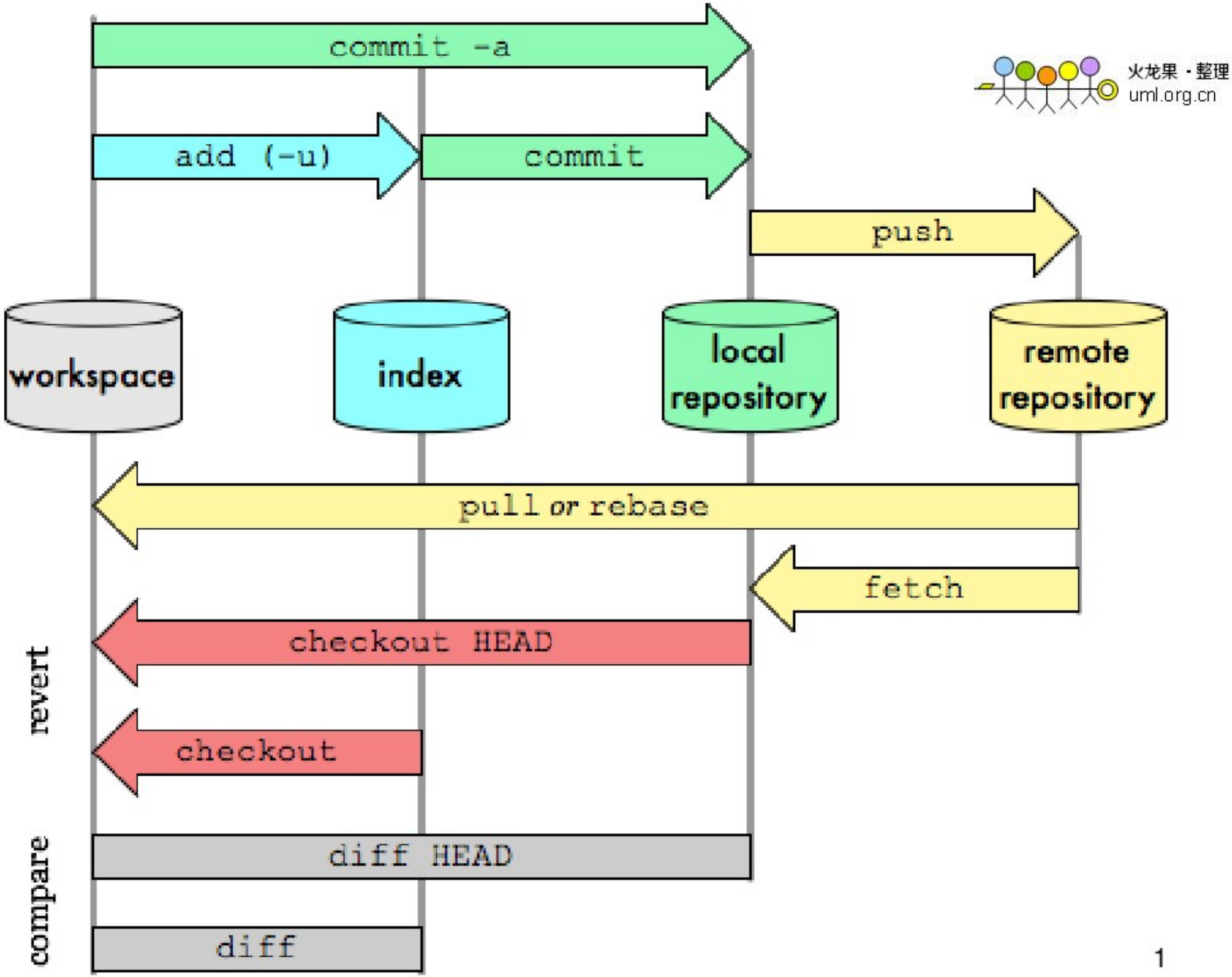

commit -a
add (-u)
commit
push
workspace
index
local repository
remote repository
pull or rebase
fetch
checkout HEAD
checkout
diff HEAD
diff
revert
compare
火龙果 · 整理
uml.org.cn

GIT--大智若愚

火龙果 · 整理
uml.org.cn

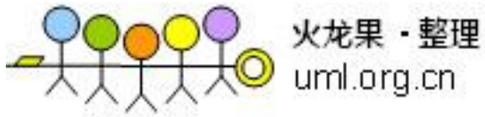
- Subversion、CVS、Perforce、Mercurial 等等，他们使用 **“增量文件系统”** （Delta Storage systems），就是说它们存储每次提交(commit)之间的差异。
- Git正好与之相反，它会把你的每次提交的文件的**全部内容**（snapshot）都会记录下来。这会是在使用Git时的一个很重要的理念。

GIT--大智若愚

火龙果 · 整理
uml.org.cn

- Git的设计理念是，任何文件，只要有任何改动，哪怕是一个字节也好，都会重新创建一个副本(即之前提到的blob)对象，若一个文件被修改了4次就会有4个副本，每一个都是独立的，都与每次提交产生commit对象所管理。
- 乍一看Git的这种设计非常消耗硬盘，确实是这样，貌似非常愚钝！但当今的计算机时代，硬盘的低廉和容量的飞速扩大，让这磁盘空间的消耗变得越来越微不足道。Git的设计者Linus就充分利用这一点，牺牲了磁盘空间，换取了无限控制上的灵活和管理的高效。

