Git Guide

James Zhan

2013-12-20

Git Guides

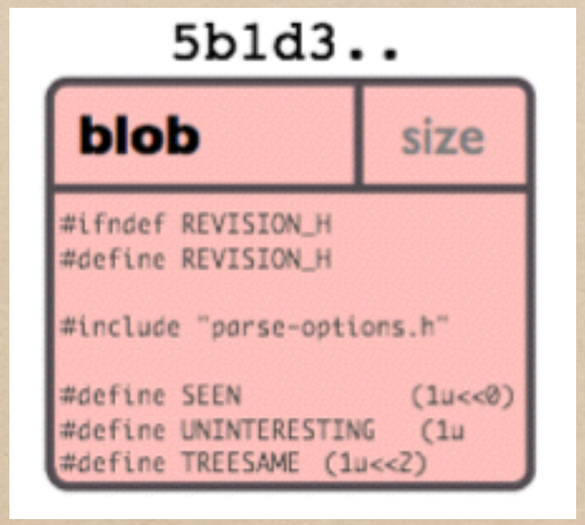
- Git 文件存储
- Git 基本操作

Git Objects

- blob
- tree
- commit
- tag

Git blob object

The git "blob" type is just a bunch of bytes that could be anything, like a text file, source code, or a picture, etc.

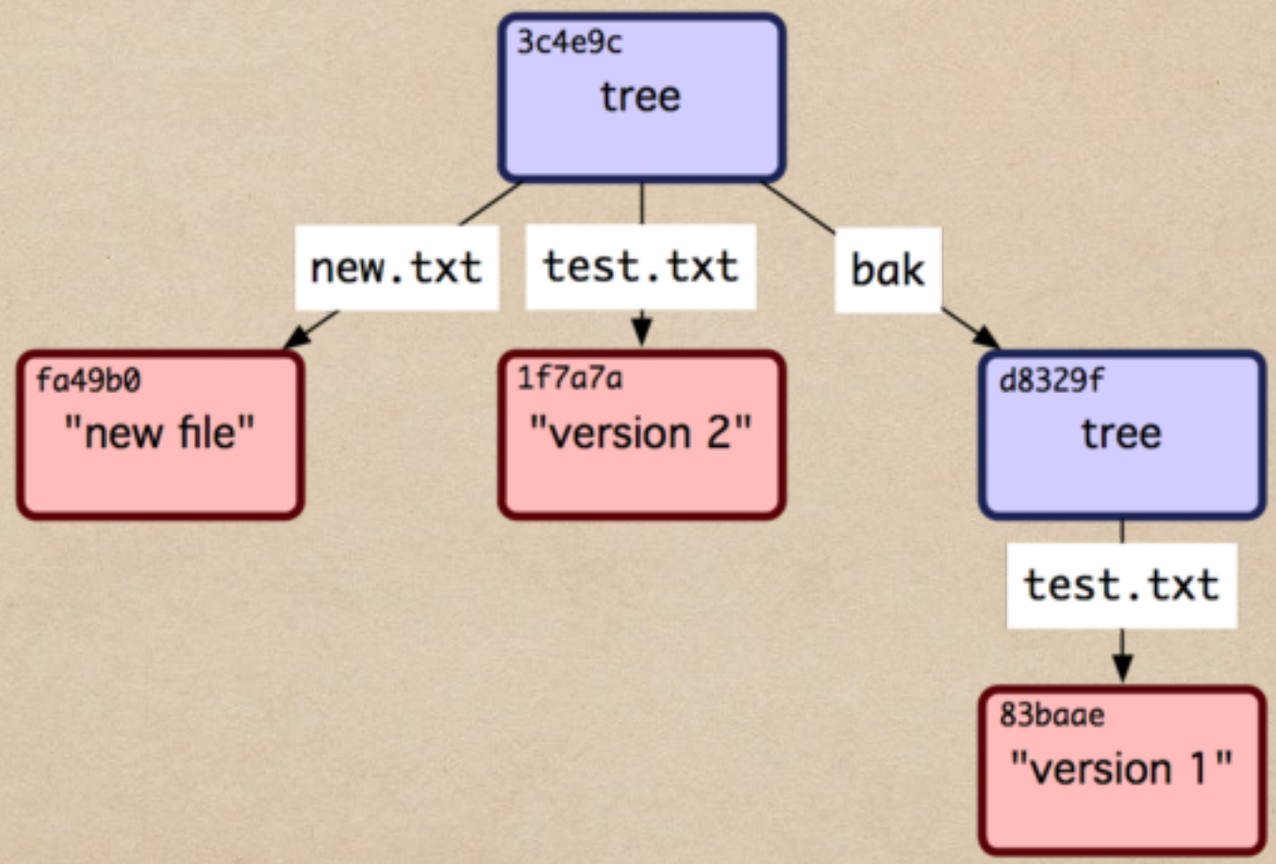


Git Tree Object

A git tree is like a filesystem directory. A git tree can point to, or include:

- Git "blob" objects (similar to a filesystem directory includes filesystem files).

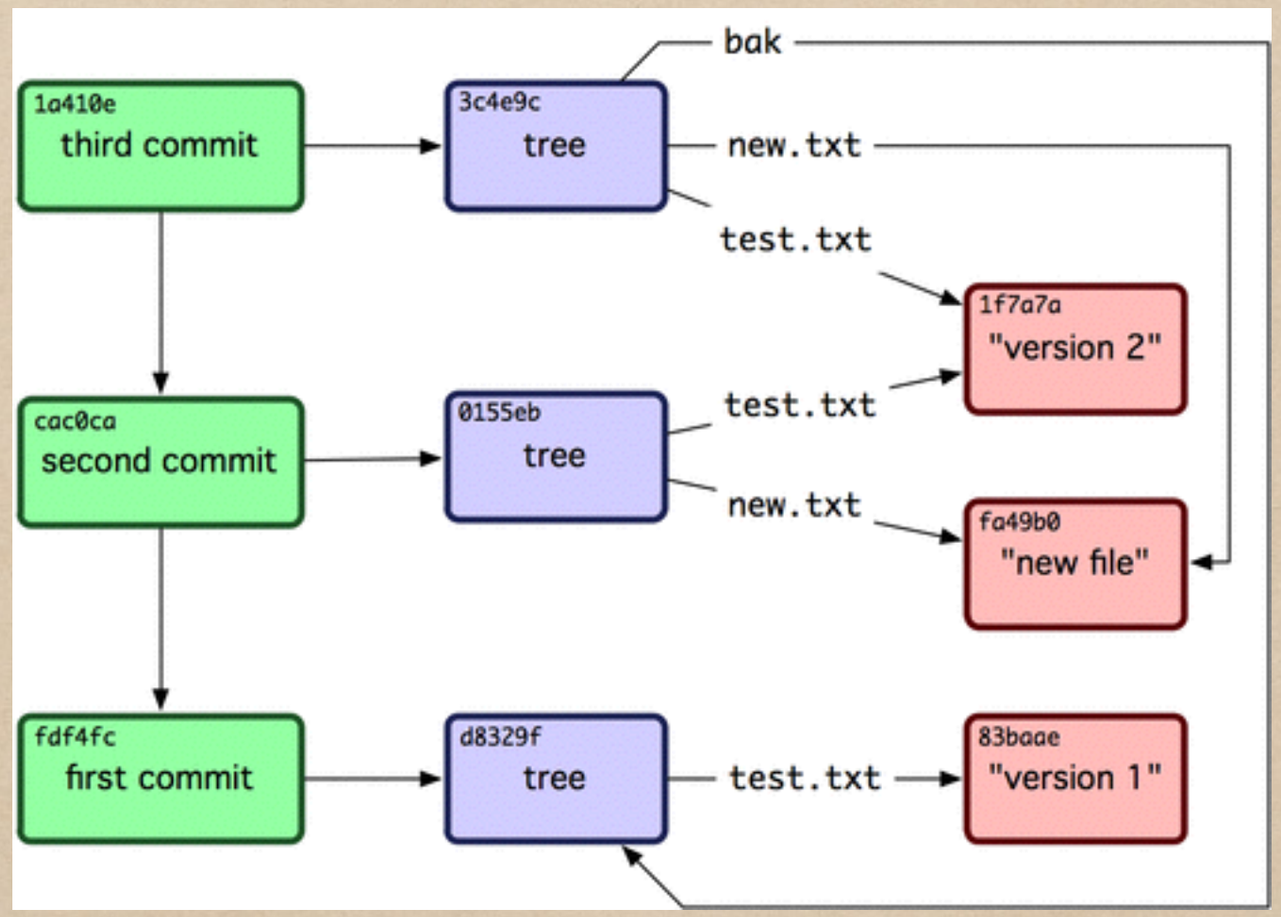
- Other git trees (similar to a filesystem directory can have subdirectories).


