MAKE
SCHOOL

WHAT IS GIT?

a distributed version control system

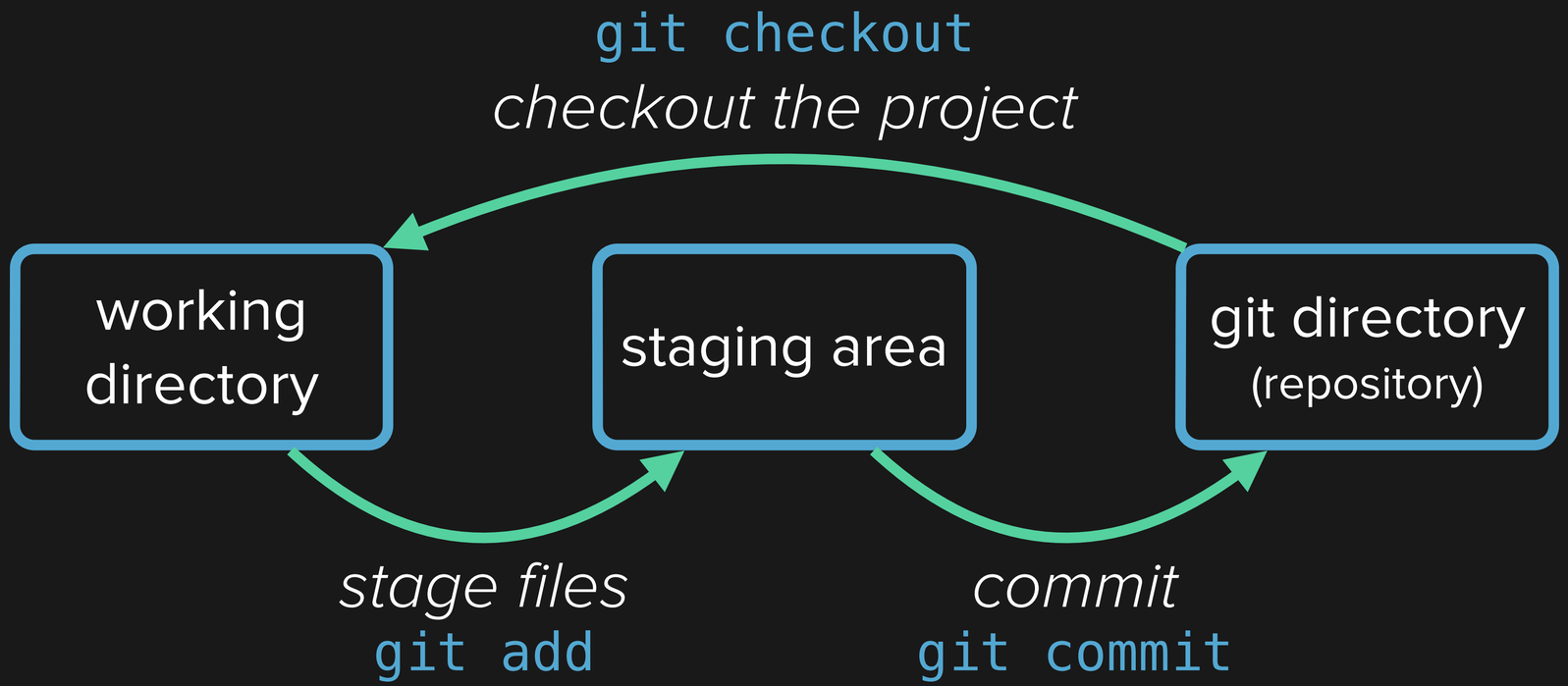
DISTRIBUTED VERSION CONTROL SYSTEM

Operations run locally

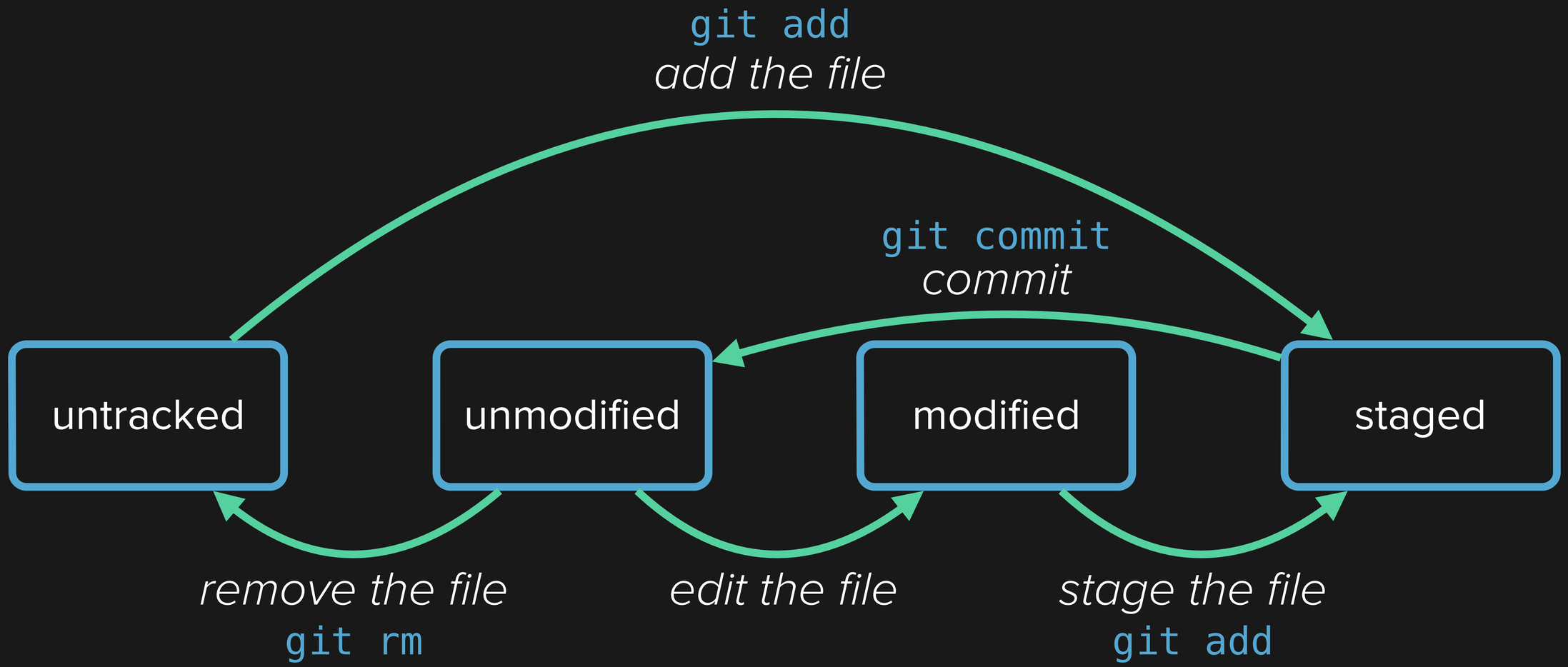
Can be *pushed* to a remote location

Much faster than centralized VCS

GIT SECTIONS



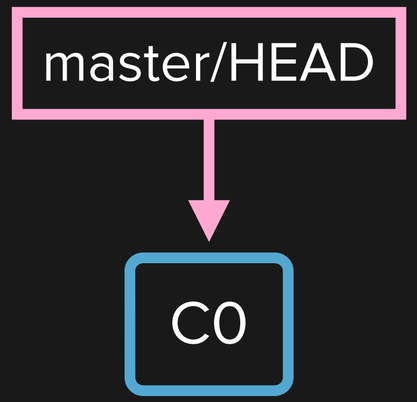
GIT FILE STATUS



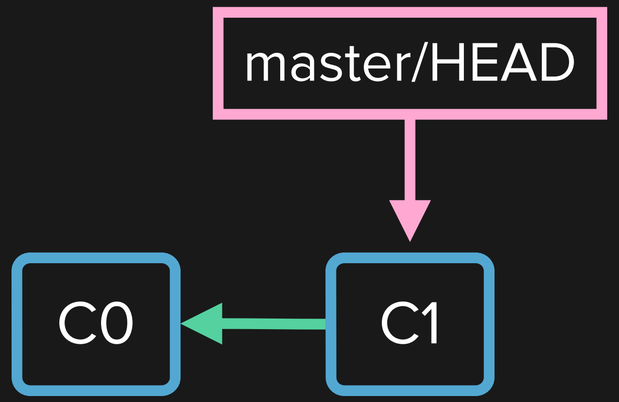
GIT COMMIT

adding a snapshot

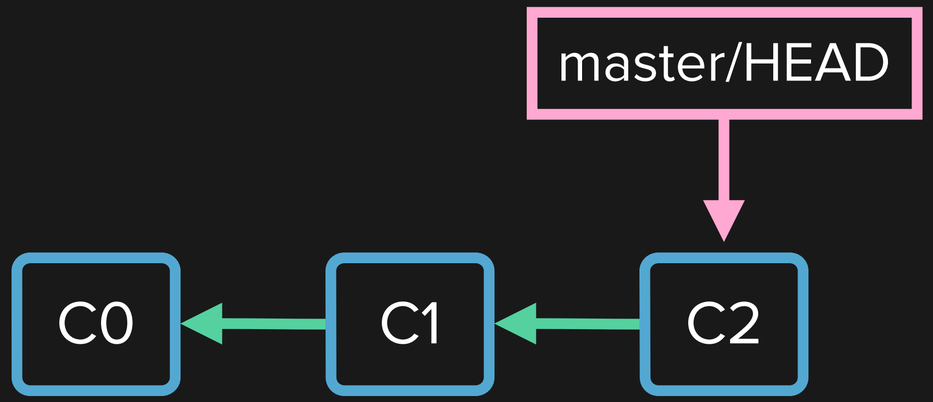
```
git init
```




```
git commit -m "<summary of changes>"
```




```
git commit -m "<summary of changes>"
```

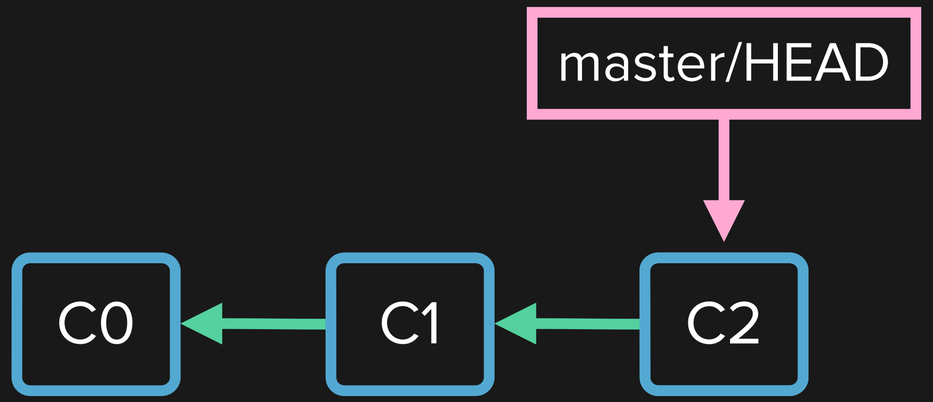


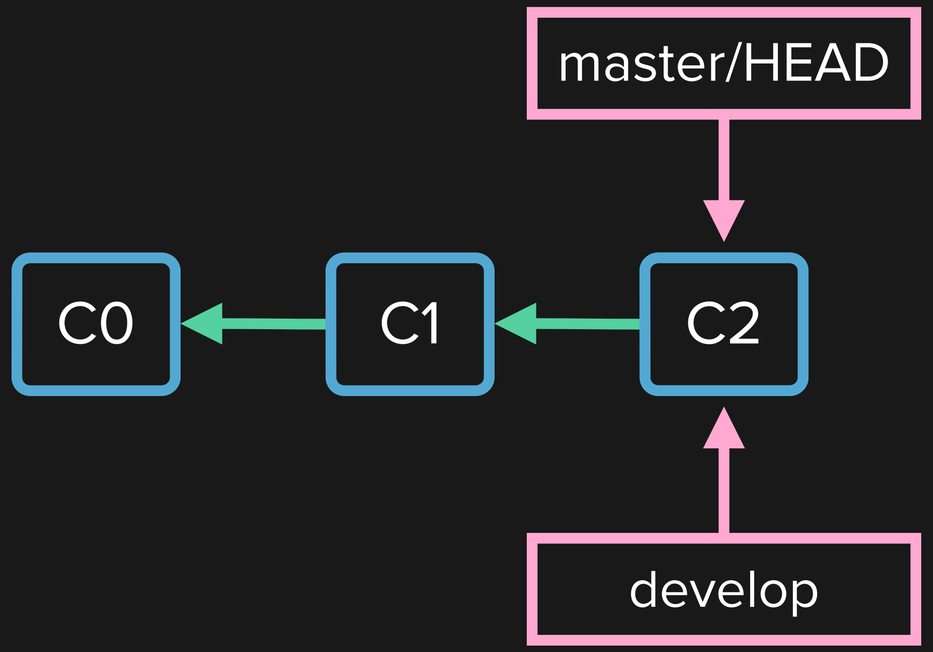

```
git commit -m "<summary of changes>"
```

