIT Remotes

Push & Pull -- Git Remotes!

1: Clone Intro

Commands:

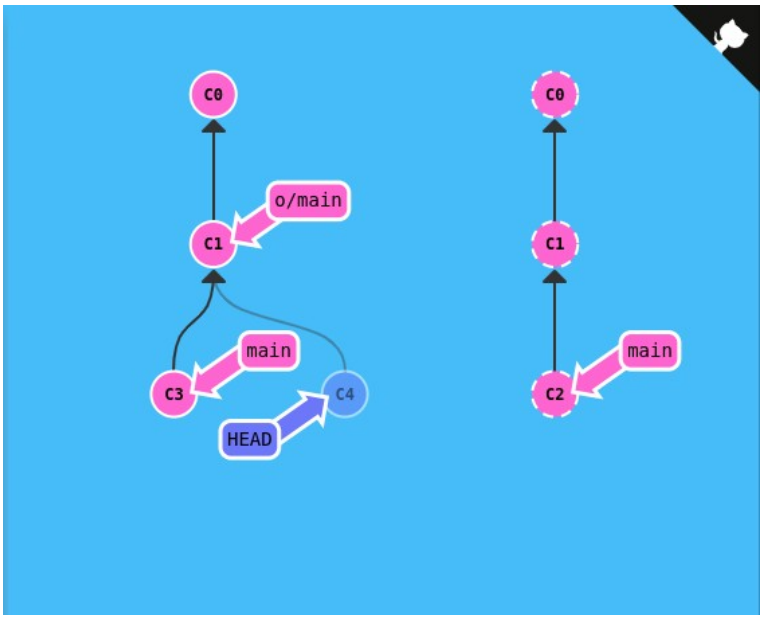
- git clone



2: Remote Branches

Commands:

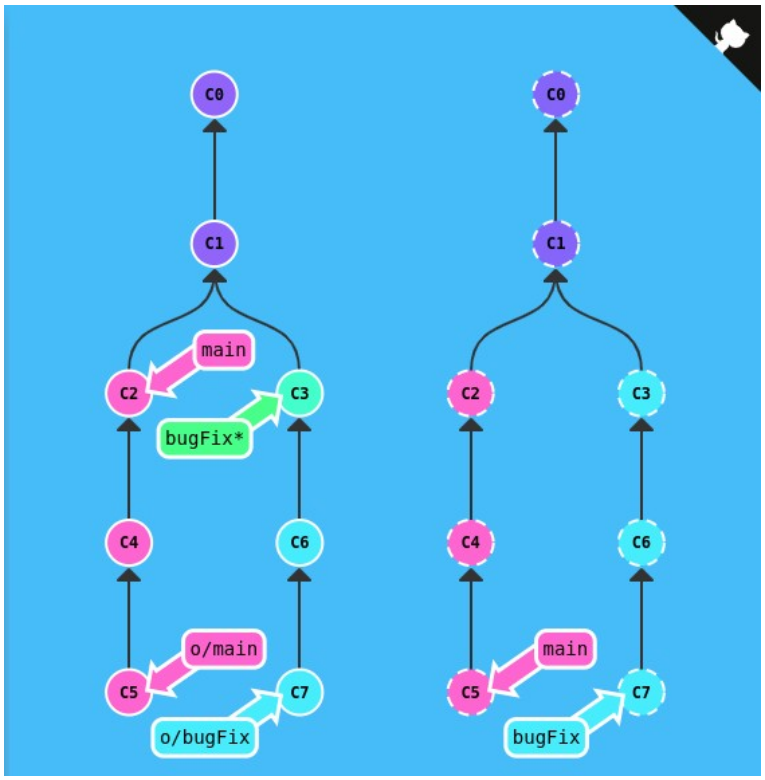
- git commit
- git checkout o/main
- git commit



3: Git Fetchin’

Command:

