JAVA 达摩班

Git & Github最佳实践

季蠡
Ashton

25 March, 2018

详解Git与版本控制

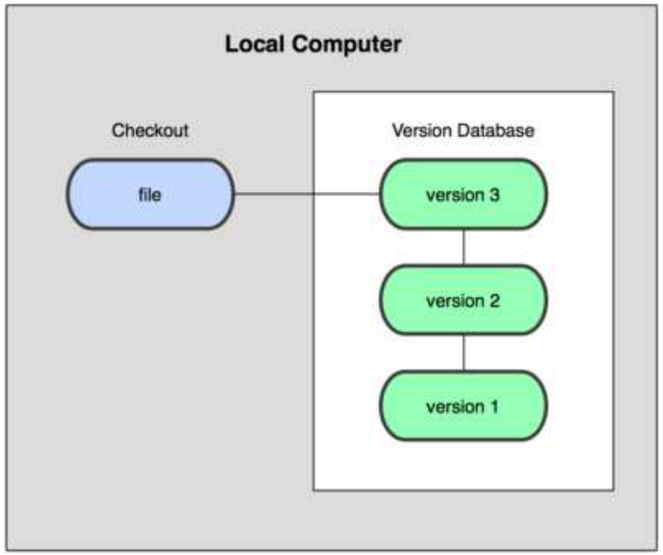
1

版本控制

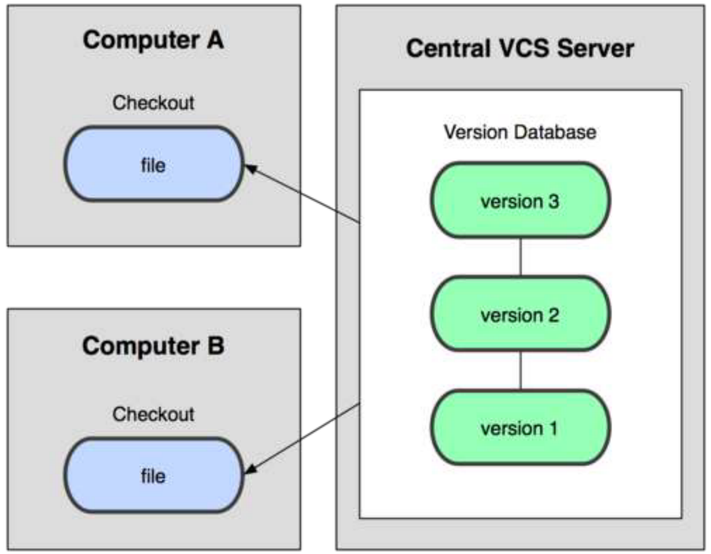
版本控制（version control system，VCS）是一个软件系统，它记录文件变化，或者设置时间点上的软件版本，这样就可以回看历史版本

VCS类型：

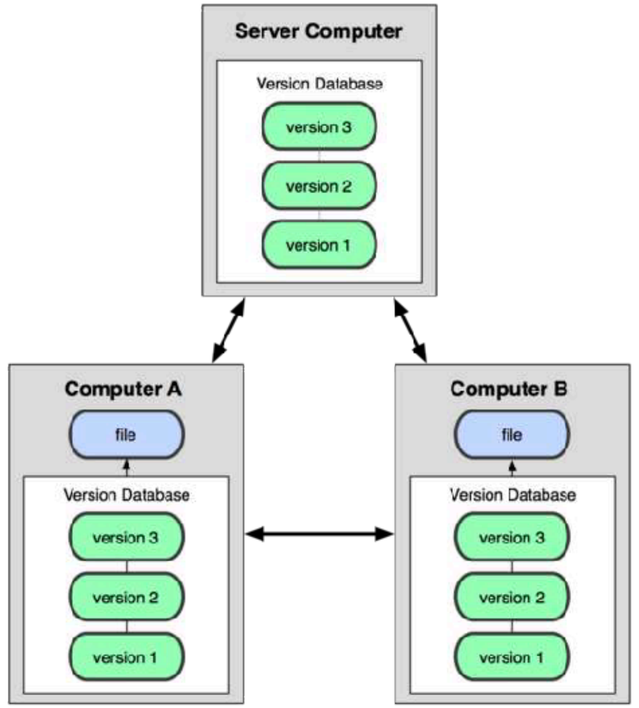
1. 本地VCS



2. 中央VCS



3. 分布式VCS



为什么Git

英文解释：饭桶，笨蛋，无用的人

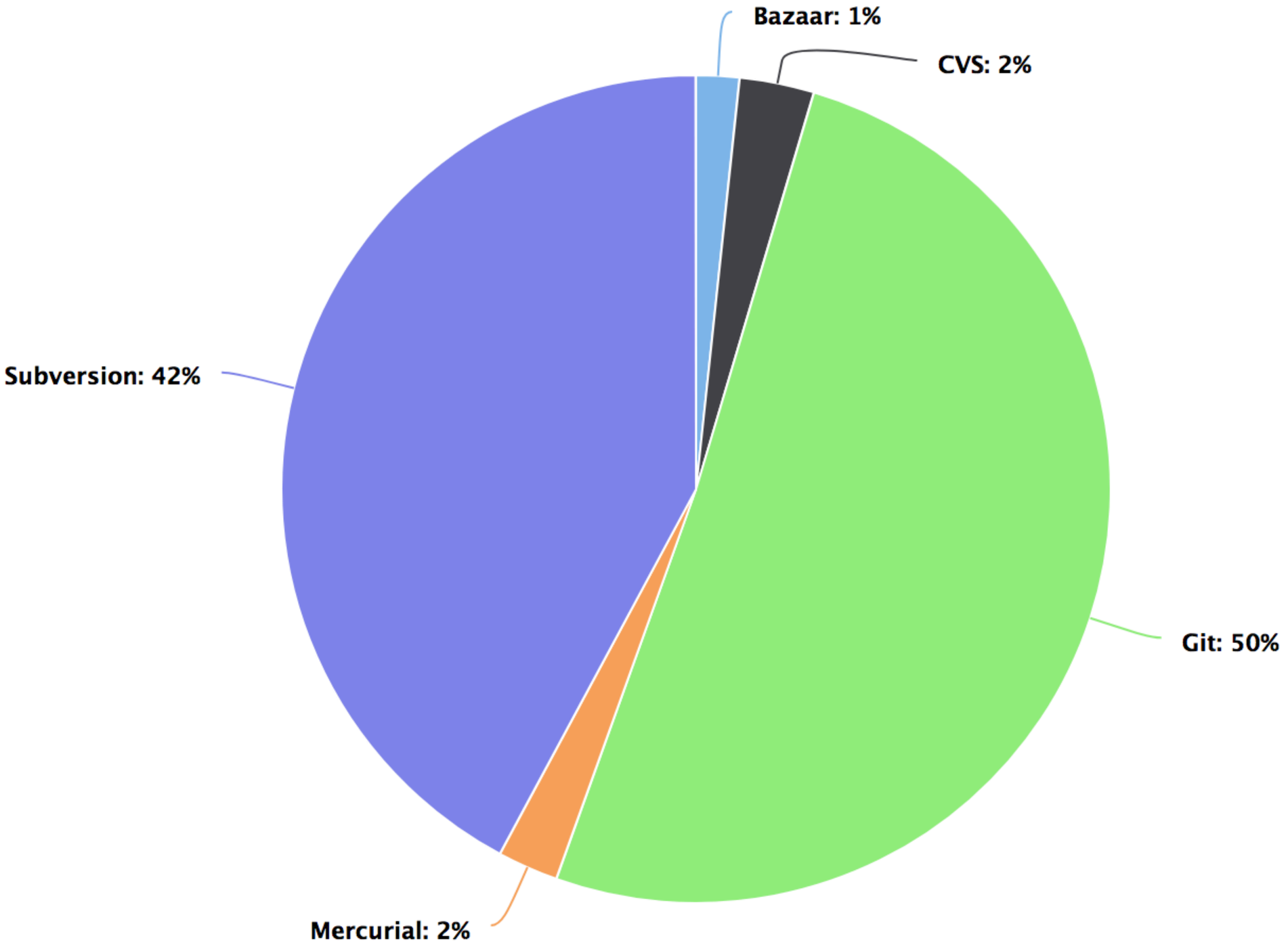
命名源自于Linus的自嘲

“I'm an egotistical bastard, so I name all my projects after myself. First Linux, now Git”

Linus Torvalds, PC World. 2012-07-14

源于2005年，Linux需要版本管理工具，Kernel 2.6.12第一次使用Git

Git 1.0发布于2005年12月


Bazaar: 1%
CVS: 2%
Subversion: 42%
Git: 50%
Mercurial: 2%

Git核心概念

快照（Snapshot）

- 用于追踪文件历史
- 记录文件在某个时间点的状态
- 你来决定什么时间，对哪些文件“拍照”
- 你可以回看之前的快照

Git核心概念

提交（Commit）

- 提交就是创建快照的动作
- 既是动词也是名词
- 项目可以看作是有提交组成的
- 提交包括三部分：
  1. 文件如何变化的信息
  2. 创建对该提交的引用指向“父提交”
  3. hash值

Git核心概念

代码仓库（Repository）

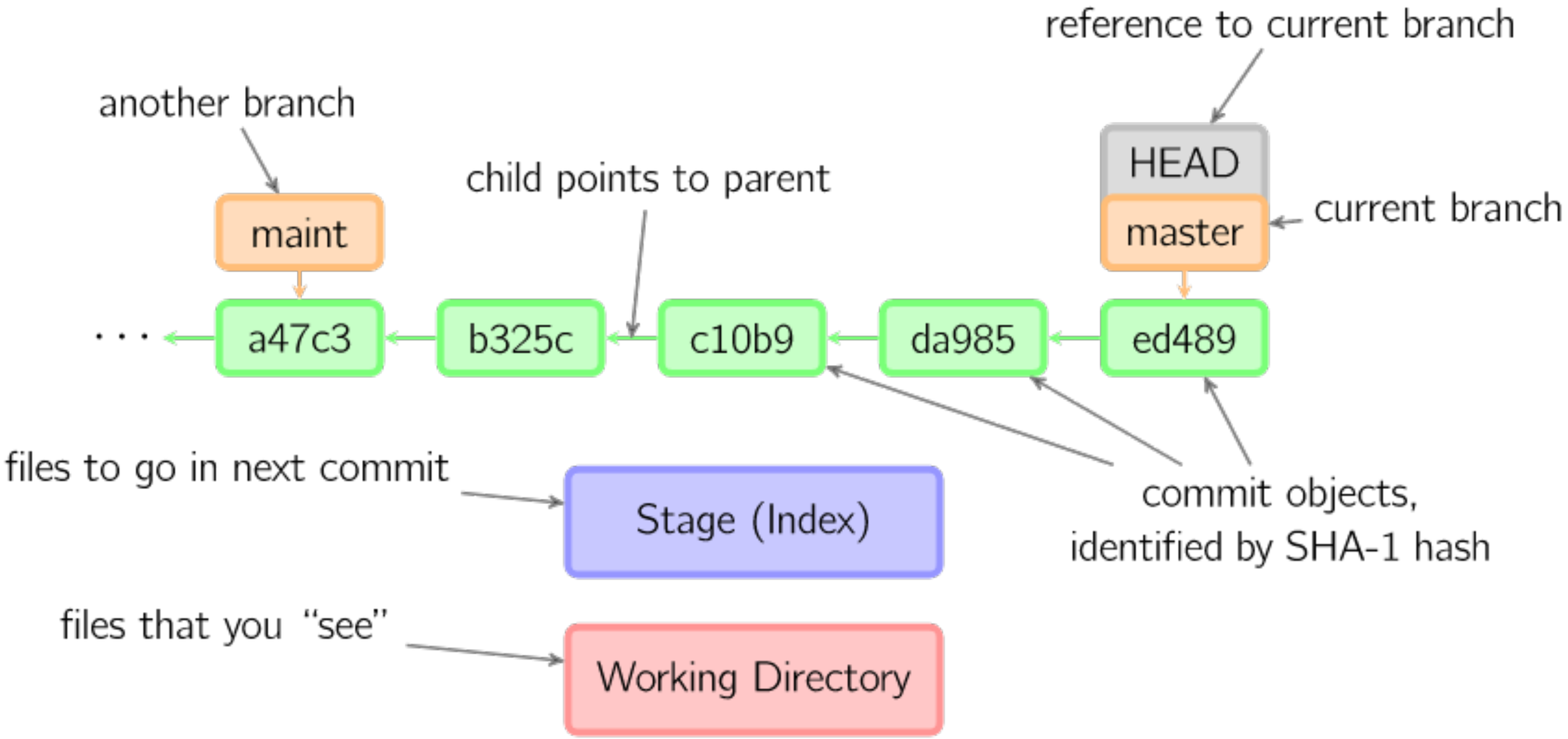
- 简称 repo
- 包括所有代码文件及其历史记录
- 包括所有的提交
- 可以在本地，也可以在远程（Github）
- 将远程repo复制到本地叫做克隆clone
- 团队协作
- 同步本地没有的commit叫做pull
- 同本远程没有的commit叫做push