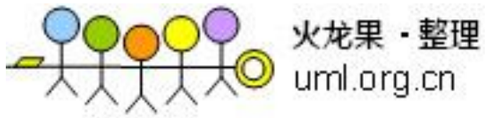
GIT对象模型



- **对象**

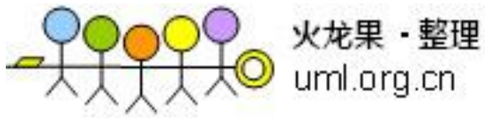
每个对象(object) 包括三个部分：类型，大小和内容。大小就是指内容的大小，内容取决于对象的类型，有四种类型的对象："blob"、"tree"、 "commit" 和"tag"。

GIT对象模型



- 对象
  - “**blob**”用来存储文件数据，通常是一个文件。
  - “**tree**”有点像一个目录，它管理一些“tree”或是“blob”（就像文件和子目录）
  - 一个“**commit**”只指向一个"tree"，它用来标记项目某一个特定时间点的状态。它包括一些关于时间点的元数据，如时间戳、最近一次提交的作者、指向上次提交（commits）的指针等等。
  - 一个“**tag**”是来标记某一个提交(commit) 的方法。

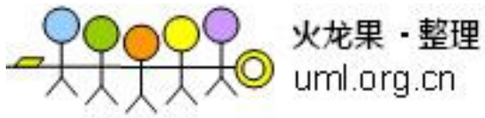
GIT对象模型



- **对象名-SHA**
  - 所有用来表示项目历史信息的文件,是通过一个40个字符的（40-digit）“对象名”来索引的，对象名看起来像这样:

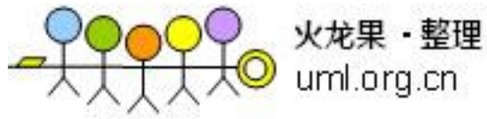
    6ff87c4664981e4397625791c8ea3bbb5f2279a3

GIT对象模型



- **对象名-SHA**
  - 这样做会有几个好处：
    - Git只要比较对象名，就可以很快的判断两个对象是否相同。
    - 因为在每个仓库（repository）的“对象名”的计算方法都完全一样，如果同样的内容存在两个不同的仓库中，就会存在相同的“对象名”。
    - Git还可以通过检查对象内容的SHA1的哈希值和“对象名”是否相同，来判断对象内容是否正确。

GIT对象模型



- **Blob对象**
  - 一个"blob对象"就是一块**二进制数据**，它**没有**指向任何东西或有任何其它属性，甚至连文件名都没有.
  - 因为blob对象内容全部都是数据，如两个文件在一个目录树（或是一个版本仓库）中有同样的数据内容，那么它们将会**共享**同一个blob对象。Blob对象和其所对应的文件所在路径、文件名是否改被更改都**完全没有关系**。
  - 可以使用**git show**命令查看内容

5b1d3..

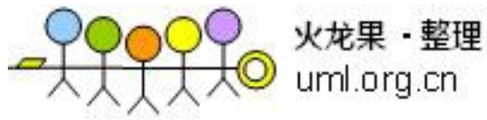
blob	size

```
#ifndef REVISION_H
#define REVISION_H

#include "parse-options.h"

#define SEEN            (1u<<0)
#define UNINTERESTING   (1u
#define TREESAME  (1u<<2)
```

GIT对象模型

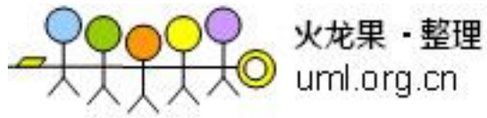


- **Tree对象**
  - 一个tree对象有一串(bunch)**指向blob对象**或是其它tree对象的指针，它一般用来表示内容之间的目录层次关系。
  - 一个tree对象可以指向(reference):
    - 一个包含文件内容的blob对象,
    - 其它包含某个子目录内容的其它tree对象.

c36d4..

tree		size
blob	5b1d3	README
tree	03e78	lib
tree	cdc8b	test
blob	cba0a	test.rb
blob	911e7	xdiff

GIT对象模型



- **Tree对象**
  - Tree对象、blob对象和其它所有的对象一样，都用其内容的SHA1哈希值来命名的；只有当两个tree对象的内容完全相同（包括其所指向所有子对象）时，它的名字才会一样，反之亦然。这样就能让Git仅仅通过比较两个相关的tree对象的名字是否相同，来快速的判断其内容是否不同。
  - 可以使用**git show/git ls-tree**命令查看对象的内容

c36d4..

tree		size
blob	5b1d3	README
tree	03e78	lib
tree	cdc8b	test
blob	cba0a	test.rb
blob	911e7	xdiff

