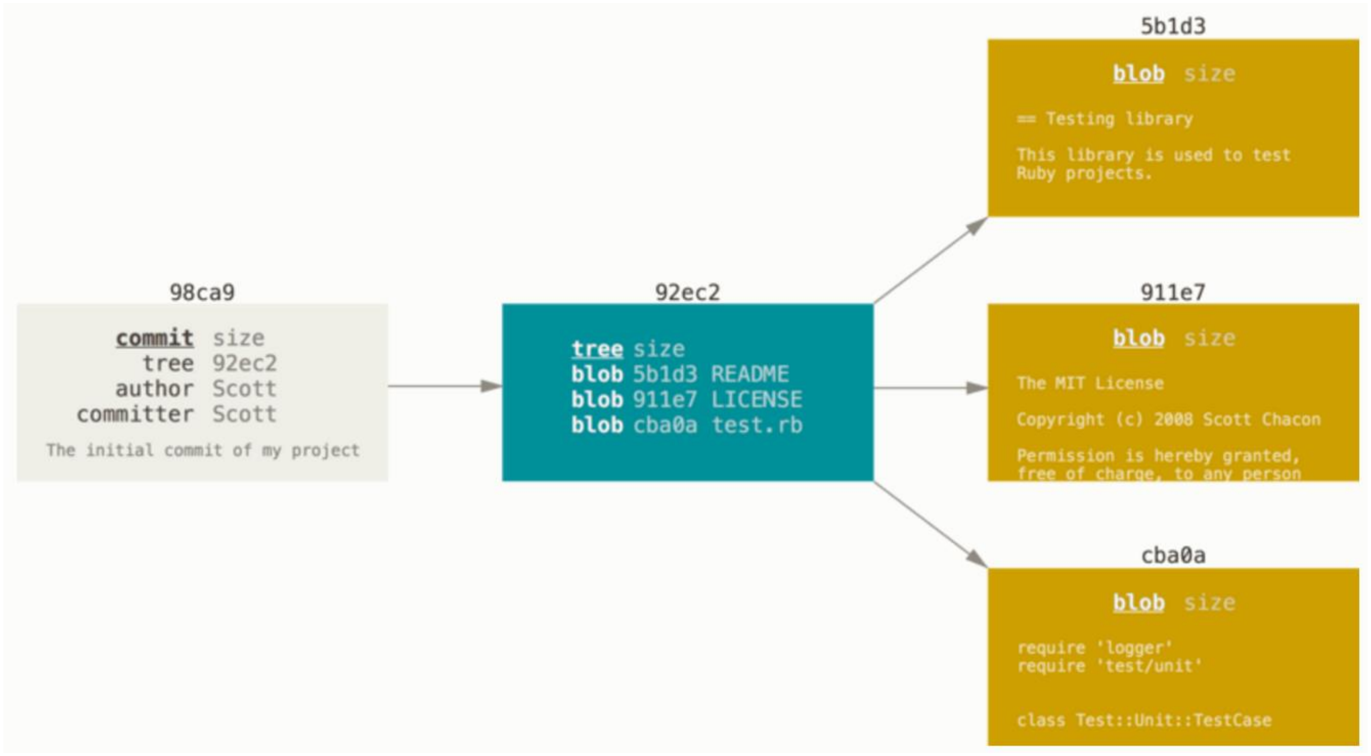
Chapter 04
GIT Branching

Open Source SW Development
CSE22300

Branching

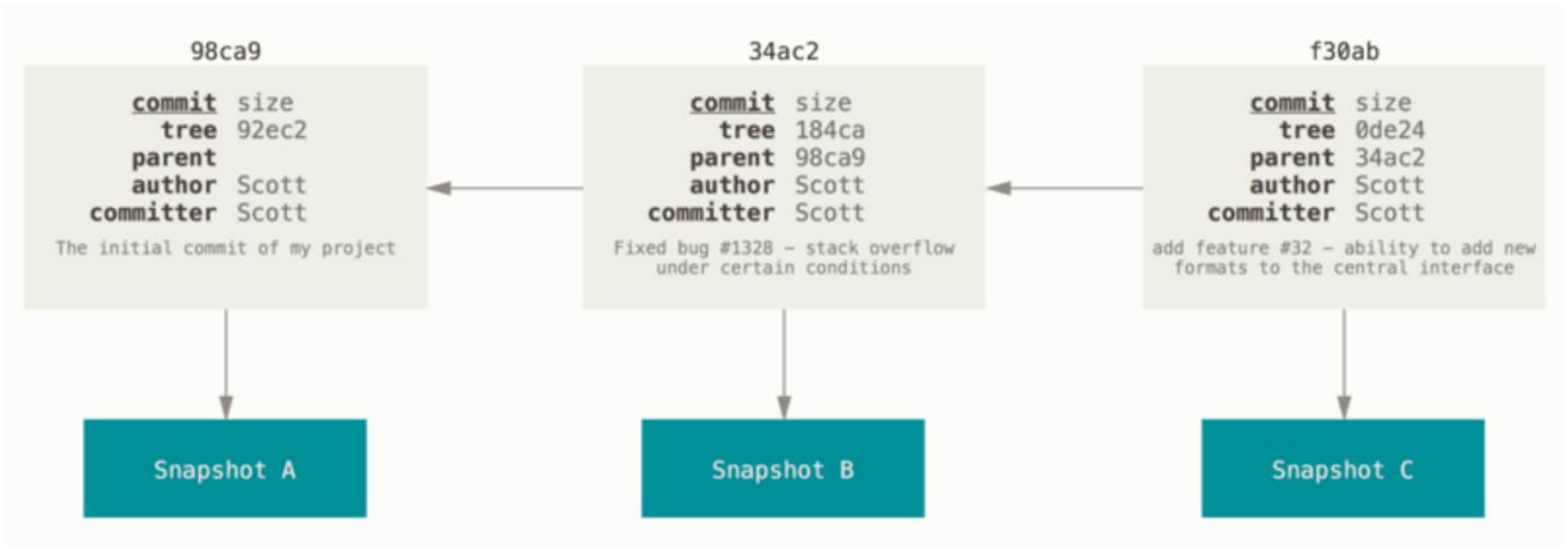
Snapshot

- **Commit Object**
  - **Contains a pointer to the snapshot of the content you staged**
  - **Name, Email, Message, Point**



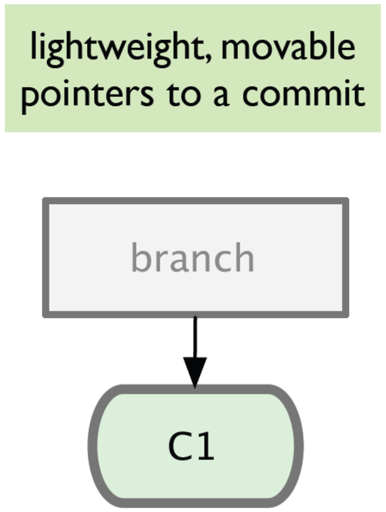
Snapshot

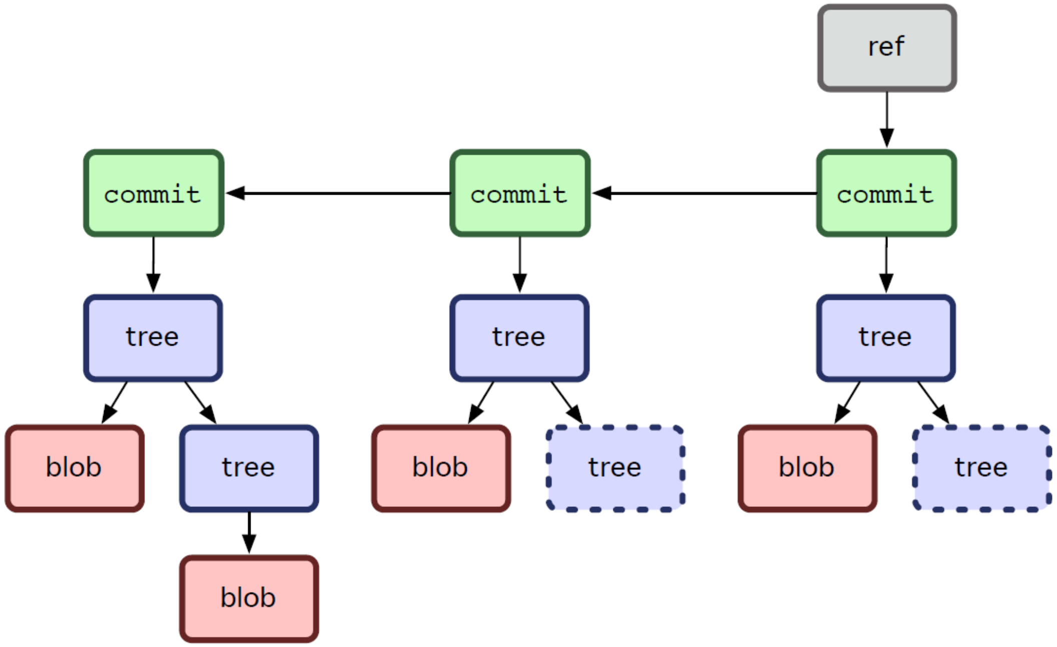
- Commit Chain
  - Parent-Child relation
  - The next commit stores a pointer to the commit



Branching

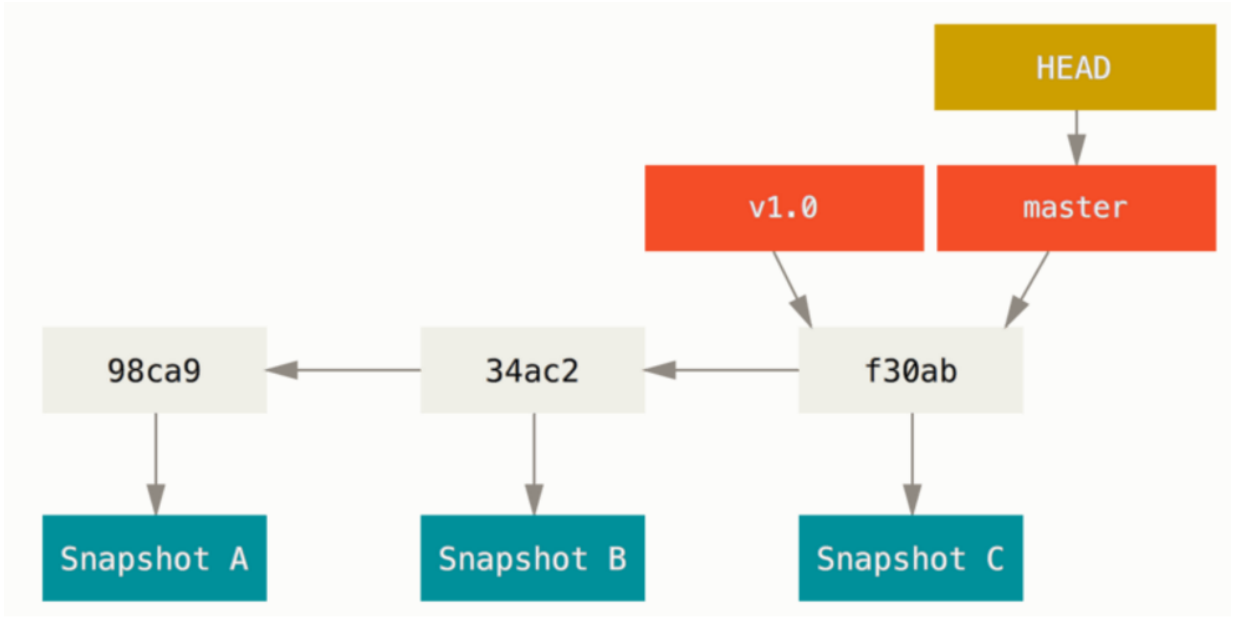
- **Git sees commit this way…**
- **Branch annotates which commit we are working on**