Git核心概念

分支（Branch）

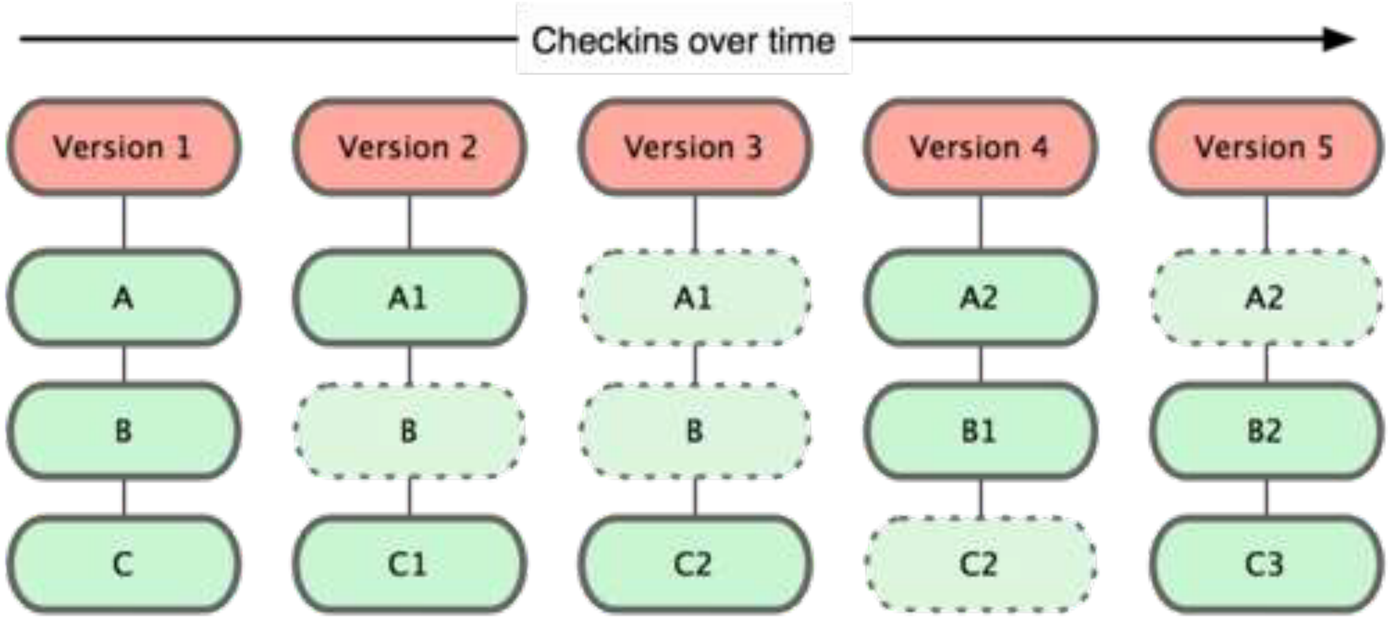
- 所有提交都发生在分支上
- 可以有很多分支
- 默认创建的主分支叫做master branch

Git核心概念


reference to current branch
another branch
HEAD
child points to parent
maint
master
current branch
· · ·
a47c3
b325c
c10b9
da985
ed489
files to go in next commit
Stage (Index)
commit objects,
identified by SHA-1 hash
files that you "see"
Working Directory

Git原理

基于快照，而不是记录文件变化
Git是分布式版本控制系统（DVCS）
Git中的数据是快照集合，它一个小型文件系统。每次提交，它会“拍照”记录当时的文件，并存储索引指向那组快照。



Git原理

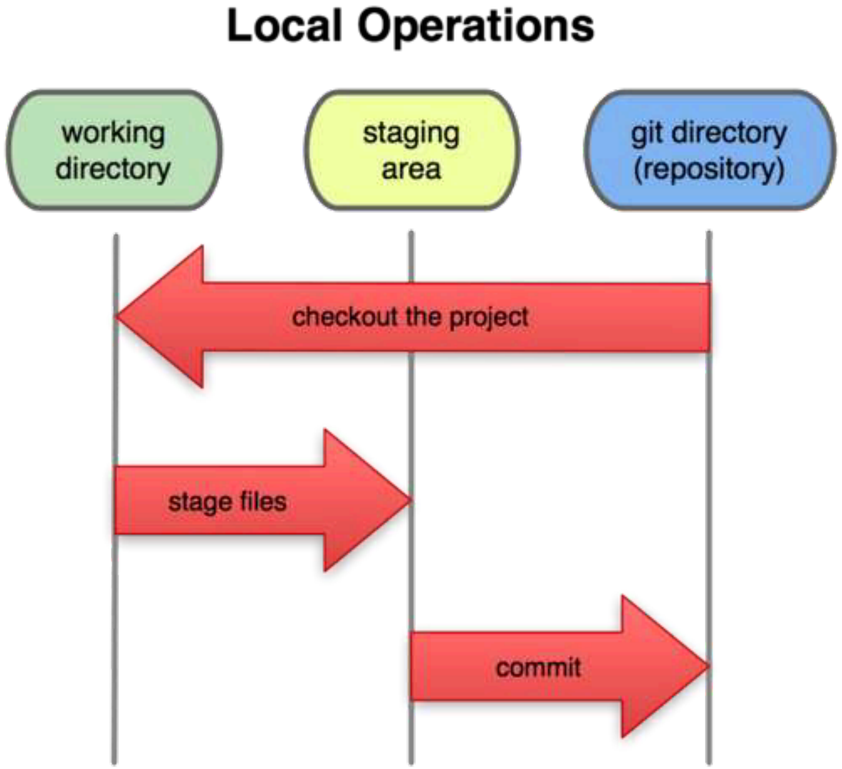
大部分操作都发生在本地文件和资源

Git有三种文件状态：提交(committed)，修改(modified)和暂存(staged)

提交是指数据已经安全的存储在本地数据库。

修改是指改变文件，但没有提交到本地数据库。

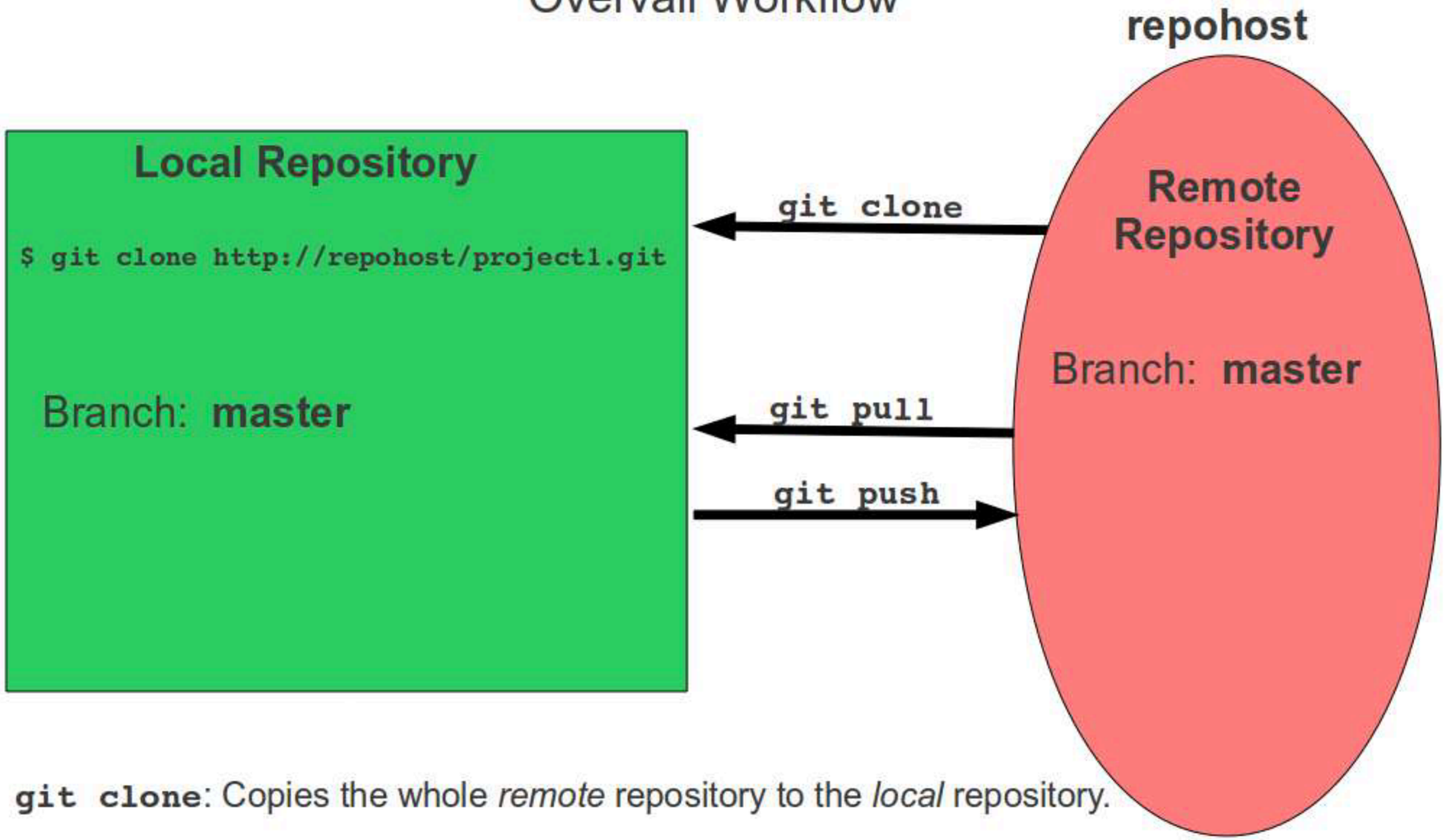
暂存是指对修改的文件做了标记，暂存准备下次提交



Git工作流程

1. 在工作目录中修改文件
2. 对修改文件添加快照，放入暂存区
3. 提交修改，暂存区域会讲快照同步到远程Git目录

Git Remote Repositories: The High-level (“10,000 foot”) View: Overvall Workflow



`git clone`: Copies the whole *remote* repository to the *local* repository.

`git pull`: Retrieves any updates from the remote repository that aren't yet in the local repository and merges them into the local repository.

`git push`: Publishes updates from the local repository to the remote repository

Git产品比较

Github

- 最流行的Git产品
- 网页 + CLI + 桌面App
- 免费 + 企业（付费）
- 平台集成（e.g. JIRA，Jenkins，Trello等）
- Gist
- 静态页面（个人博客，产品展示）

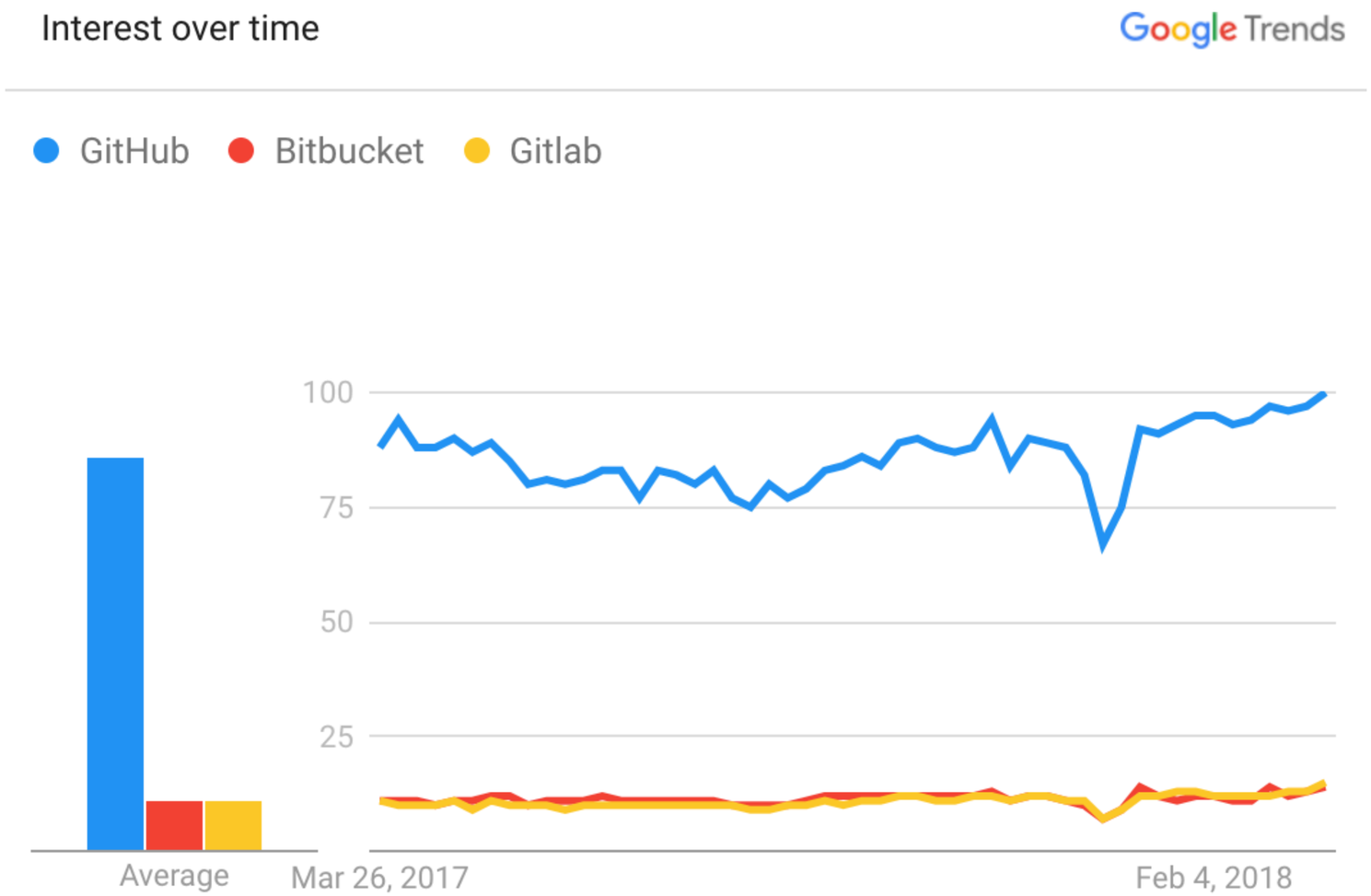
Gitlab

- 提供和Github几乎一摸一样的功能
- 社区版 + 企业版，社区版提供更多免费功能
- 价格比Github便宜
- Gitlab可以装在任何地方（云平台，虚拟机，各种系统）

Bitbucket

- Atlassian产品之一
- 支持Git和Mercurial
- 免费 + 商业版（免费可以拥有private repo）
- Gitlab可以装在任何地方（云平台，虚拟机，各种系统）

Git产品比较


Interest over time
Google Trends
GitHub
Bitbucket
Gitlab
100
75
50
25
Average
Mar 26, 2017
Feb 4, 2018
Worldwide. Past 12 months. Web Search.

