1. 基本入门
    1.1 git目录构成
    1.2 hash code如何生成
        1.2.1 什么是头部信息?
        1.2.2 什么是当前文件内容?
2. 小试牛刀，看看git保存我们的提交的
    2.1 git是如何存储文件的?
3. git存储类型
    3.1 tree
    3.2 blob
    3.3 commit

> 简介:本教程旨在让你可以初步了解git内部是如何运作的，仅仅灌鸭模式让您能快速的了解git内部。如要发掘更多的细节、更多好玩的东西，建议多看git教程。推荐pro git这本书，此书可以在git官网下载。

1. 基本入门

1.1 git目录构成

```
➜  .git git:(master) ls
HEAD        branches    config      description hooks       info        objects     refs
```

HEAD:当前被检出的分支指向一个refs hash code.

branches: 类似refs功能，新版已经不再使用了，废弃此目录。

config:包含当前工作目录的配置信息，比如我们的git账号、密码、用户信息之类的。

description: 供gitweb使用

hooks:放置 服务端或者客户端的钩子脚本

info:包含非记录在.gitignore内的排除文件

objects：git的数据仓库

refs：记录每个提交所指向对应版本文件的hash

index:暂存区内容

1.2 hash code如何生成

git的hash code,是通过将当前文件内容与头部信息在一起做`SHA-1`校验运算得出来的校验和。

1.2.1 什么是头部信息?

git 首先取得文件类型，然后添加一个空格，随后是数据内容长度，最后一个是空字节(null byte)。

这就是一个git头部信息。

```
>> header = "blob #{content.length}\0"
=> "blob 16\u0000"
```

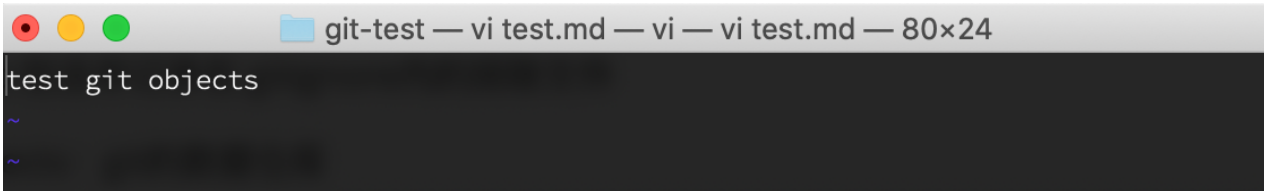
1.2.2 什么是当前文件内容?

就是当前修改的文件内容，不同的内容有不同的信息。git有三种存储类型`blob`,`tree`,`commit`。比如`blob`就是整体的文件内容。`tree`则是包含提交的文件列表。`commit`就是提交信息。

