

Git

网易有道 张宇辰

Agency

- Git 历史
- 为什么用 Git
- Git 基础知识
- 基本命令
- 暂存区与指针控制 (git reset/git checkout)
- 与 SVN 协同 (git svn)

Git 历史

开源社区又一神作

Linus Torvalds

- Linux 项目的发起者
- 痛恨 VCS&SVN
- 大神、脾气火爆、非常固执



1991~2002 大神只用邮件收集开源社区的 patch, 手工集成到自己电脑里

手工集成.....手工.....手工.....伤不起啊！



开源社区

2002~2005 大神终于选了 Bitlocker 做版本控制, 一个商业软件



开源社区

商业软件.....商业.....商业.....

2005 年， Bitlocker 结束了对开源社区的授权。

2005 年， Bitlocker 结束了对开源社区的授权。

我只是好奇...



Andrew Tridgell

2005 年， Bitlocker 结束了对开源社区的授权。



Andrew Tridgell

我只是好奇...

一夜回到解放前



开源社区

2005 年， Bitlocker 结束了对开源社区的授权。



Andrew Tridgell

我只是好奇...

一夜回到解放前



开源社区



Linus Torvalds

咱自己写一个！

- 2005 年 4 月 3 日，开始开发 Git
- 同年 4 月 6 日，项目发布
- 同年 4 月 7 日，Git 作为自身的项目控制工具，仅仅用了 5 天
- 同年 4 月 18 日，第一次多分支合并
- 同年 4 月 29 日，Linus 表示对 Git 的性能满意了
- 同年 6 月 16 日，Linux 2.6.12 发布，开始用 Git 维护，此时仅仅经过 74 天

Git 今日



debian



Perl

github

3,500,000+ repositories



MeeGo

Qt



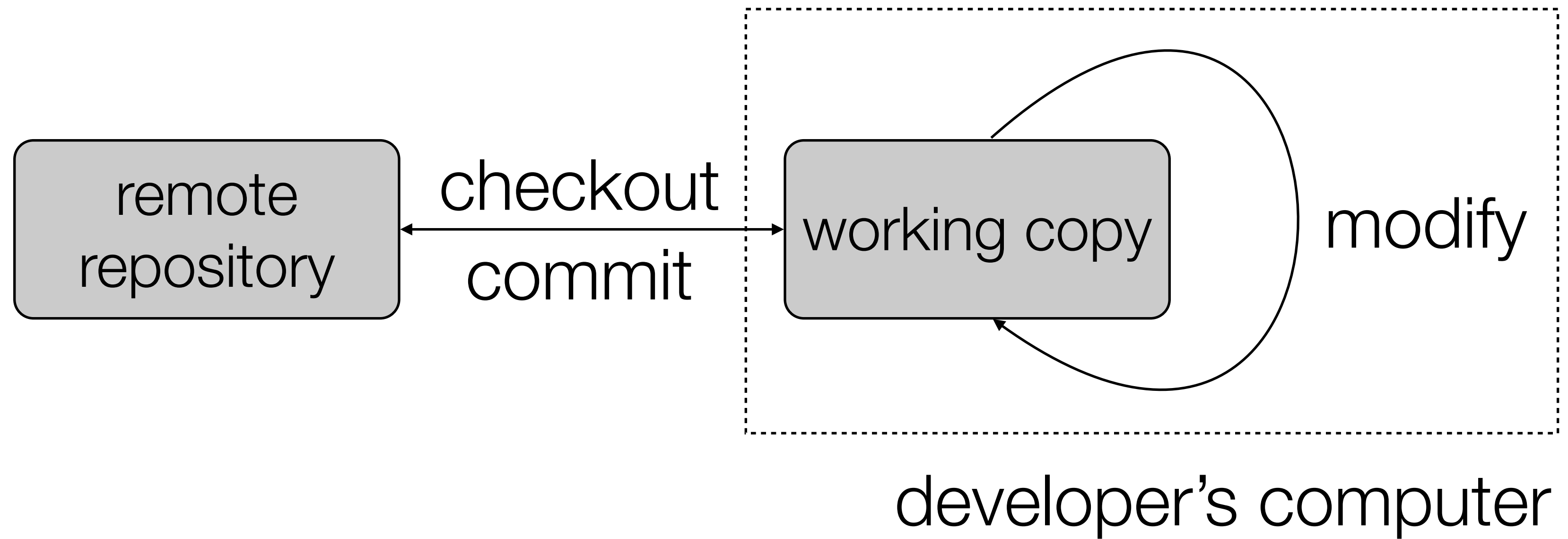
为什么选择 Git

SVN 为什么不爽

Bad case

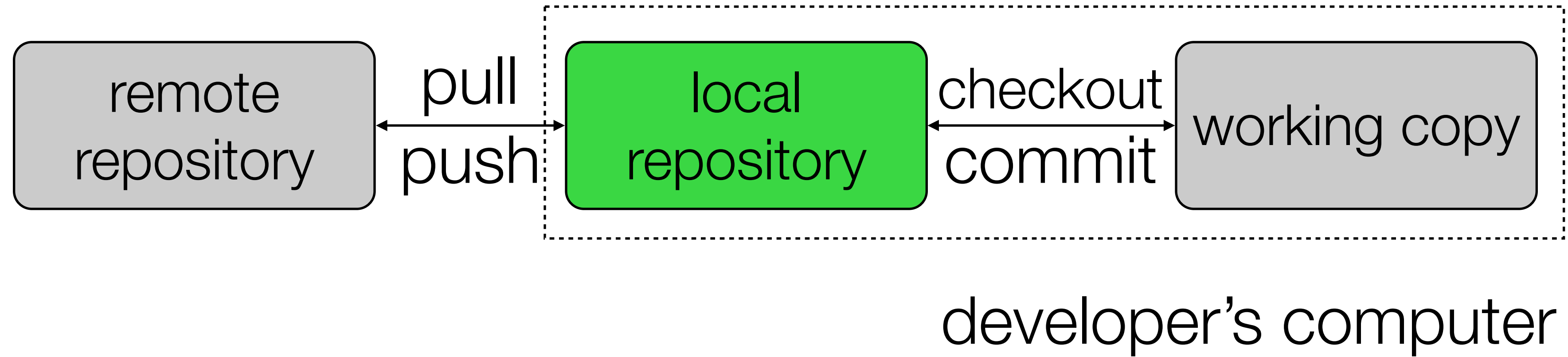
- 开发过程中接到紧急需求
- 等待 review 的代码污染工作区
- 离线开发怎么办
- 写 Log 时不知道写什么

SVN

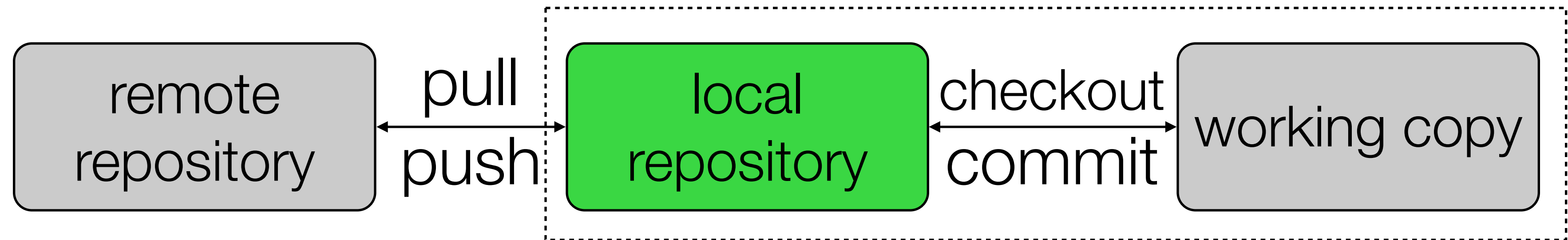


所有的修改只能发生在没有版本控制的本地 working copy 中

Git



Git



developer's computer

速度快

历史记录可控

可以离线工作

Git -- DVCS

remote
repository

devA
repository

devB
repository

...

Distributed Version Control System

除此之外

- 无处不在的自动分页 (less)
- diff 支持逐字比较 (git diff --word-diff)
- 版本库又小又快
- 支持多种网络传输协议 (http/ssh) 以及更强大的 git 协议, 可以显示传输进度
- 丰富且简单的 config 配置

安装 Git

Linux & Mac

- Git: <http://git-scm.com>



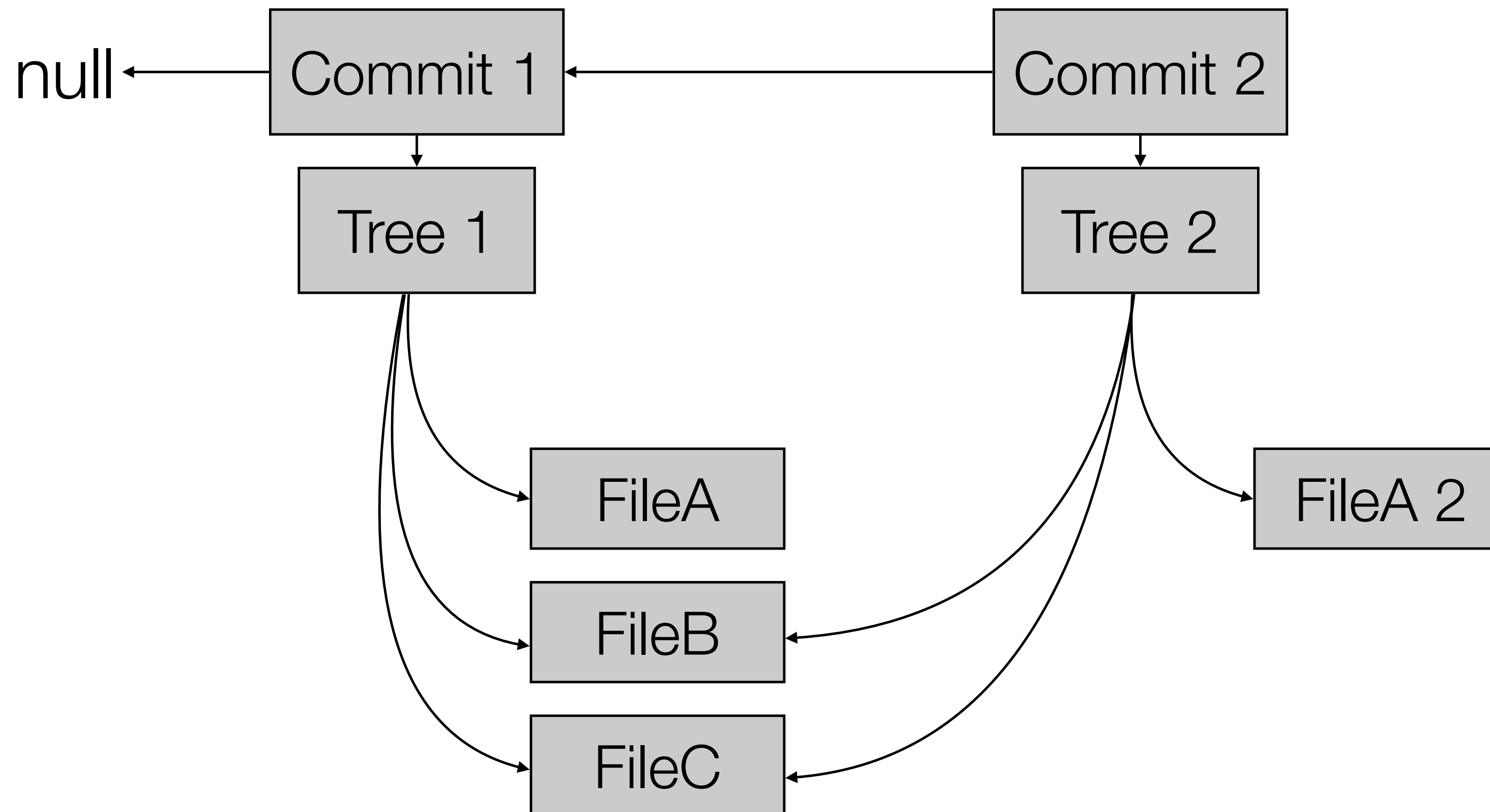
Windows

- Cygwin + git (推荐)
 - 命令更丰富
- msysGit + TortoiseGit
 - 图形界面
 - 对中文目录名和文件名处理有问题

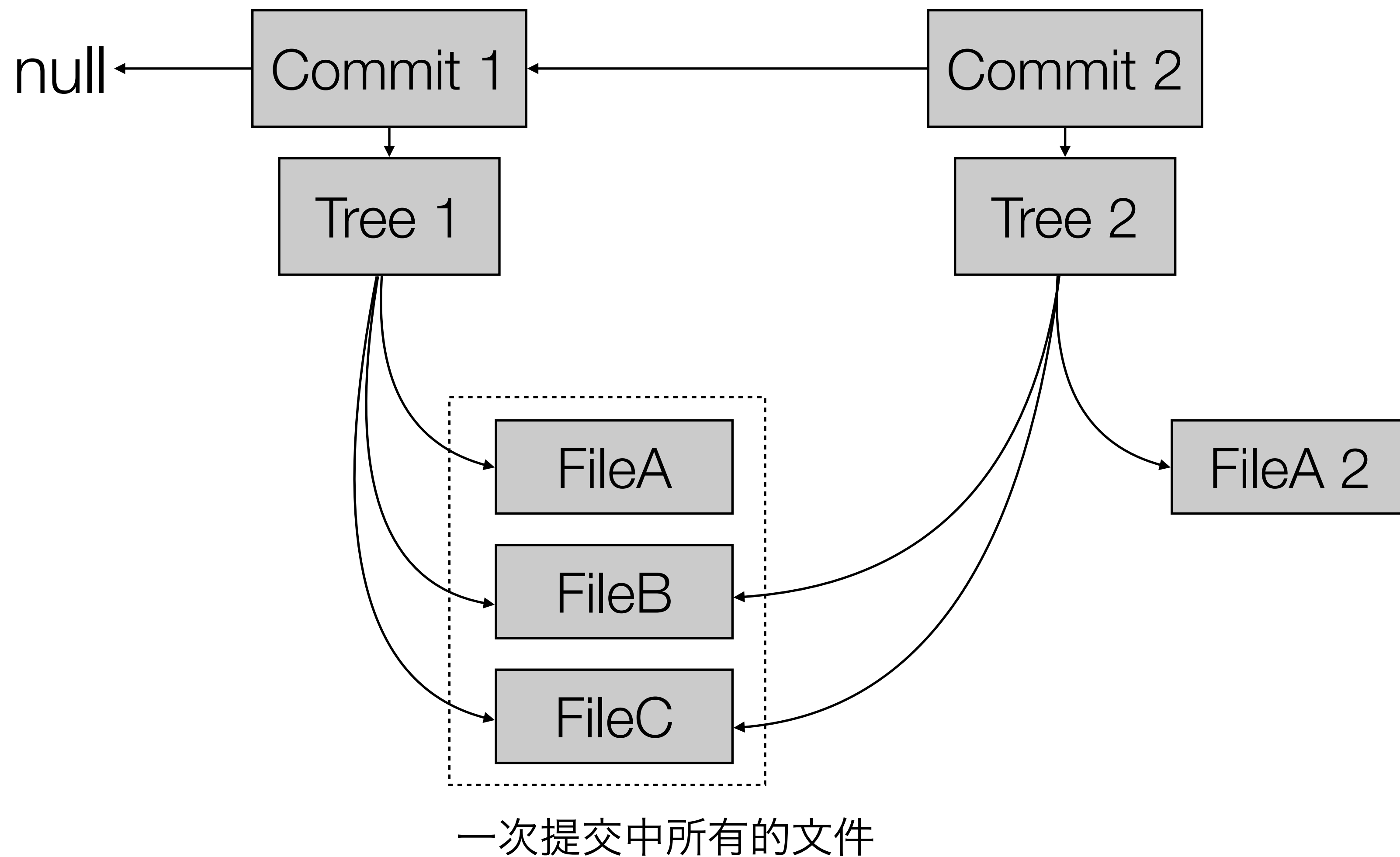
Git 基础知识

Git 就是个文件系统

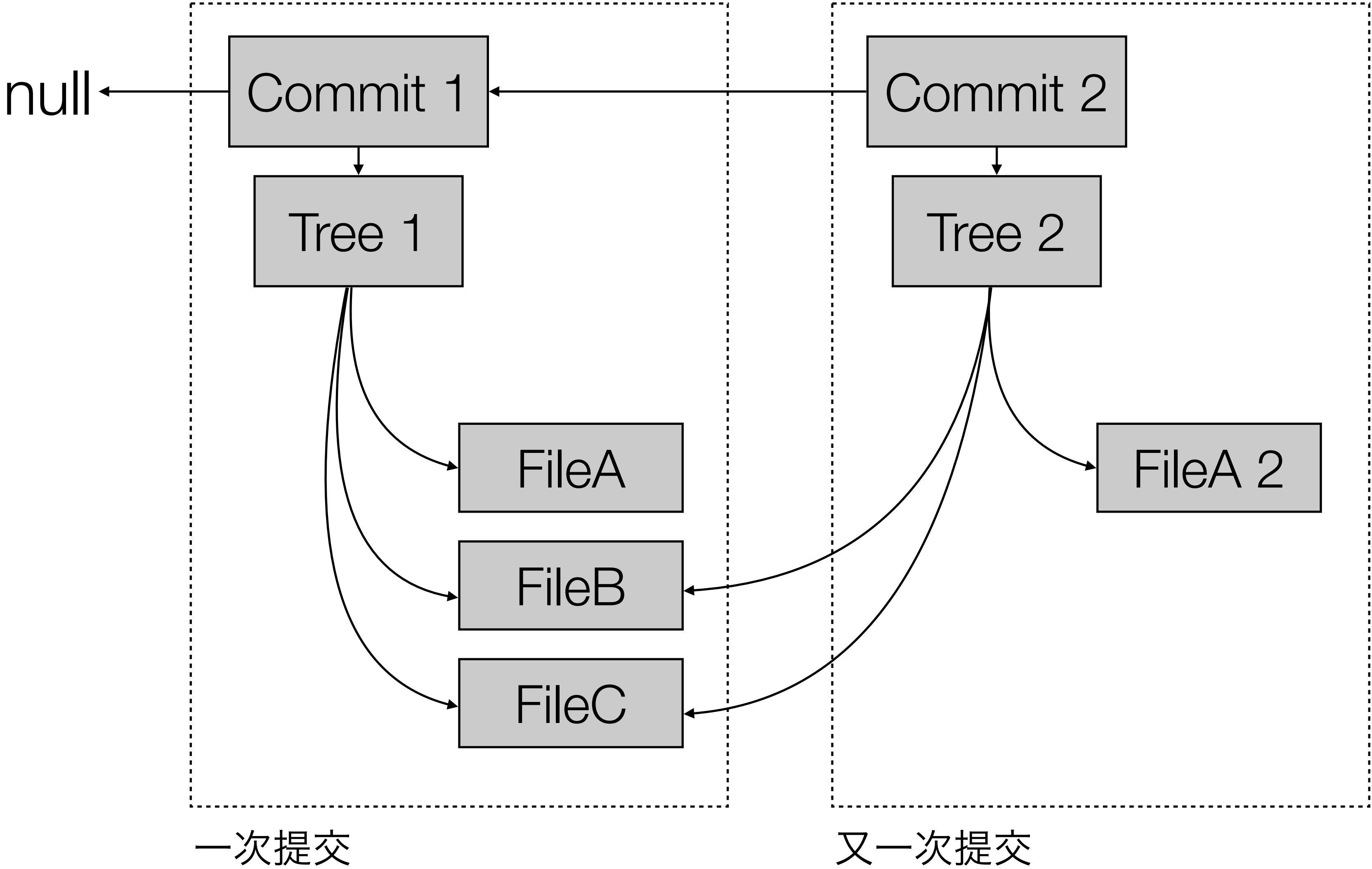
Git 内部对象



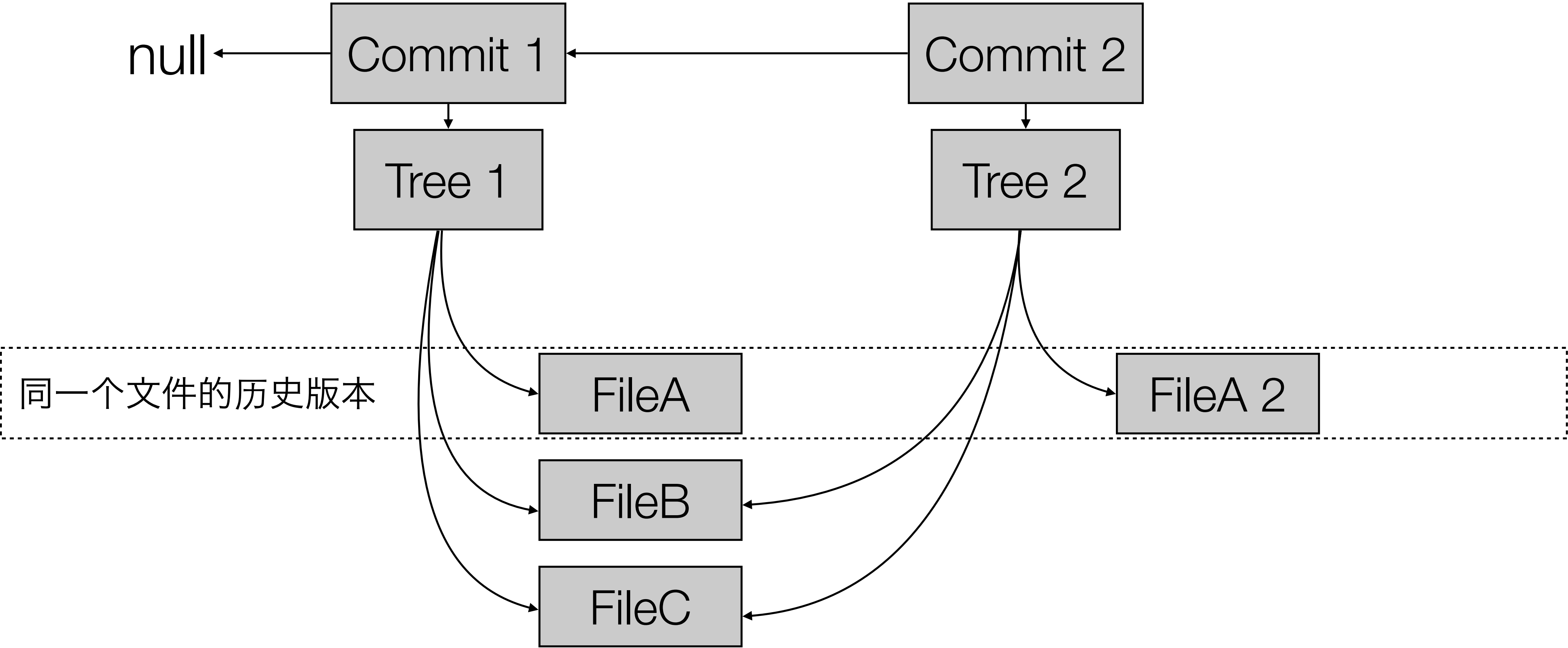
Git 内部对象



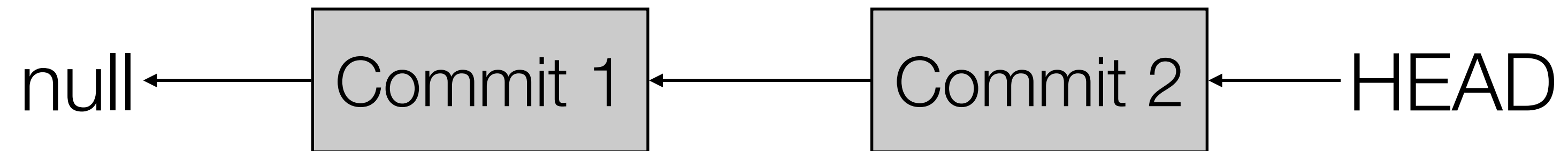
Git 内部对象



Git 内部对象



Git 内部对象



提交对象不可变且拥全局唯一 id

除第一个提交外，每个提交都保存上一个提交的引用

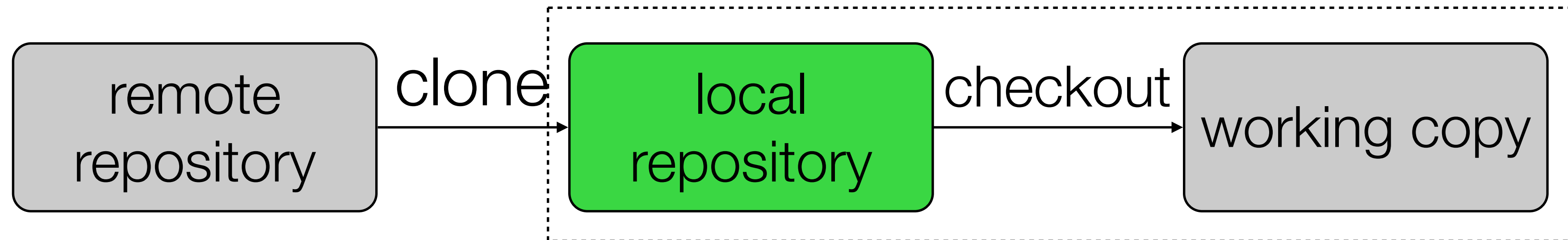
-- Git 里面就是个大链表

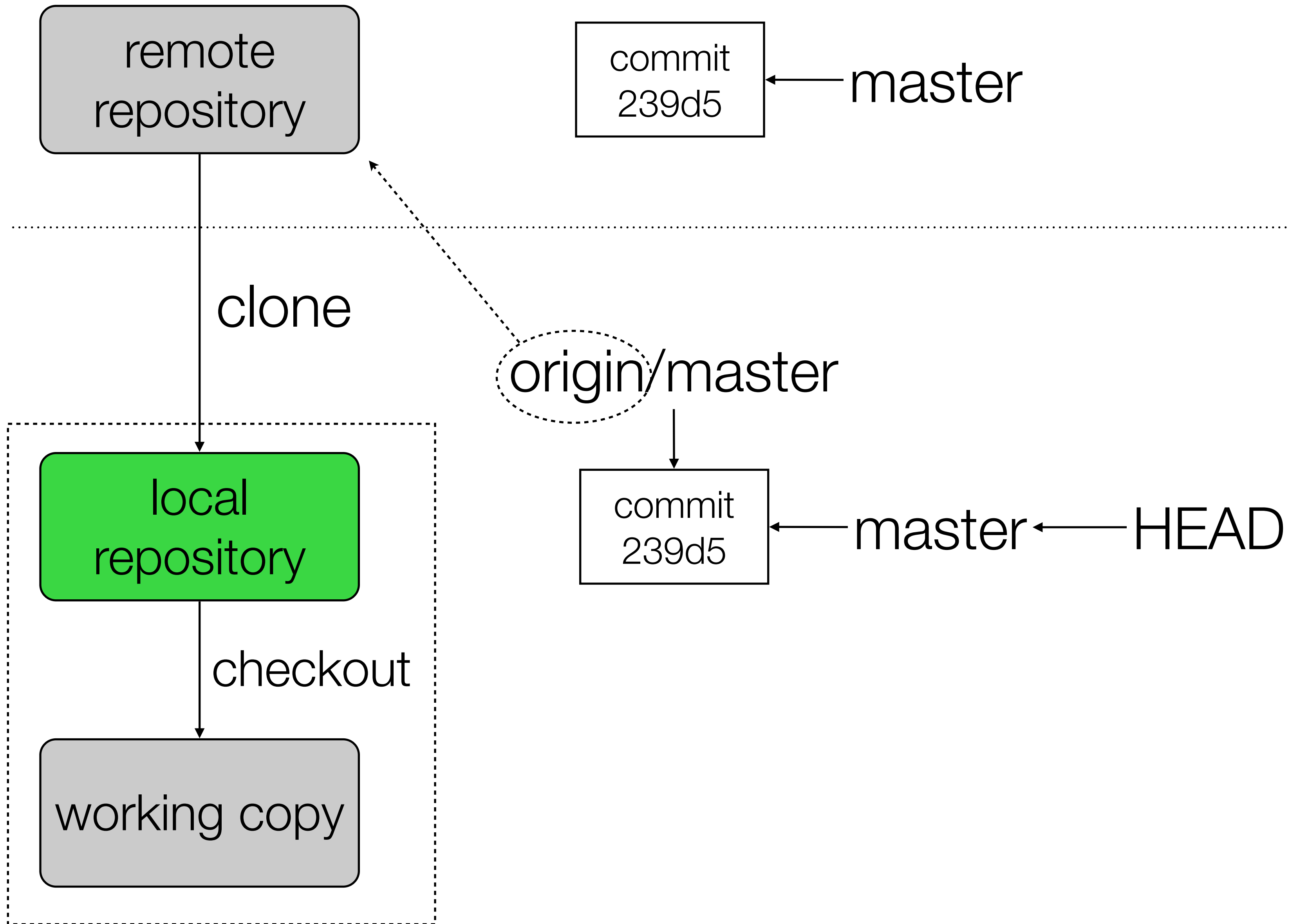
Git 基础命令

不是检出
(checkout)

```
git clone remote_repo_url
```

克隆一份远程仓库成为本地仓库





HEAD: 指向当前工作的 branch

master: 默认的主干分支指针

origin/master: 远程仓库 origin 中 master 分支的指针

origin: 远程仓库名, 在 git clone 时默认创建




```
echo "hello" > world
git add world
```

将文件加入版本库

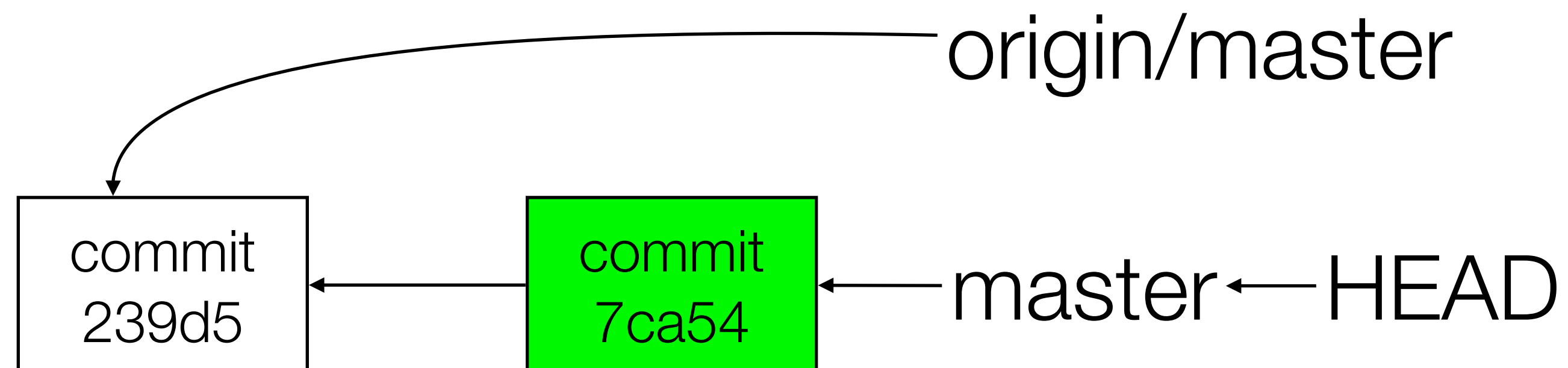
svn add



git commit -m “init commit”