Git/Github 安装与配置 2

Mac Git安装和配置

1. 安装

手动　https://git-scm.com/downloads

自动　brew install git

GUI工具

https://git-scm.com/download/gui/mac

https://desktop.github.com/

2. 检测安装

git version

Mac Git安装和配置

3. git设置

git config --global user.name “commit-name“

git config --global user.email “commit-email@example.com”
将邮件添加到自己github账户下 https://github.com/settings/emails

4. Clone选项

官方推荐HTTPS

第一个Github项目

1. 注册 https://github.com/

2. 创建项目（代码仓库） https://github.com/new

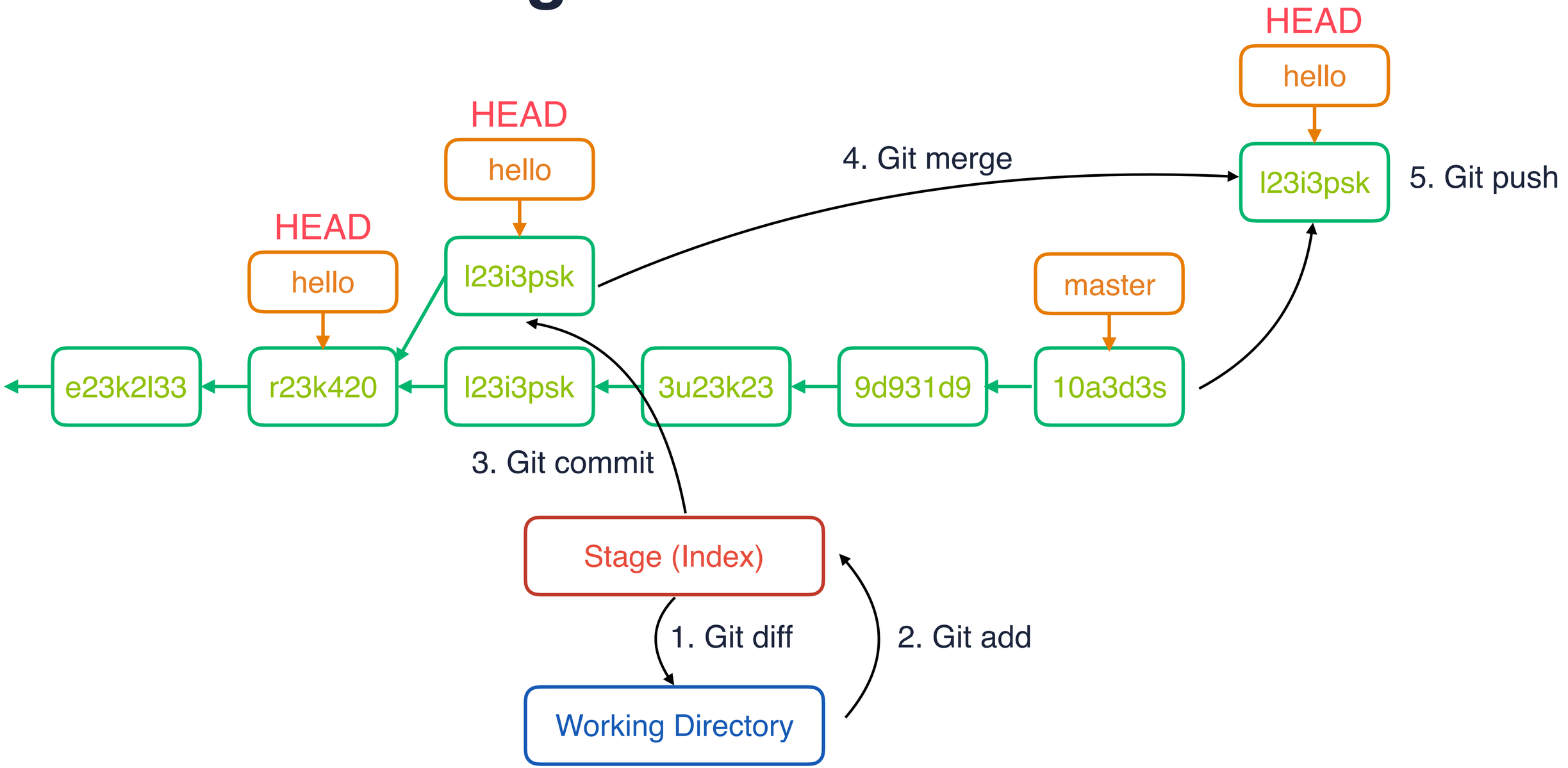
3. 本地创建项目

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin <repo_url>
git push -u origin master
```

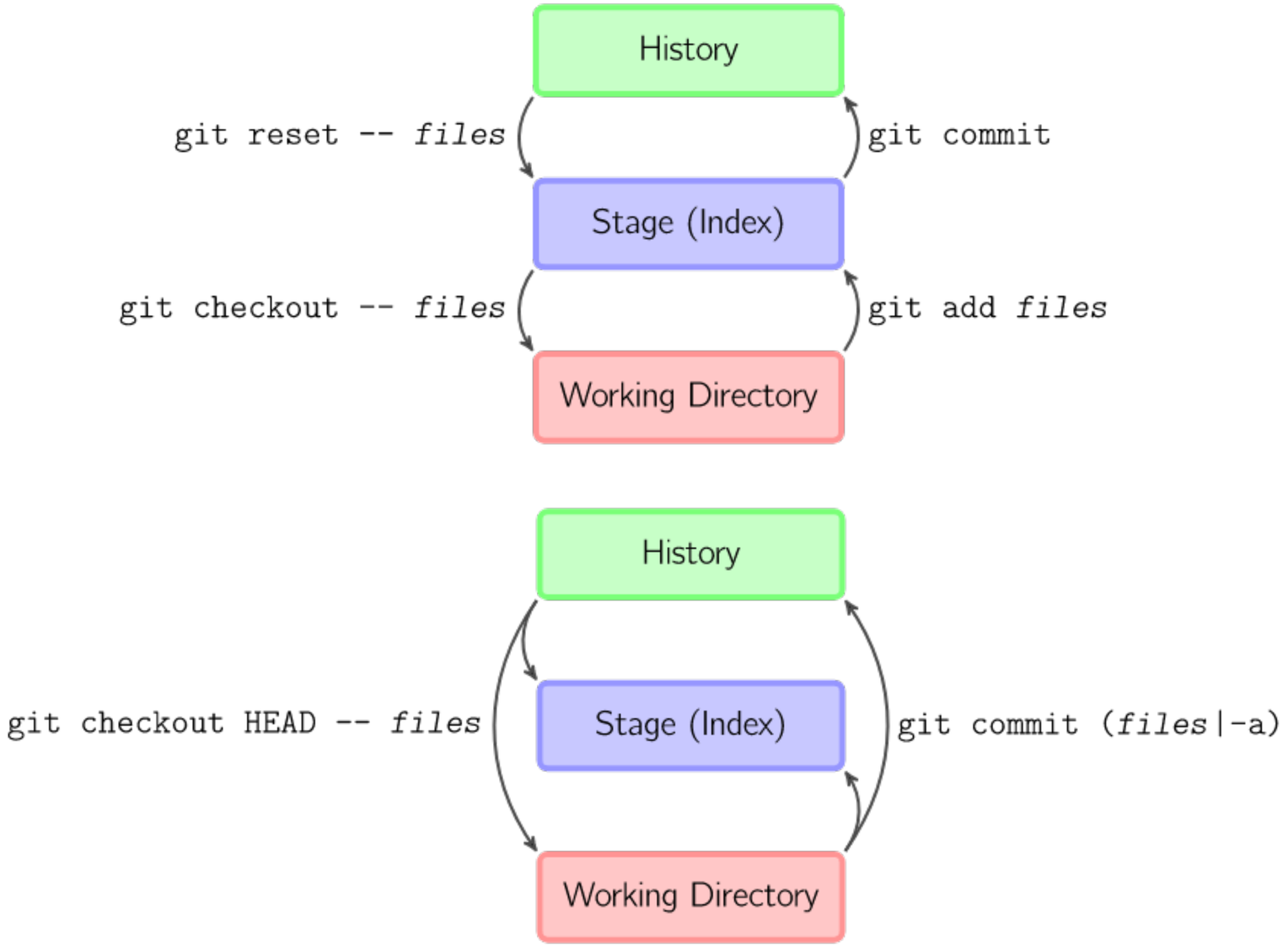
我们的账号 https://github.com/impredator