GIT BRANCH & CHECKOUT

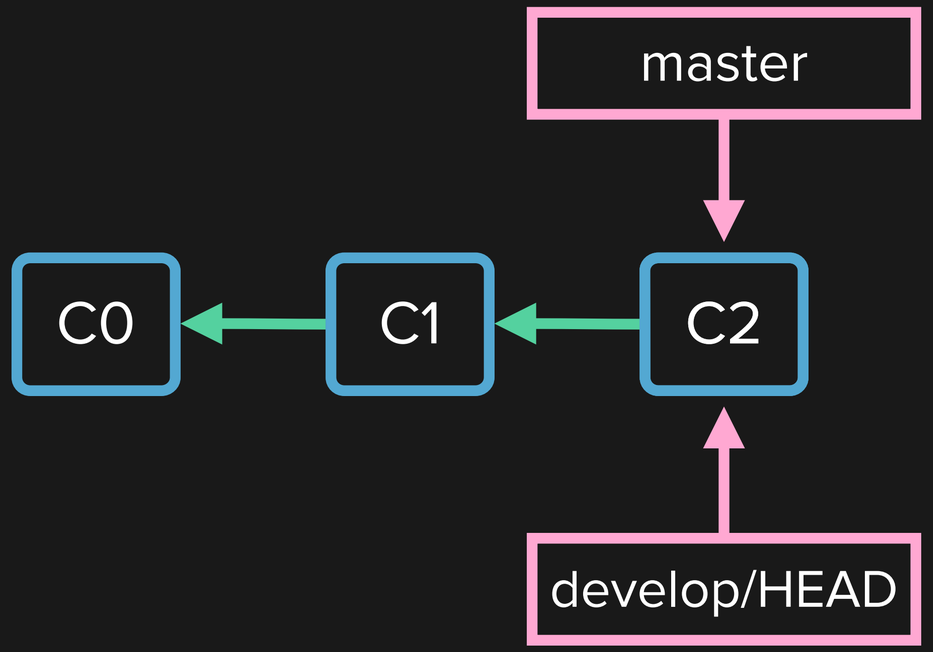
creating parallel timelines

MAKE SCHOOL

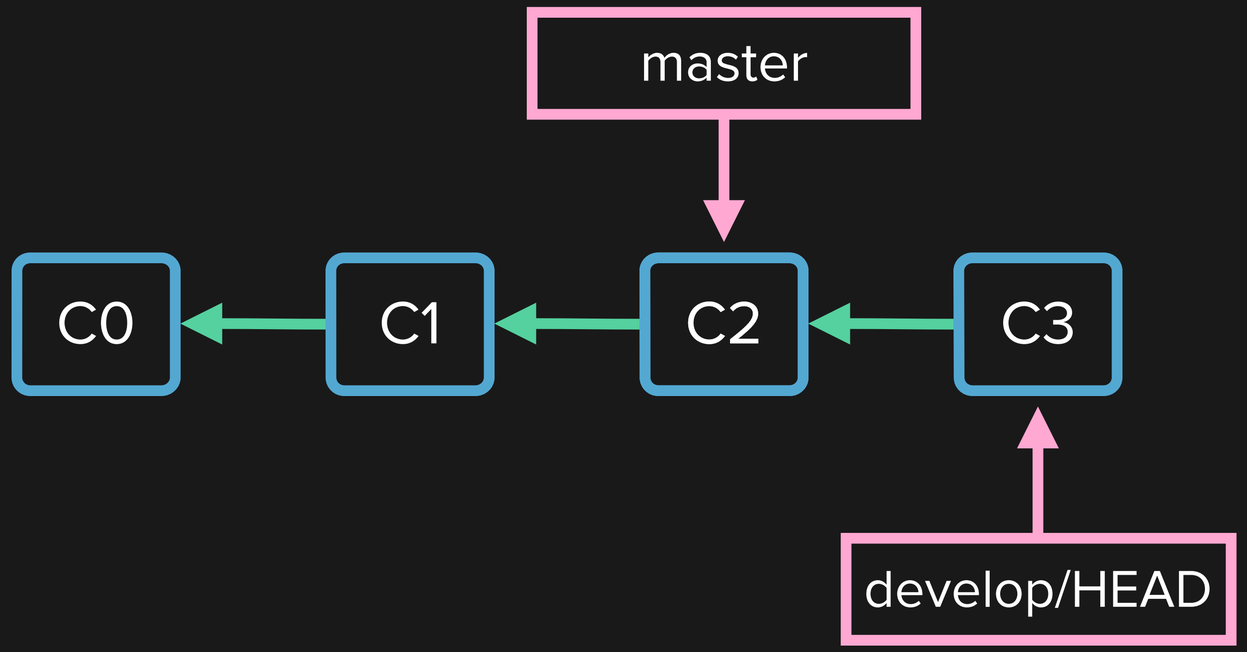




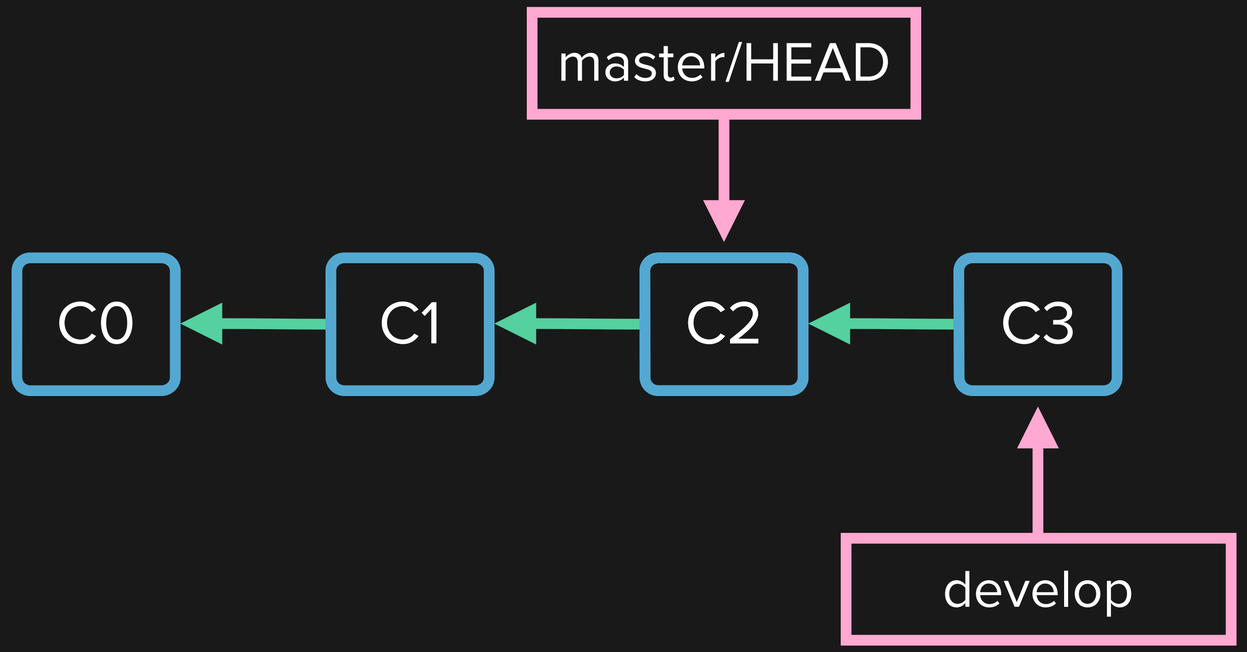

```
git branch develop
```



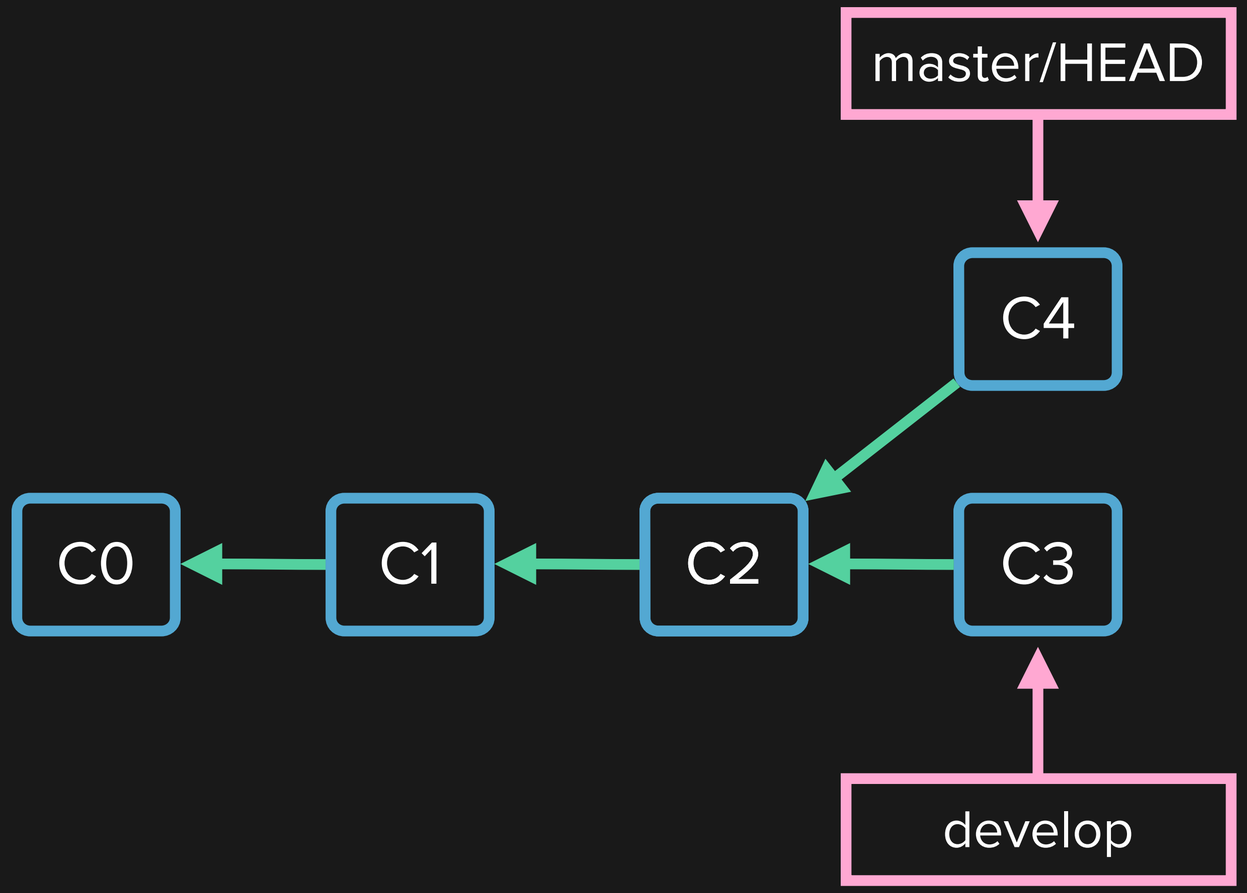

```
git checkout develop
```



`git commit`




```
git checkout master
```