第一个提交

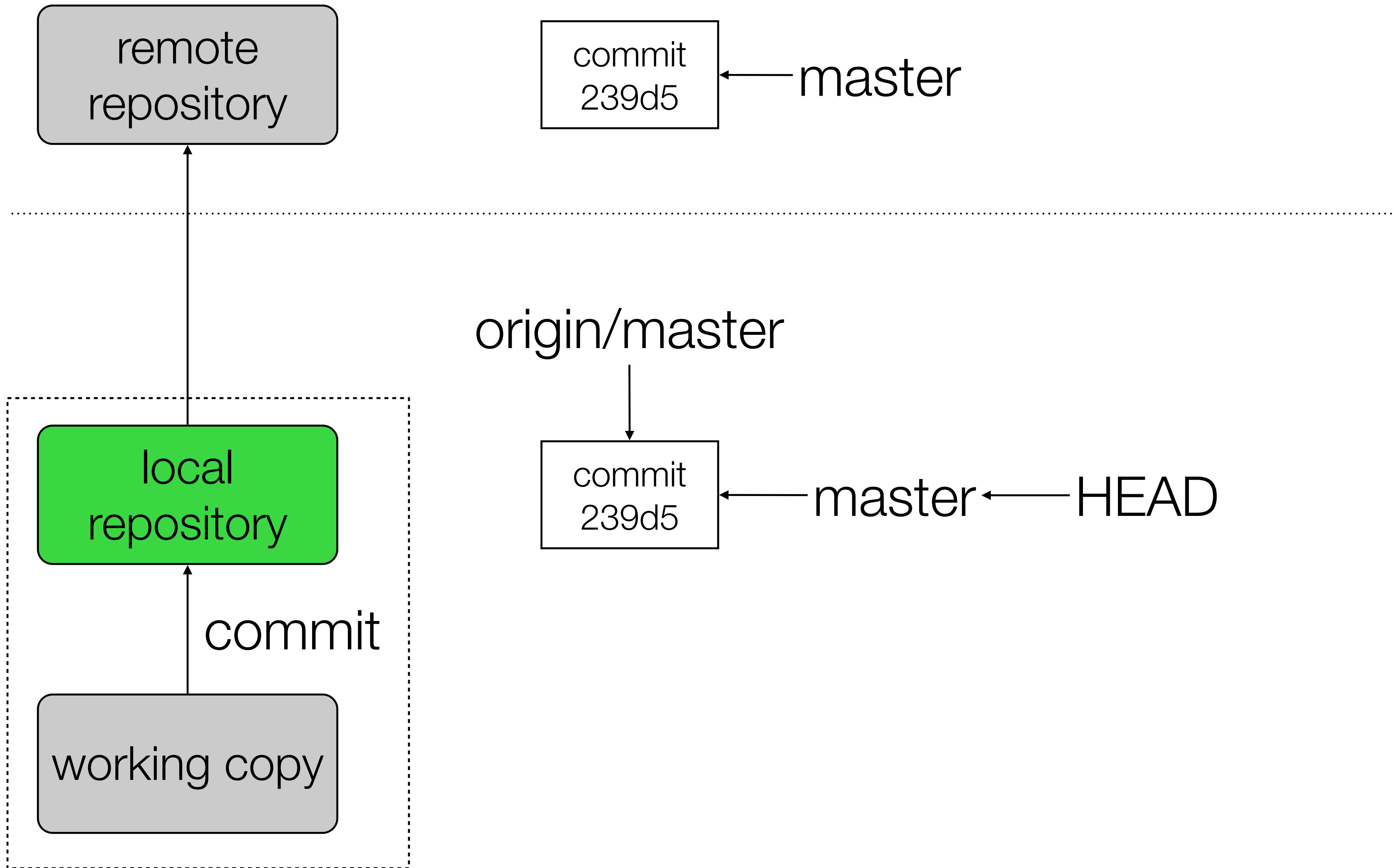
svn commit

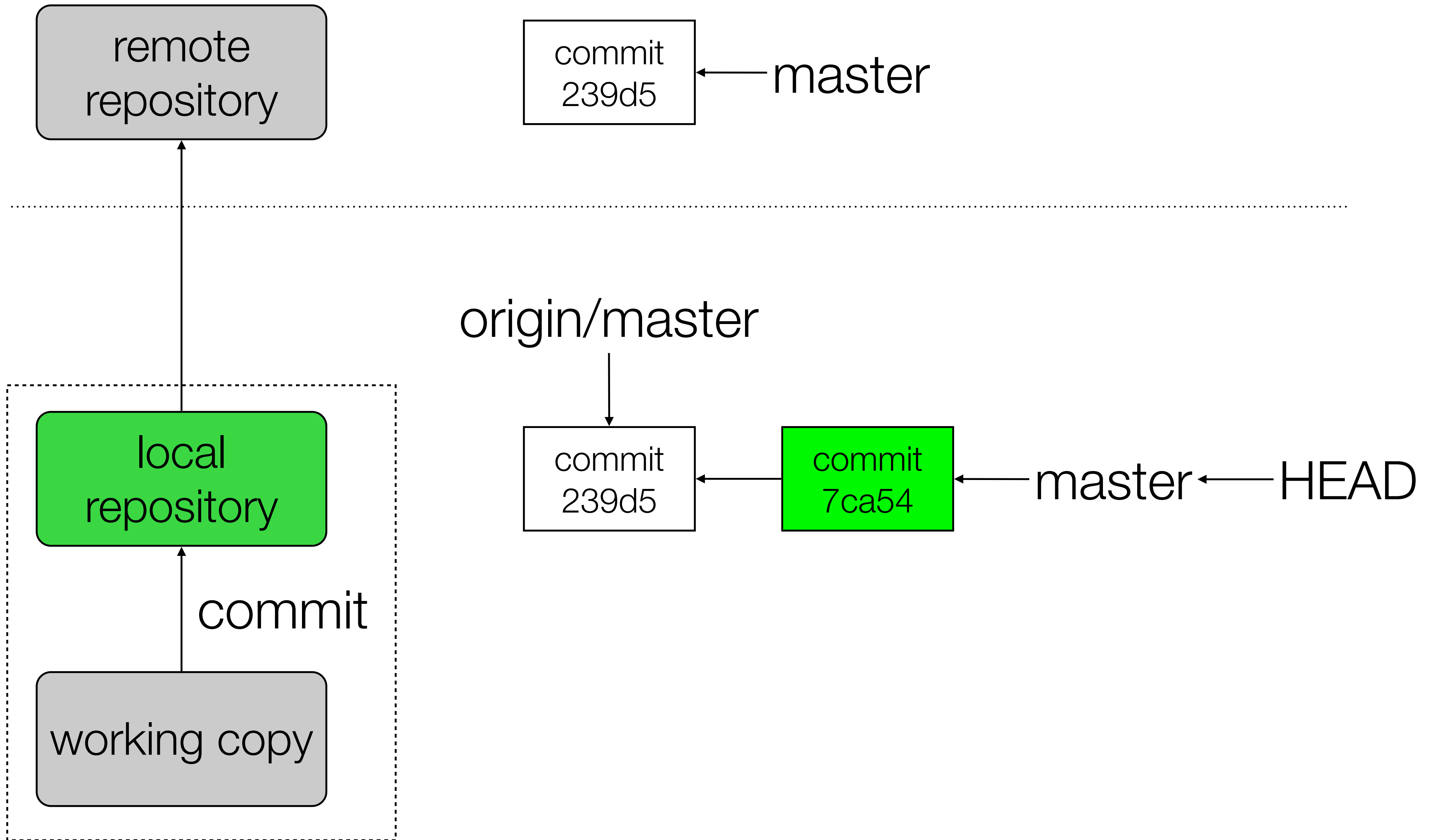


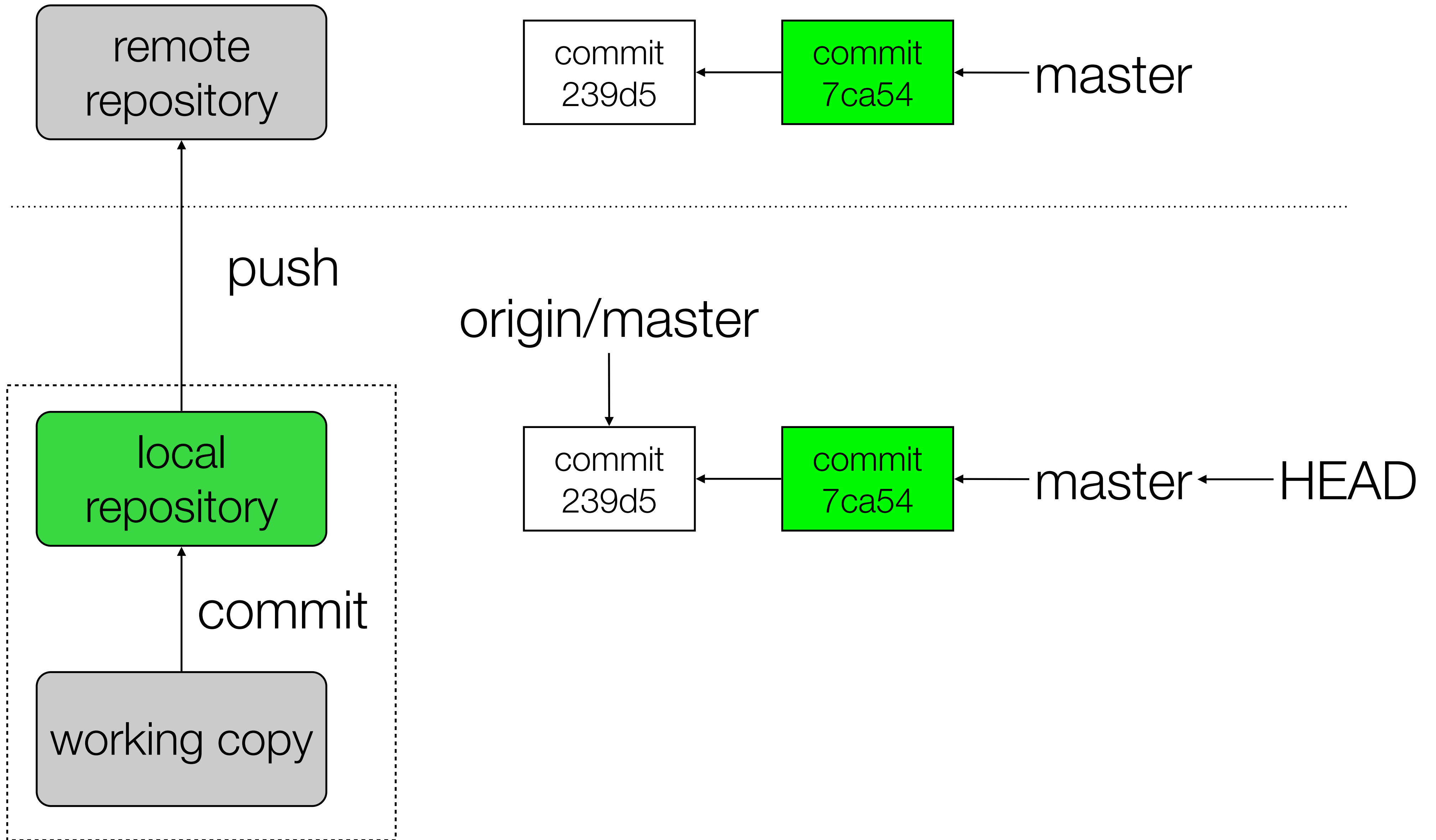
git push origin master

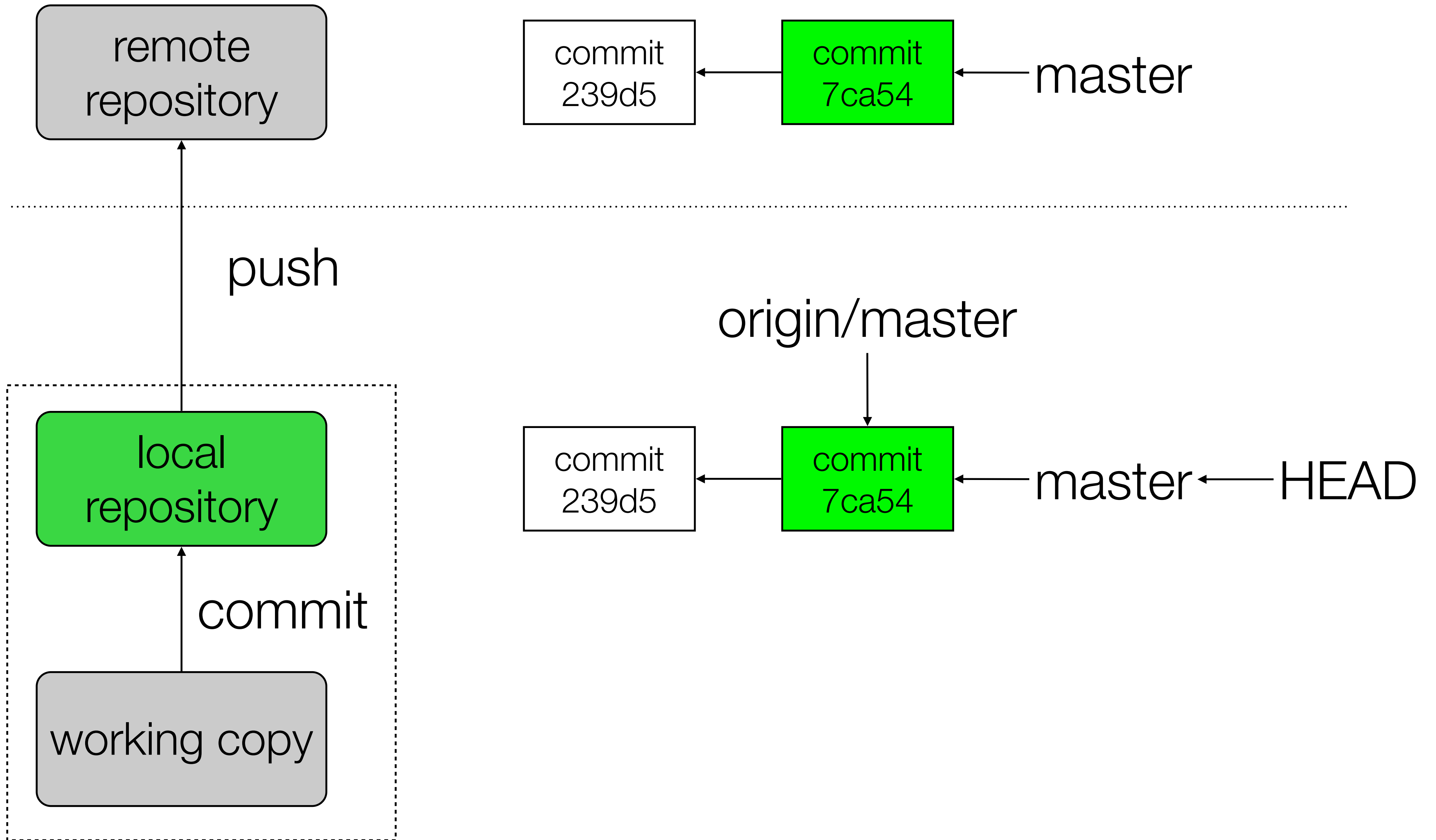
将本地仓库推送到远程仓库上

svn commit





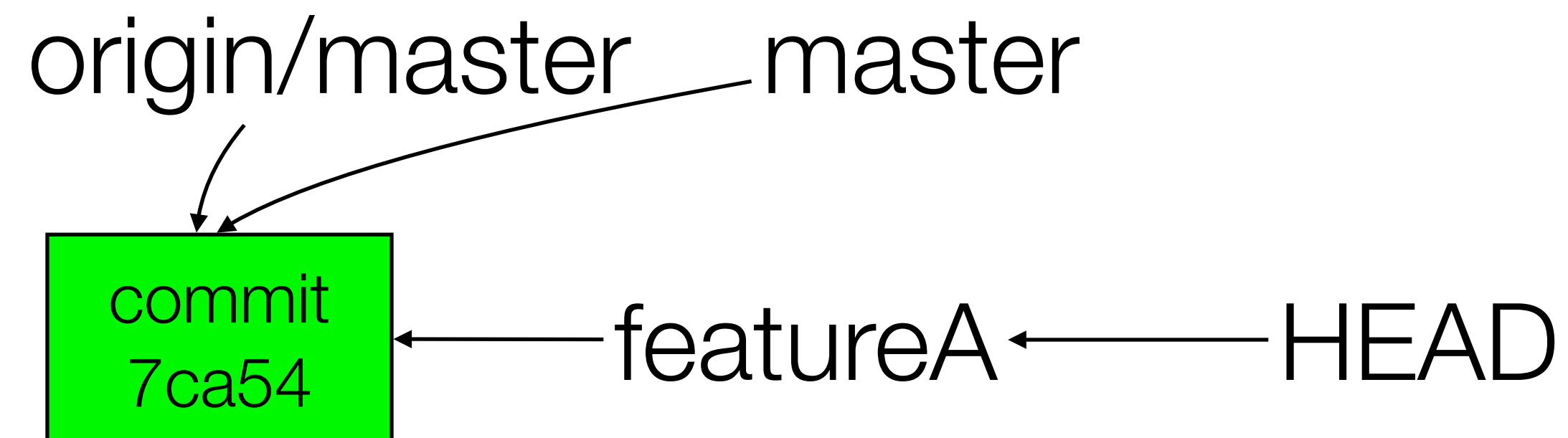




Git 分支

```
git branch featureA  
git checkout featureA
```

在本地仓库创建一个分支

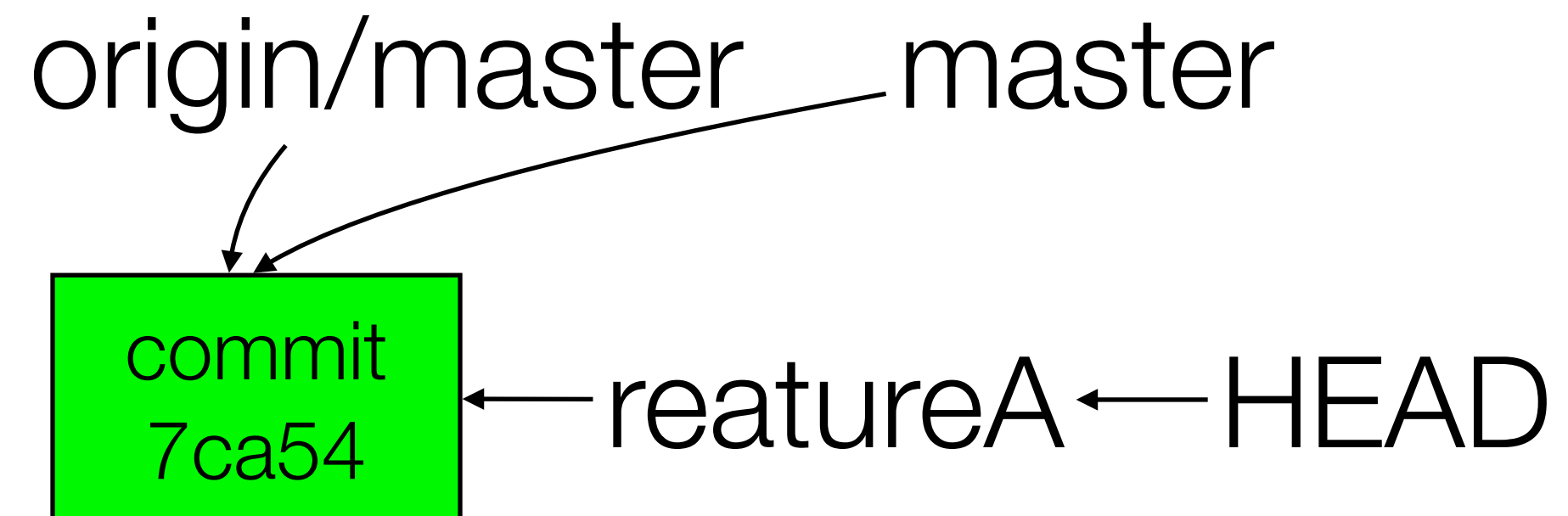


Git 分支

git branch -a

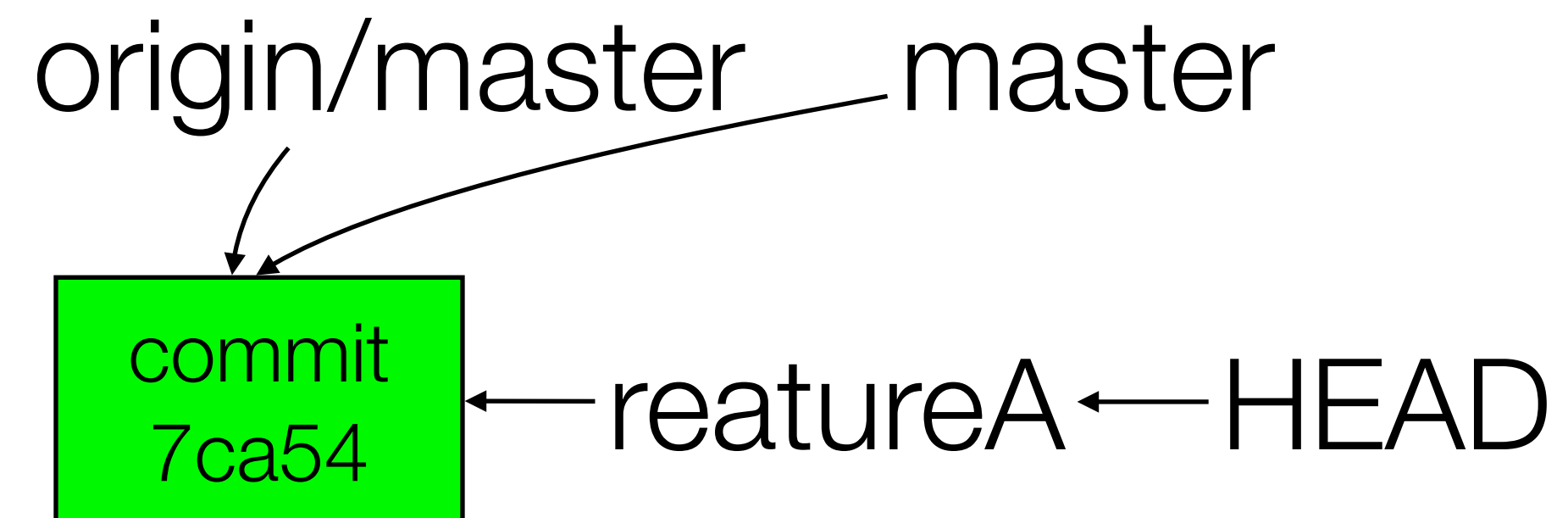
查看所有分支

```
master
* foo
remotes/origin/master
```



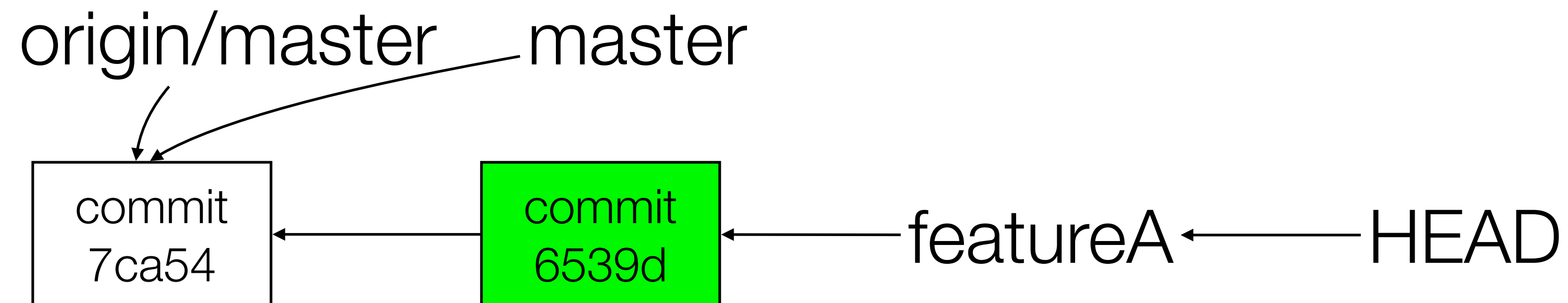
Git 分支

```
echo "r2" >> world; git add world; git commit -m 'r2'
```



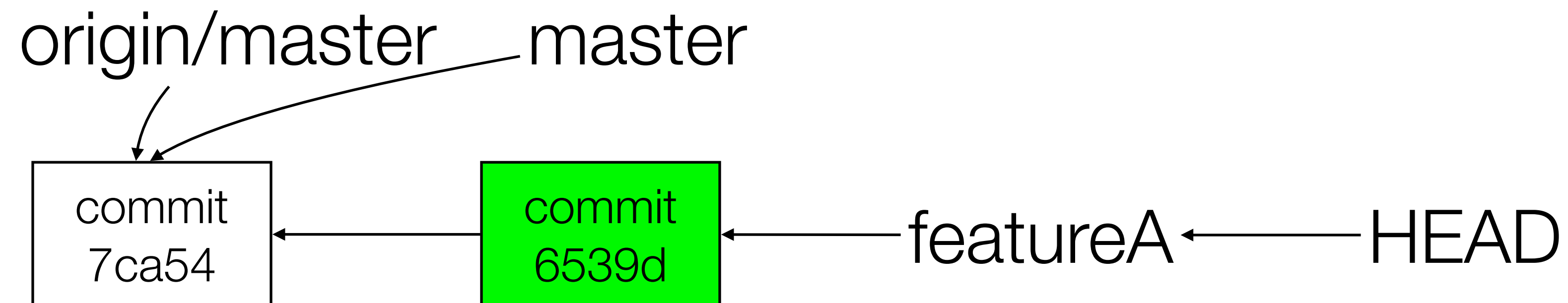
Git 分支

```
echo "r2" >> world; git add world; git commit -m 'r2'
```



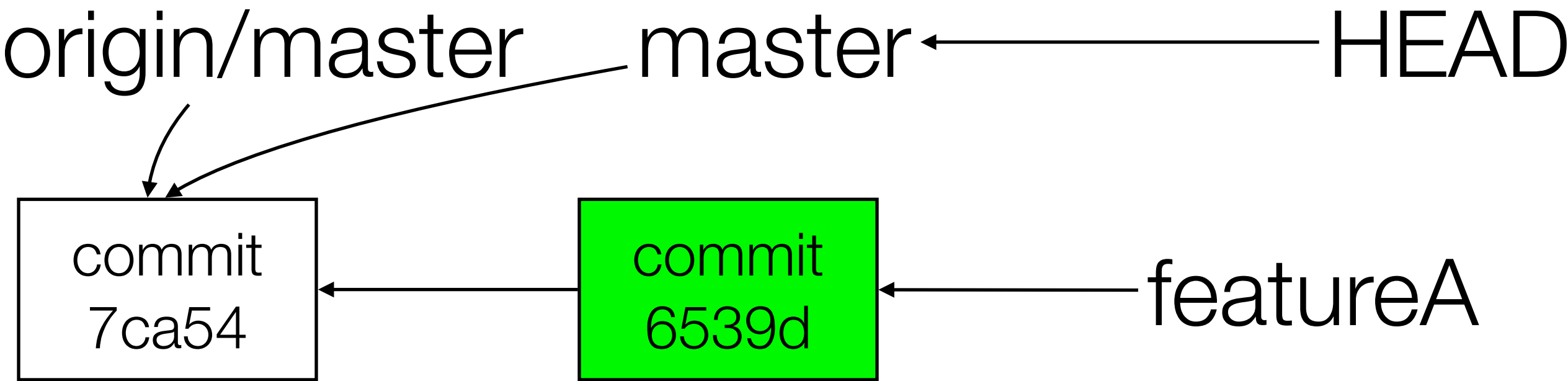
Git 分支切换

git checkout master



Git 分支切换

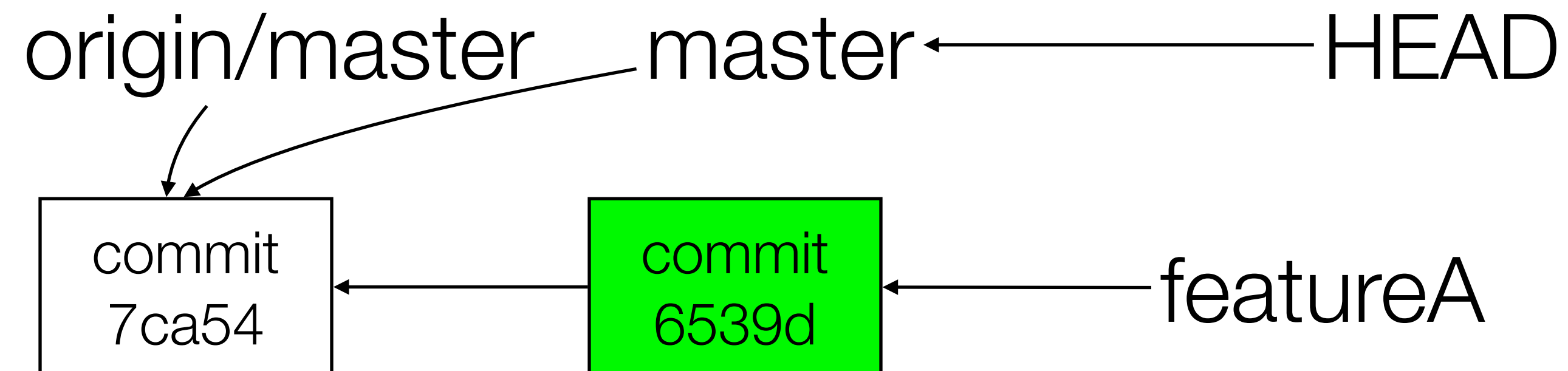
git checkout master



Git 分支切换

git merge master

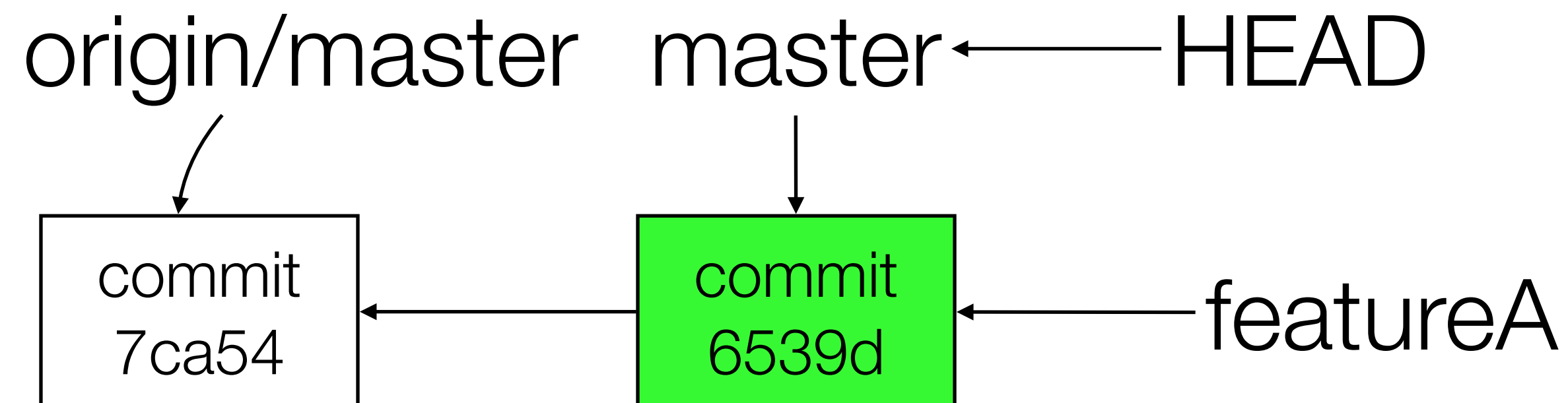
快进 (fast forward) 合并



Git 分支合并

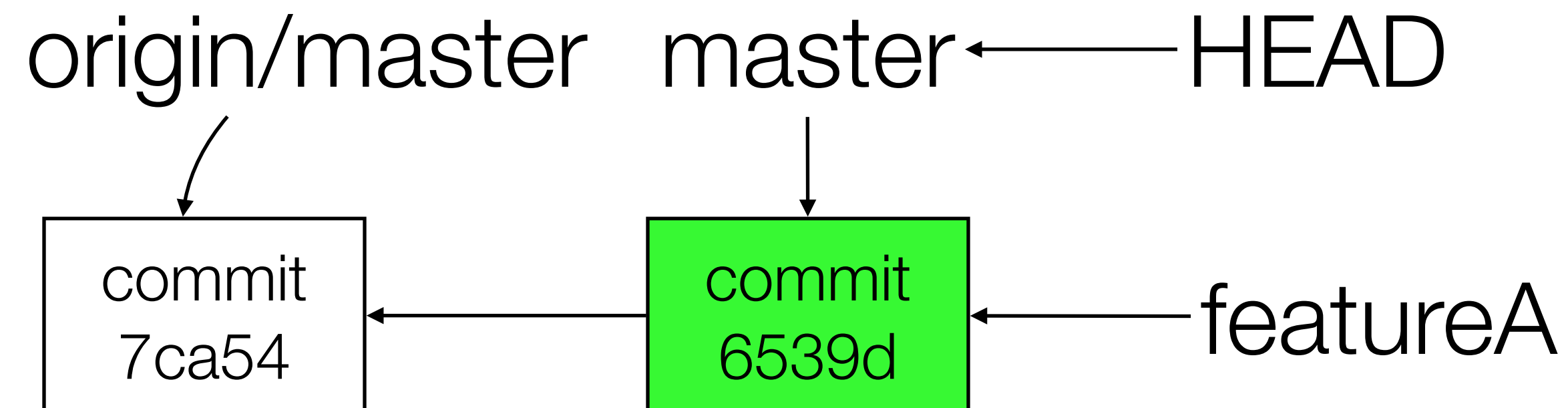
git merge featureA

快进 (fast forward) 合并



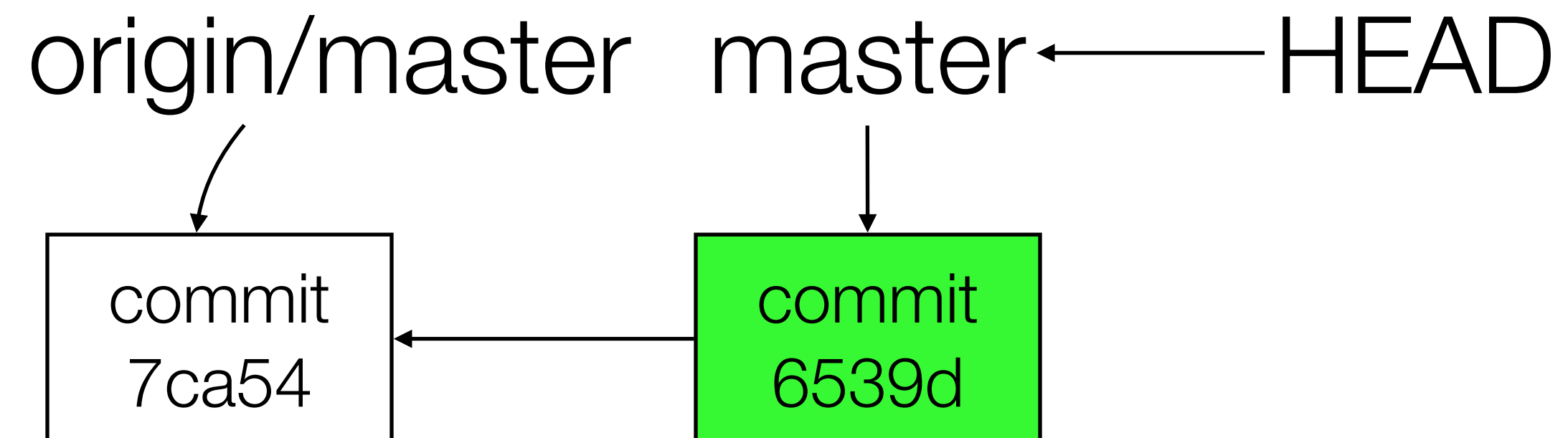
Git 分支合并

```
git branch -d featureA
```

