工作区、版本库（暂存区、分支）

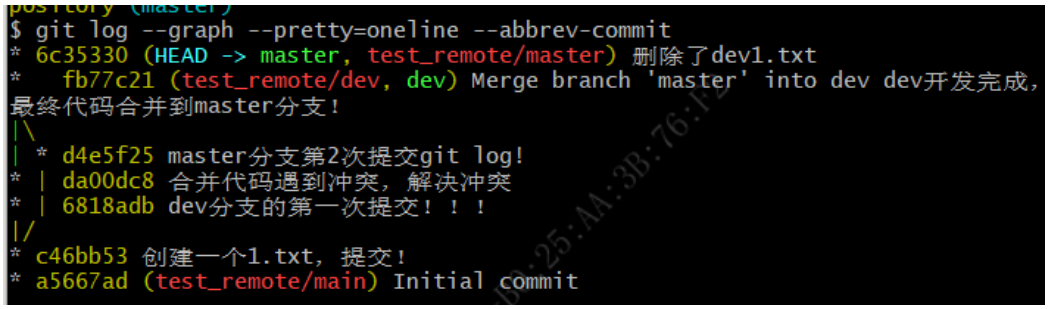
远程仓库——关联远程仓库、push、删除关联、clone、pull

分支——创建、切换、删除、合并、查询当前分支

标签——创建、删除、查看

看分支的详细历史时间节点：

```
$ git log --graph --pretty=oneline --abbrev-commit
```



1. 安装git，检查版本git --version
2. 初始化设置全局用户名和邮箱

```
$ git config --global user.name "Your Name"
$ git config --global user.email "email@example.com"
```

3. 创建版本库

```
$ mkdir learngit
$ cd learngit
$ pwd
/Users/michael/learngit
-- 初始化（会产生一个,git文件夹）
$ git init
Initialized empty Git repository in
/Users/michael/learngit/.git/
```

4. 添加想要添加的文件，最后提交到本地仓库

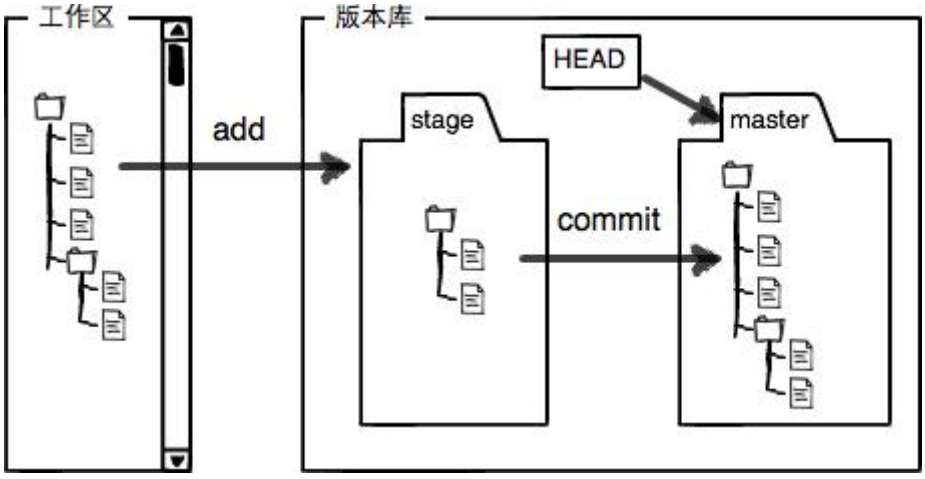
```
$ git add file1.txt
$ git add file2.txt file3.txt

$ git commit -m "add 3 files."
```

工作区、版本库、暂存区stage、分支：

add是将文件修改添加到工作区中的暂存区

commit是将文件从暂存区添加到分支



5. 要随时掌握工作区的状态，使用`git status`命令。如果`git status`告诉你有文件被修改过，用`git diff`可以查看修改内容。（**这两个命令要经常用**）
6. 版本回退

```
-- 上一个版本就是HEAD^，上上一个版本就是HEAD^^，当然往上100个版本写100个^比较容易数不过来，所以写成HEAD~100
$ git reset --hard HEAD^
HEAD is now at e475afc add distributed

-- 回退指定版本
-- 最新的那个版本append GPL已经看不到了！好比你从21世纪坐时光穿梭机来到了19世纪，想再回去已经回不去了，肿么办？办法其实还是有的，只要上面的命令行窗口还没有被关掉，你就可以顺着往上找啊找啊，找到那个append GPL的commit id是1094adb...，于是就可以指定回到未来的某个版本：git reset --hard commitid
$ git reset --hard 1094a
HEAD is now at 83b0afe append GPL

-- 查看commit提交历史记录以及commit id
git log
```

7. 撤销修改（就是工作区的代码回退到版本库一致的代码）

```
-- git checkout -- file意思就是，把readme.txt文件在工作区的修改全部撤销
$ git checkout -- readme.txt
```

