视频 |【尚硅谷】Git与GitHub基础全套完整版教程（快速上手，一套搞定）

💡B站视频

- 视频

  https://player.bilibili.com/player.html?bvid=BV1pW411A7a5&p=2&page=2

💡笔记

1 总体学习结构

github 版本控制

Git 简介

Git 命令执行

Gitlab服务器环境搭建（暂不学）

Git 图形化界面操作（暂不学）

2 Github 的版本控制

2.1. 为什么需要版本控制

1. 版本控制方便查询历史操作
2. 方便多人协作

3. 历史备份
4. 具有权限控制功能

2.2. 版本控制介绍

2.2.1 版本控制方式

SVN选择增量式版本管理方式，Git选择文件系统快照形式

2.2.2 版本控制工具

1. 集中式：SVN，CVS（存在一台服务器中心控制，缺点：中心控制的服务器很重要）
2. 分布式：Github（优势：防止中心服务器故障）

3 Git 简介

3.1 Git 优势

1. 操作可在本地完成，不需联网
2. 完整性保证（hash控制文件完整性，防止文件传输过程中丢失）
3. 尽可能添加数据而不是删除或修改数据
4. 分支操作快捷流畅
5. 与Linux命令全面兼容

3.2 Git 程序安装过程

Adjusting your Path environment根据情况选择，希望使用command line操作Github的最好选择“Use Git from the Windows Command Prompt”，此为目前系统推荐

