版本控制：

协同修改

数据备份

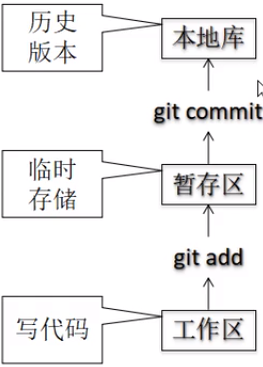
版本管理

分布式版本控制（git）和集中式版本控制

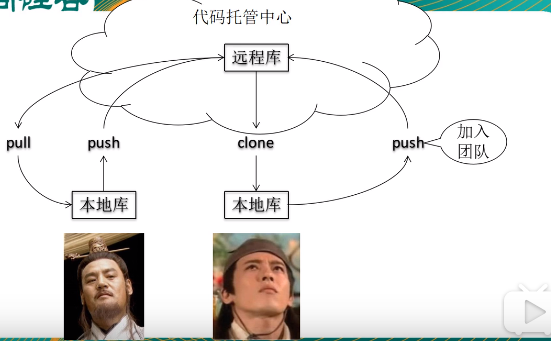
权限控制

历史记录

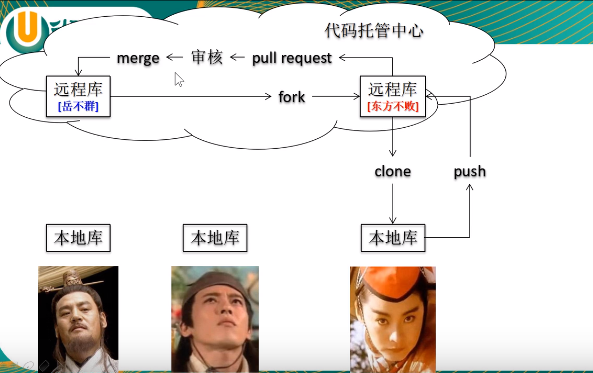
分支管理



团队内部协作：



跨团队协：



Git操作

本地库初始化：

1.目录下git bash git init 初始化

2.ll –lA 查看隐藏目录

· 3..git目录中存放的是本地库相关的子目录和文件，不要删除修改

设置签名:

用户名：

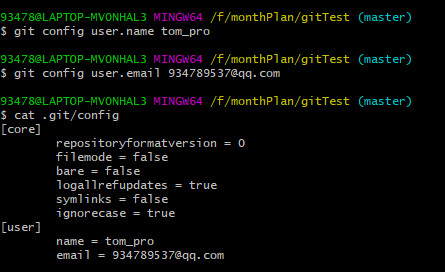
Email地址：

作用：区分不同开发人员的身份

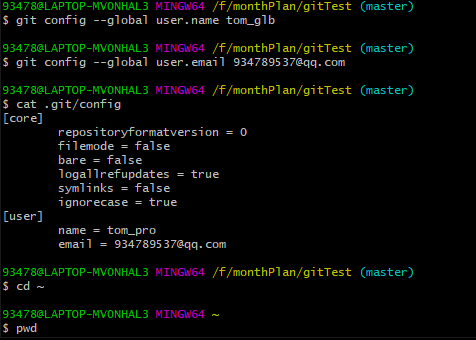
辨析：

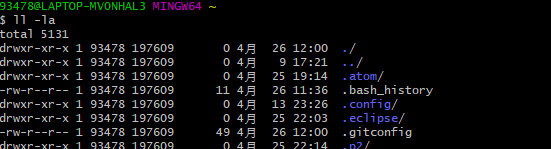
命令：

1. 项目/仓库级别：仅在当前本地库范围有效 （优先级高）
   1. Git config user.name Tom\_pro
   2. Git config user.email ……
   3. 信息保存目录：.git/config



1. 系统用户级别：登录当前操作系统的用户范围
   1. Git config –global user.name Tom\_pro
   2. Git config user.email ……
   3. 信息保存目录：~/.gitconfig