Git 命令

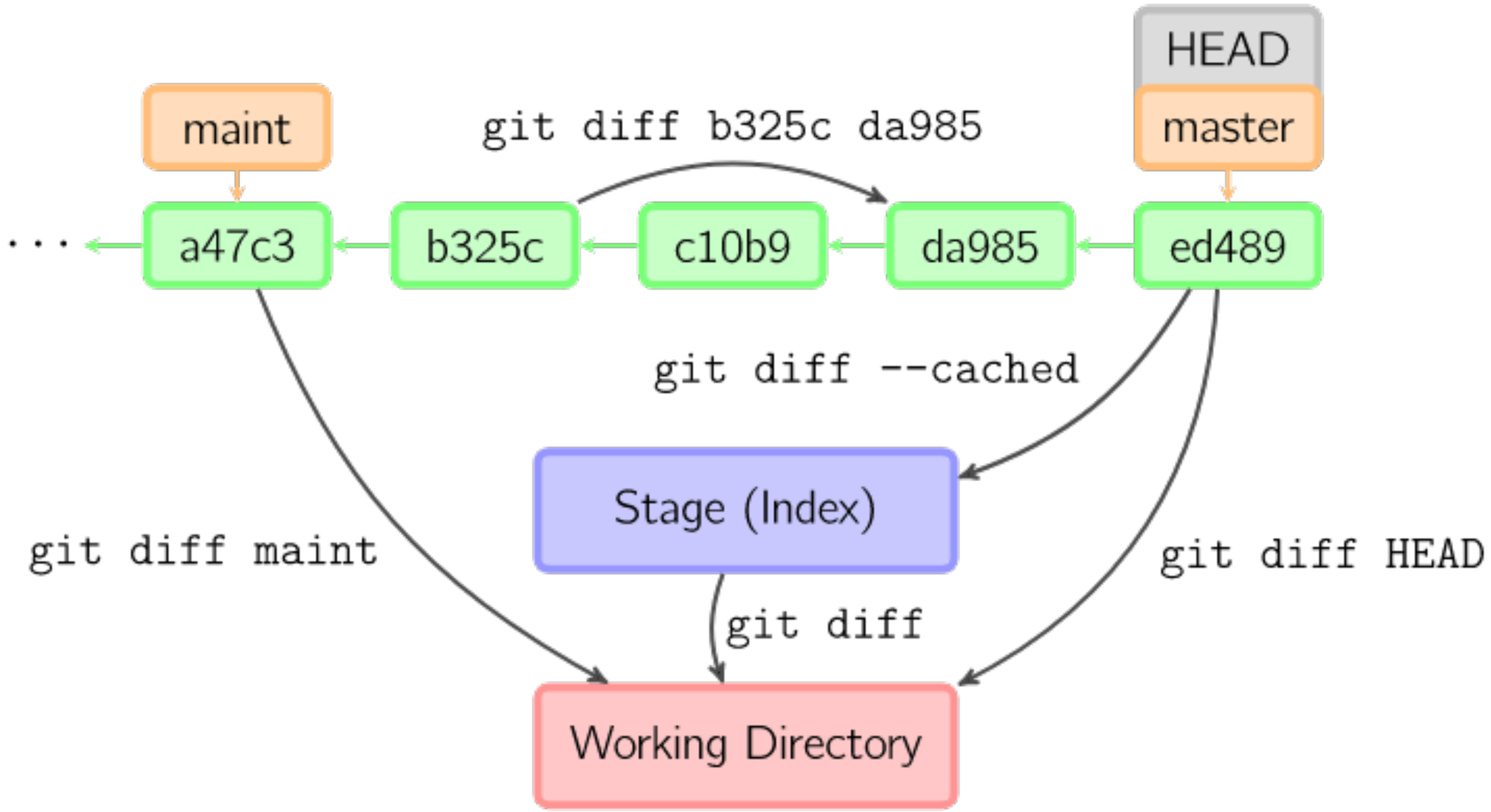
git workflow



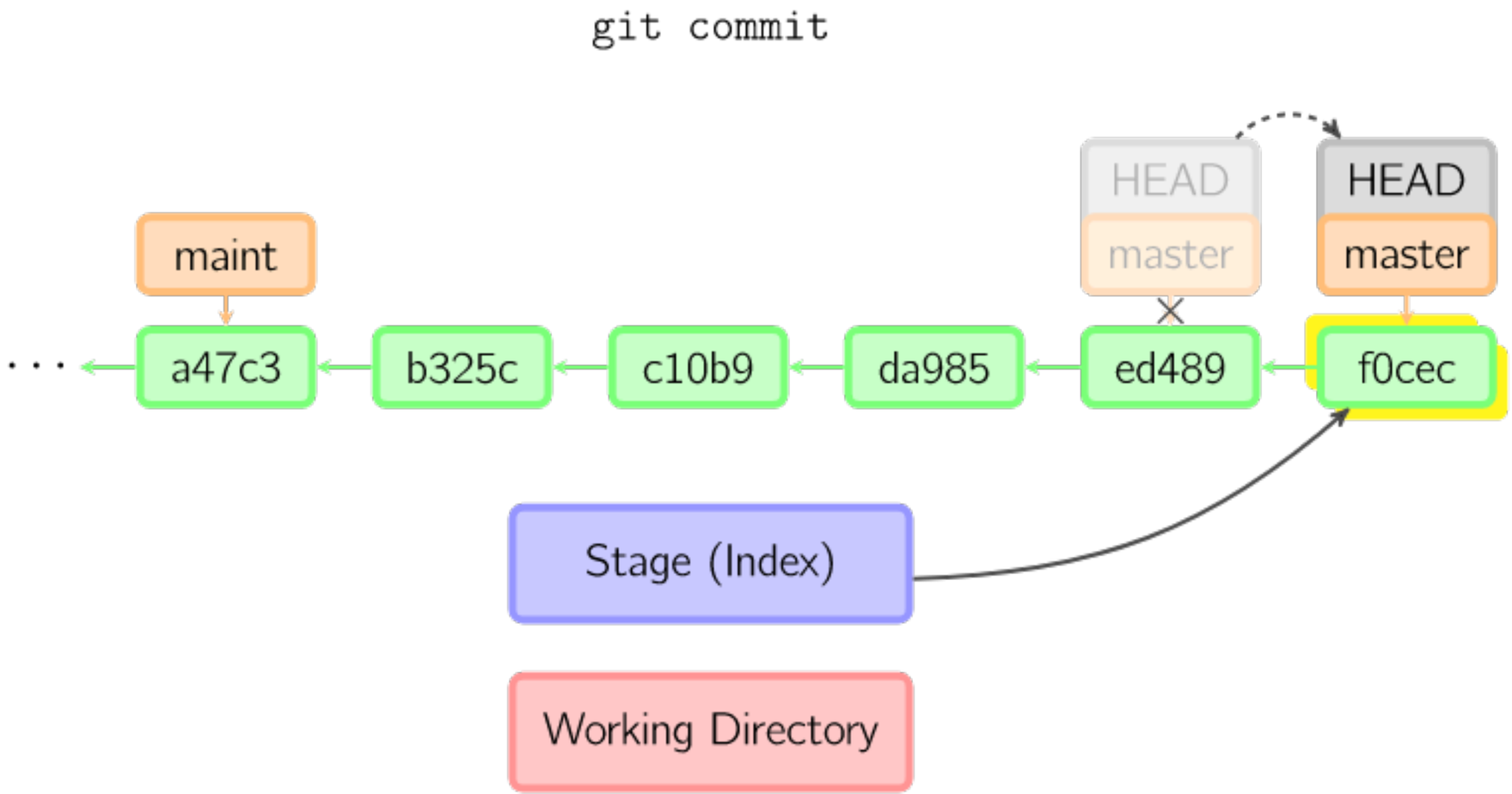
git add


History
git reset -- files
git commit
Stage (Index)
git checkout -- files
git add files
Working Directory
History
git checkout HEAD -- files
Stage (Index)
git commit (files|-a)
Working Directory

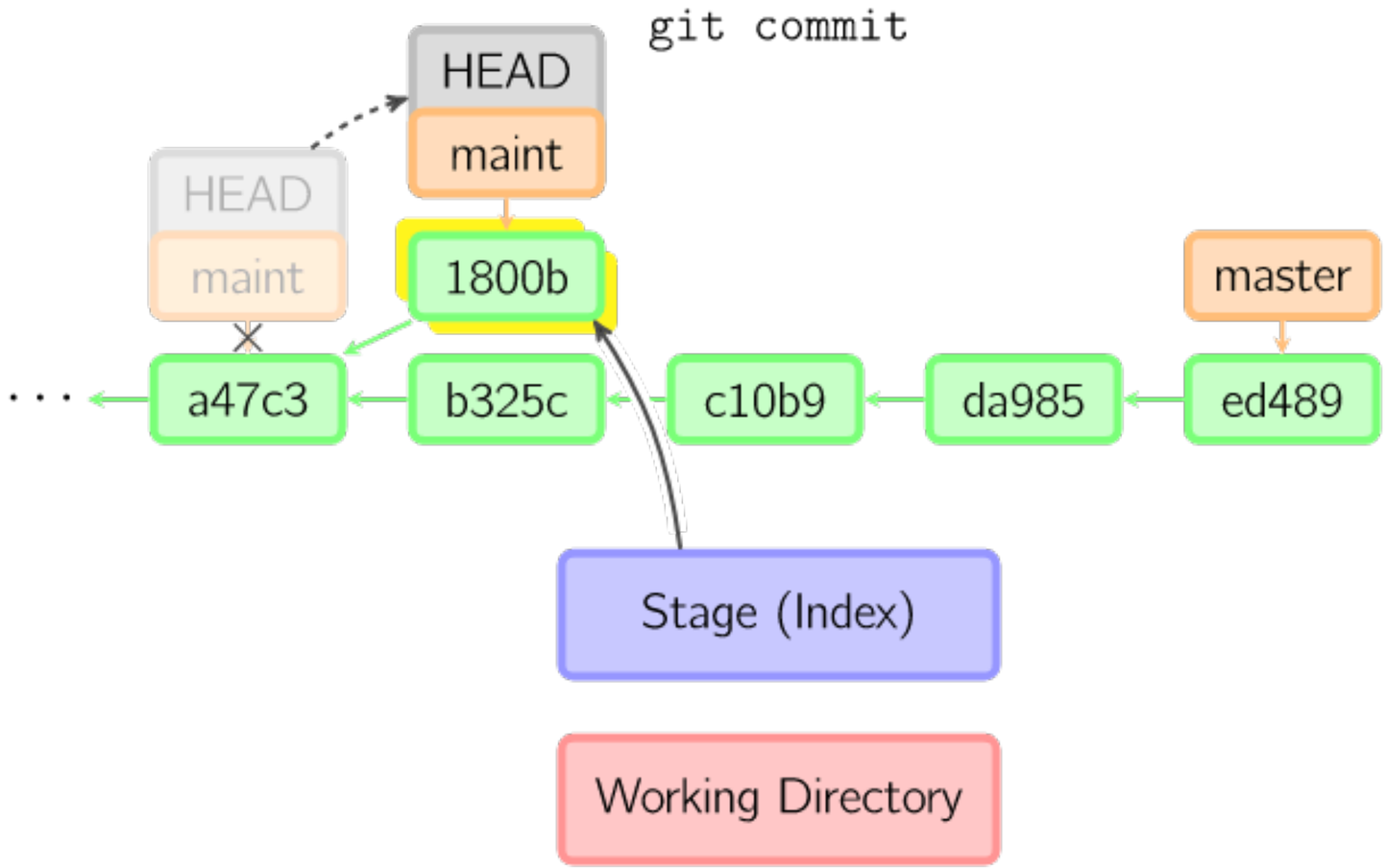
git diff


HEAD
maint
master
git diff b325c da985
...
a47c3
b325c
c10b9
da985
ed489
git diff --cached
Stage (Index)
git diff maint
git diff HEAD
git diff
Working Directory

git commit

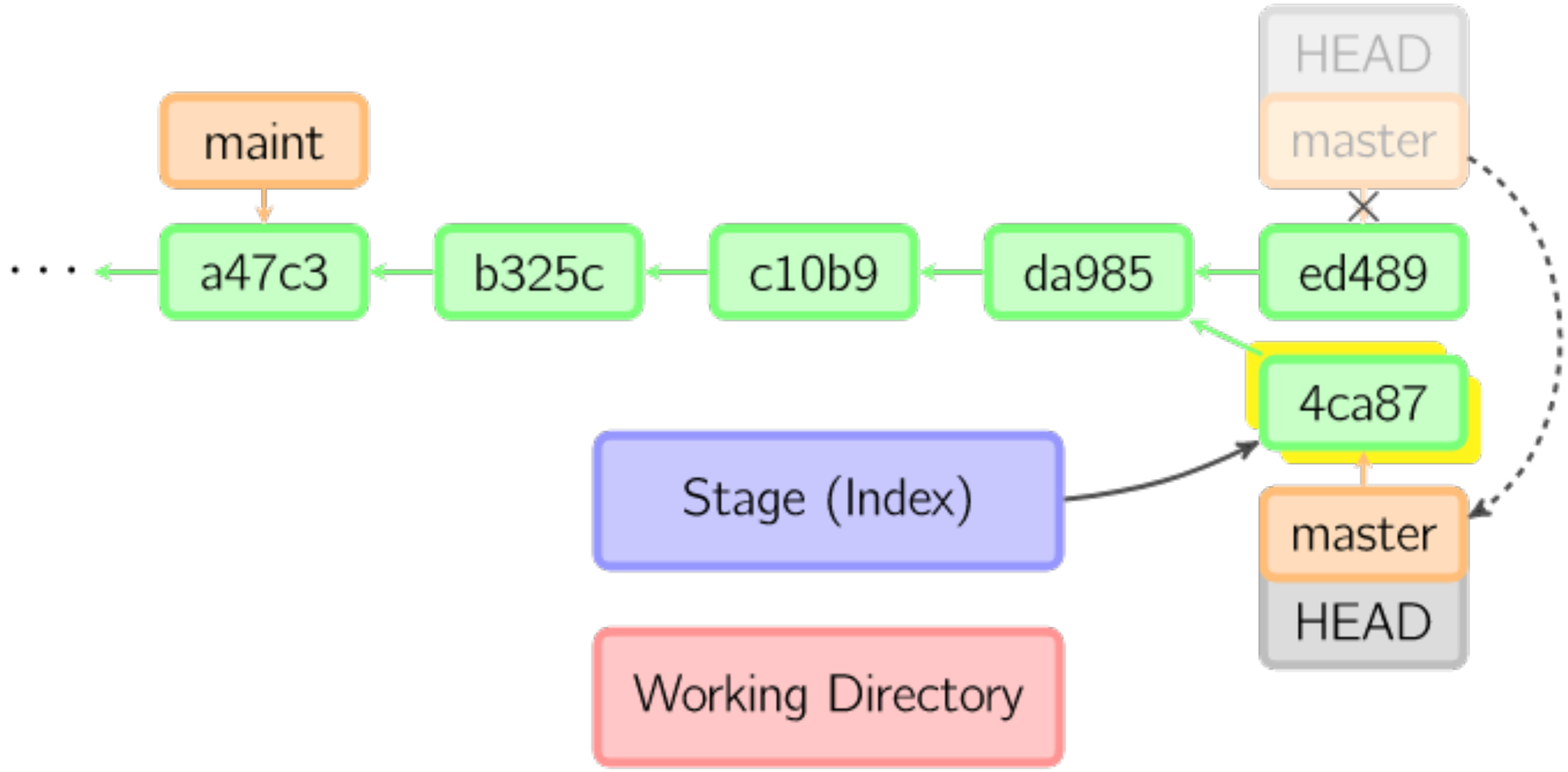

git commit
HEAD
master
HEAD
master
maint
a47c3
b325c
c10b9
da985
ed489
f0cec
Stage (Index)
Working Directory