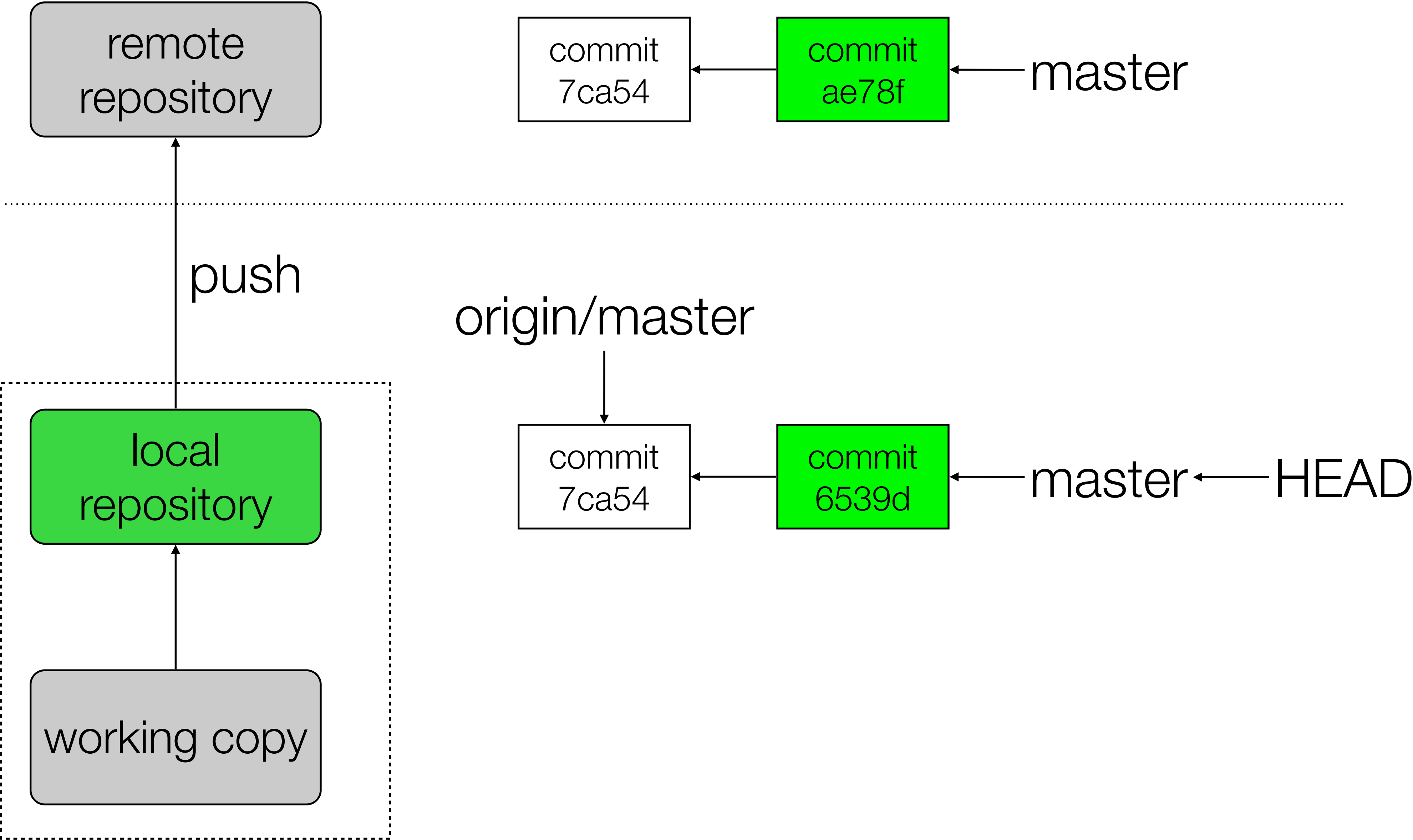


Git 分支合并

`git branch -d featureA`



Git 冲突解决

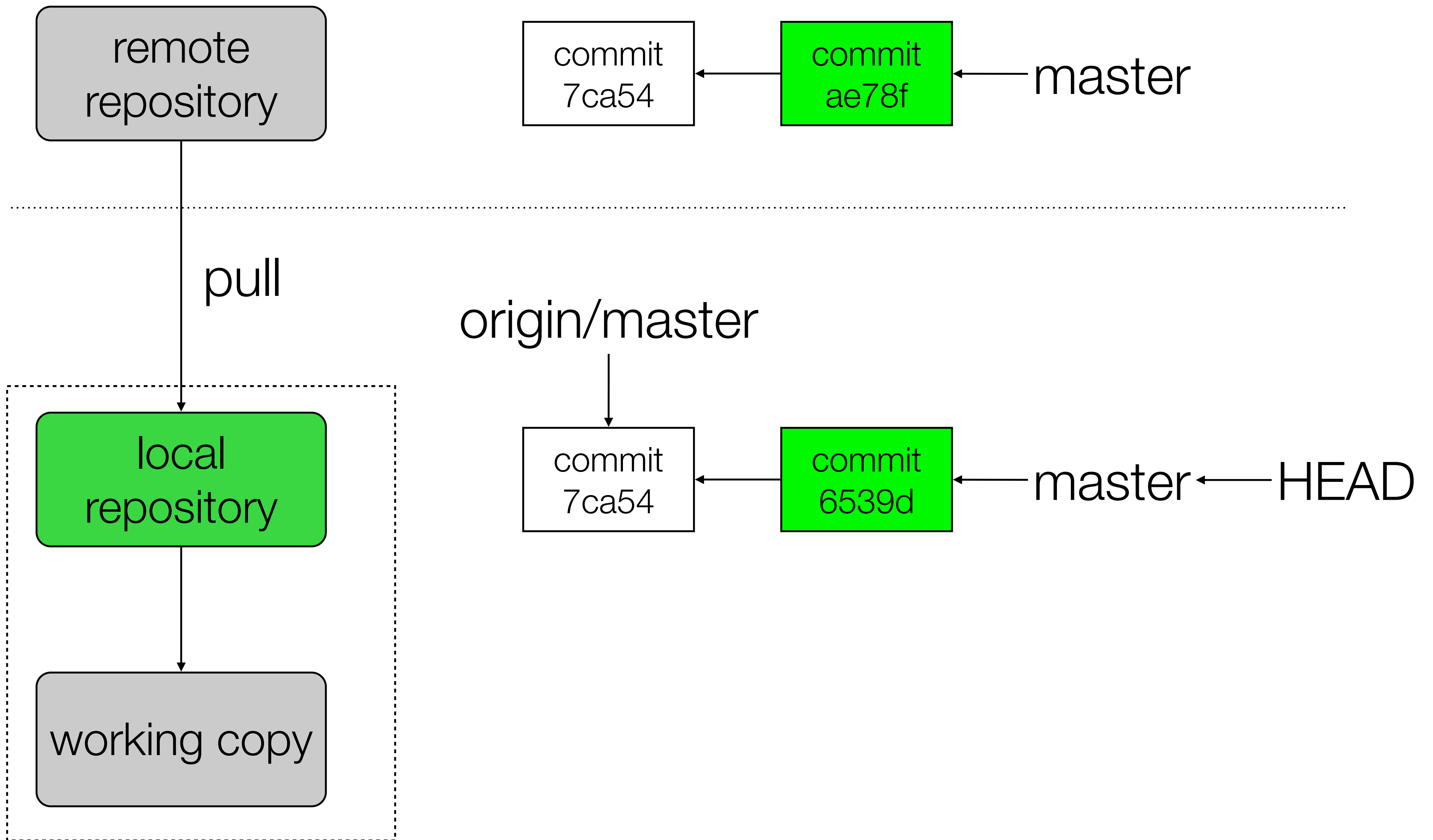


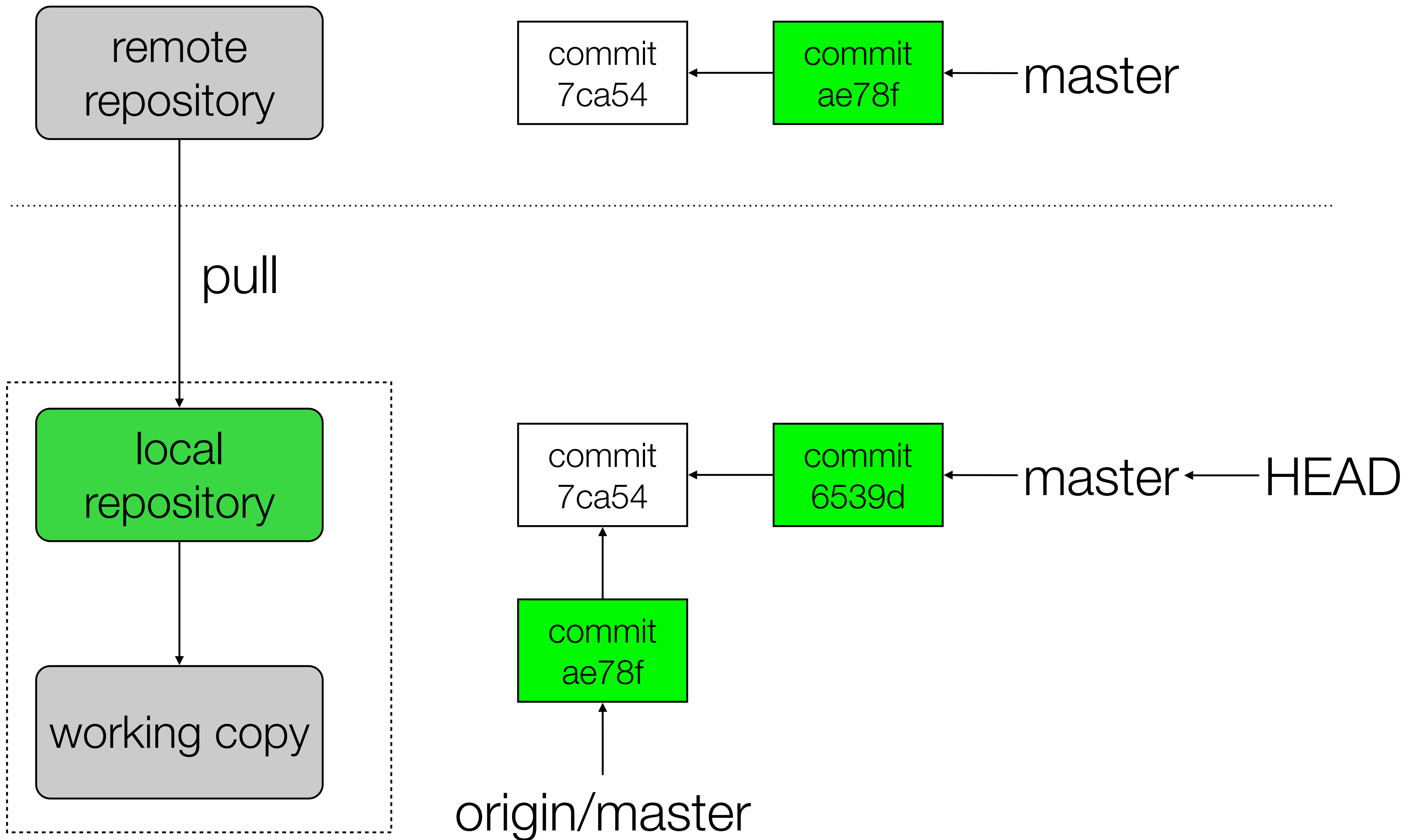
Git 冲突解决

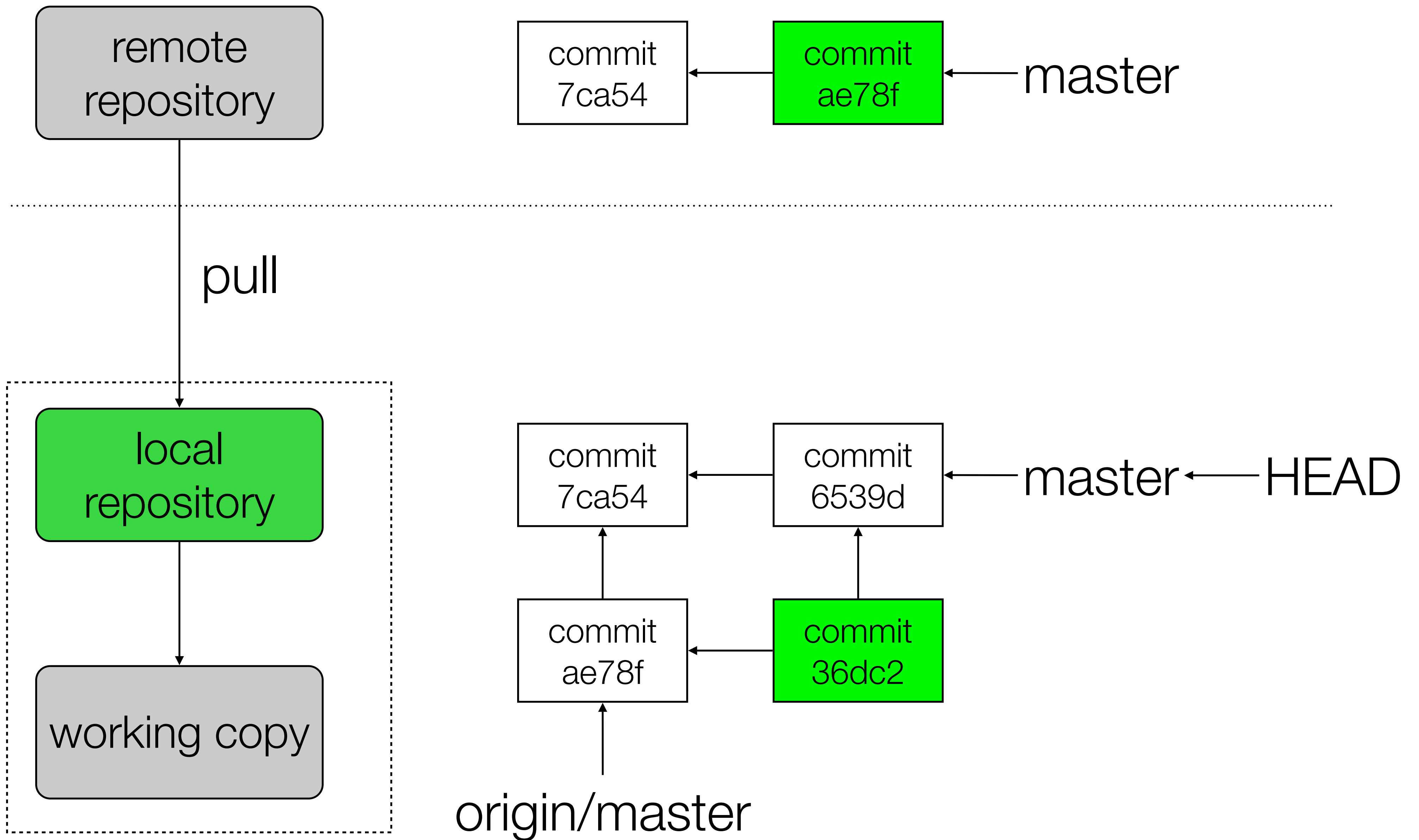
git push

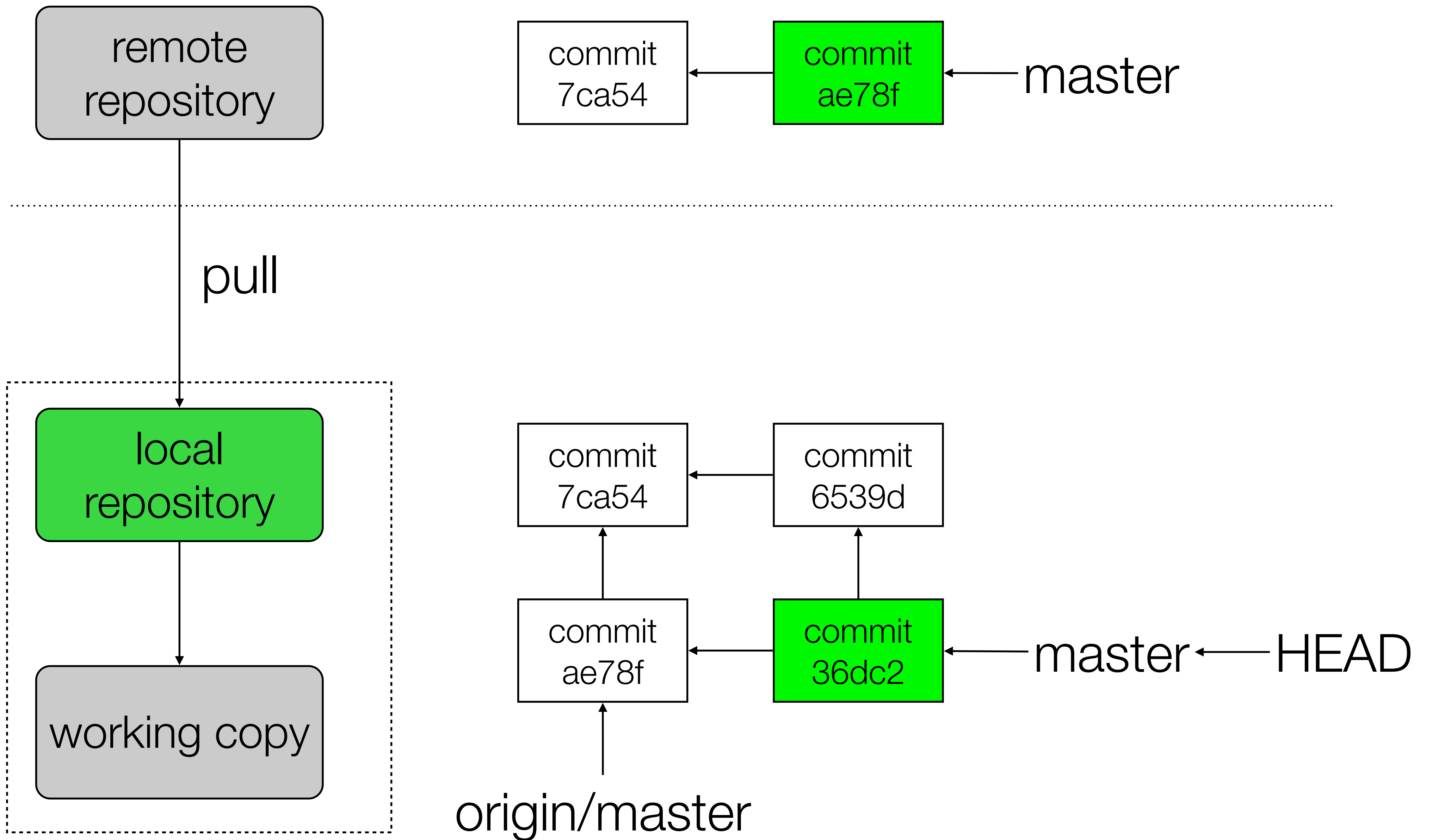
```
To origin
! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'origin'
To prevent you from losing history, non-fast-forward updates were rejected
Merge the remote changes (e.g. 'git pull') before pushing again. See the
'Note about fast-forwards' section of 'git push --help' for details.
```