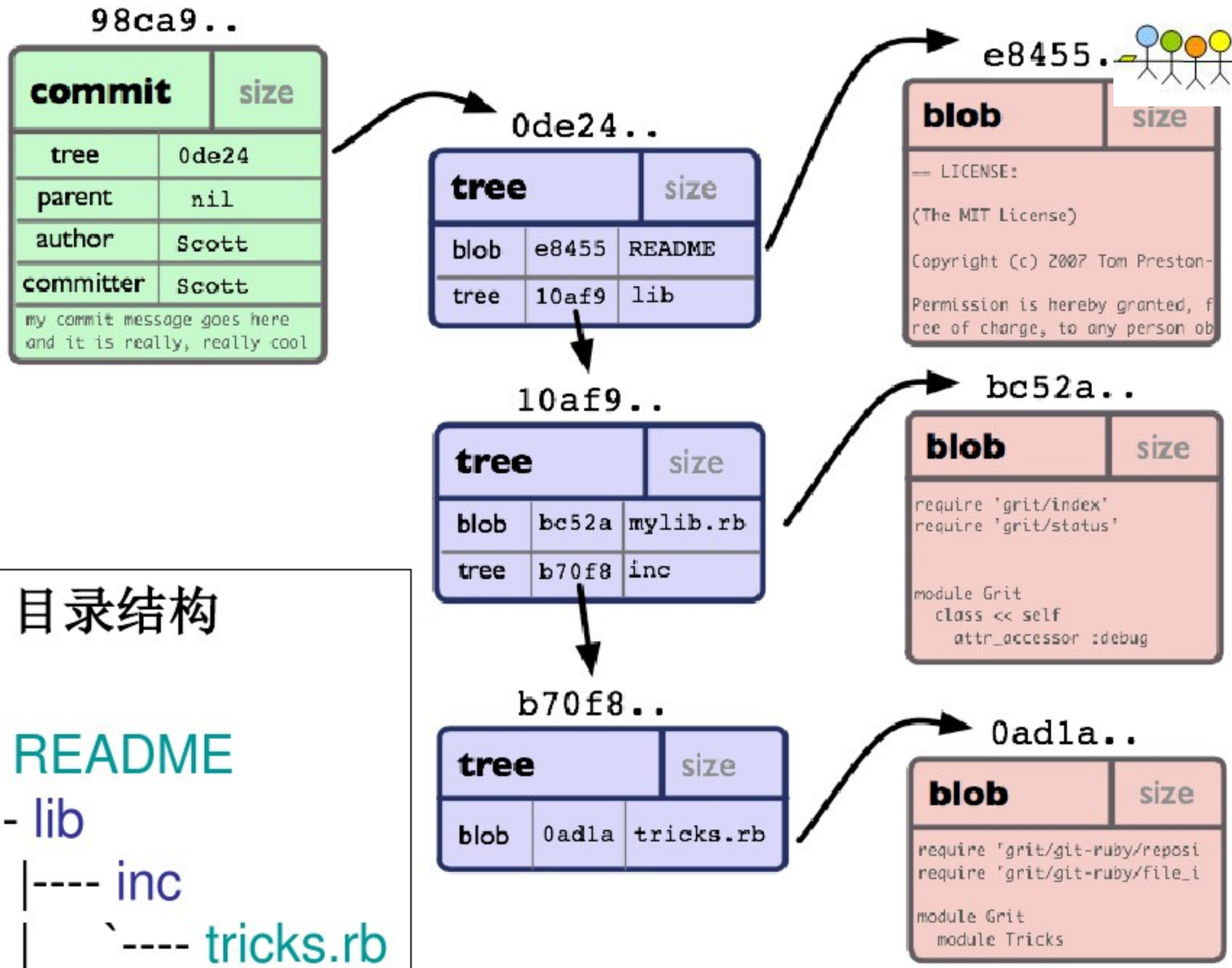
GIT对象模型

火龙果 · 整理
uml.org.cn

- **Commit对象**
  - "commit对象"指向**一个"tree对象"**, 并且带有相关的描述信息.
  - 一个提交(commit)由以下的部分组成:
    - **一个 tree对象**: tree对象的SHA1签名, 代表着目录在某一时间点的内容.
    - **父对象 (parent(s))**: 提交(commit)的SHA1签名代表着当前提交前一步的项目历史. 上面的那个例子就只有一个父对象; 合并的提交(merge commits)可能会有**不只一个**父对象.
    - **作者** : 做了此次修改的人的名字, 还有修改日期.
    - **提交者**（committer): 实际创建提交(commit)的人的名字, 同时也带有提交日期. TA可能会和作者不是同一个人; 例如作者写一个补丁(patch)并把它用邮件发给提交者, 由他来创建提交(commit).
    - **注释** 用来描述此次提交.

ae668..

commit	size
tree	c4ec5
parent	a149e
author	Scott
committer	Scott
my commit message goes here and it is really, really cool	



