**Git原理 - 详解.git目录（二）**

剩下的四个文件很重要：HEAD 文件、（尚待创建的）index 文件，和 objects 目录、refs 目录。

它们都是 Git 的核心组成部分。

* objects 目录存储所有数据内容；
* refs 目录存储指向提交对象的指针，包括：分支、远程仓库和标签等；
* HEAD 文件指向当前工作的分支；
* index 文件保存暂存区的信息。

**6、objects目录**

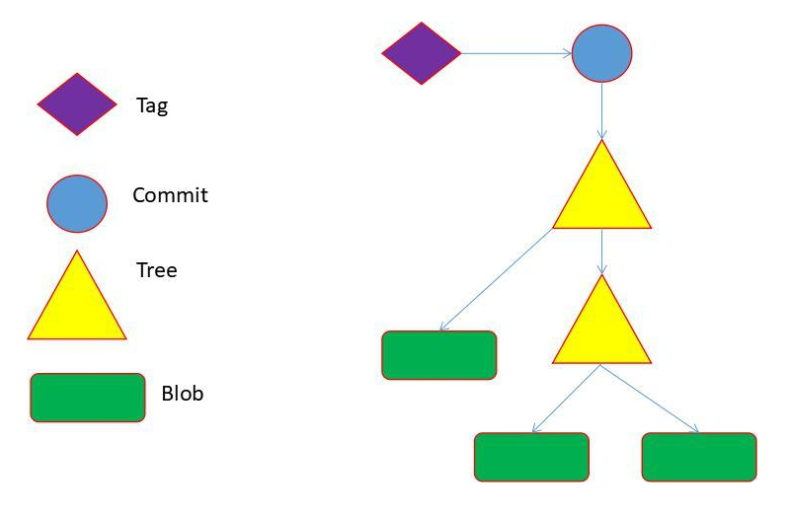
objects目录用于存储Git中的所有对象。

可以看到在一个新的Git本地版本库中， Git 对 objects 目录进行了初始化，并创建了 pack 和 info 子目录，但均为空。

数据对象、树对象、提交对象、标签对象，均以单独文件的形式保存在 .git/objects 目录下。

虽然数据对象的内容几乎可以是任何东西，但提交对象和树对象的内容却有各自固定的格式。

Git中四个对象关系如下：



|  |
| --- |
| Pack文件夹说明:  Git有一个自我梳理的过程，如果Git仓库中松散的文件比较多的话，pack会对这些文件进行打包，打了包的文件是放在pack文件夹里边的。  Git 打包对象时，会查找命名及大小相近的文件，并只保存文件不同版本之间的差异内容。 |

（具体说明，可以看前面**Git对象**相关的文章）

**7、refs目录**

如果你对仓库中从一个提交（比如 1a410e）开始往前的历史感兴趣，那么可以运行 git log 1a410e 这样的命令来显示历史提交记录，不过你需要记得 1a410e 是你查看历史的起点提交。 如果我们有一个文件来保存 SHA-1 值，而该文件有一个简单的名字， 然后用这个名字指针来替代原始的 SHA-1 值的话，会更加简单。

在 Git 中，这种简单的名字被称为“引用（references，或简写为 refs）”。 你可以在 .git/refs 目录下找到这类含有 SHA-1 值的文件。

.git/refs 目录内容如下：

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (dev) $ ll .git/refs/ total 0 drwxr-xr-x 1 L 197121 0 4月 22 14:42 heads/ drwxr-xr-x 1 L 197121 0 4月 11 14:50 tags/ |

可以看到有两个目录，一个是heads/目录，另一个是tags/目录。

（下面是重点）

**（1）heads/目录**

heads/目录中存放的是当前版本库中的分支。每创建一个分支，在heads/目录中就多出一个，以分支名来命名的文件。如下所示

|  |
| --- |
| Bash # 1.查看.git/refs/heads/目录中的文件 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ ll .git/refs/heads/ total 2 -rw-r--r-- 1 L 197121 41 4月 22 14:38 master -rw-r--r-- 1 L 197121 41 4月 22 14:42 test  # 2.查看当前版本库中的分支 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ git branch \* master  test |