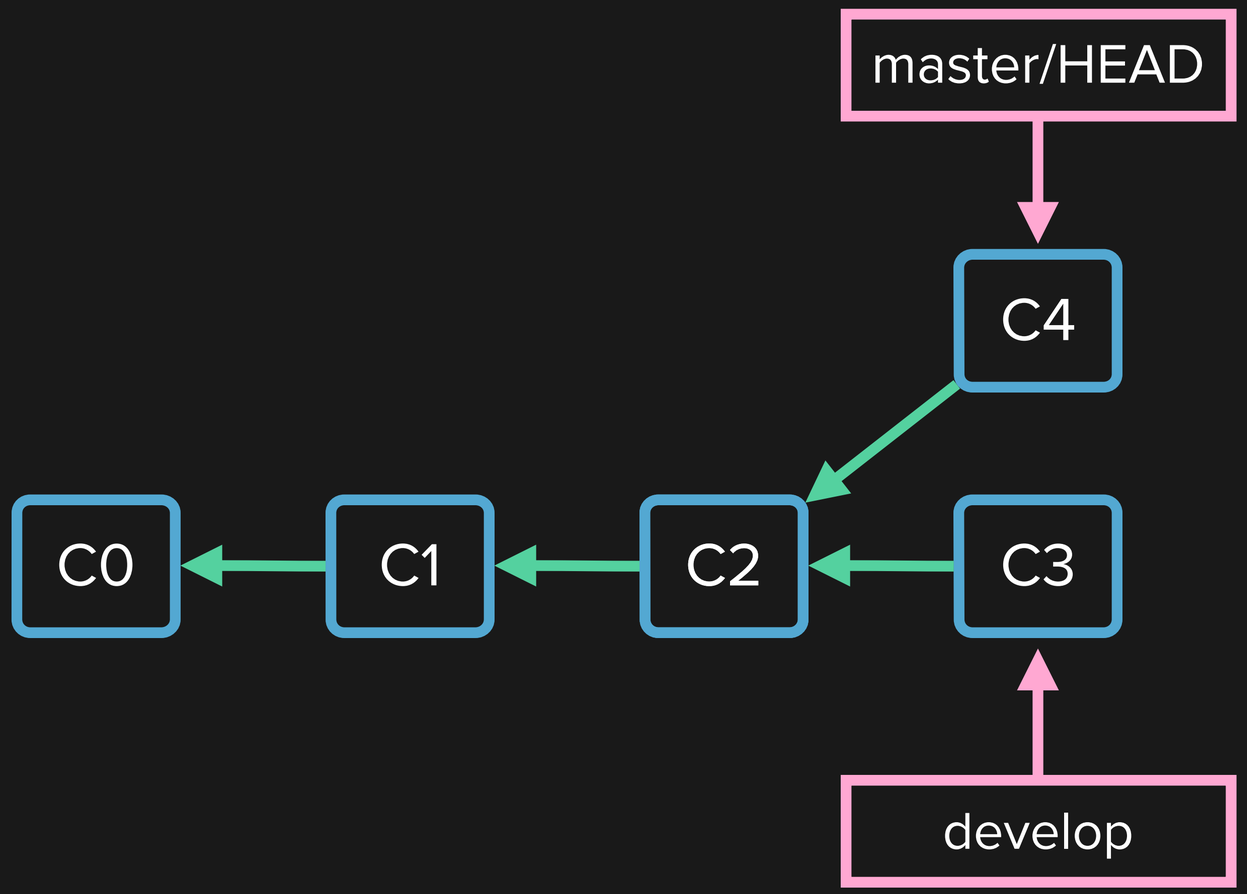
`git commit`

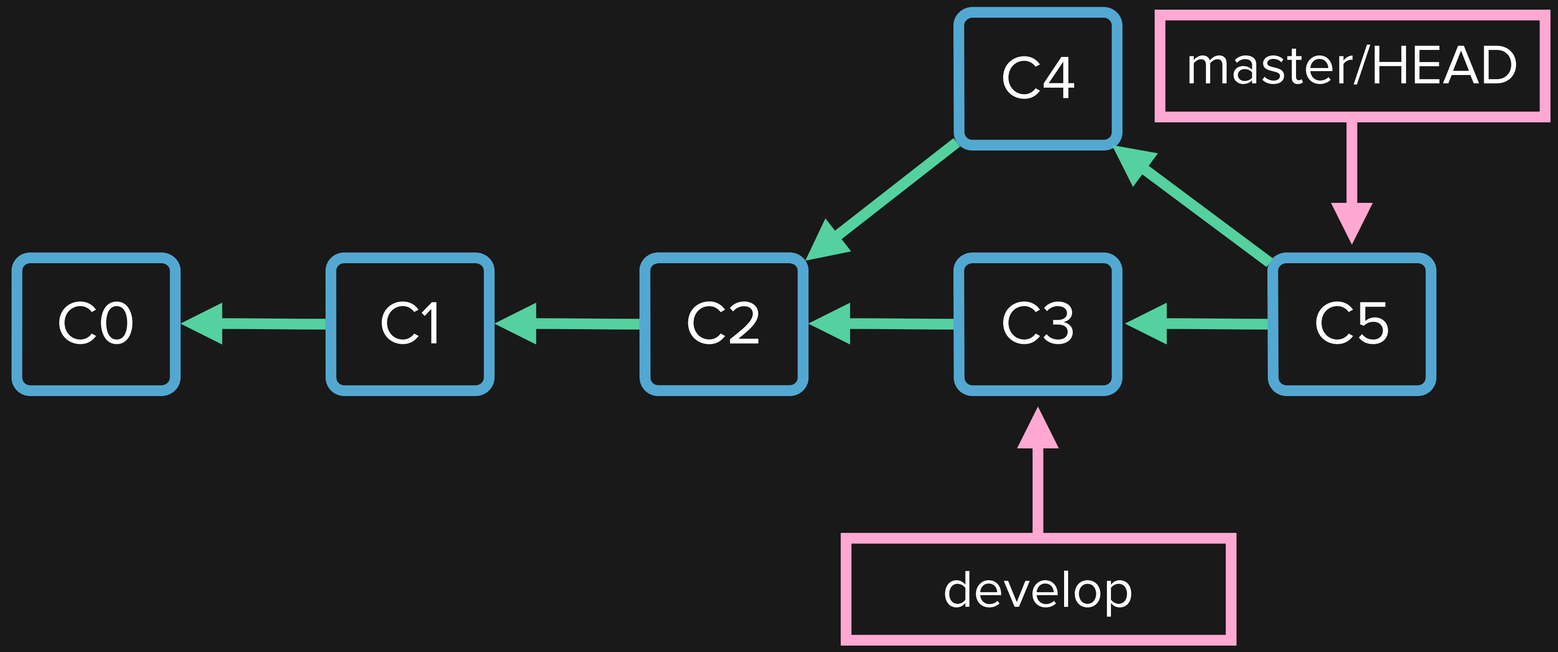
NOW WE ARE IN TROUBLE...

git merge to the rescue!

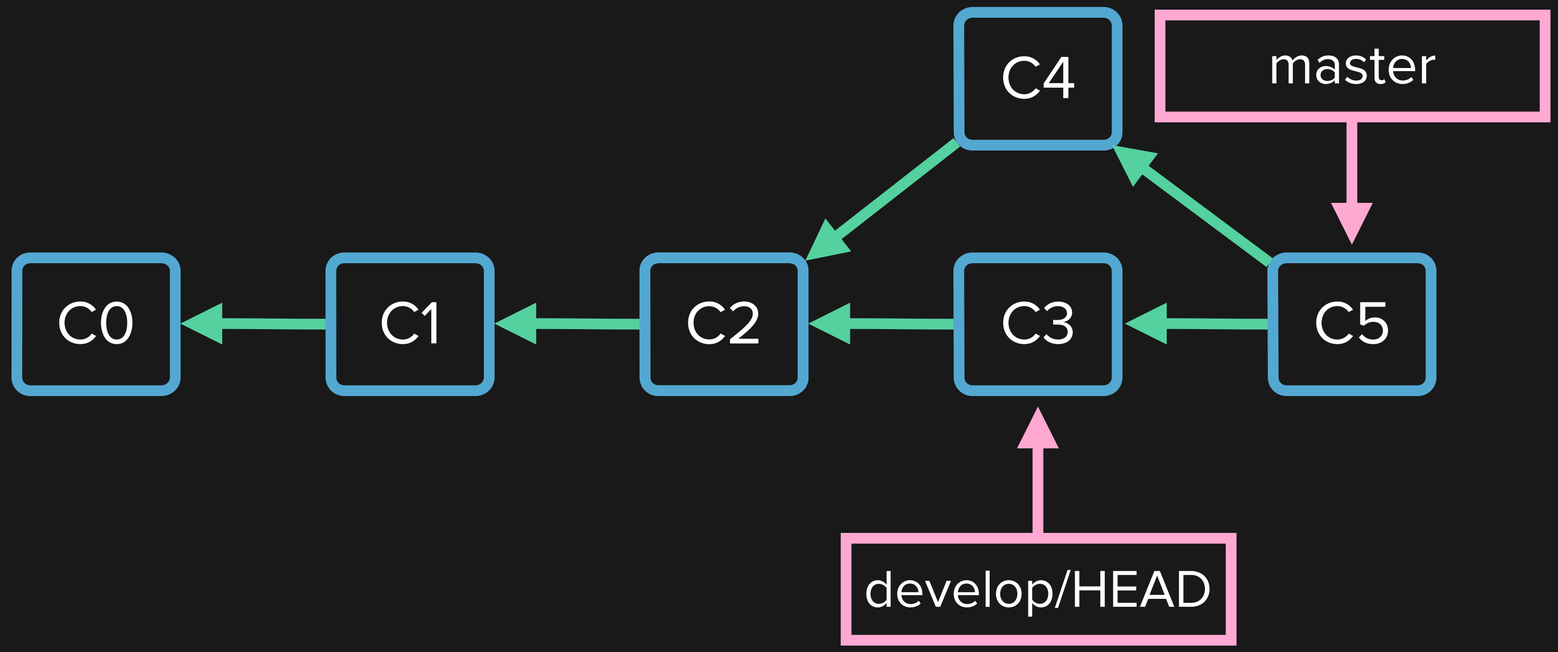
GIT MERGE

join our branches back together

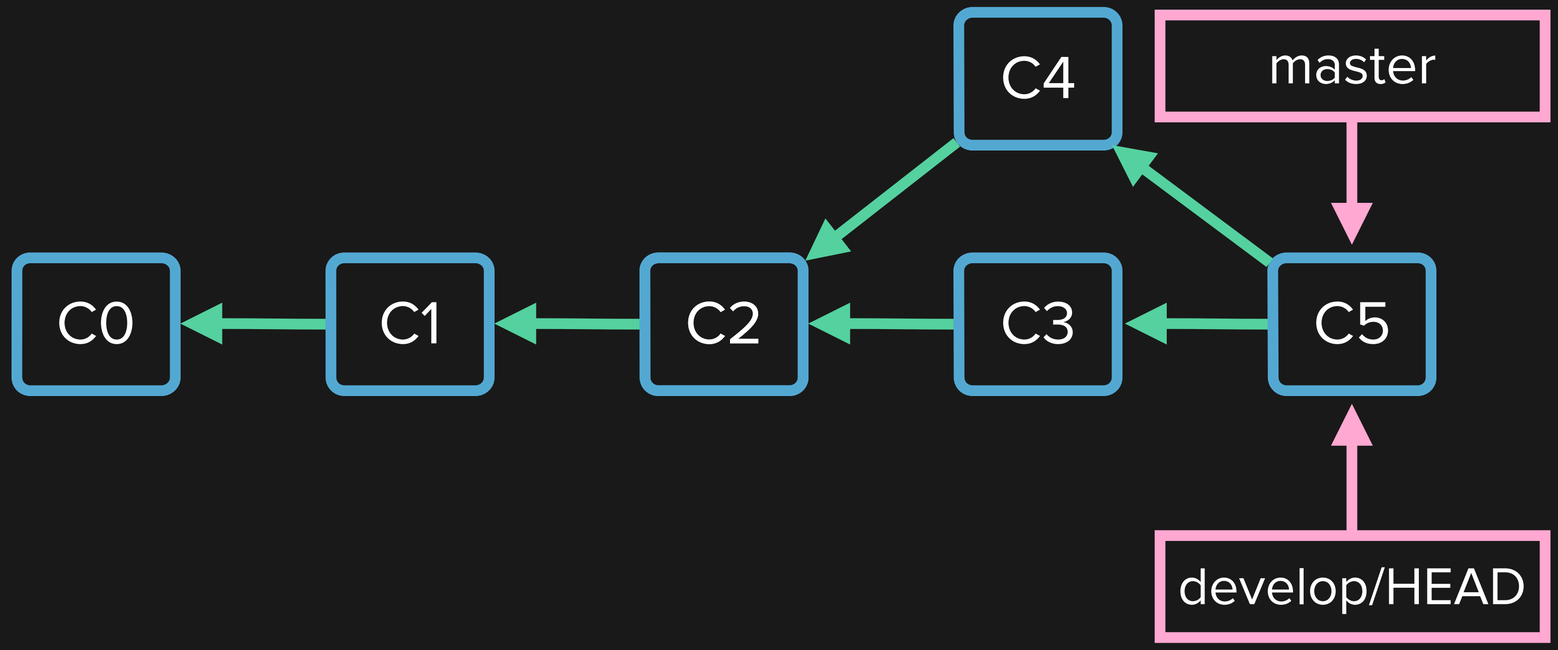





```
git merge develop
```




```
git checkout develop
```