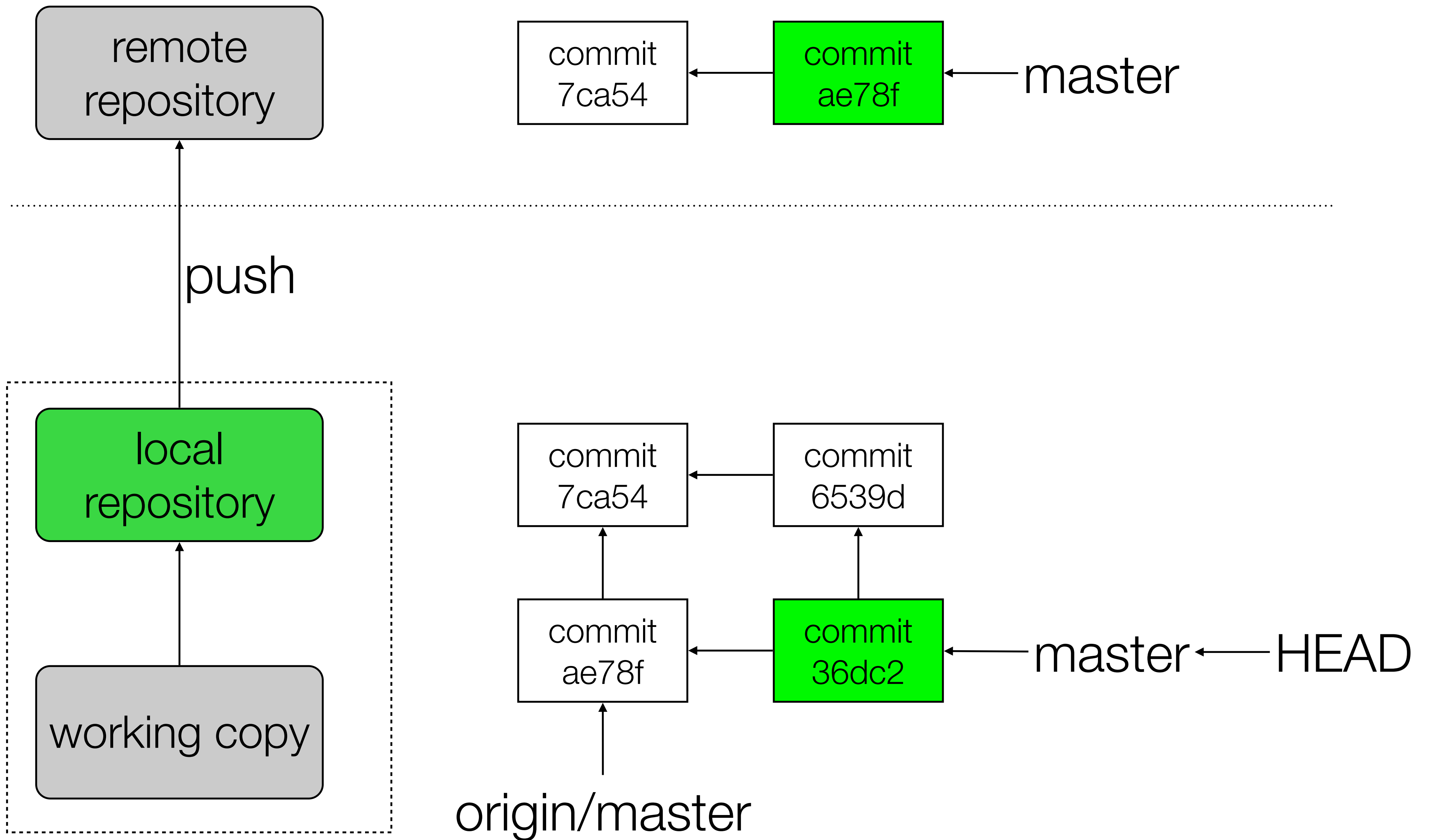
类似 svn 的 out of date 提示

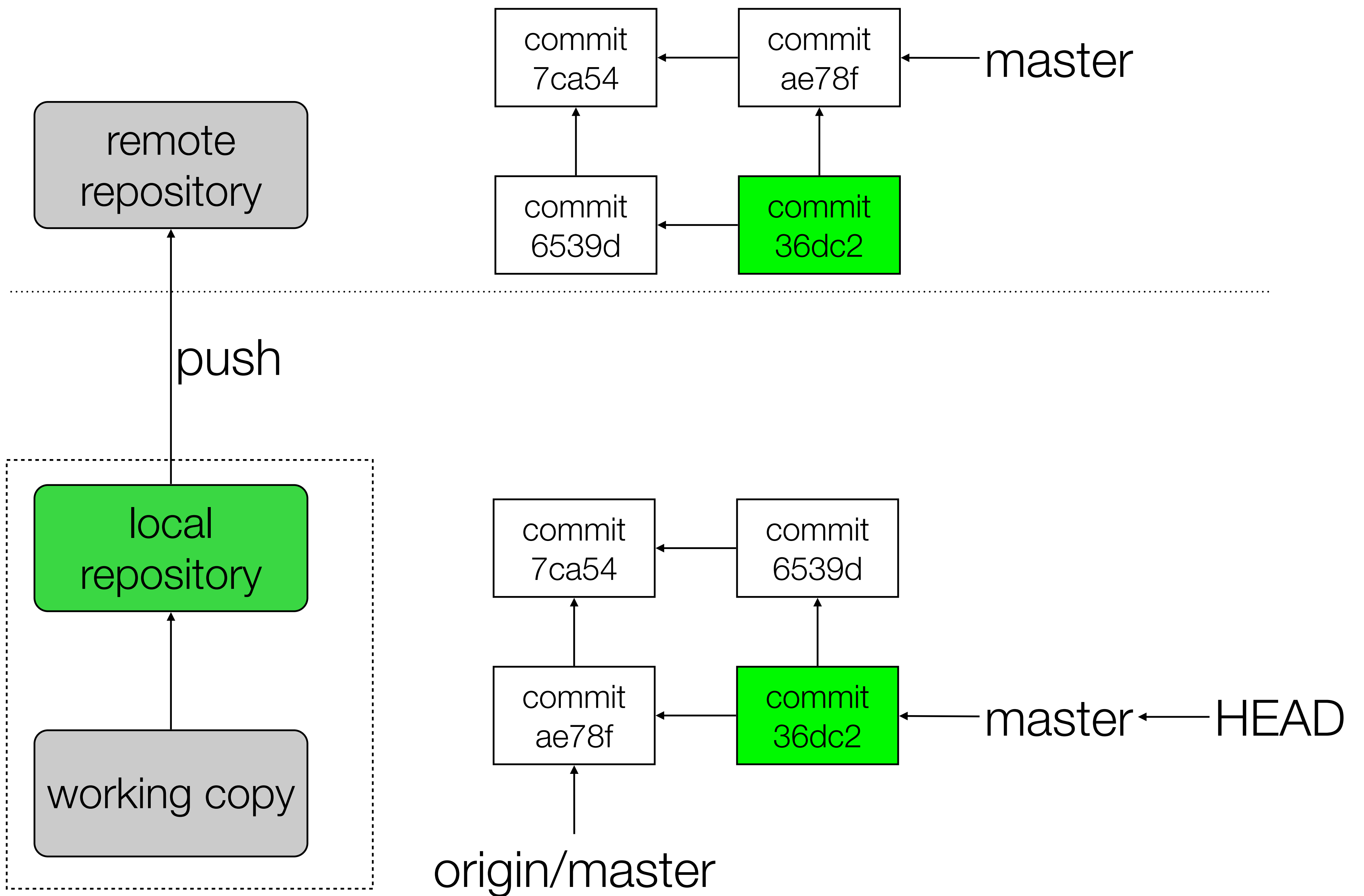


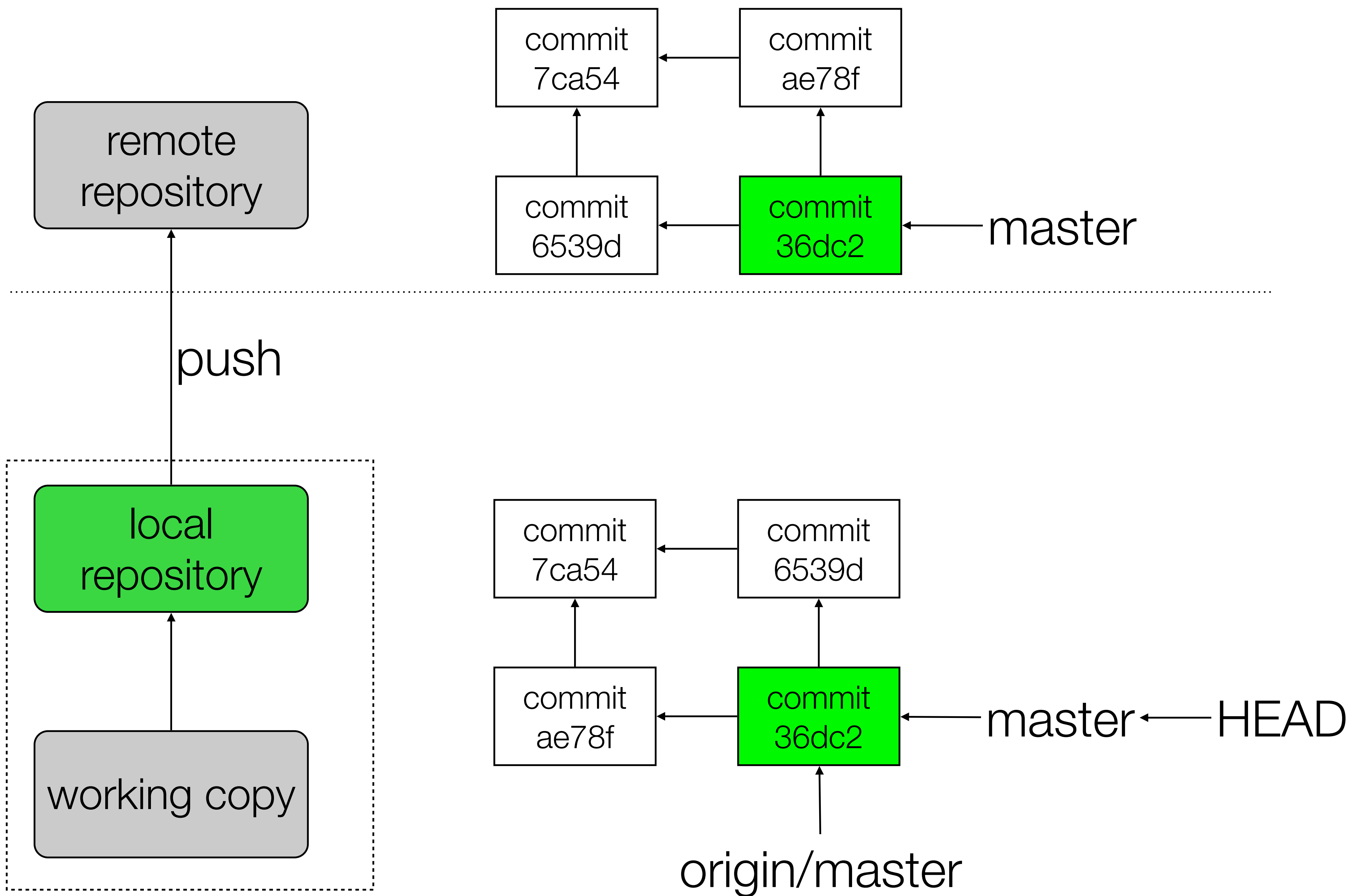








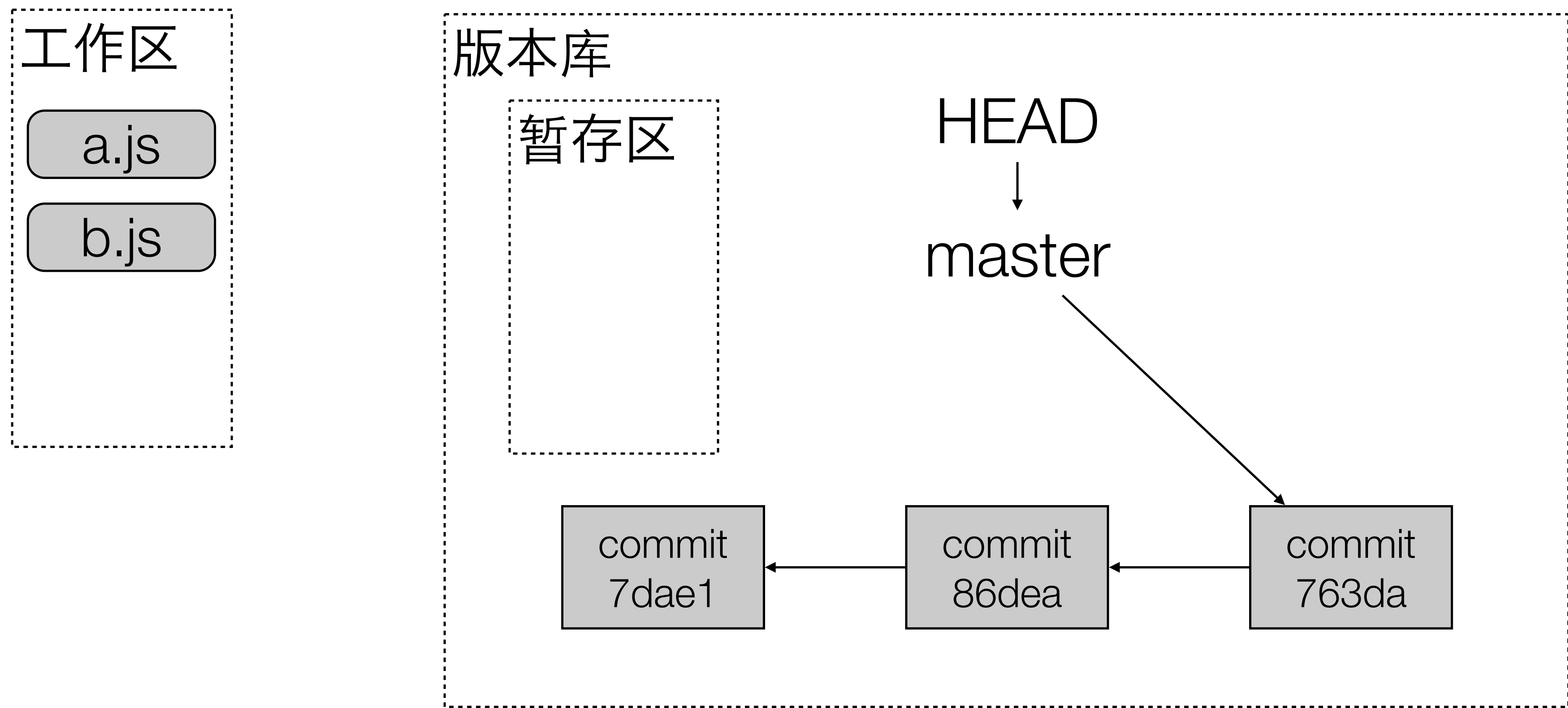




暂存区与指针控制

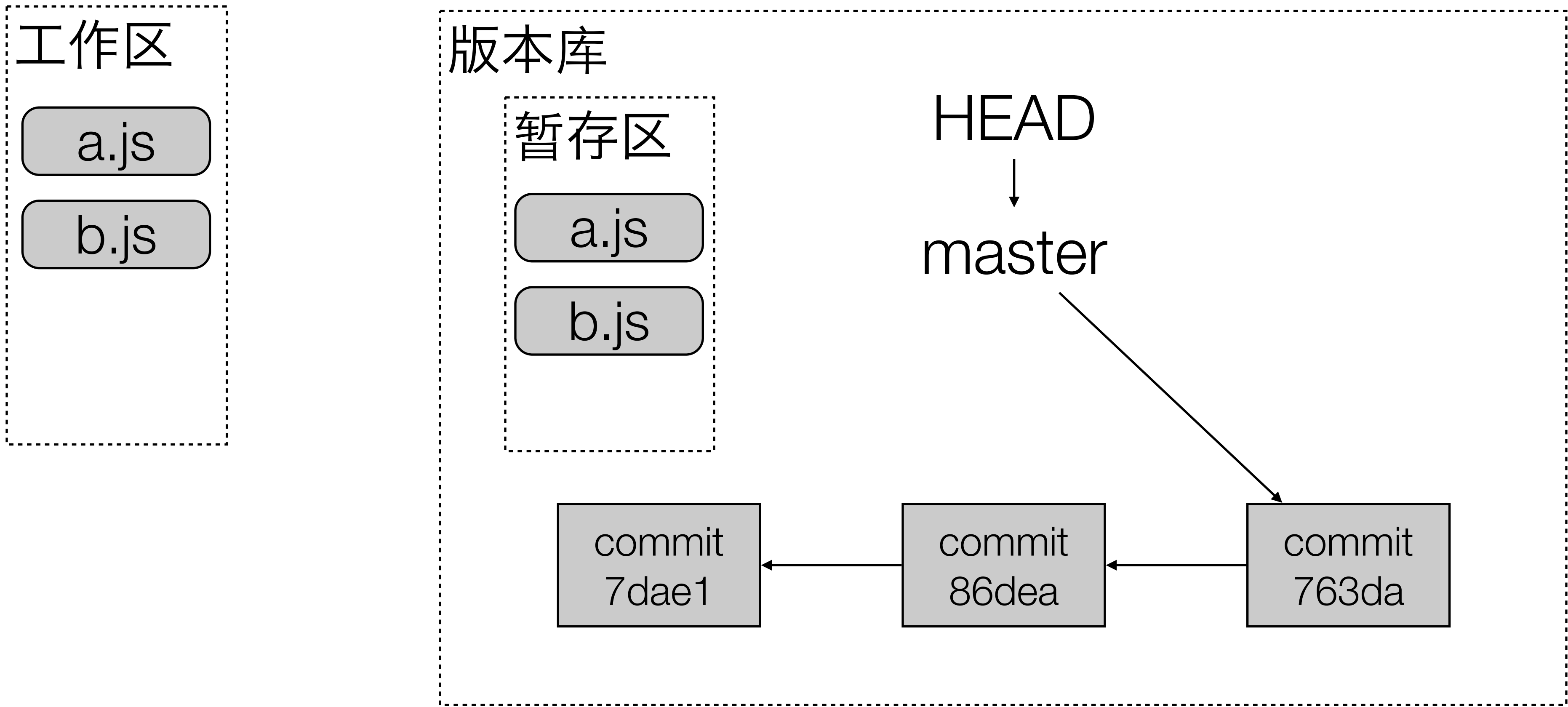
git add
git checkout
git reset

Git 暂存区



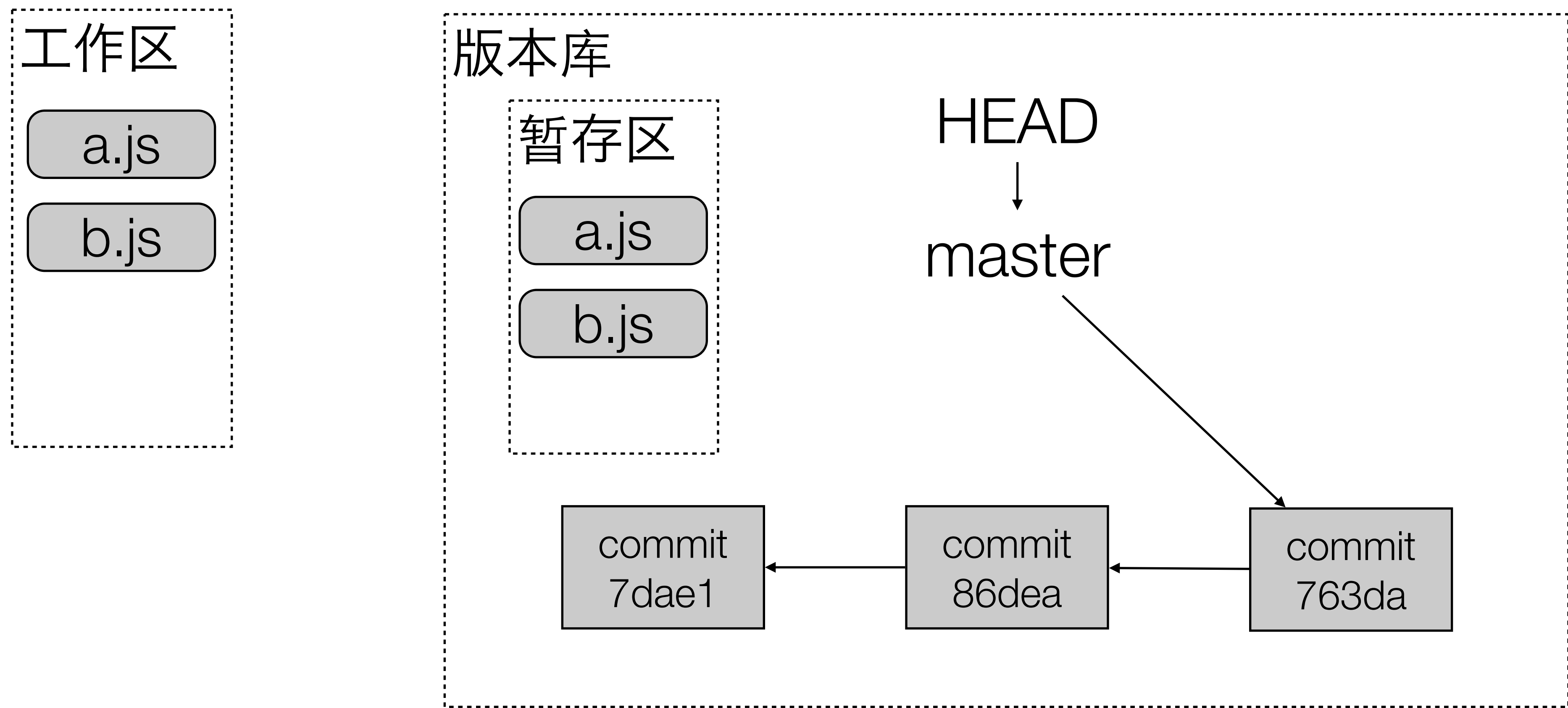
Git 暂存区

```
git add a.js b.js
```



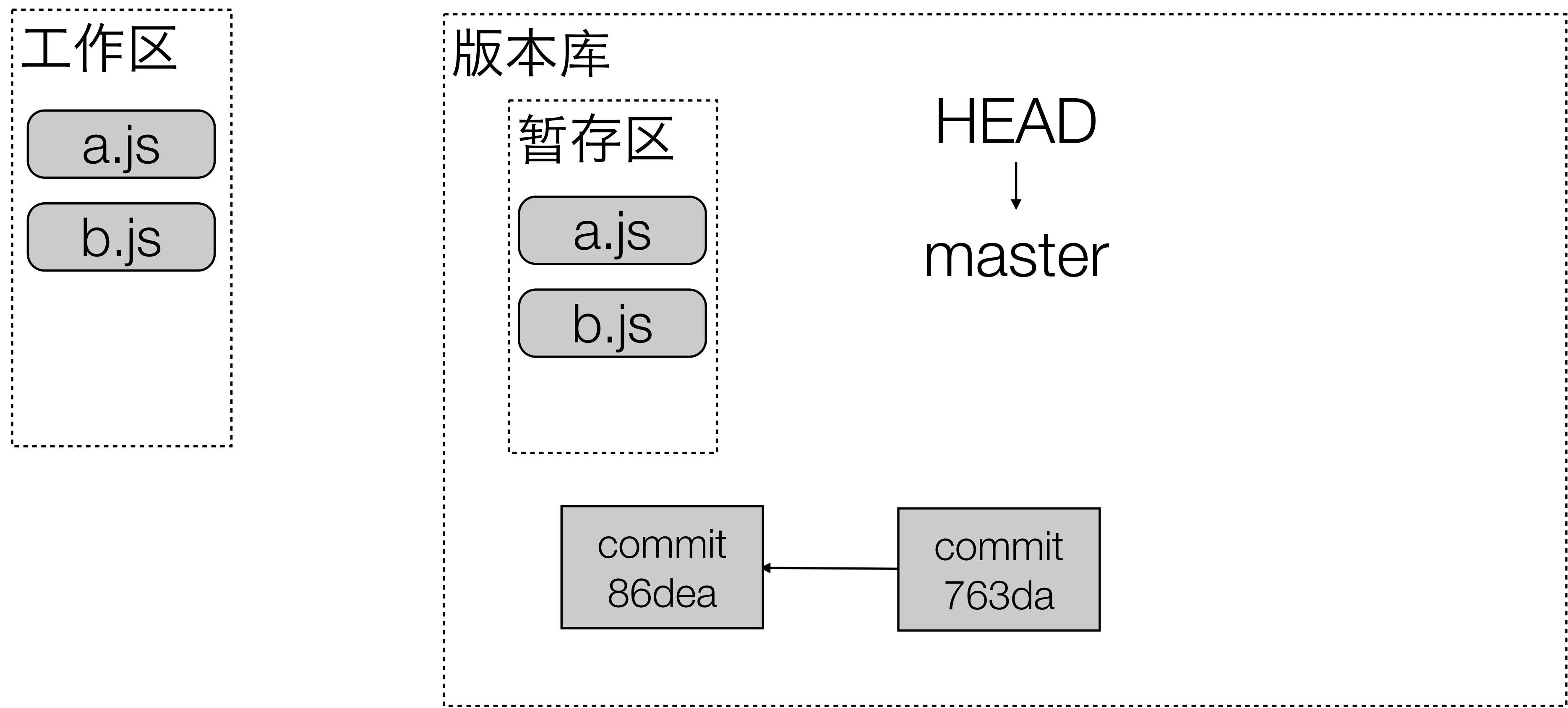
Git 暂存区

git commit



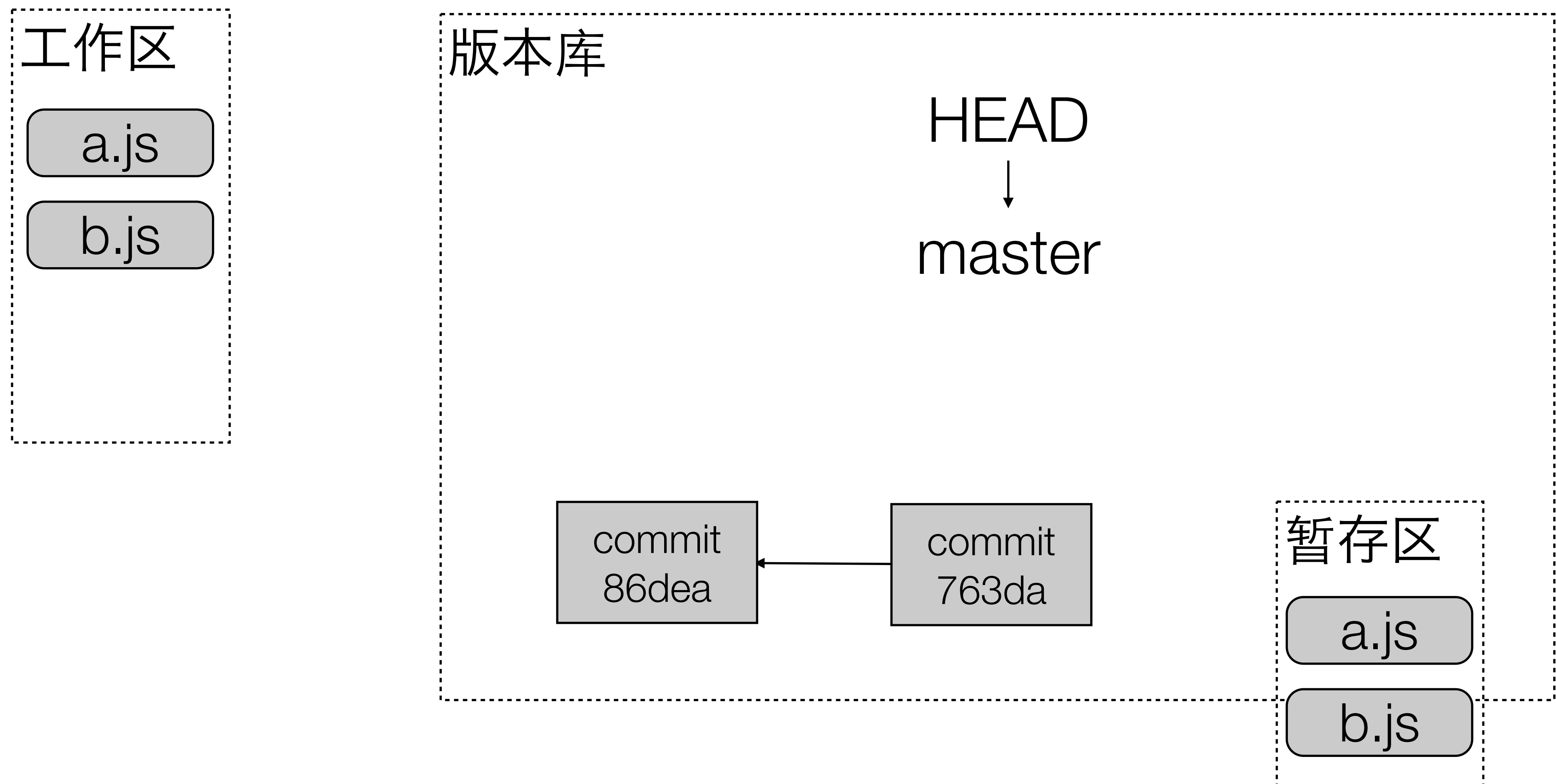
Git 暂存区

git commit



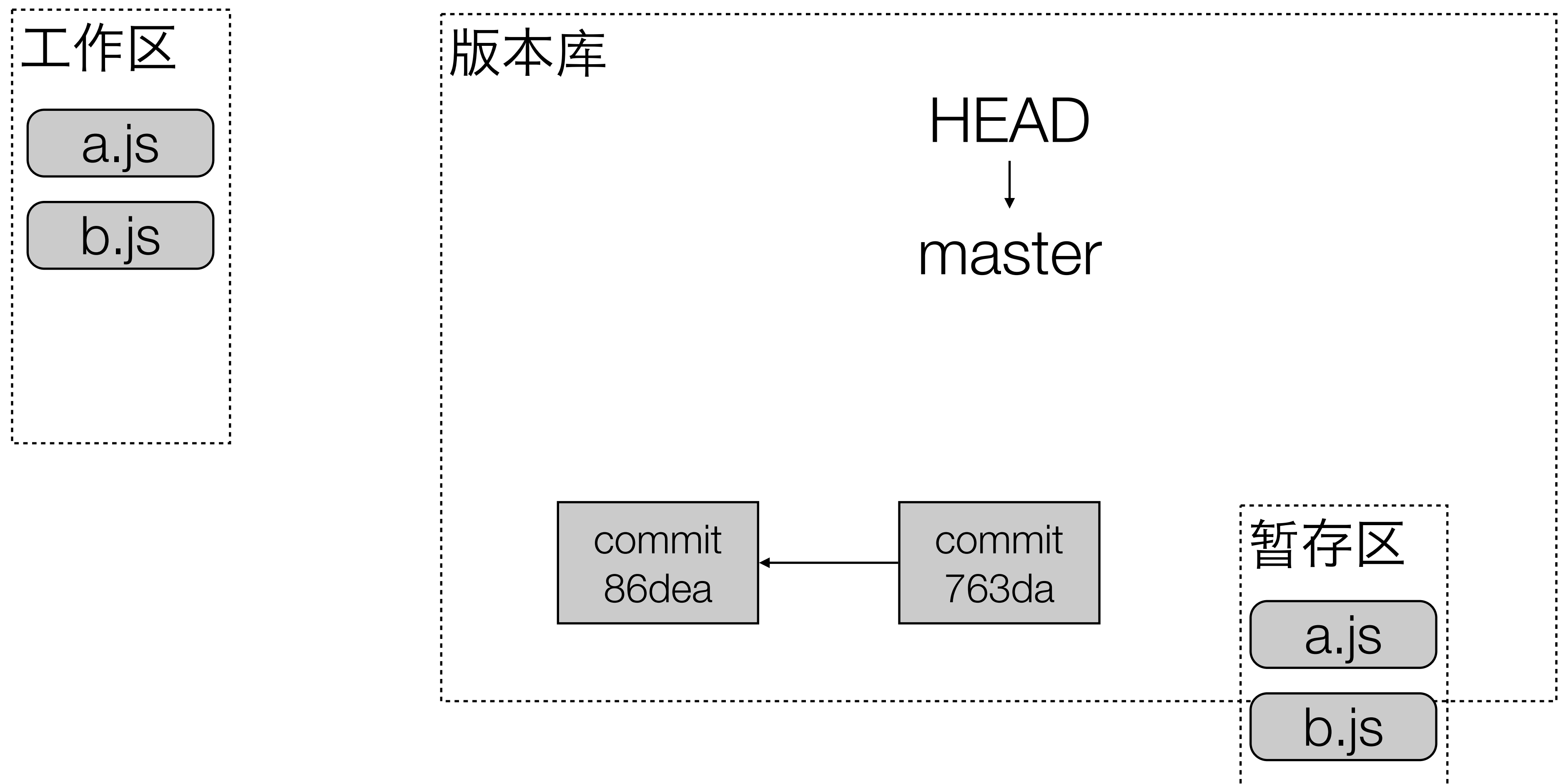
Git 暂存区

git commit



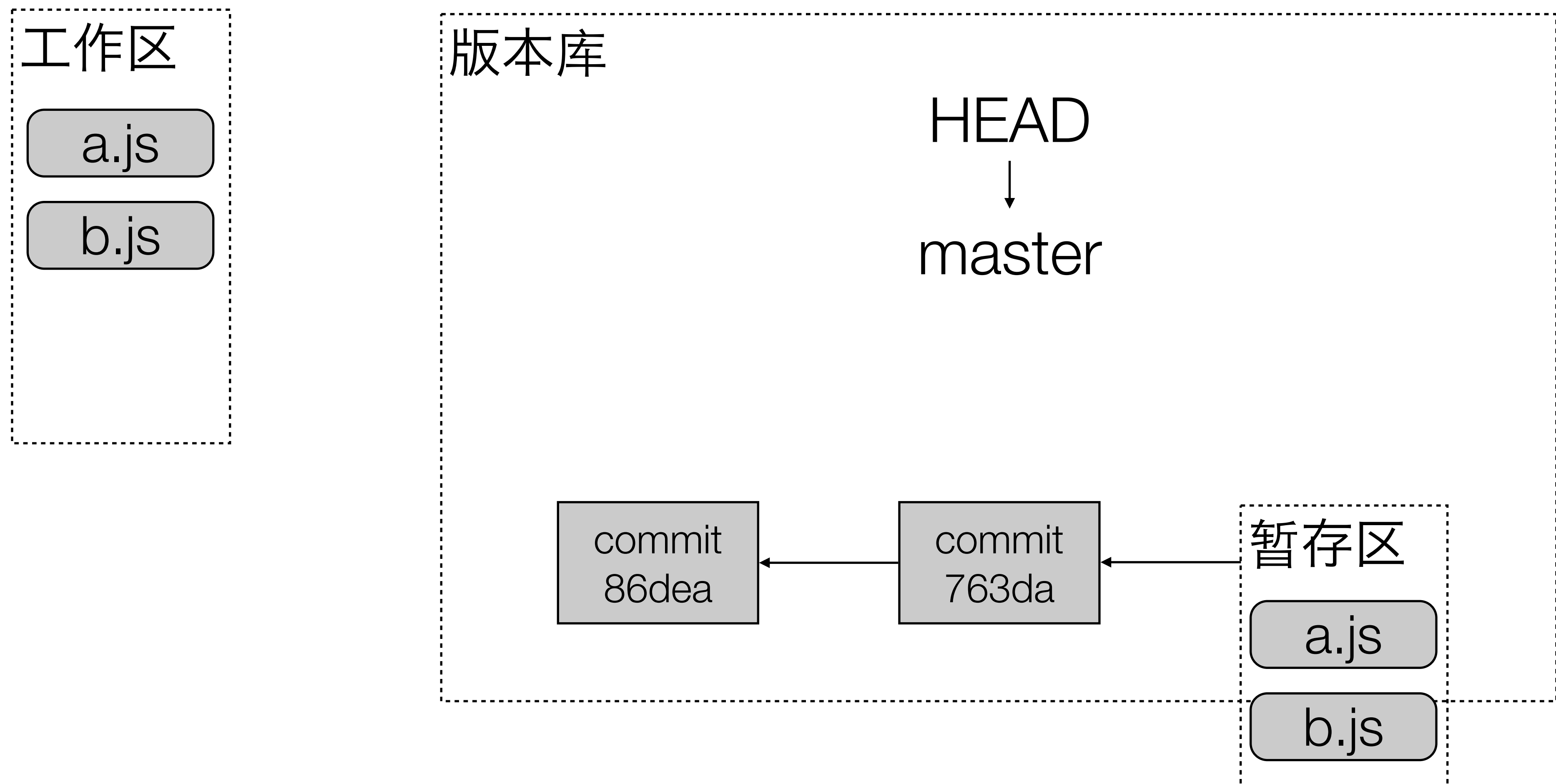
Git 暂存区

git commit



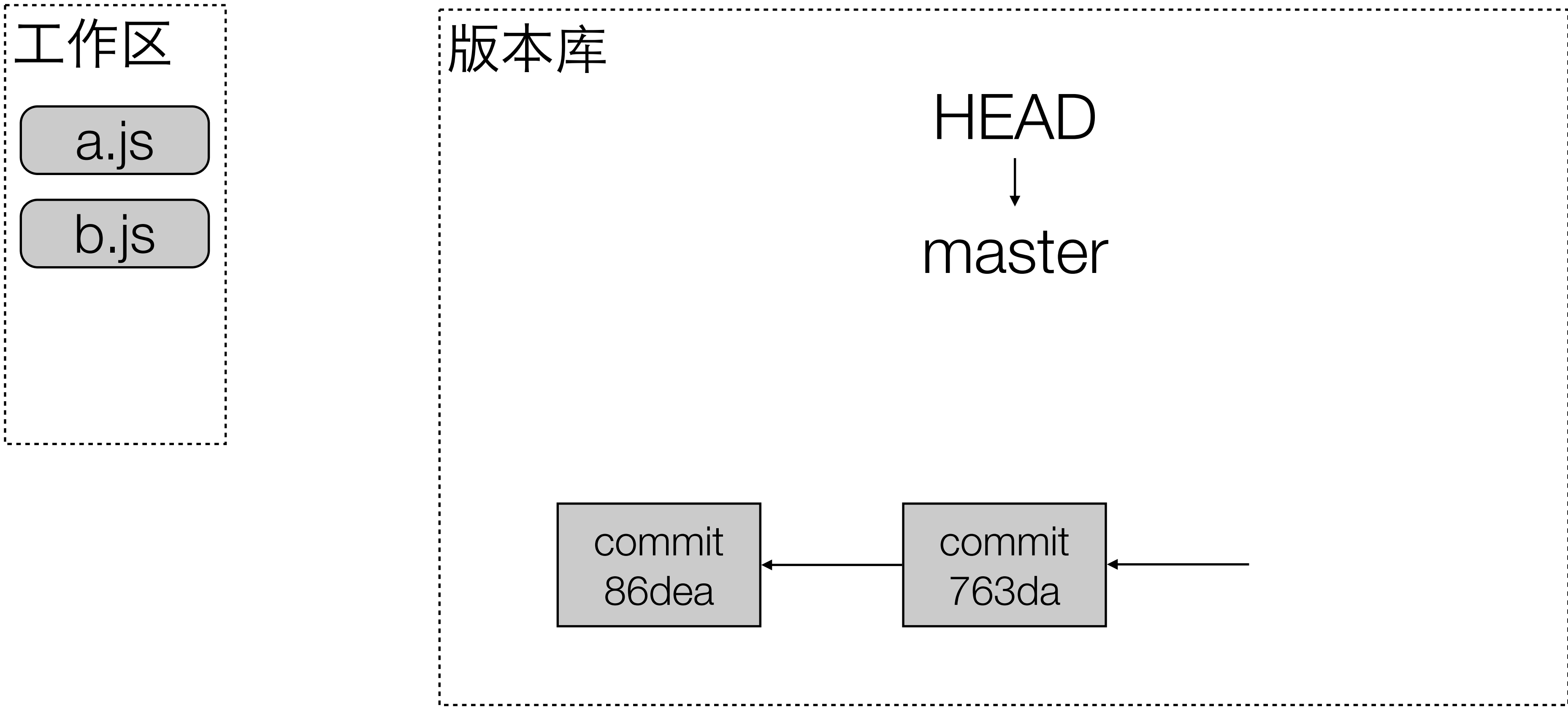
Git 暂存区

git commit

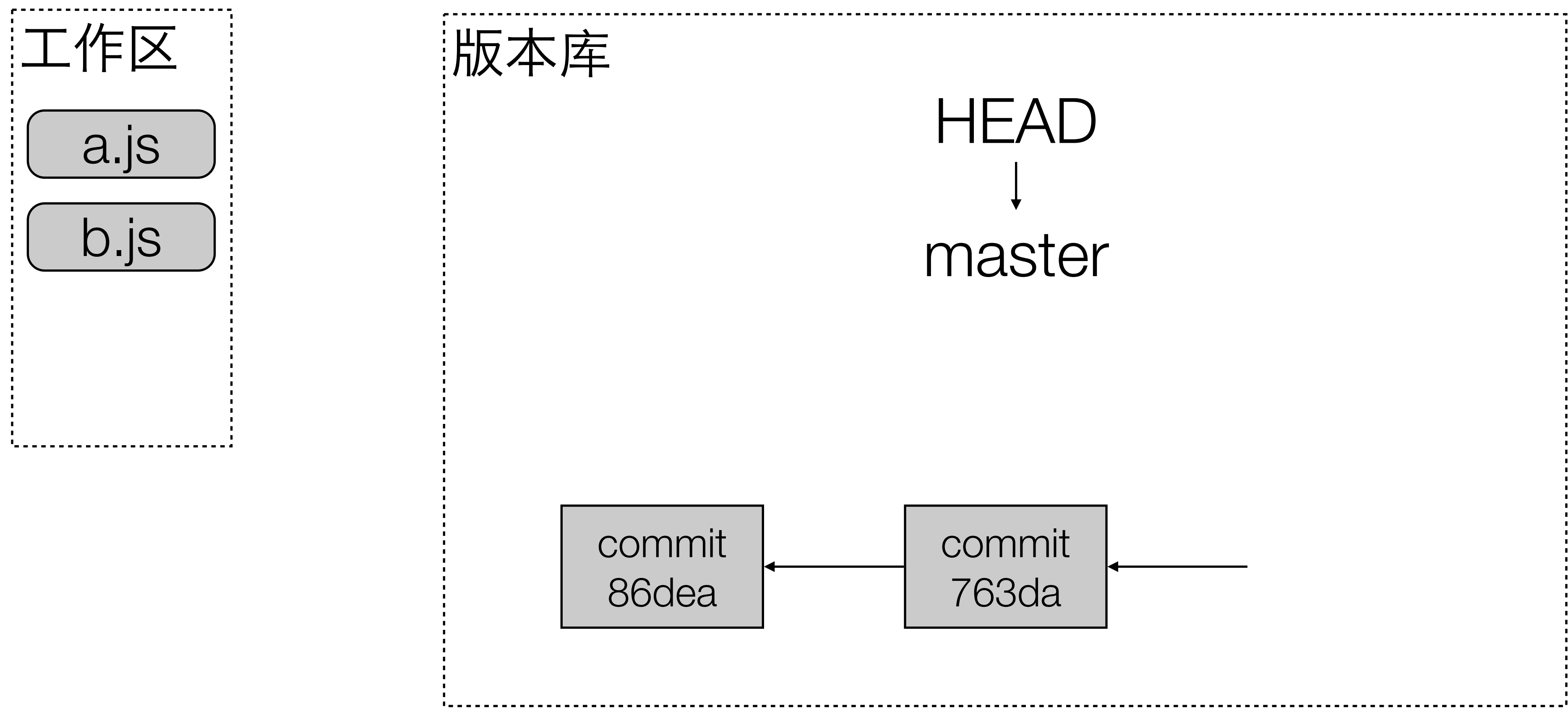


Git 暂存区

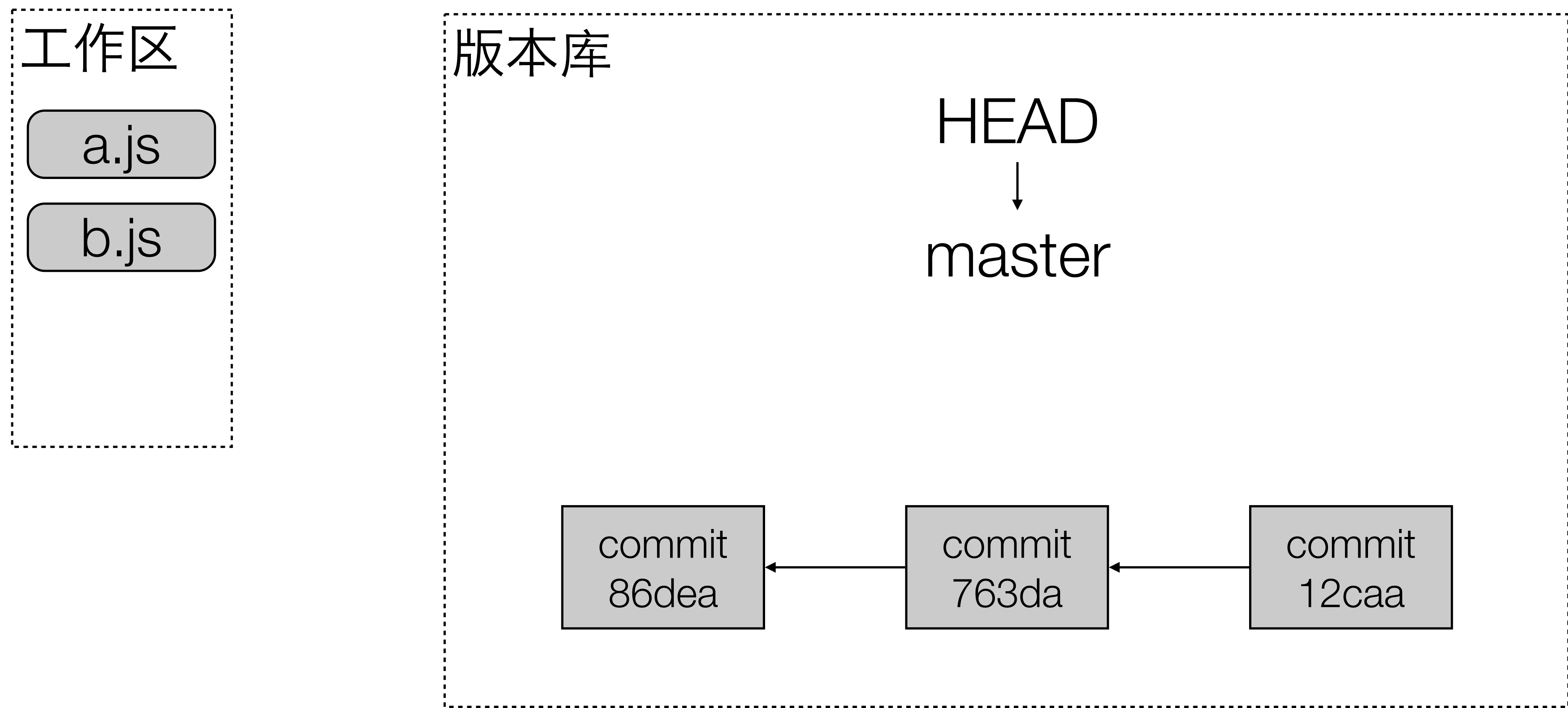
git commit



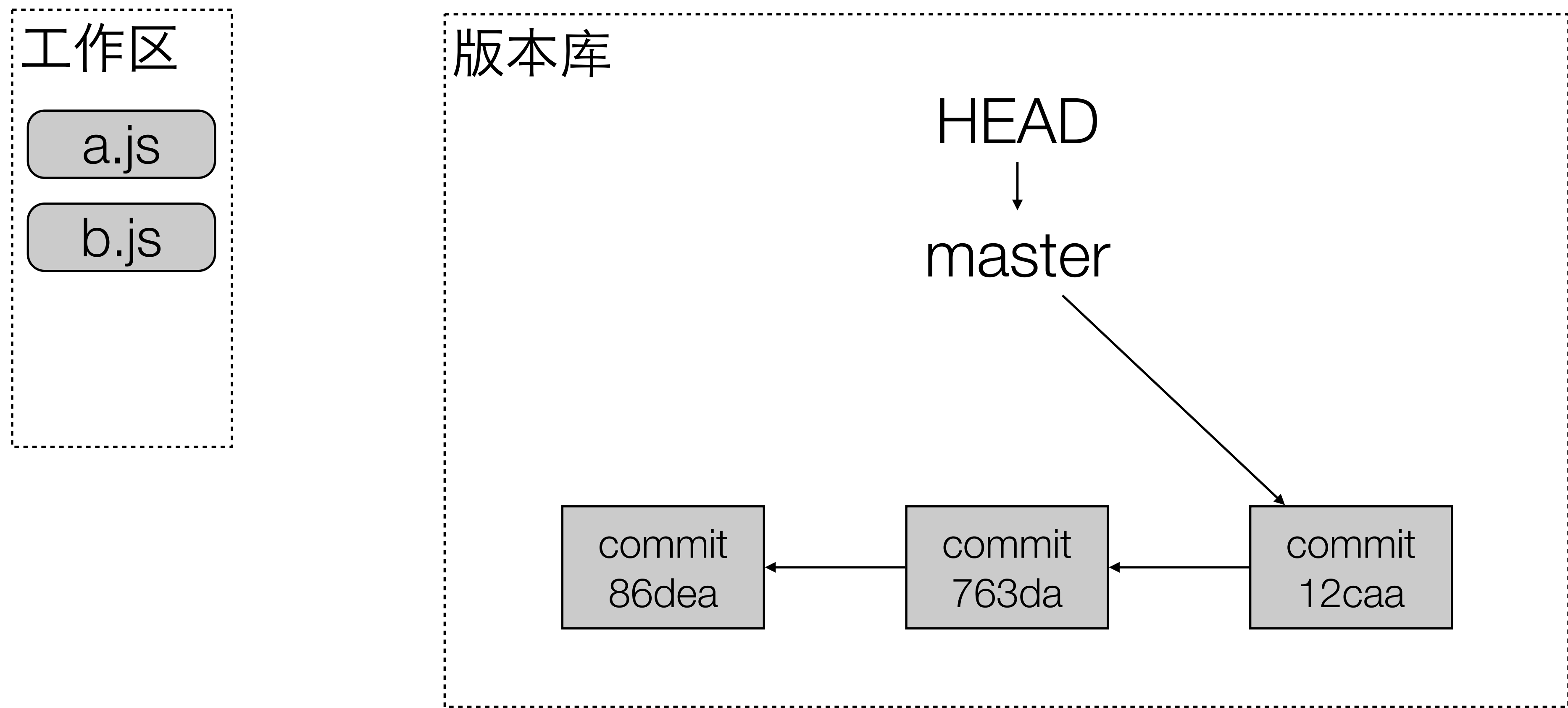
Git 暂存区



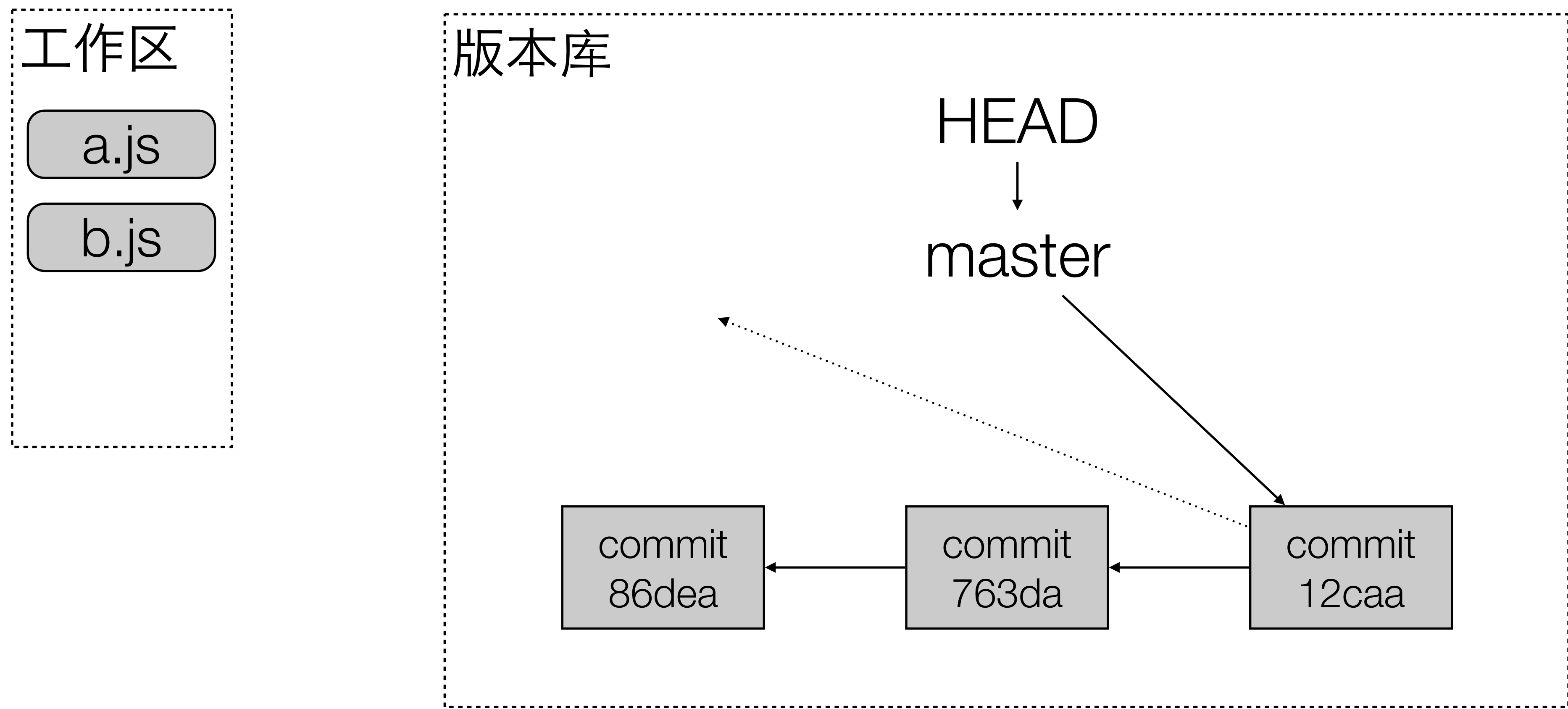
Git 暂存区