我们可以理解，.git/refs/heads/目录中存放的是版本库的分支。

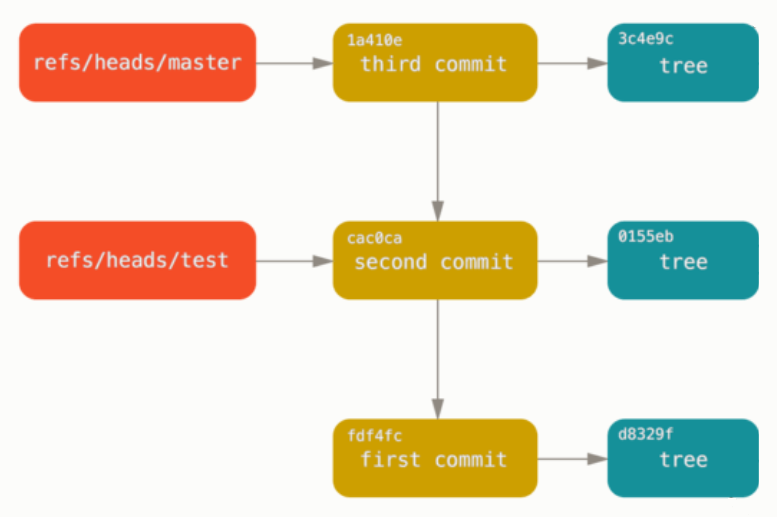
我们接下来查.git/refs/heads/目录中，文件的内容：

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ cat .git/refs/heads/master d6f67eeefe7e0d49cdaf6cadb50d219e9b0d5674 |

可以看到，分支文件的内容是一个提交的索引值。

即：heads/目录中的分支文件，里面存储的是该分支的指针，指向的是具体哪一个提交。

所以分支和提交的关系图如下：



**总结：这基本就是 Git 分支的本质：一个指向某一系列提交之首的指针或引用。**

**（2）tags/目录**

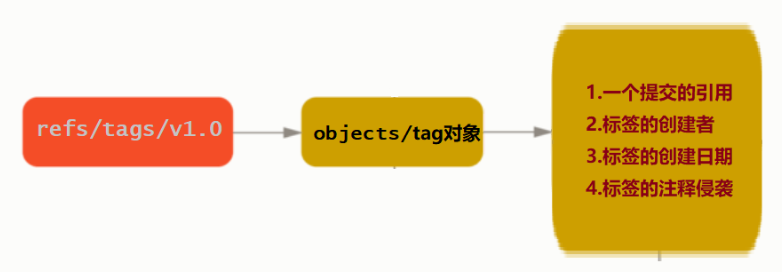
tags/目录，里面存放的是当前版本库中的标签，和heads/目录中的分支同理（如上）。

因为标签可以理解成一个永不移动的分支。

但是和分支不同的是，tags/目录中的文件代表一个标签。文件中存储的是，该标签存储在Git仓库（即objects目录）中的标签对象的索引值。

而标签对象的内容包含了一个提交对象的索引，该索引执行一个提交。

如下图所示：



**8、HEAD文件**

现在的问题是，当你执行 git branch <branch> （创建分支）时，Git 如何知道最新提交的 SHA-1 值呢？ 答案是 HEAD 文件。

HEAD 文件通常是一个符号引用（symbolic reference），指向目前所在的分支的最新一次提交。 所谓符号引用，表示它是一个指向其他引用的指针。

查看 HEAD 文件的内容，通常我们看到类似这样的内容：

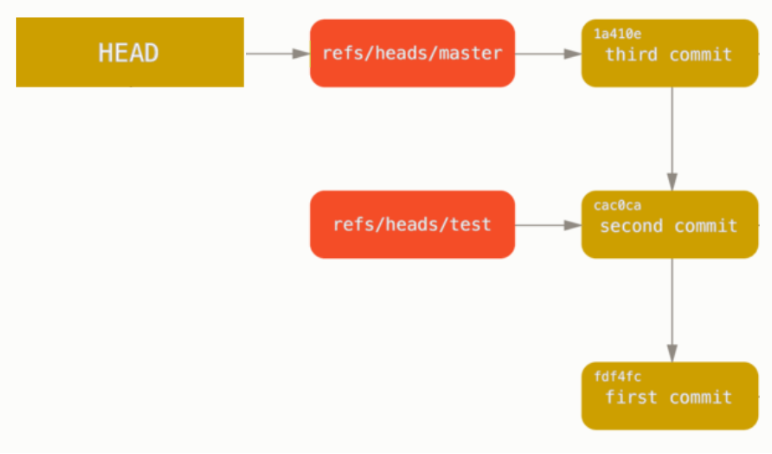
|  |
| --- |
| Bash $ cat .git/HEAD ref: refs/heads/master |

如果执行 git checkout test，Git 会像更新 HEAD 文件,，如下：

|  |
| --- |
| Bash $ cat .git/HEAD ref: refs/heads/test |

|  |
| --- |
| 提示：如果编辑HEAD文件，其实和执行切换分支命令git checkout的效果是一样的。 |

HEAD和分支的关系如下图：



当我们执行 git commit 命令时，该命令会创建一个提交对象，并用 HEAD 文件中的引用（分支），来指向该提交对象。（我的理解就是，当进行了一次提交时，其实是HEAD指针移动指向新的提交，而HEAD指针引用了分支，所以分支也指向了该提交。）