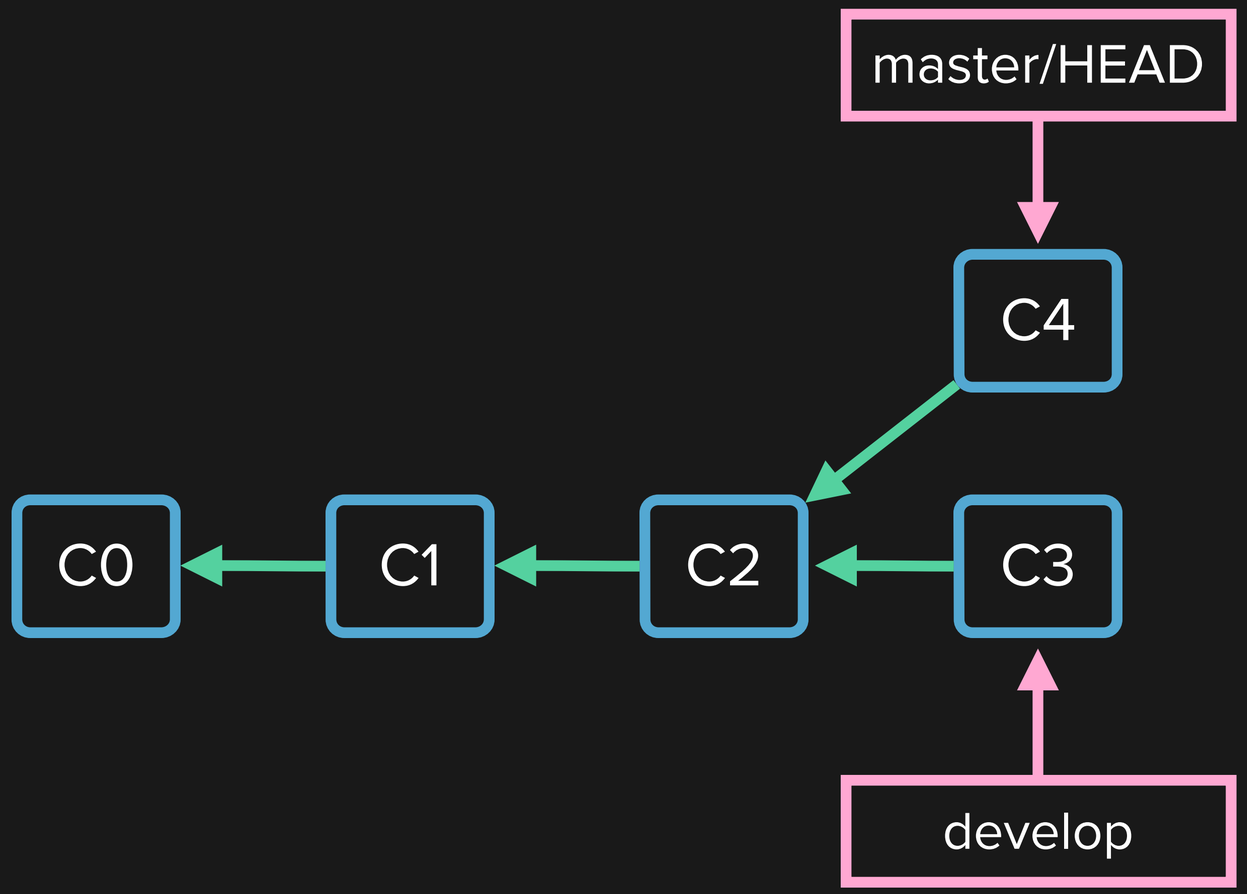

```
git merge master
```

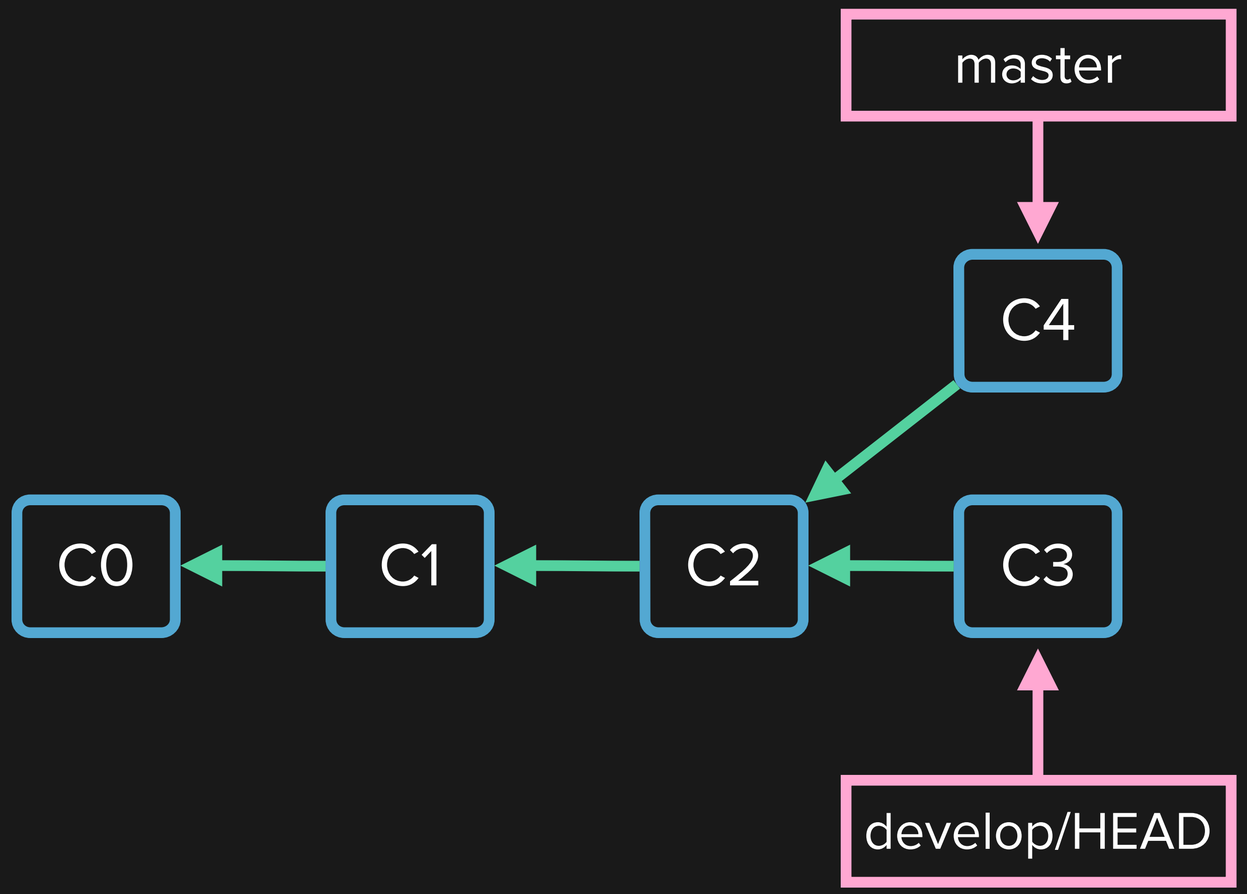
THAT'S PRETTY MESSY...

git rebase can be cleaner!