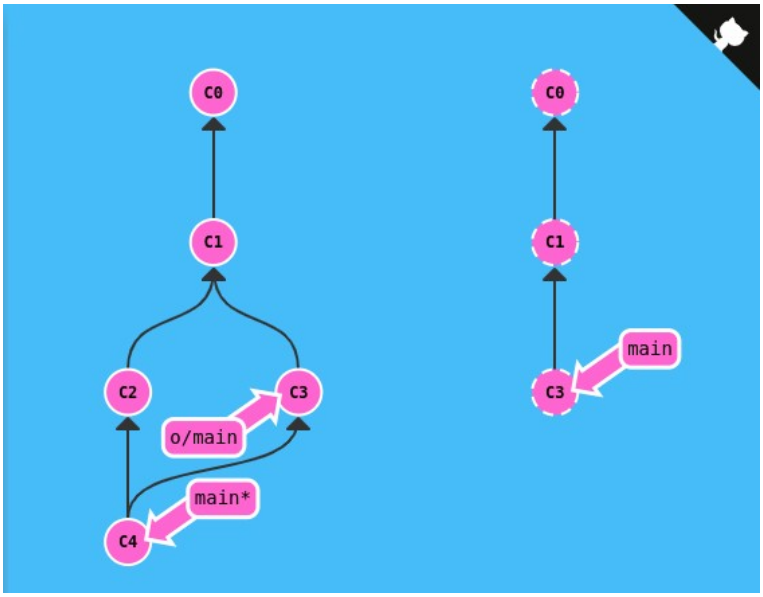
- git fetch



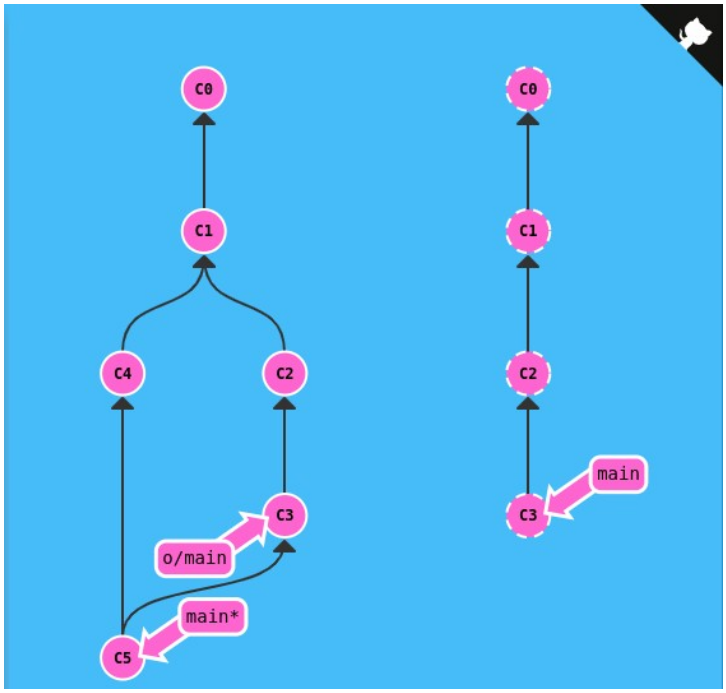
4: Git Pulin’

Command:

- git pull



5: Faking Teamwork



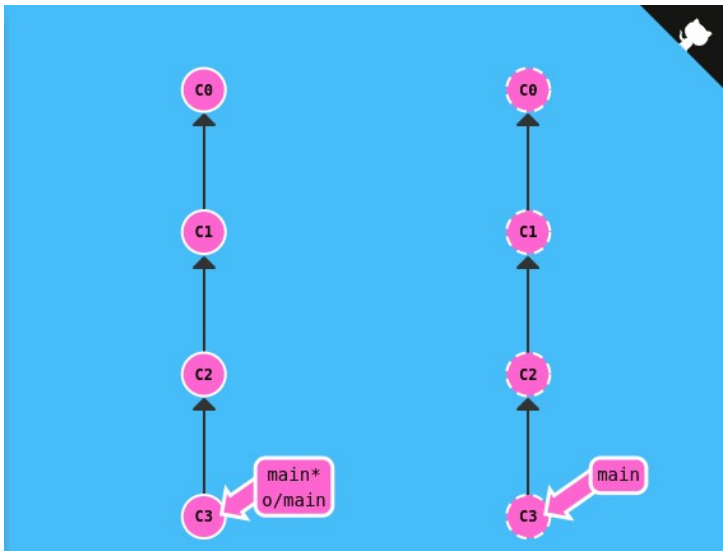
Commands:

- git clone
- git fakeTeamwork 2
- git commit
- git pull

6: Git Pushin’

Commands:

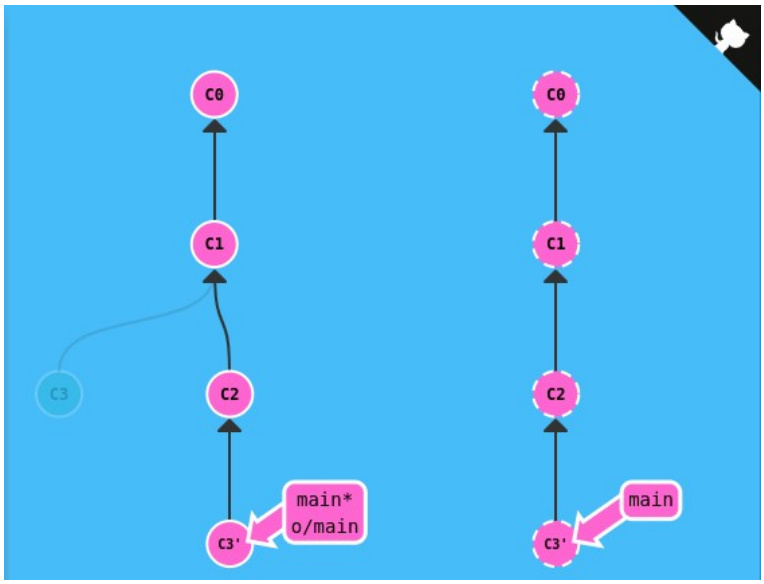
- git commit
- git commit
- git push



7: Diverged History

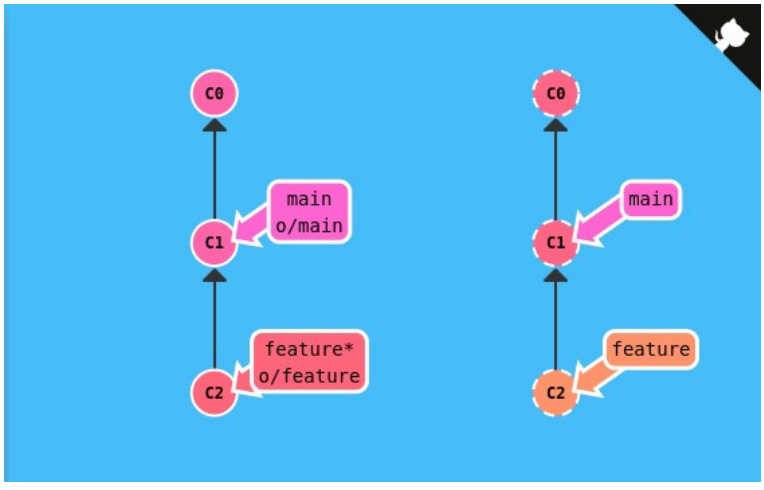
Commands:

- git clone
- git fakeTeamwork
- git commit
- git pull –rebase
- git push



8: Locked Main

- git switch -c feature
- git push
- git checkout main
- git reset HEAD^
- git checkout feature