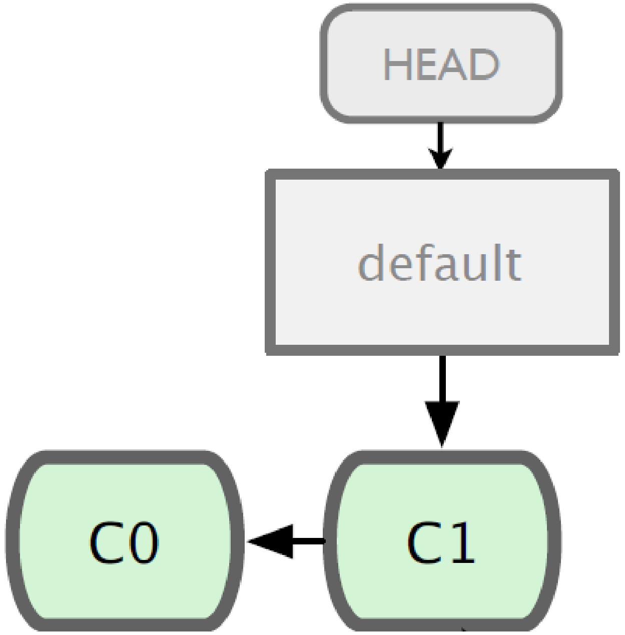


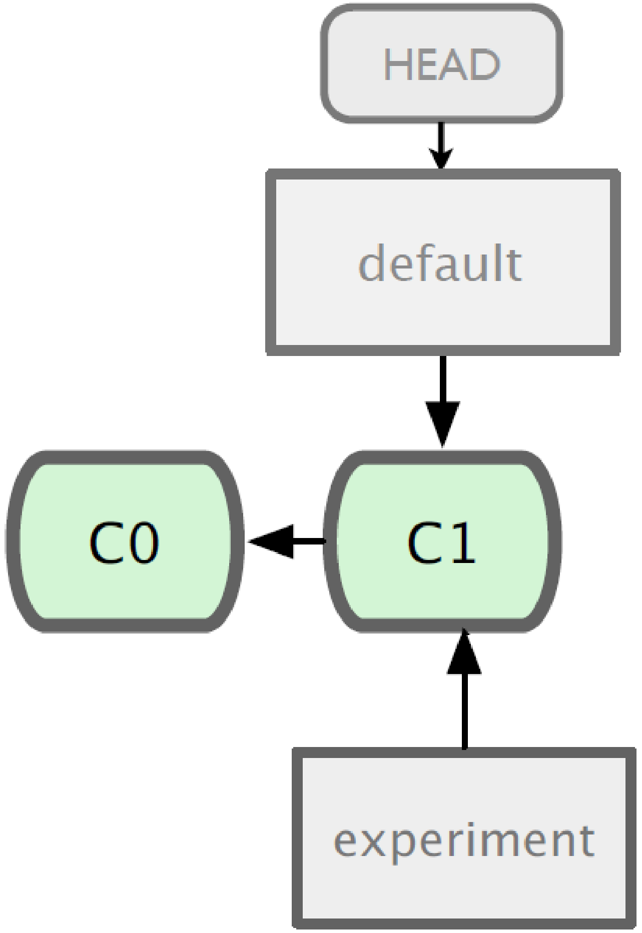


Branching

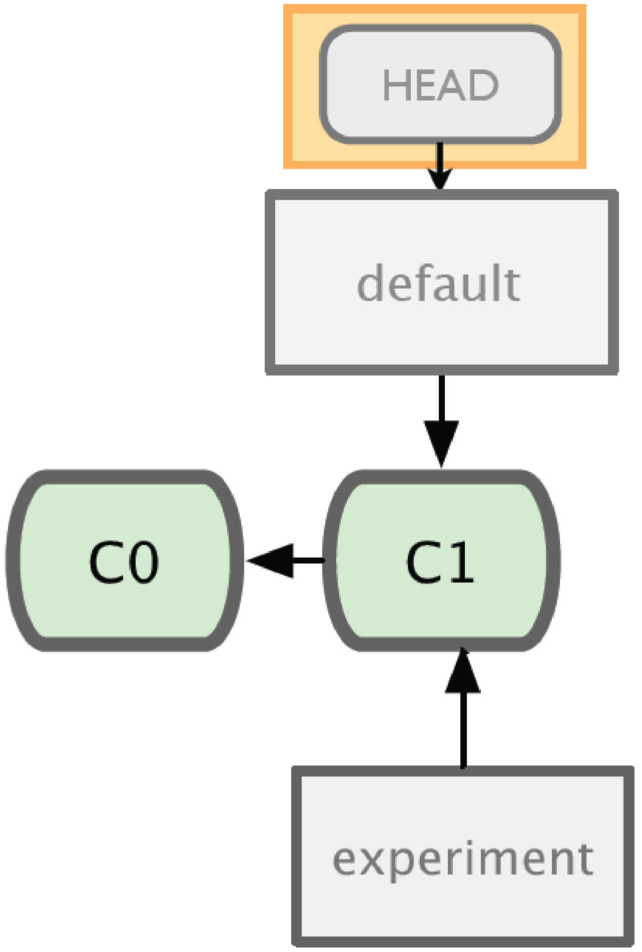
- **Master**
  - **Default branch in Git**
- **HEAD**
  - **Special Pointer**
  - **Points to the local branch you're currently on**