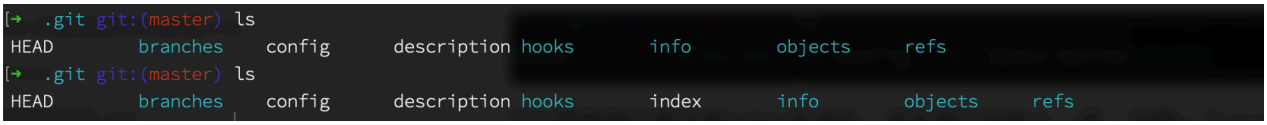
2. 小试牛刀，看看git保存我们的提交的

我们在`.git`目录中查看所有文件

可以看到我们是没有index目录的。现在我们回到上一层，创建一个`test.md`文件，然后写点内容进去。

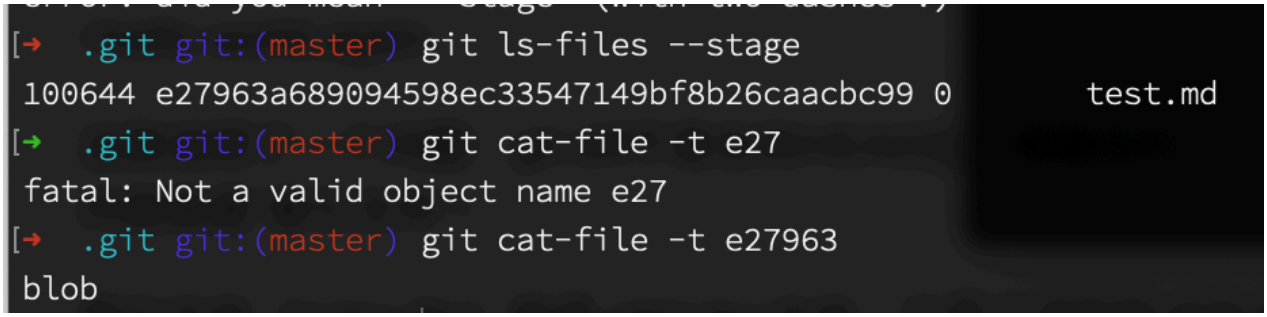


接着，我们运行`git add .`将文件添加至暂存区。接着，我们来查看`.git`目录的文件与文件夹。



可以看到，git创建了index文件。

运行以下命令，来看看index中都包含了什么内容:



一共运行了两个命令，第一个`git ls-files --stage`是查看暂存区的文件内容，第二个`git cat-file -t e27963`是查看，符合这个hash值的文件类型。知道类型之后，可以使用这个命令来查看内容。

```
[➜  .git git:(master) git cat-file blob e27963
test git objects
```

可以看到，这个hash值对应的文件内容就是我输入的内容。

现在我们`commit`保存修改。

这个时候，`objects`文件夹中就出现了hash开头的文件夹，与对应的内容。

```
[➜  .git git:(master) find objects
objects
objects/be
objects/be/49bf893045c6e6d2dd6f6113d4abf222ec9aff
objects/e2
objects/e2/7963a689094598ec33547149bf8b26caacbc99
objects/pack
objects/info
objects/76
objects/76/87f2b4a862376467b3ff82ec05b70cdf0e4236
```

通过`git cat-file`可以查看对应hash值中的内容。

```
[➜  .git git:(master) git cat-file -p be49bf
100644 blob e27963a689094598ec33547149bf8b26caacbc99    test.md
```

要注意一个点，就是 objects下的文件夹，后续都是hash code.比如`be/49bf`,那么他的hash code就是`be49bf`(**“SHA-1 值的前两个字符作为子目录名称，后 38 个字符则作为子目录内文件的名称”**)。

```
[➜  .git git:(master) git cat-file -p be49bf
100644 blob e27963a689094598ec33547149bf8b26caacbc99    test.md
[➜  .git git:(master) git cat-file -p e27963a
test git objects
[➜  .git git:(master) git cat-file -p 7687
tree be49bf893045c6e6d2dd6f6113d4abf222ec9aff
author yodfz <yodfz@qq.com> 1583051290 +0800
committer yodfz <yodfz@qq.com> 1583051290 +0800

test
➜  .git git:(master)
```

可以看到，这些hash结构的内容，对应的，分别是当前提交内容包含哪些文件、文件内容、commit信息。

2.1 git是如何存储文件的?

我对`test.md`文件末尾追加了一些内容。现在我们来看看，git存储中,test.md文件是多少大小。

```
➜  .git git:(master) git cat-file -t 487c6aae23ae41757584ab2ba2ac46f17e2b5ad2
commit
➜  .git git:(master) git cat-file commit 487c6aae23ae41757584ab2ba2ac46f17e2b5ad2
tree 96329098e899e8c9499d247db3e4eb5e28908f82
parent 7687f2b4a862376467b3ff82ec05b70cdf0e4236
author yodfz <yodfz@qq.com> 1583053733 +0800
committer yodfz <yodfz@qq.com> 1583053733 +0800

add content
➜  .git git:(master) git cat-file tree 96329098e899e8c9499d247db3e4eb5e28908f82
100644 test.md????
                  ?V??lA%K8?V??%
➜  .git git:(master) git cat-file -p 96329098e899e8c9499d247db3e4eb5e28908f82
100644 blob bd8698920bb156bdbe6c41254b100138ba5688b5    test.md
➜  .git git:(master) git cat-file -s bd8698920bb156bdbe6c41254b100138ba5688b5
35
```