3c4e9c
tree
new.txt
test.txt
bak
fa49b0
"new file"
1f7a7a
"version 2"
d8329f
tree
test.txt
83baae
"version 1"

Git Commit Object

A git commit object includes:

- Information about who commited (made) the change/check-in/ commit. For example, it stores the name and email address.
- A pointer to the git tree object that represents the git repository when the commit was done
- The parent commit to this commit (so we can easily find out the situation at the previous commit).


bak
1a410e
third commit
3c4e9c
tree
new.txt
test.txt
1f7a7a
"version 2"
test.txt
cac0ca
second commit
0155eb
tree
new.txt
fa49b0
"new file"
fdf4fc
first commit
d8329f
tree
test.txt
83baae
"version 1"

Git Tag Object

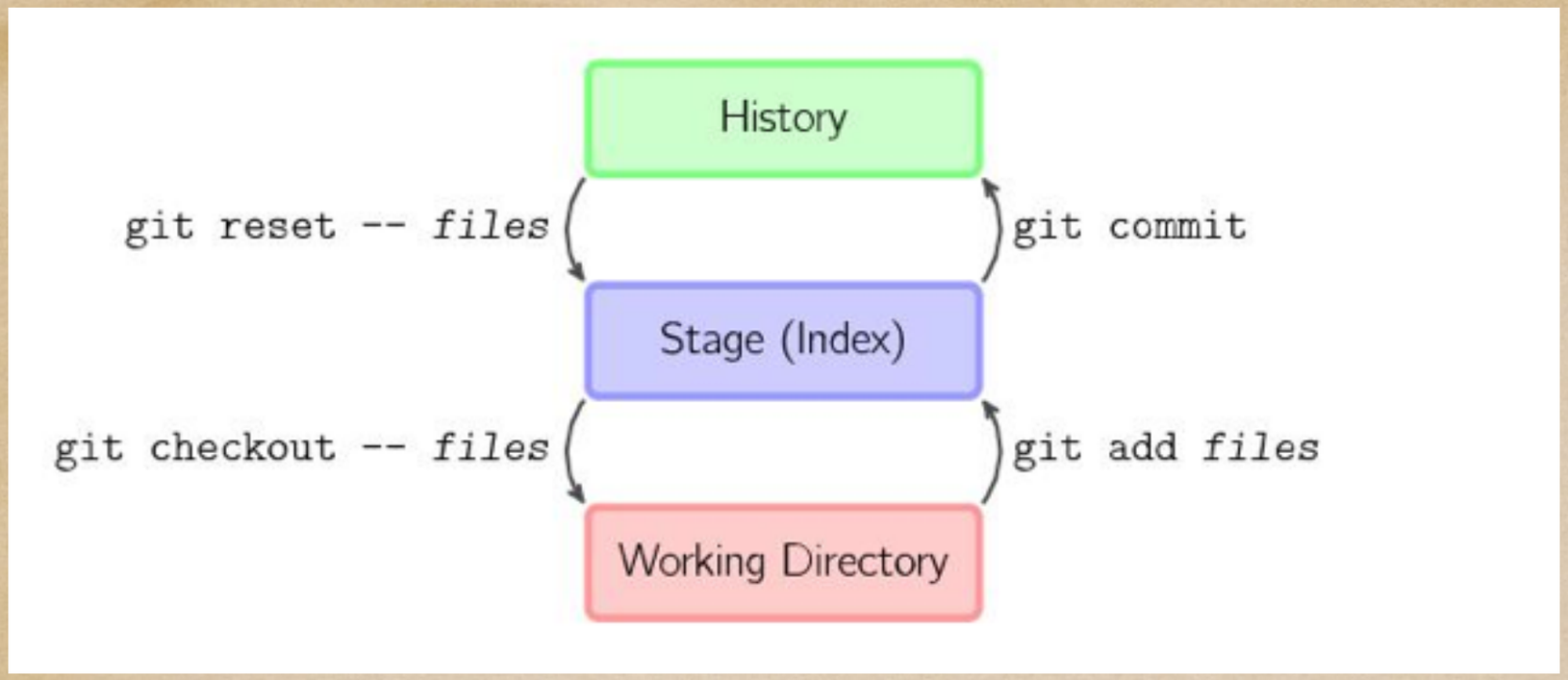
A git tag object points to any git commit object. A git tag can be used to refer to a specific tree, rather than having to remember or use the hash of the tree.

Ok, Let's try it.

git 基本命令

- Git 有超过100个子命令
- 每个子命令有不同的参数，不同的参数有时候语义相差甚远。
- 掌握基本命令，如果有特殊用法不清楚，可以参号Google和Stackoverflow。

废话：当然有精力把每个命令研究清楚也是极好的，但是这个太耗费时间了，而且过不了2个月不用，你就会忘的一干二净。我的建议是，先把基本命令掌握好，然后高级命令过一遍，以后碰到相应问题的时候，可以快速的寻找解决方案。