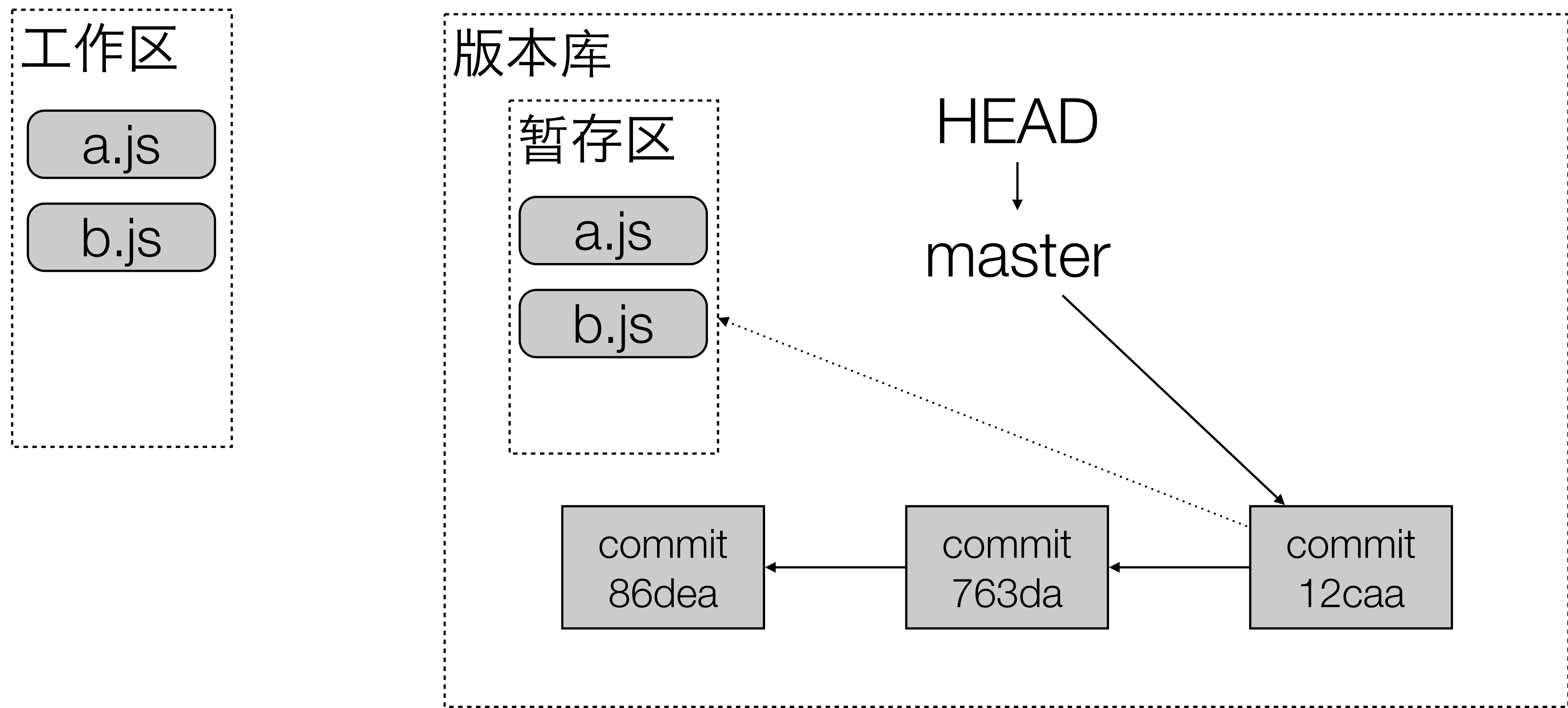
Git 暂存区



Git 暂存区

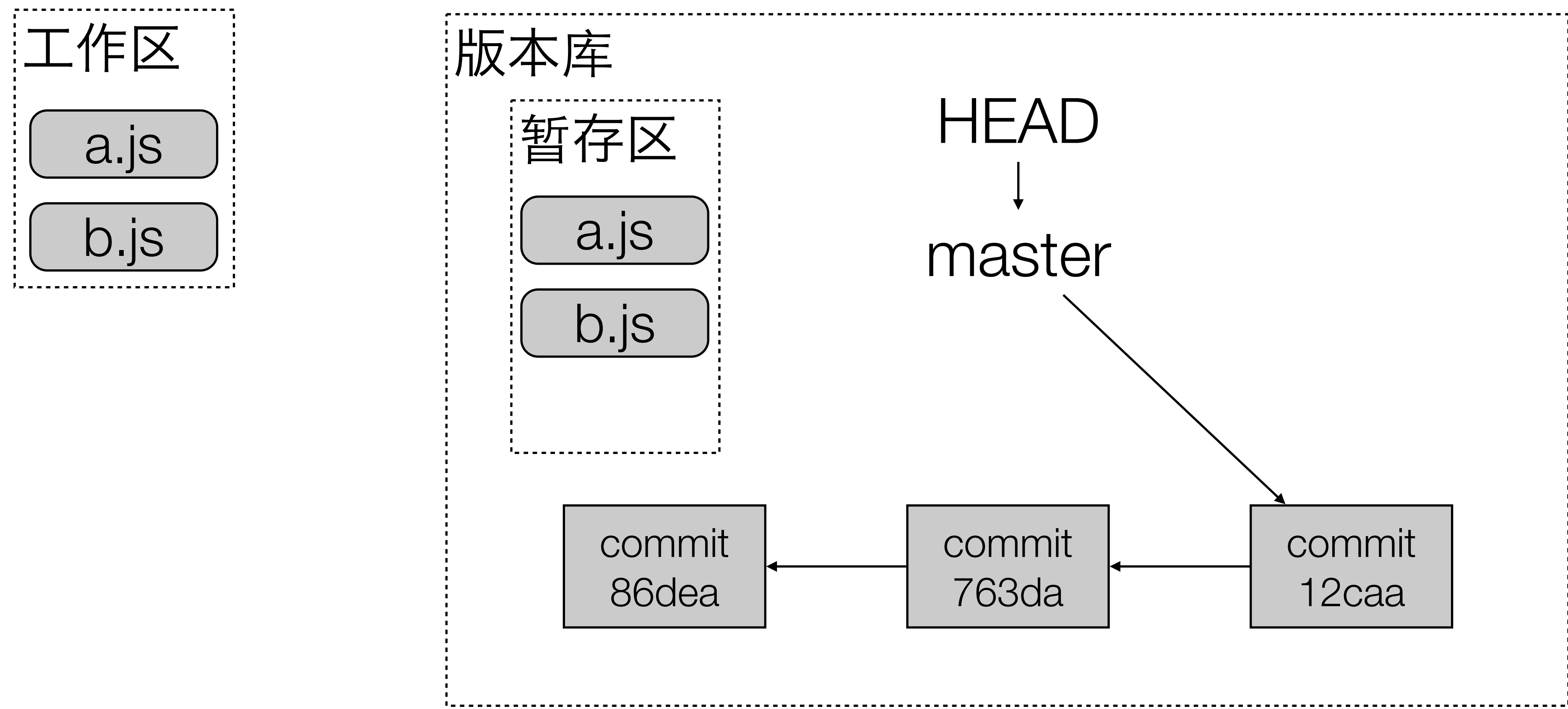


Git 暂存区



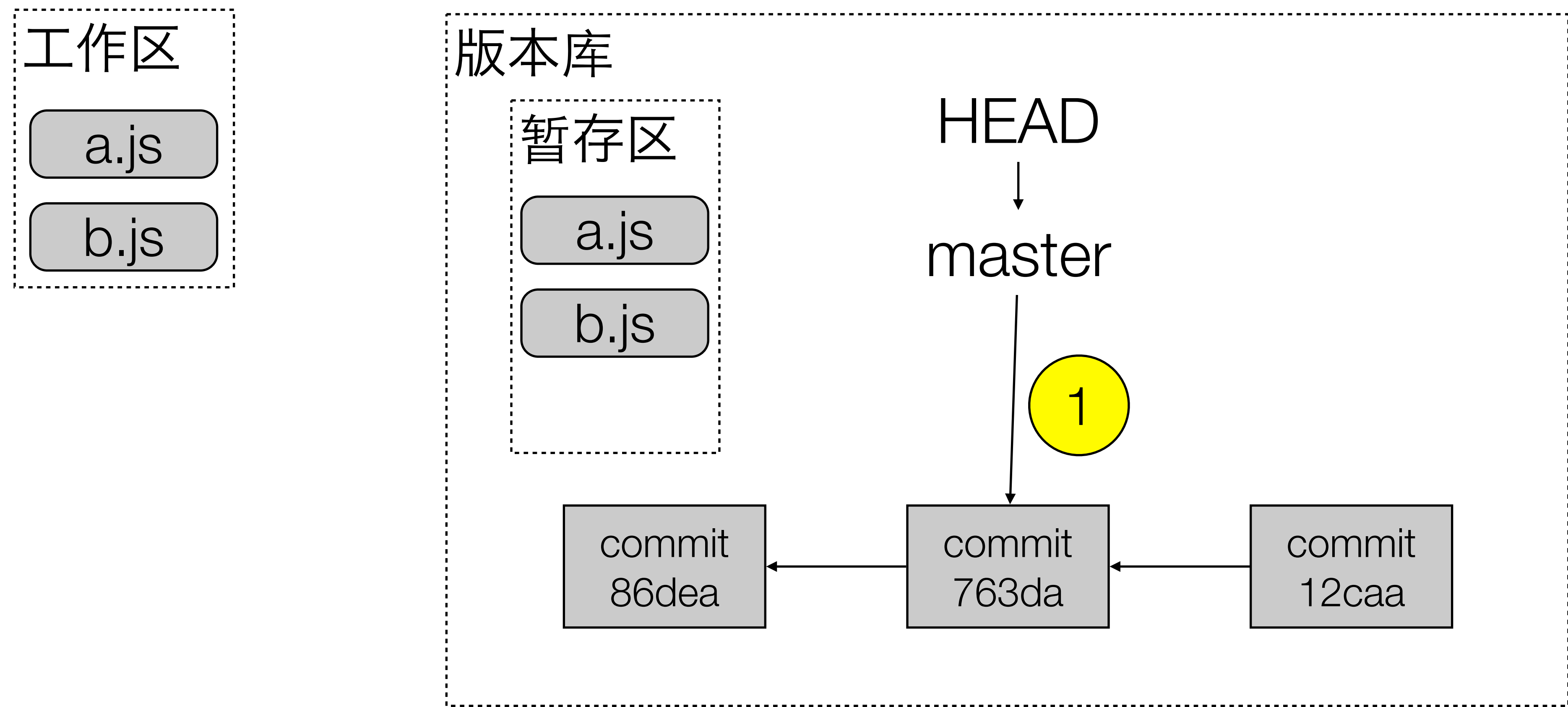
Git 分支指针控制

```
git reset --soft 763da
```



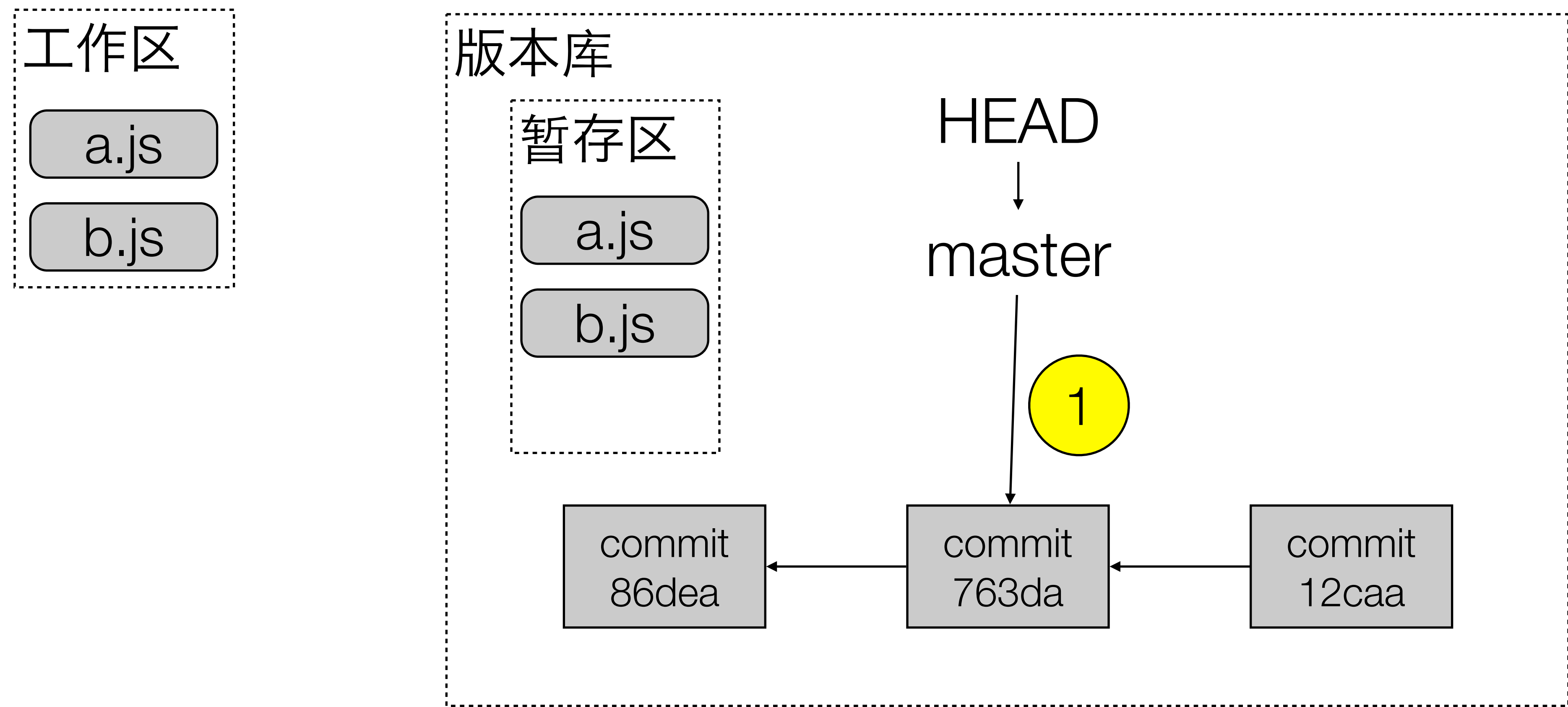
Git 分支指针控制

```
git reset --soft 763da
```



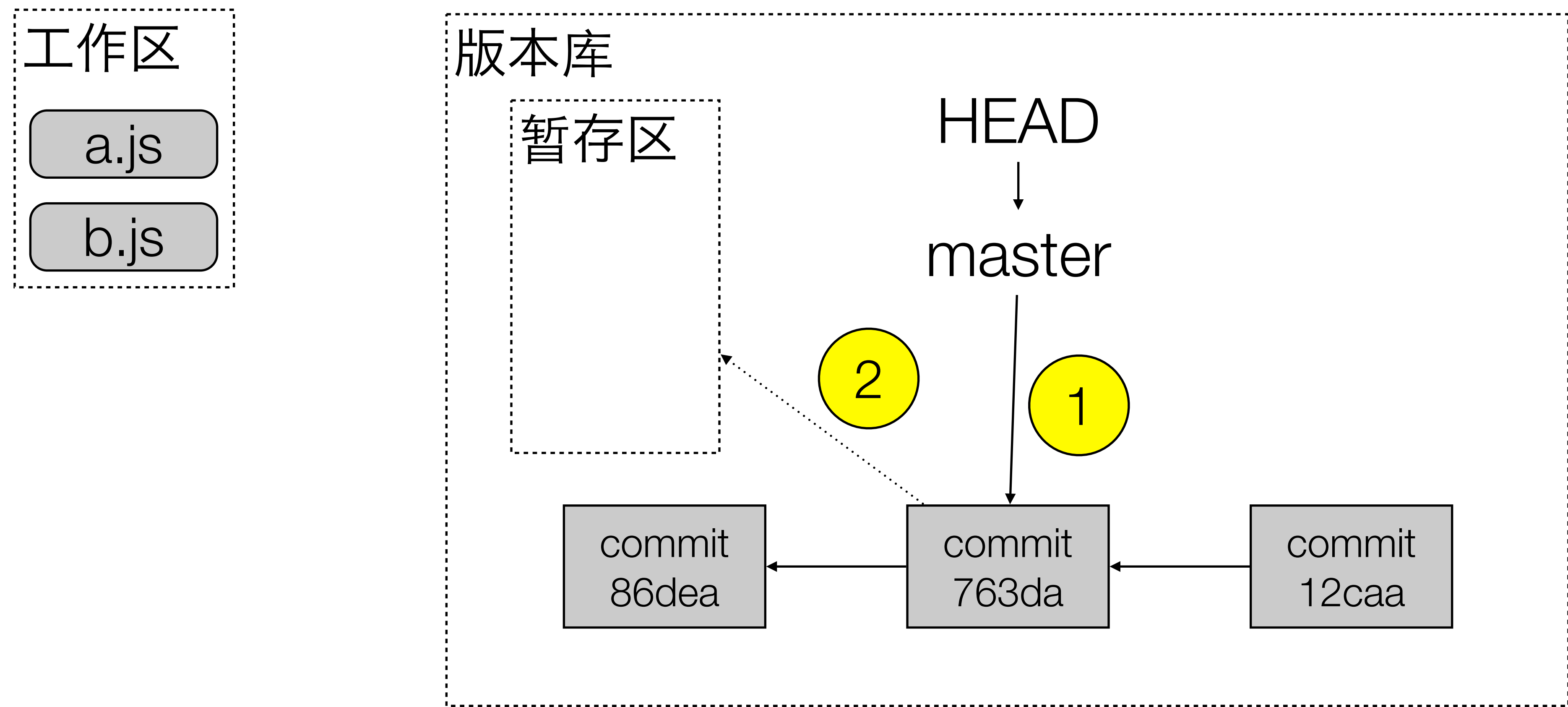
Git 分支指针控制

```
git reset [--mixed] 763da
```



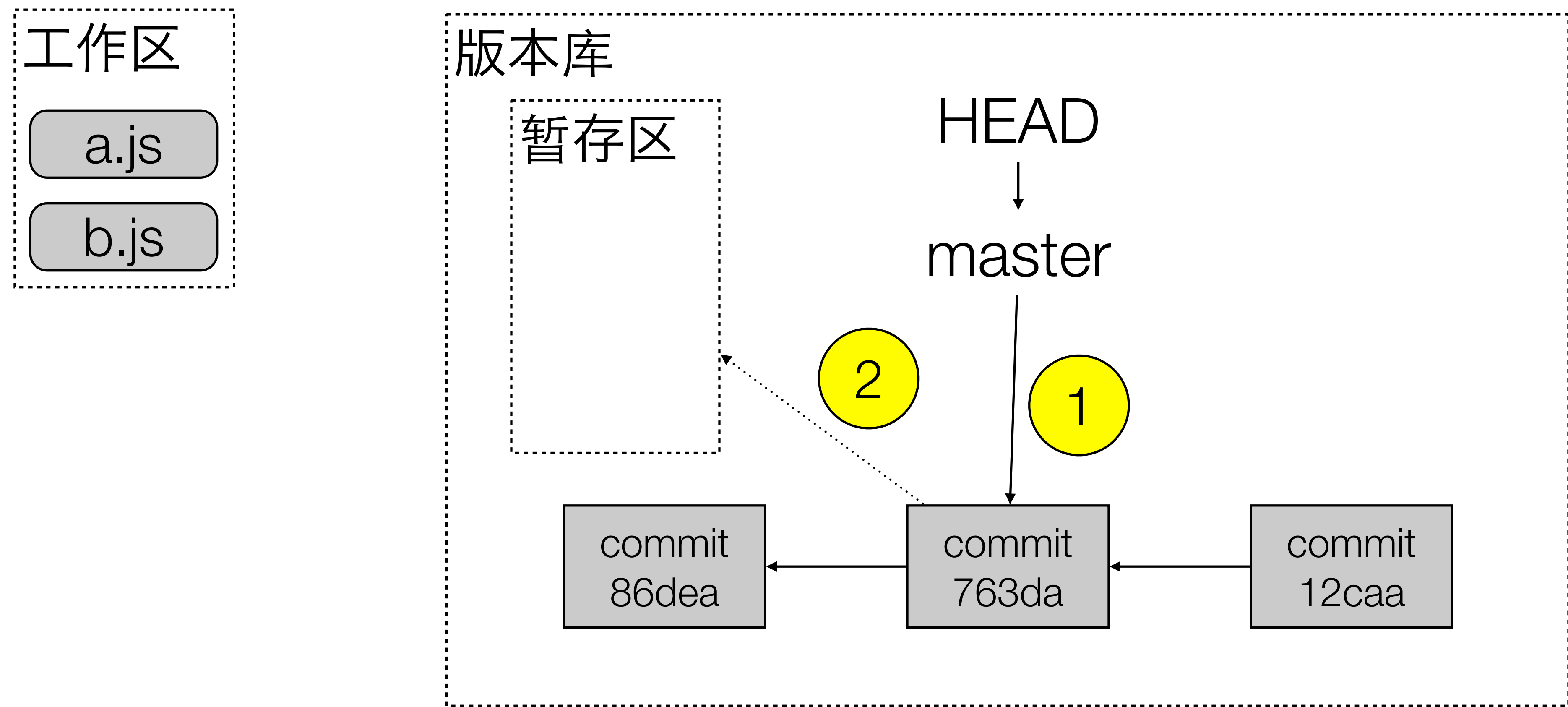
Git 分支指针控制

```
git reset [--mixed] 763da
```



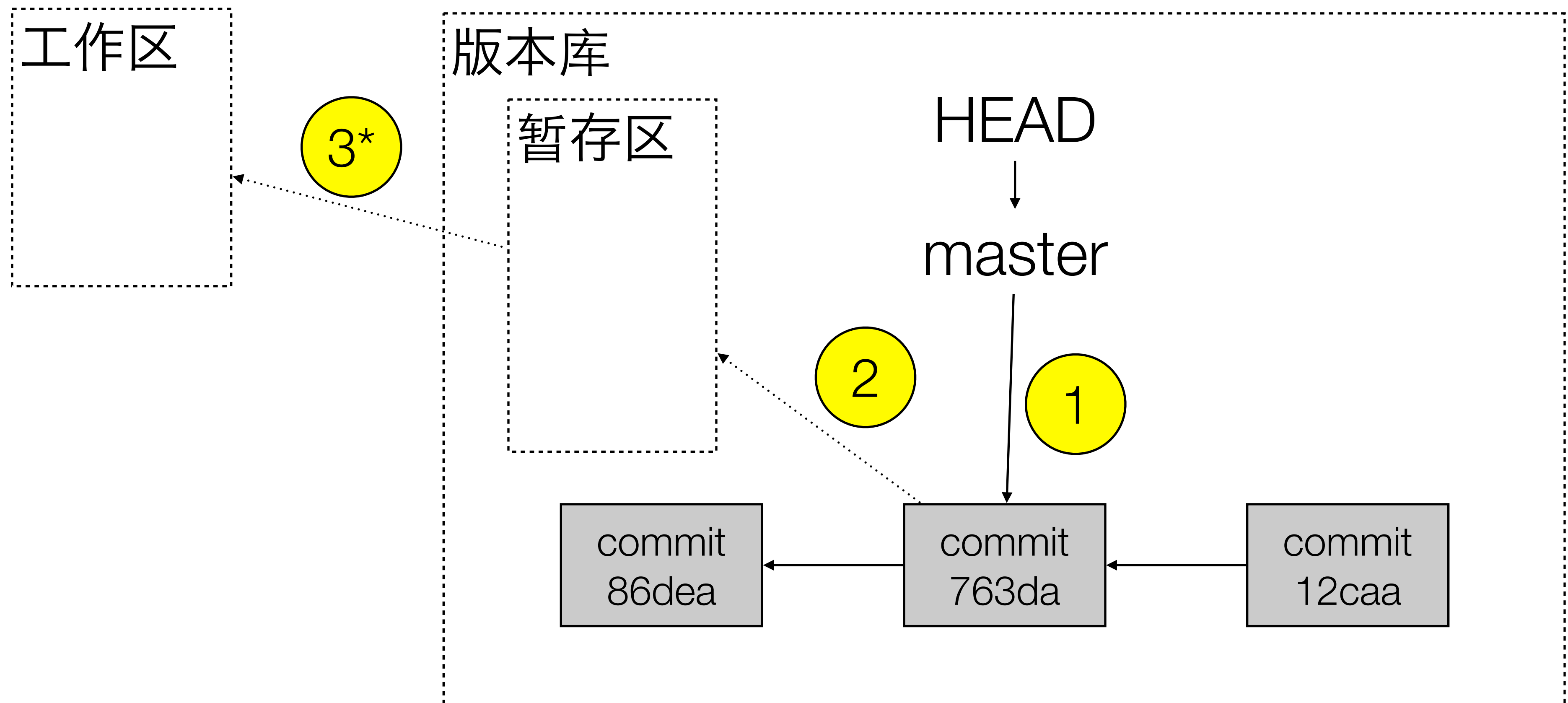
Git 分支指针控制

```
git reset --hard 763da
```



Git 分支指针控制

git reset --hard 763da

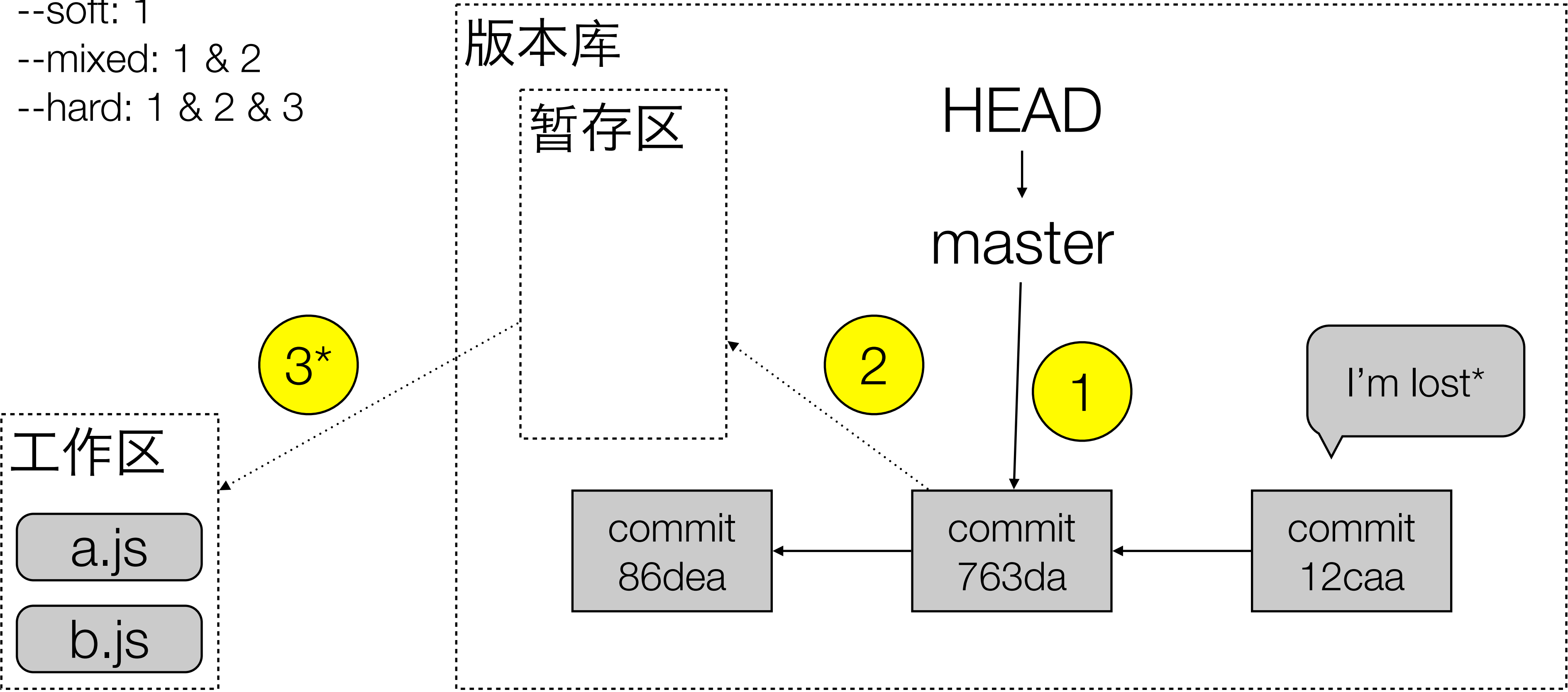


* 此处仅做演示 --hard 的效果，实际上 git 不会删除暂存区中不存在的文件，只会添加新文件和重置文件修改

Git 分支指针控制

git reset

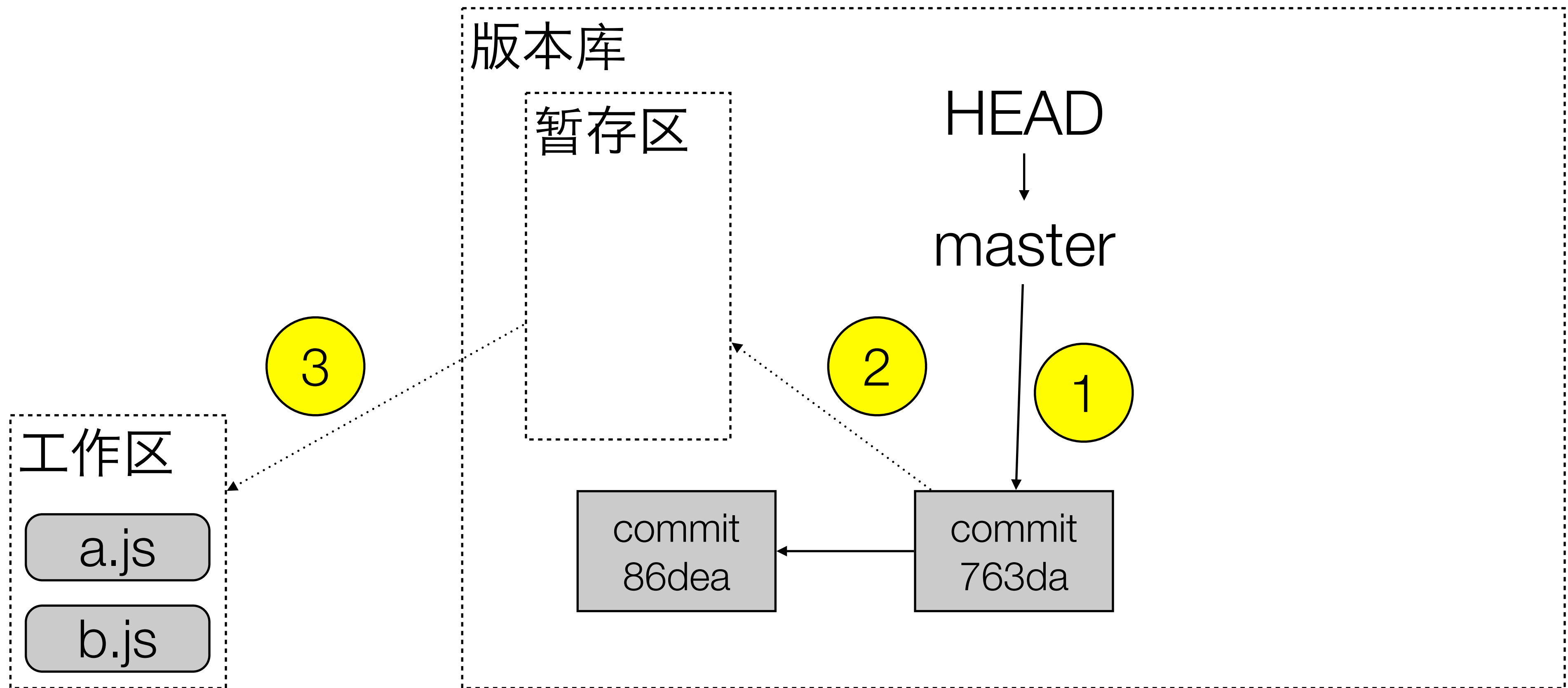
- soft: 1
- mixed: 1 & 2
- hard: 1 & 2 & 3



* 可以通过 `git reflog` 查看 HEAD 指针最近的移动历史

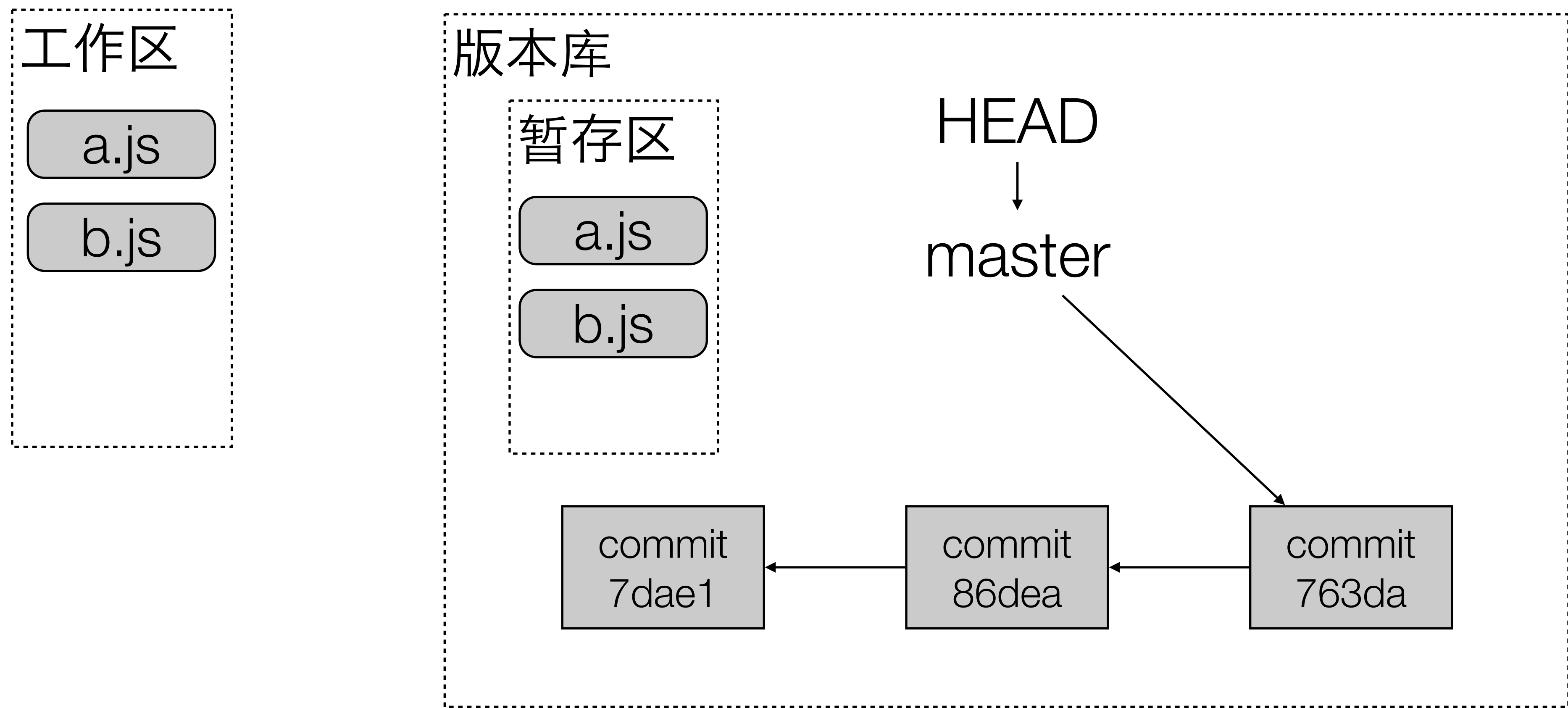
Git 分支指针控制

3 == git checkout .