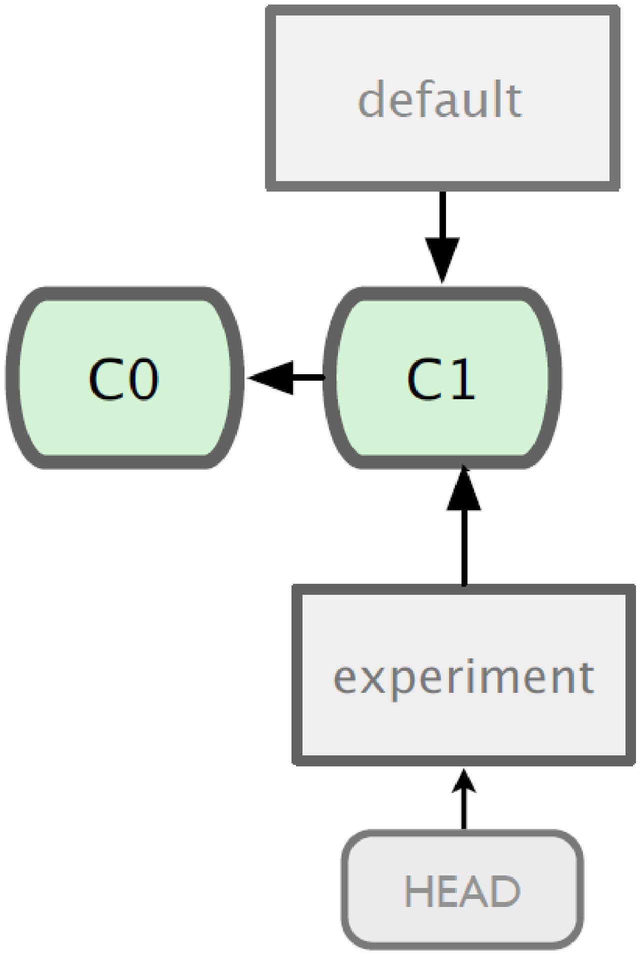

HEAD
default
C0
C1



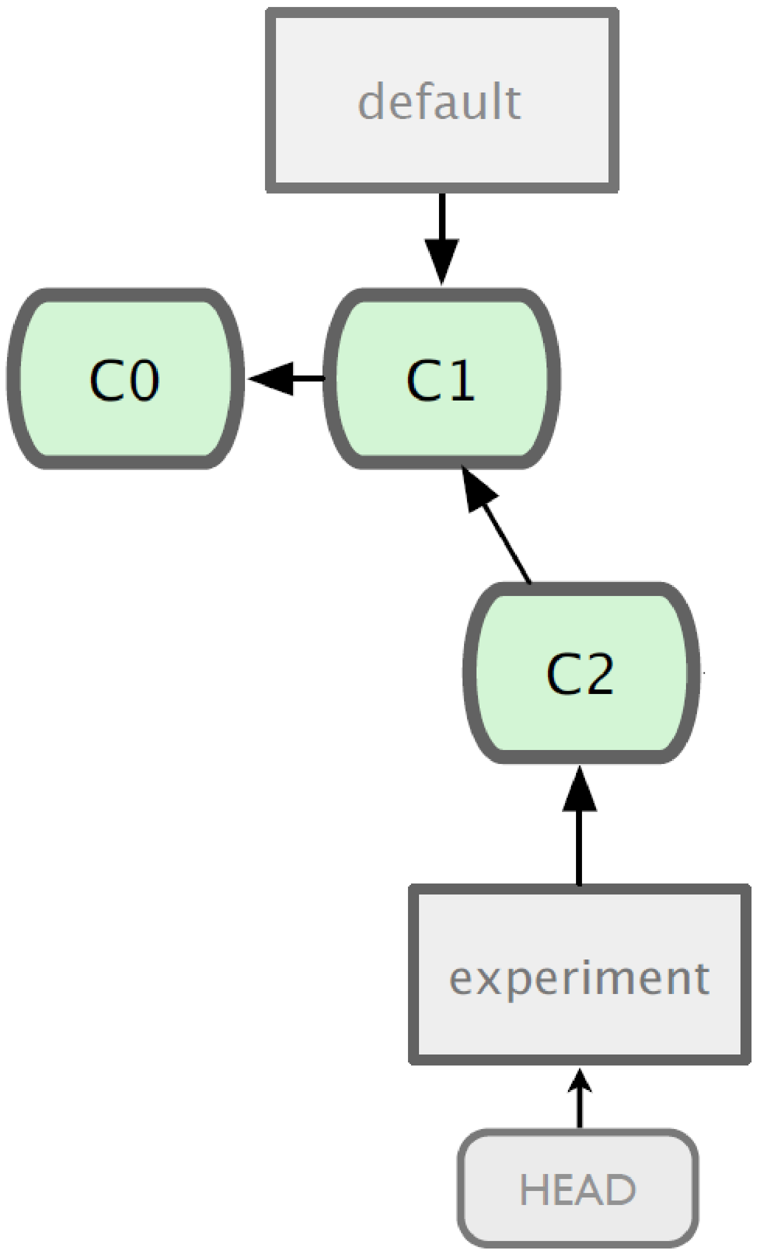
git branch experiment



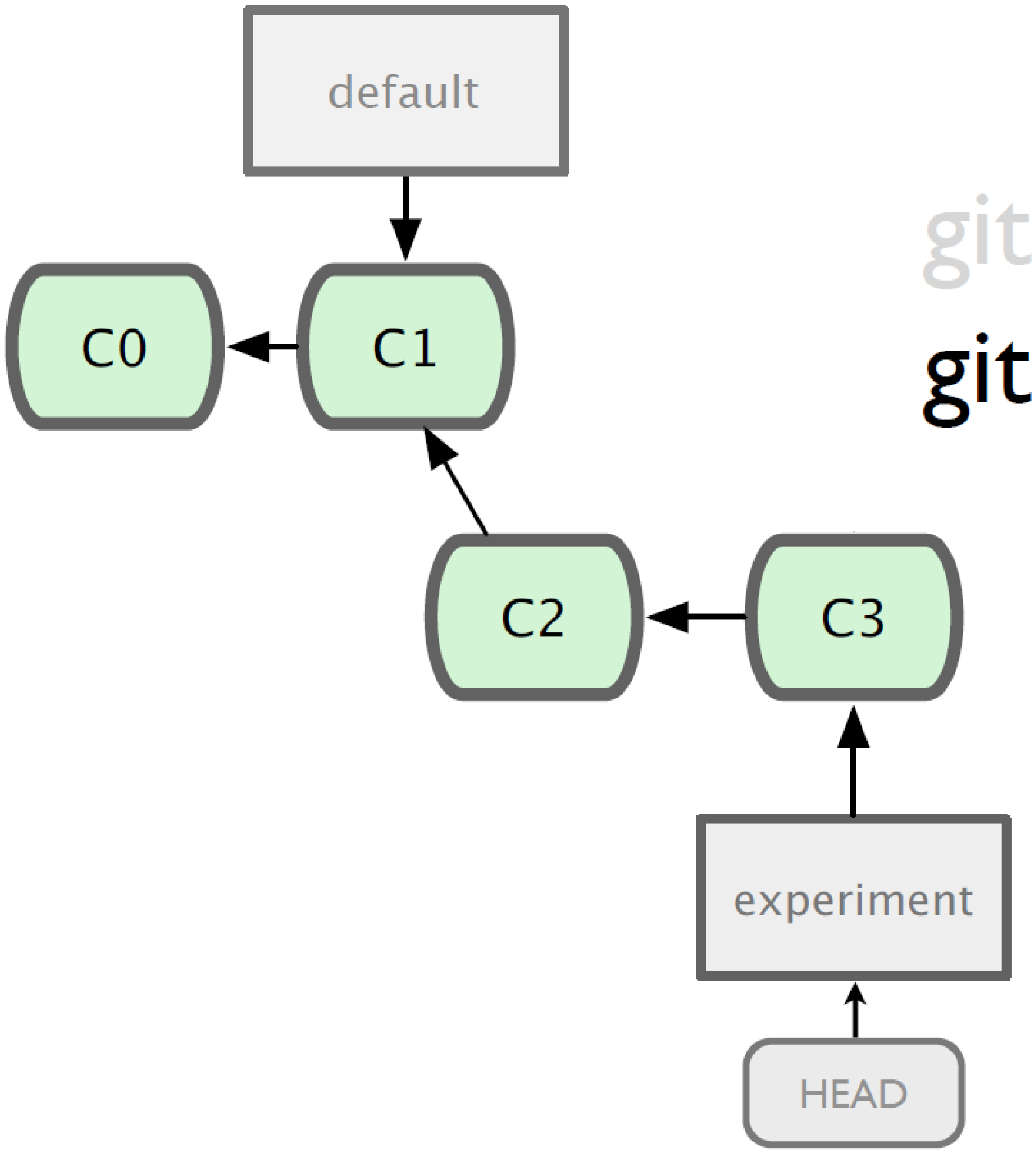
```
$ git branch
* default
  experiment
```