火龙果 · 整理
uml.org.cn

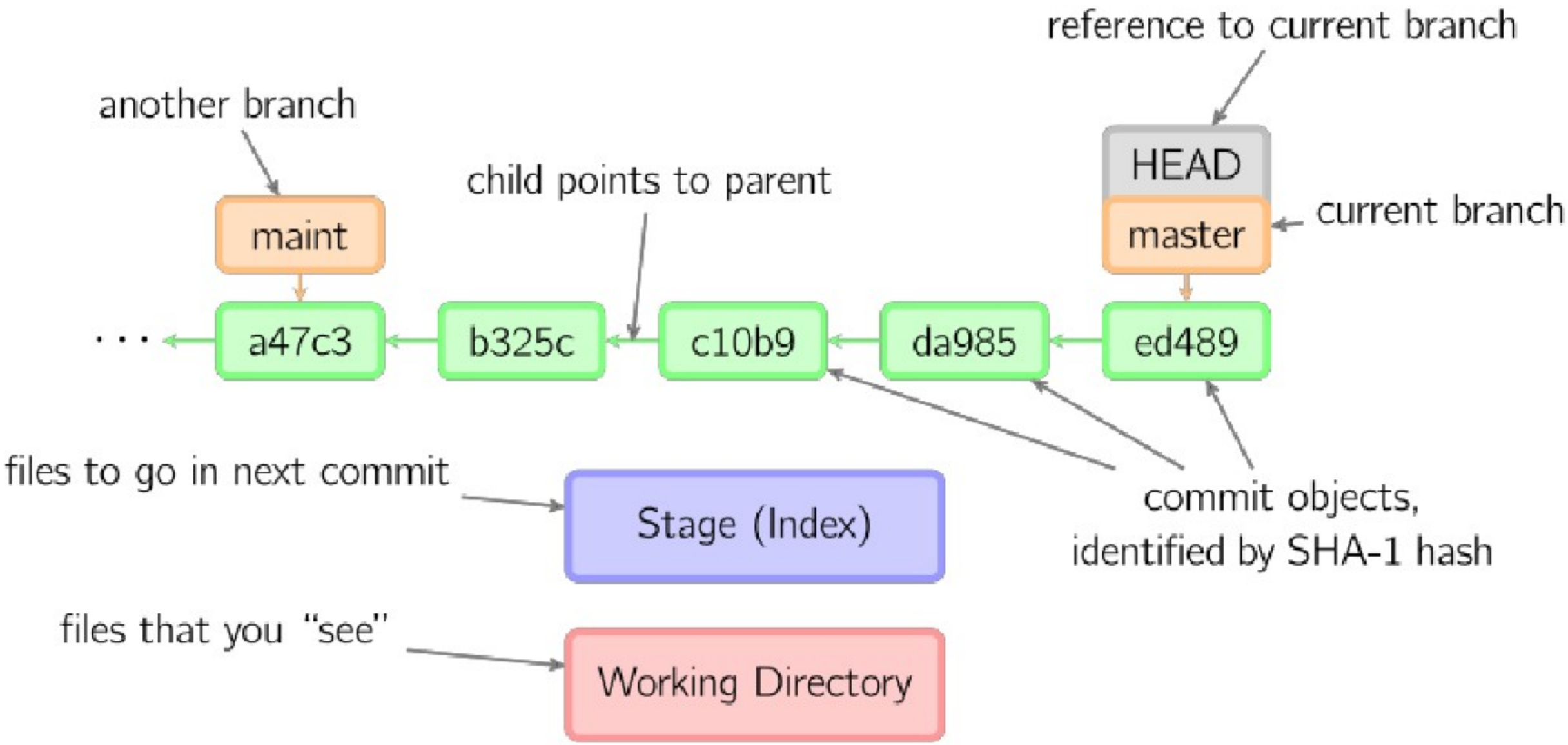
目录结构

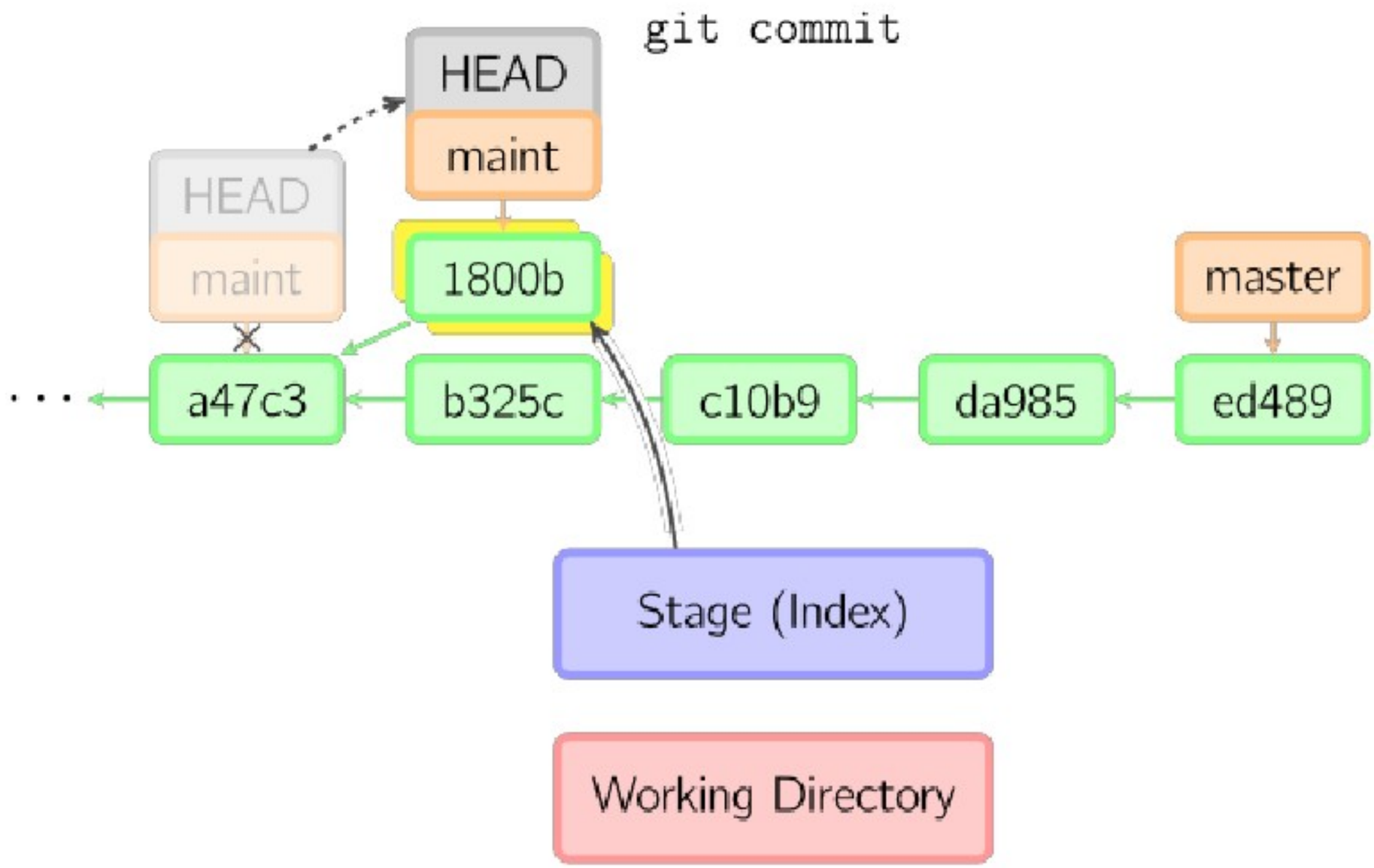
```
.
|--- README
`---- lib
      |---- inc
      |       `---- tricks.rb
      `---- mylib.rb
```