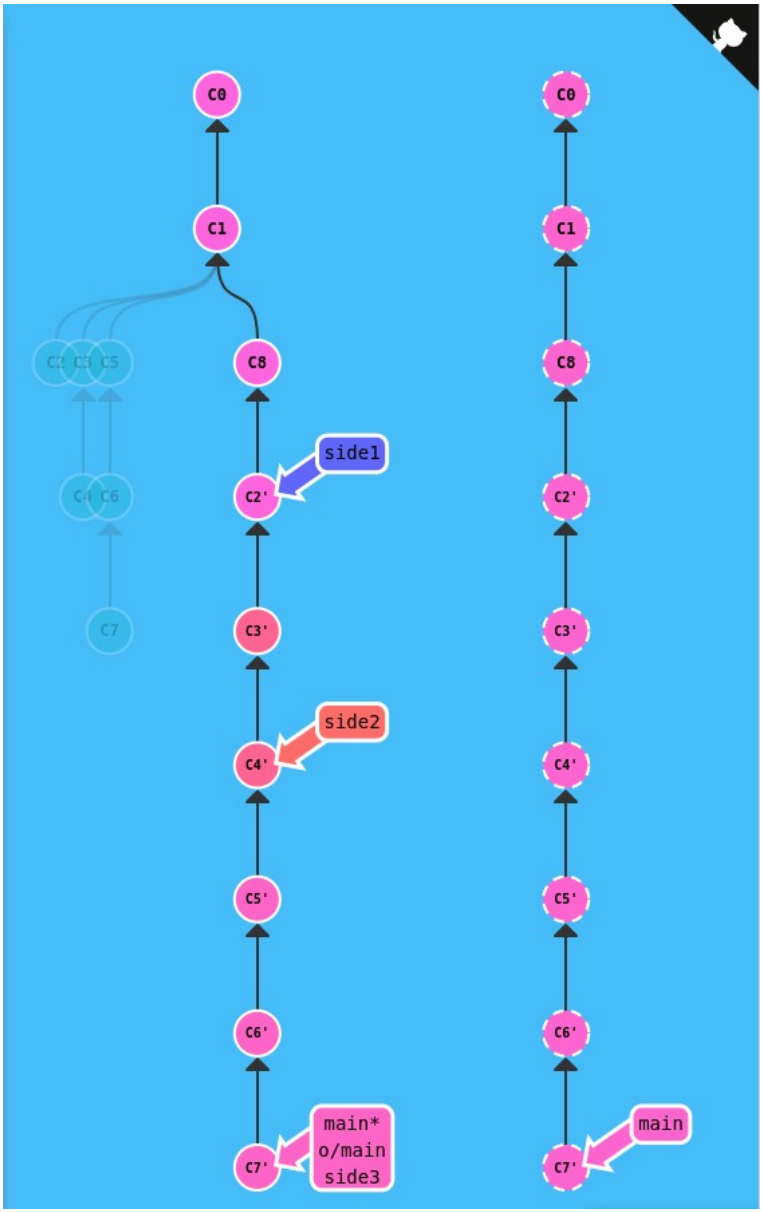


To Origin And Beyond -- Advanced Git Remotes!

1: Push Main!

Commands:

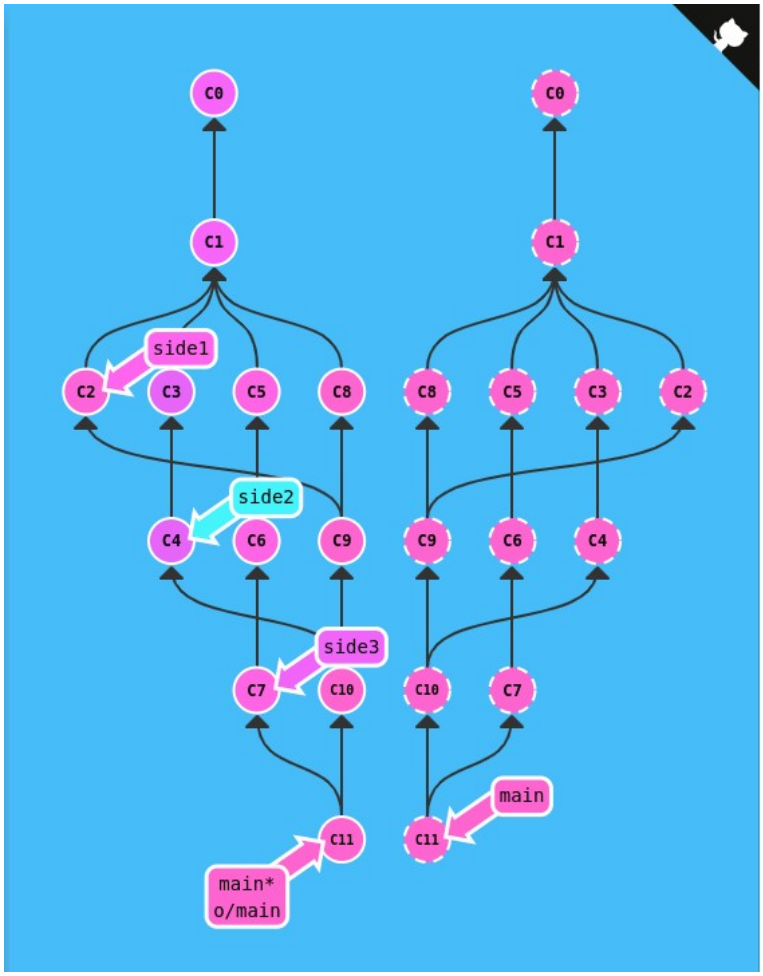
- git fetch
- git rebase o/main side1
- git rebase side1 side2
- git rebase side2 side3
- git rebase side3 main
- git push