git checkout experiment



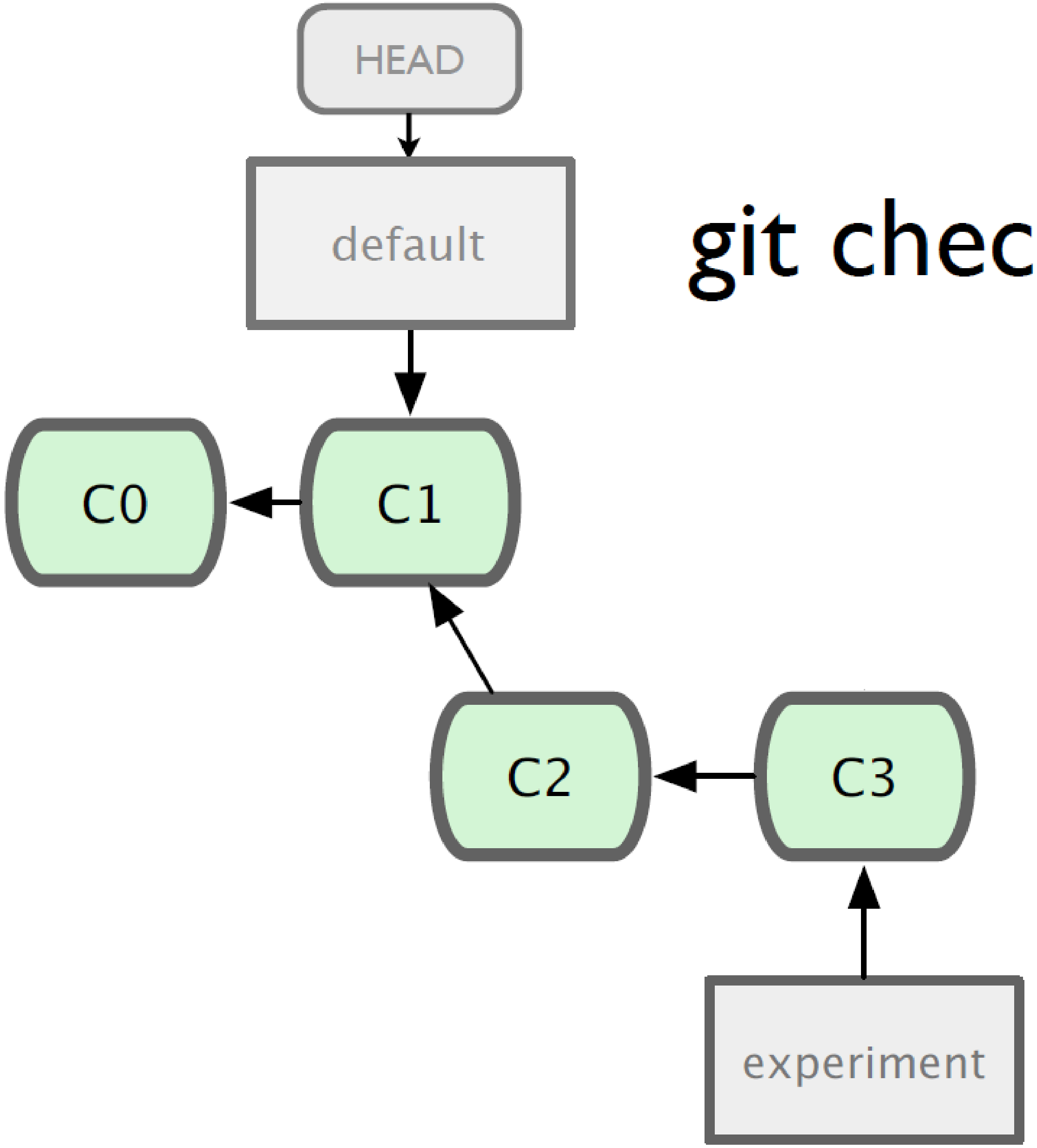
git commit



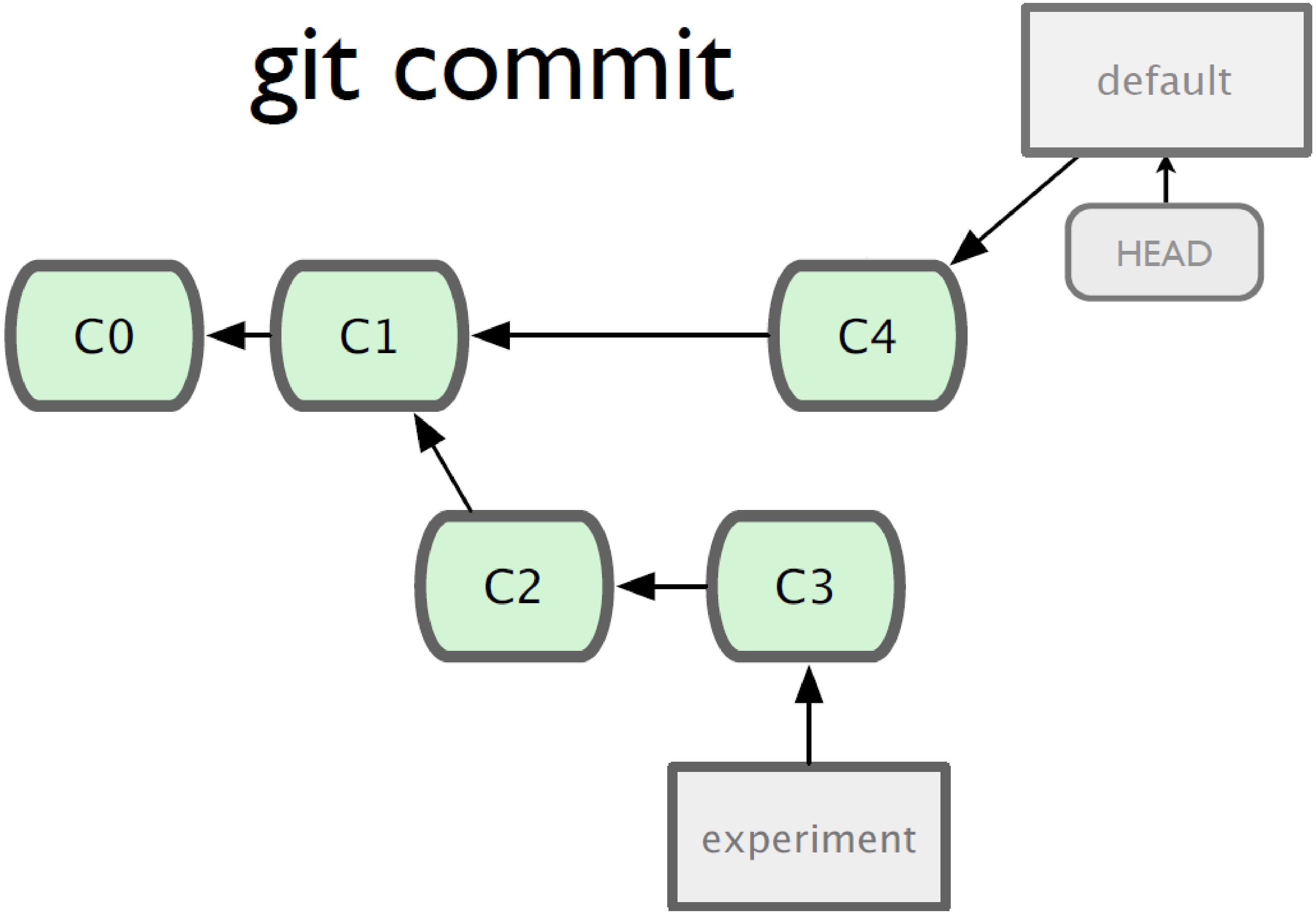
git commit

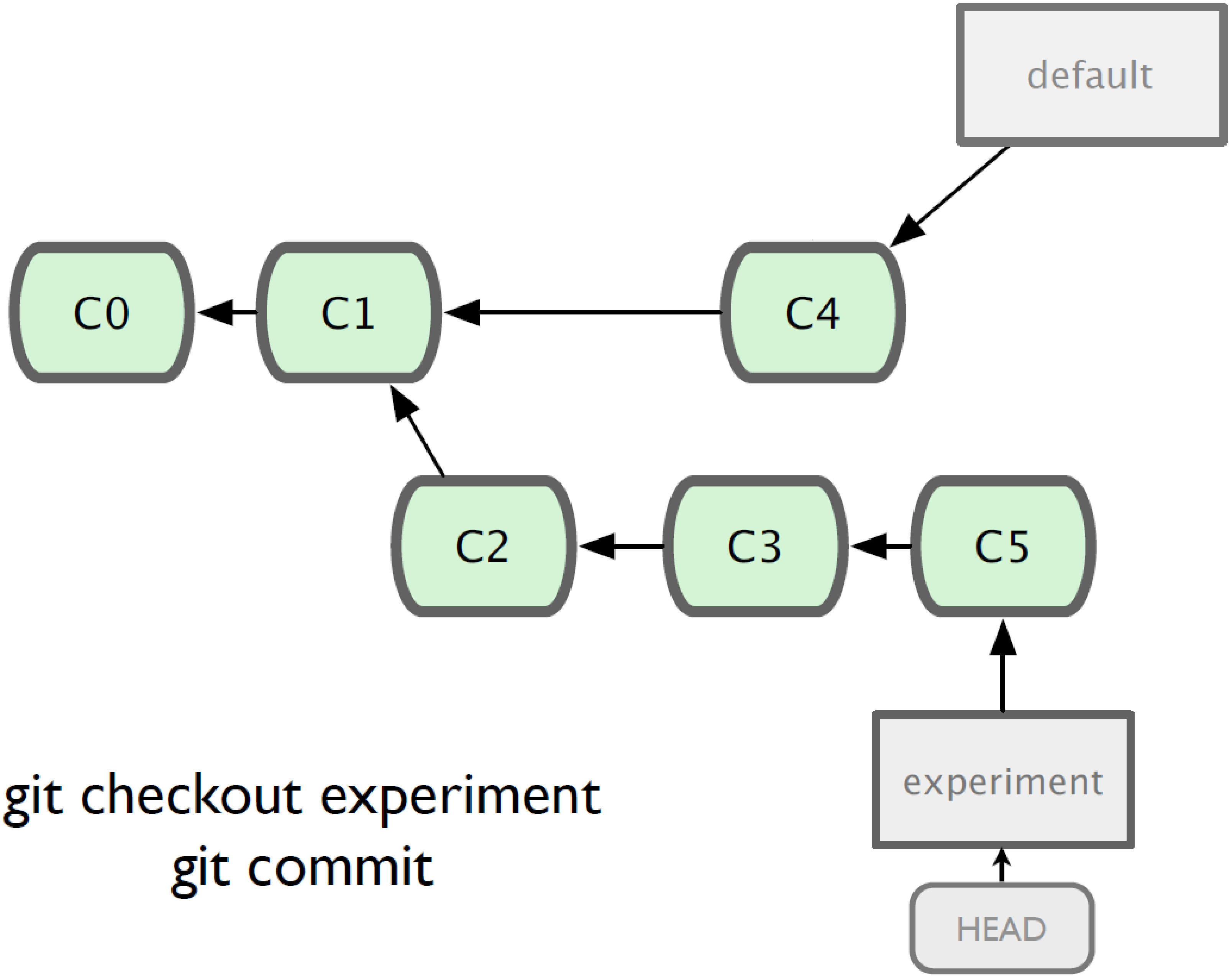
git commit

git checkout default



git commit

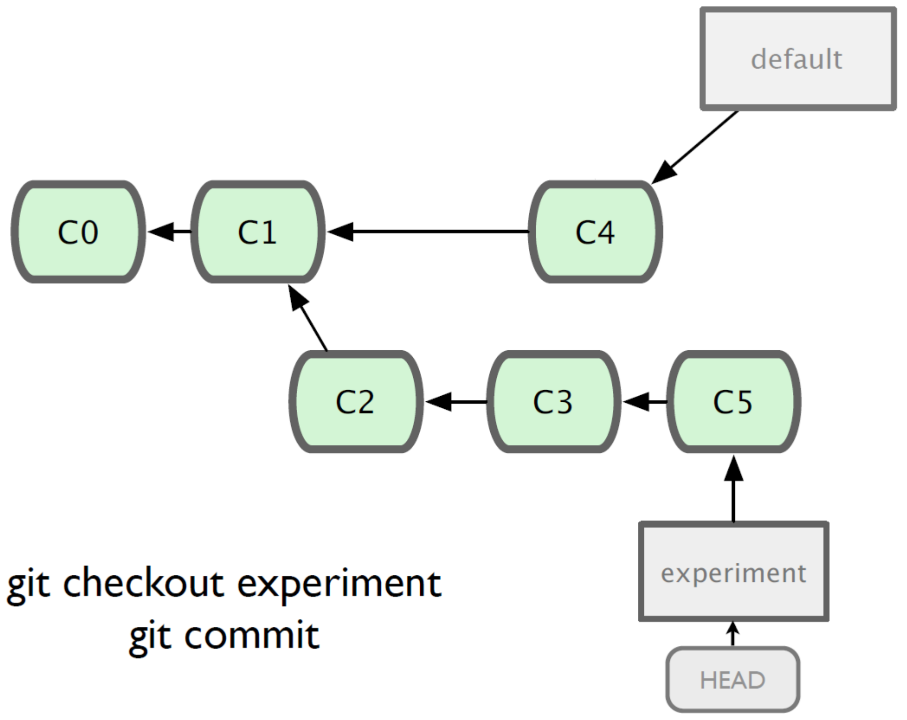




git checkout experiment
git commit

Merging

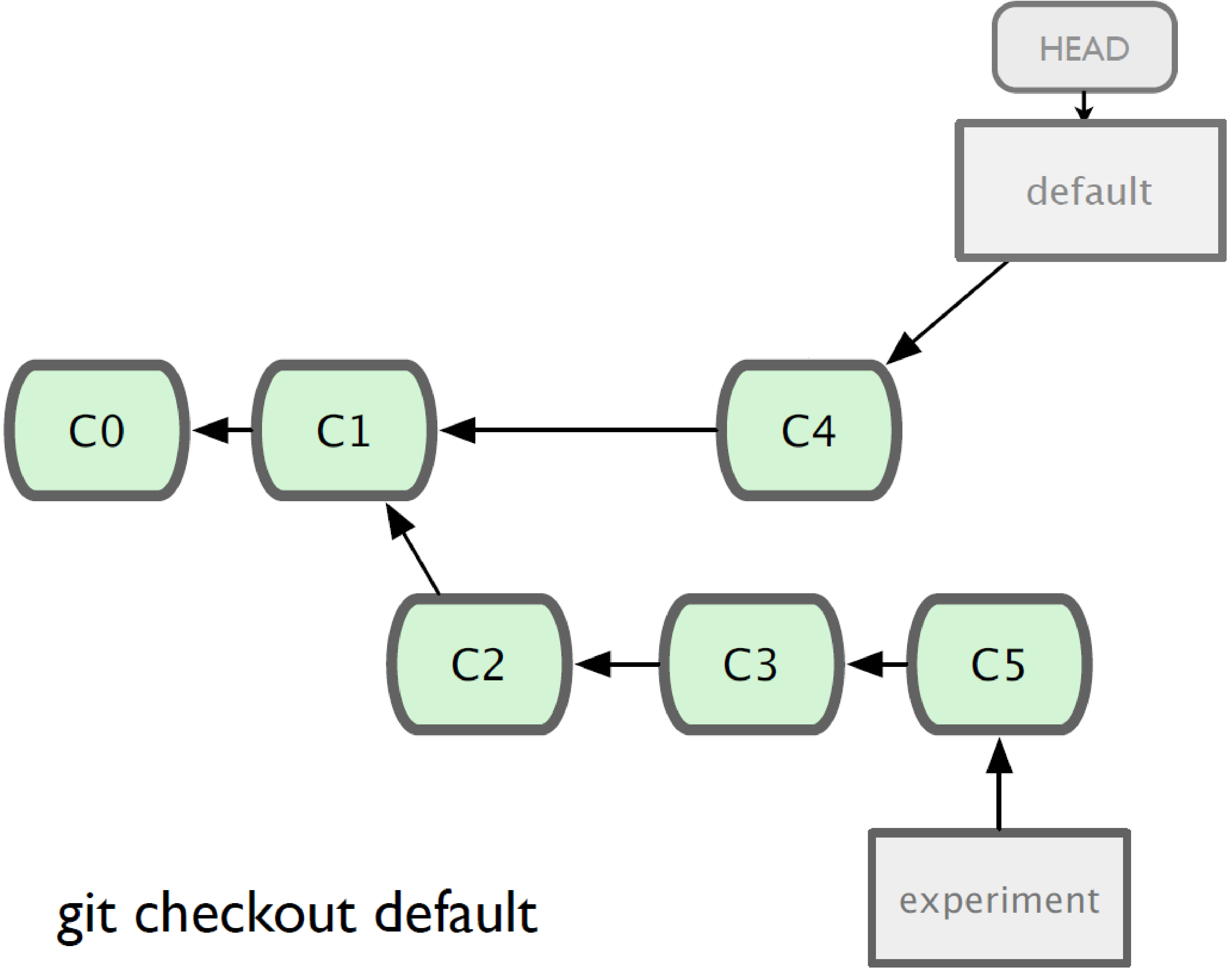
- **What do we do with this mess?**
  - **Merge them**