上面的四条命令在工作目录、暂存目录(也叫做索引)和仓库之间复制文件。

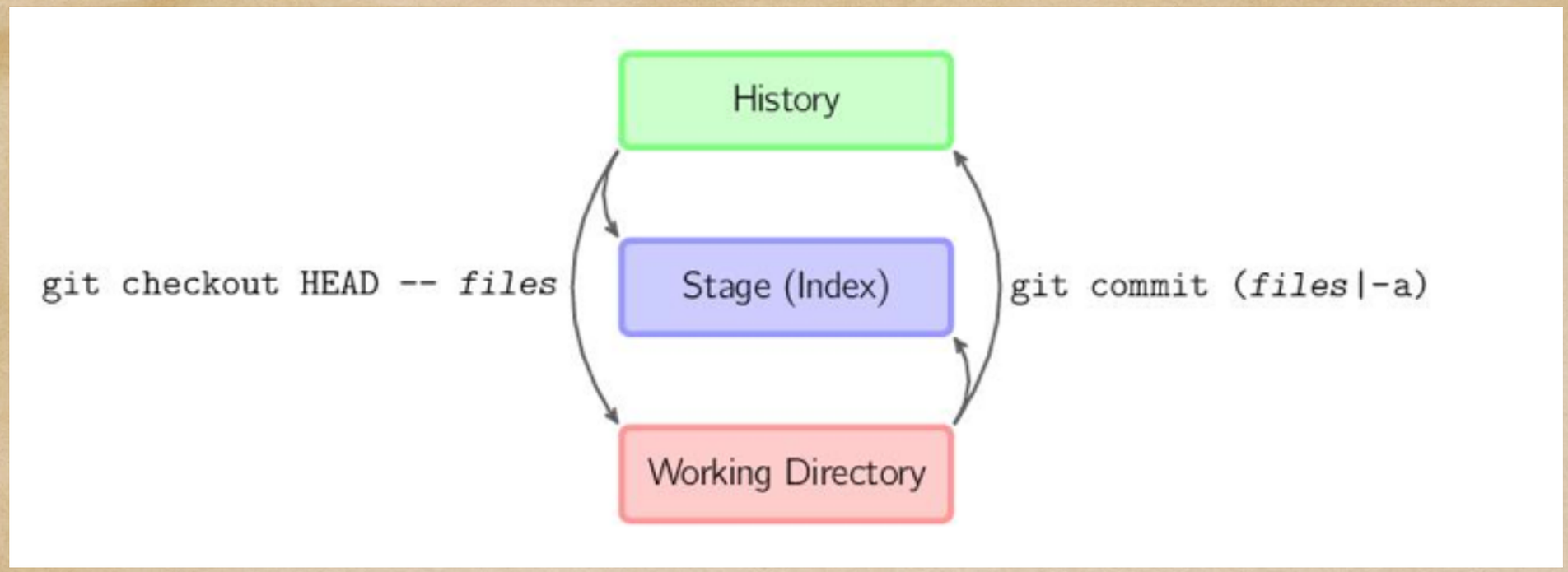
git add files 把当前文件放入暂存区域。

git commit 给暂存区域生成快照并提交。

git reset -- files 用来撤销最后一次git add files，你也可以用git reset 撤销所有暂存区域文件。

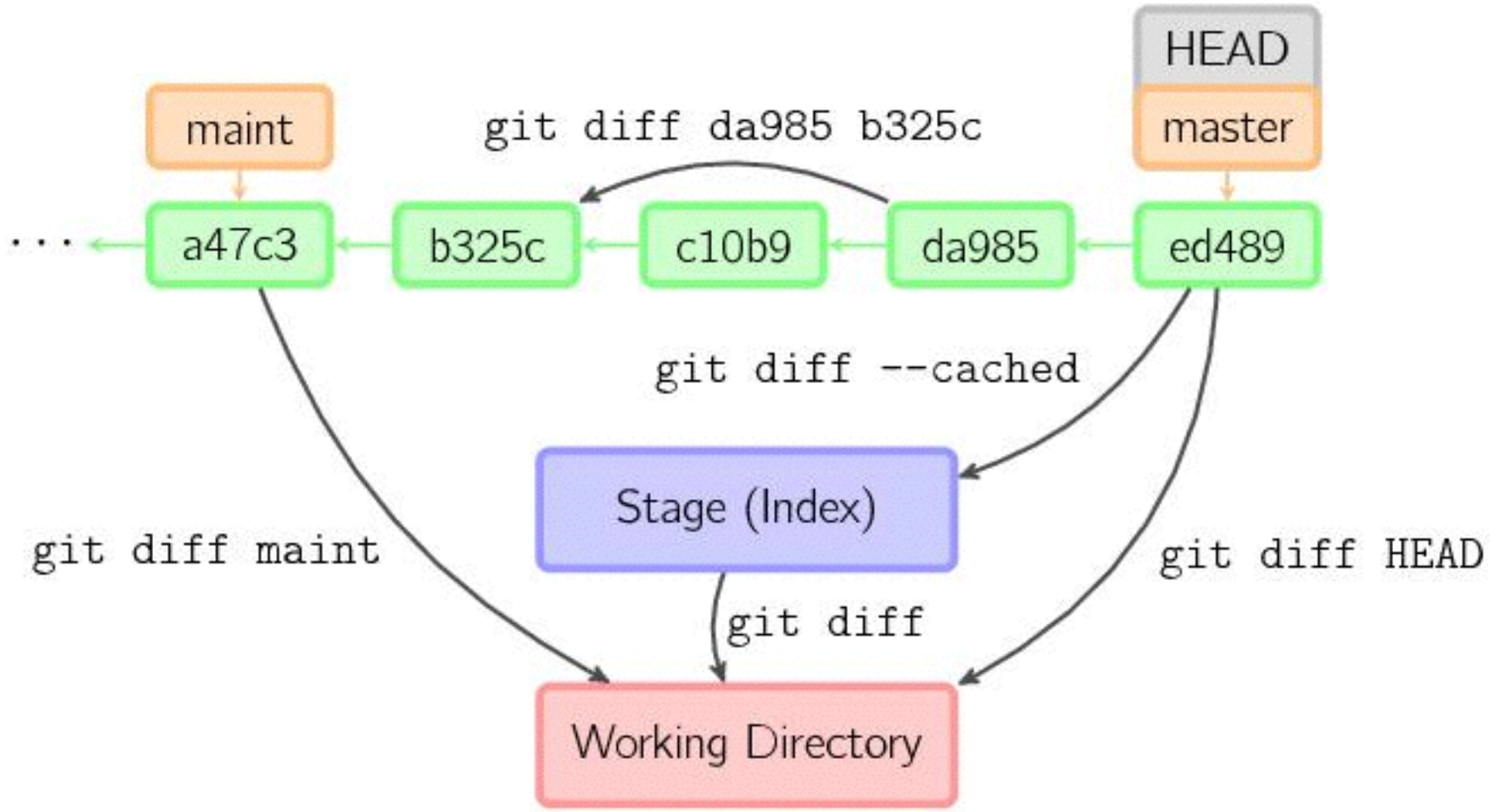
git checkout -- files 把文件从暂存区域复制到工作目录，用来丢弃本地修改。