git commit


git commit
HEAD
maint
HEAD
maint
1800b
master
a47c3
b325c
c10b9
da985
ed489
Stage (Index)
Working Directory

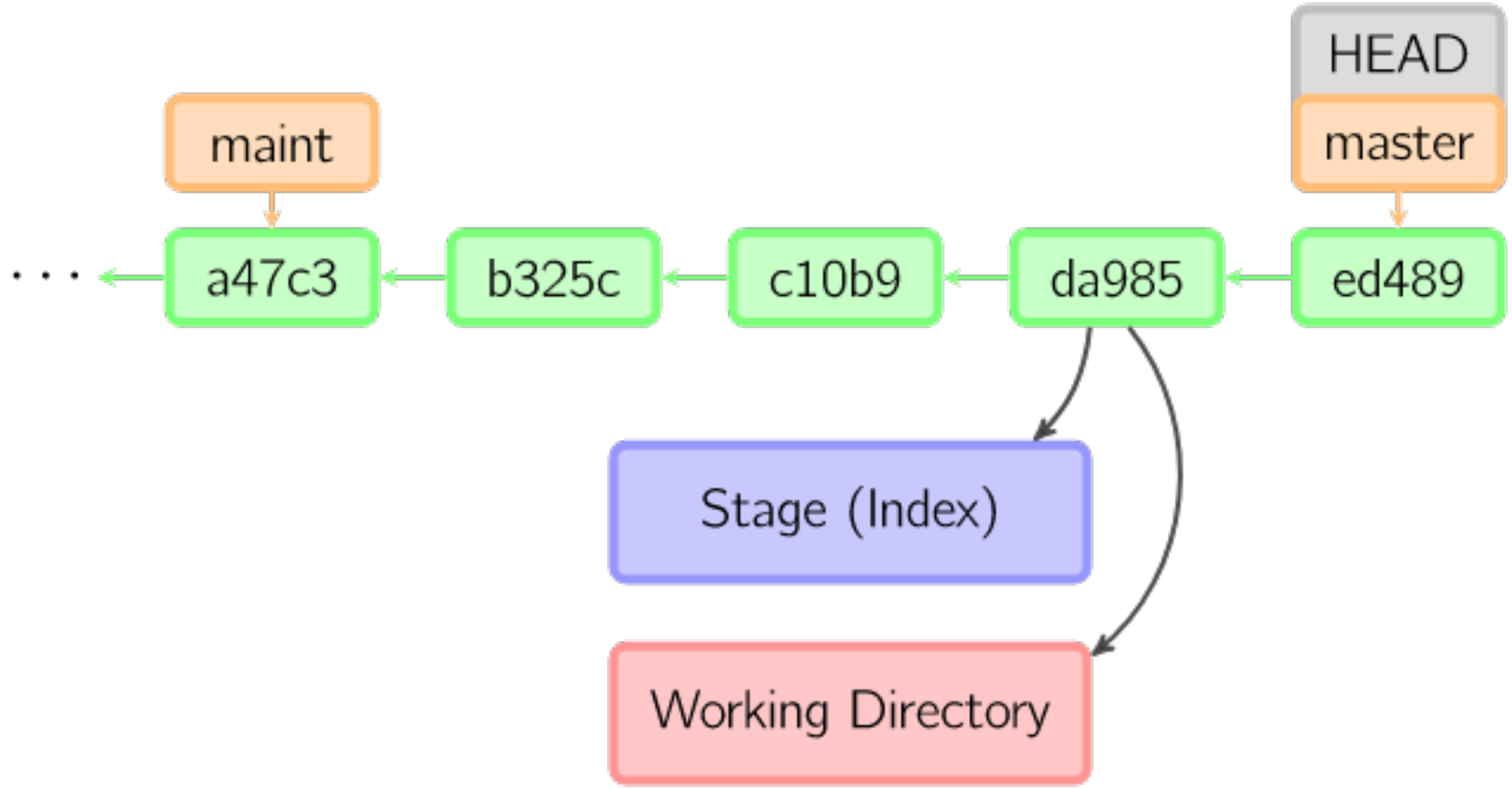
git commit

`git commit --amend`

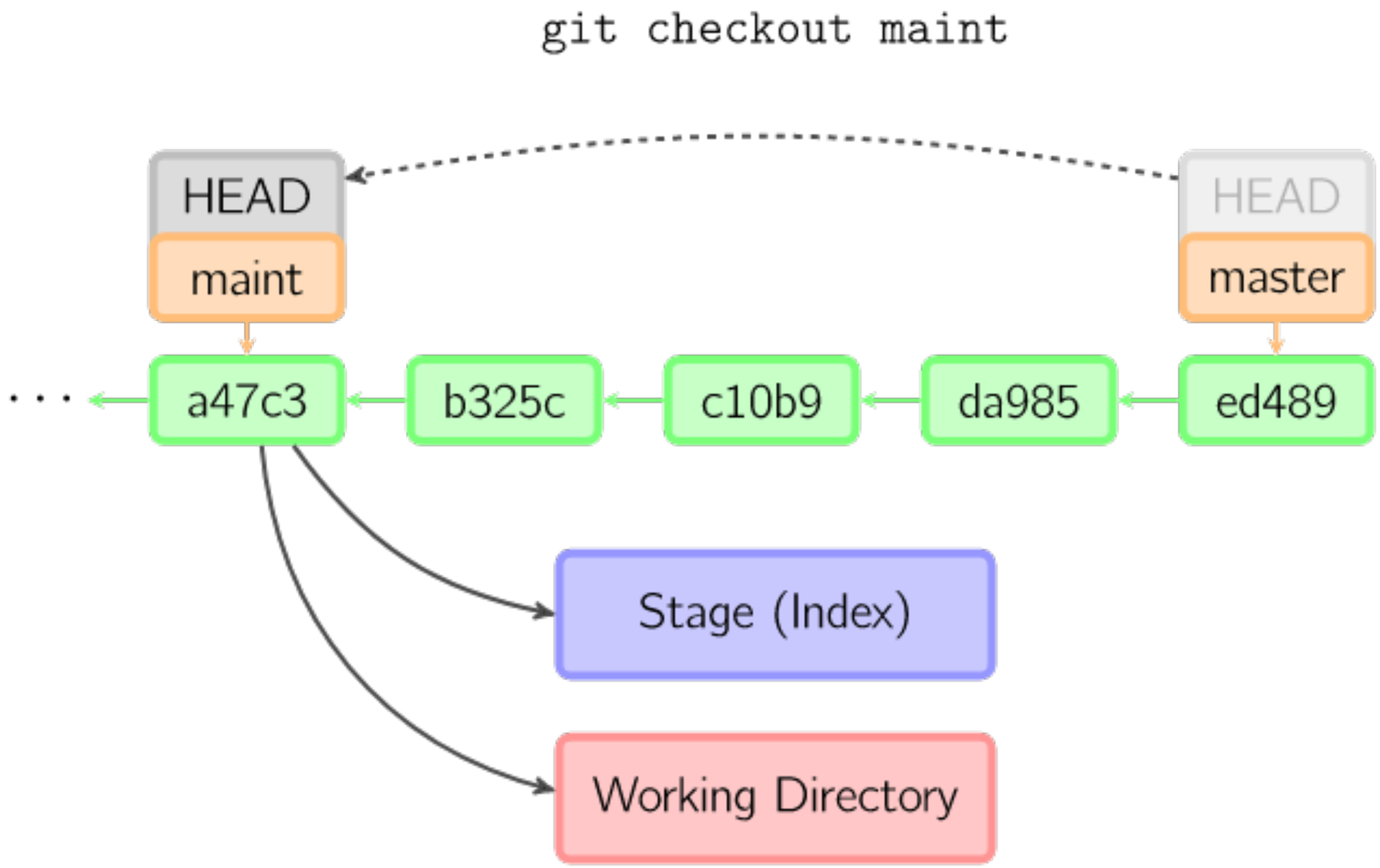


git checkout

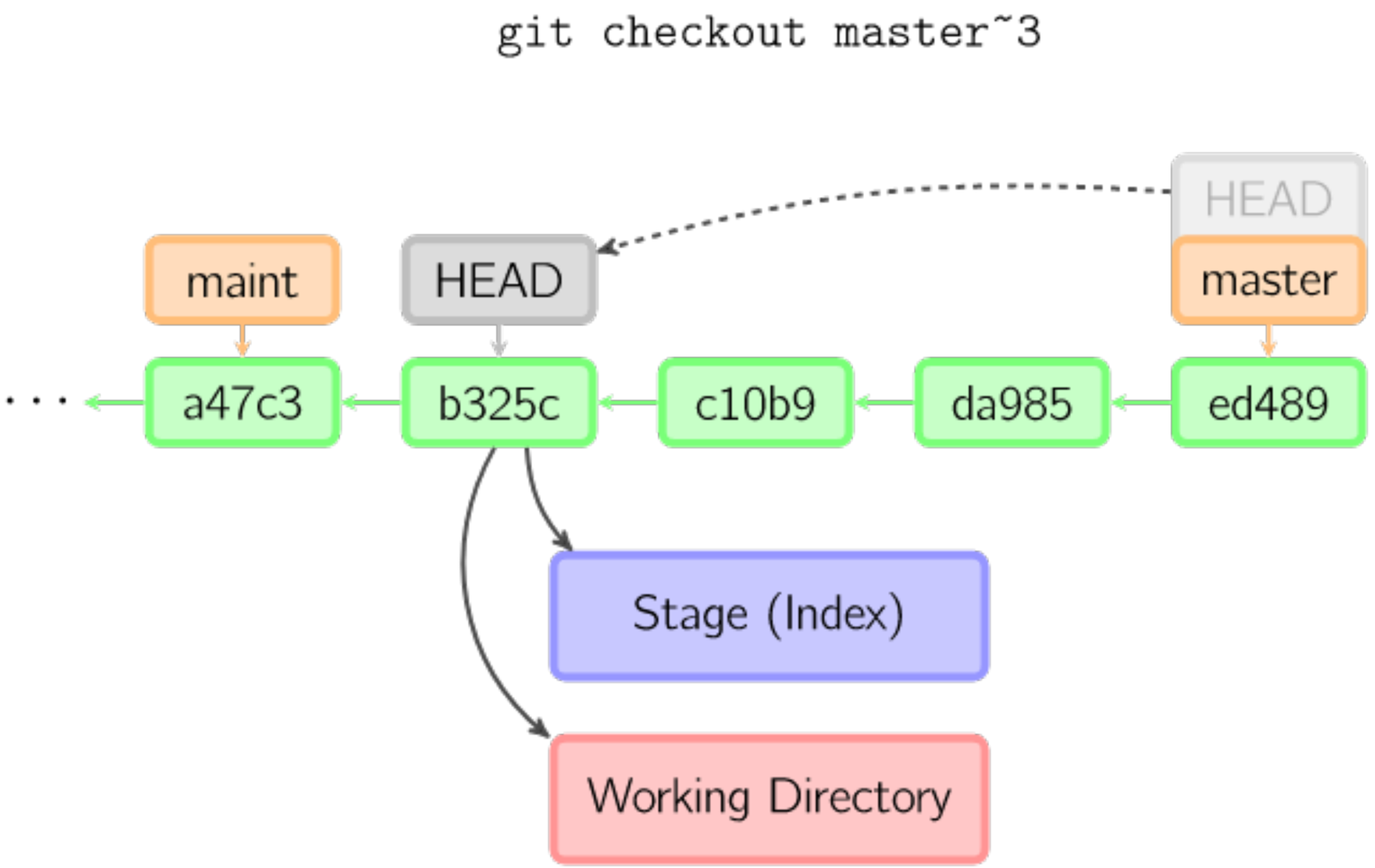
`git checkout HEAD~ files`



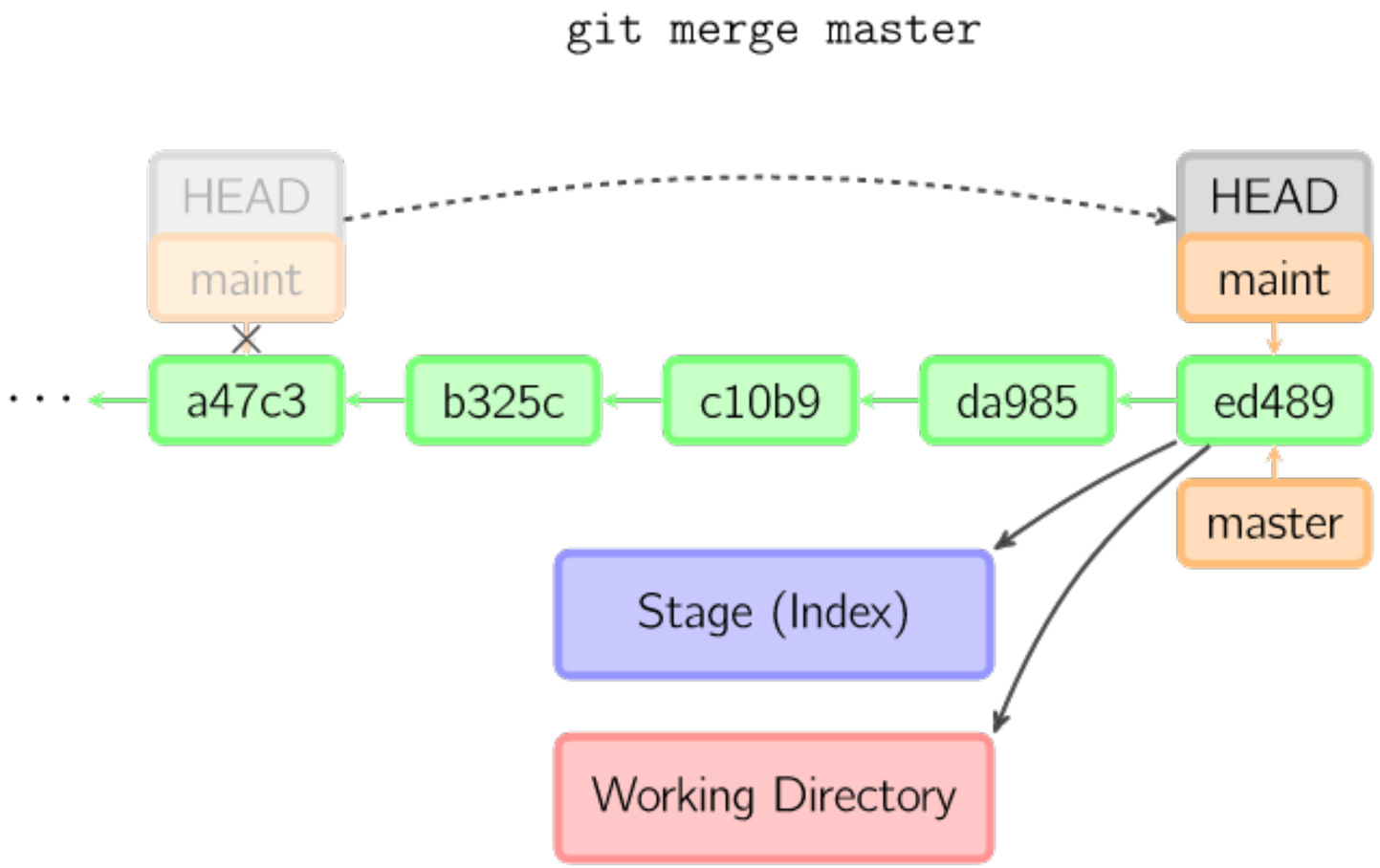
git checkout



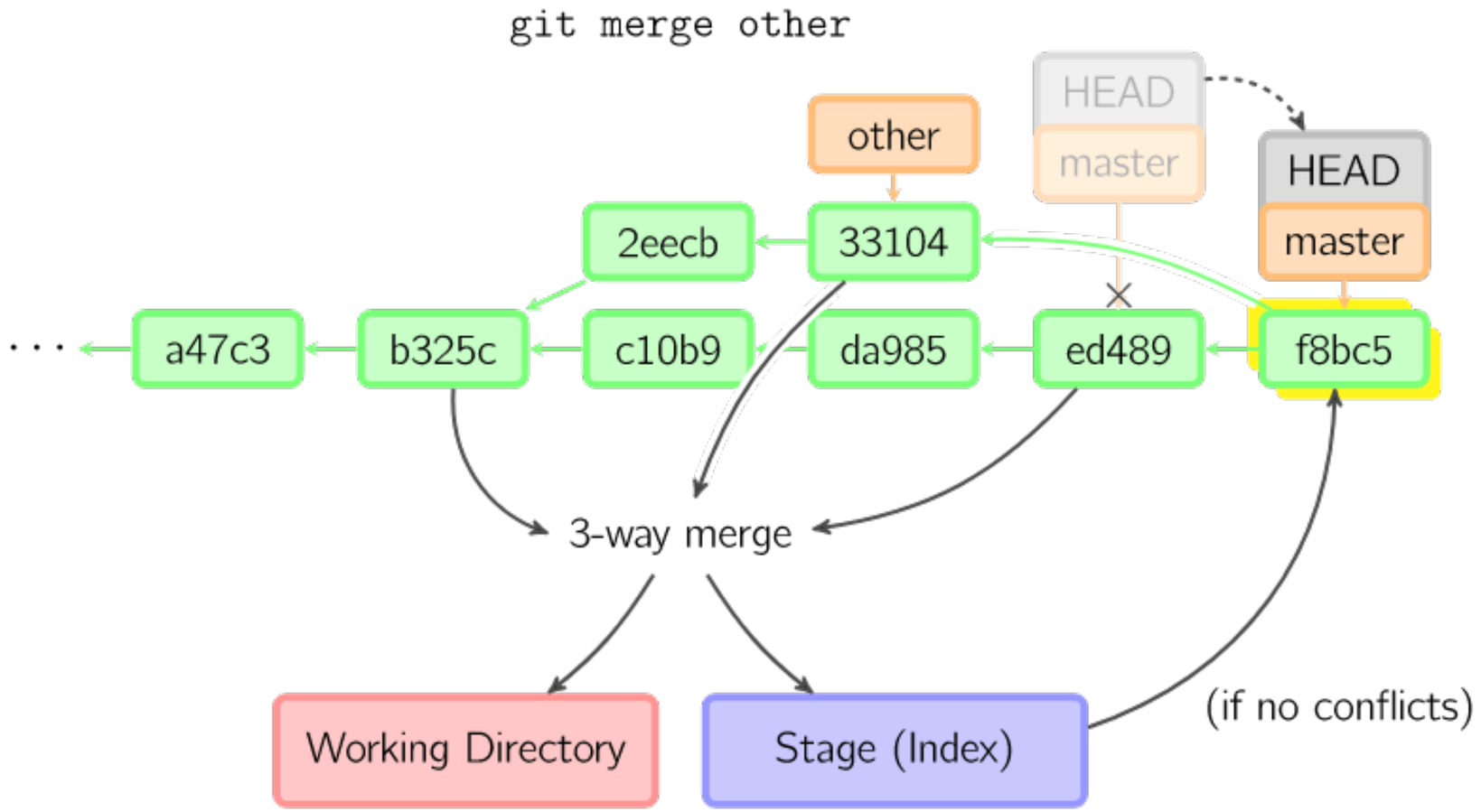
git checkout



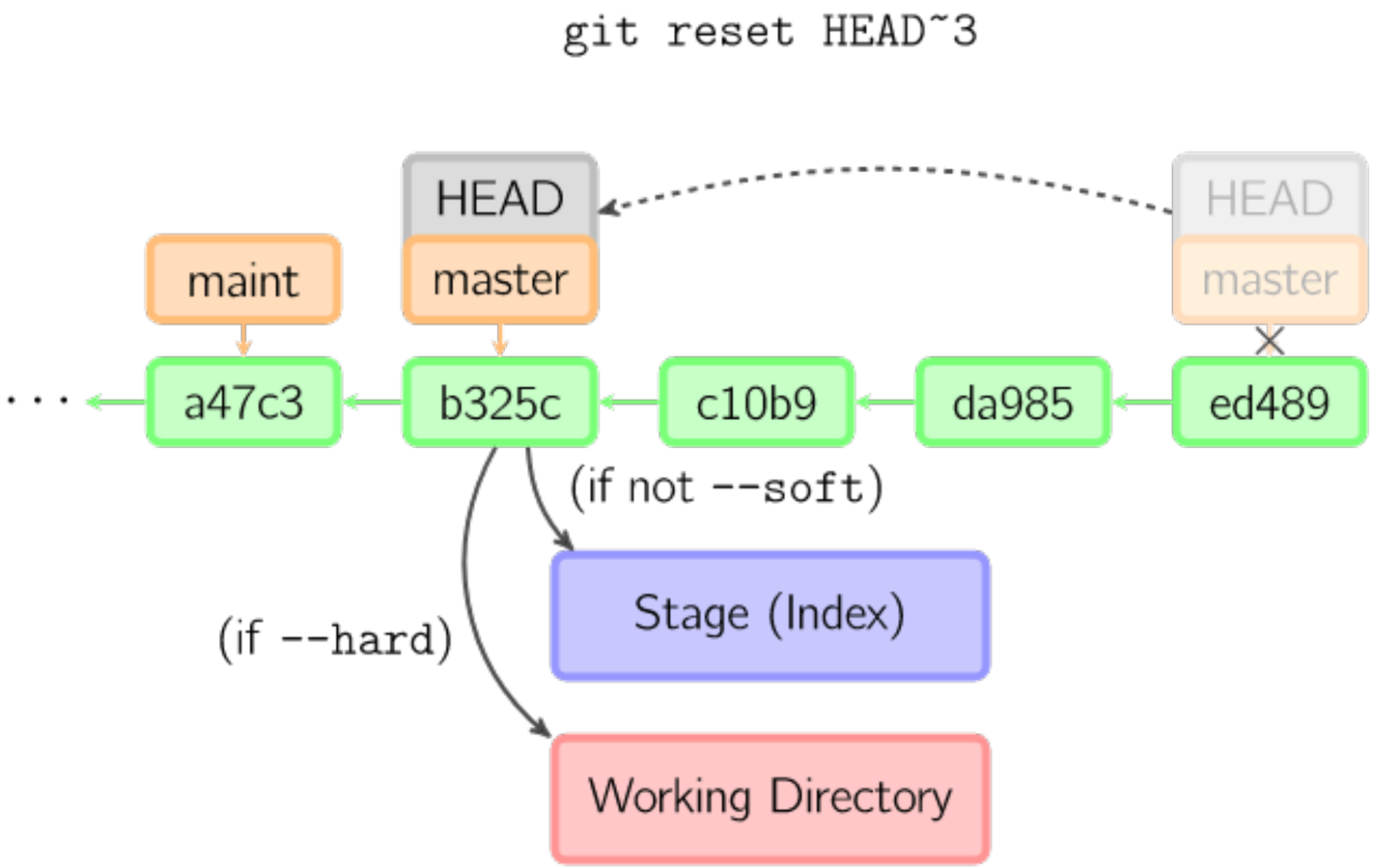
git merge


git merge master
HEAD
maint
HEAD
maint
a47c3
b325c
c10b9
da985
ed489
master
Stage (Index)
Working Directory

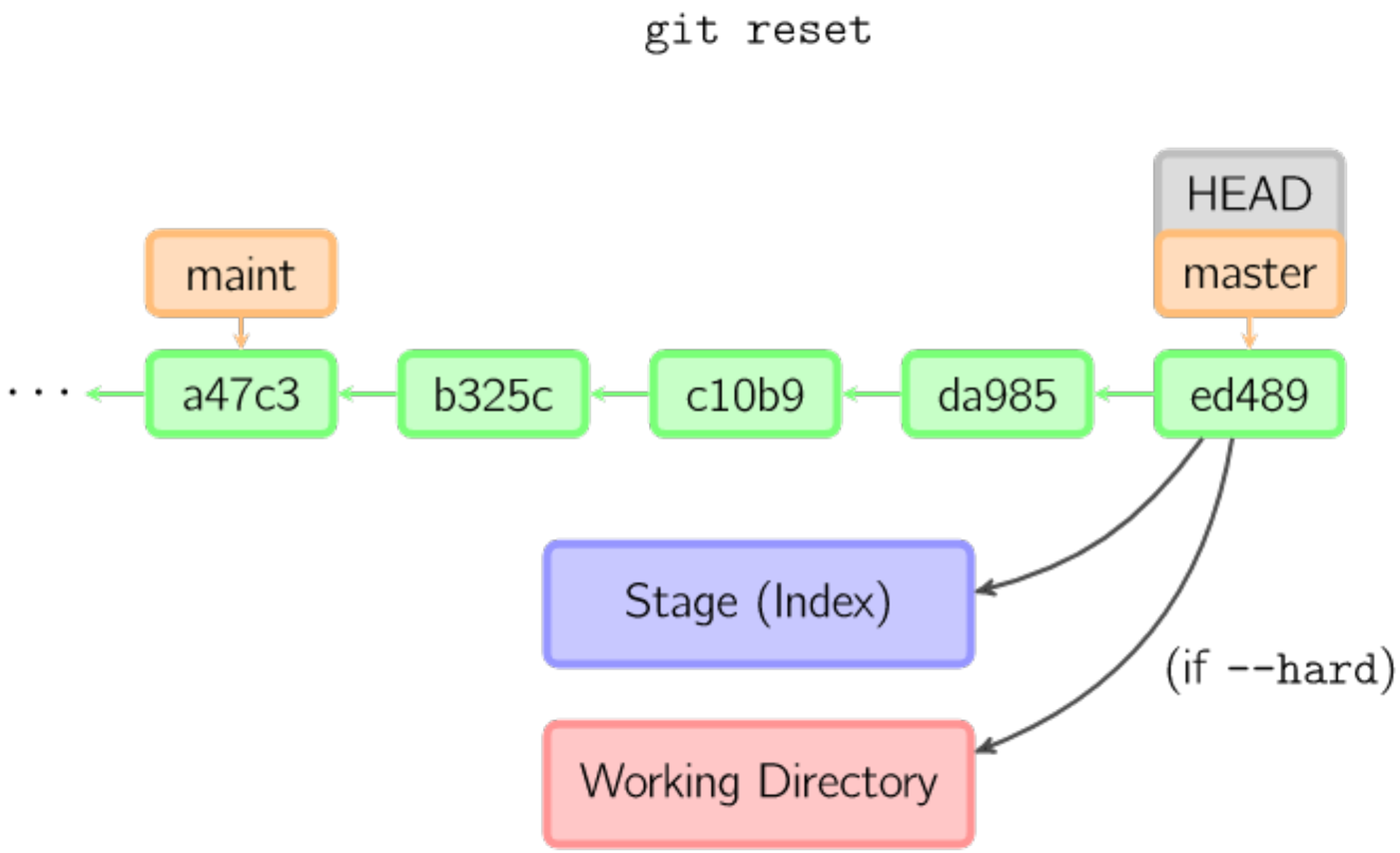
git merge


git merge other
HEAD
master
other
HEAD
master