4.3 专属命令

4.3.1 状态查看操作

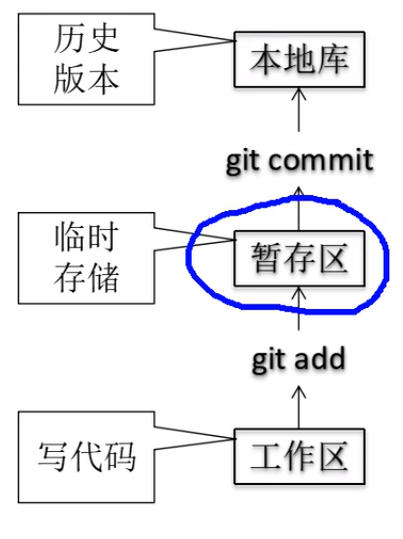
Git status 查看工作区、暂存区状态

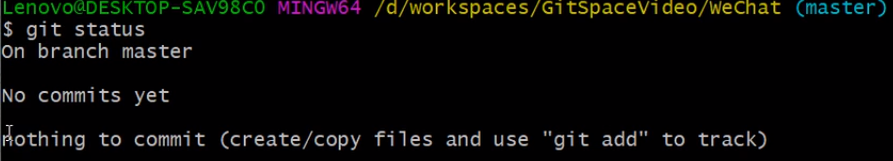
4.3.2 添加操作

Git add [file name] 将工作区的”新建/修改”添加进暂存区

4.3.3 提交操作

Git commit –m “commit message” [file name] 将暂存区的内容提交到本地库





On branch master 在master的分支上

No commits yet 也没有提交

Nothing to commit 没有什么可以提交的

Git rm –cached goog.txt 将goog.txt项目移出暂存区

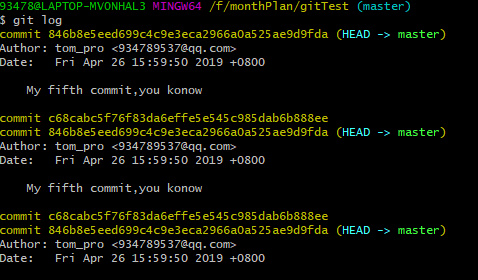
Git commit goog.txt

版本的前进后退

Git log 查看日志 q键退出日志

4.3.4查看历史记录

Git log



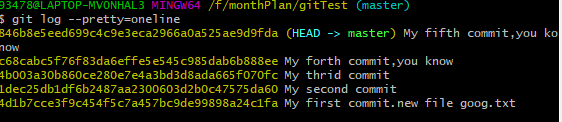
多屏显示控制方式：

空格向下翻页

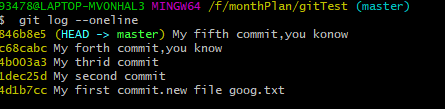
B 向上翻页

Q 退出

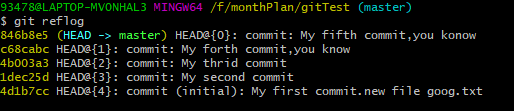
Git log –pretty=oneline



Git log –oneline （当前日志）



Git reflog（显示所有日志）

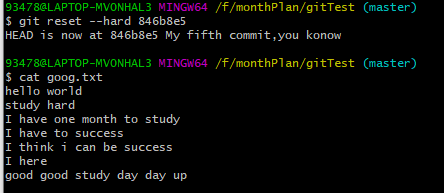


HEAD@{1} 当前版本指针需要移动多少步移动到需要的版本

4.3.5 前进后退

* 基于索引值操作（推荐）

Git reset –hard c68cabc(索引值) 指针跳转到这个版本

****

* **^ 只能后退**

**Git reset -–hard HEAD^^^**

**一个^后退一步**

****

* **~ 只能后退**

**git reset –hard HEAD~3**

**~后面数字为几后退几步**

****

**4.3.6 reset命令的三个参数对比**

**--Soft参数**

**仅仅在本地库移动指针（本地库移除返回版本，工作区和缓存区没有变）**

**--mixed 参数 （本地库、缓存区移除返回版本，工作区没有变）**

**在本地库移动HEAD**

**重置暂存区**

**--hard参数 （本地库、缓存区、工作区都移除返回版本）**

**在本地移动HEAD**

**重置暂存区**

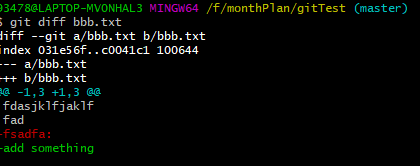
**重置工作区**

**4.3.7 删除文件并找回**

* + - **前提：删除前，文件已经提交到本地库**
    - **操作：git reset –hard [指针位置索引]**
      * **删除操作已经提交到本地库：指针位置指向历史记录**
      * **删除操作尚未提交到本地库：指针位置使用HEAD**

