Git对象模型

# Git对象

在Git系统中有四种类型的对象，所有的Git操作都是基于这四种类型的对象。上面我们介绍了Git对象，在Git系统中，每个Git对象都有一个特殊的ID来代表这个对象，这个特殊的ID就是我们所说的SHA1哈希值。

SHA1哈希值是通过SHA1算法（SHA算法家族的一种）计算出来的哈希值，对于内容不同的对象，会有不同的SHA1哈希值。如果你读过前面一篇文章，就肯定还记得我们是怎么根据commit id撤销更新的，这里的commit id就是一个SHA1哈希值。Git系统中有如下四种类型的对象:

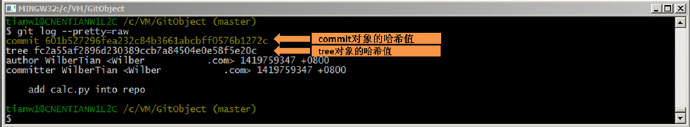
* "blob"：这种对象用来保存文件的内容。（对应文件）
* "tree"：可以理解成一个对象关系树，它管理一些"tree"和 "blob"对象。(对应目录)
* "commit"：只指向一个"tree"，它用来标记项目某一个特定时间点的状态。它包括一些关于时间点的元数据，如时间戳、最近一次提交的作者、指向上次提交（初始commit没有这一项）。
* "tag"：给某个提交(commit) 增添一个标记。

# Git对象模型实例

第一步：新建一个仓库，添加一个"calc.py"的文件

通过" git log --pretty=raw"可以得到每个commit的SHA1哈希值，也可以得到这个commit对应的tree的哈希值。所以，一个commit对象一般包含以下信息：

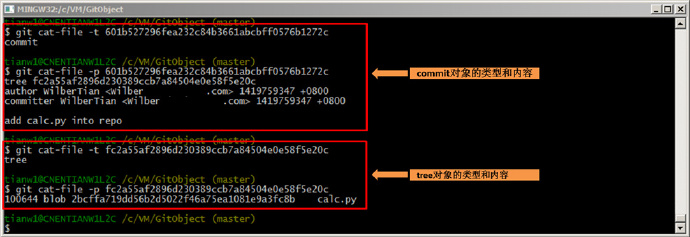
* 代表commit的哈希值
* 指向tree 对象的哈希值
* 作者
* 提交者
* 注释



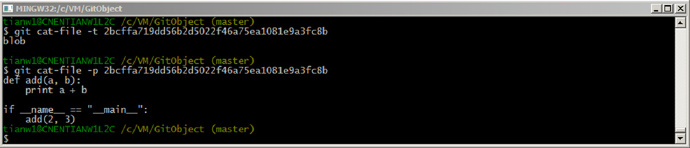
在Git对象模型的研究中，有一个很有用的命令"git cat-file"，可以通过这个命令查询特定对象的信息:

* git cat-file -t key：通过一个对象的哈希值可以通过这条命令查看对象的类型（blob、tree、commit或tag）
* git cat-file -p key：通过对象的哈希值可以查看这个对象的内容

下面我们看一下在这次提交中commit、tree对象的类型和内容。这里可以看到，在这一次的commit中tree对象的内容指向一个blob对象。



同样，我们可以查看blob对象的类型和内容，这里可以看到blob的内容就是我们要提交的文件的内容。

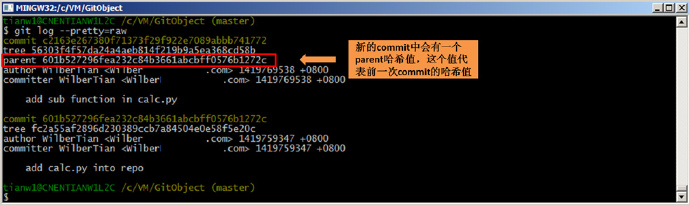


所以，根据上面的一些数据，我们可以看到在这一次的提交中产生了三个对象，同时看到了commit、tree、blob三个对象的关系如下：

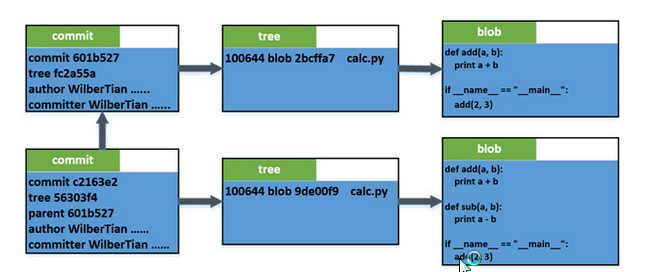


第二步：更新"calc.py"文件，添加sub函数

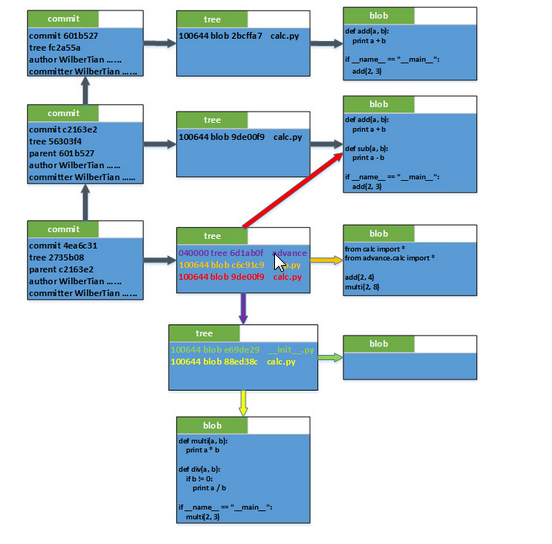
继续使用"git log --pretty=raw"查看commit log。



同样通过"git cat-file"我们可以看到每一个对象的类型和内容，这里就不一步一步上图了，直接给出所有的对象关系。这里需要注意的一点，Perforce、SVN和CVS属于"增量文件系统" （Delta Storage systems），它们每次只存储提交(commit)之间的差异。而对于Git，它会把你的每次提交的文件的全部内容（snapshot）都会记录下来。



第三步：增加一个"app.py"；增加"advance"文件夹，包括"\_\_init\_\_.py"和"calc.py"。同样的方式，我们可以得到一张更复杂的对象关系图：



总结：一个commit对象首先指向父commit；每个commit包含1个tree，每个tree指向多个tree(目录)，blob(文件) 以及元信息(提交人，邮件，提交时间等信息)；