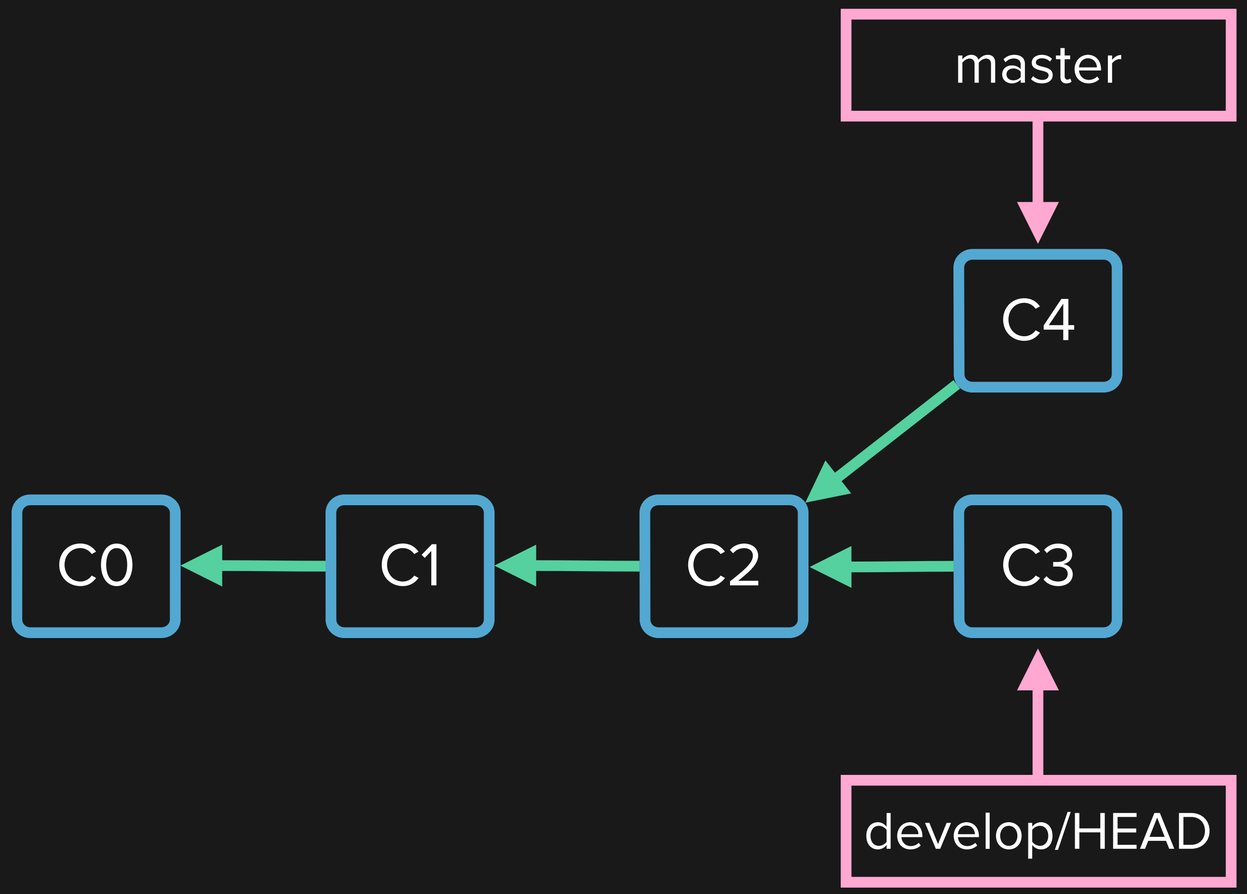
GIT REBASE

replay commits in order

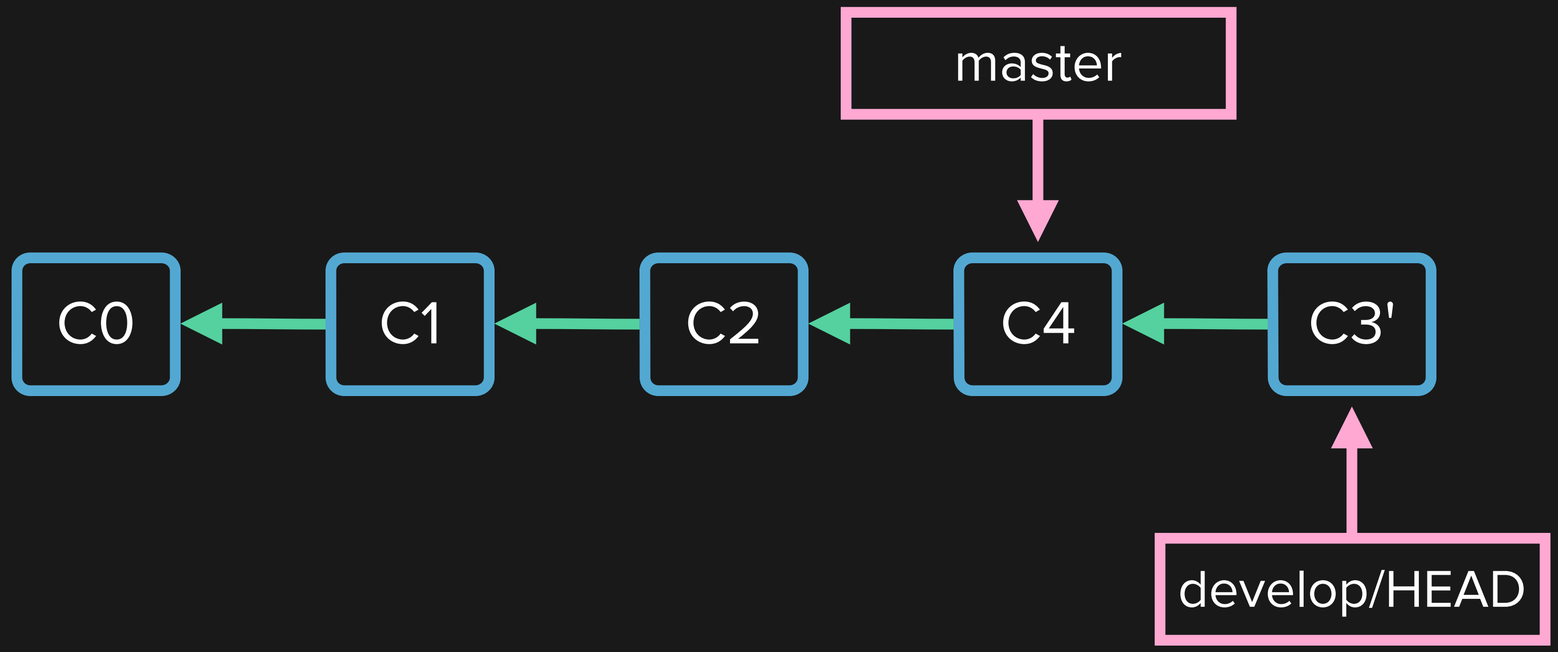


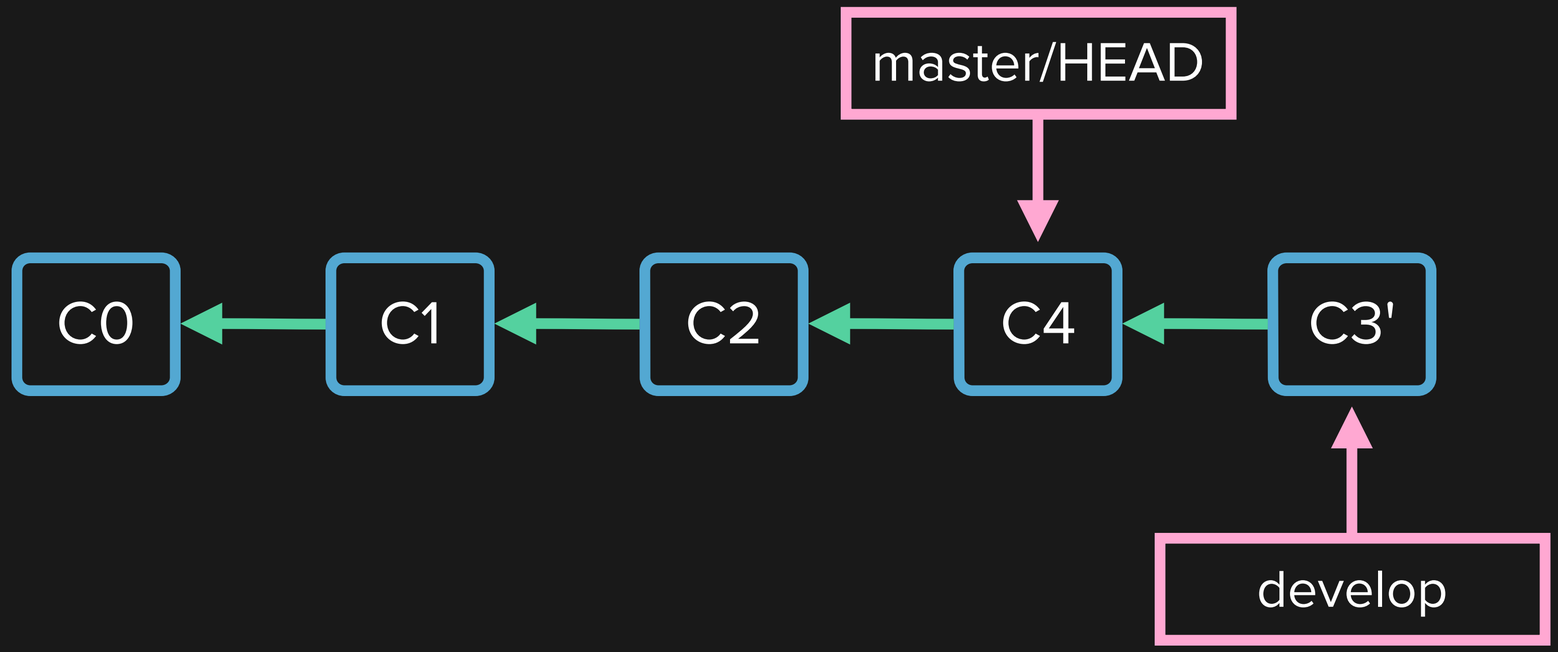



```
git checkout develop
```

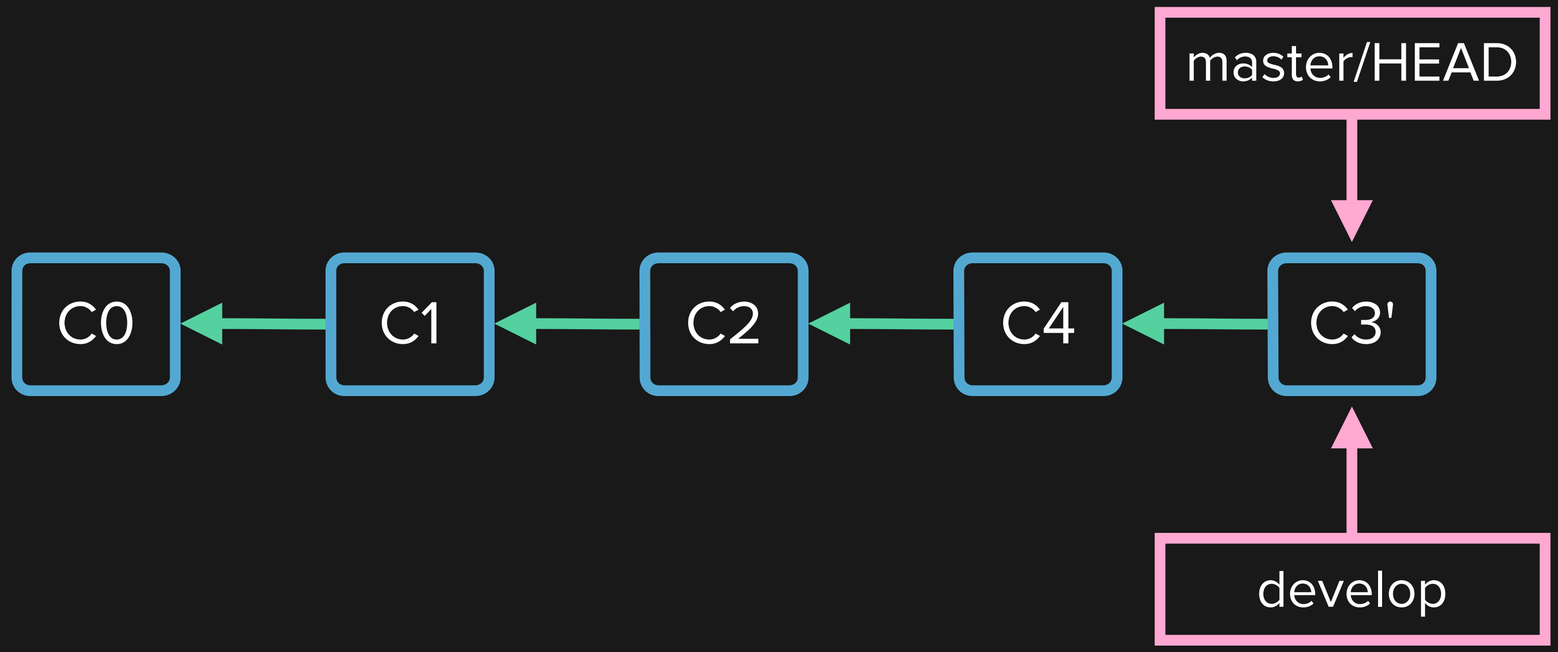



```
git rebase master
```






```
git checkout master
```




```
git merge develop
```

NEVER EVER REBASE A PUBLIC BRANCH

you'll anger the programming gods...

NEVER EVER REBASE A PUBLIC BRANCH

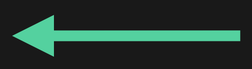
seriously, don't do it

DEALING WITH CONFLICTS

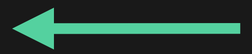
sometimes two commits change the same file

```
$ git merge master
Auto-merging file1
CONFLICT (content): Merge conflict in file1
Automatic merge failed; fix conflicts and then commit the result.
```

```
$ cat file1
<<<<<<< HEAD
This was added in C2
=======
This was added in C1
>>>>>>> master
```



in current branch



in master branch

RESOLVING MERGE CONFLICTS

Use editor of choice to resolve

`git add` files once manual merge finished

`git commit` or `git rebase --continue`

SHARING AND BACKING UP

using remote repositories -- putting the distributed in DVCS

MAKE SCHOOL

GIT REMOTES

Non-local copy of your repository

Sync changes with *push* and *pull*

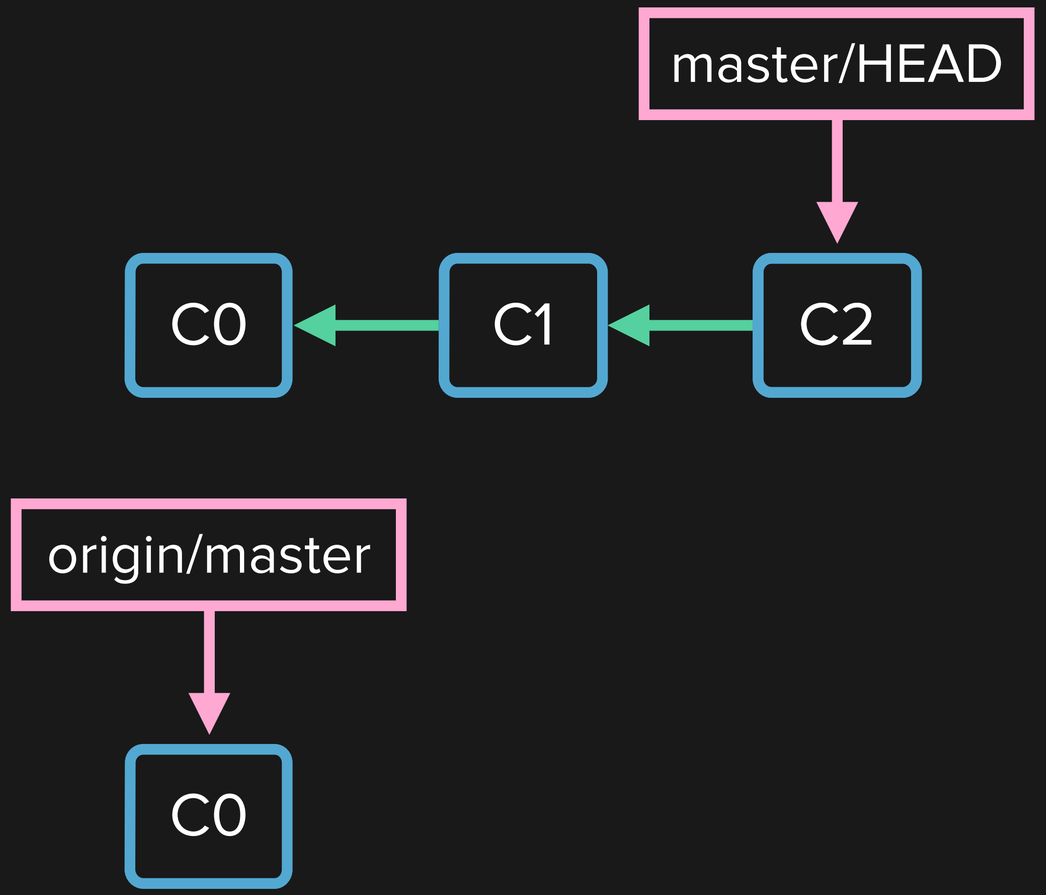
Allows for collaboration and backup

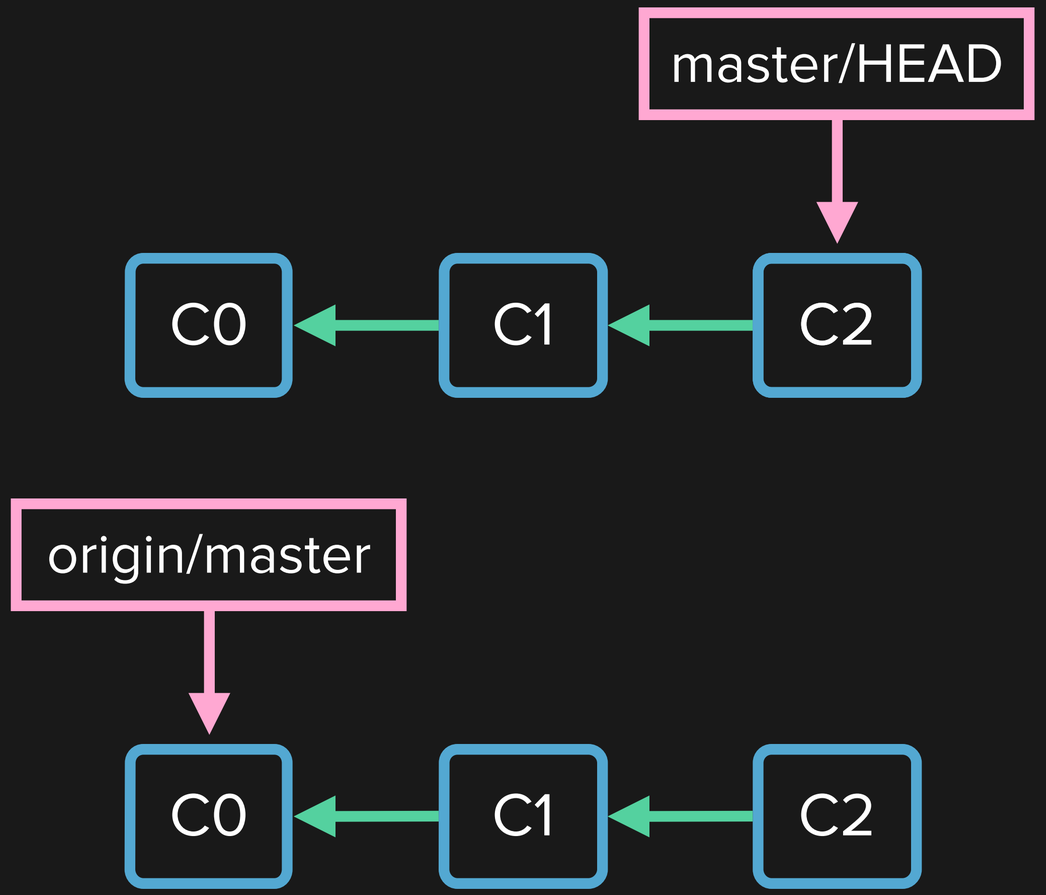
GitHub and Bitbucket host git remotes for free

GIT PUSH

Moves commits from a current local branch to a remote branch

```
git push <remote> <branch>
```






```
git push origin master
```

