GIT CONCEPTS - Sean Corzo 8/14/2023

GIT COMMIT

A commit is a lightweight pointer to a lightweight delta snapshot of changes since the last commit.

New commits are applied to whichever branch you have checked out.

Git commit creates a new commit pointer.

GIT CHECKOUT

Checking out a branch makes it active.

HEAD

HEAD points to whatever you have checked out.

Usually that is a branch, so HEAD -> branch -> c1

It is possible to check out a specific commit rather than a branch

In that case, the relationship is: HEAD -> c1 # this is called a **"detached head"**

GIT BRANCH

Branches are just pointers to specific commits.

Creating a new branch attaches the pointer to the commit in the active branch.

Once you create a branch, you need to check it out in order to make it active.

GIT MERGE

Used to merge branches.

Creates a new commit that merges the two branches.

Example:

```
    # Check out main branch
    git checkout main

    # merge bugFix into main
    Git merge bugFix
```

RESET

git reset reverses changes by moving a branch reference backwards in time to an older commit. In this sense you can think of it as "rewriting history;" git reset will move a branch backwards as if the commit had never been made in the first place.

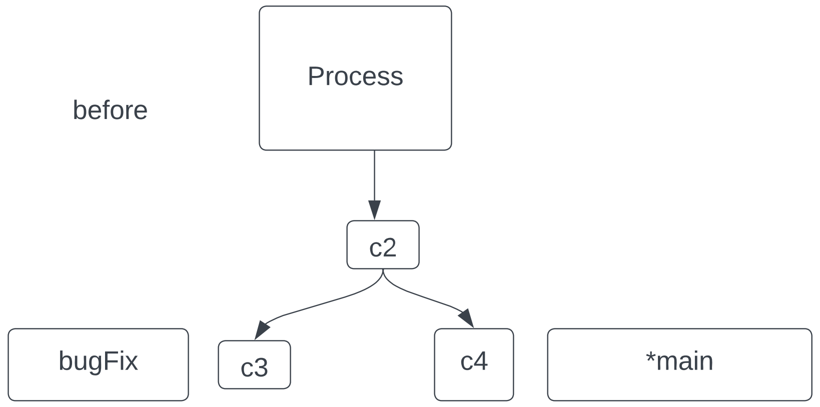
REVERT

While resetting works great for local branches on your own machine, its method of "rewriting history" doesn't work for remote branches that others are using.

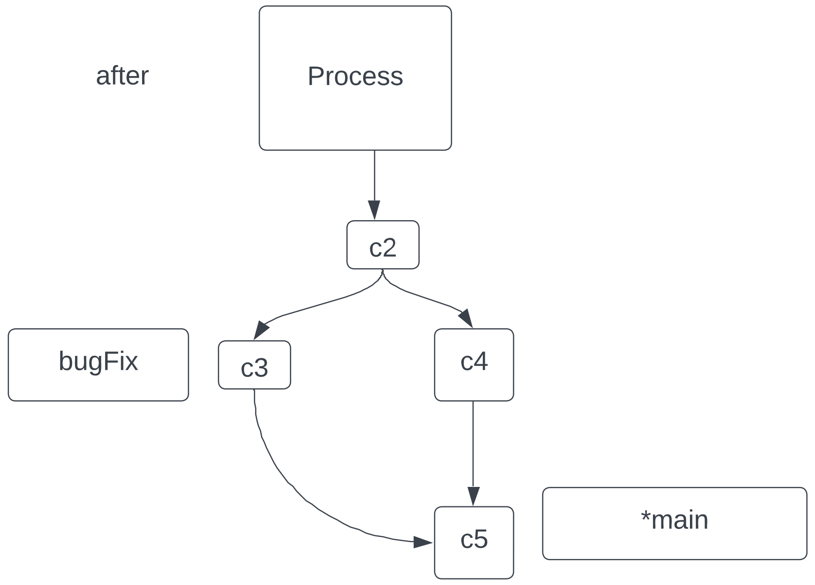
In order to reverse changes and share those reversed changes with others, we need to use git revert.

Merge Example

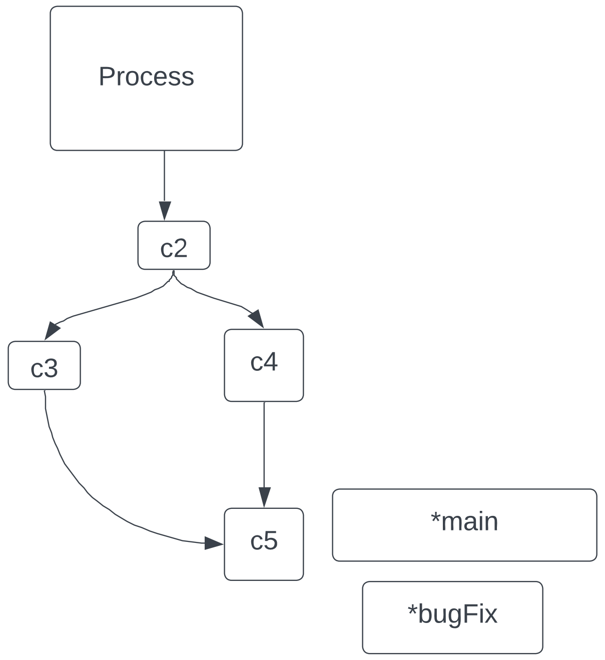
before



after



finally, merge bugFix into
main with:
git checkout bugFix
git merge main
# note that no new
commit is created


Process
c2
c3
c4
c5
*main
*bugFix