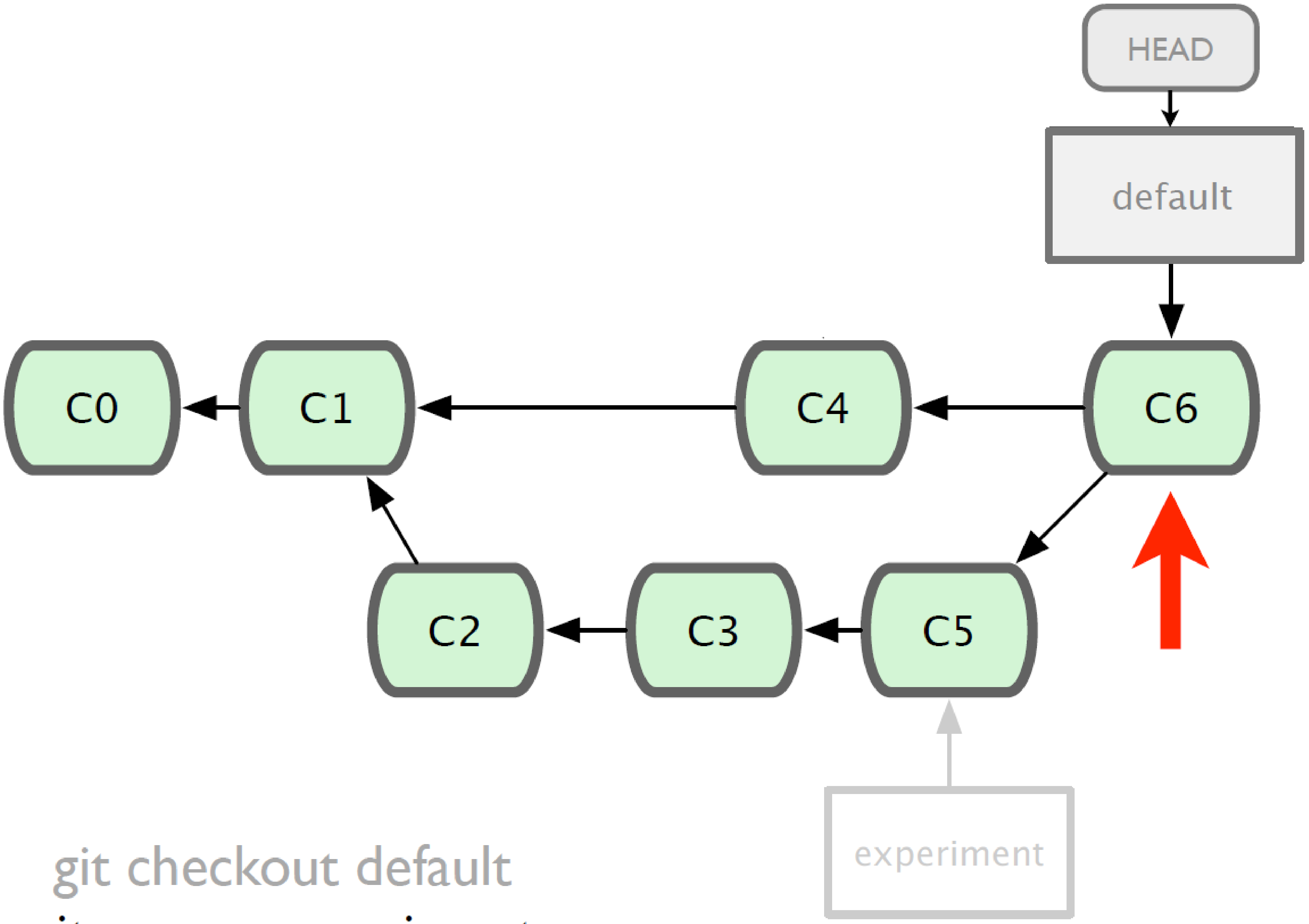
git checkout experiment
git commit

Merging

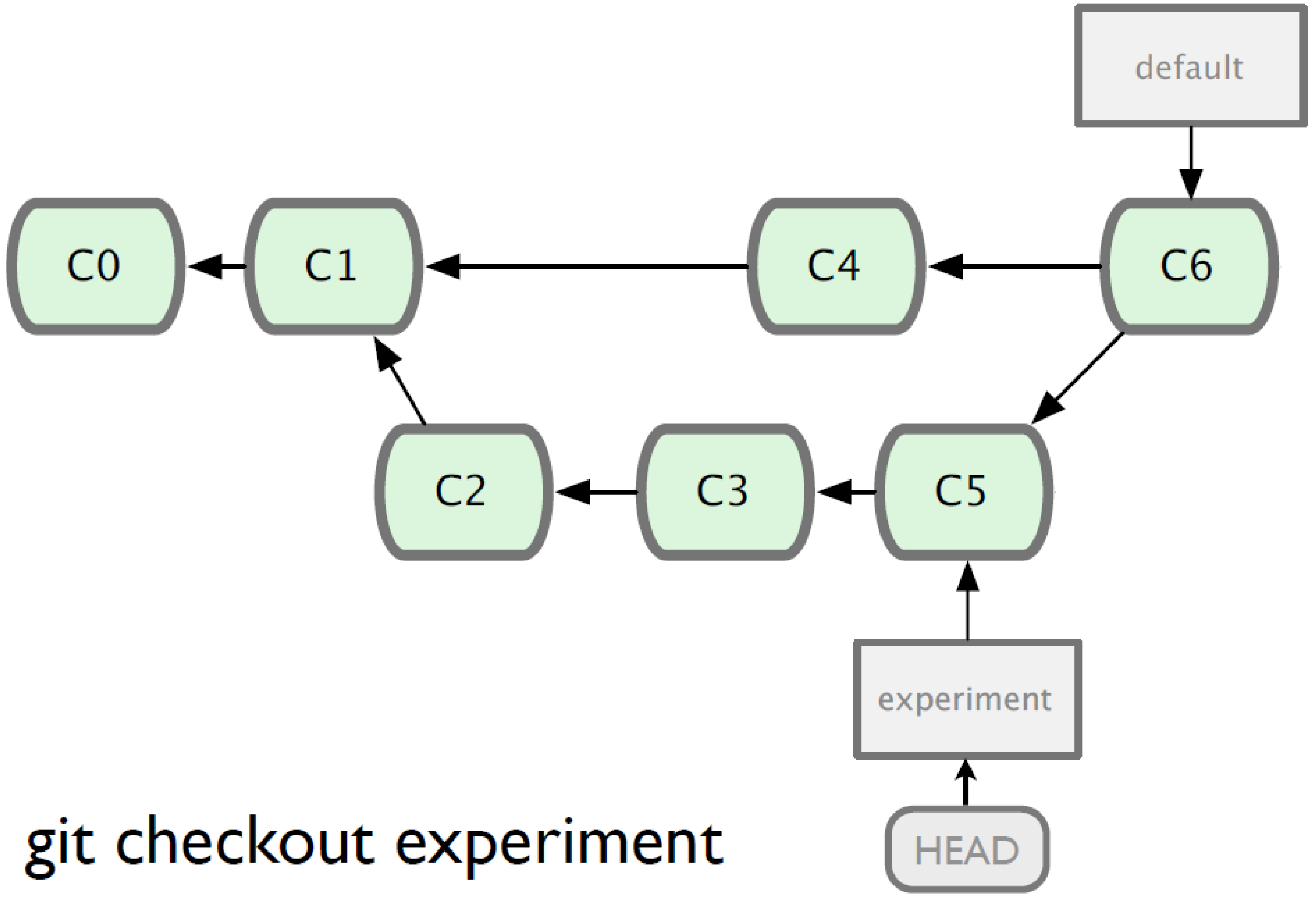
- Steps to merge two branch
  - Checkout the branch you want to merge onto
  - Merge the branch you want to merge