**4.3.8 比较文件差异**

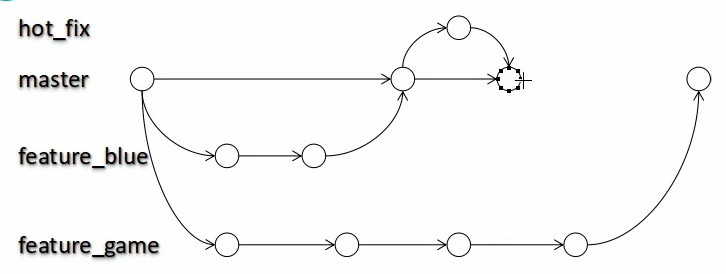
* + - **Git diff[file name]**
      * **将工作区中的文件和缓存区进行比较**
    - **Git diff[本地库中历史版本][file name]**
      * **将工作区中的文件和本地库历史记录比较**
    - **不带文件名比较多个文件**

****

**4.4分支管理**

**4.4.1 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**



**4.4.2 分支的好处**

* + **同时并行推进多个功能开发，提高开发效率**
  + **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，失败的分支删除重新开始即可**

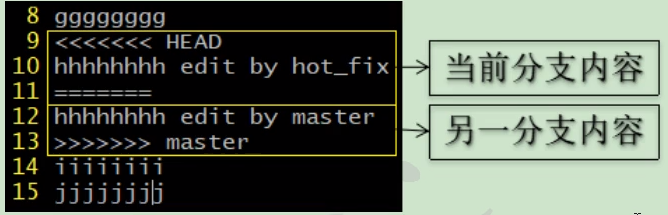
**4.4.3 分支操作**

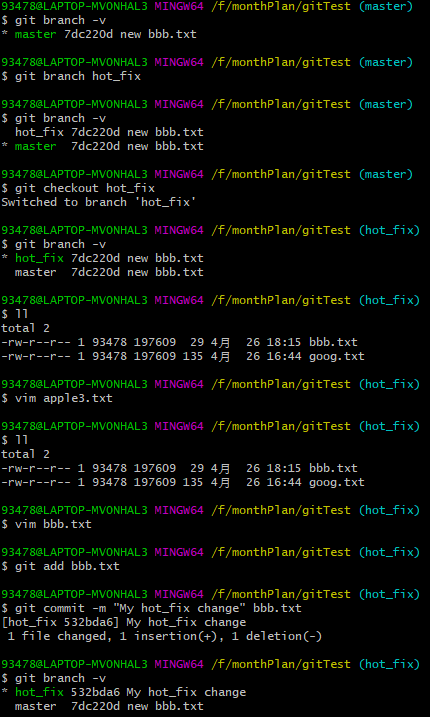
* + **创建分支**
    - **Git branch[分支名]**
  + **查看分支**
    - **Git branch -v**
  + **切换分支**
    - **Git checkout[分支名]**
  + **合并分支**
    - **第一步：切换到接受修改的分支上（被合并，增加新内容）**

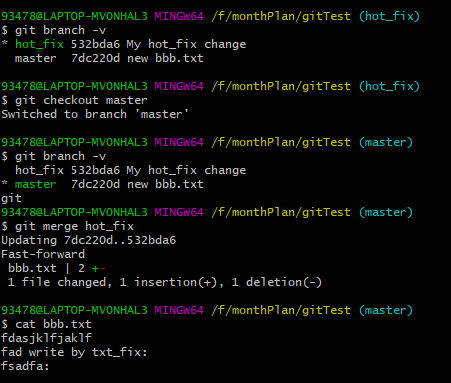
**Git checkout[被合并的分支名]**

* + - **第二步：执行merge命令**

**Git merge[有新内容的分支名]**

* + **解决冲突**
    - **冲突的表现**
    - ****
    - **冲突的解决**
      * **第一步：编辑文件，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 [file name]**
      * **第四步: git commit –m “日志内容”** 
        + **注意：此时commit一定不能带具体的文件名**