监控 Git 生成的文件

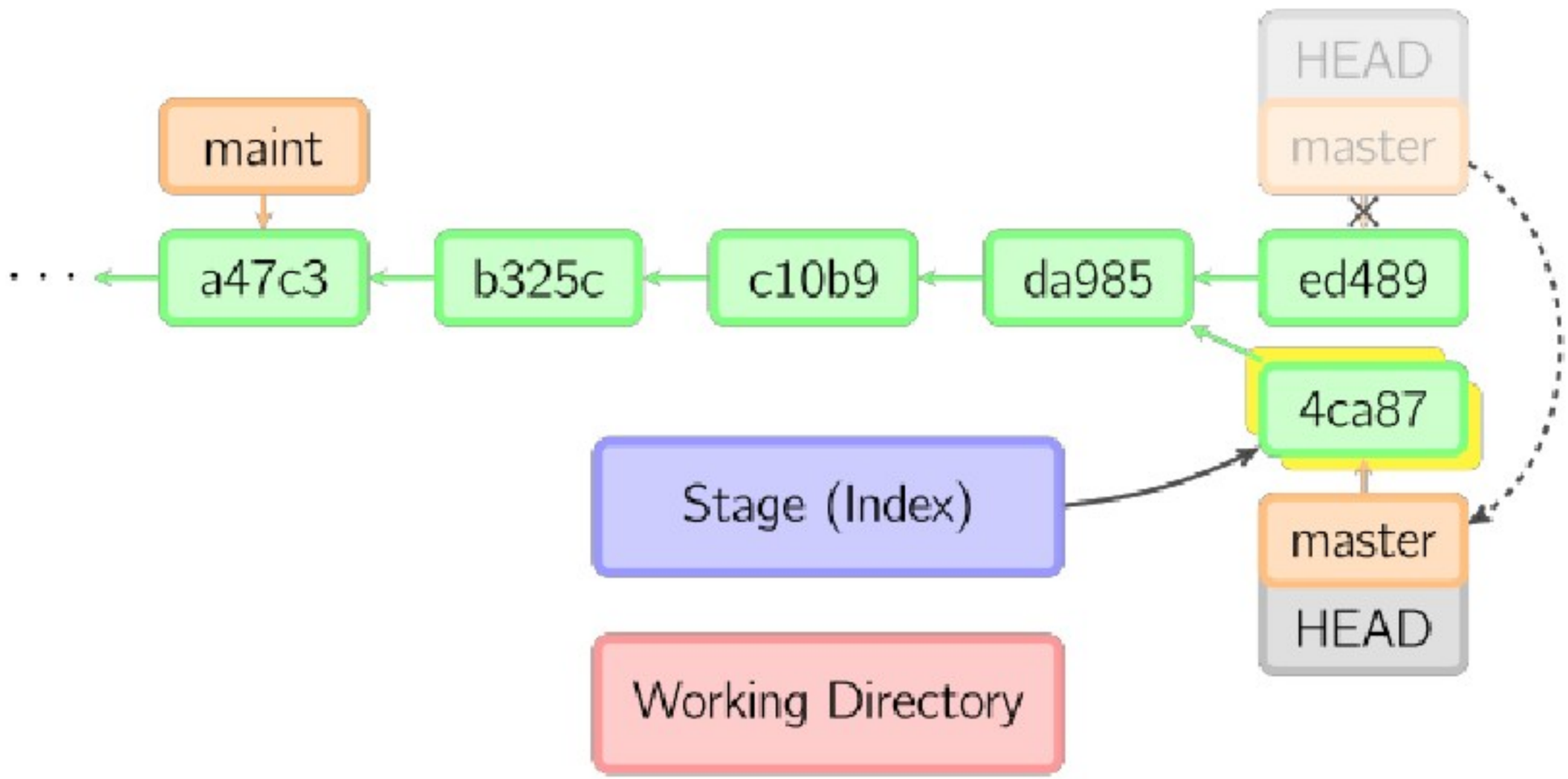
火龙果 · 整理
uml.org.cn

- 工具：Inotify
  - **inotifywait** -rme modify,create,delete /test
- 执行下面的命令，监控Git生成的对象和其他文件
  - git init
  - git status
  - git add
  - git commit
  - git branch -b new
  - git merge