**9、index文件**

index文件：该文件就是我们平时说的 **暂存区** ，保存了下次将提交的文件列表信息，我们执行git add命令后，这个文件就会更新刚刚添加的文件信息。

提示：刚刚初始化的Git本地版本库中是没有index文件的，只有执行一次暂存操作后，才在.git目录自动生成index文件。

**10、logs目录**

logs文件夹中保存所有提交的引用记录。

打开logs文件夹可以看到其中有两个文件，refs文件夹和HEAD文件。

|  |
| --- |
| Bash L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (master) $ ll .git/logs/ total 4 -rw-r--r-- 1 L 197121 908 4月 22 18:35 HEAD drwxr-xr-x 1 L 197121 0 4月 22 14:35 refs/ |

**（1）HEAD文件**

HEAD文件保存的是，所有的引起HEAD指针移动的操作记录，使用git reflog命令，查询的结果就是来自这个文件。

**（2）refs目录**

refs文件夹中有两个文件夹：heads目录和remotes目录。

* heads目录中都是以分支命名的文件，即：每个文件名对应着本地版本库的一个分支。
* 每个文件中，记录的都是该分支历史操作记录。
* remotes目录和heads目录的作用同理，只不过remotes目录中存储的是远程分支的历史操作记录。

**（3）说明**

heads目录中所有分支历史操作记录的总和，是HEAD文件文件的内容。

例如版本库中有两个分支，分别查看他们历史操作记录：

|  |
| --- |
| Bash # 1.查看master分支的历史操作记录 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (dev) $ cat .git/logs/refs/heads/master 0000000000000000000000000000000000000000 f58d840994fe43cf8cdf8fe036dbadb393b7df62 sun\_wk <sun\_wk@126.com> 1619073338 +0800 commit (initial): 第1次提交 f58d840994fe43cf8cdf8fe036dbadb393b7df62 d6f67eeefe7e0d49cdaf6cadb50d219e9b0d5674 sun\_wk <sun\_wk@126.com> 1619073495 +0800 commit: 第2次提交  # 2.查看dev分支的历史操作记录 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (dev) $ cat .git/logs/refs/heads/dev 0000000000000000000000000000000000000000 d6f67eeefe7e0d49cdaf6cadb50d219e9b0d5674 sun\_wk <sun\_wk@126.com> 1619073563 +0800 branch: Created from HEAD d6f67eeefe7e0d49cdaf6cadb50d219e9b0d5674 d00abd3a31bfa95cc88e1fb2df21421b2d3743a9 sun\_wk <sun\_wk@126.com> 1619073598 +0800 commit: 第3次提交 d00abd3a31bfa95cc88e1fb2df21421b2d3743a9 9ec0e8f8a861d8da31427f0508aa4c00d3242988 sun\_wk <sun\_wk@126.com> 1619073751 +0800 commit: 第4次提交  # 3.查看HEAD文件的历史操作记录。 L@DESKTOP-T2AI2SU MINGW64 /j/git-repository/git\_learning (dev) $ cat .git/logs/HEAD 0000000000000000000000000000000000000000 f58d840994fe43cf8cdf8fe036dbadb393b7df62 sun\_wk <sun\_wk@126.com> 1619073338 +0800 commit (initial): 第1次提交 f58d840994fe43cf8cdf8fe036dbadb393b7df62 d6f67eeefe7e0d49cdaf6cadb50d219e9b0d5674 sun\_wk <sun\_wk@126.com> 1619073495 +0800 commit: 第2次提交 d6f67eeefe7e0d49cdaf6cadb50d219e9b0d5674 d6f67eeefe7e0d49cdaf6cadb50d219e9b0d5674 sun\_wk <sun\_wk@126.com> 1619073563 +0800 checkout: moving from master to dev d6f67eeefe7e0d49cdaf6cadb50d219e9b0d5674 d00abd3a31bfa95cc88e1fb2df21421b2d3743a9 sun\_wk <sun\_wk@126.com> 1619073598 +0800 commit: 第3次提交 d00abd3a31bfa95cc88e1fb2df21421b2d3743a9 9ec0e8f8a861d8da31427f0508aa4c00d3242988 sun\_wk <sun\_wk@126.com> 1619073751 +0800 commit: 第4次提交 |

如上我们可以看到，master分支和dev分支的历史操作记录总和，就是HEAD文件中的内容。

作者：繁华似锦Fighting 链接：https://www.jianshu.com/p/4cc3959dd6c2 来源：简书 著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。