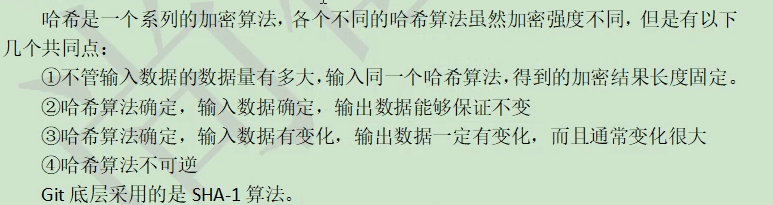
****

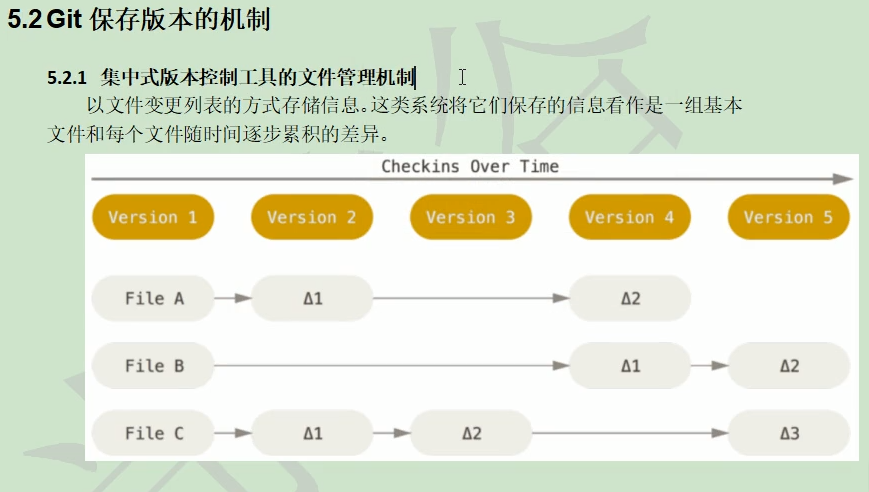
****

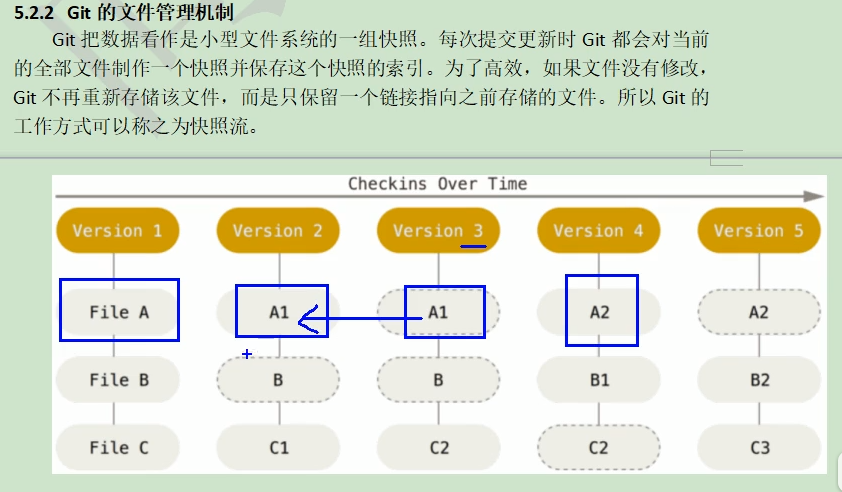
**5.Git基本原理**

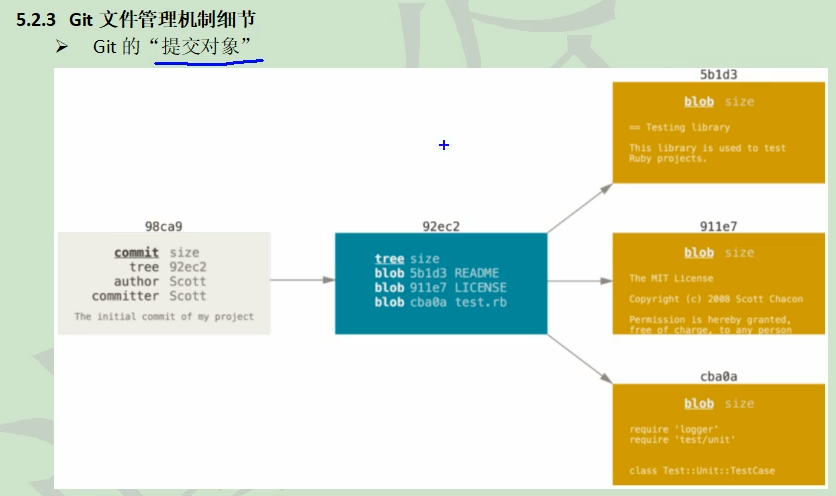
**5.1哈希**

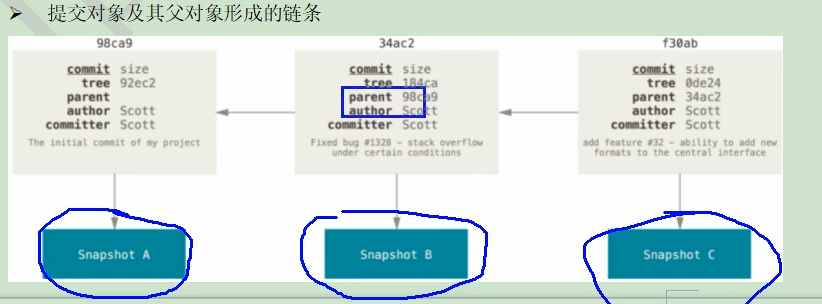
**哈希是一种加密算法**

****

****

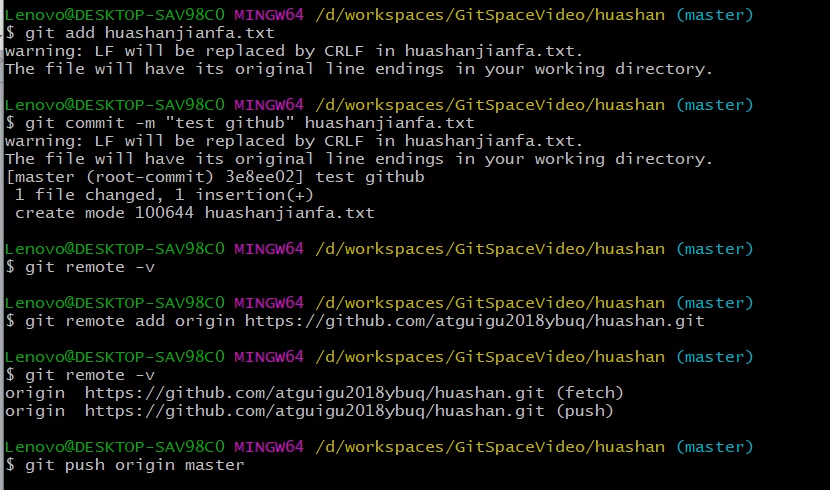
****

****

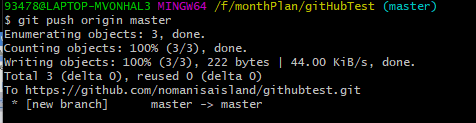