3.3 Git 结构介绍

工作区（写代码）、暂存区（临时储存）、本地库（历史版本）

3.4 Git和代码托管中心（任务：维护远程库）

3.4.1 局域网环境下

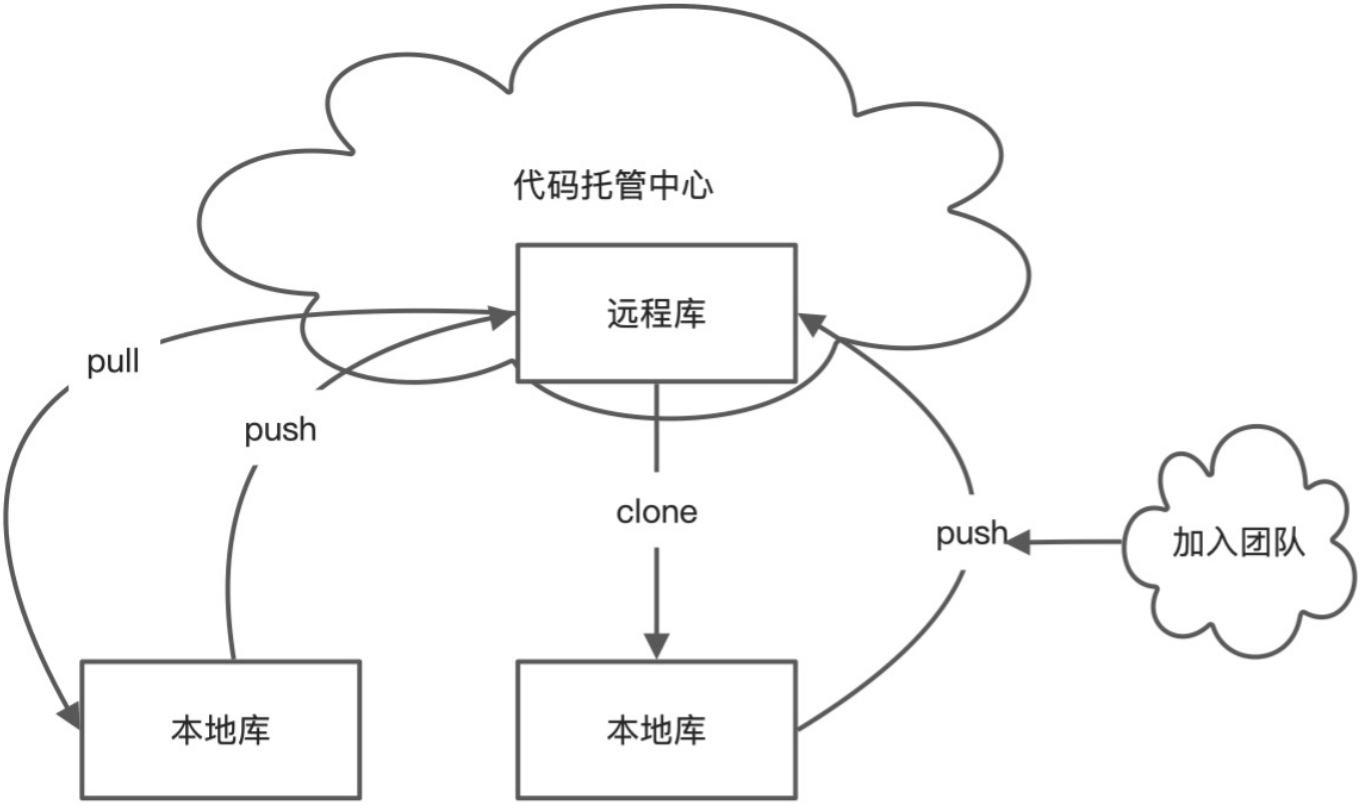
GitLab服务器

3.4.2 外网环境下

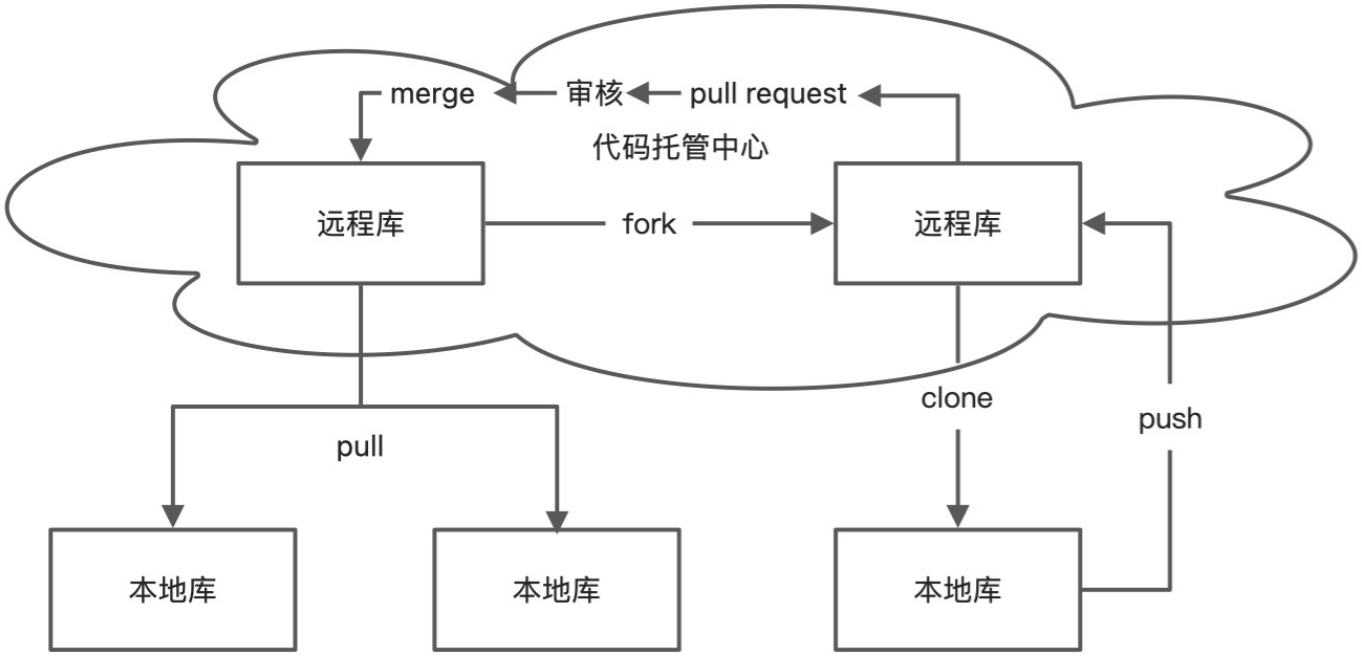
GitHub，码云（中国国内访问较快）

3.4.3 本地库和远程库：

1. 团队内部协作



2. 跨团队协作



4 Git 命令行操作

4.1 本地库操作

4.1.1 git bash

Windows可以进入git bash界面写linux code进行操作

4.1.2 git init 本地库初始化

1. 进入 repository directory
2. git init（初始化）

   结果显示 initialized empty Git repository in D:/your/path
3. command line:
   a. ls –lA（查看该目录下所有文件，包含隐藏文件）
   b. ls –lG（查看该目录下所有文件，不包含隐藏文件）
4. 进行git init之后current directory下会出现 .git/ 文件夹，其中包含 config, description, HEAD等相关文件，不要删除，也不要随意修改。

4.1.3 设置签名

1. 形式

- 用户名：tom
- Email address: goodMorning@atguigu.com（用户名与email没有直接联系）

2. 作用：区分不同开发人员身份
3. 注意：此本地库设置的签名与登录远程库（代码托管中心）的账号、密码没有任何关系。
4. 命令：

- git config user.name tom（项目级别/仓库级别签名）
- git config user.email goodMorning@atguigu.com（项目级别/仓库级别签名）
- git config --global user.name tom（系统用户级别签名）
- git config --global user.email goodMorning@atguigu.com（系统用户级别签名）

5. 签名级别规则

- 项目级别/仓库级别：仅在当前本地库范围内有效
  - 使用命令后信息储存位置：本地库path/.git/config
- 系统用户级别：登录当前操作系统的用户范围（例如，登录Windows系统的用户名）
  - 使用命令后信息储存位置：用户家目录下~/.gitconfig
- 级别优先级：就近原则（项目级别优先于系统用户级别，二者都有时采用项目级别；

4.1.4 添加提交和检查状态命令

- git status检查当前local repository下文件状态：查看工作区、暂存区状态
- vim编辑器comand
  - 进入vim编辑器页面后→“:set nu”可以显示行号，“i”键进入编辑模式
- git add [file name] 添加操作：将工作区的“新建/修改”添加到暂存区
- git commit – m "commit message" [file name] 提交操作：将暂存区的内容提交到本地库

4.1.5 版本穿梭

1. git log: 可以看到所有版本更新记录
   a. 多屏显示控制方式
      i. 空格向下翻页
      ii. b向上翻页
      iii. q退出
   b. git log ––pretty=oneline 更漂亮的形式显示
   c. git log ––oneline 中hash值只显示一部分，看起来更清洁
   d. git reflog 中多了HEAD@{num}，num为移动到当前版本需要的步数
2. 前进后退历史版本
   a. 分类
      i. 基于索引值操作即上一步提到的“HEAD”（推荐）
         1. git reset ––hard 9a9ebe0，其中9a9ebe0为局部索引值
      ii. 使用^符号：只能后退
         1. git reset ––hard HEAD^，一个^退一个版本
         2. git reset ––hard HEAD^^退两个版本，以此类推
      iii. 使用~符号：只能后退
         1. git reset ––hard HEAD~3，退后3个版本
   b. Git文档查看tips：git help reset，help后面加要查看的命令
   c. reset命令的三个参数对比（––hard，––soft，––mixed）
      i. ––soft：仅仅在本地库移动HEAD指针
      ii. ––mixed：在本地库移动HEAD指针，重置暂存区
      iii. ––hard：在本地库移动HEAD指针，重置暂存区，重置工作区 （一般推荐使用 hard，因为hard使本地库、暂存区、工作区保持一致）
3. 删除文件并找回

a. 前提：删除前，文件存在时的状态提交到了本地库

b. 操作：git reset --hard HEAD或者git reset --hard [指针位置]

i. 指针位置：历史记录或当前位置

ii. 若删除操作已经提交到本地库：指针位置为历史记录

iii. 若删除操作未提交到本地库：指针位置为当前版本位置

4. 比较文件差异

a. git diff [文件名]