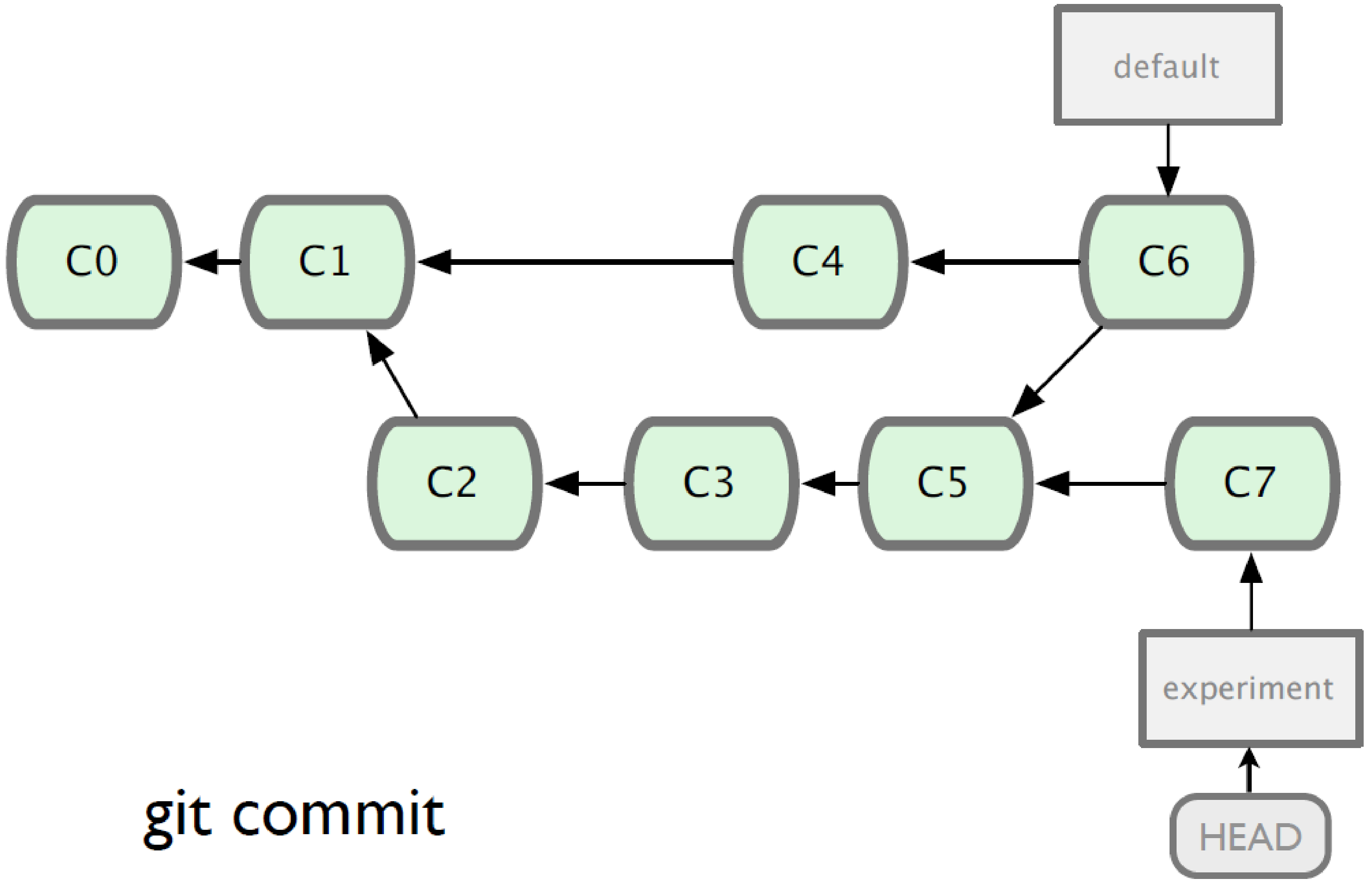
git checkout default



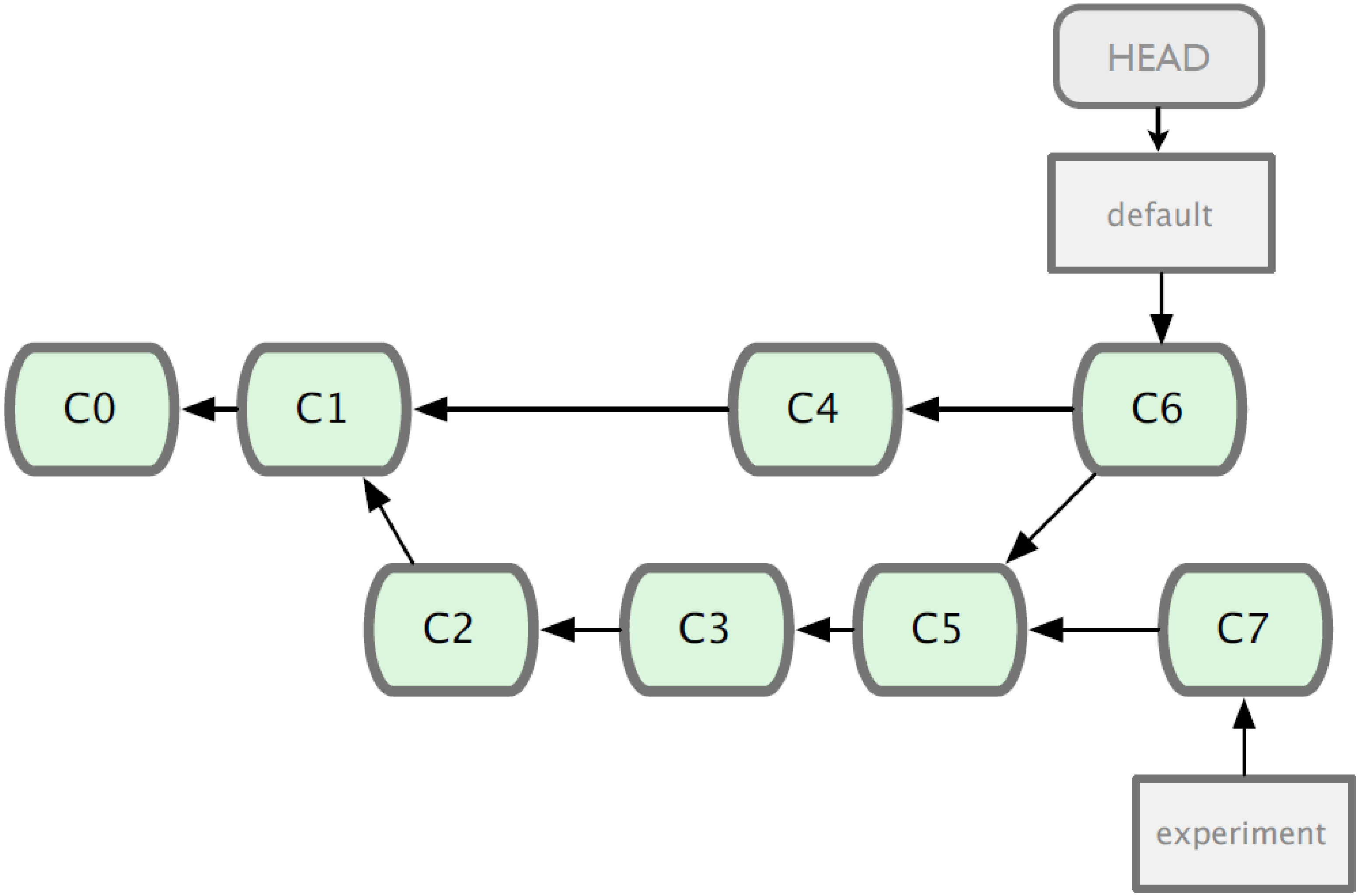
```
git checkout default
git merge experiment
```



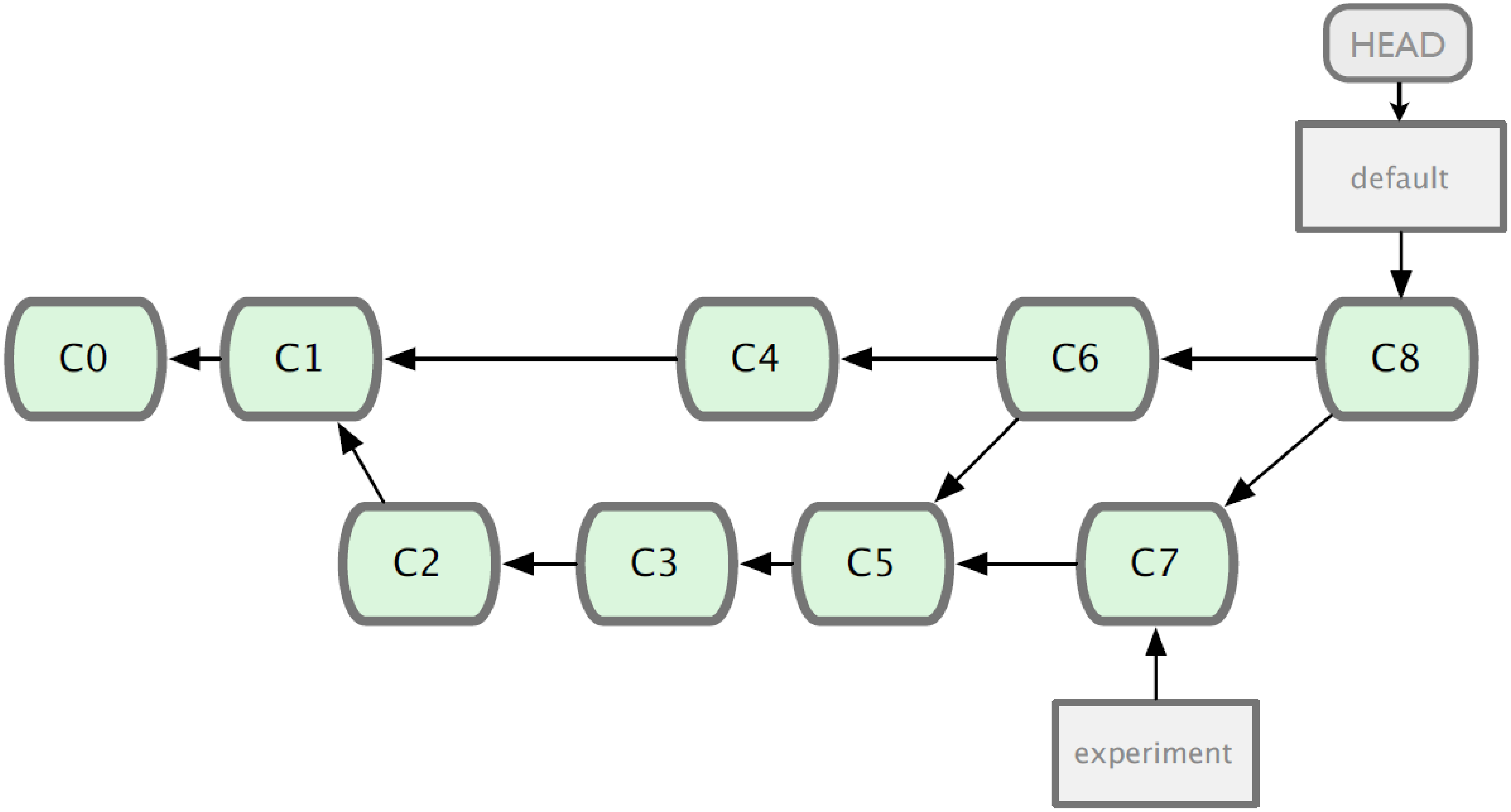
git checkout experiment



git commit



git checkout default



git merge experiment

Branching and Merging

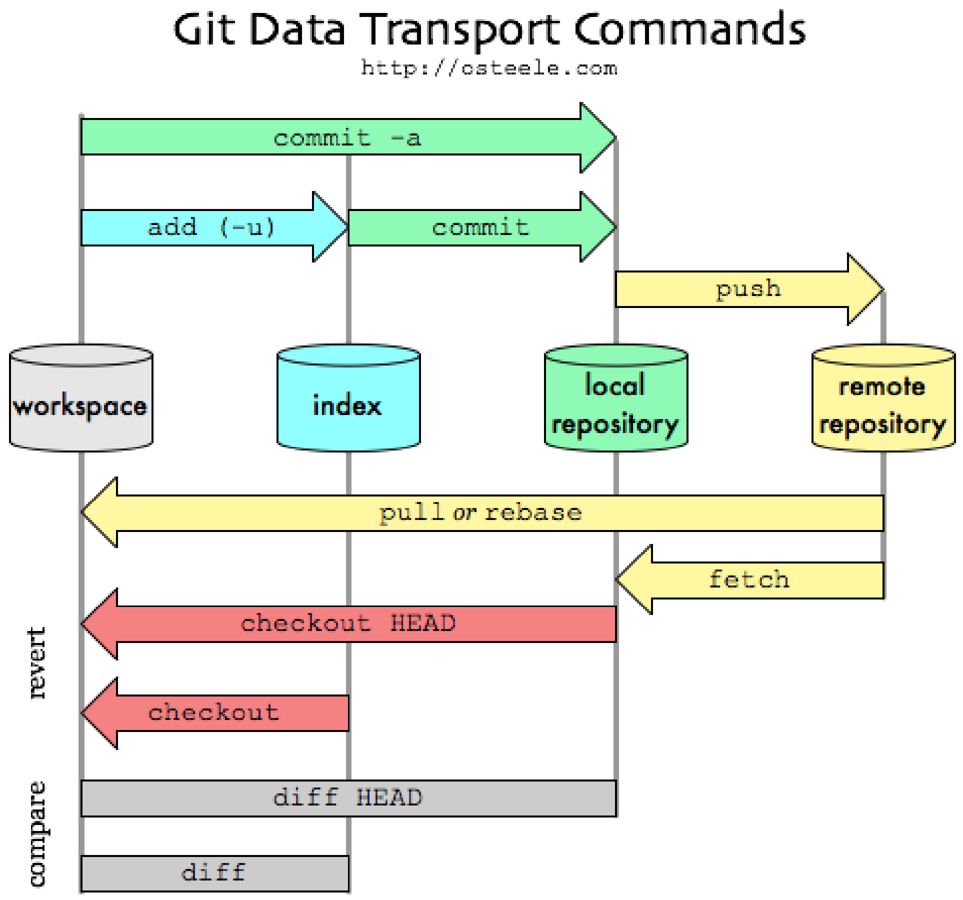
- **Why this is cool?**
  - **Non-linear development**

```
clone the code that is in production
create a branch for issue #53 (iss53)
work for 10 minutes
someone asks for a hotfix for issue #102
checkout ‘production’
create a branch (iss102)
fix the issue
checkout ‘production’, merge ‘iss102’
push ‘production’
checkout ‘iss53’ and keep working
```