2eecb
33104
a47c3
b325c
c10b9
da985
ed489
f8bc5
3-way merge
Working Directory
Stage (Index)
(if no conflicts)

git reset

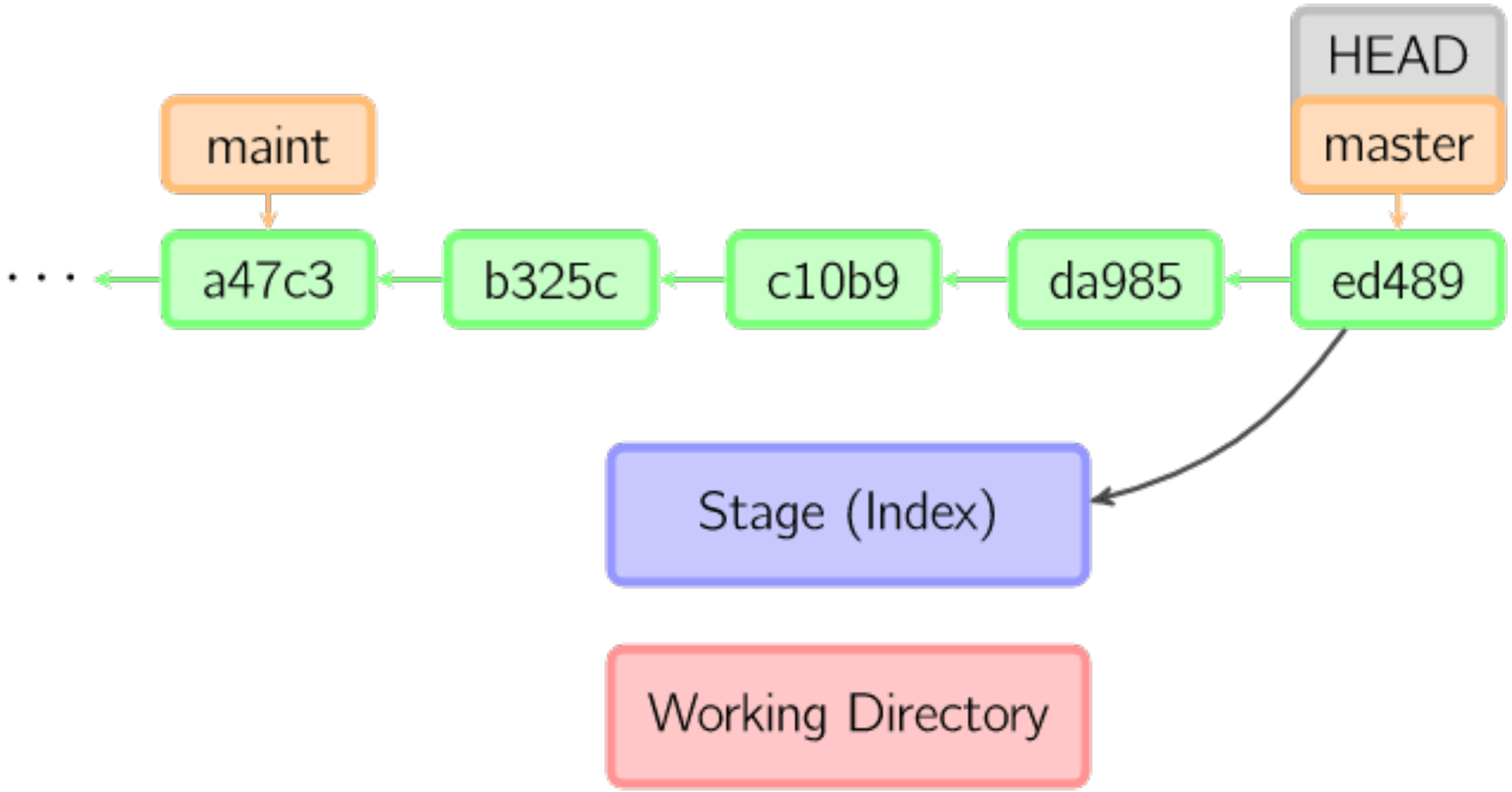

git reset HEAD~3
HEAD
maint
master
HEAD
master
a47c3
b325c
c10b9
da985
ed489
(if not --soft)
Stage (Index)
(if --hard)
Working Directory

git reset

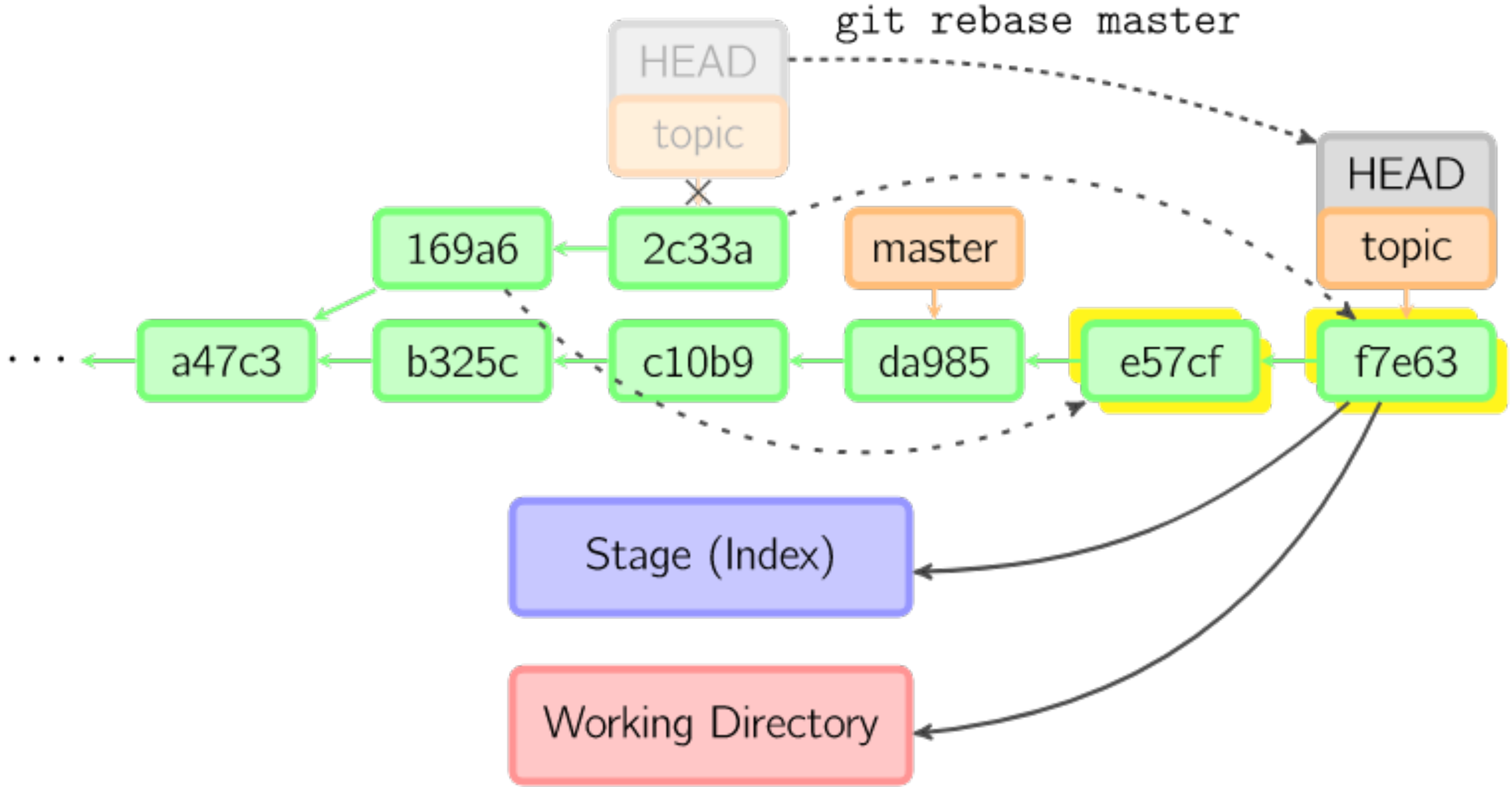

git reset
HEAD
maint
master
...
a47c3
b325c
c10b9
da985
ed489
Stage (Index)
(if --hard)
Working Directory

git reset

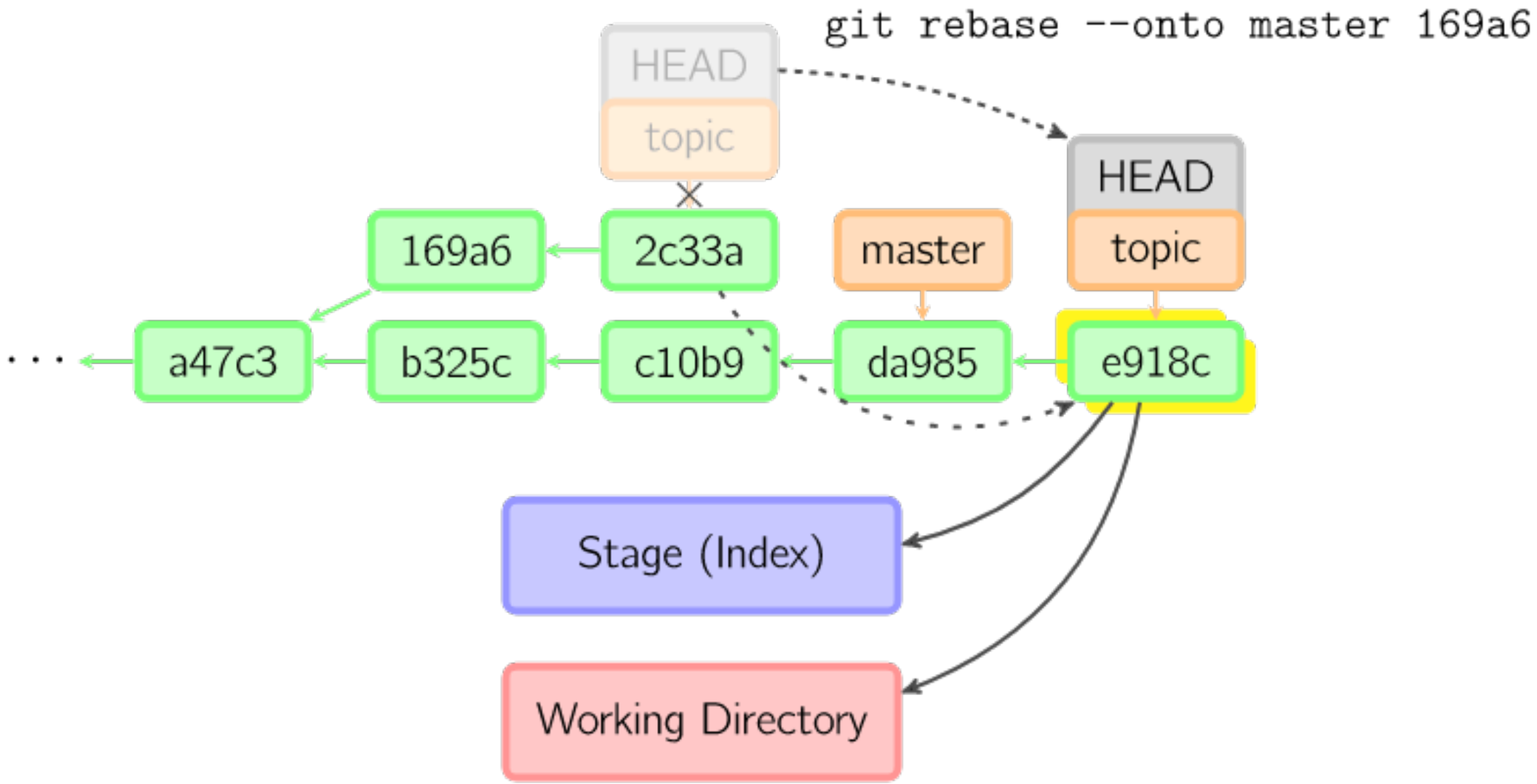
`git reset -- files`



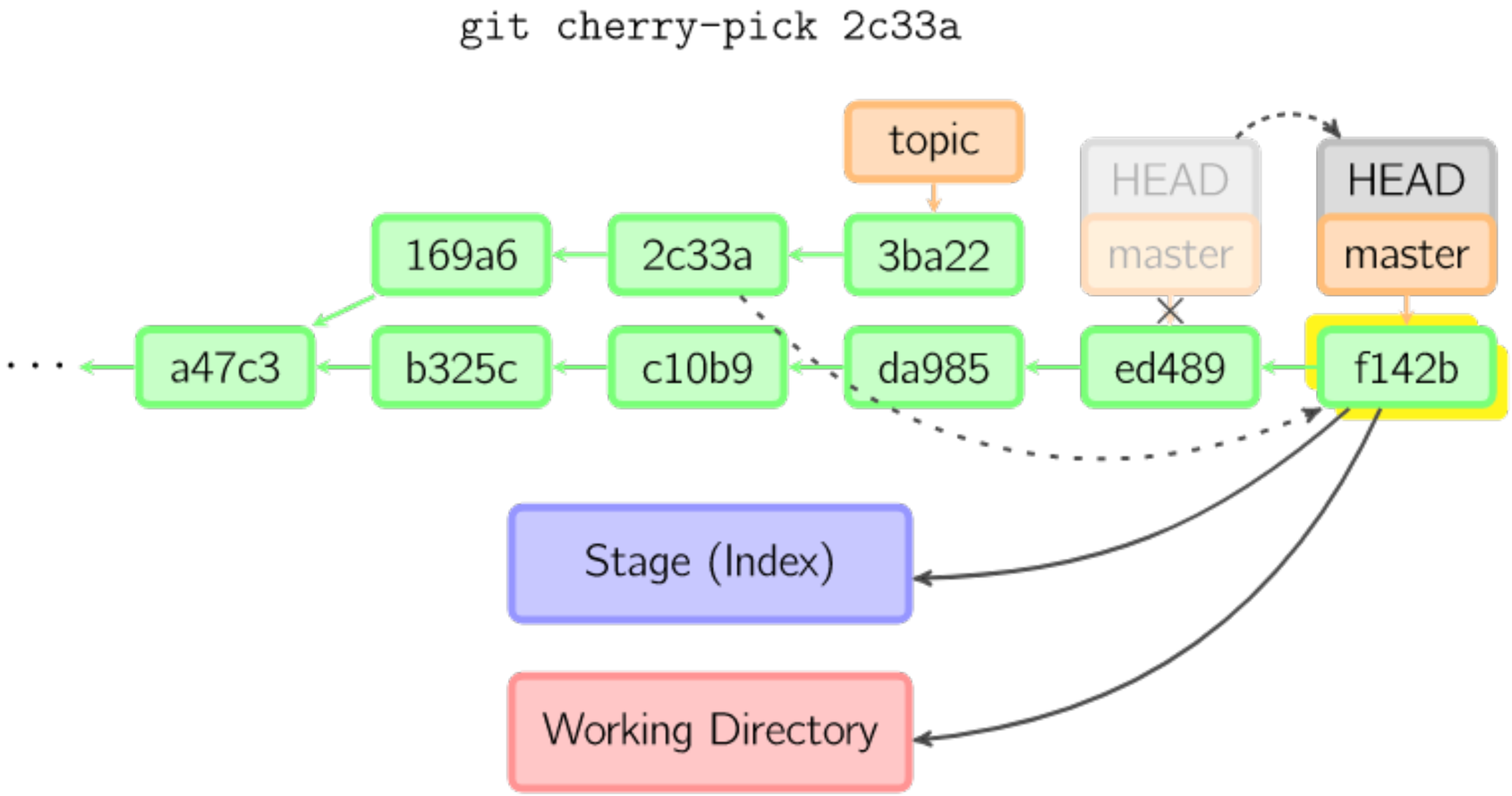
git rebase


git rebase master
HEAD
topic
HEAD
topic
169a6
2c33a
master
a47c3
b325c
c10b9
da985
e57cf
f7e63
Stage (Index)