你可以用 git reset -p, git checkout -p, or git add -p进入交互模式。

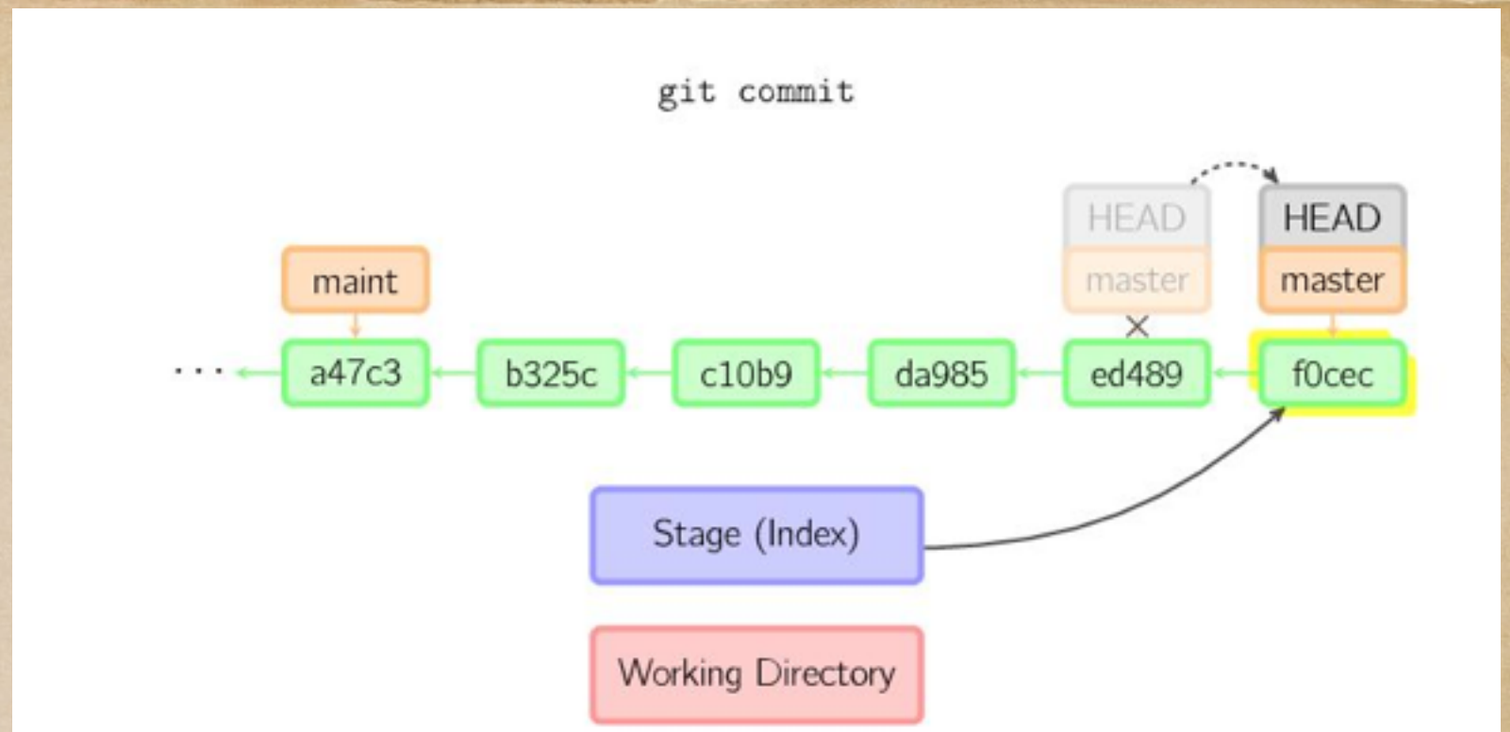


- git commit -a 相当于运行 git add 把所有当前目录下的文件加入暂存区域再运行。git commit.
- git commit files 进行一次包含最后一次提交加上工作目录中文件快照的提交。并且文件被添加到暂存区域。
- git checkout HEAD -- files 回滚到复制最后一次提交。