i. 将工作区中的文件和暂存区进行比较

b. git diff [本地库中历史版本][文件名]，eg. git diff HEAD^ apple.txt

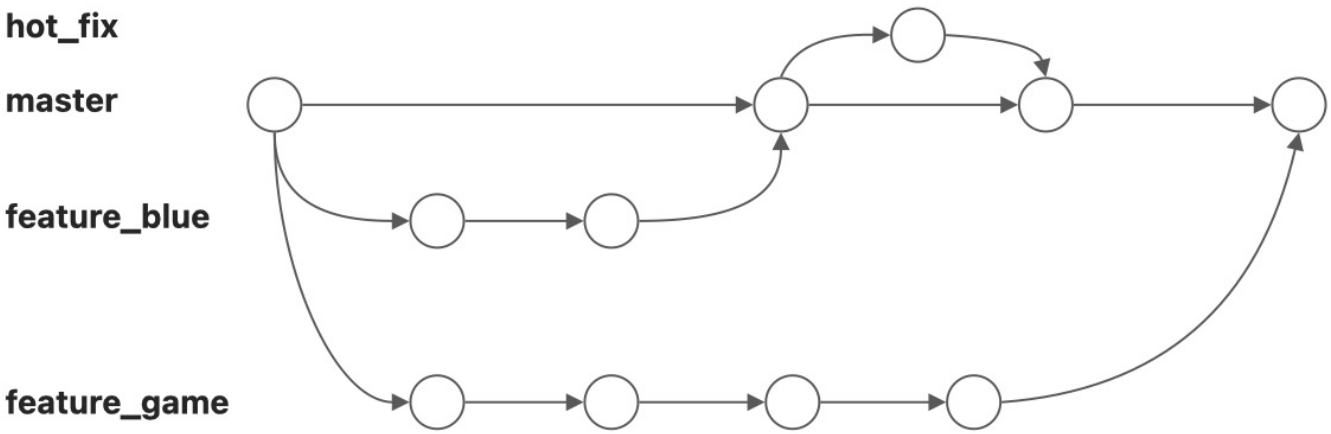
i. 将工作区中的文件和本地库历史记录比较

c. 不带文件名则比较多个文件

4.1.6 分支管理

1. 什么是分支?

在版本控制过程中，使用多条线同时推进多个任务。



2. 分支的好处

a. 同时并行推进多个功能，提高开发效率

b. 各个分支在开发过程中，如果某一个分支开发失败，不会对其他分支有任何影响

3. 分支操作

a. 创建分支

- git branch [分支名]

b. 查看分支

- git branch –v

c. 切换分支
   - git checkout [分支名]

d. 合并分支
   i. 第一步：切换到接受修改的分支上（被合并的分支）
      - git checkout [被合并分支名]
   ii. 第二步：执行 merge 命令
      - git merge [有新内容分支名]

e. 解决冲突
   i. 例：如下命令尝试把 master中的修改合并至hot_fix分支
   ii. 冲突的表现

```
Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/WeChat (hot_fix)

$ git merge master

Auto-merging good.txt

CONFILICT (content): Merge conflict in good.txt

Automatic merge failed; fix conflicts and then commit hte result.

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/WeChat (hot_fix|MERGING)

$ vim good.txt
```

aaaaaa

bbbbbb

cccccc

UUUUUUUUUU

ddddddddd

eeeeeeeee

fffffffffff ########

ggggggggg

<<<<<<<< HEAD 当前分支内容

hhhhhhhhh edit by hot_fix

========

hhhhhhhhh edit by master 另一分支内容

>>>>>>>> master

iiiiiiiii

jjjjjjjj

```
Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/WeChat (hot_fix|MERGING)

$ git add good.txt

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/WeChat (hot_fix|MERGING)

$ git status

On branch hot_fix

All conflicts fixed but you are still merging

  (use "git commit" to conclude merge)

Changes to be committed:

                modified:      good.txt

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/WeChat (hot_fix|MERGING)

$ git commit – m "resolve conflict" good.txt

fatal: cannot do a partial commit during a merge

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/WeChat (hot_fix|MERGING)

$ git commit – m "resolve conflict"

[hot_fix 235081a] resolve conflict

//---This shows that we have successfully merged the branch in 本地库

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/WeChat (hot_fix)

$ git status

On branch hot_fix

nothing to commit, working tree clean
```

iii. 冲突的解决步骤总结

- 第一步：手动编辑冲突文件，删除特殊符号
- 第二步：把文件修改到满意的程度，保存退出
- 第三步：git add [文件名]
- 第四步：git commit – m "日志信息"
  - 注意：此处 commit 后面一定不能带具体文件名

4.1.7 Git 的基本原理

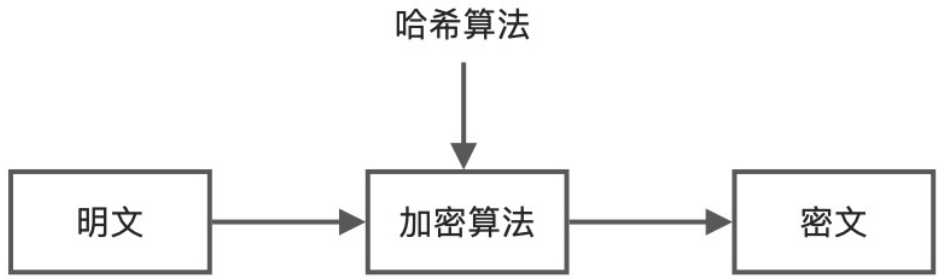
1. 哈希（SHA）

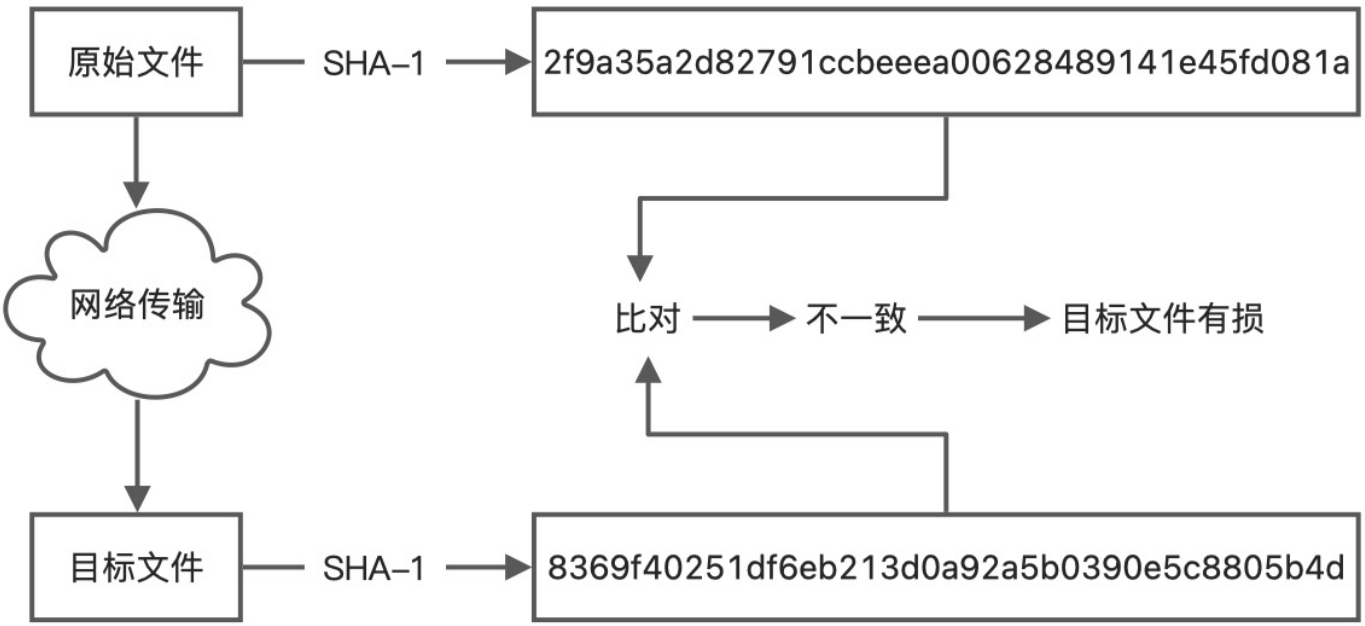
哈希是一个系统的加密算法，各个不同的哈希算法虽然加密程度不同，但是有以下几个共同点：