2: Merging with remotes

Commands:

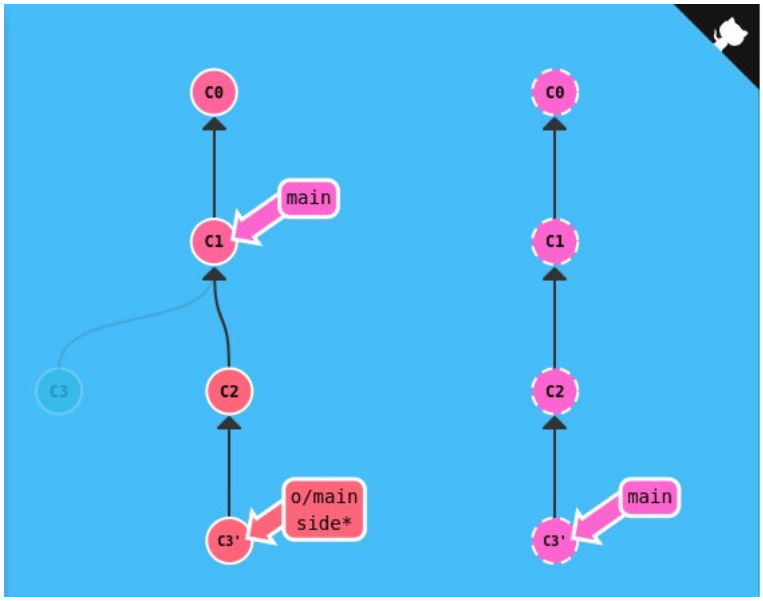
- git checkout main
- git pull –rebase
- git merge side1
- git merge side2
- git merge side3
- git push



3: Remote Tracking

Commands:

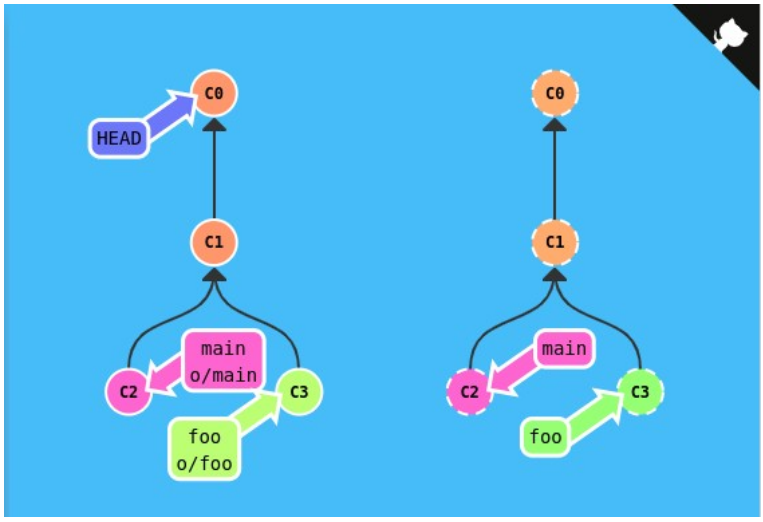
- git checkout -b side o/main
- git commit
- git pull –rebase
- git push