Remote Repository

Working with Remote Repositories

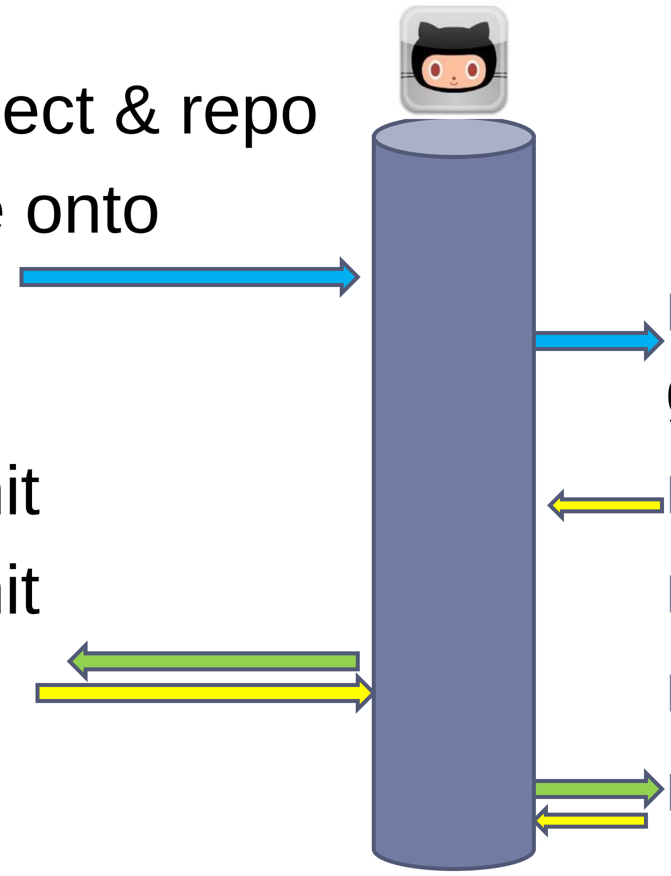
- **Commands for synching with remote repositories**
  - **clone, push, fetch, pull**



Typical Workflow

Person A

- Setup project & repo
- push code onto github
- edit/commit
- edit/commit
- pull/push



Person B

- clone code from github
- edit/commit/push
- edit…
- edit… commit
- pull/push

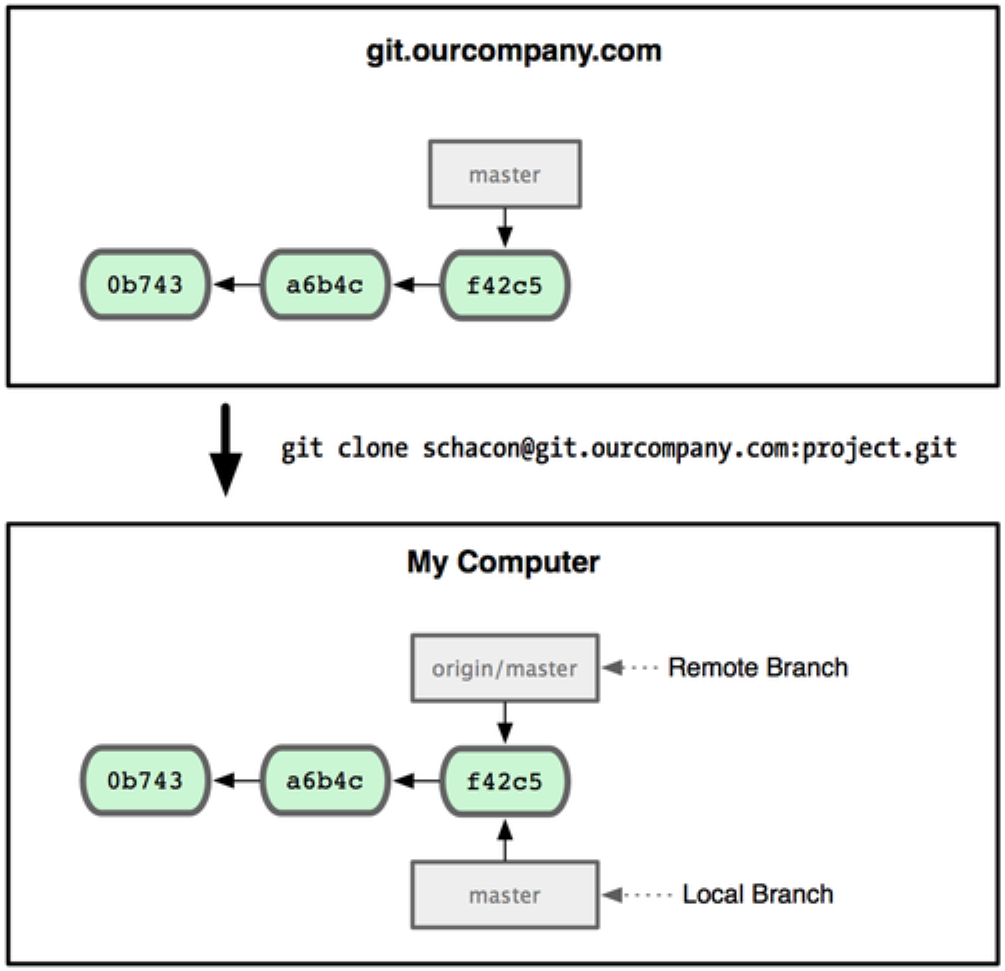
This is just the flow, specific commands on following slides.
It's also possible to create your project first on github, then clone (i.e., no git init)

Remote Branch

- All the branching we’ve done so far has been local
- Remote repository is a bare repository
  - Remote repositories (e.g., github) also have branches
  - There is no working directory

- Four transfer protocols
  - http – this is what I recommend/use
  - local (not covered – good for shared filesystems)
  - git (not covered – fast but more setup)
  - SSH (supplementary material at end of slides, not covered)

Remote Branch



```
git clone schacon@git.ourcompany.com:project.git
```

Git server
right now, only have master

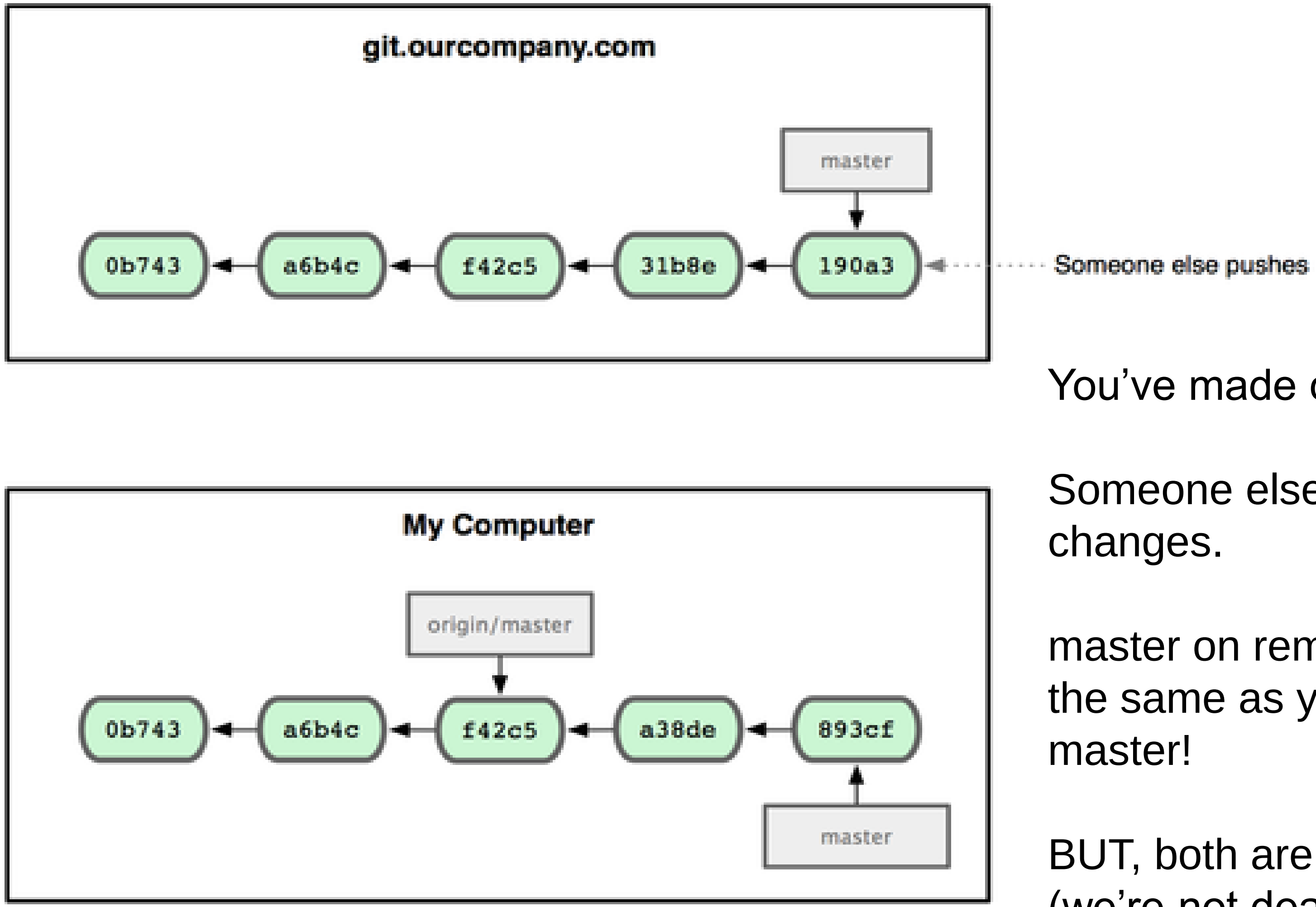
clone files from server into your working directory.

Remote branch – on server – [remote]/[branch]

clone names your remote *origin* by default

Local branch – [branch]