- 不管输入数据的数据量有多大，输入同一个哈希算法，得到的加密结果长度固定。
- 哈希算法确定，输入数据确定，输出数据能够保证不变
- 哈希算法确定，输入数据有变化，输出数据一定有变化，而且通常变化很大
- 哈希算法不可逆

Git 底层采用的是 SHA-1 算法。

根据哈希算法的特性，哈希算法可以被用来验证文件一致性，原理如下图：



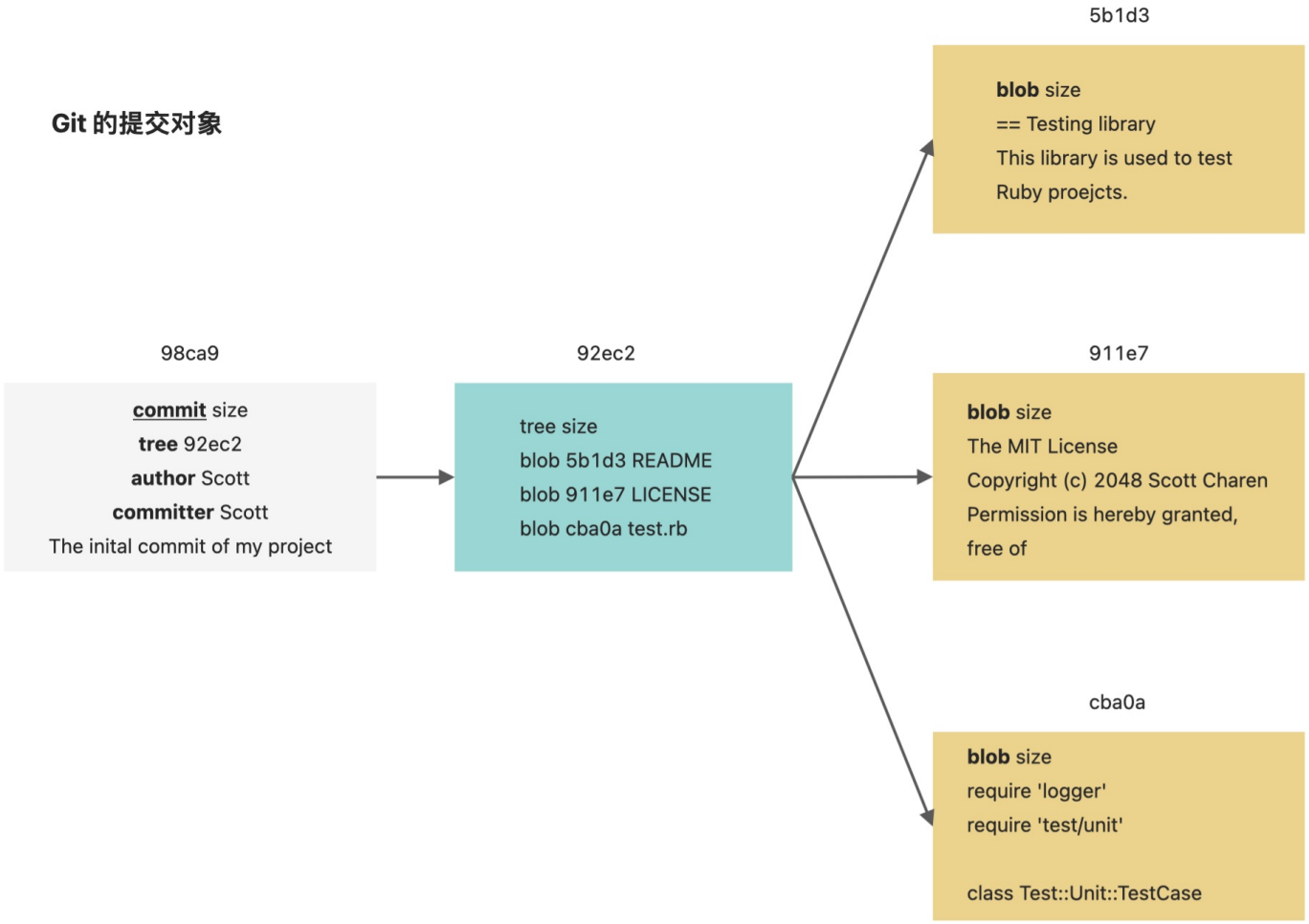


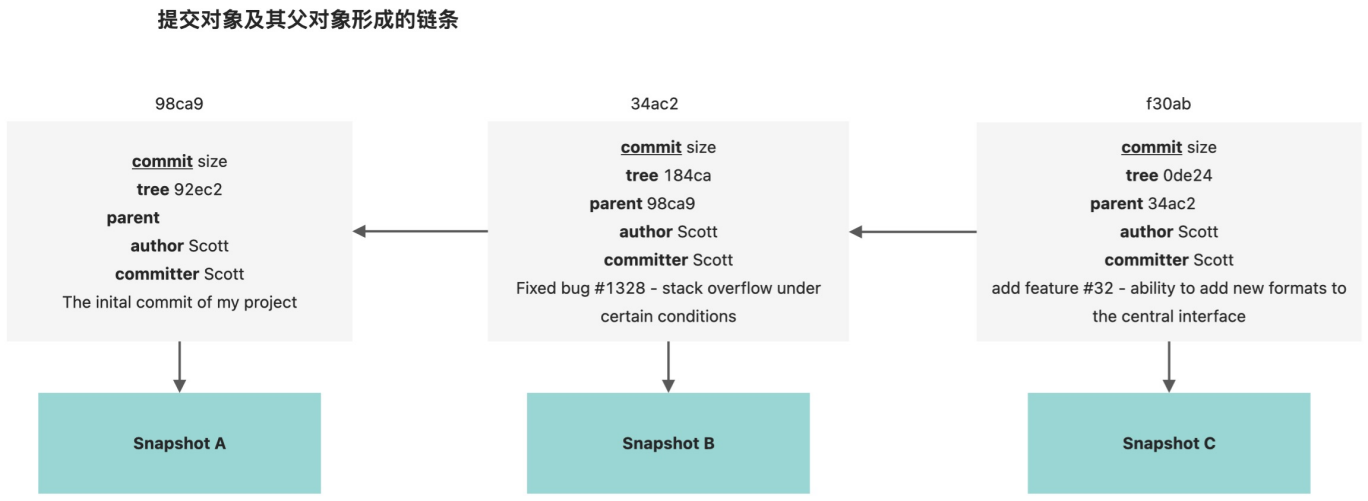
2. Git 保存版本的机制
    a. 集中式版本控制工具的文件管理机制
        i. 以文件变更列表的方式储存信息。这类系统将他们保存的信息看作是一组基本文件和每个文件随时间逐步累积的差异。

b. Git 的文件管理机制

i. Git 把数据看成是小型文件系统的一组快照。每次提交更新时，Git都会对当前的全部文件制作一个快照并保存这个快照的索引。为了高效，如果文件没有修改，Git 不再重新储存该文件，而只是保留一个链接指向之前存储的文件。