****

**远程库**

**Remote用来储存网址方便记忆**

****

**Git push origin master 将本地库文件推送到远程库**

****

**Git clone xxxxxxxx 克隆远程库项目**

**Clone的三种效果**

* **完整的把远程库下载到本地**
* **创建origin远程地址别名**
* **初始化本地库**

**组员想要提交代码需要被邀请加入**

****

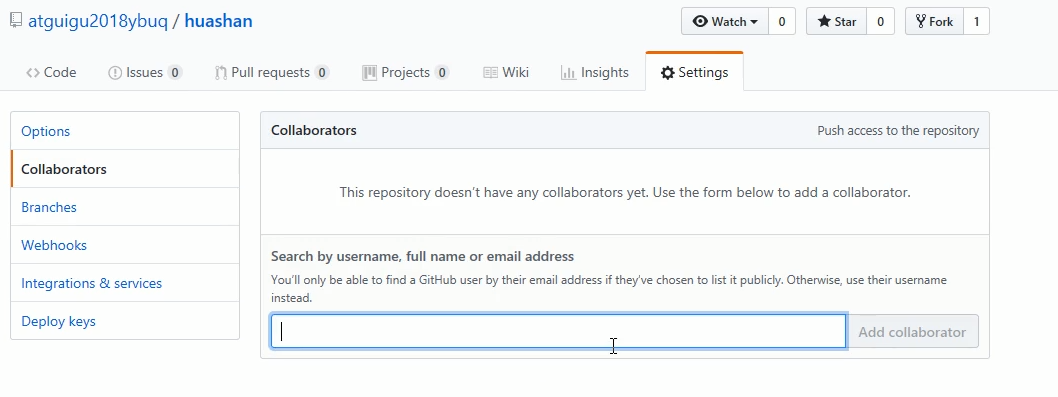
****

**此处可以删除git账号密码**

**Window7没有这个可以用shh登录，避免每次都需要登录账号密码**

****

****

**fetch**

**6.7 拉取**

* **Pull=fetch+merge**
* **Git fetch[远程库地址别名][远程分支名]**
* **Git merge[远程库地址别名/远程分支名]**
* **Git pull [远程库地址别名][远程分支别名] 把项目拉取到本地**