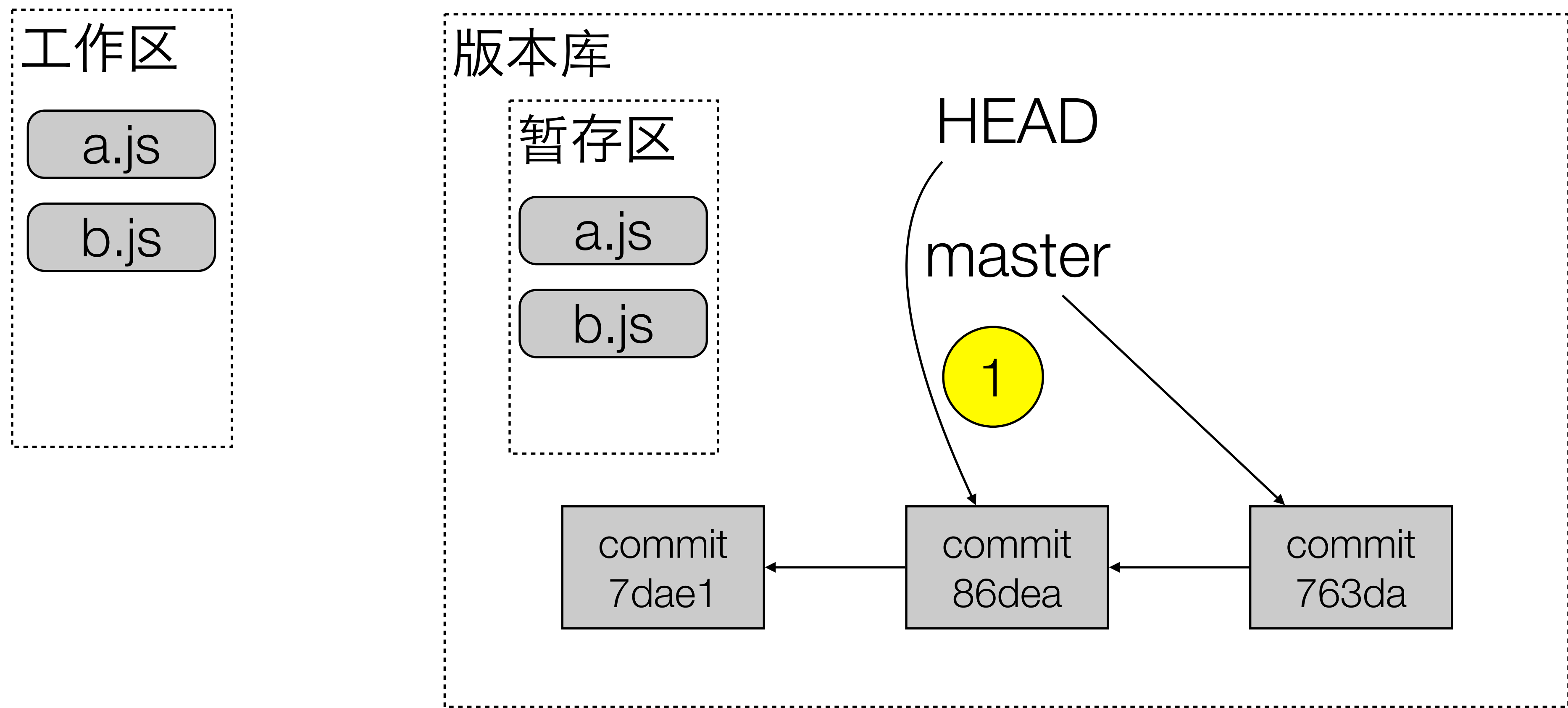
Git HEAD 指针控制

git checkout 86dea



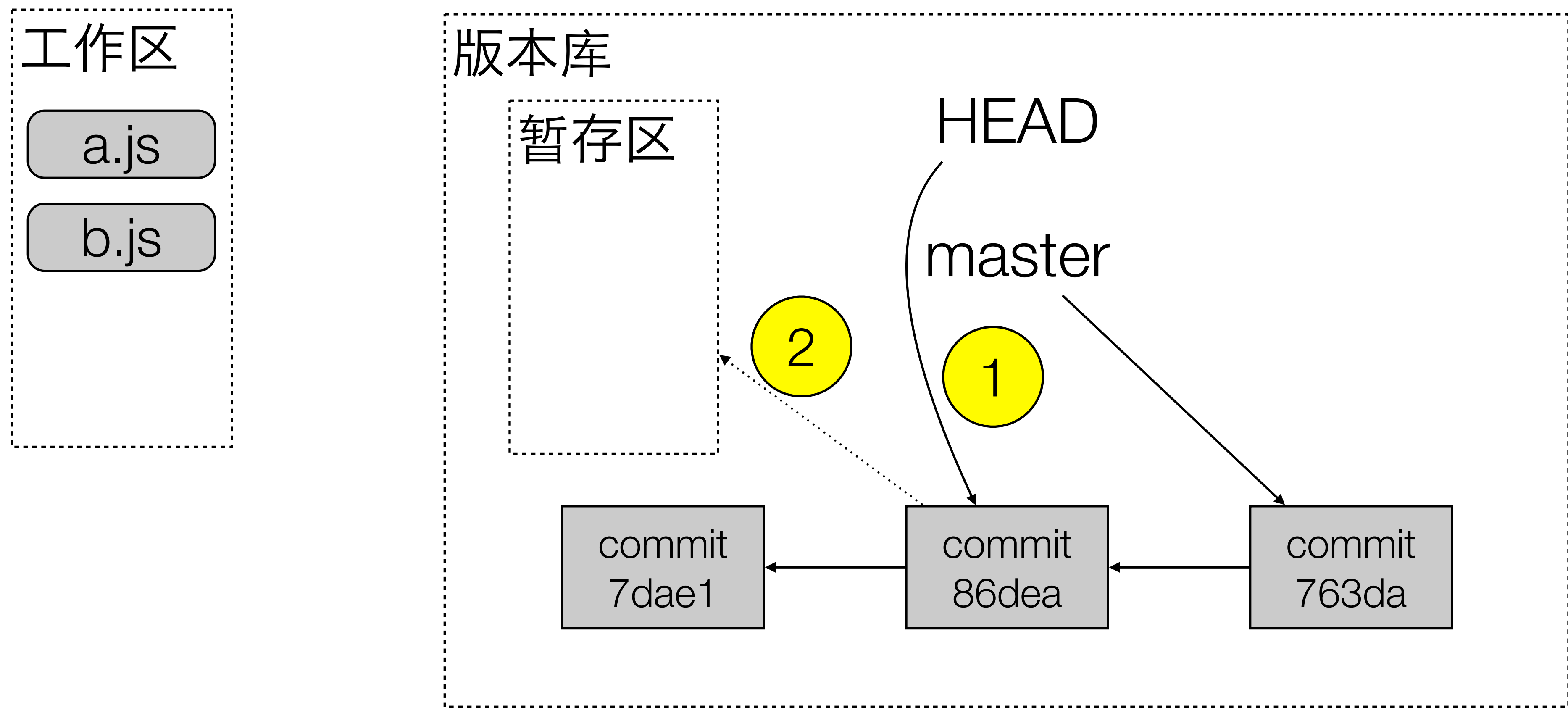
Git HEAD 指针控制

git checkout 86dea



Git HEAD 指针控制

git checkout 86dea



detached HEAD 状态

git checkout 86dea

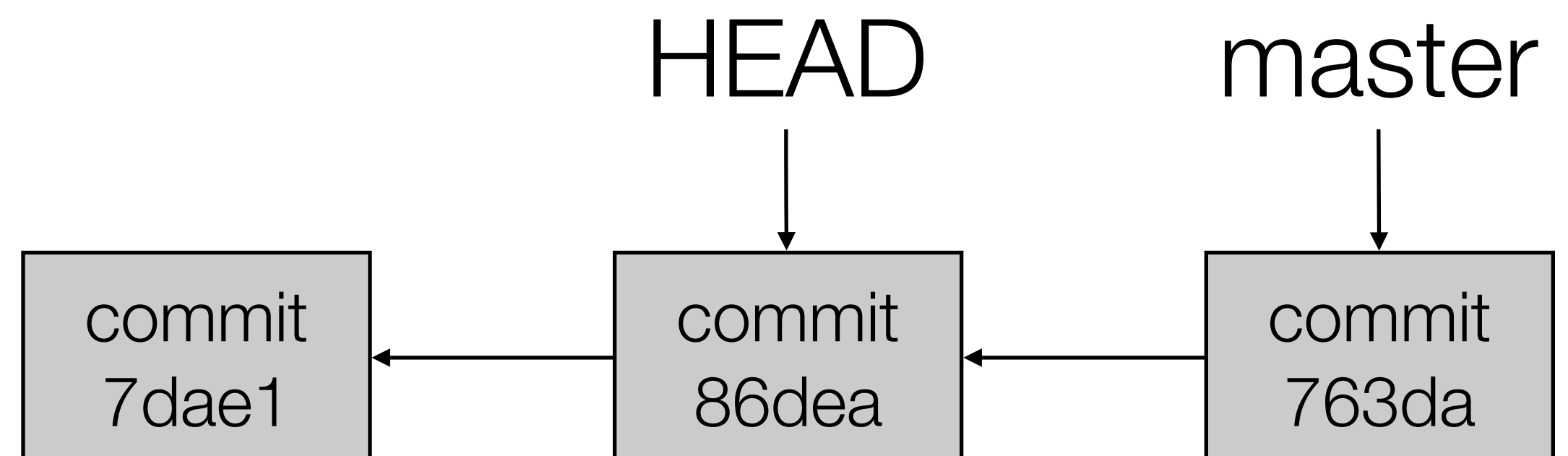
Note: checking out '86dea'.

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.

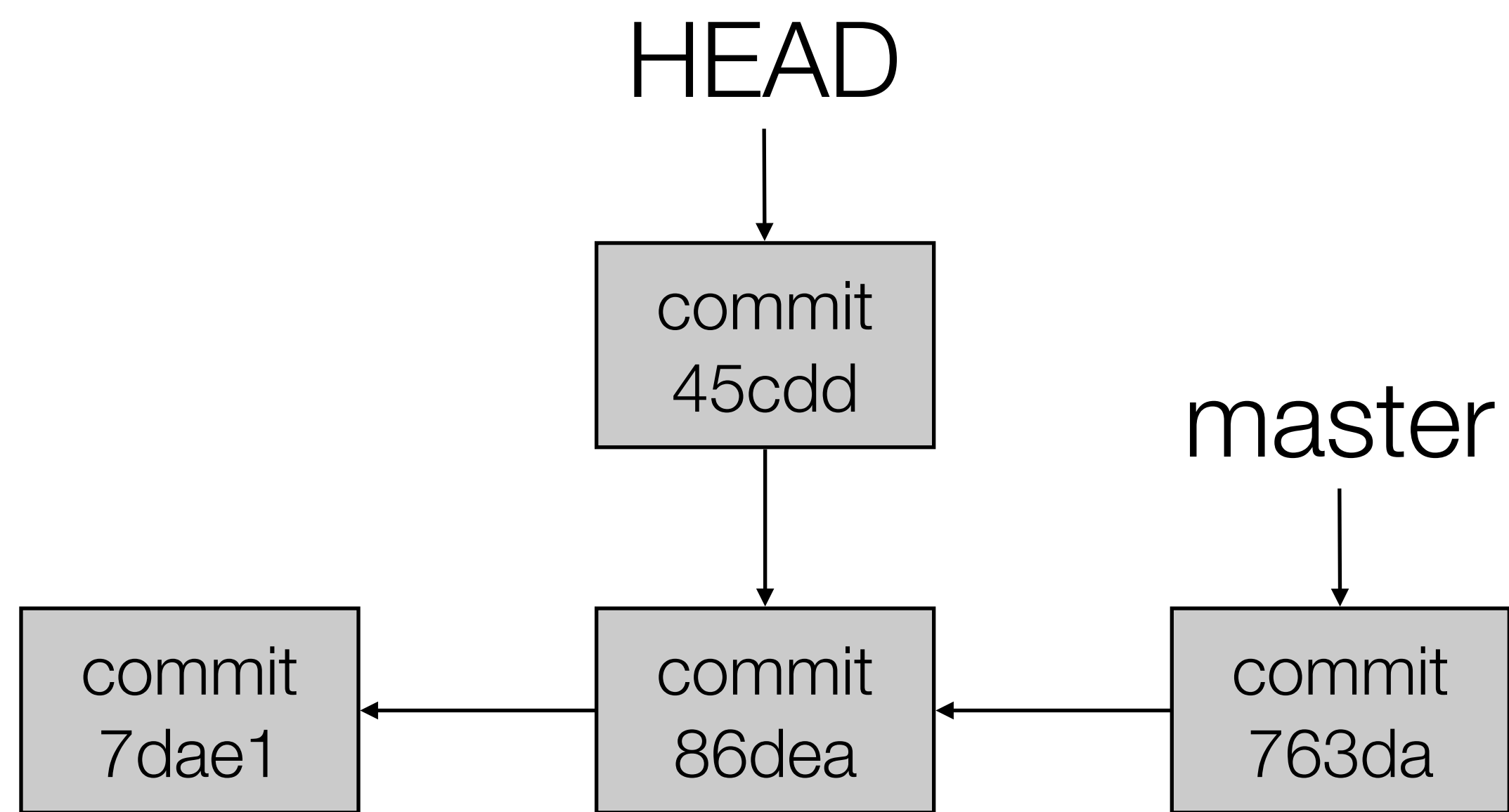
If you want to create a new branch to retain commits you create, you may do so (now or later) by using -b with the checkout command again. Example:

```
git checkout -b new_branch_name
```