4: Git push arguments

Commands:

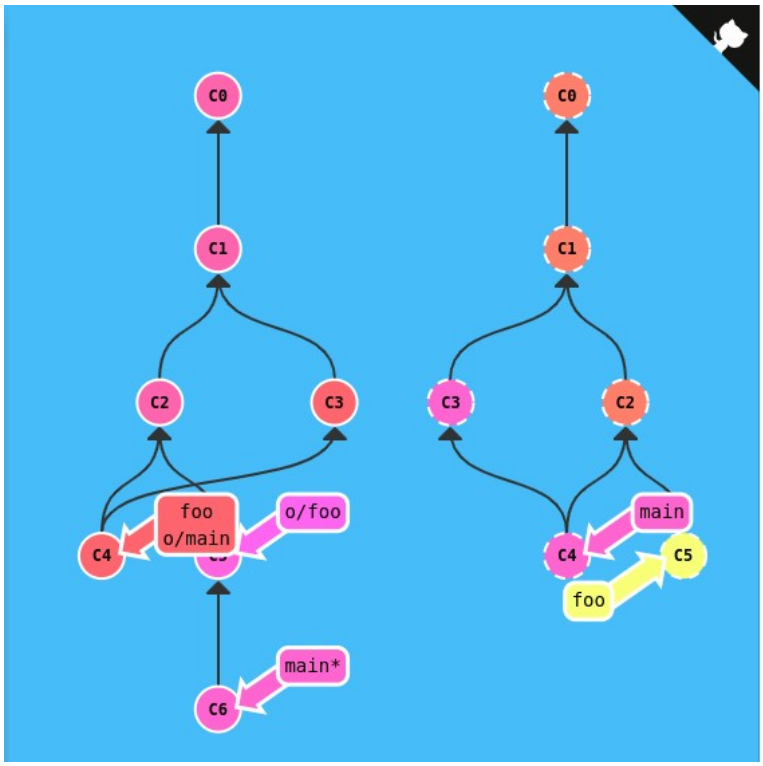
- git push origin main
- git push origin foo



5: Git push arguments – Expanded!

Commands:

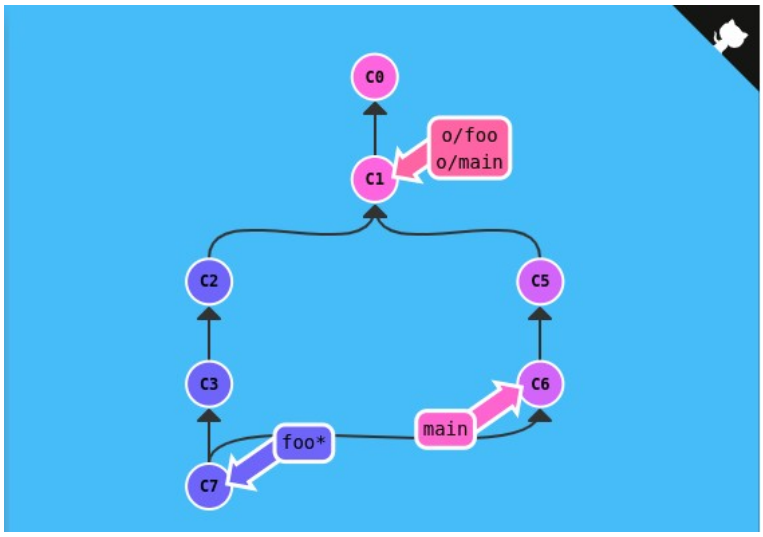
- git push origin foo:main
- git push origin main^:foo



6: Fetch Arguments

Commands:

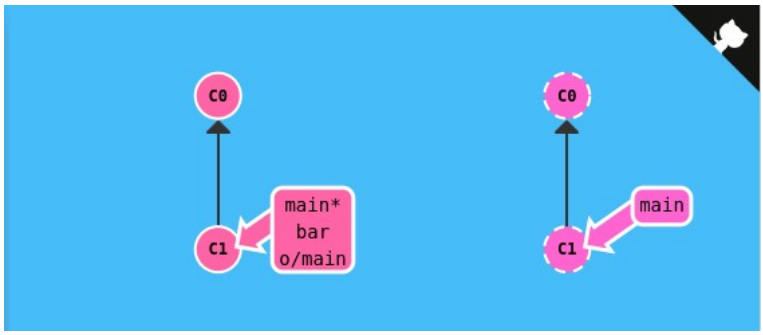
- git fetch origin main^:foo
- git fetch origin foo:main
- git checkout foo
- git merge main



7: Source of nothing

Commands:

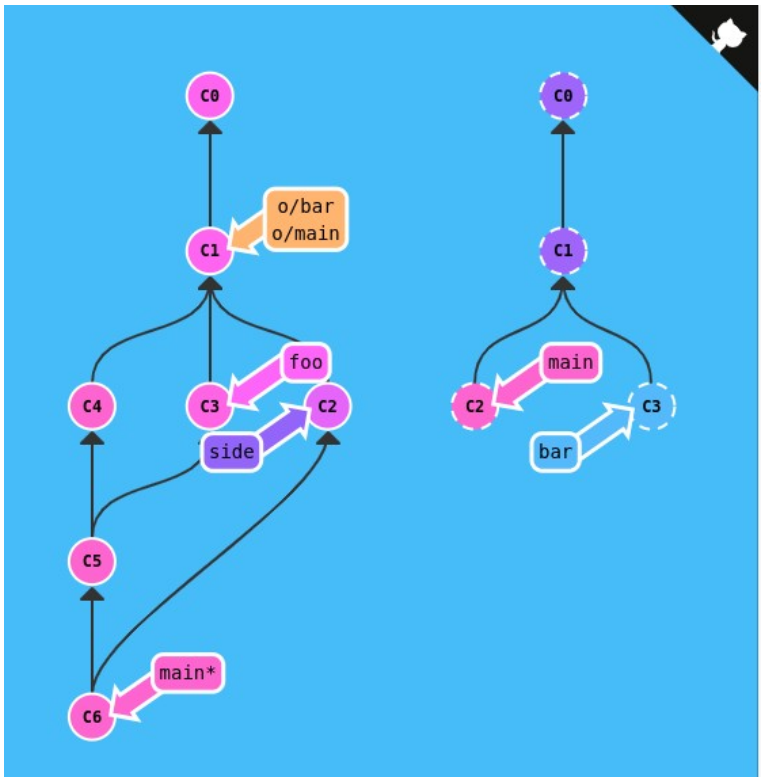
- git push :foo
- git fetch origin :bar



8: Pull arguments

Commands:

- git pull origin bar:foo
- git pull origin main:side



Part 3 - GIT Interactive Rebase

Checkout topic and start interactive rebase

```
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git c topic
Branch 'topic' set up to track remote branch 'topic' from 'origin'.
Switched to a new branch 'topic'
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git c master
hint: Waiting for your editor to close the file...
[detached HEAD 11dfa22] commit B + C
 Author: Marc Kurz <marc.kurz@fh-hagenberg.at>/cd2020-ex01$ git rebase -i topic
```

Changing the commit history

```
 GNU nano 6.2
pick 940e726 commit B
pick 56ad339 commit C
pick f5a0a7f commit D
pick a5730d8 commit E
pick 2ba1b96 commit F
pick 0770d33 commmit G
pick d780c7f commit H
pick fe771ee updated year
pick 64049ed Update README.md

# Rebase 6ed4610..64049ed onto 6ed4610 (9 commands)
```

```
 GNU nano 6.2
pick 940e726 commit B
squash 56ad339 commit C
edit f5a0a7f commit D
pick 2ba1b96 commit F
pick a5730d8 commit E
pick d780c7f commit H
pick fe771ee updated year
pick 64049ed Update README.md

# Rebase 6ed4610..64049ed onto 6ed4610 (9 commands)
```

Above the necessary changes to the commit history can be seen.