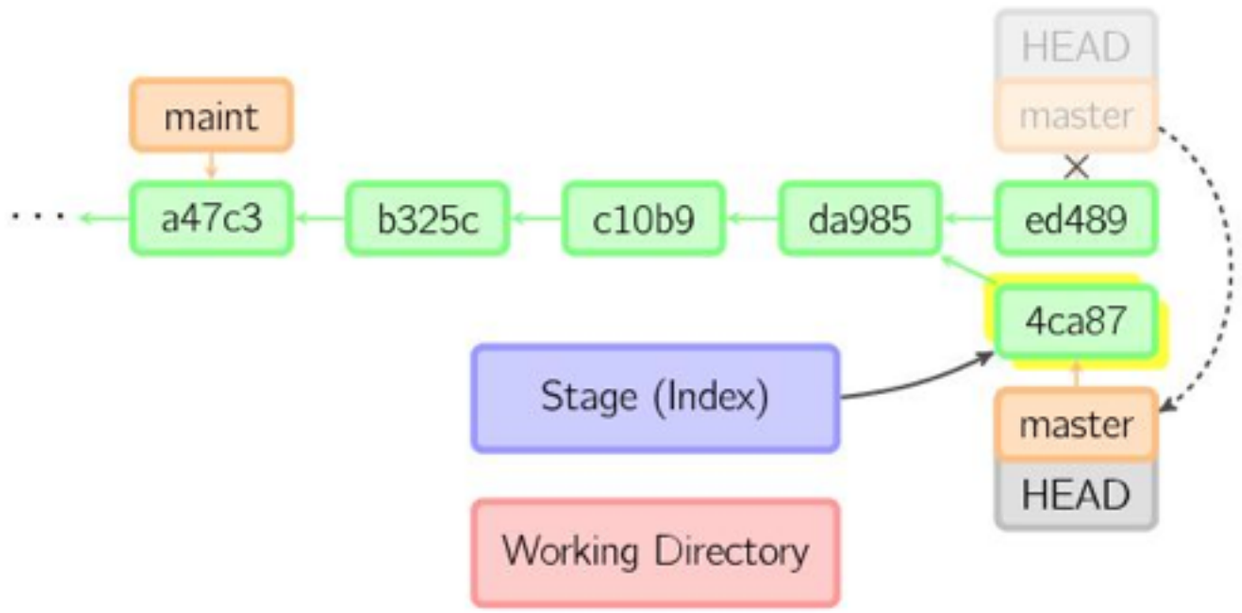
git diff


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

git commit



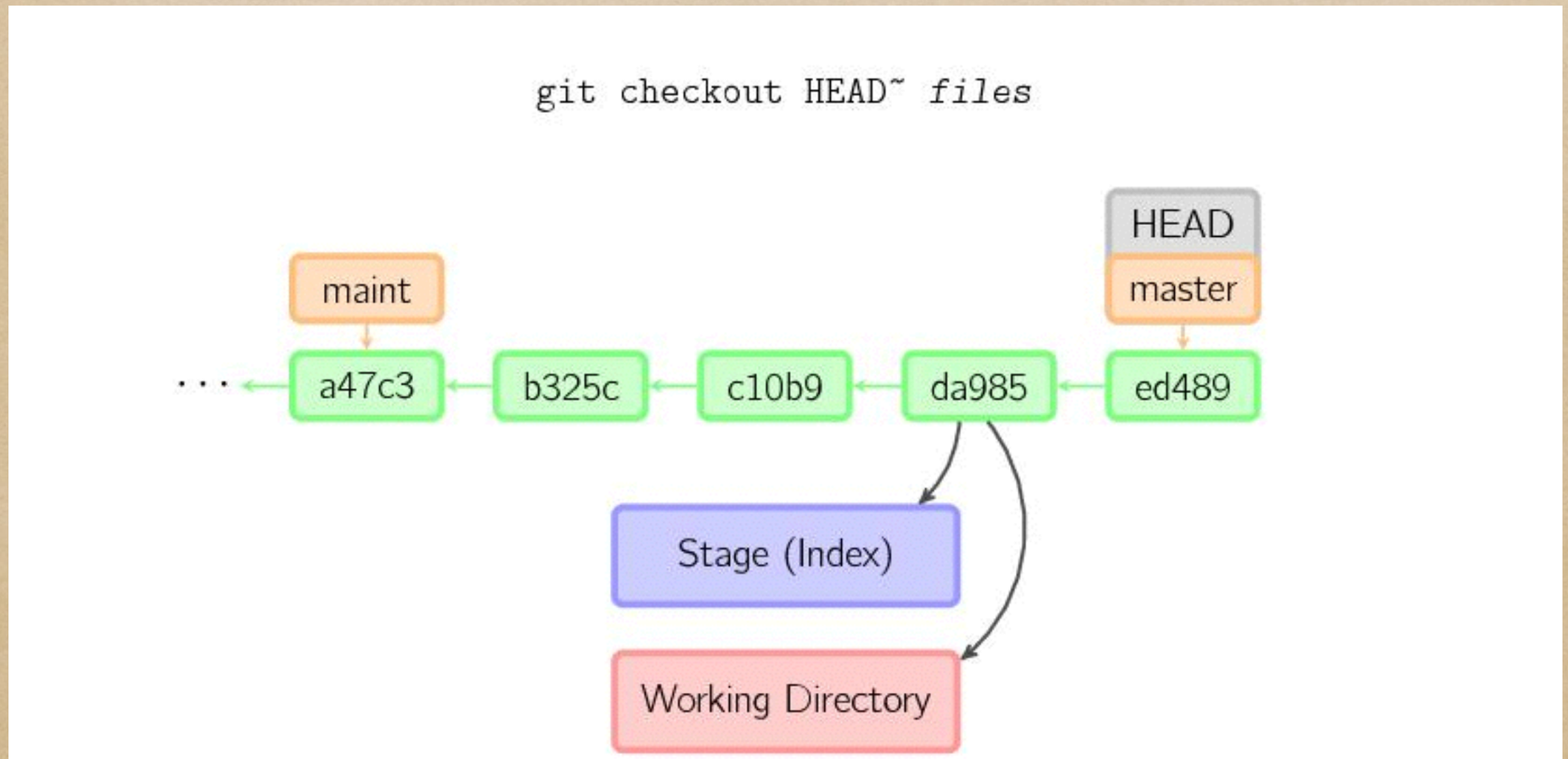

```
git commit --amend
```



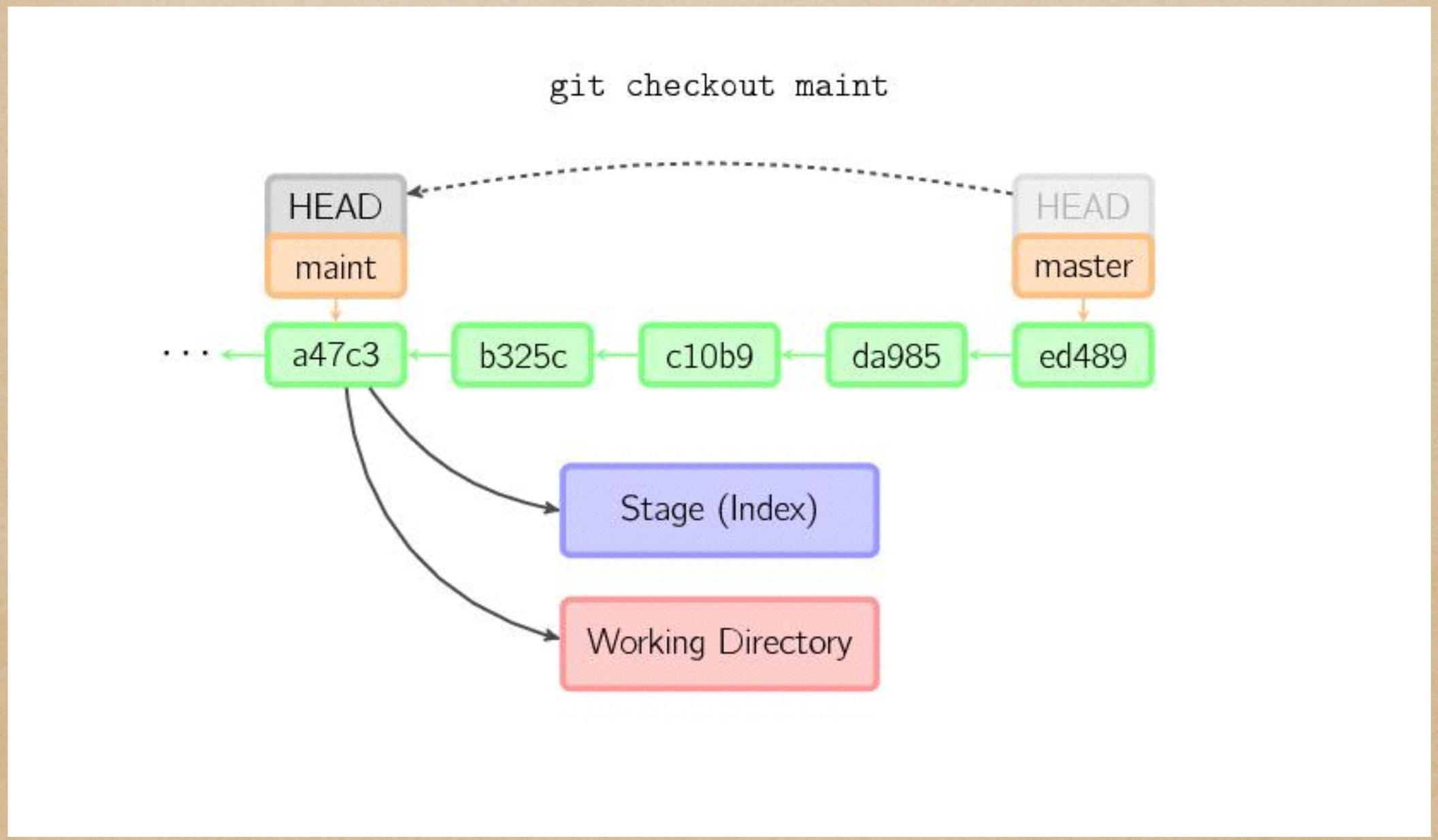
git checkout

checkout 命令通常用来从仓库中取出文件，或者在分支中切换。

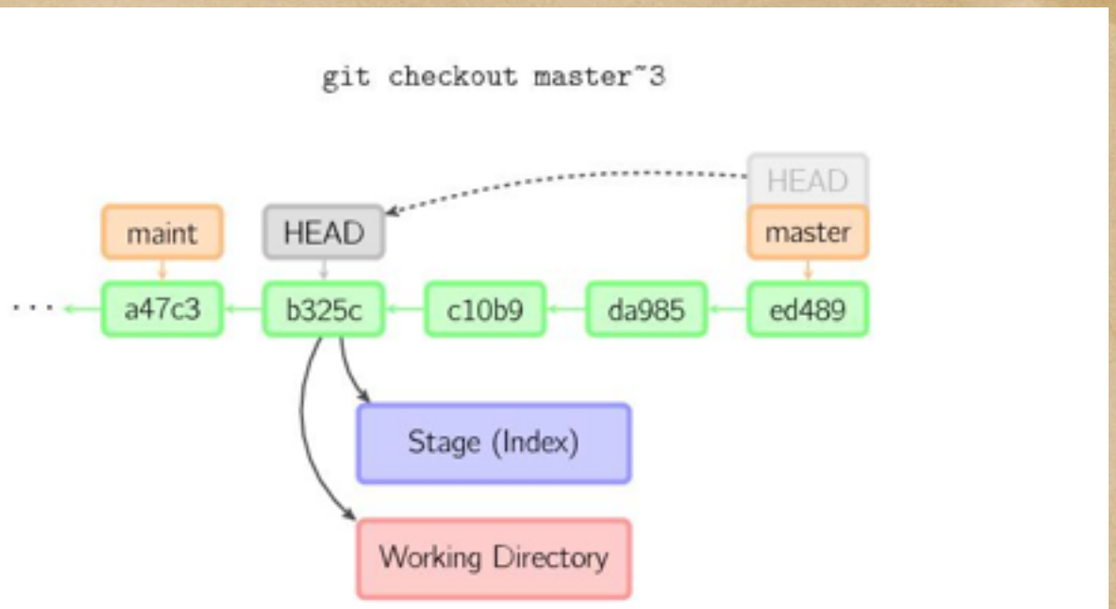
checkout 命令让 git 把文件复制到工作目录和暂存区域。比如git checkout HEAD~ foo.c把文件从foo.c提交节点HEAD~ (当前提交节点)复制到工作目录并且生成索引。注意当前分支没有变化。

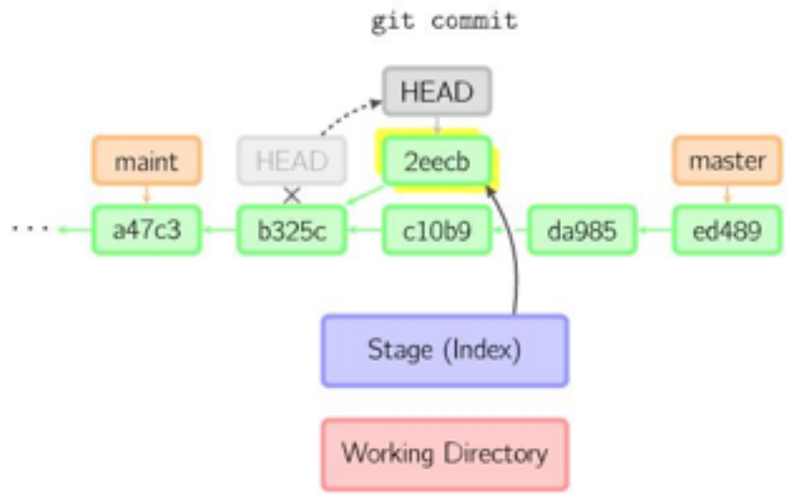


如果没有指定文件名，而是一个本地分支，那么将切换到那个分支去。同时把索引和工作目录切换到那个分支对应的状态。



如果既没有指定文件名，也没有指定分支名，而是一个标签、远程分支、SHA-1值或者是像master~3类似的东西，就得到一个匿名分支，称作detached HEAD。这样可以很方便的在历史版本之间互相切换。但是，这样的提交是完全不同的，详细的在下面。

