**Git分支 - Git分支相关理论**

**1、Git对象之间的关系**

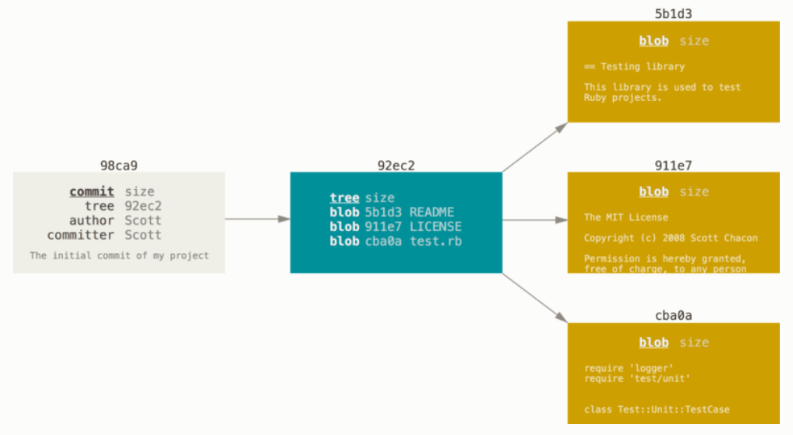
我们之前学了Git的三个对象：提交对象、树对象、数据对象。

我们假设现在有一个工作目录，里面进行了三次提交，包括一次新增文件和两次对文件的修改。

* 每次一把工作区中的文件添加到暂存区时，暂存操作会为每一个文件计算校验和，然后会把当前版本的文件快照（即文件的内容）保存到 Git 仓库中 （Git 使用 *blob* 对象来保存它们），最后将校验和加入到暂存区域等待提交。
* 每一次使用 git commit 命令进行提交操作时，Git会先计算每一个子目录的校验和， 然后在 Git 仓库中这些校验和保存为**树对象**。
* 随后，Git 便会创建一个提交对象， 它除了包含上面提到的那些信息外，还包含指向这个树对象的指针。 如此一来，Git 就可以在需要的时候重现此次保存的快照。

此时的Git仓库中就会有五个Git对象：三个 *blob* 对象（保存着文件快照）、一个 **树** 对象 （记录着目录结构和 blob 对象索引）以及一个 **提交** 对象（包含着指向前述树对象的指针和所有提交信息）。

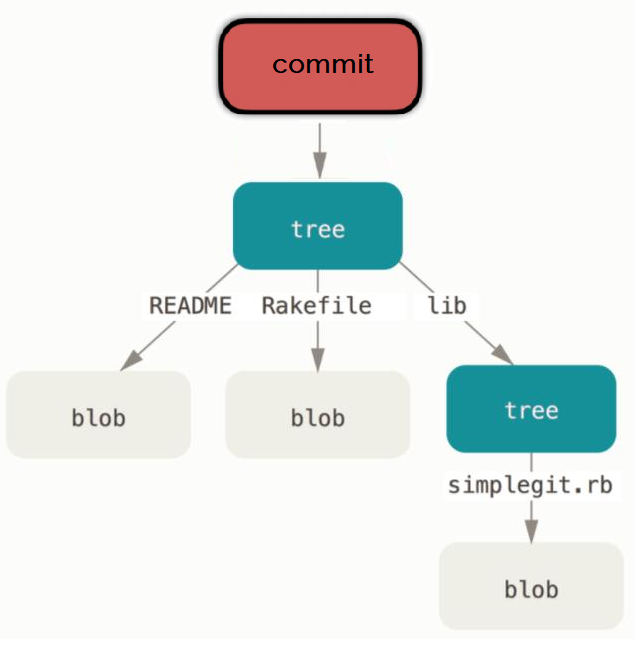
这个对象之间的关系，如下图：



总结提交对象、树对象、数据对象之间的关系：

1. 一个提交操作生成一个提交对象。
2. 一个提交对象包含一个（暂存区）生成Tree对象。（对Tree对象的封装，单方向一对一）
3. 一个（暂存区）生成Tree对象，里面包含子Tree对象和Blob对象。子Tree对象还可以继续包含子Tree对象和Blob对象。
4. 这样子Tree对象对应文件系统中的目录+文件名，Blob对象对应着文件中的内容，这就是Git中数据存储的形式。
5. 一个Blob对象对应着一个文件某一时刻的版本。