GIT PULL

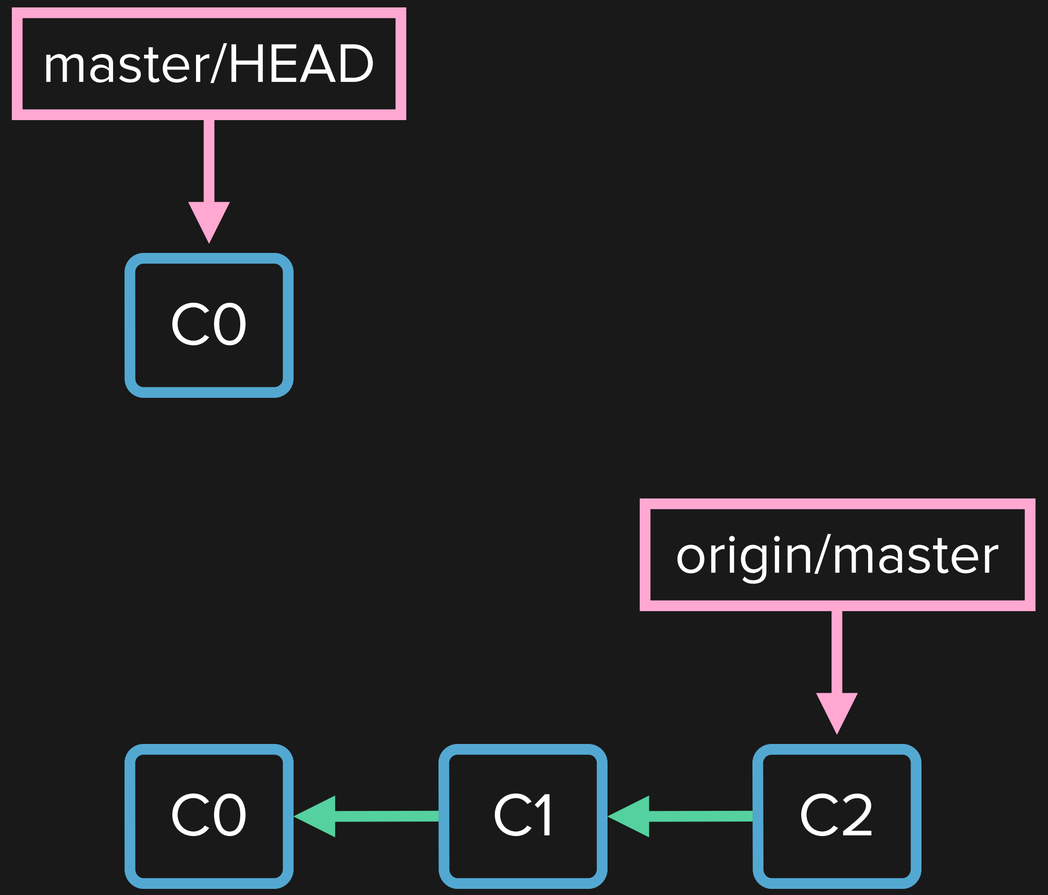
Moves commits from a remote branch to a local branch

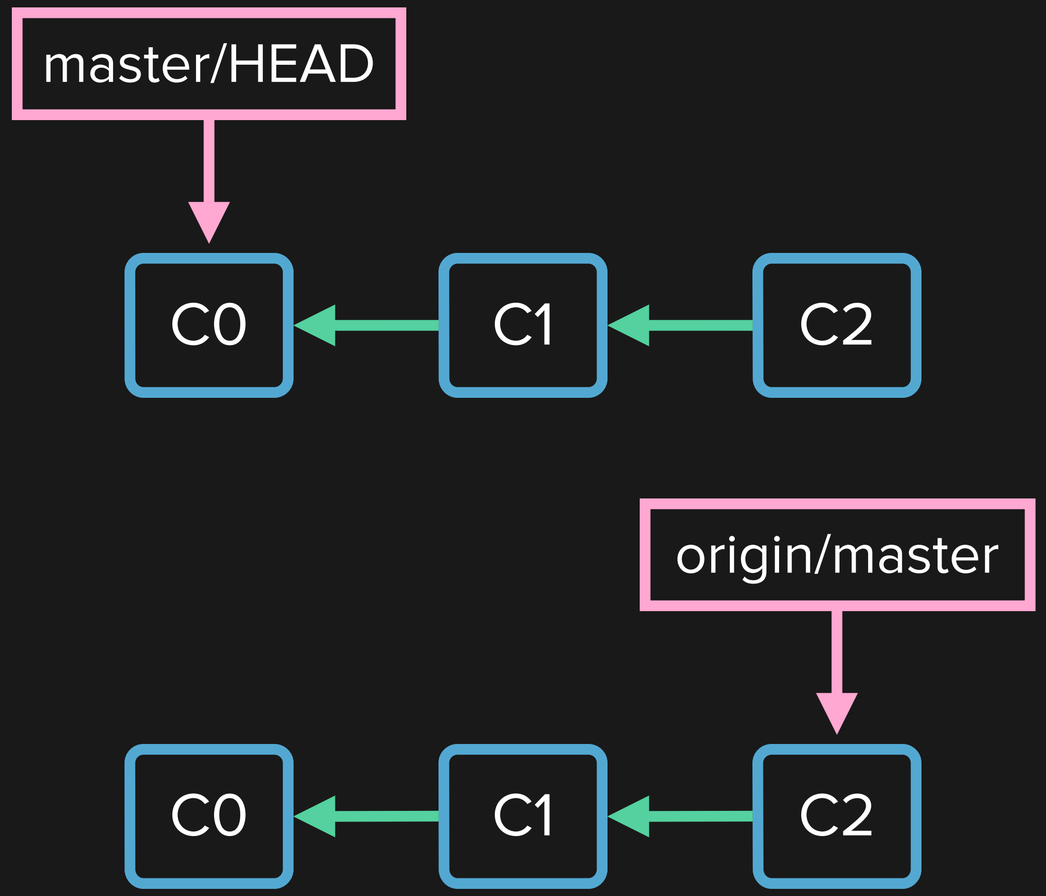
`git fetch` followed by `git merge`

`git pull --rebase` to rebase instead of merge

```
git pull <remote> <branch>
```