Working Directory

git rebase


git rebase --onto master 169a6
HEAD
topic
HEAD
topic
169a6
2c33a
master
a47c3
b325c
c10b9
da985
e918c
Stage (Index)
Working Directory

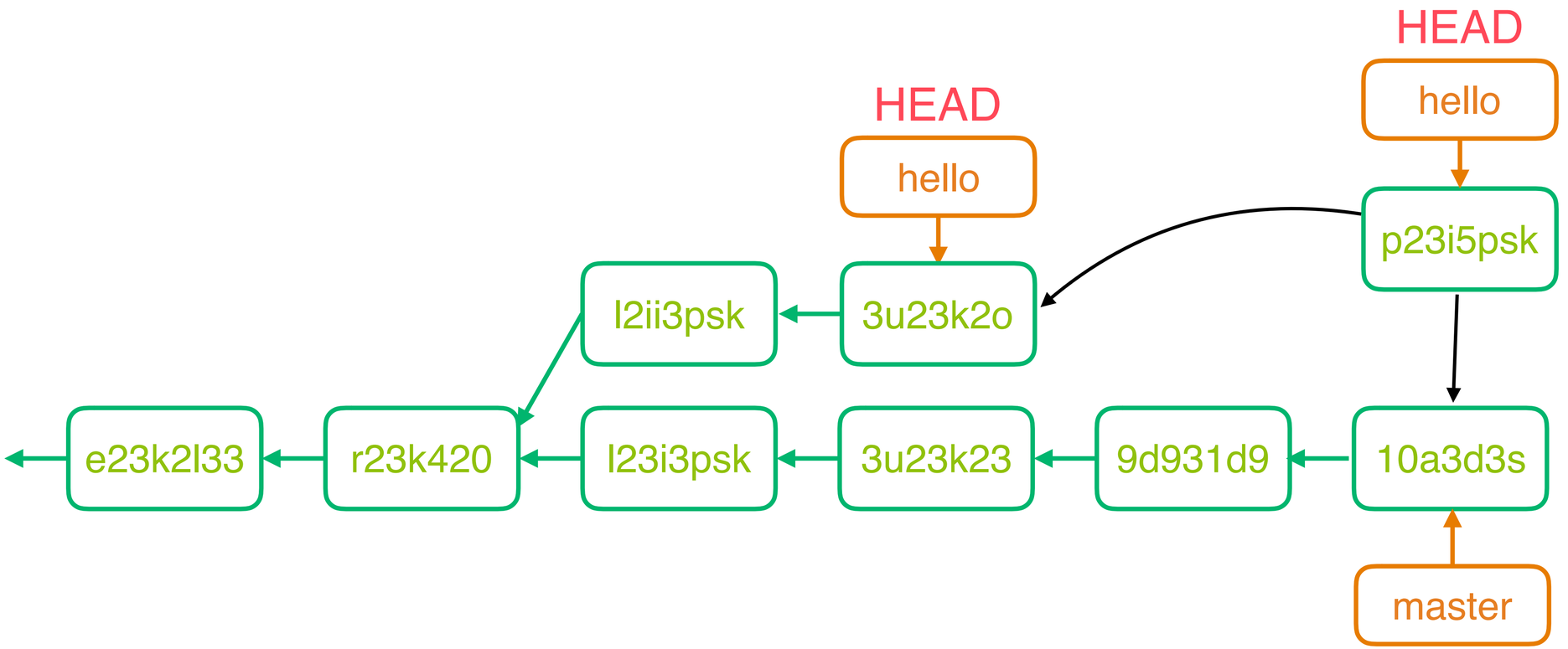
git cherry-pick



Git高级应用与最佳实践

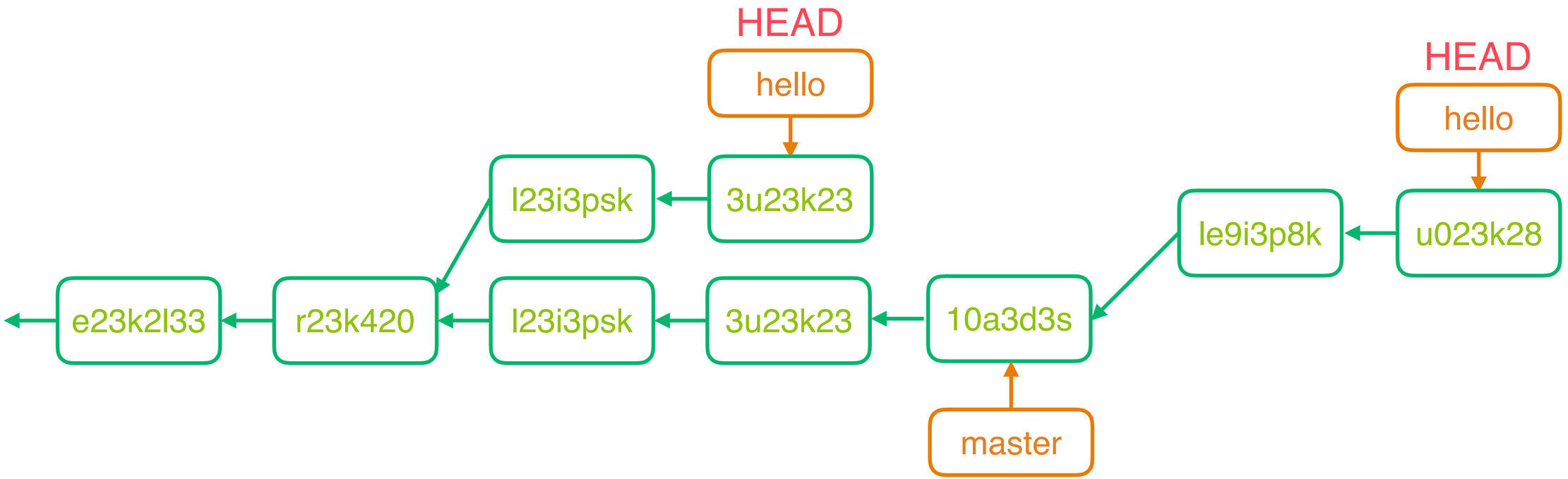
3

Merge vs Rebase



git merge master

Merge vs Rebase



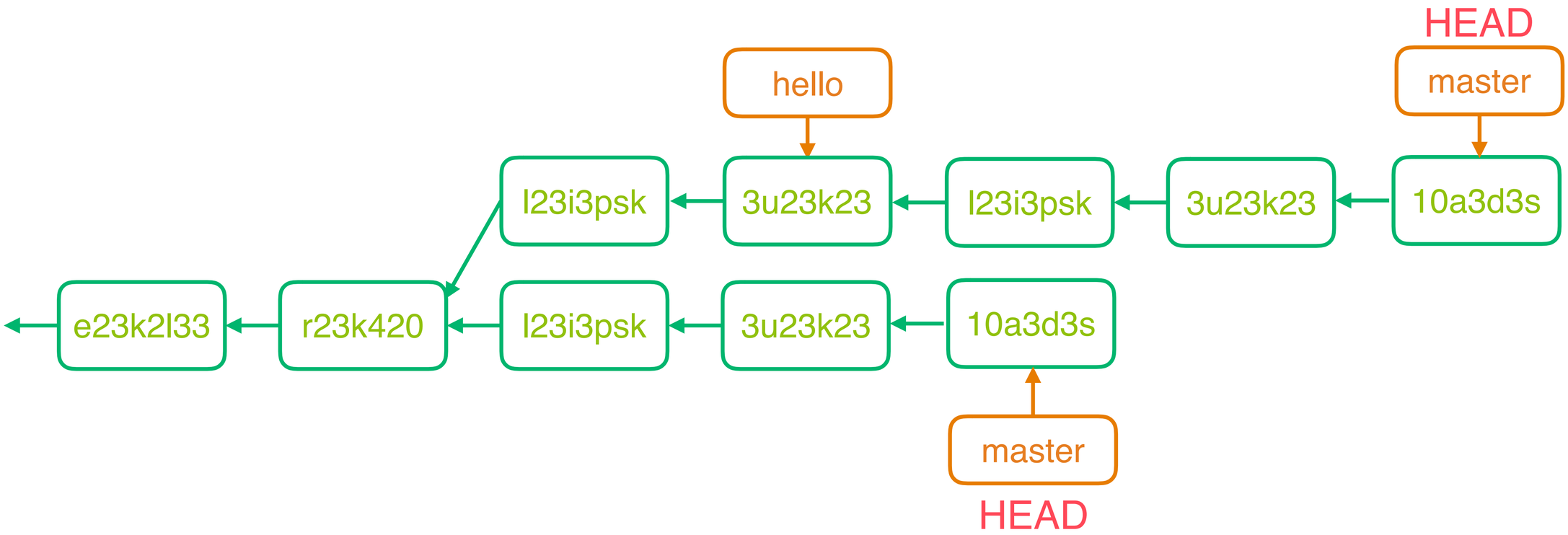
git rebase master

Merge vs Rebase

解决同样的问题，只是方法不同

额外创建一个提交	不会多生成一个提交
保留分支历史，看到分支结构	线性化合并分支，破坏分支历史
非破坏性操作，避免风险	破坏性操作，存在潜在风险
随着项目增加，不便于理解	需要遵守黄金法则

Diverged Rebase



“

“不要在公共分支上使用
REBASE”

最佳实践之
分支管理

1. 创建一个新分支
2. 每个组（个人）工作在自己的分支上
3. 小步提交

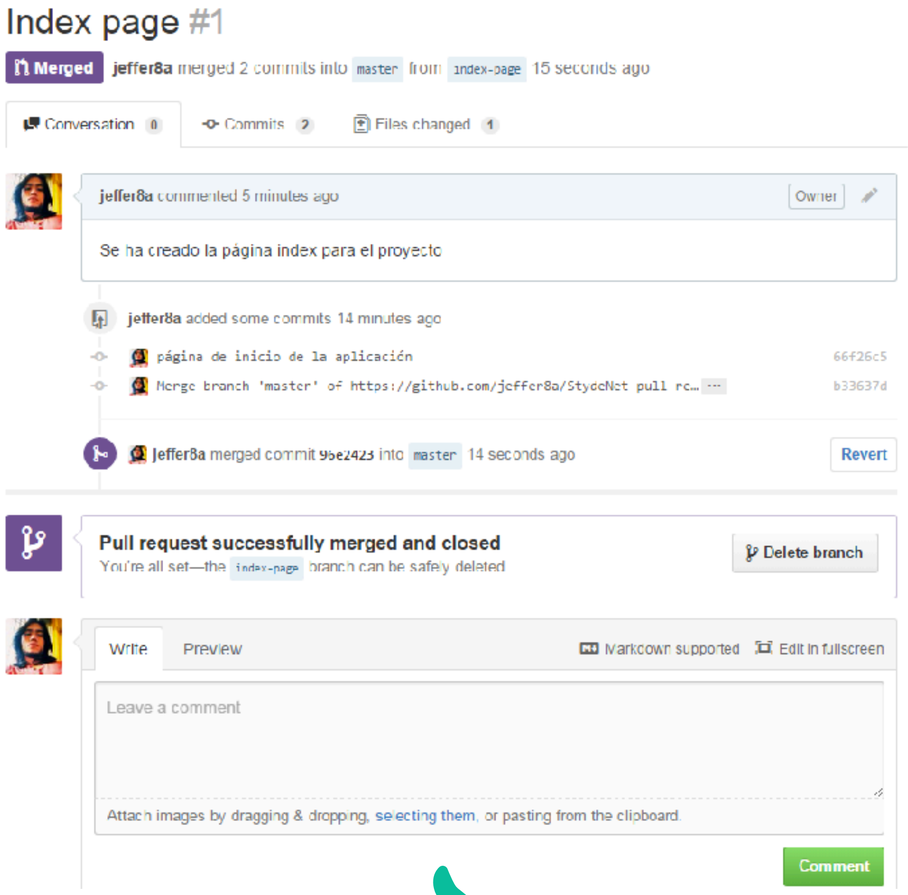
4. 经常同步主分支
5. 重复3，4直到完成该功能