三种之间的关系如下图：



那么问题来了：**每一个Commit对象，是怎样的组合到一起的呢？**

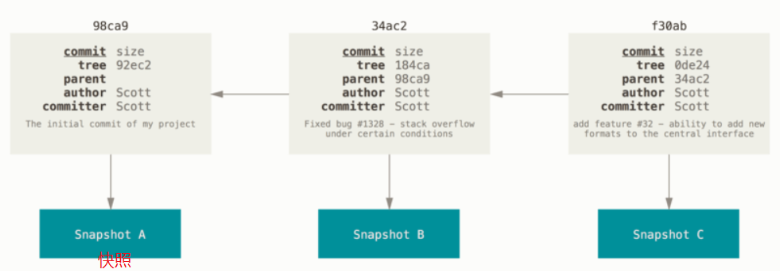
**2、提交对象与分支的关系**

**（1）提交对象与分支的关系**

Git版本管理系统是以时间线来对版本进行管理的，这条时间线上会有很多的时间节点，这些时间节点就是一个个的Commit对象。

**即：Git的每次提交，都会自动把它们串成一条时间线，这条时间线就是一个分支。**

如下图所示：每一次提交产生的提交对象，会包含一个指向上次提交对象（父对象）的指针，这样就形成了一条链状结构，就相当于一条线。



**（2）分支说明**

**Git的分支，其实本质上仅仅是指向提交对象的可变指针。**

Git仓库初始化之后，会默认创建一个master分支，即主分支。

如果没有新建分支，那么就只有一条时间线，即只有一个分支，master分支（主分支）。

每次提交操作之后，会生成新的提交对象（如上图）， master 分支会在每次提交时自动向前移动。（也就是自动指向最新的提交对象）

**（3）HEAD与分支的关系**

我们在学习Git的时候，常常会看到HEAD这个名称，它指的是什么呢？

Git中维护一个名为HEAD的引用变量，我们将此变量称为指针，它的目的是引用或指向本地版本库中的特定提交。

当我们进行新的提交时，指针将改变或移动，以指向新的提交。

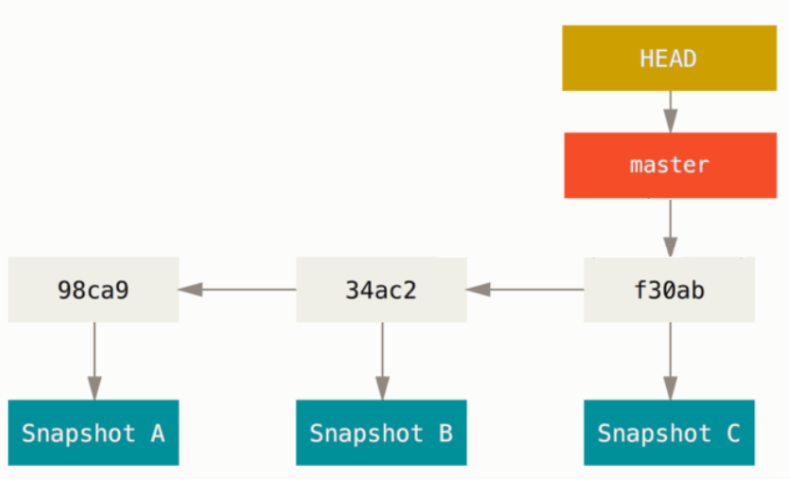
HEAD始终指向Git本地版本库中当前正在工作的分支的尖端（即最新一次提交）。

**概括来说：HEAD是对当前分支中，最后一次提交的引用。**（可以将HEAD想象为是，当前分支最后一次提交的别名。）

再继续：

HEAD严格来说不是指向提交，而是指向master（分支），master（分支）才是指向具体的提交，所以，HEAD指向的就相当于是当前分支的最新一次提交。

如下图：



Git用master指向最新的提交，再用HEAD指向master，就能确定当前分支，以及当前分支的提交点。

（当然HEAD还有一种分离的状态，我们以后单说，关于HEAD就先理解到这里就很详细了）

作者：繁华似锦Fighting 链接：https://www.jianshu.com/p/b4229f49db1c 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。