ii. 细节



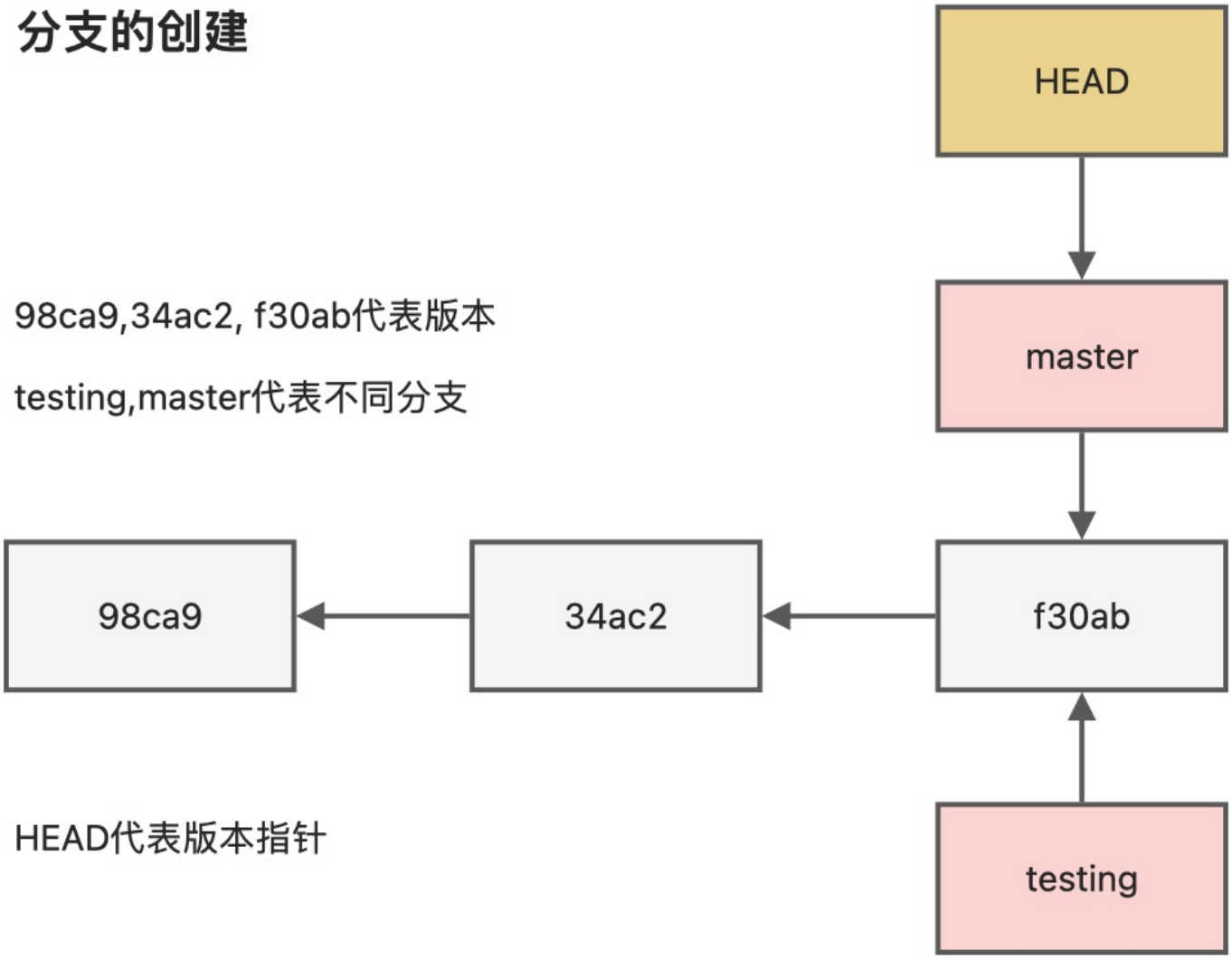


3. Git 分支管理

a. Git 分支的操作只在于切换 HEAD 指针的位置

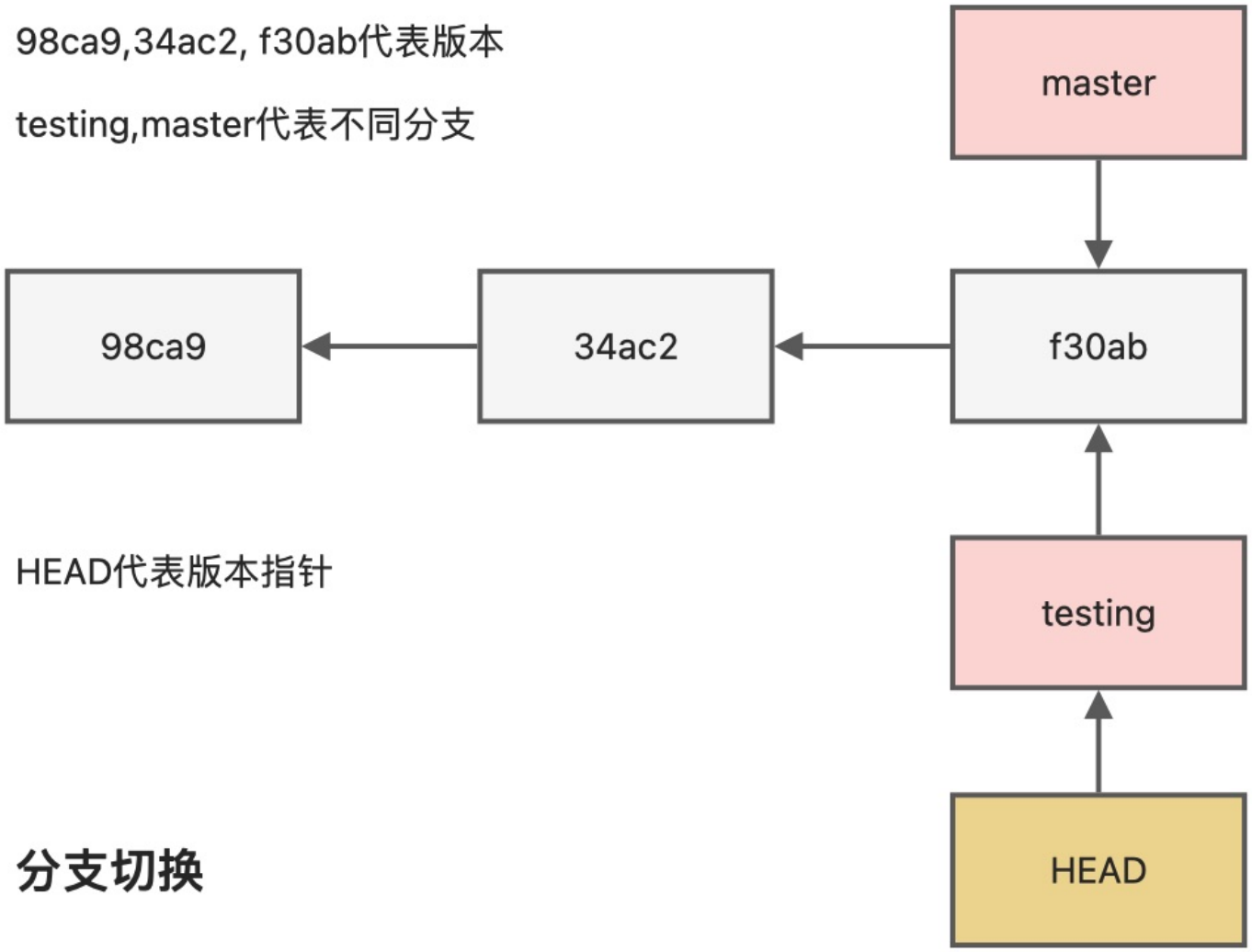
b. 如下图

分支的创建



98ca9,34ac2, f30ab代表版本

testing,master代表不同分支



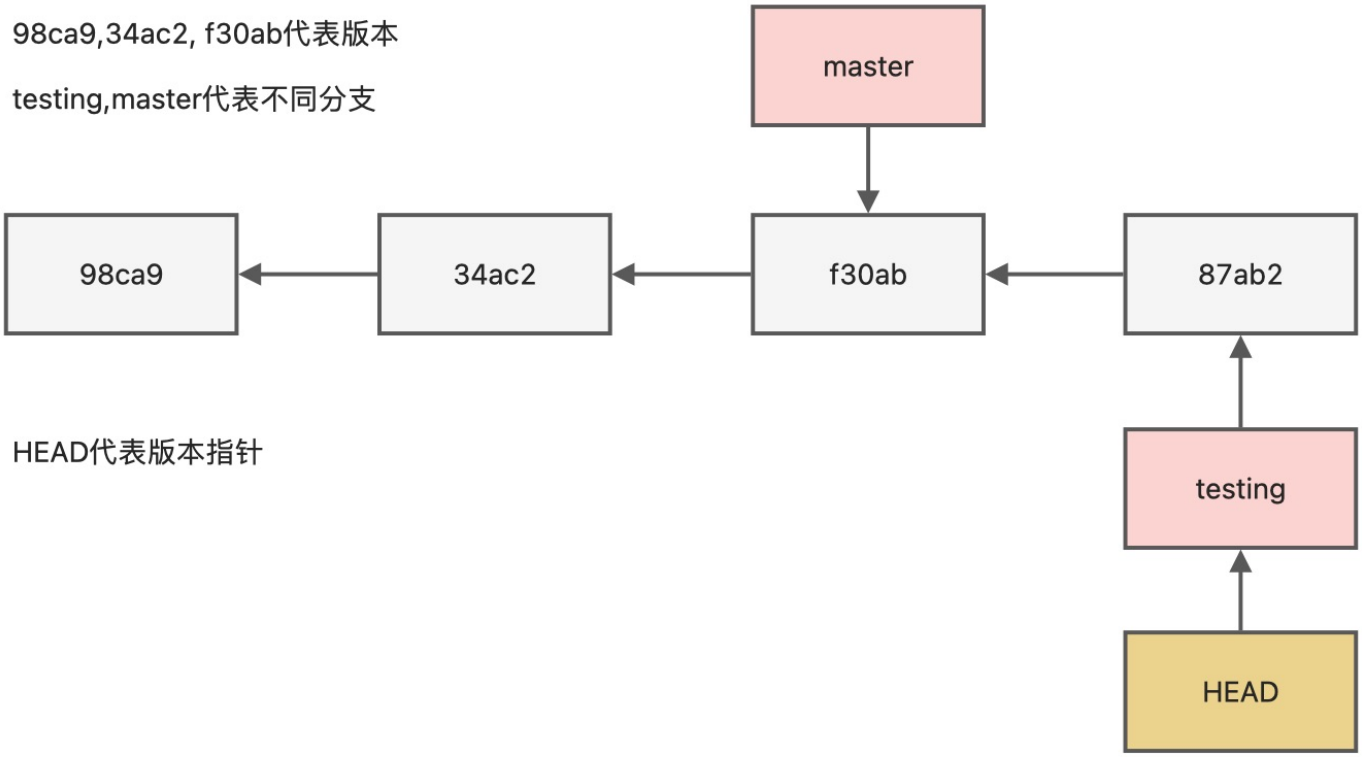
HEAD代表版本指针

分支切换

分支操作-对testing分支进行修改

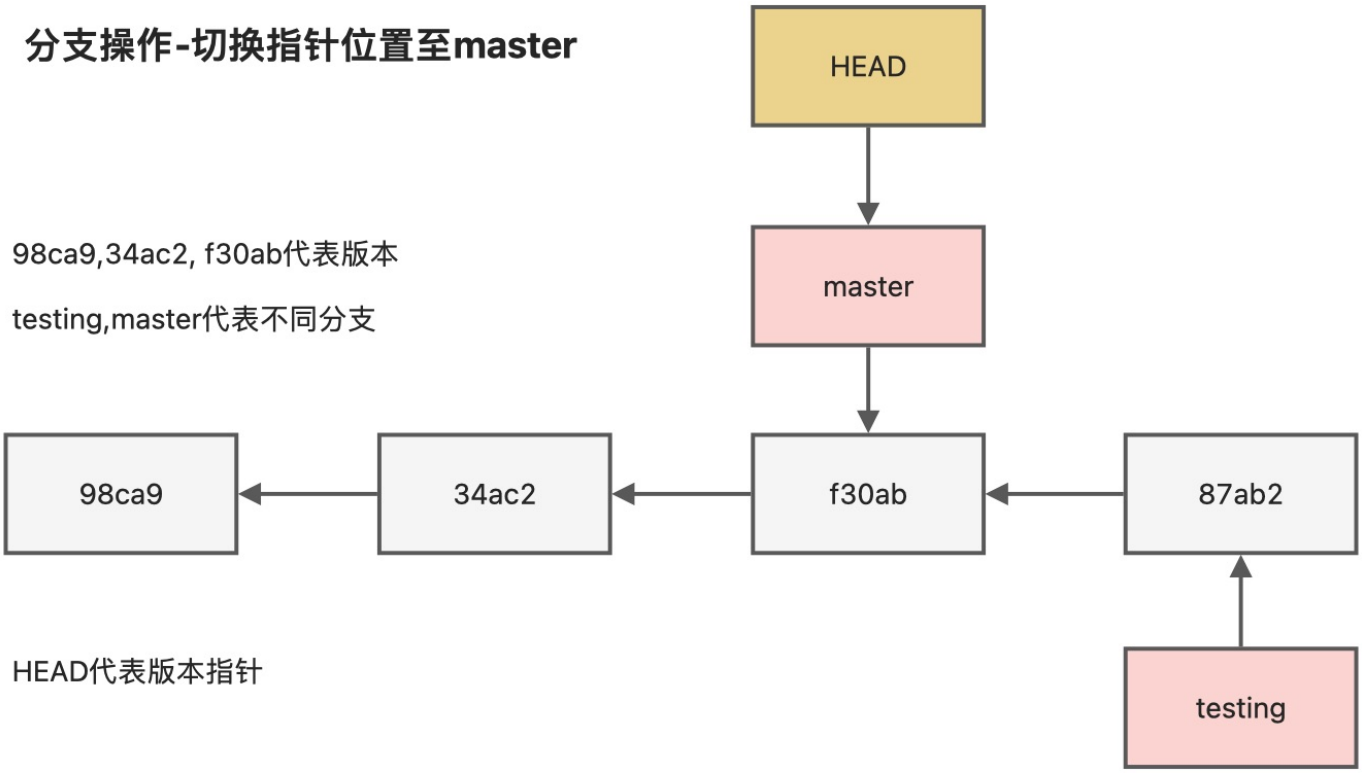
98ca9,34ac2, f30ab代表版本

testing,master代表不同分支



HEAD代表版本指针

分支操作-切换指针位置至master

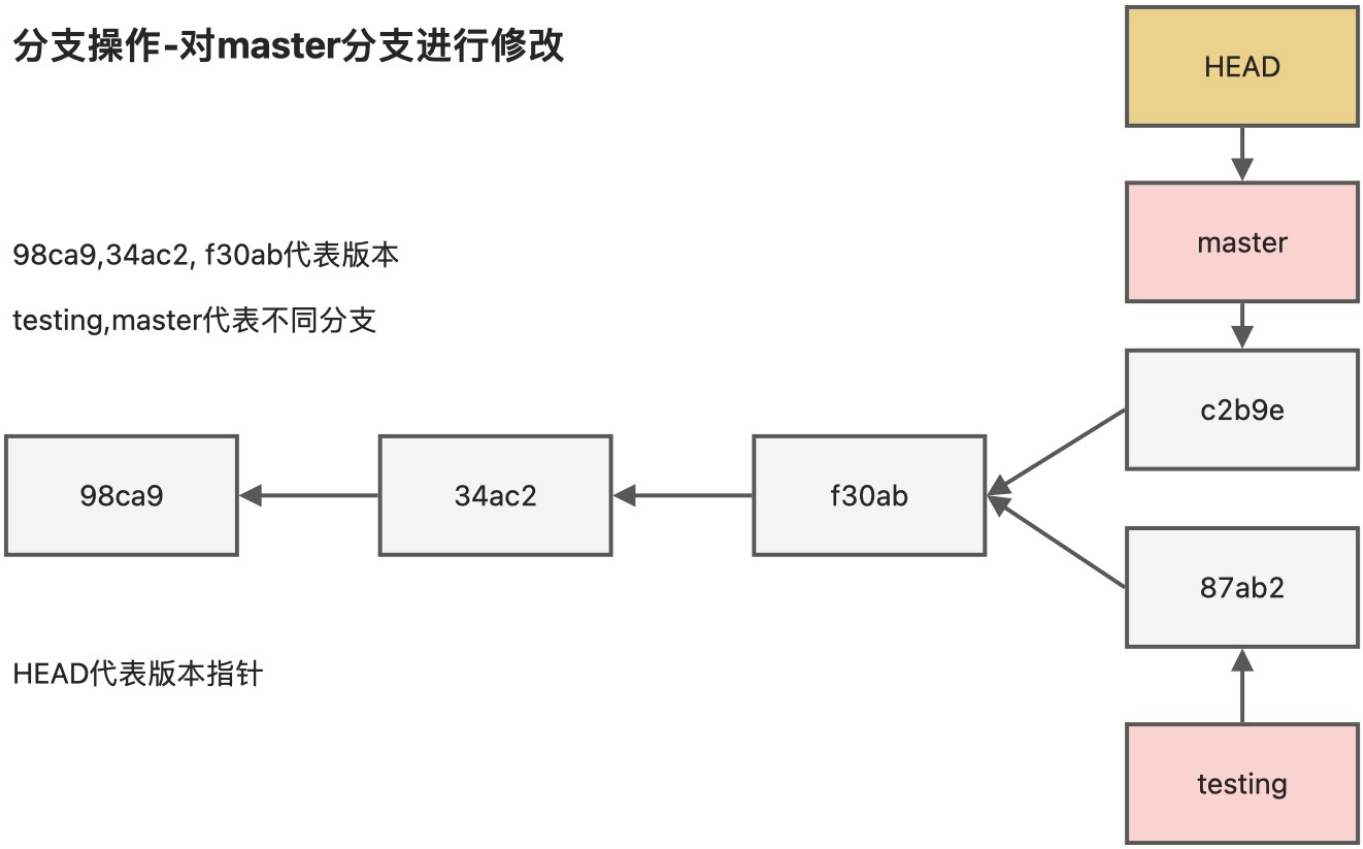


98ca9,34ac2, f30ab代表版本

testing,master代表不同分支

HEAD代表版本指针

分支操作-对master分支进行修改



98ca9,34ac2, f30ab代表版本

testing,master代表不同分支

HEAD代表版本指针

4.2 Github远程库

4.2.1 GitHub账号

https://github.com 注册

4.2.2 创建远程库

1. 前提：已创建本地库（git init –> git add . –> git commit –m "add file"）
2. 创建新的远程库（create a new repository）

4.2.3 在本地创建远程库地址别名

1. git remote –v 查看已设定的远程库地址别名
2. git remote add origin https://github.com/{$username}/{$repository_name}.git（$username and $repository_name needs to be replaced by your username and repository name, drop {}.）
3. 例：

```
Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/huashan (master)
$ git remote -v
Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/huashan (master)
$ git remote add origin https://github.com/atguigu2018ybuq/huashan.git
Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/huashan (master)
$ git remote -v
origin https://github.com/atguigu2018ybuq/huashan.git (fetch)
origin https://github.com/atguigu2018ybuq/huashan.git (push)
```

4.2.4 推送操作

1. **git push origin master** （master代表推送的分支）

4.2.5 克隆操作

1. 命令：**git clone https://github.com/atguigu2018ybuq/huashan.git**
2. git clone命令的三个效果：
    a. 完整地把远程库下载到本地
    b. 创建origin远程地址别名
    c. 初始化本地库

4.2.6 邀请他人加入团队

进入github页面的repository主页，点击settings，找到Collaborators，加入他人账号

4.2.7 远程库修改的拉取（pull命令）

pull是fetch和merge的结合

1. **git fetch [远程库地址别名] [远程分支名], e.g. git fetch origin master** 进行抓取操作。注意：该操作完成后本地文件并没有发生变化。

```
Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/huashan (master)

$ cat huashanjianfa.txt

华山剑法，天下第一！

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/huashan (master)

$ git fetch origin master

remote: Counting objects: 19, done.

remote: Compressing objects: 100% (12/12), done.

...

From https://github.com/atguigu2018ybuq/huashan

* branch            master     -> FETCH_HEAD

3e8ee02..79de212   master     -> origin/master

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/huashan (master)

$ cat huashanjianfa.txt

华山剑法，天下第一！

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/huashan (master)

$ git checkout origin/master

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/huashan (master)

$ cat huashanjianfa.txt

华山剑法，天下第一！

我是令狐冲，我比岳不群还厉害！

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/huashan (master)

$ git checkout master

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/huashan (master)

$ cat huashanjianfa.txt

华山剑法，天下第一！

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/huashan (master)

$ git merge origin/master

Lenovo@DESkTOP-SAV98C0 MINGW64 /d/workspaces/GitSpaceVideo/huashan (master)