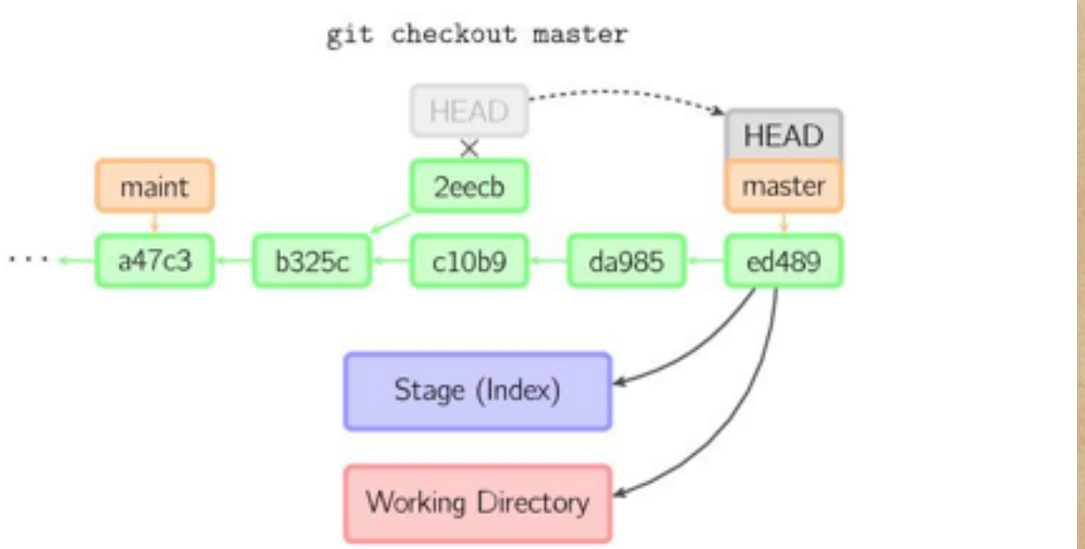




HEAD是分离的时候， 提交可以正常进行， 但是没有更新已命名的分支. 。(可以看作是匿名分支。)

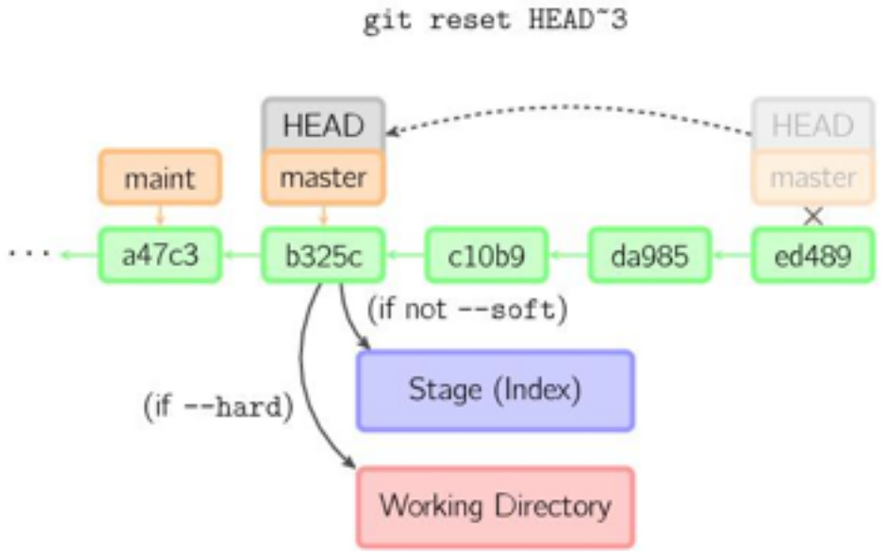
如果此时切换到别的分支，那么所作的工作会全部丢失。注意这个命令之后就不存在2eecb了。

如果你想保存当前的状态，可以用这个命令创建一个新的分支: git checkout -b name。



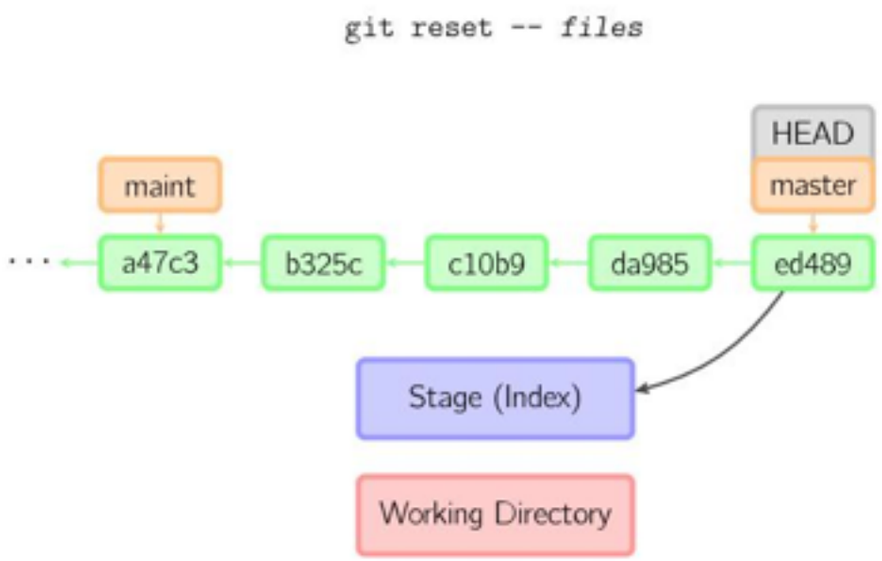
git reset

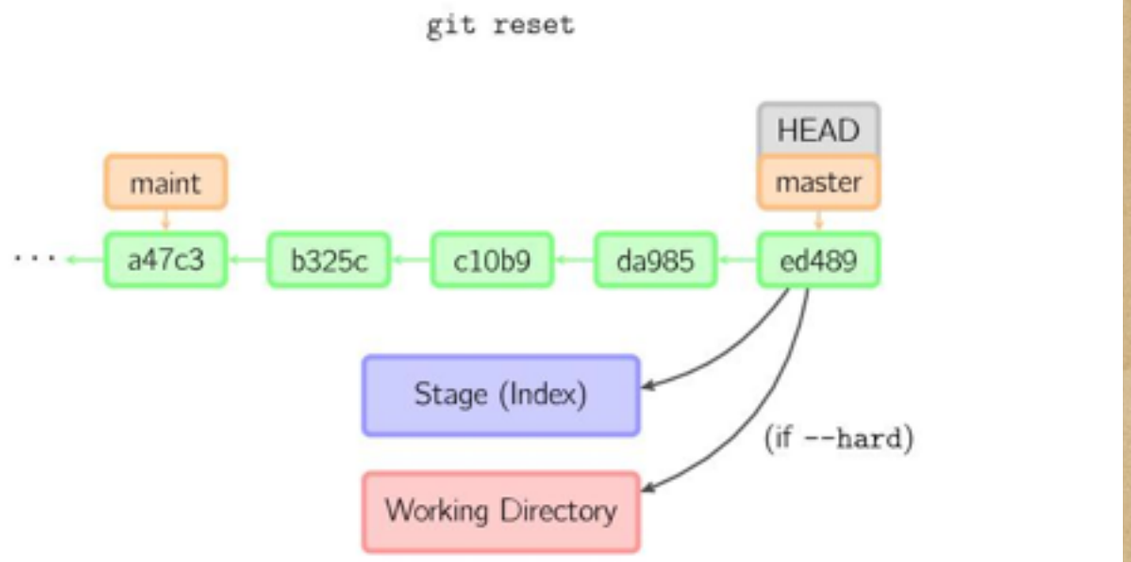
reset 命令把当前分支指向另一个位置，并且有选择的变动工作目录和索引。也用来在从历史仓库中复制文件到索引，而不动工作目录。