```
git checkout -b ‘branch-name’

git add .
git commit -m ‘desc’
git pull origin master --rebase
git push origin ‘branch-name’
```

最佳实践之
PR


Passport

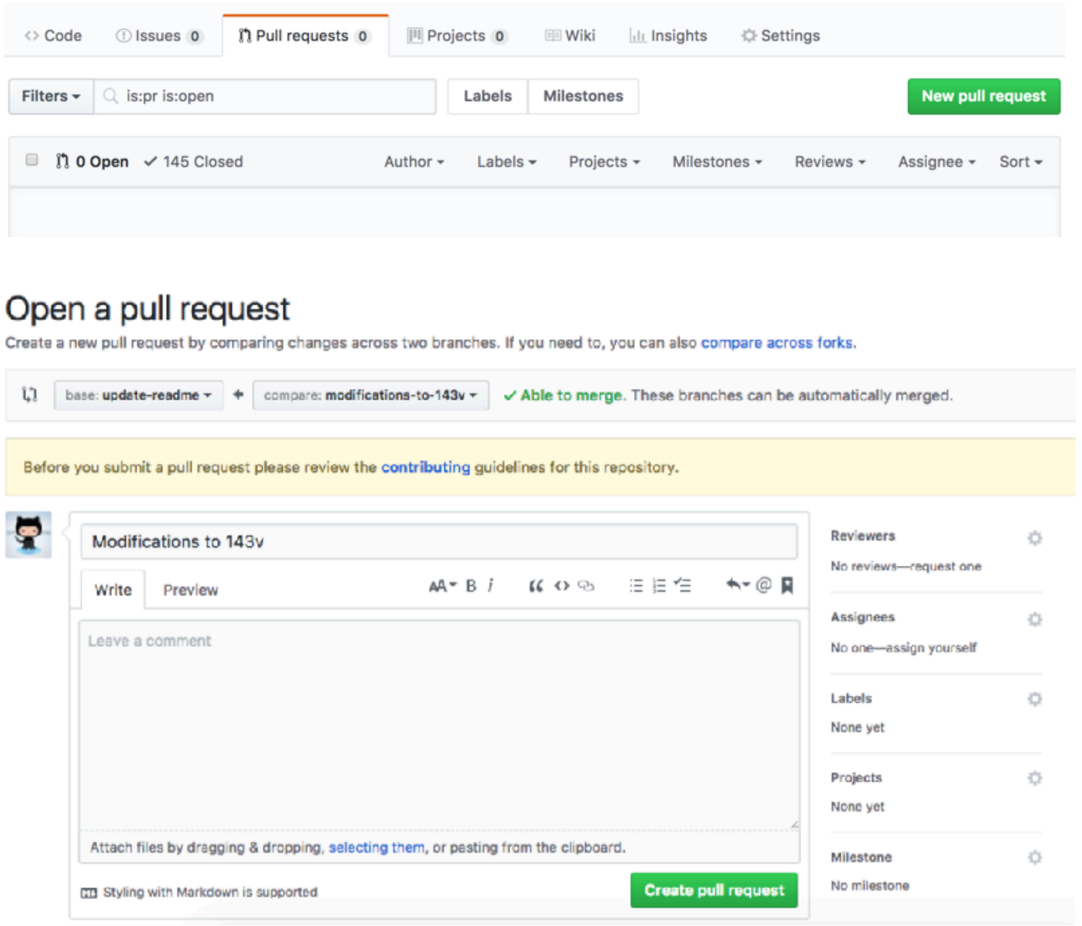

Index page #1
Merged
jeffer8a merged 2 commits into master from index-page 15 seconds ago
Conversation 0
Commits 2
Files changed 1
jeffer8a commented 5 minutes ago
Owner
Se ha creado la página index para el proyecto
jeffer8a added some commits 14 minutes ago
página de inicio de la aplicación
66f26c5
Merge branch 'master' of https://github.com/jeffer8a/StydeNet pull re…
b33637d
Jeffer8a merged commit 9be2423 into master 14 seconds ago
Revert
Pull request successfully merged and closed
You're all set—the index-page branch can be safely deleted.
Delete branch
Write
Preview
Markdown supported
Edit in fullscreen
Leave a comment
Attach images by dragging & dropping, selecting them, or pasting from the clipboard.
Comment

最佳实践之
PR

Pull Request 是团队协作中代码审查的一种方式。PR用于告诉其他成员你希望做的改变，或者你正在做的功能已经完成，成员可以审查，同意或者添加评论表示质疑，在经过修改后达成一致，最终将代码合并。

最佳实践之
PR

1. 创建Pull Request
2. 填写PR内容：完成的功能，截图对比等
3. 团队Review代码：同意或评论
4. 代码修改，再次提交
5. 合并完成



作业

1. 在线练习 https://try.github.io

2. 学习MarkDown语法　参考链接

3. 提交自己的一段代码

4. 创建自己的github页面　Guide

Thanks!

Any questions?