通过上图发现，最新提交的文件在git存储里面是`35`字节。那么，之前的版本是多少呢？我们来看看。

```
➜  .git git:(master) git cat-file -s e27963
17
```

是`17`个字节。由此，我们可以得出一个结论，所有的文件，每次修改提交在git中都是一个全新的文件。

那么由此带来的一个问题就是，按照这样来说的话，我们的项目目录不是会很大?

git开发者也想到了这个问题，所以，我们在git push的时候是否会发现，有类似下面的信息：

```
Counting objects: 18, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (18/18), done.
Total 18 (delta 3), reused 0 (delta 0)
```

这个是git帮我们做了压缩。对应的git命令是`git gc`。我们来运行一下这个命令看看。

```
➜  .git git:(master) find objects -type f
objects/be/49bf893045c6e6d2dd6f6113d4abf222ec9aff
objects/bd/8698920bb156bdbe6c41254b100138ba5688b5
objects/e2/7963a689094598ec33547149bf8b26caacbc99
objects/96/329098e899e8c9499d247db3e4eb5e28908f82
objects/48/7c6aae23ae41757584ab2ba2ac46f17e2b5ad2
objects/76/87f2b4a862376467b3ff82ec05b70cdf0e4236
➜  .git git:(master) git gc
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), done.
Total 6 (delta 0), reused 0 (delta 0)
➜  .git git:(master) find objects -type f
objects/pack/pack-e19d1da6647ab26c0048dad7e149981328e52fec.idx
objects/pack/pack-e19d1da6647ab26c0048dad7e149981328e52fec.pack
objects/info/packs
```

可以看到，运行命令之后，`objects`下的内容都不见了。随之而来则是生成了`idx`,`pack`文件。

新创建的**包文件**和**索引文件**。 包文件包含了刚才从文件系统中移除的所有对象的内容。 索引文件包含了包文件的偏移信息，我们通过索引文件就可以快速定位任意一个指定对象。

使用`git verify-pack -v`来查看当前索引文件中包含了哪些内容

```
➜  .git git:(master) git verify-pack -v objects/pack/pack-755b0d872704dc612fe3a2b331e173ad010a7191.idx
20ab77368416865bb36407c693b996632efda17a commit 196 141 12
e1b6f58a41b0468aaabcfa0a6c617e4353b201e5 commit 195 140 153
487c6aae23ae41757584ab2ba2ac46f17e2b5ad2 commit 200 143 293
7687f2b4a862376467b3ff82ec05b70cdf0e4236 commit 145 110 436
e0ffd0c13ce5a7c2cf85473faebe2d4d793b66ff blob   146 101 546
3b28ef2514e39a0dea074264852301499e709861 tree   35 46 647
ffe4804c540df95c52016b1dec1ce0b3fece69cf tree   35 46 693
49bfbc8282b38291e9178544779d518b7bed3653 blob   6 17 739 1 e0ffd0c13ce5a7c2cf85473faebe2d4d793b66ff
96329098e899e8c9499d247db3e4eb5e28908f82 tree   35 46 756
bd8698920bb156bdbe6c41254b100138ba5688b5 blob   35 43 802
be49bf893045c6e6d2dd6f6113d4abf222ec9aff tree   35 46 845
e27963a689094598ec33547149bf8b26caacbc99 blob   17 27 891
non delta: 11 objects
chain length = 1: 1 object
objects/pack/pack-755b0d872704dc612fe3a2b331e173ad010a7191.pack: ok
➜  .git git:(master)
```

由于前几次的文件过小，git还是采用了完全快照模式。所以进行了多次修改，我尝试将这个文件变大之后，现在我们可以看到`e0ffd0c1`这个提交与`49bfbc`这个提交。git采用了差异保存模式保存文件。