Remote Branch



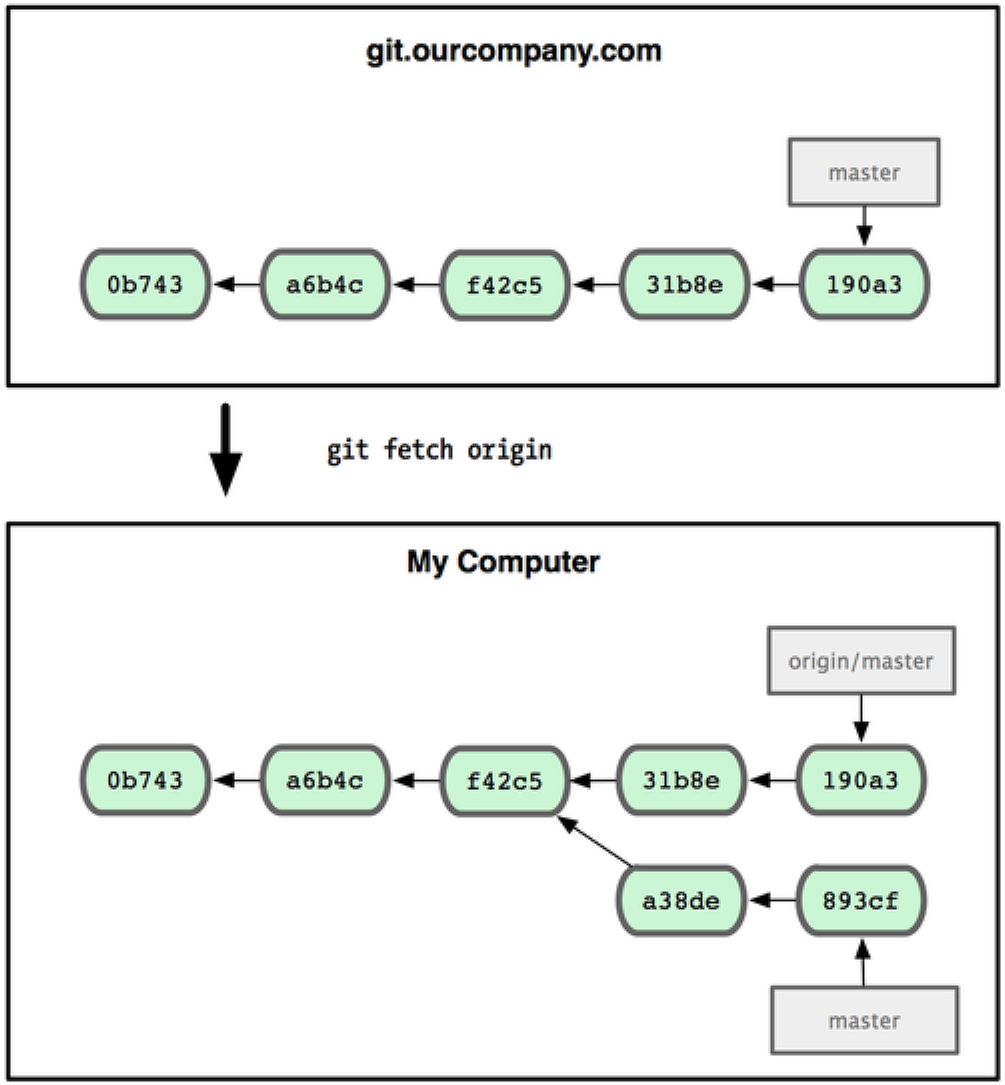
You've made changes.

Someone else pushed changes.

master on remote NOT the same as your master!

BUT, both are master (we're not dealing with local branches yet)

Remote Branch



note: fetch doesn’t merge!
Need to:

- git fetch origin
- git merge origin/master
- (handle conflicts if any, note that branch name is origin/master)
- git push

You can also do:
git pull origin master
(does fetch and merge)

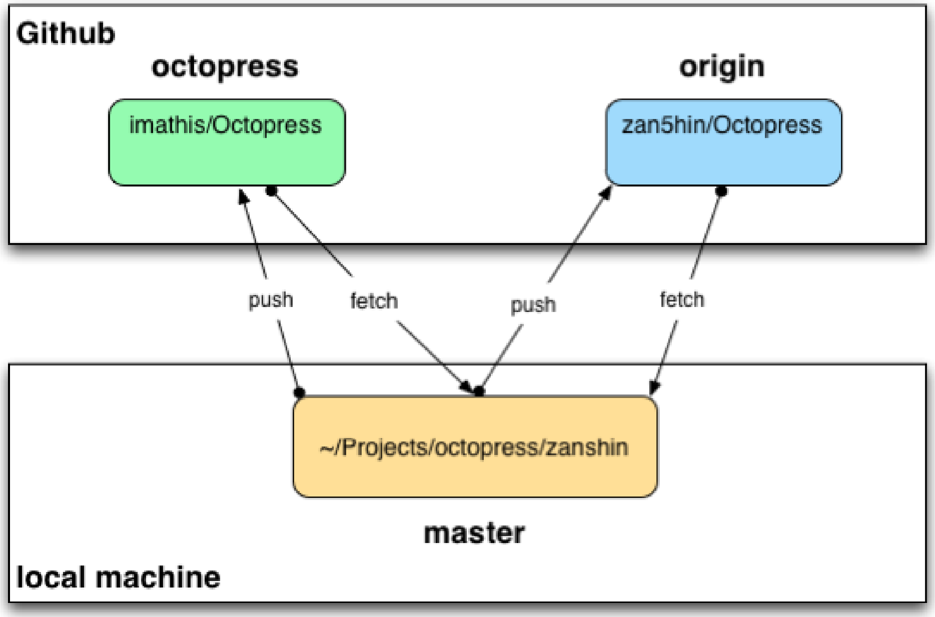
Tracking Branch

- Tracking Branch
  - Local branch that has a direct relation to a remote branch
  - If you’re on a tracking branch and type git push, Git knows which server and branch to push to

- git pull
  - Fetches remote references and merges
- git clone
  - Automatically creates a master branch and tracks origin/master
- git checkout –track
  - Add other tracking branches

Forking

- If you want to contribute to a project but don't have push access, you can do a fork… create your own copy.
- Main project can pull in those changes later by adding them as remotes and merging in the code from the fork.



Clone

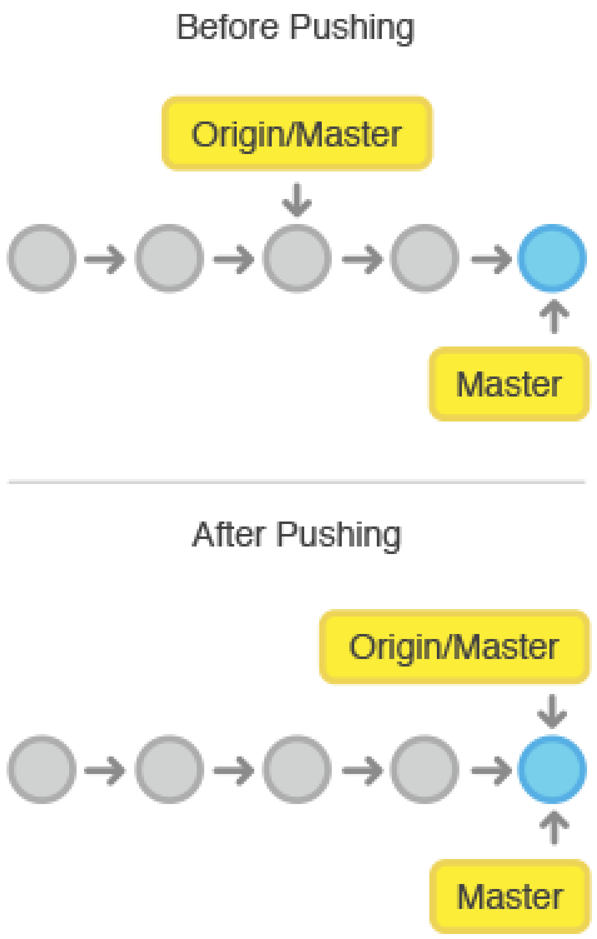
- Clone
  - Creates a local repository starting from a remote one.
- Basic usage.
  - git clone <repository>
- Example:
  - git clone ssh://git@khuhub.khu.ac.kr:Prof.JinSeongwook/LectureNotes.git
  - Other protocols are available depending on the server installed.

Push

- git push
  - If there are tracking branches, pushes commits from those to the remote ones
  - Non tracking branches are ignored

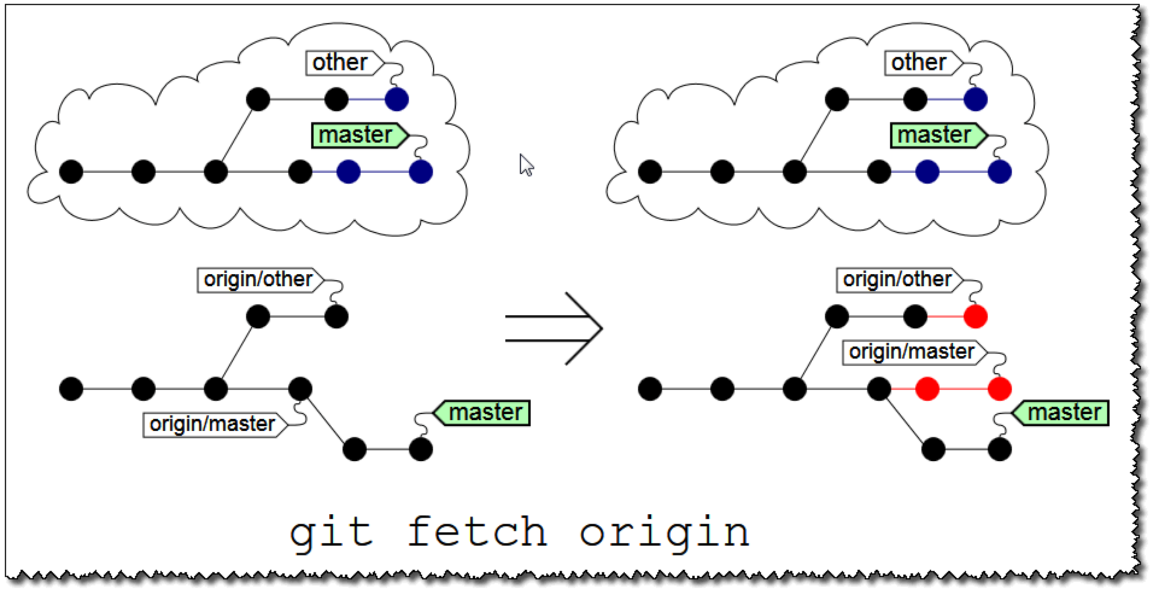
- Create a new remote branch
  - git push origin <new branch>

– Delete a remote branch
  - git push origin :<remote branch>



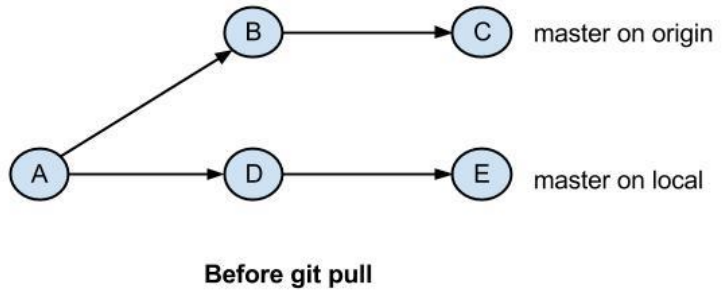
Fetch

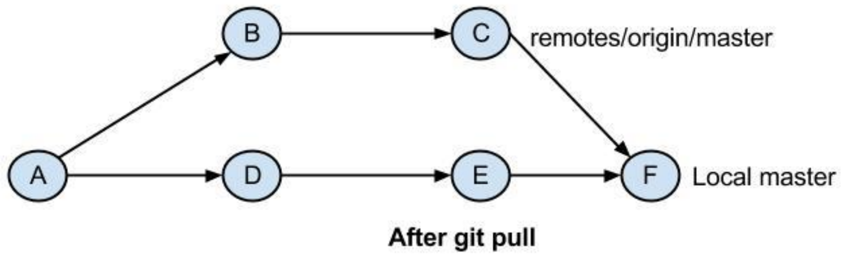
- **git fetch**
  - **Fetch allows to receive commits from remote repositories**
  - **Only for tracking branches – does not touch the current branch**
- **git fetch origin <branch>**
  - **Fetches that specific branch but keeps it in the special commit FETCH_HEAD, not attached to any branch**



Pull

- **git pull**
  - **git fetch + git merge**
- **This command WILL change your local checkout**


B
C
master on origin
A
D
E
master on local
Before git pull


B
C
remotes/origin/master
A
D
E
F
Local master
After git pull