如果不给选项，那么当前分支指向到那个提交。如果用--hard选项，那么工作目录也更新，如果用--soft选项，那么都不变。

如果没有给出提交点的版本号，那么默认用 HEAD。这样，分支指向不变，但是索引会回滚到最后一次提交，如果用--hard选项，工作目录也同样。

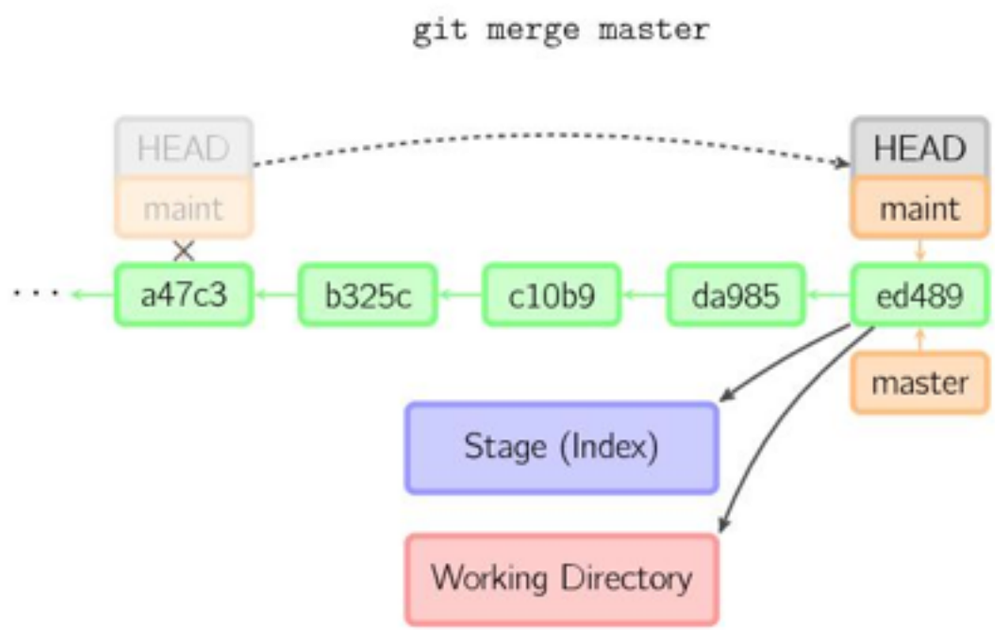




如果给了文件名(或者 -p选项)，那么工作效果和带文件名的 checkout 差不多，除了索引被更新。

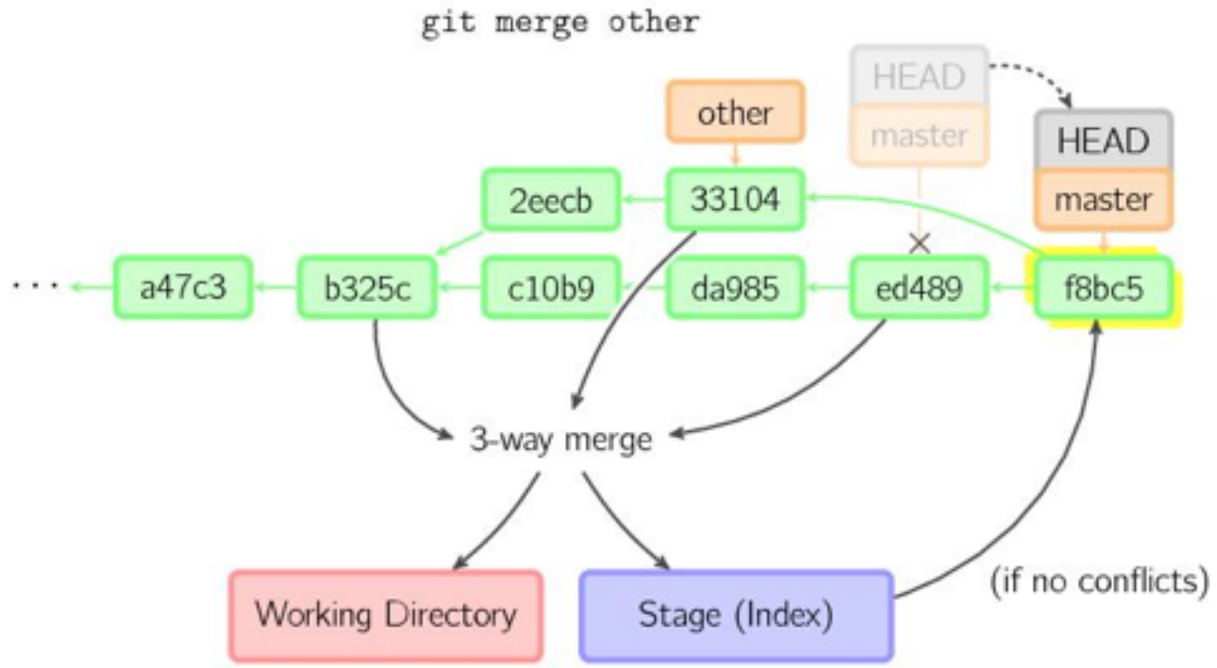
git merge

merge 命令把不同分支合并起来。合并前，索引必须和当前提交相同。



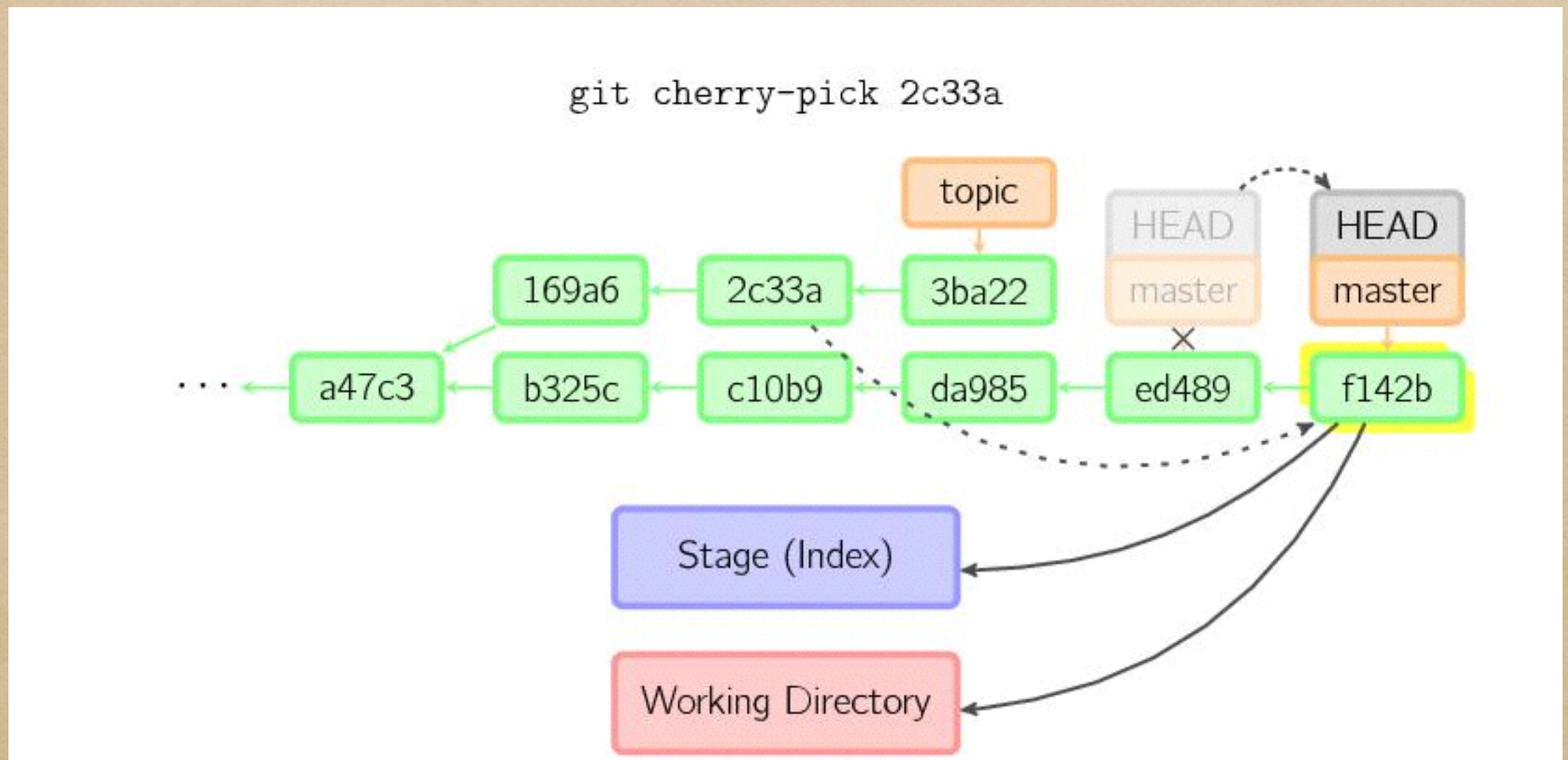
如果另一个分支是当前提交的祖父节点，那么合并命令将什么也不做。另一中情况是如果当前提交是另一个分支的祖父节点，就导致 fast-forward 合并。指向只是简单的移动，并生成一个新的提交。