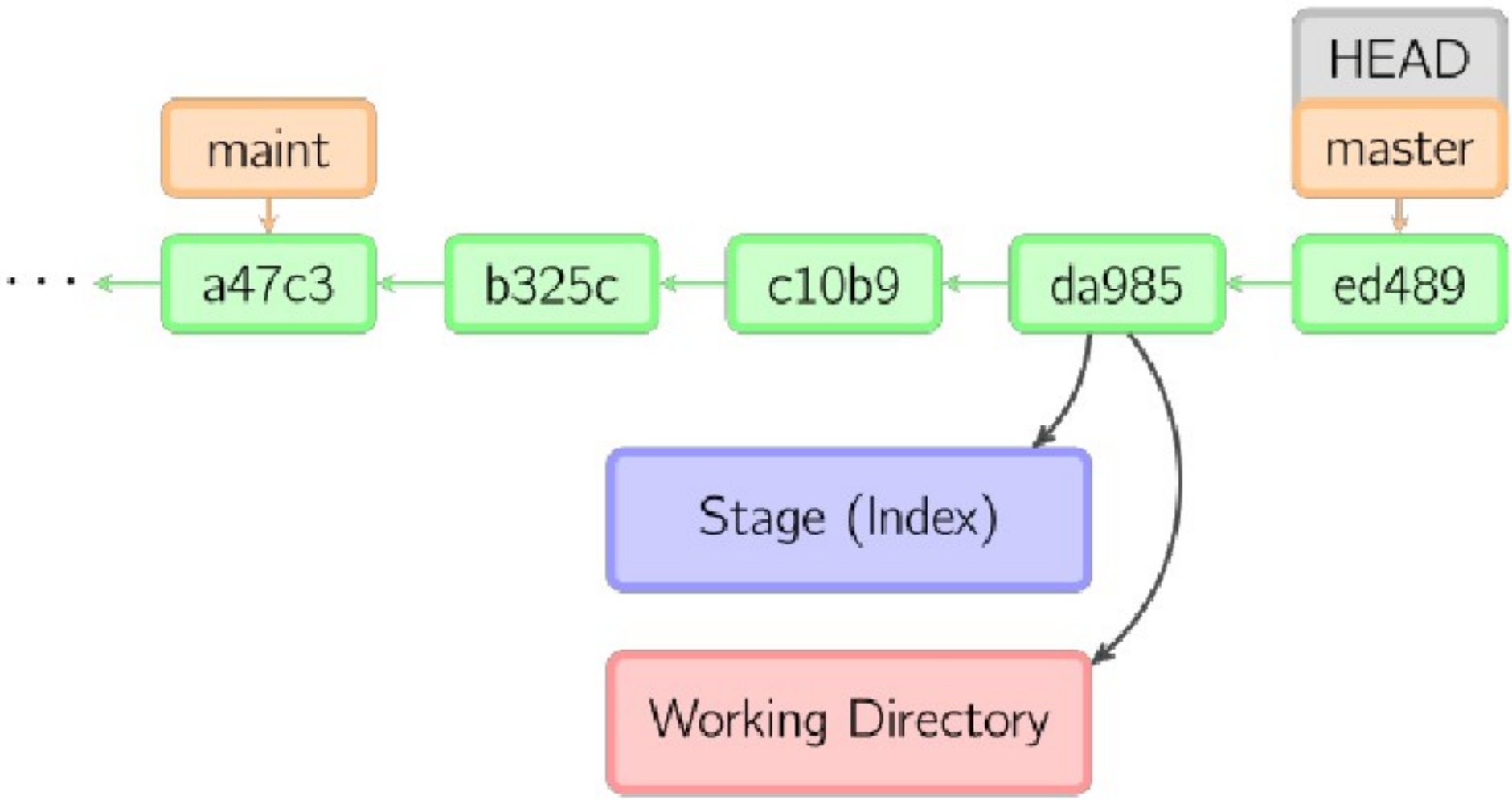

reference to current branch
another branch
child points to parent
HEAD
current branch
maint
master
· · ·
a47c3
b325c
c10b9
da985
ed489
files to go in next commit
Stage (Index)
commit objects,
identified by SHA-1 hash
files that you "see"
Working Directory