8. 从版本库中删除文件

```
-- 从版本库的暂存区和工作区删除文件
$ git rm test.txt
rm 'test.txt

-- 版本库暂存区的修改提交到分支，使分支也删除文件
$ git commit -m "remove test.txt"
[master d46f35e] remove test.txt
 1 file changed, 1 deletion(-)
 delete mode 100644 test.txt
```

9. 操作远程仓库

```
-- 在本机命令窗口创建ssh通讯协议的公钥和私钥，一路回车即可。生成公钥和私钥文件在当前用户目录下
$ ssh-keygen -t rsa -C "youremail@example.com"
-- 然后去远程仓库GitHub或者Gitee的ssh keys添加该主机的公钥
-- 这样远程仓库就“认识”这台机器，可以方便的上次代码上去了！
-- 可以不用ssh协议上传，也可以使用https协议，但速度远没有ssh快
```

本地项目推送到远程仓库

```
-- 关联远程仓库   git remote add 远程仓库名称 ssh/https地址
$ git remote add origin
git@github.com:michaelliao/learngit.git

-- git push -u 远程仓库名称 分支
$ git push -u origin master
-- 我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令
$ git push origin master
```

删除远程仓库的关联

```
-- 查看远程库信息
$ git remote -v
origin  git@github.com:michaelliao/learn-git.git (fetch)
origin  git@github.com:michaelliao/learn-git.git (push)

-- 删除远程库 git remote rm 远程仓库名称（关联的时候自己起的名字）
$ git remote rm origin
```

从远程仓库克隆项目（无需关联）

```
-- git clone ssh/https地址
$ git clone git@github.com:michaelliao/gitskills.git
Cloning into 'gitskills'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 3
Receiving objects: 100% (3/3), done.
```

拉取当前分支

```
-- 从远程抓取分支，使用git pull，如果有冲突，要先处理冲突
-- git pull 远程仓库名称（关联的时候自己起的名字） 远程仓库上的分支
$ git pull test_remote main
```

10. 创建分支

```
--git checkout -b 分支名称    其中-b参数表示创建并切换
$ git checkout -b dev
Switched to a new branch 'dev'
-- 相当于
$ git branch dev
$ git checkout dev
Switched to branch 'dev'
-- 也可以使用 创建并切换分支（下面的好理解）
$ git switch -c dev
-- 切换分支
$ git switch master
```

删除分支

```
$ git branch -d dev
Deleted branch dev (was b17d20e).
```

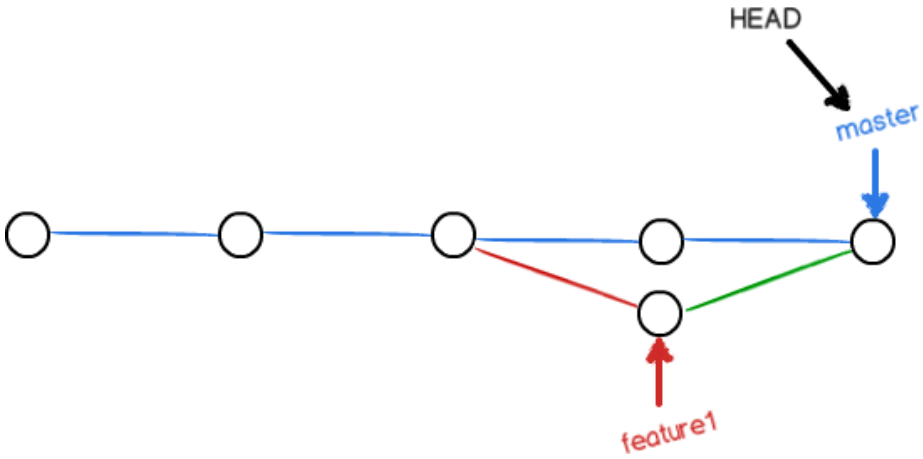
合并分支

```
$ git merge dev
Updating d46f35e..b17d20e
Fast-forward
 readme.txt | 1 +
 1 file changed, 1 insertion(+)

-- 合并分支时，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并
```

查看分支

```
$ git branch
* master
```



11. 标签管理（标签是本地的）

```
-- 打标签
$ git tag v1.0

-- 查询当前分支全部标签
$ git tag
v1.0

-- 查看标签详细信息
$ git show v0.9
commit f52c63349bc3c1593499807e5c8e972b82c8f286 (tag:
v0.9)
Author: Michael Liao <askxuefeng@gmail.com>
Date:   Fri May 18 21:56:54 2018 +0800
    add merge
diff --git a/readme.txt b/readme.txt

-- 给历史版本打标签（f52c633为commit id）
$ git tag v0.9 f52c633

-- 删除标签
$ git tag -d v0.1
Deleted tag 'v0.1' (was f15b0dd)
```