**git fetch origin master**

**fetch只是从远程库抓取内容，但是本地文件不会改变**

**git checkout origin/master**

**可以查看到远程库的内容**

**好处，可以查看远程库下载的内容，后续进行比较是否可以合并**

**可以合并后：**

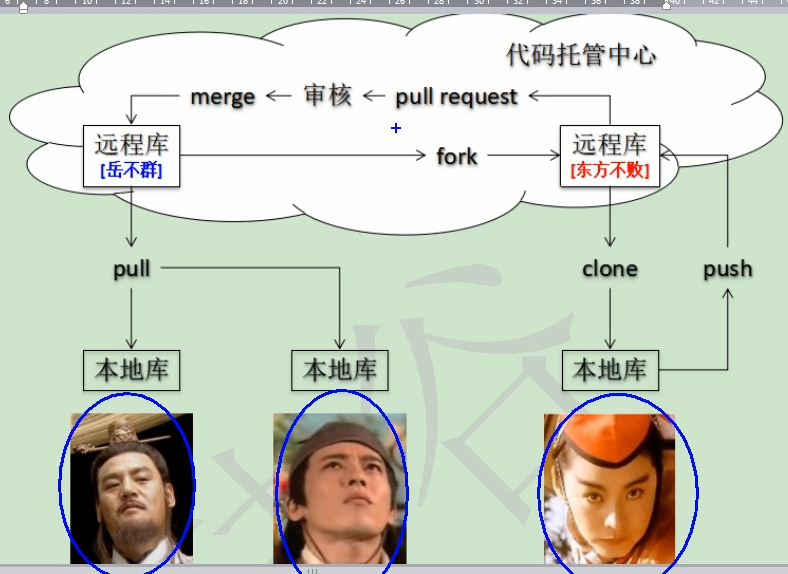
**Git merge origin/master**

**合并远程库内容**

**6.8解决冲突**

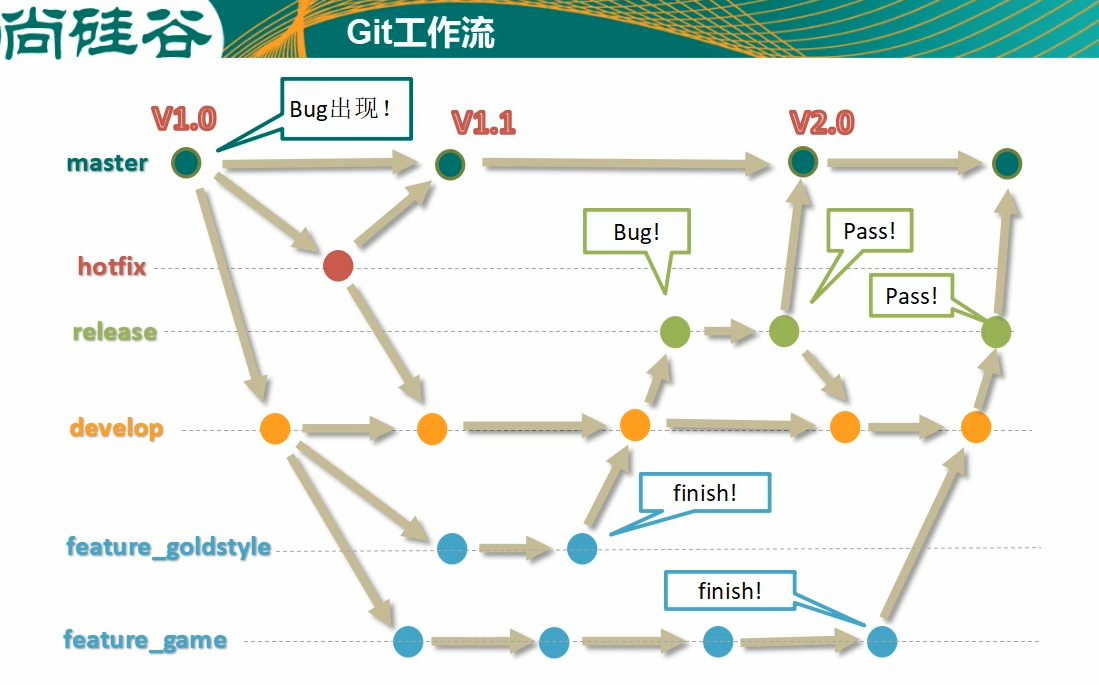
* **要点**
  + **如果不是基于gitHub远程库最新版本所做的修改，不能推送的话，必须先拉取下来**
  + **拉取下来后如果进入冲突状态，则安照分支冲突解决操作即可**
* **类比**
  + **债权人：老王**
  + **债务人：小刘**
  + **老王说：10天后归还 ps：第一次提交者可以提交**
  + **老王媳妇说：5天后归还 ps：第二次提交者不能提交**
  + **老王媳妇需要和老王沟通确认保持一致**