MAKE SCHOOL

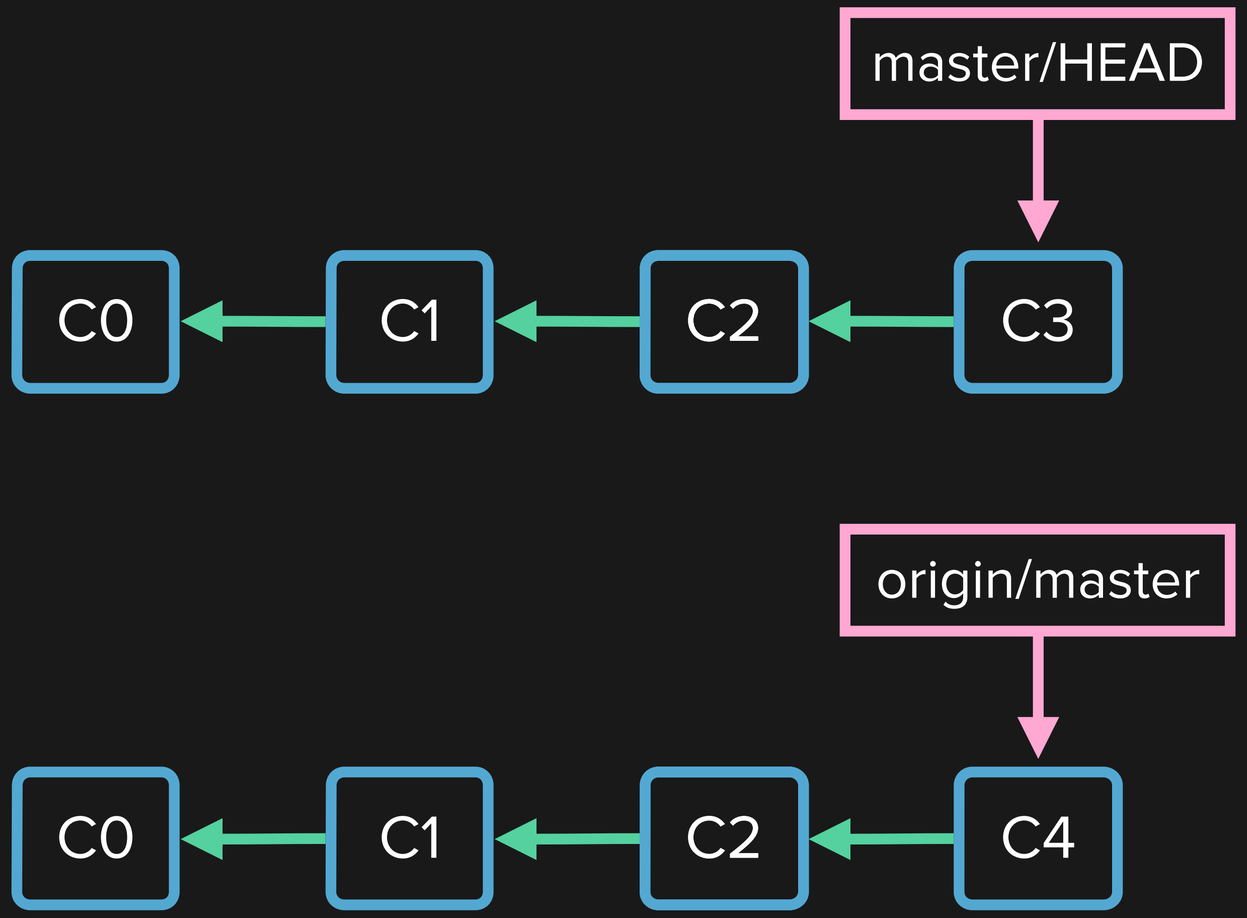


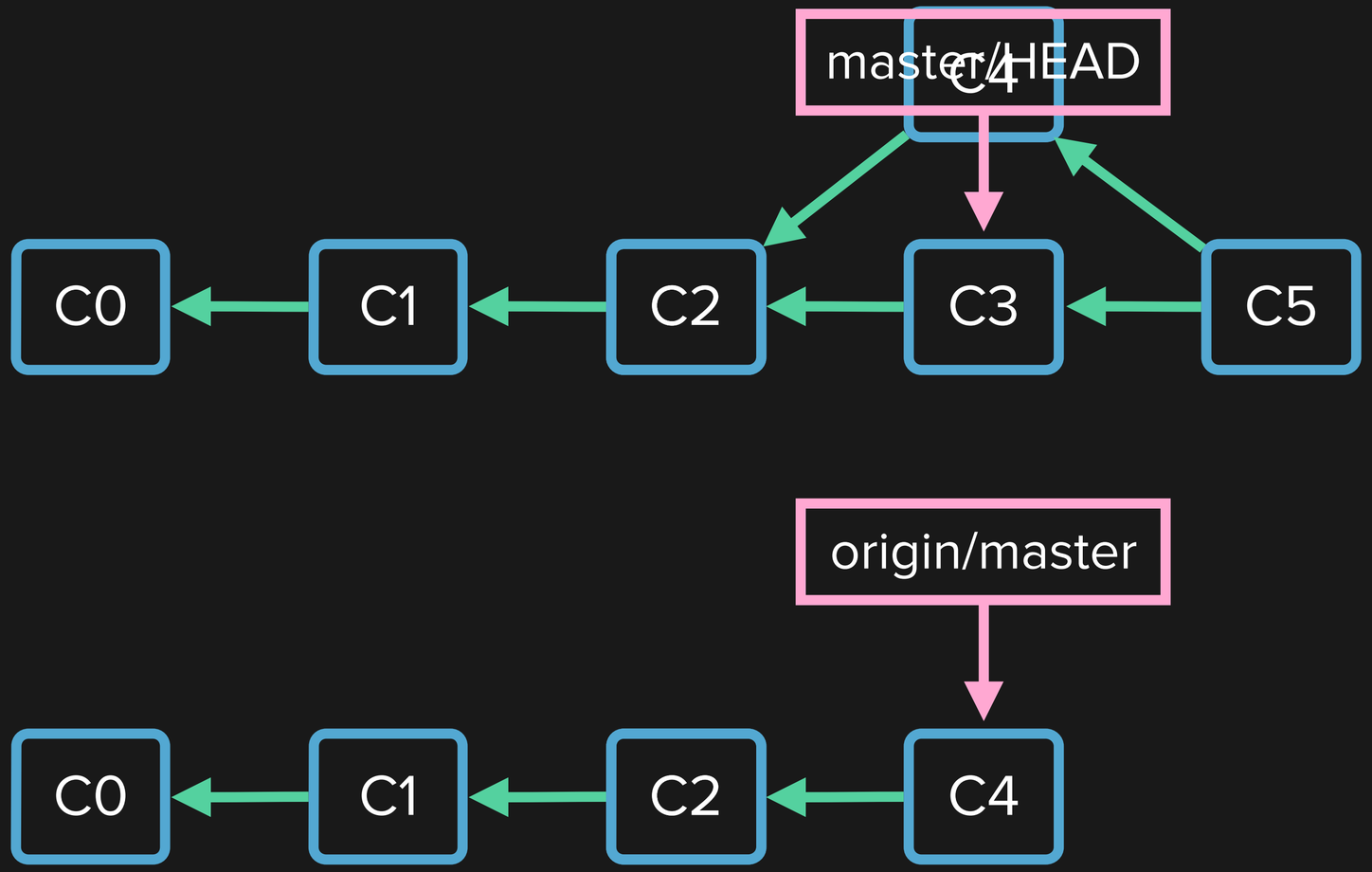



```
git pull origin master
```

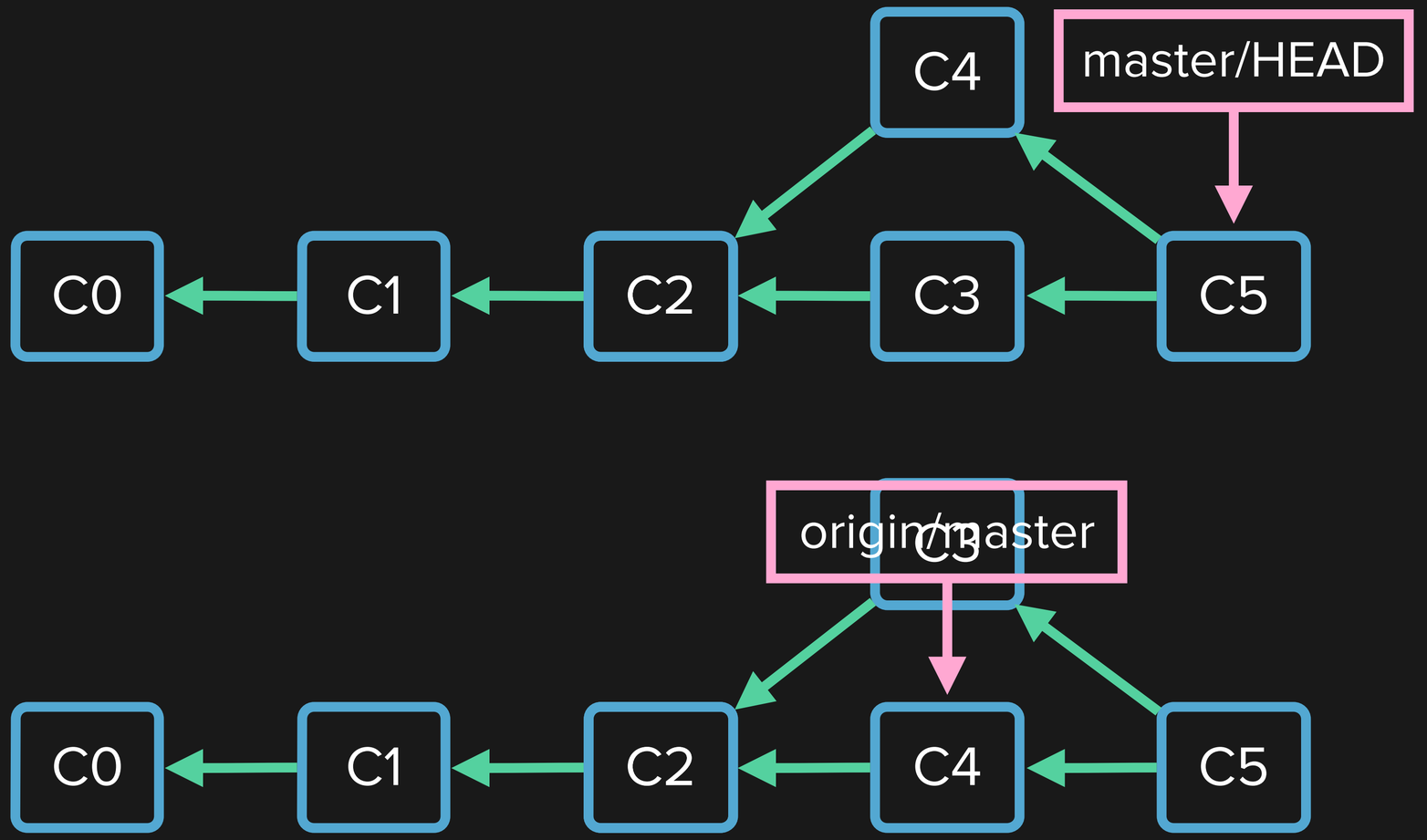
CHANGES ON REMOTE

git pull



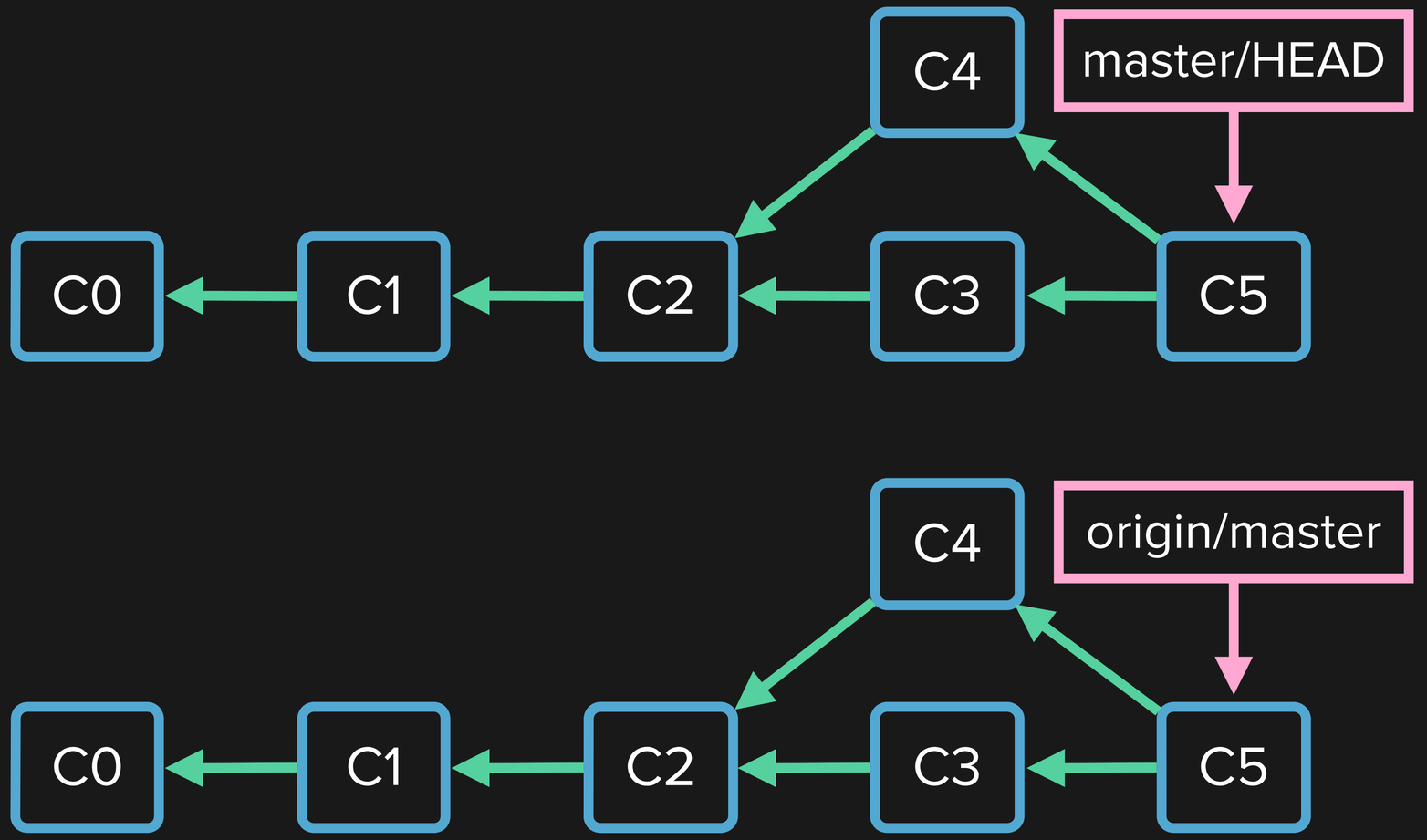



```
git pull origin master
```



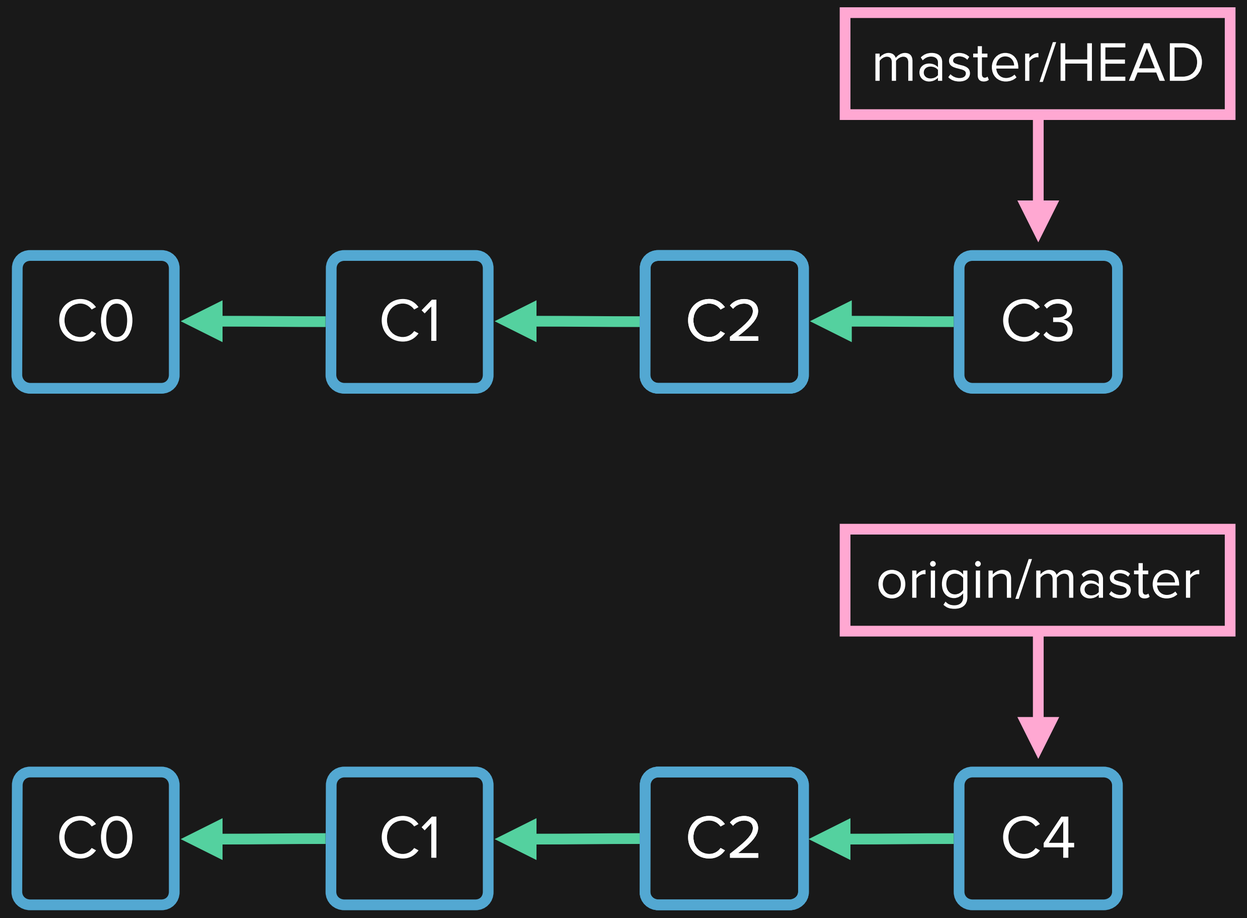

```
git push origin master
```

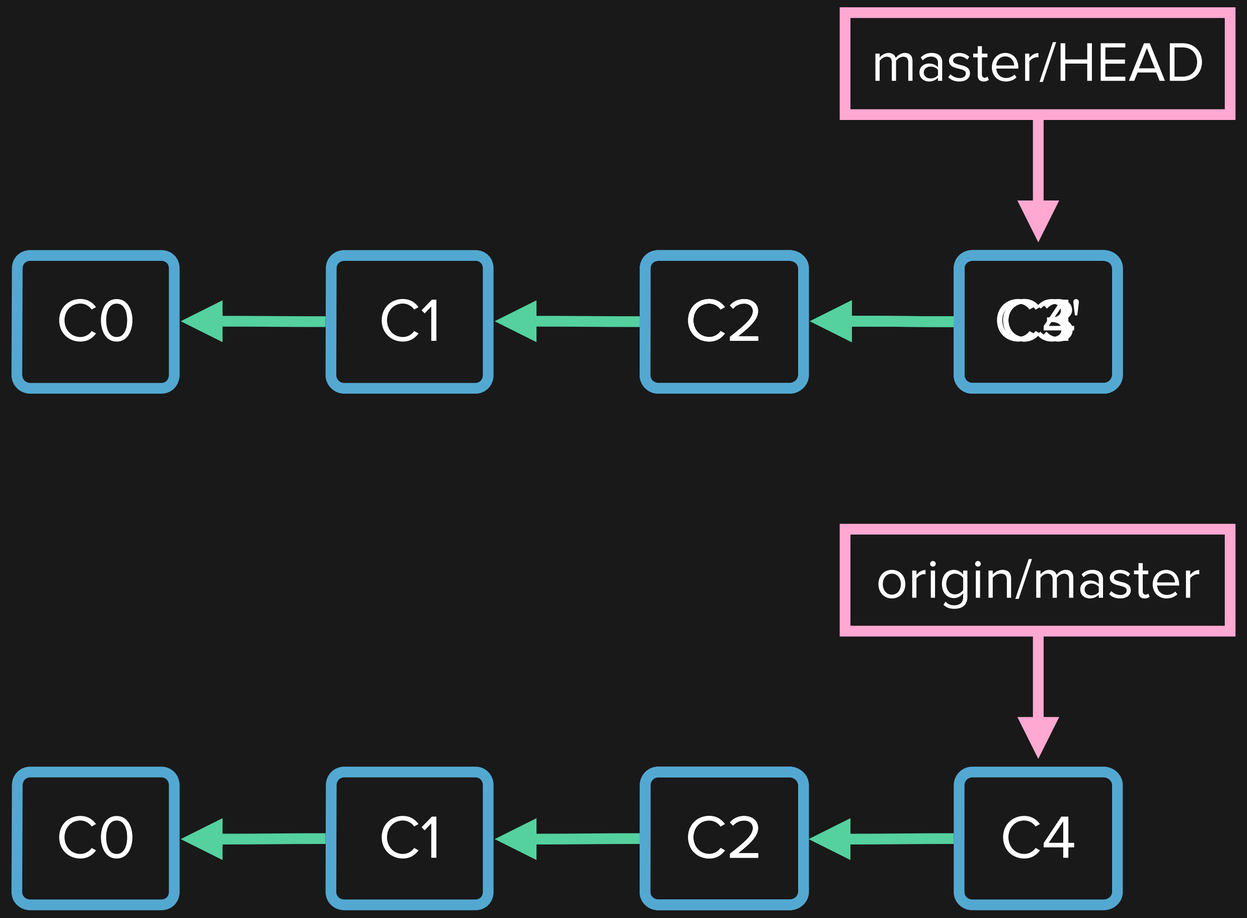
MAKE SCHOOL



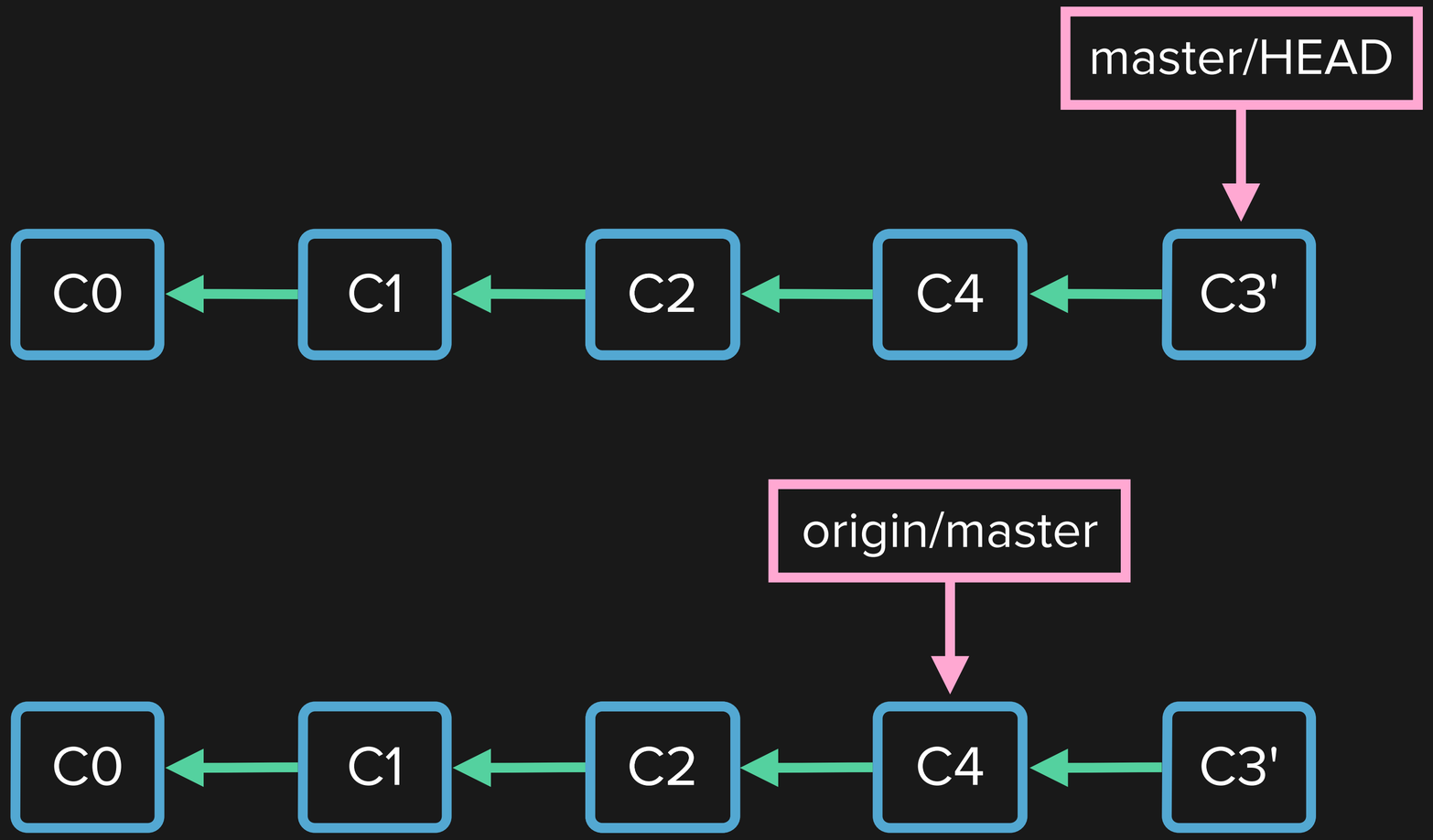
CHANGES ON REMOTE

git pull --rebase






```
git pull --rebase origin master
```




```
git push origin master
```

USEFUL COMMANDS

GIT INIT

Start a new, empty repository

Run from based directory of project

```
git init
```

GIT STATUS

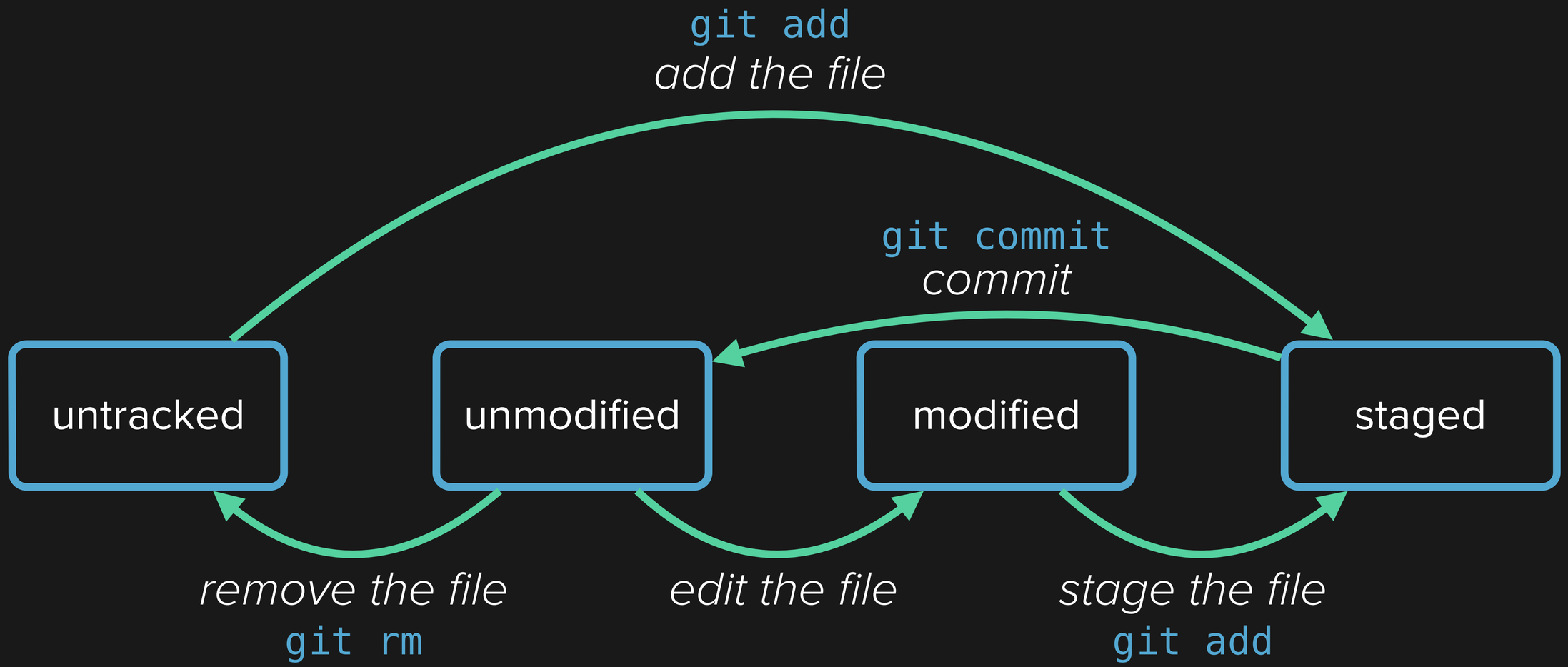
Show current file status of repository

Useful for figuring out what will be committed

```
git status
```

MAKE SCHOOL

GIT FILE STATUS



GIT DIFF

View file diffs between states

Many different options

```
git diff --staged
git diff <commit1> <commit2>
git diff <branch1> <branch2>
```

GIT LOG

Shows all commits on current branch

```
git log
```

Use to find commit hash

If log is long, press 'q' to return to Terminal

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

MAKE SCHOOL

GIT ADD

Move tracked file to staging

Move untracked file to staging and track

```
git add --all
git add <file>
```

```
git add <directory>
git add *.txt
```

MAKE SCHOOL

GIT RM

Stop tracking a file

Works the same as `git add`

```
git rm <file>
git rm <directory>
git rm *.txt
```

MAKE SCHOOL

GIT COMMIT

Move staged files into a commit

Adds commit to current branch

Keep commits to 50 characters or less

Use imperative mood (commanding tone)

http://chris.beams.io/posts/git-commit/

```
git commit -m "<summary of changes>"
```

MAKE SCHOOL

GIT BRANCH

List current branches

```
git branch
```

Create new branches or timelines

```
git branch <branch_name>
```

Does not switch to branch, use `checkout`

GIT CHECKOUT

Moves between branches

```
git checkout <branch>
```

Can be used to grab file from previous commit

```
git checkout <commit> <file>
```