HEAD is now at 78f871f... remove doc

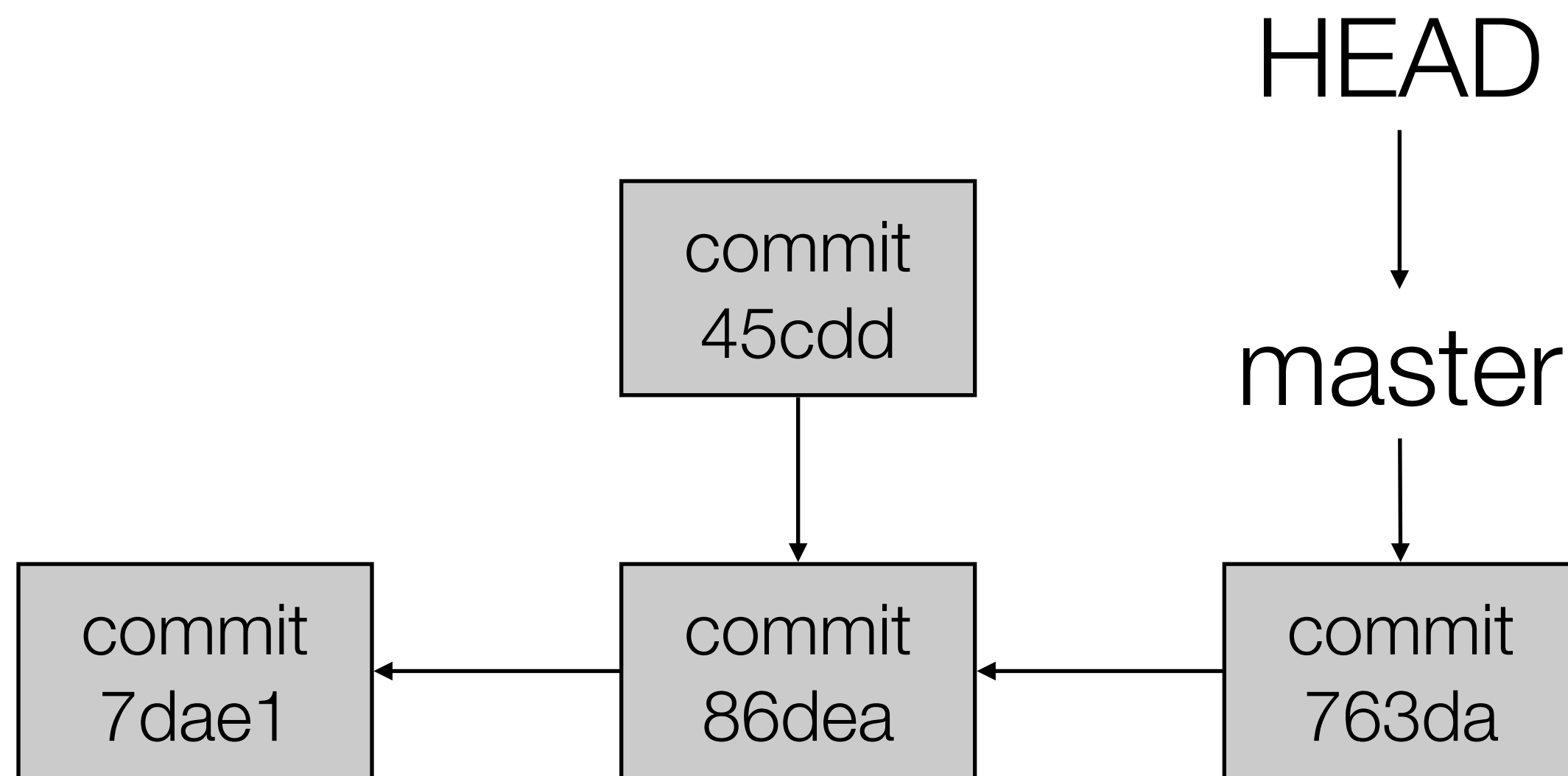


detached HEAD 状态

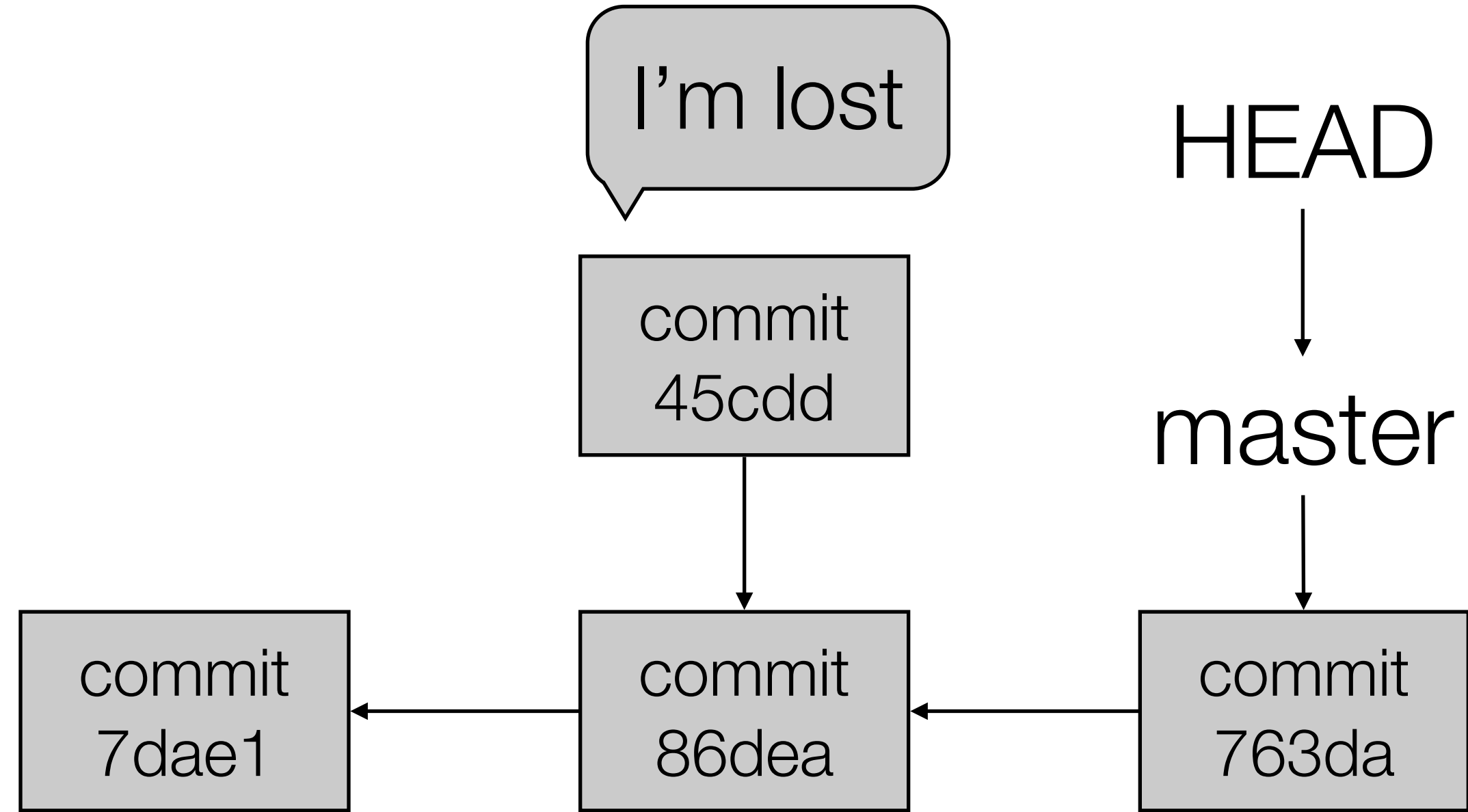


detached HEAD 状态

git checkout master

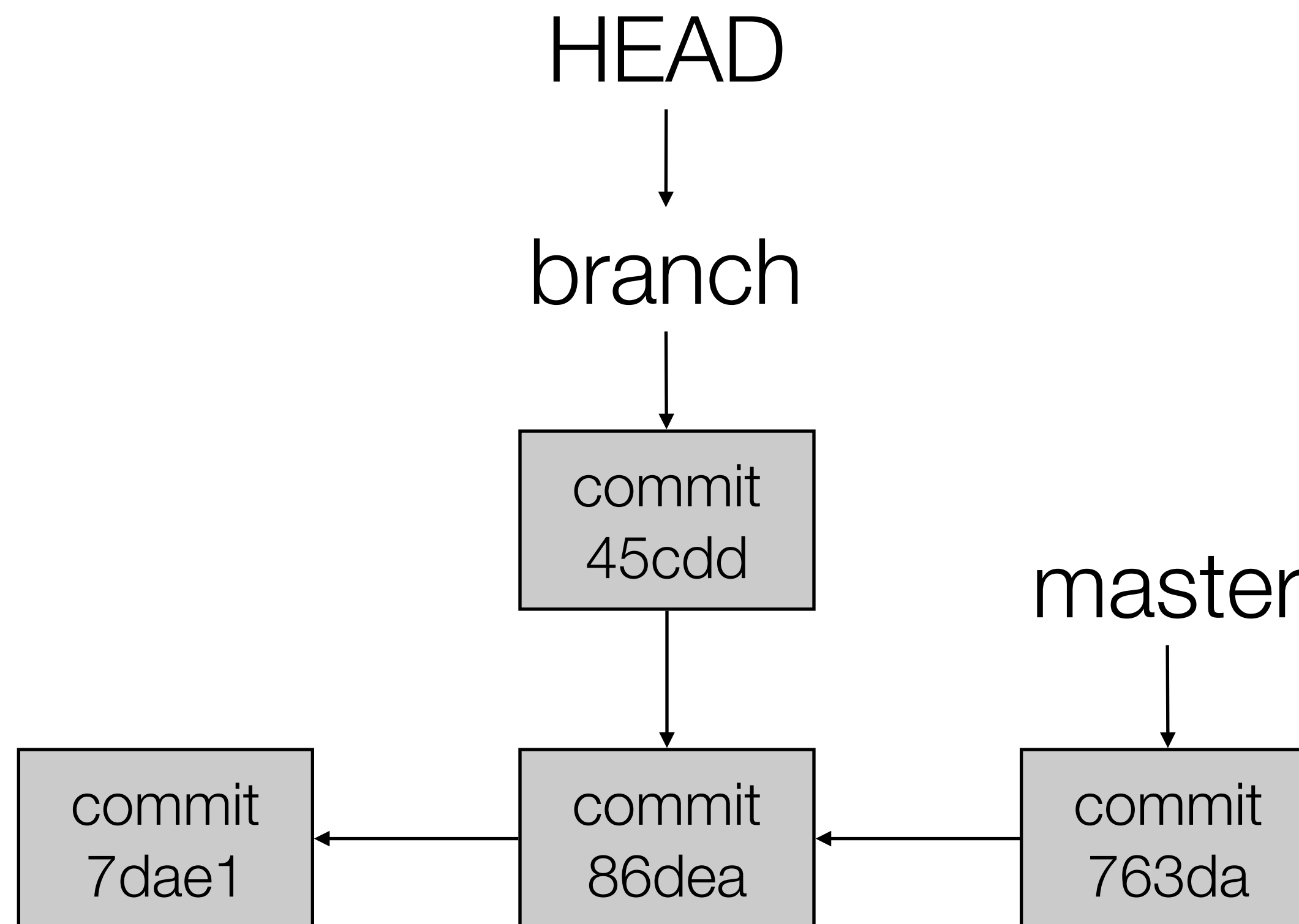


detached HEAD 状态



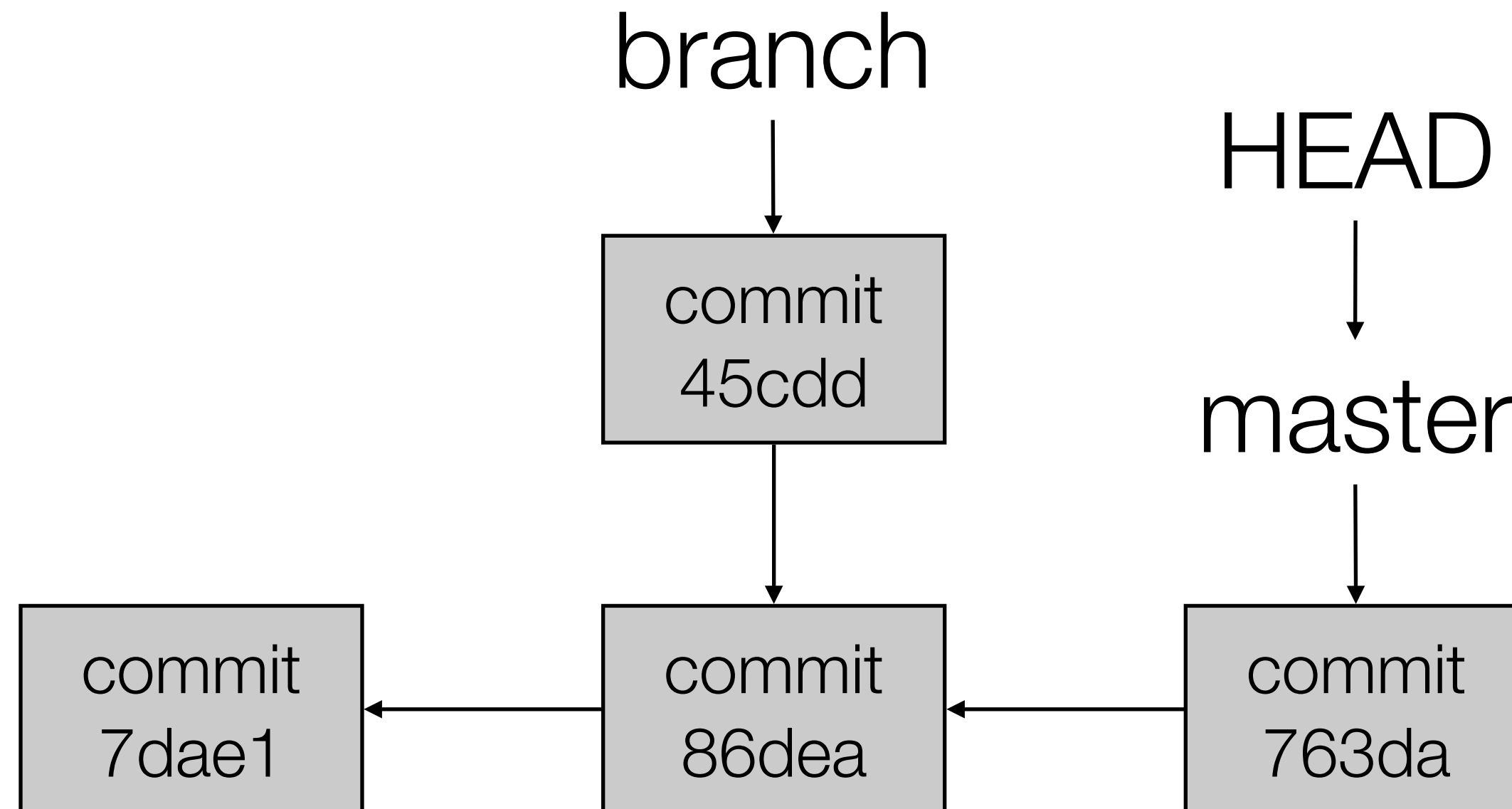
在 detached HEAD 状态上创建分支

git checkout -b branch



在 detached HEAD 状态上创建分支

git checkout master

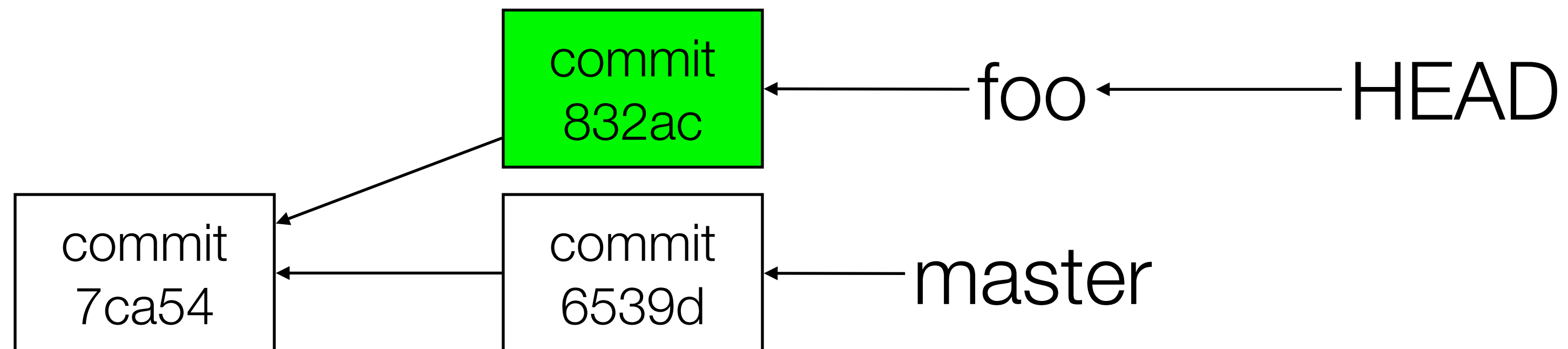


修改历史记录

git rebase

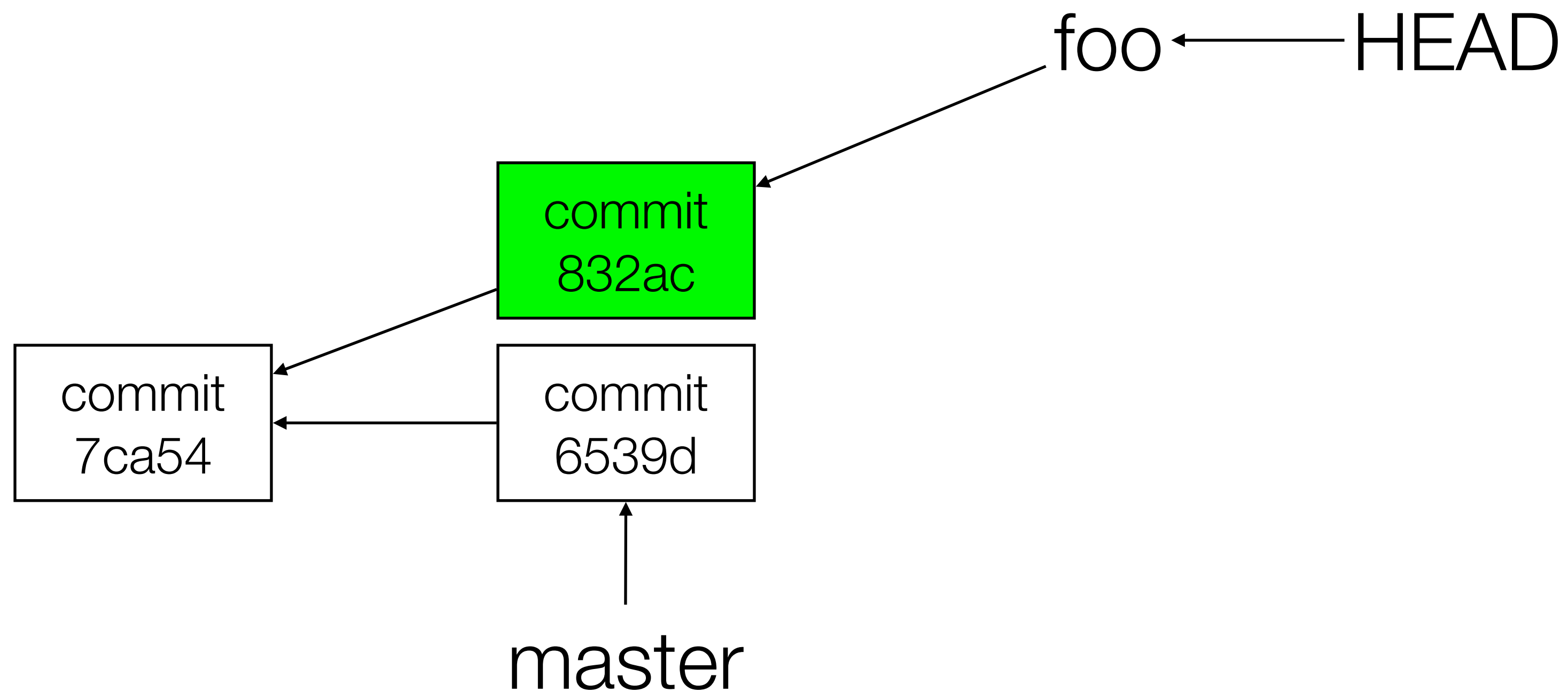
通过 rebase 合并分支

git rebase master



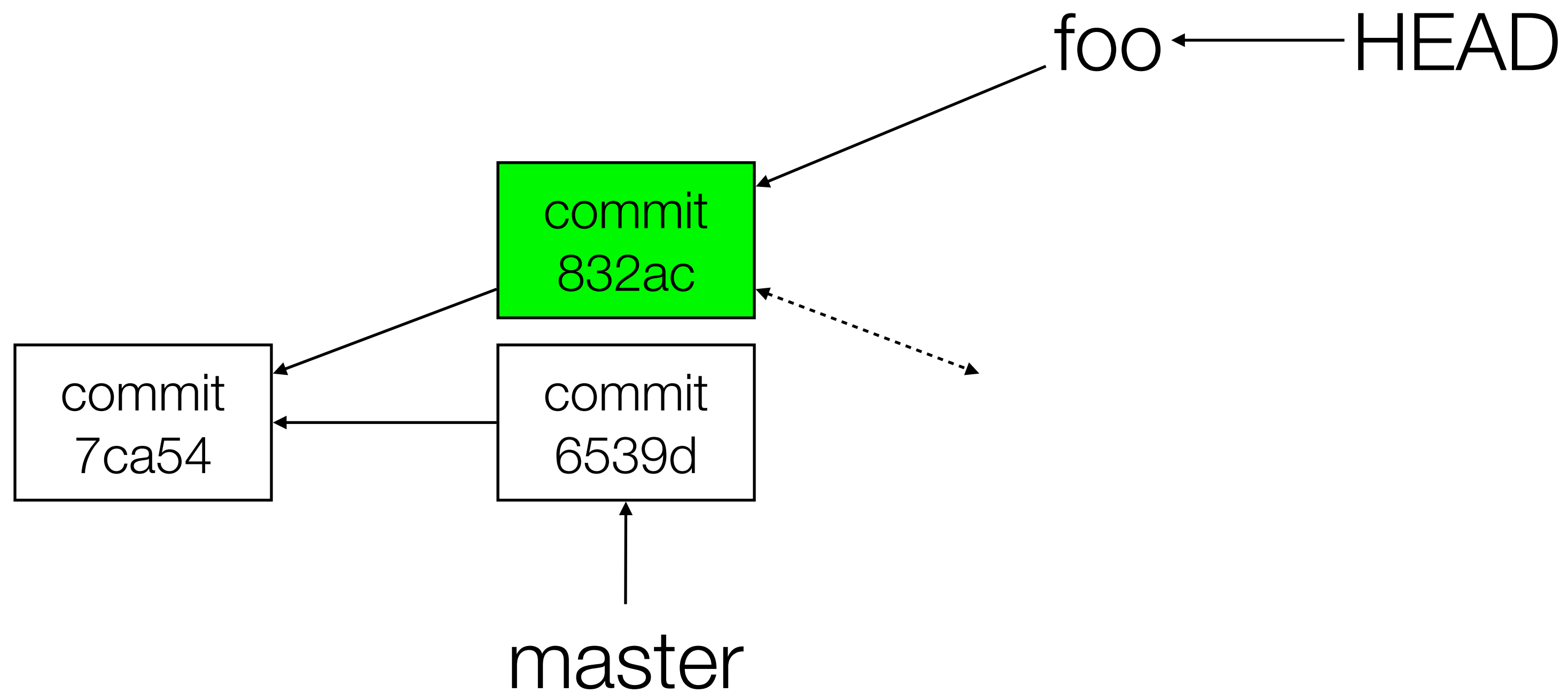
通过 rebase 合并分支

git rebase master



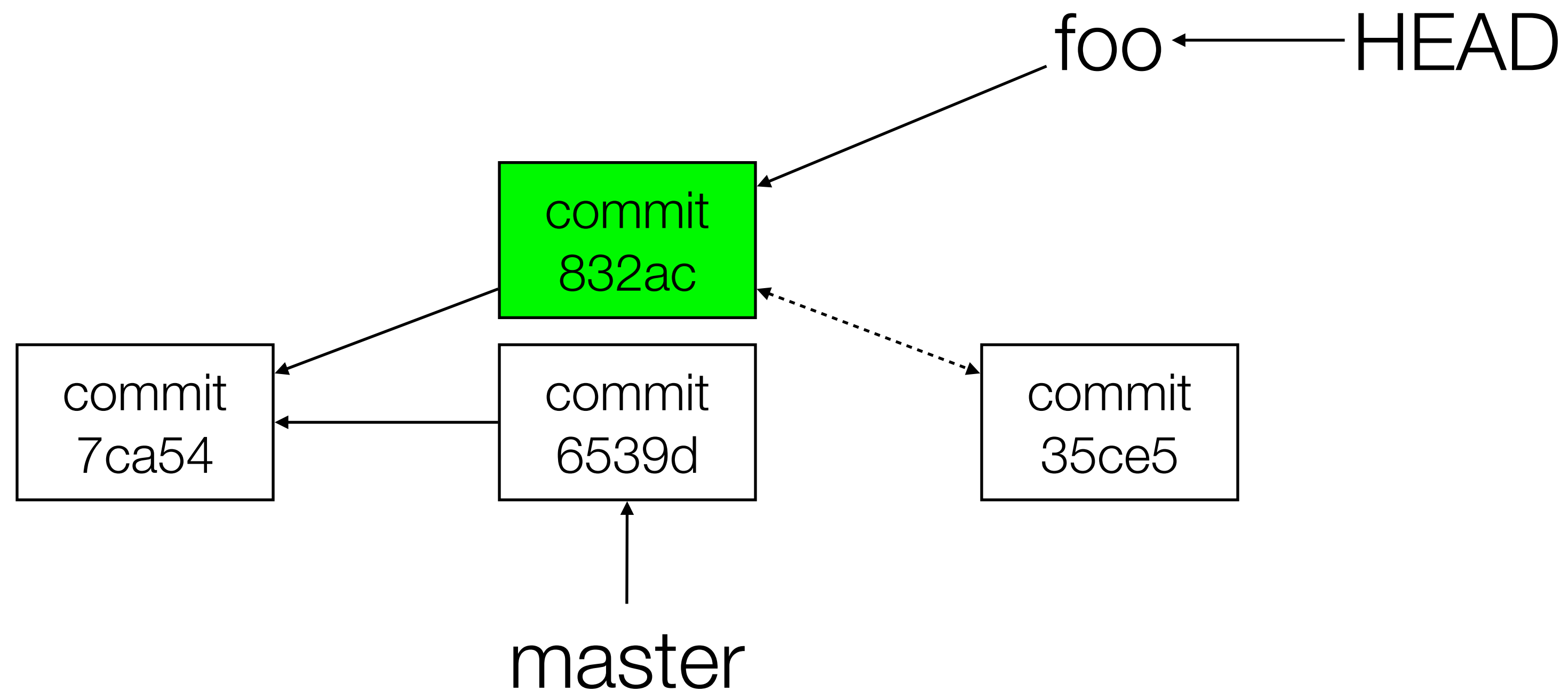
通过 rebase 合并分支

git rebase master



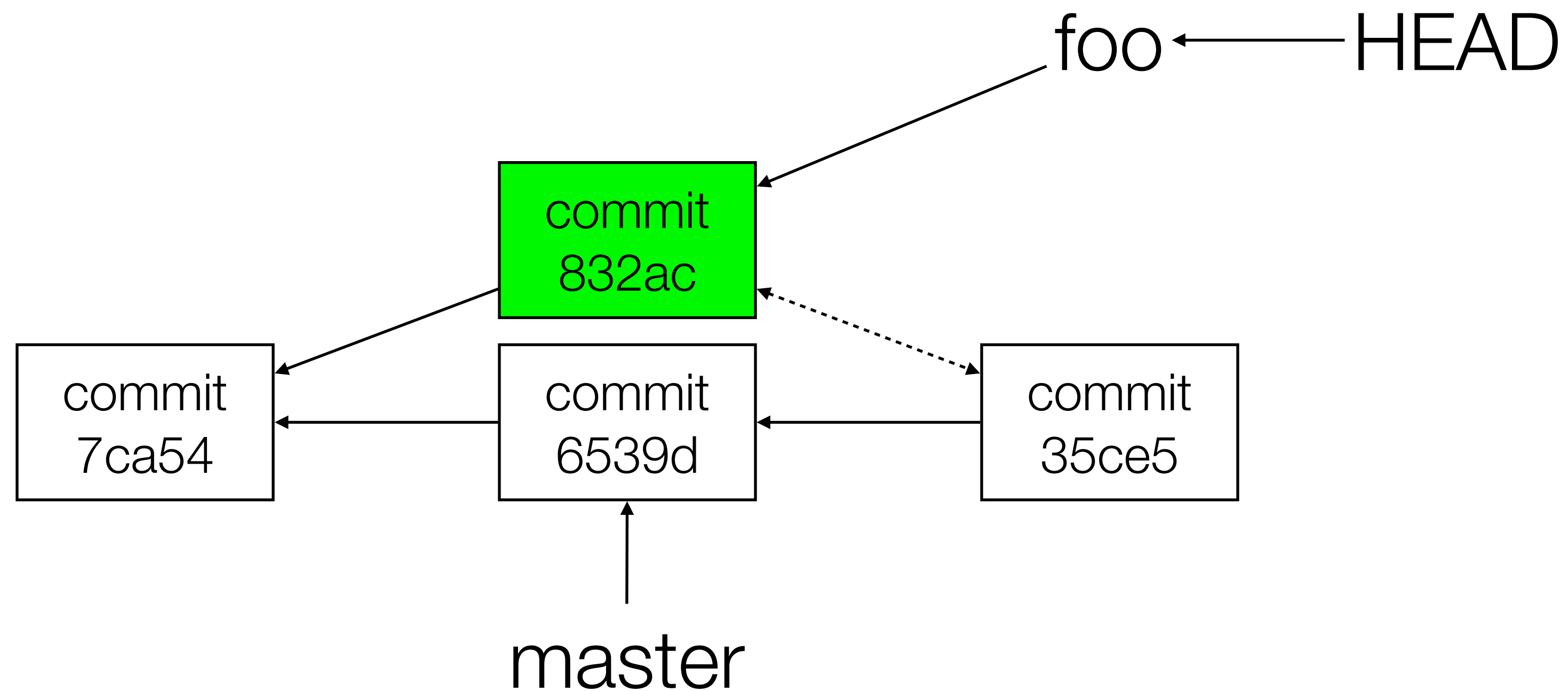
通过 rebase 合并分支

git rebase master



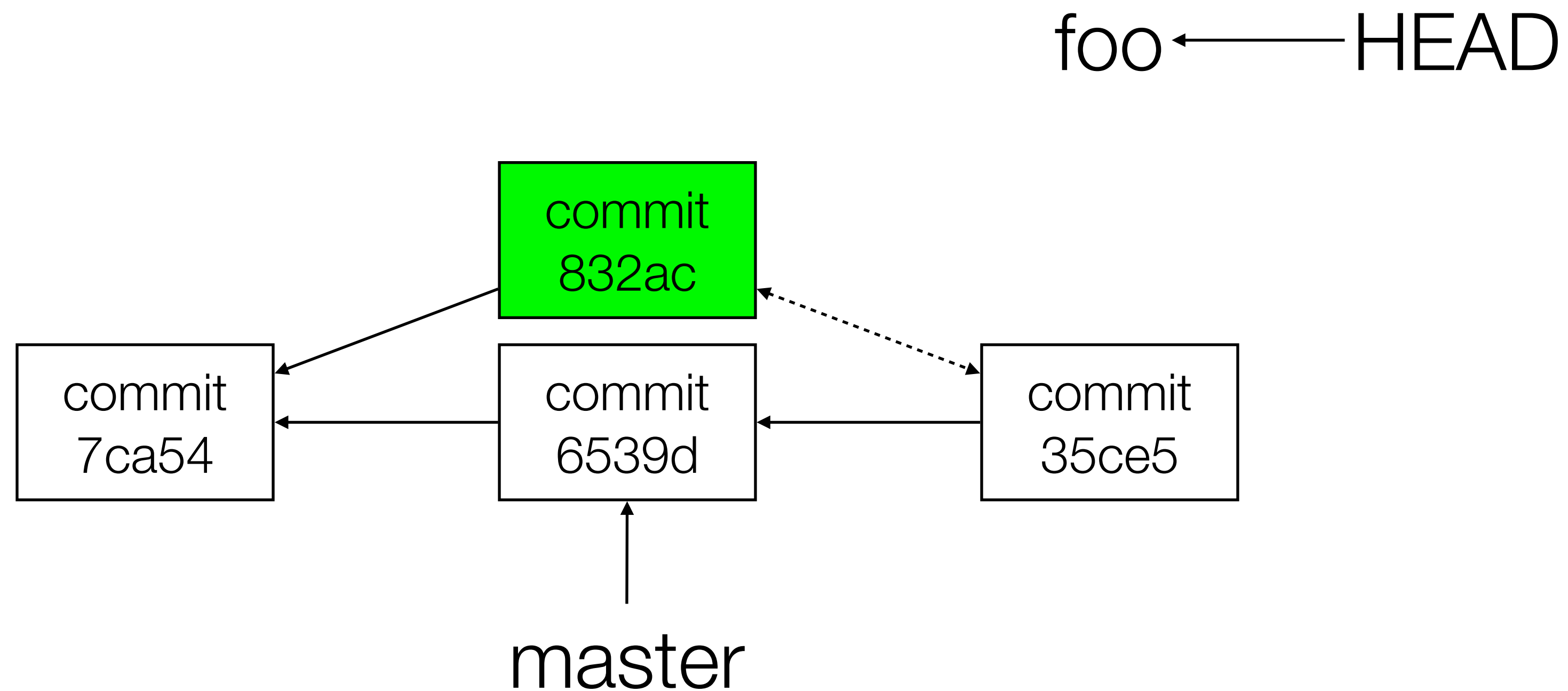
通过 rebase 合并分支

git rebase master



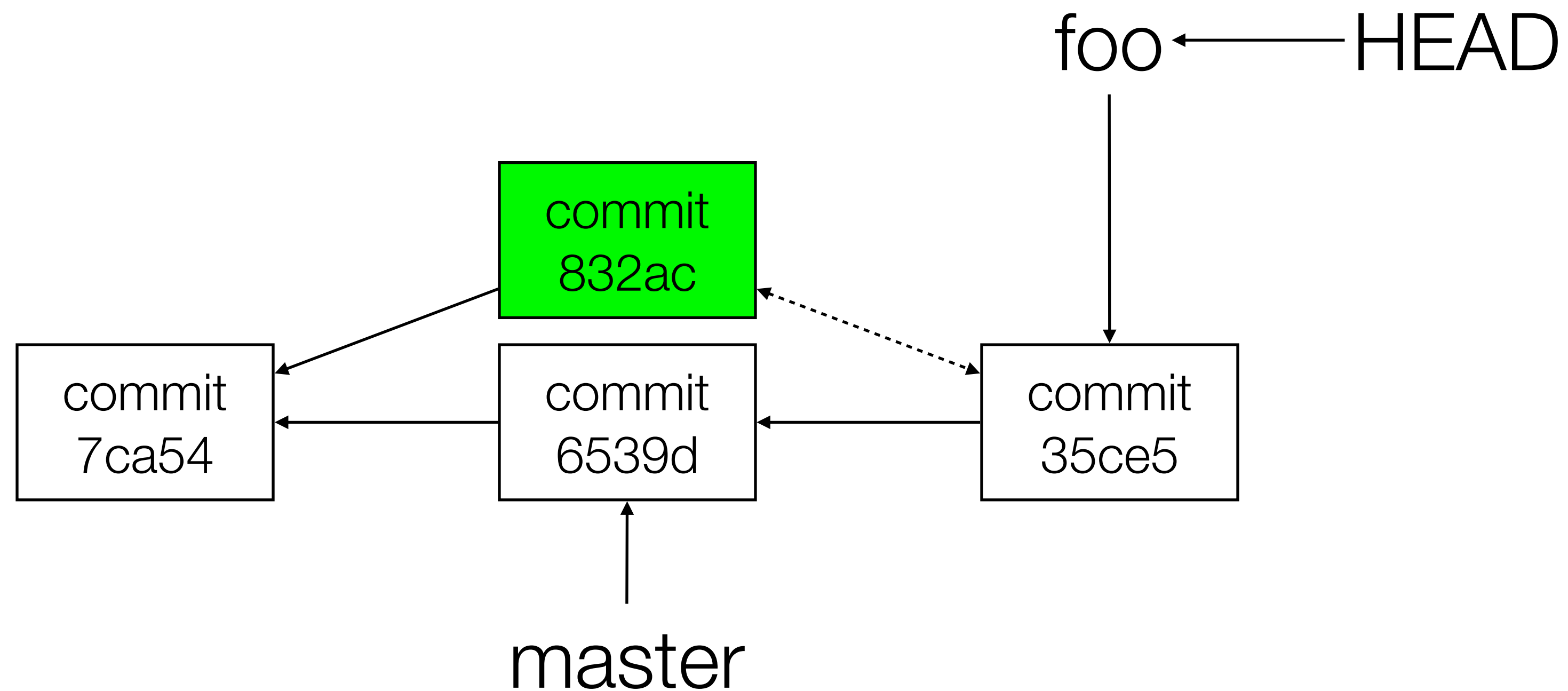
通过 rebase 合并分支

git rebase master



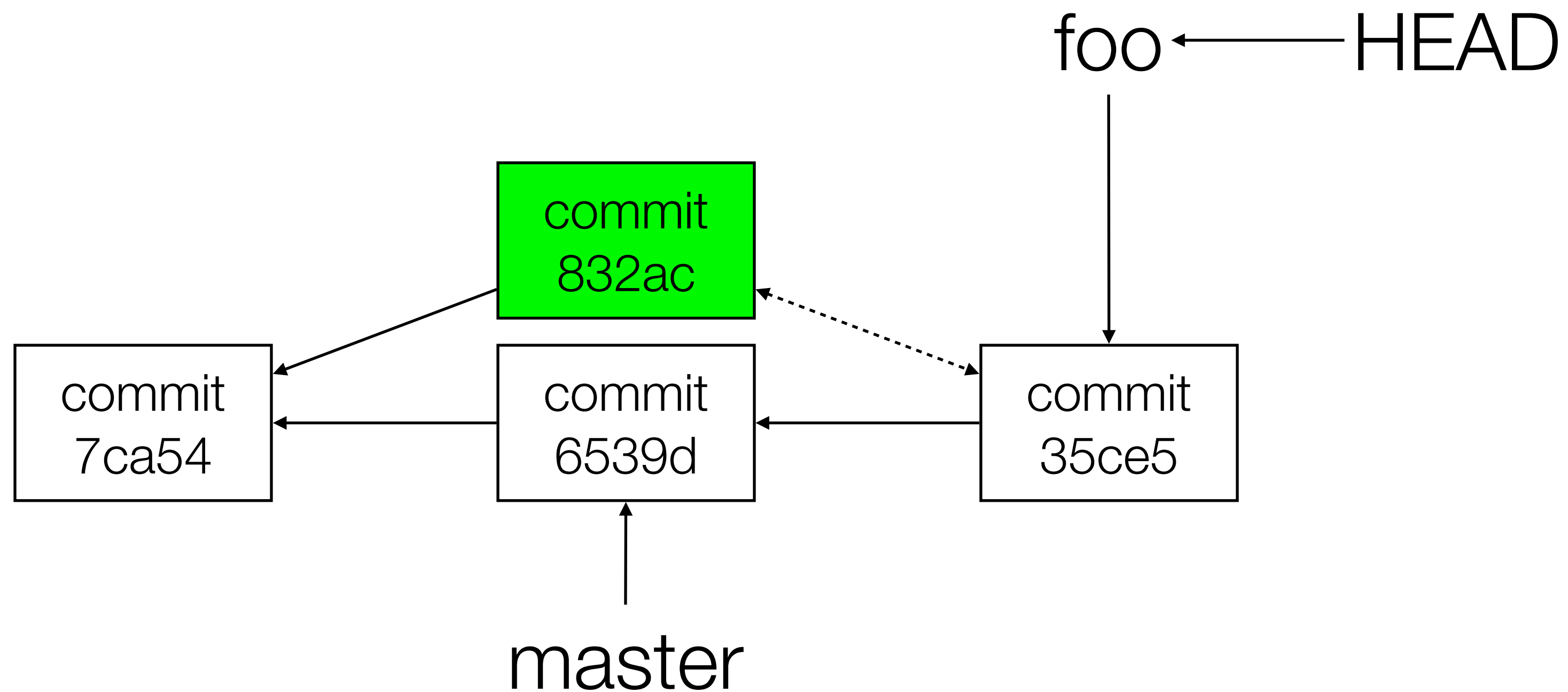
通过 rebase 合并分支

git rebase master



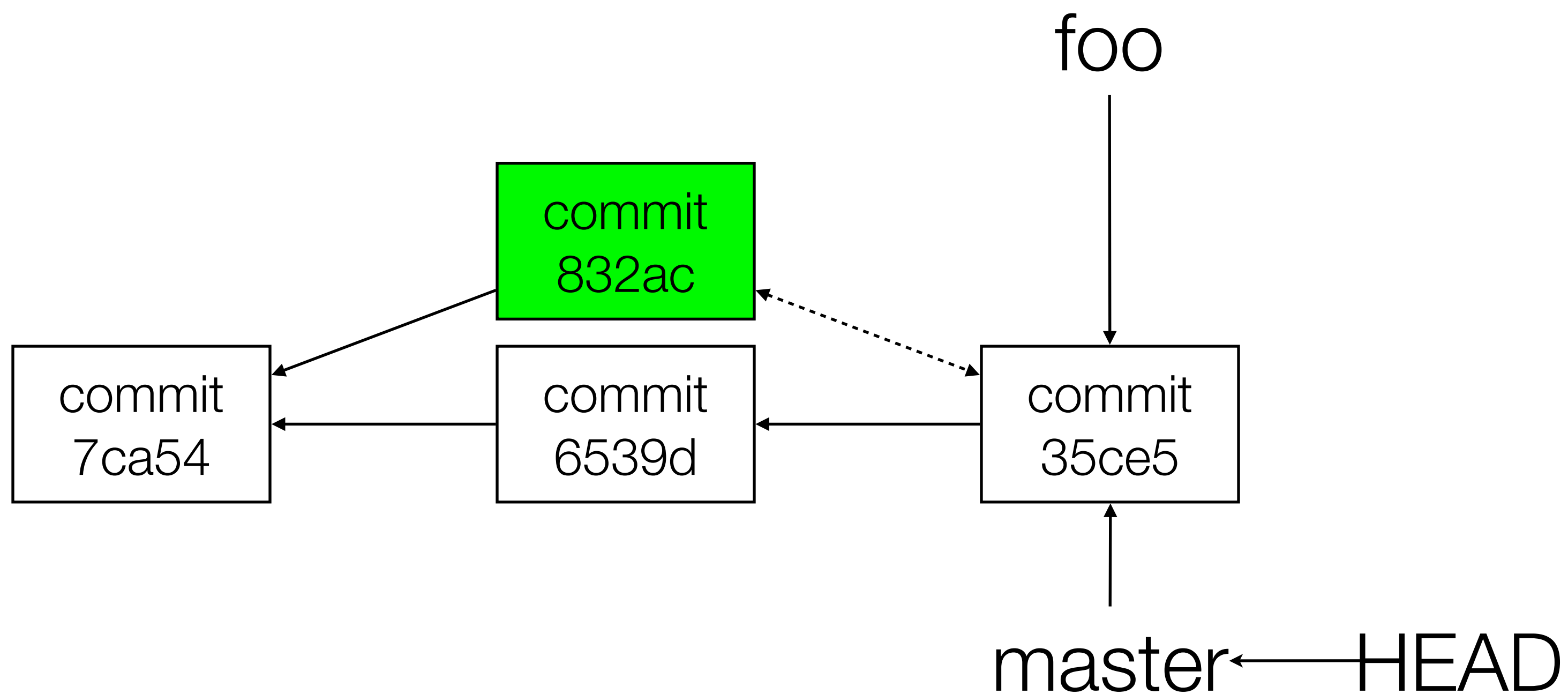
通过 rebase 合并分支

git checkout master; git merge foo



通过 rebase 合并分支

git checkout master; git merge foo



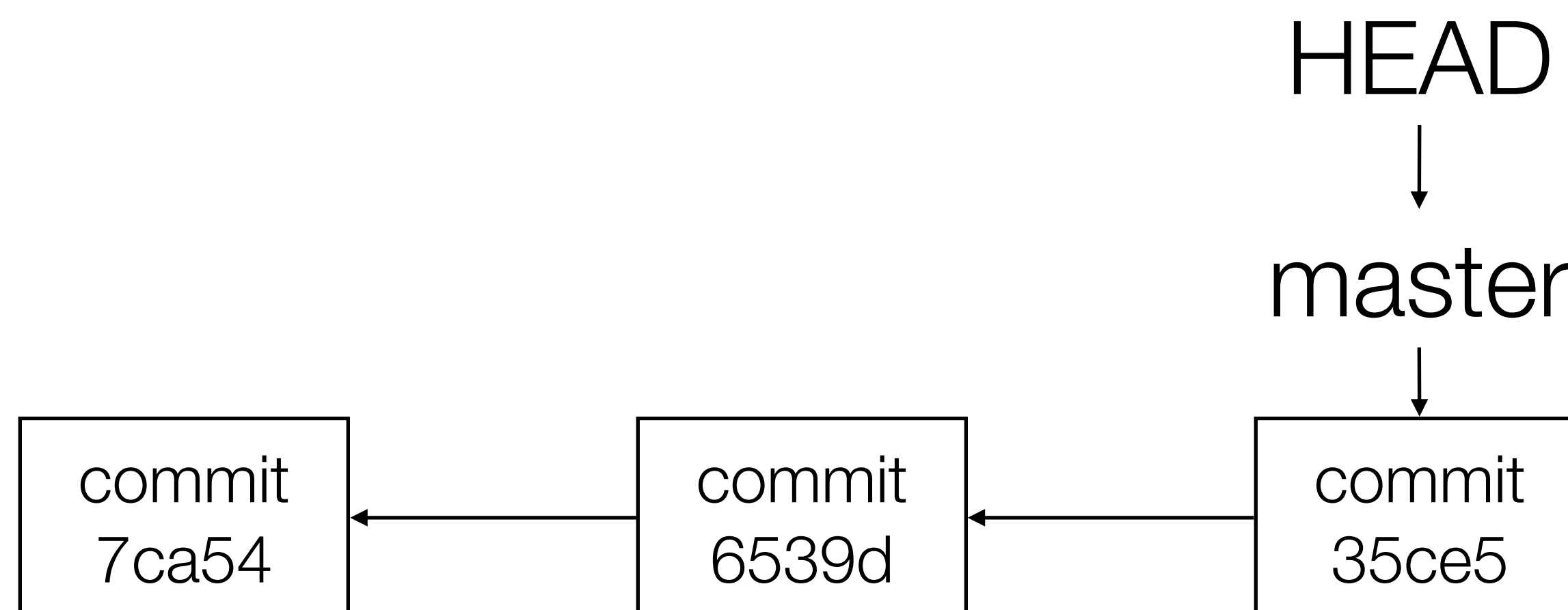
通过 rebase 合并分支

Pros:

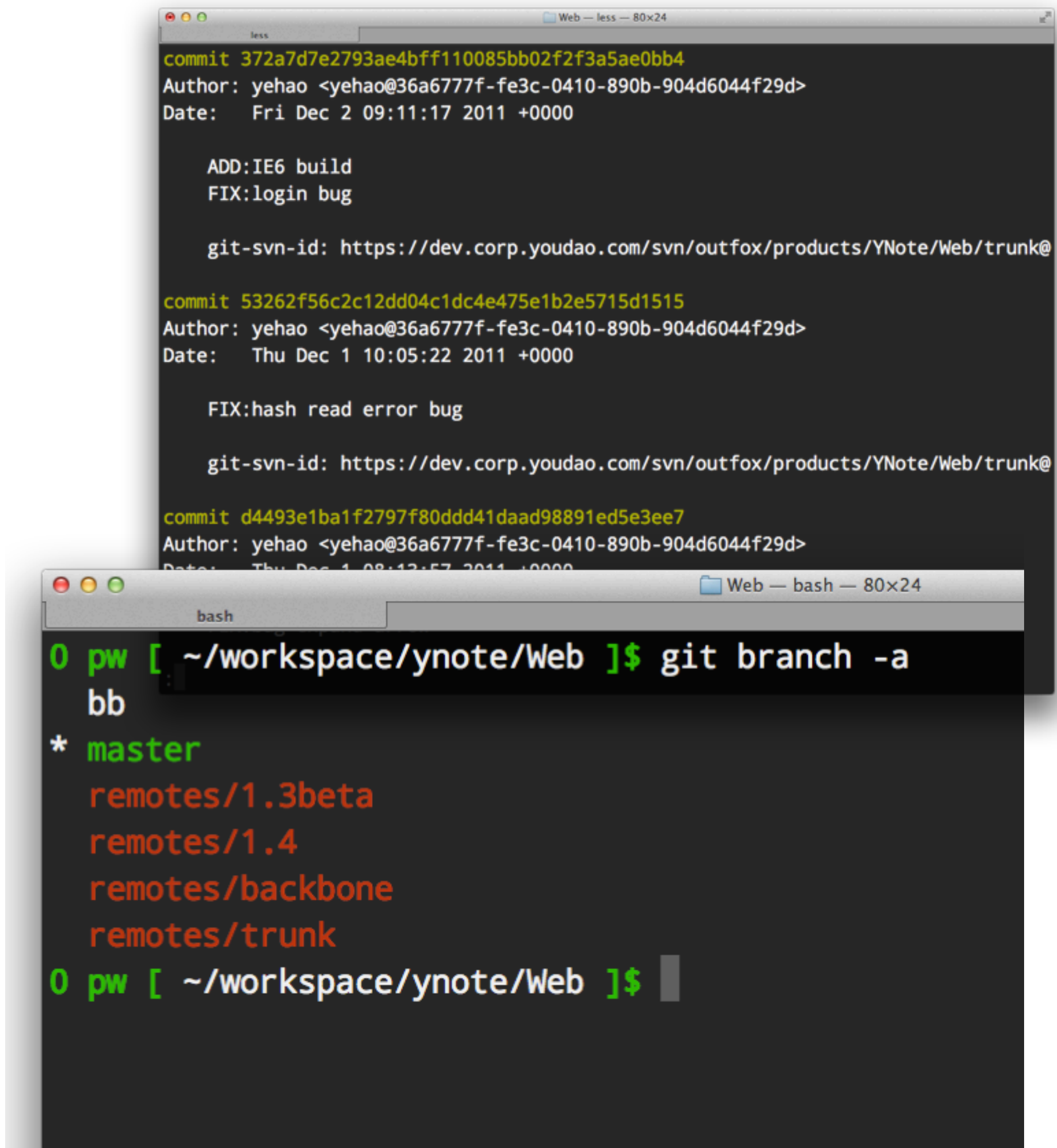
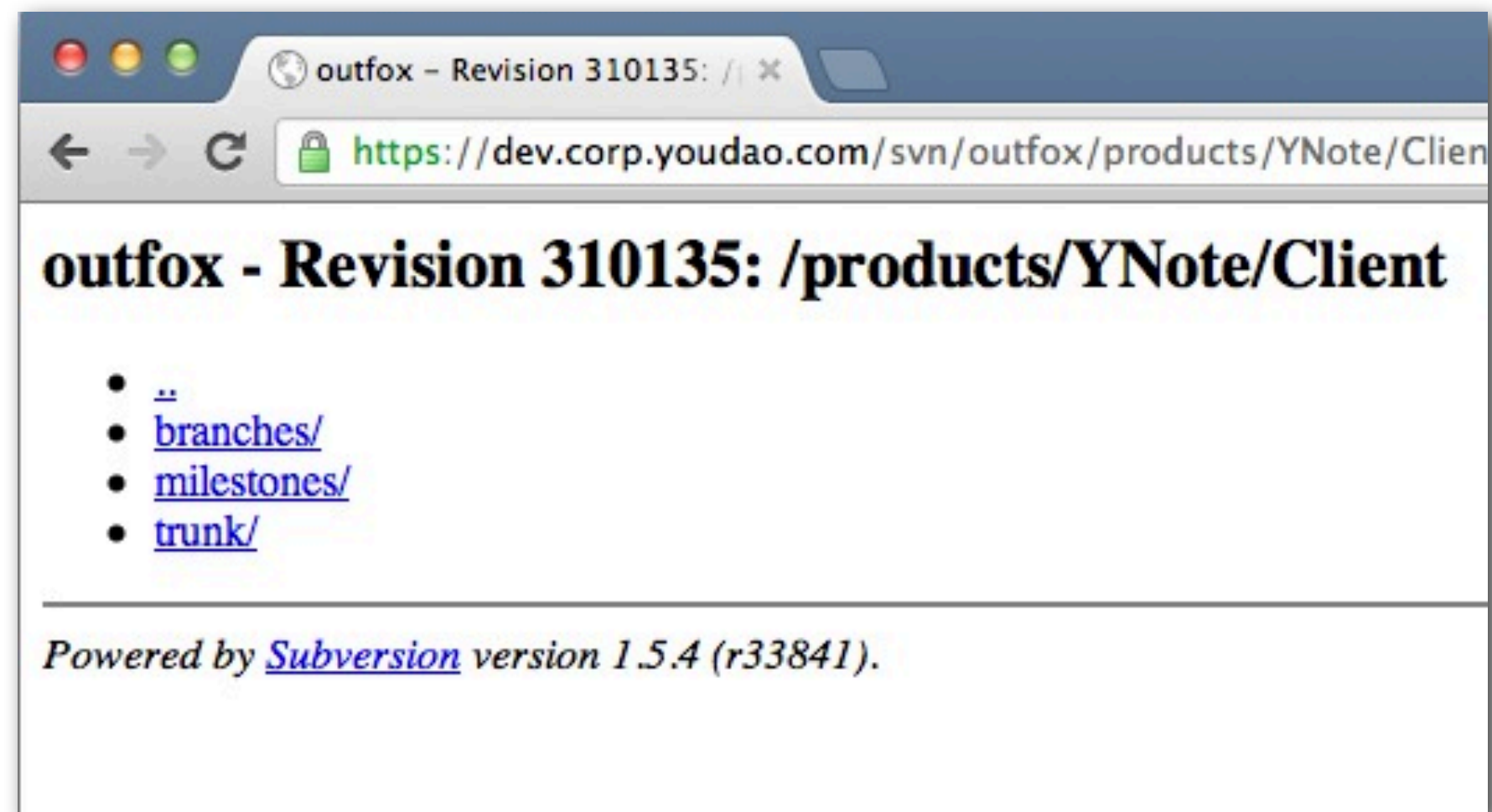
提交记录更简单