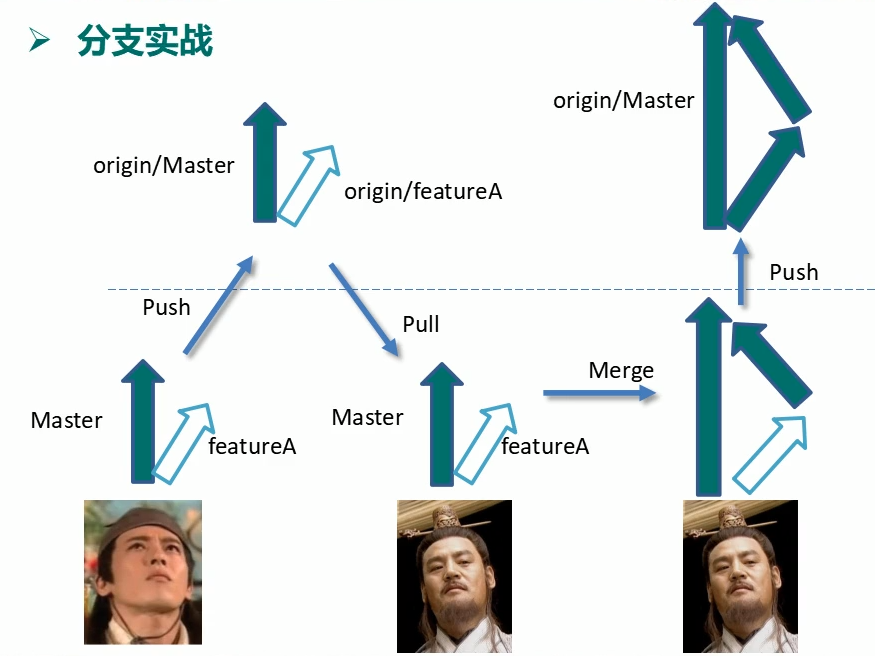
**跨团队协作**

****

**GitHub Fock一下（创建自己的远程库），然后clone到本地，修改提交到远程库，pull request一下就等待审核即可**

**点Merge pull request 合并代码 合并前可以填写一下日志信息**

****

****

**具体操作**

1. **创建分支**
2. **推送到远程库**
3. **切换分支审查代码**
4. **检出远程新分支**
5. **合并分支**

**Gitlab服务器搭建过程 gitlab是自己创建的一个远程库服务器**

1. Java [av47103781](https://www.bilibili.com/video/av47103781/)  
   02、Spring [av47103781](https://www.bilibili.com/video/av47103781/)  
   03、Spring Mvc [av47176832](https://www.bilibili.com/video/av47176832/)  
   04、MyBatis [av47228830](https://www.bilibili.com/video/av47228830/)  
   05、Hibernate [av47382482](https://www.bilibili.com/video/av47382482/)  
   06、Spring Boot [av47230137](https://www.bilibili.com/video/av47230137/)  
   07、Spring Cloud [av47228830](https://www.bilibili.com/video/av47228830/)  
   08、Dubbo [av47009143](https://www.bilibili.com/video/av47009143/)  
   09、MySQL [av47702905](https://www.bilibili.com/video/av47702905/)  
   10、Redis [av47423174](https://www.bilibili.com/video/av47423174/)  
   11、MongoDB [av47425352](https://www.bilibili.com/video/av47425352/)  
   12、Elasticsearch [av48279989](https://www.bilibili.com/video/av48279989/)  
   13、Zookeeper [av47773419](https://www.bilibili.com/video/av47773419/)  
   14、Kafka [av47773990](https://www.bilibili.com/video/av47773990/)  
   15、Linux [av47701443](https://www.bilibili.com/video/av47701443/)  
   16、Docker [av47715282](https://www.bilibili.com/video/av47715282/)  