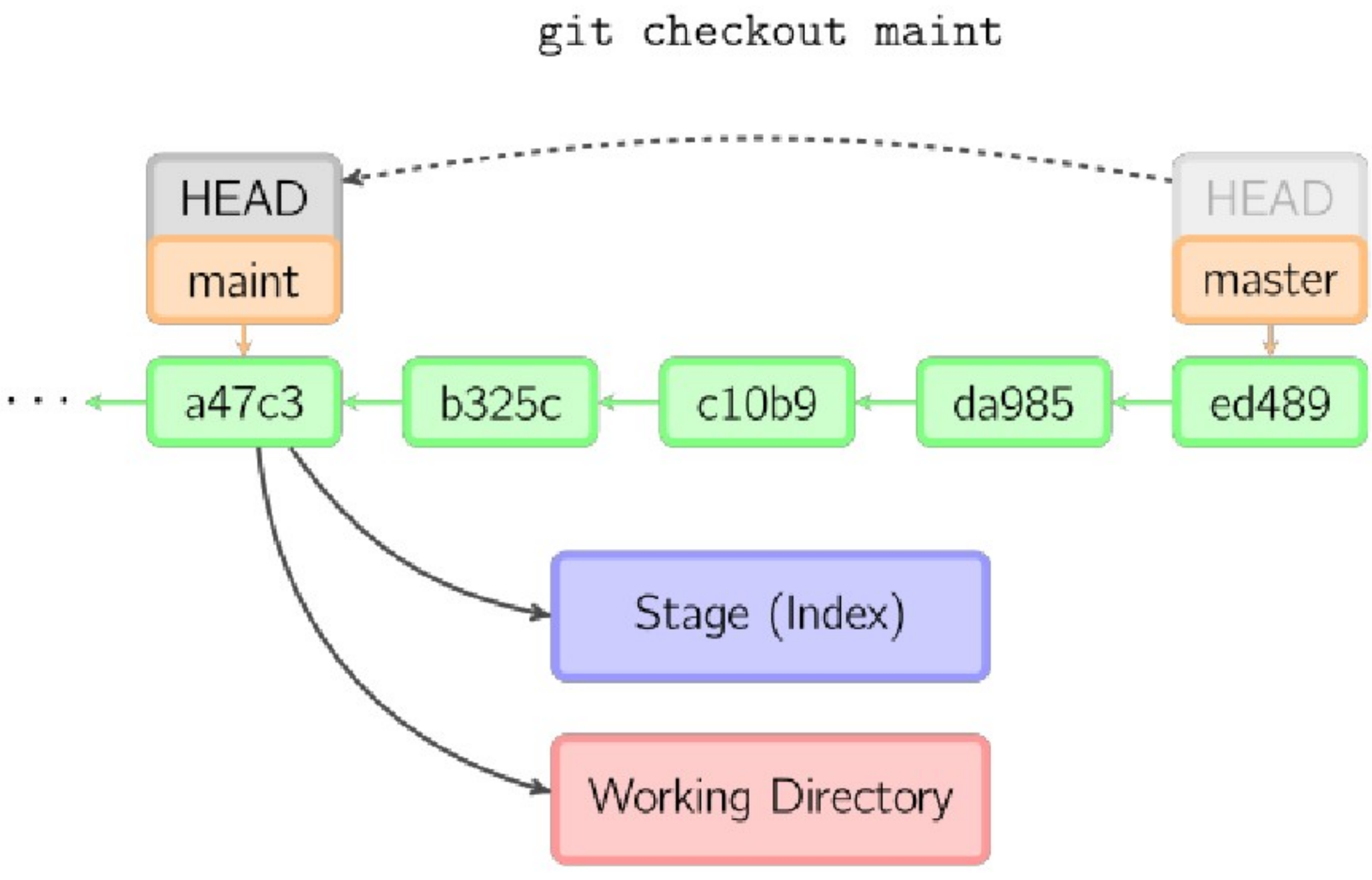

git commit
HEAD
maint
HEAD
maint
1800b
master
...
a47c3
b325c
c10b9
da985
ed489
Stage (Index)
Working Directory