否则就是一次真正的合并。默认把当前提交(ed489 如下所示)和另一个提交(33104)以及他们的共同祖父节点(b325c)进行一次三方合并。结果是先保存当前目录和索引，然后和父节点33104一起做一次新提交。



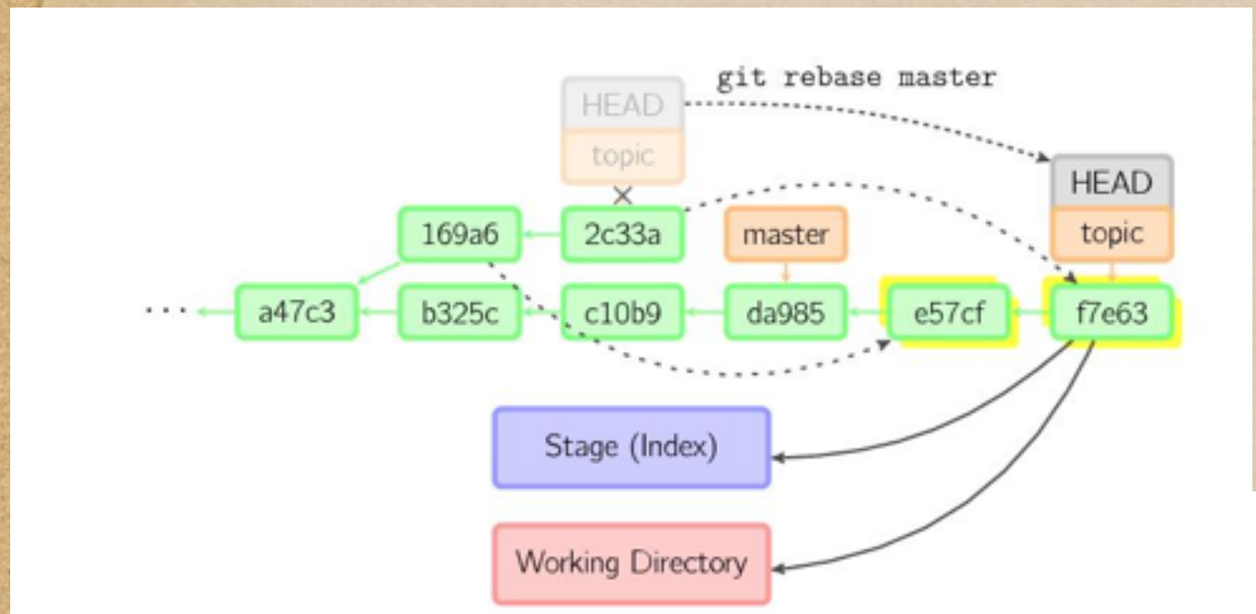
git cherry-pick

cherry-pick 命令"复制"一个提交节点并在当前复制做一次完全一样的新提交。

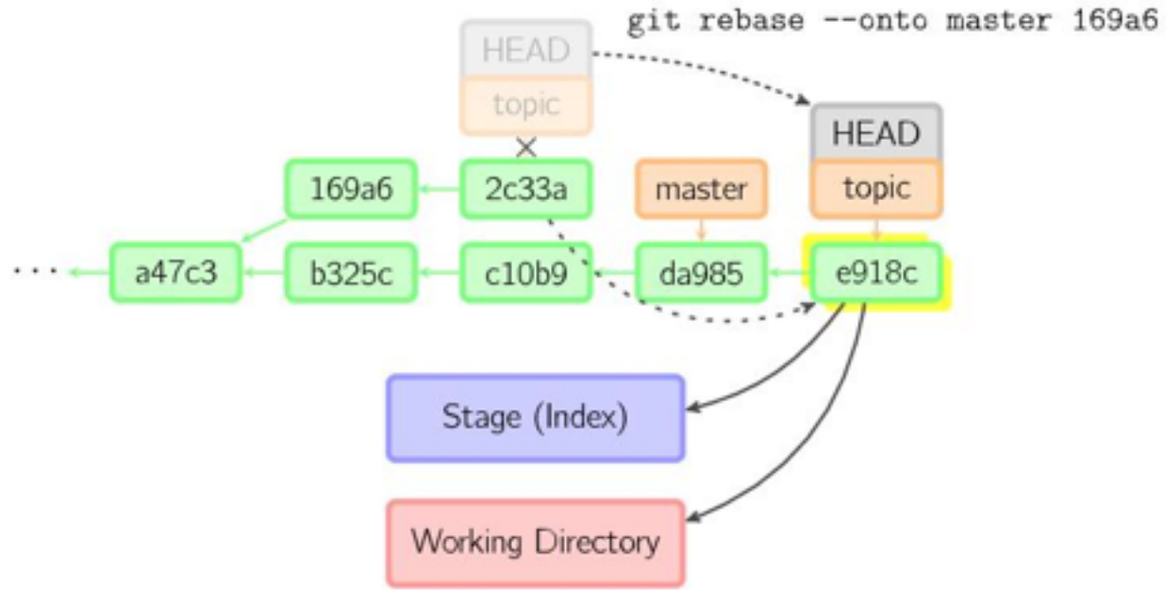


git rebase

衍合是合并命令的另一种选择。合并把两个父分支合并进行一次提交，提交历史不是线性的。衍合在当前分支上重演另一个分支的历史，提交历史是线性的。 本质上，这是线性化的自动的 cherry-pick



上面的命令都在topic分支中进行，而不是master分支，在master分支上重演，并且把分支指向新的节点。注意旧提交没有被引用，将被回收。
要限制回滚范围，使用--onto选项。下面的命令在master分支上重演当前分支从169a6以来的最近几个提交，即2c33a。



Thanks