```

```
$ cat huashanjianfa.txt

华山剑法，天下第一！

我是令狐冲，我比岳不群还厉害！
```

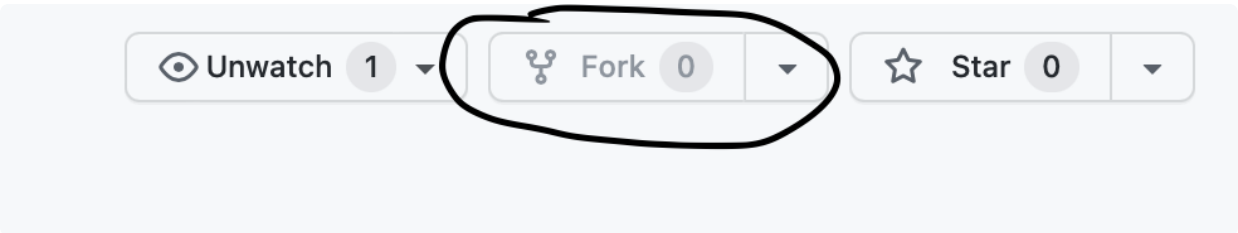
2. git merge [远程库地址别名/远程分支名]，例如：git merge origin/master
3. git pull [远程库地址别名][远程分支名]

4.2.8 协同开发时冲突的解决

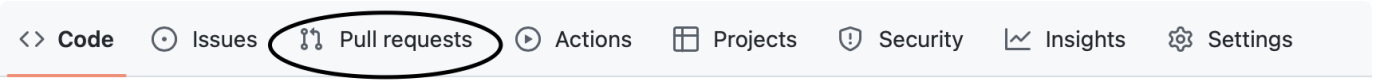
1. 如果不是基于Github远程库的最新版所做的修改，不能被推送，必须先拉取。
2. 拉取下来后如果进入冲突状态（conflict），则按照“分支冲突解决”进行操作即可（见4.1.6分支管理）。

4.2.9 跨团队协作

1. github repository主页选择fork



2. git clone fork后的repository到本地
3. 进行本地修改，然后推送到远程库
4. 进行 pull request –> New pull request –> create pull request



5. 另一边被fork的一方会收到pull request提示，审核代码 –> merge pull request 进行合并
6. 将远程库内容拉取到本地库

4.2.10 SSH登录

1. 进入当前用户的根目录

   $ cd ~

2. 删除.ssh 目录

$ rm –rvf .ssh

3. 运行命令生成.ssh密钥目录

   $ ssh–keygen –t rsa –C atguigu2018ybuq@aliyun.com

   注意：这里的–C必须是大写的C

4. 进入.ssh目录查看文件列表

   $ cd .ssh

   $ ls –lF

5. 查看 id_rsa.pub 文件内容

   $ cat id_rsa.pub

6. 复制 id_rsa.pub 文件内容，登录 Github，点击用户头像→Settings→SSH and GPG keys
7. New SSH Key
8. 输入复制的密钥信息
9. 回到Git bash创建远程地址别名