git commit --amend

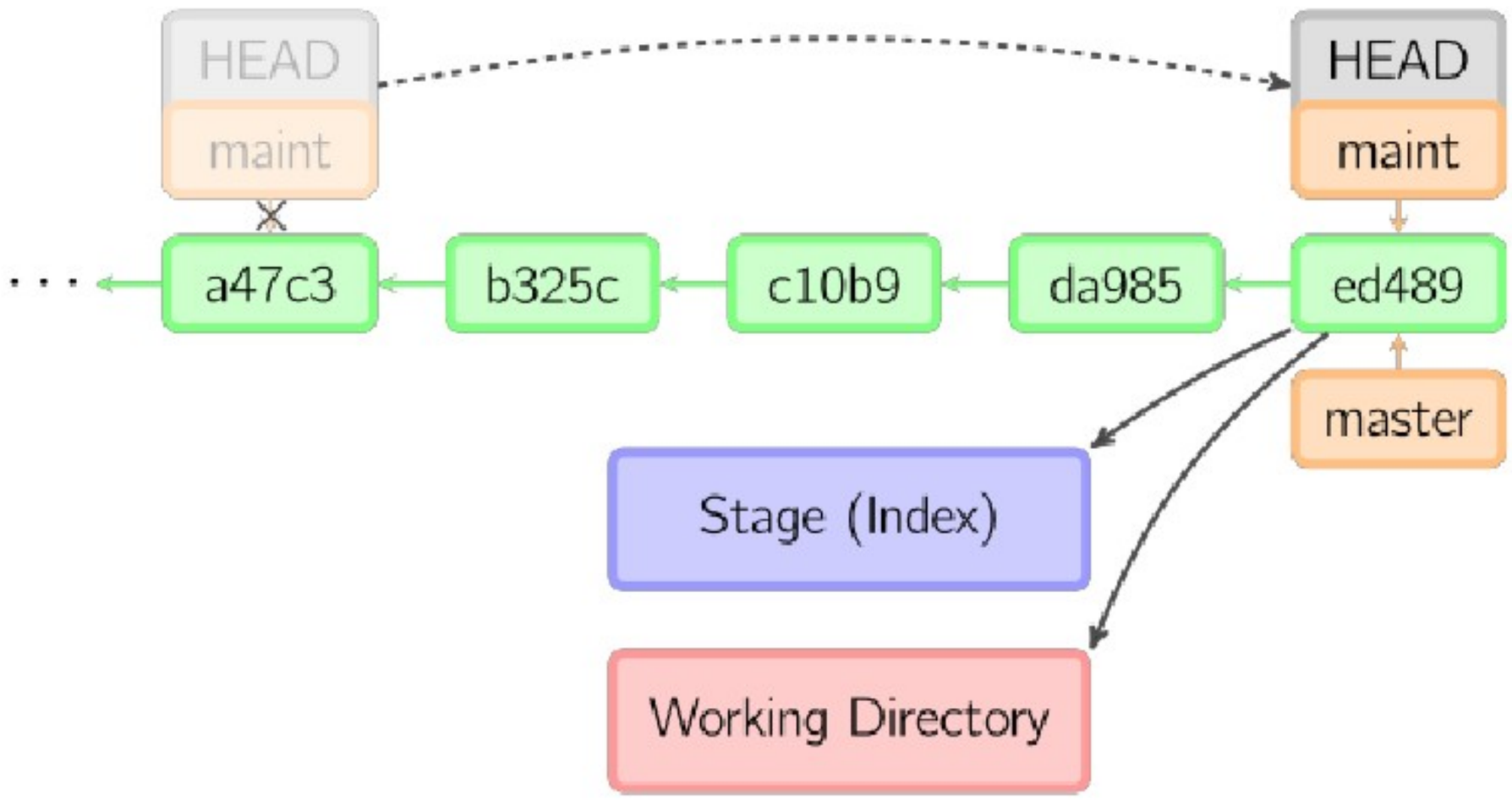
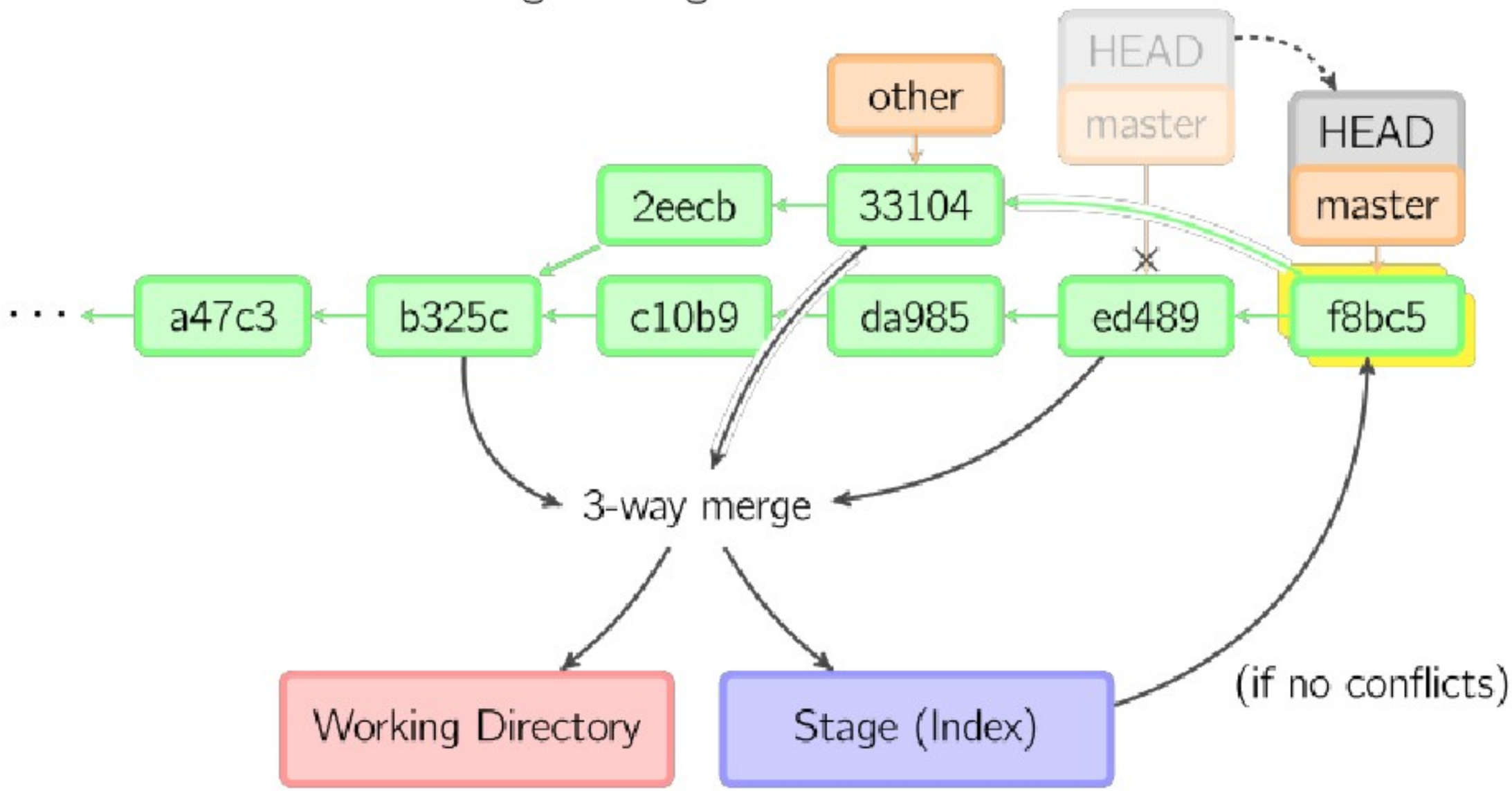


git checkout HEAD~ *files*




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

git merge master



git merge other