Cons:

操作略微复杂



Git + SVN 协同



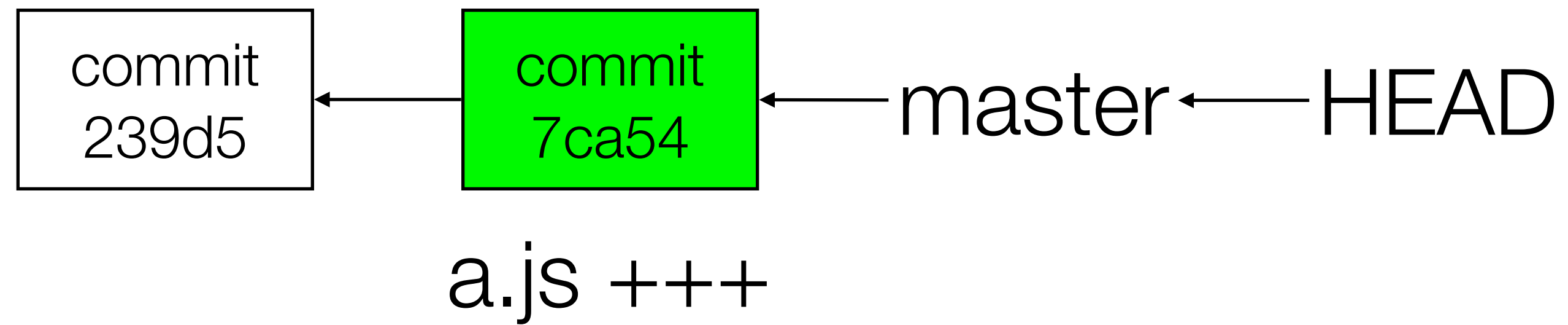
git svn clone -r 310135:HEAD --trunk=trunk --branches=branches --branches=milestones https://..../YNote/Client

git 命令		svn 命令
git pull		svn update
git push		svn commit

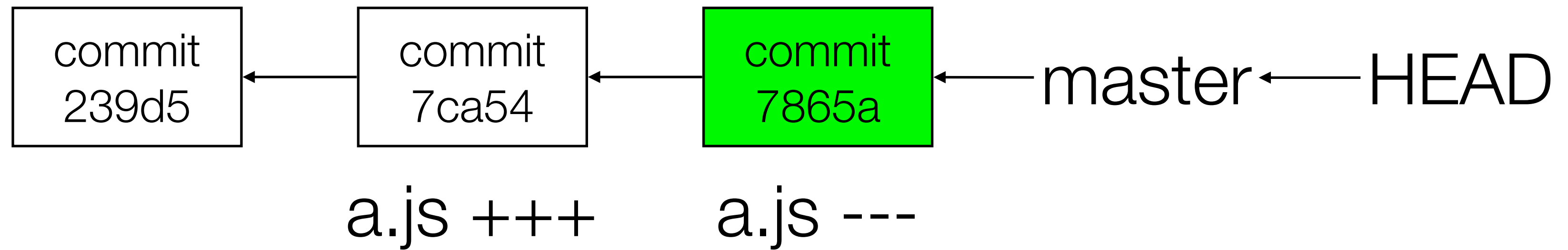
git 命令	git svn 胶水命令	svn 命令
git pull	git svn rebase	svn update
git push	git svn dcommit	svn commit

Git 技巧

Git revert

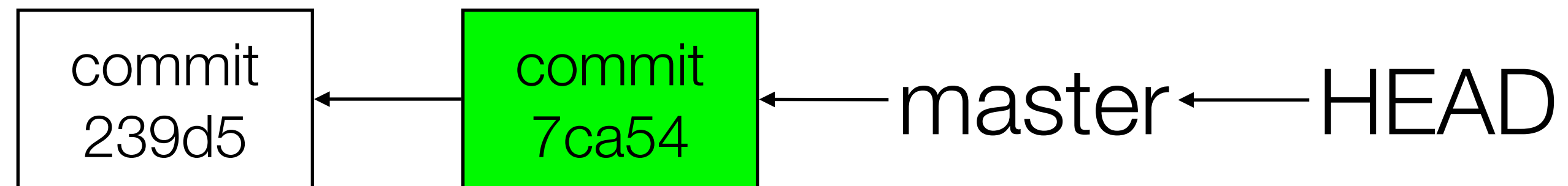


Git revert

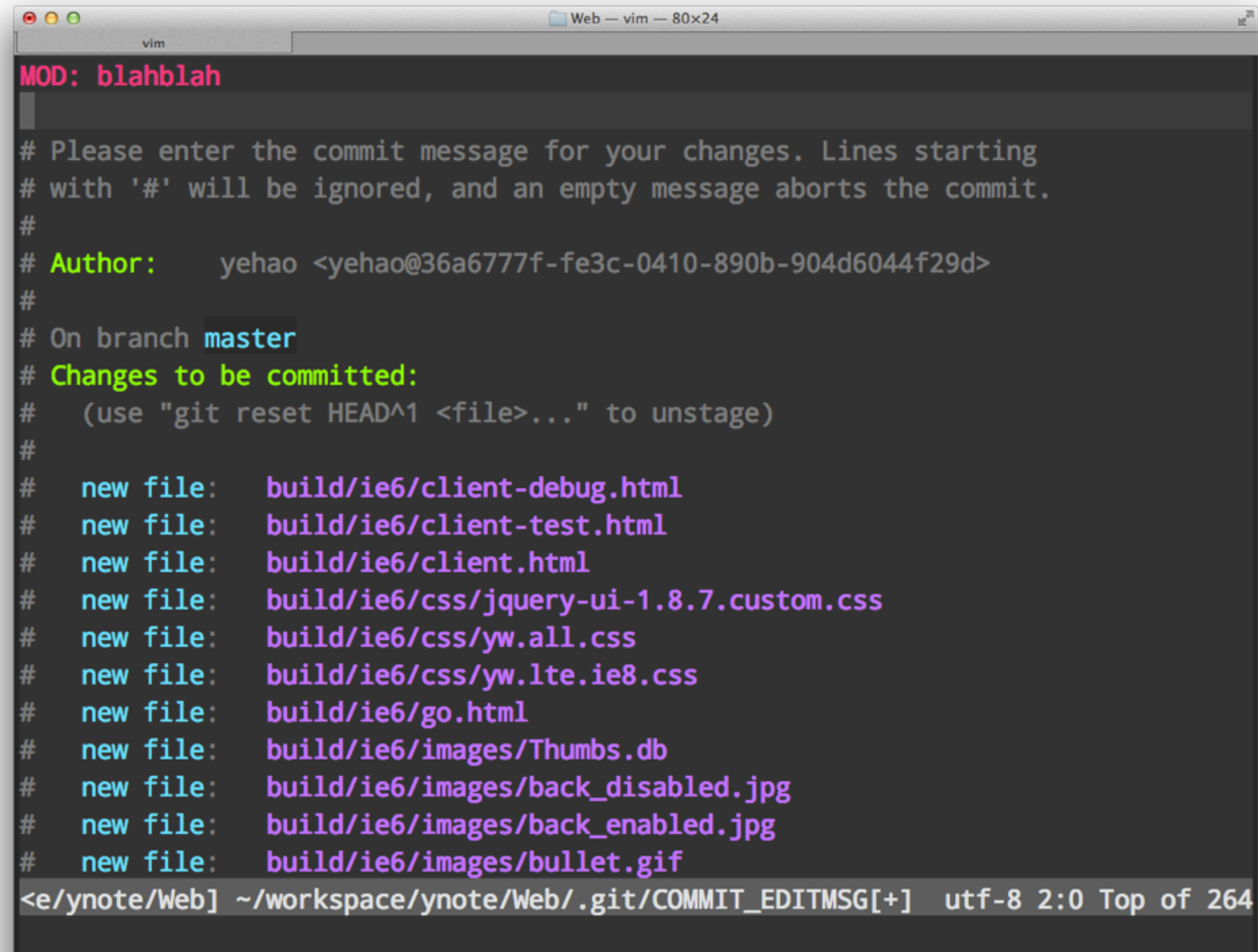


git commit --amend

MOD: blahblah



git commit --amend



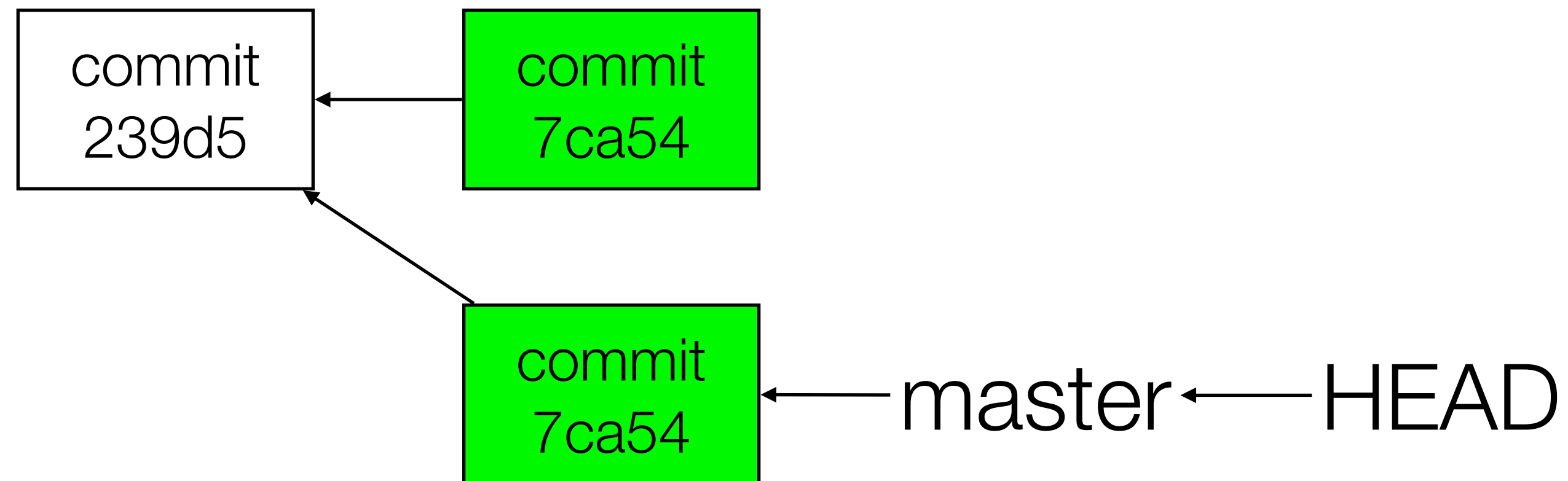
The screenshot shows a vim editor window titled "Web — vim — 80x24". The editor is in the middle of editing a commit message. The first line is "MOD: blahblah" in red. Below it is a series of instructions for entering the commit message, including a warning about lines starting with '#', the author's name and email, the current branch (master), and a list of files to be committed. The list of files includes build/ie6/client-debug.html, build/ie6/client-test.html, build/ie6/client.html, build/ie6/css/jquery-ui-1.8.7.custom.css, build/ie6/css/yw.all.css, build/ie6/css/yw.lte.ie8.css, build/ie6/go.html, build/ie6/images/Thumbs.db, build/ie6/images/back_disabled.jpg, build/ie6/images/back_enabled.jpg, and build/ie6/images/bullet.gif. The bottom status bar shows the file path as <e/ynote/Web], the current directory as ~/workspace/ynote/Web/.git/, the file name as COMMIT_EDITMSG[+], the encoding as utf-8, the line number as 2:0, and the total number of lines as Top of 264.

```
MOD: blahblah

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Author:      yehao <yehao@36a6777f-fe3c-0410-890b-904d6044f29d>
#
# On branch master
# Changes to be committed:
#   (use "git reset HEAD^1 <file>..." to unstage)
#
#   new file:   build/ie6/client-debug.html
#   new file:   build/ie6/client-test.html
#   new file:   build/ie6/client.html
#   new file:   build/ie6/css/jquery-ui-1.8.7.custom.css
#   new file:   build/ie6/css/yw.all.css
#   new file:   build/ie6/css/yw.lte.ie8.css
#   new file:   build/ie6/go.html
#   new file:   build/ie6/images/Thumbs.db
#   new file:   build/ie6/images/back_disabled.jpg
#   new file:   build/ie6/images/back_enabled.jpg
#   new file:   build/ie6/images/bullet.gif
<e/ynote/Web] ~/workspace/ynote/Web/.git/COMMIT_EDITMSG[+]  utf-8 2:0 Top of 264
```

git commit --amend

MOD: blahblah



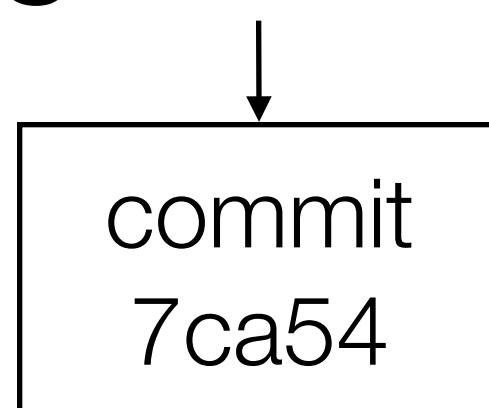
MOD: ...

通过 rebase 合并提交记录

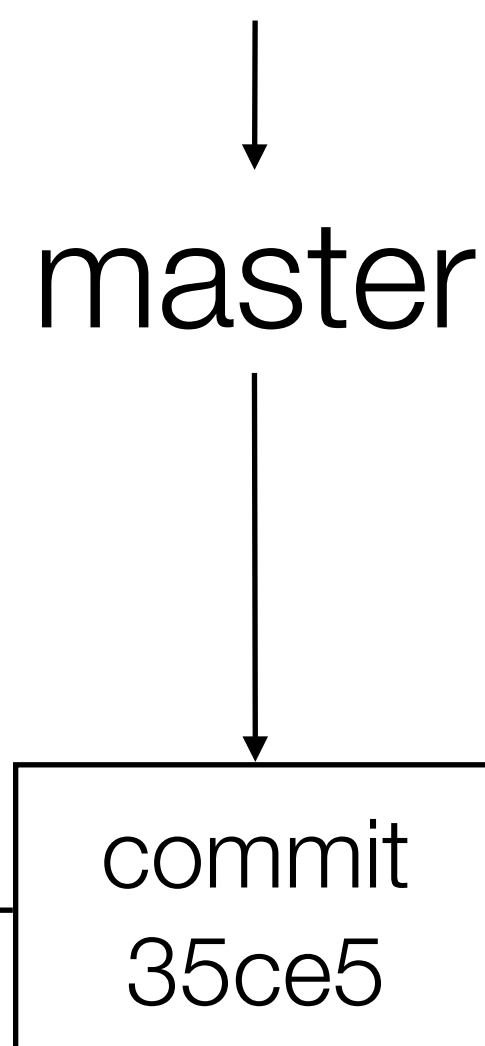
git rebase -i origin/master

-i 进入交互式 rebase

origin/master

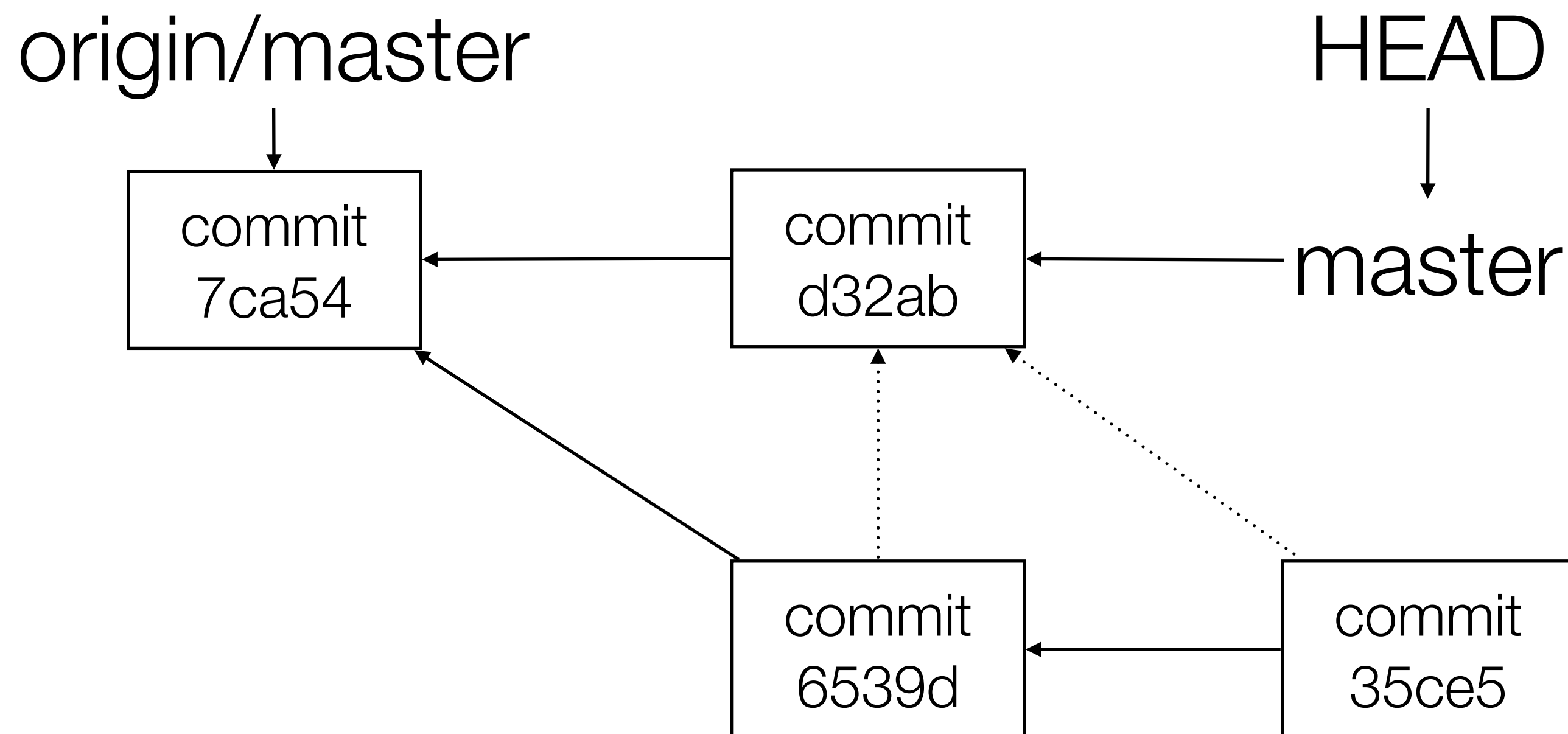


HEAD



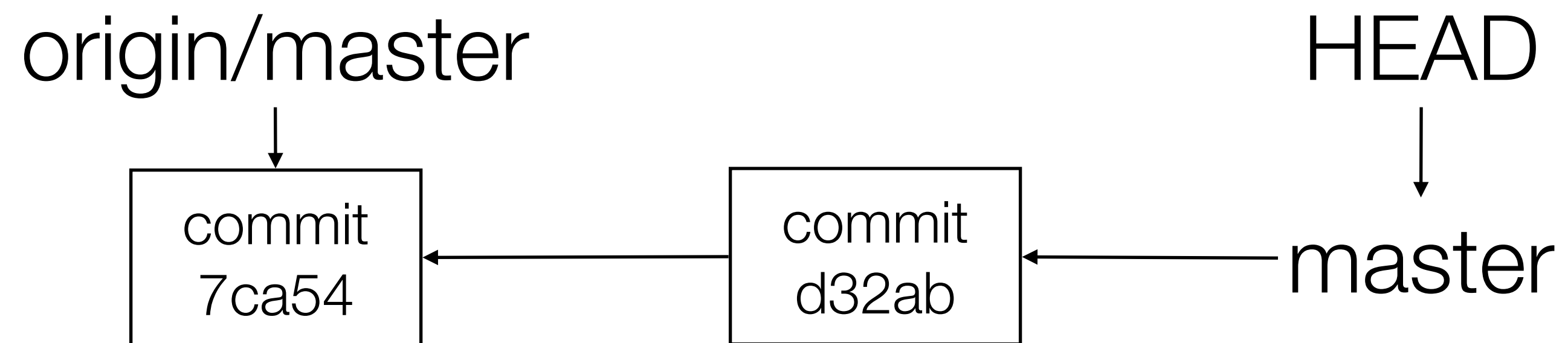
通过 rebase 合并提交记录

`git rebase -i origin/master` # -i 进入交互式 rebase



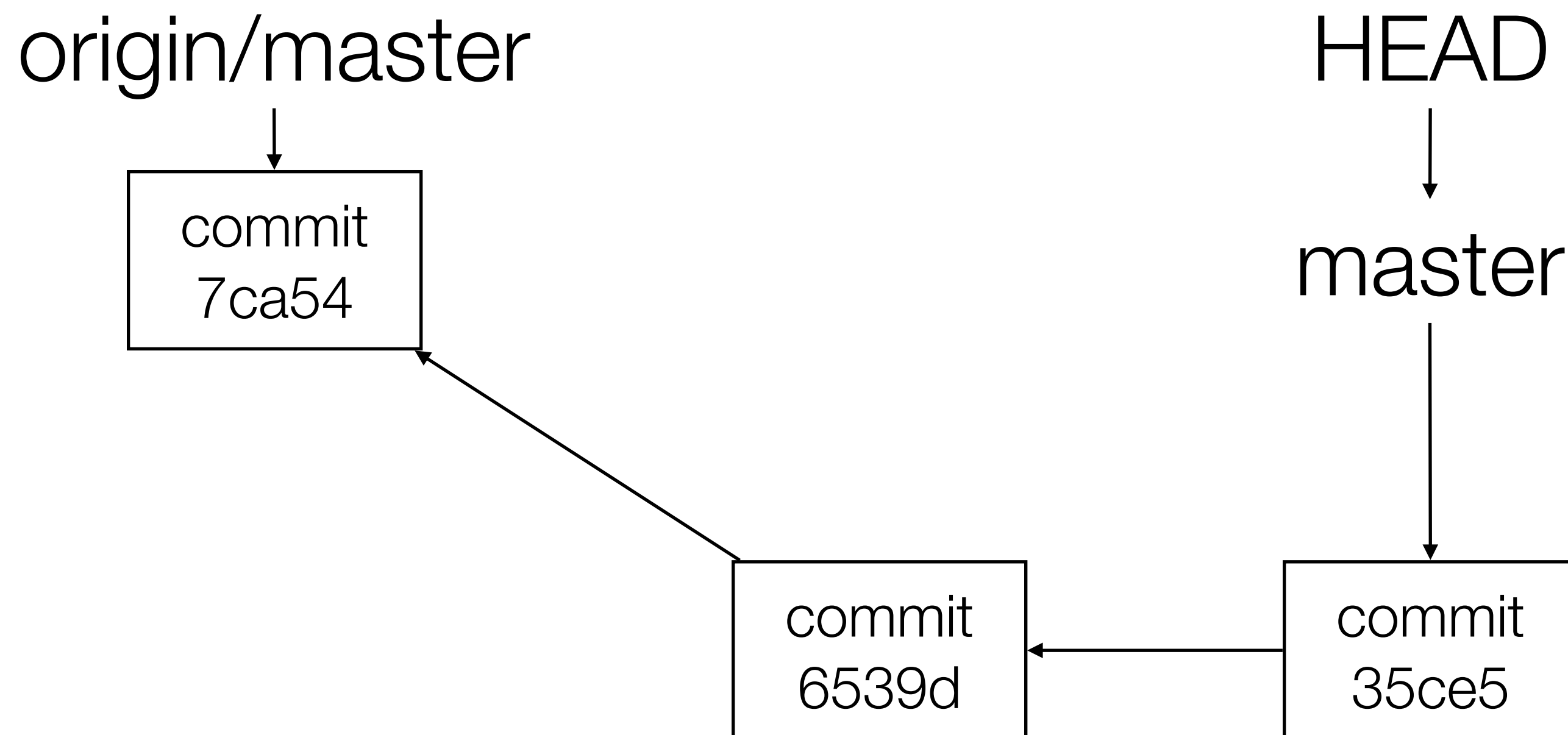
通过 rebase 合并提交记录

`git rebase -i origin/master` # -i 进入交互式 rebase



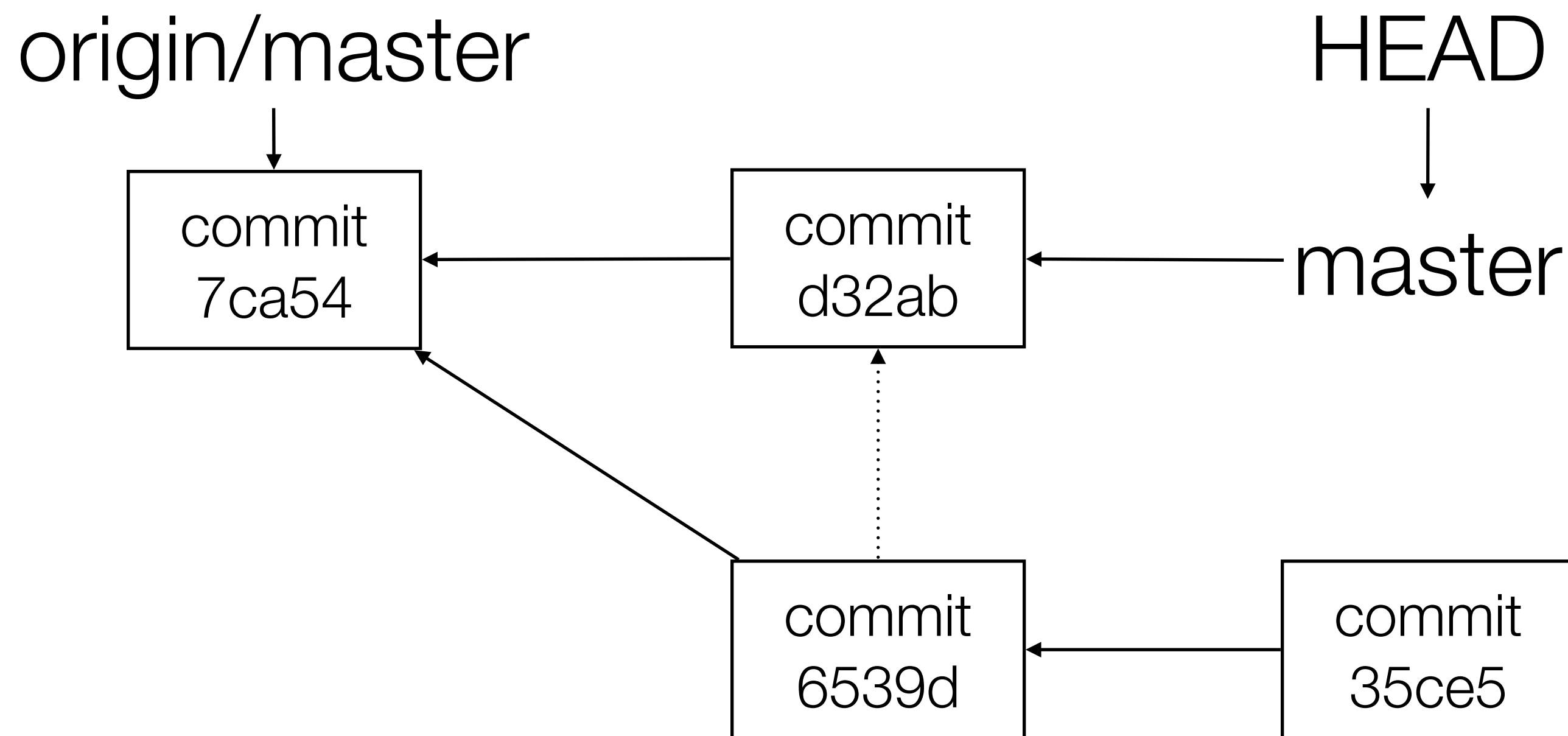
通过 rebase 丢弃提交记录

`git rebase -i origin/master` # -i 进入交互式 rebase



通过 rebase 丢弃提交记录

`git rebase -i origin/master` # -i 进入交互式 rebase



Git rebase tip

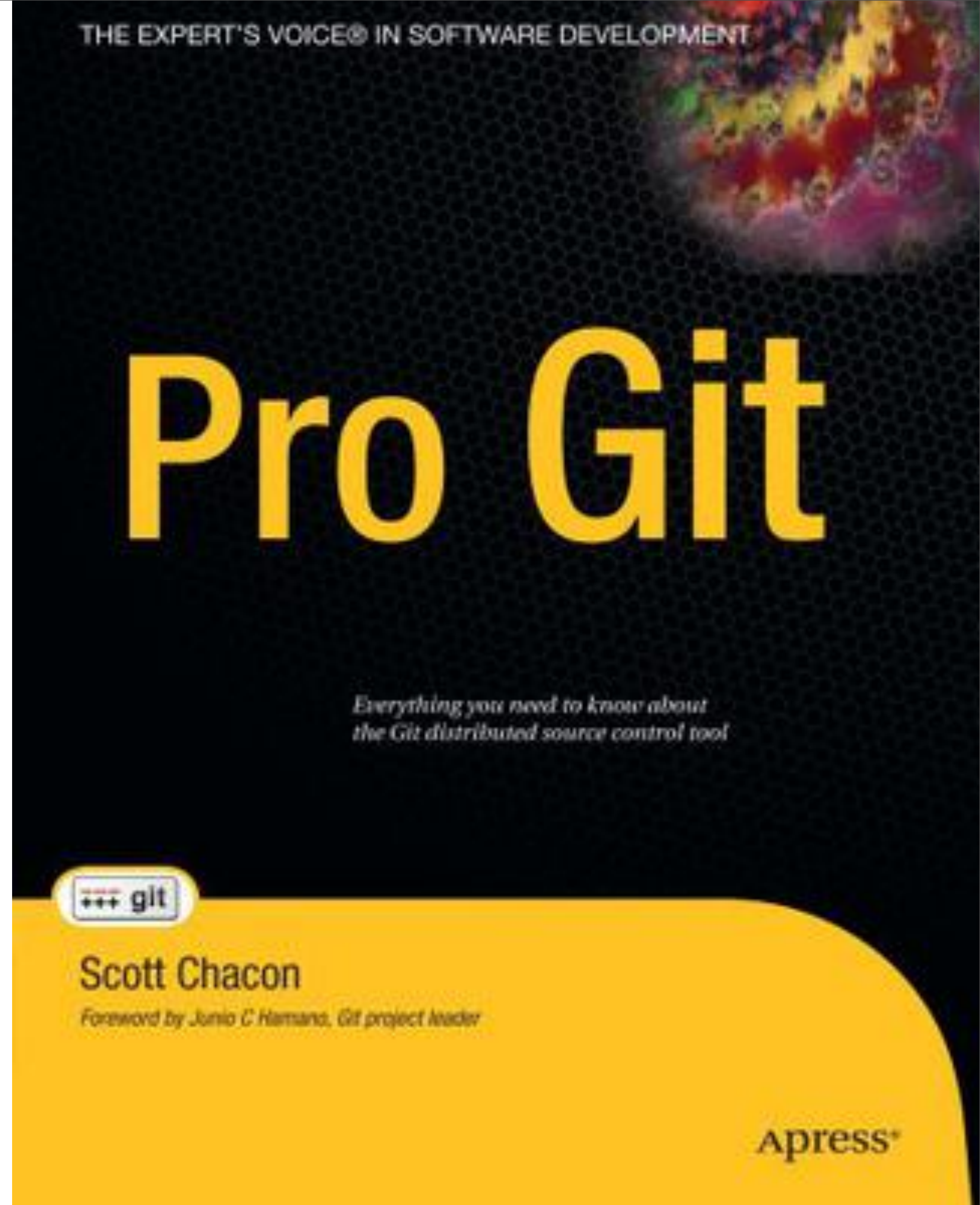
- 文件只要进了 git 的版本库, 就是安全的
- rebase 本质上是基于已有提交构造新提交
- 永远不要 rebase 已经推送到公共仓库的提交

Git 学习资源

Pro git

免费的在线电子书

<http://progit.org>





Git领域的集大成之作，在广度、深度和实践性上均史无前例
国内顶级Git专家亲自撰写，Git官方维护者等数位专家联袂推荐



蒋鑫 著

Got Git! The Definitive Guide of Git

Git 权威指南

机械工业出版社
China Machine Press

Git 权威指南

Q&A

张宇辰 zhangyc@rd.netease.com

git 学习 POPO 群: 1195049