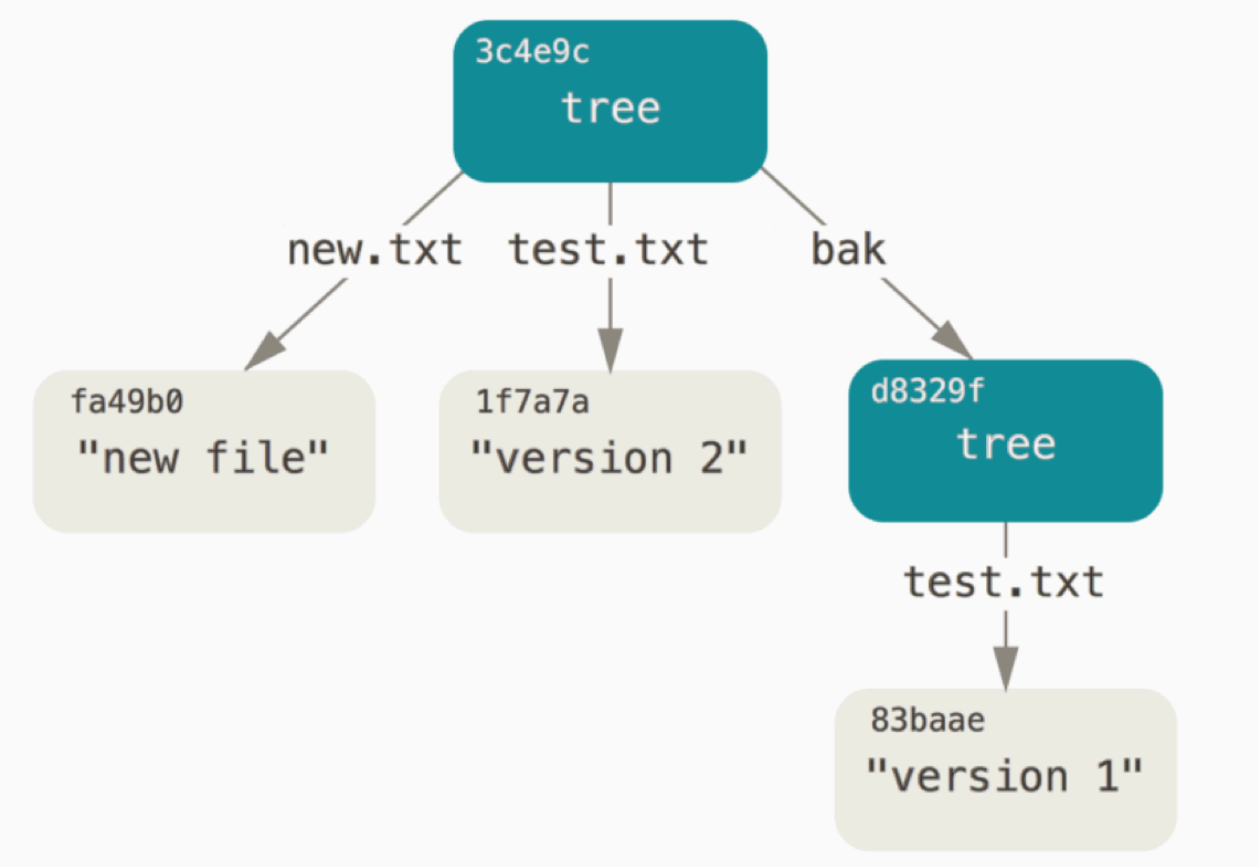
git之所以让原始版本使用差异模式保存，是为了能快速访问文件的最新版本。

3. git存储类型

3.1 tree

树对象，是保存文件内容变更的简化模型。通过下图可以发现，一个文件夹对应一棵树，如果包含了另外一个文件将，那么，将通过tree hash code 来访问。简单的说，就是tree仅有一层真实文件目录，文件夹都是通过引用来访问。

tree由不同版本的文件组成。他们看起来像这样的结构:



3.2 blob

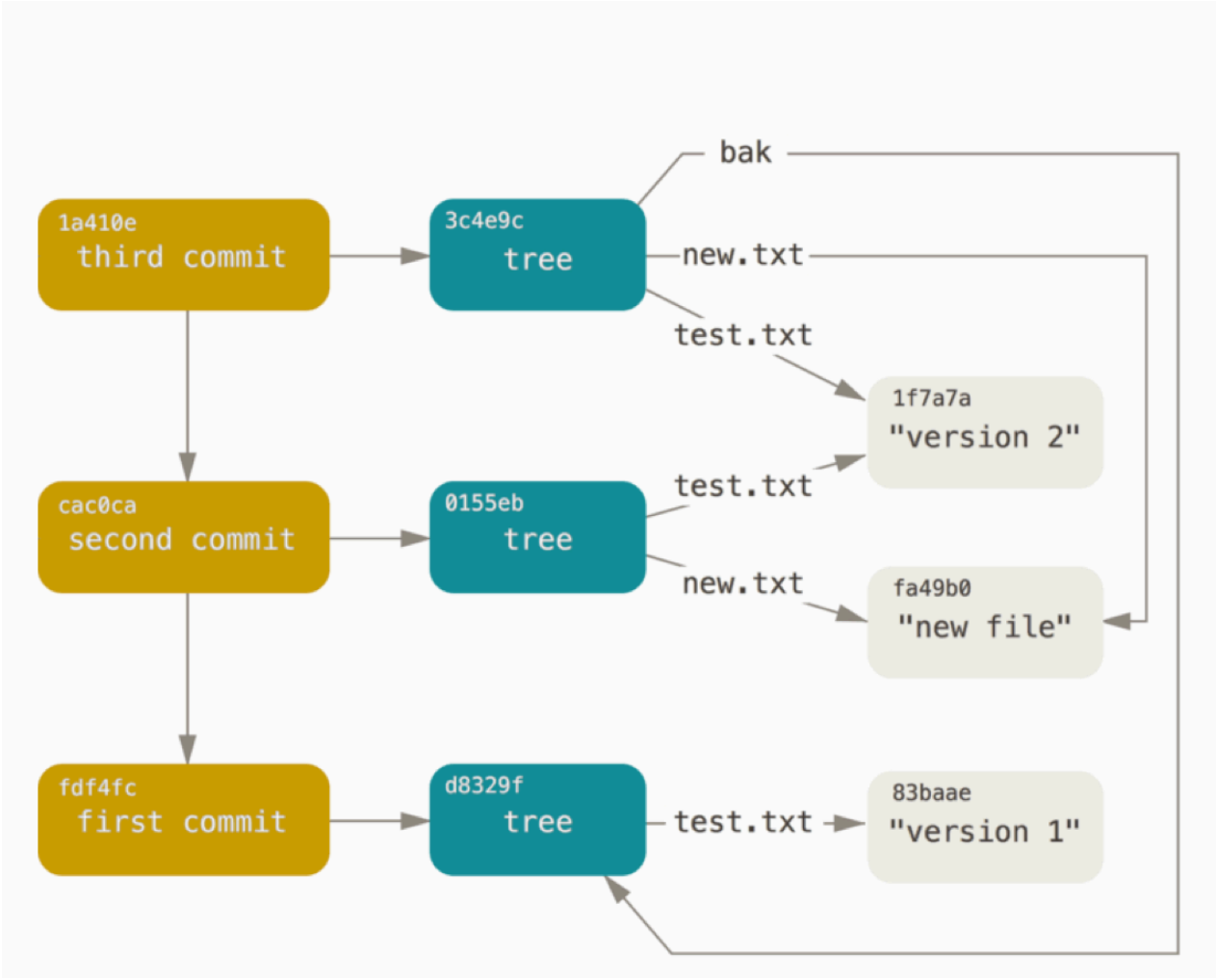
blob对应的都是实体文件信息。hash指定blob的文件名，存储在objects中，属于最原子结构。

3.3 commit

每一个commit信息，包含了指向的某个tree节点，父`commit`节点，提交人，提交信息等。

```
tree 96329098e899e8c9499d247db3e4eb5e28908f82
parent 7687f2b4a862376467b3ff82ec05b70cdf0e4236
author yodfz <yodfz@qq.com> 1583053733 +0800
committer yodfz <yodfz@qq.com> 1583053733 +0800

add content
```


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