   17、Maven [av47382482](https://www.bilibili.com/video/av47382482/)  
   18、Git [av47701443](https://www.bilibili.com/video/av47701443/)  
   19、Jenkins [av47714706](https://www.bilibili.com/video/av47714706/)  
   20、IDEA [av47382482](https://www.bilibili.com/video/av47382482/)  
   21、Shiro [av47382482](https://www.bilibili.com/video/av47382482/)  
   22、React [av48033695](https://www.bilibili.com/video/av48033695/)  
   23、Vue [av48034319](https://www.bilibili.com/video/av48034319/)  
   24、电商项目实战 [av48308453](https://www.bilibili.com/video/av48308453/)  
   25、RBAC 权限系统 [av48305719](https://www.bilibili.com/video/av48305719/)  
   26、数据结构与算法 [av48505556](https://www.bilibili.com/video/av48505556/)  
   27、Oracle [av48502473](https://www.bilibili.com/video/av48502473/)  
   28、ActiveMQ [av48562474](https://www.bilibili.com/video/av48562474/)  
   29、TCP/IP [av48562426](https://www.bilibili.com/video/av48562426/)  
   30、HTTP [av48561455](https://www.bilibili.com/video/av48561455/)  
   31、Netty [av48570202](https://www.bilibili.com/video/av48570202/)  
   32、Gradle [av48581119](https://www.bilibili.com/video/av48581119/)  
   33、HBase [av48581792](https://www.bilibili.com/video/av48581792/)  
   34、Kubernetes [av48582456](https://www.bilibili.com/video/av48582456/)  
   35、Hive [av48609427](https://www.bilibili.com/video/av48609427/)  
   36、Hadoop [av48621769](https://www.bilibili.com/video/av48621769/)  
   37、Spark [av48668079](https://www.bilibili.com/video/av48668079/)

**尚硅谷**