1. Delete G
2. Swap E and F
3. Squash B and C
4. Split D to 3 separate commits (see later)

This needs to be saved now.

Squash B and C

After saving a window pops up in which the commit message can be adapted.

```
# This is a combination of 2 commits.
# This is the 1st commit message:

commit B

# This is the commit message #2:

commit C

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
```

```
# This is a combination of 2 commits.
# This is the 1st commit message:

commit B + C

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
```

On the rights side can be see that the commit message is now “commit B + C”. This needs now to be saved. Now the commit D has to be changed in the next step.

```
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git glog
* e33963b (HEAD) commit D
* 77a1e56 commit B + C
* 6ed4610 (origin/topic, topic) commit A
* ffe0970 commit 0
* 65475c2 Initial commit
```

Split commit D

Now the edit of commit D has to be made.
After saving, the HEAD is set to the current commit (commit D) which needs to be edited.

Now, the HEAD can be set 1 node before the current commit in order to revert that all files of D are committed at once.

```
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git reset HEAD^
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git status
interactive rebase in progress; onto 6ed4610
Last commands done (3 commands done):
   squash 56ad339 commit C
   edit f5a0a7f commit D
  (see more in file .git/rebase-merge/done)
Next commands to do (5 remaining commands):
   pick 2ba1b96 commit F
   pick a5730d8 commit E
  (use "git rebase --edit-todo" to view and edit)
You are currently editing a commit while rebasing branch 'master' on '6ed4610'.
  (use "git commit --amend" to amend the current commit)
  (use "git rebase --continue" once you are satisfied with your changes)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .idea/
        FileD1.txt
        FileD2.txt
        FileD3.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Now the files can be added and committed 1 by 1.

```
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git add FileD1.txt
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git commit -m "commit D1"
[detached HEAD 94ab54b] commit D1
 1 file changed, 1 insertion(+)
 create mode 100644 FileD1.txt
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git status
interactive rebase in progress; onto 6ed4610
Last commands done (3 commands done):
   squash 56ad339 commit C
   edit f5a0a7f commit D
  (see more in file .git/rebase-merge/done)
Next commands to do (5 remaining commands):
   pick 2ba1b96 commit F
   pick a5730d8 commit E
  (use "git rebase --edit-todo" to view and edit)
You are currently editing a commit while rebasing branch 'master' on '6ed4610'.
  (use "git commit --amend" to amend the current commit)
  (use "git rebase --continue" once you are satisfied with your changes)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .idea/
        FileD2.txt
        FileD3.txt

nothing added to commit but untracked files present (use "git add" to track)
```

The same goes for D2 and D3

```
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git add FileD2.txt
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git commit -m "commit D2"
[detached HEAD ef12ce3] commit D2
 1 file changed, 1 insertion(+)
 create mode 100644 FileD2.txt
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git add FileD3.txt
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git commit -m "commit D3"
[detached HEAD 1a55daf] commit D3
 1 file changed, 1 insertion(+)
 create mode 100644 FileD3.txt
```

Finishing the rebase

Now the rebase needs to be continued and it can be finished successfully:

```
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git rebase --continue
Successfully rebased and updated refs/heads/master.
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$ git glog
* 869cd33 (HEAD -> master) Update README.md
* 71b21da updated year
* f90f57b commit H
* f0b7c04 commit E
* a614904 commit F
* 1a55daf commit D3
* ef12ce3 commit D2
* 94ab54b commit D1
* 11dfa22 commit B + C
* 6ed4610 (origin/topic, topic) commit A
penz@pop-os:~/Workspace/Private/cicd-exercises/cd2020-ex01$
```

Part 4 - GIT ReReRe

ReReRe stands for **re**use **re**corded **re**solution.
The intention is to solve already happened conflicts (like merge conflicts) automatically. It can be enabled by running the following config setting:

```
$git config --global rerere.enabled true
```

If there is a long lived branch one can occasionally merge, resolve the conflicts and then back out of the merge. How to solve this particular conflict is now remembered. When the long-lived branch is then finally merged with the destination branch, those conflicts can the be solved automatically.