Shortcut to create and checkout branch

```
git checkout -b <branch>
```

Check out an old commit -- always reattach HEAD!

```
git checkout <commit>
```

DETACHED HEAD

Your HEAD is not on a branch -- any commits made will be lost!

Happens when you checkout previous commit

Create new branch and checkout to reattach

GIT RESET

Undo previous commit and restage changes

```
git reset --soft HEAD^
```

Undo previous commit and destroy changes

```
git reset --hard HEAD^
```

GIT CLONE

Create local copy of remote repository

Use SSH clone

```
git clone <SSH_address>
```

GIT REMOTE

Add a new remote

```
git remote add <name> <address>
```

List remotes

```
git remote -v
```

Show remote details (including branches)

```
git remote show <remote>
```

Remove a remote

```
git remote rm <name>
```

GIT PUSH

Push from current local branch to remote

```
git push <remote> <branch>
```

Delete remote branch

```
git push <remote> :<branch>
```

Set push default

```
git push -u <remote> <local_branch>
```

GIT PULL

Pull code from remote

```
git pull <remote> <branch>
```

GIT COMMIT WORKFLOW

Make changes to files

Move changed files to staging

```
git add --all
```

Commit staged files to local git repository

```
git commit -m "<summary of changes>"
```

Push changes to remote git repository

```
git push origin master
```

CONFIGURATION

```
git config --global user.name "Your Name"
git config --global user.email your.name@students.makeschool.com
git config --global color.ui true
git config --global core.editor nano
git config --global alias.lol "log --graph --decorate --oneline"
git config --global alias.lola "log --graph --decorate --oneline --all"
```

COMMANDS NOT COVERED

`git stash` *stash changes without committing*

`git tag` *"tag" commits for releases*

`git reflog` *view ALL commits*

`git rebase -i` *rewrite history, combine commits*

`git blame` *see who committed what line*

WILL COVER IN FUTURE

All things GitHub

- Forks
- Pull Requests
- Managing multiple remotes
- Issues

FURTHER RESOURCES

https://git-scm.com/doc

pcottle.github.io/learnGitBranching/

MAKE SCHOOL

MAKE
SCHOOL