# 版本控制

## 版本控制介绍

个人开发：便于查询历史增改

团队开发：便于多人开发、便于查询历史增改

## 版本控制工具介绍

协同修改

数据备份

版本管理：SVN采用增量式管理的方式，Git采用文件系统快照的方式

权限控制

历史记录

分支管理

# Git简介

## Git的优势

大部分操作在本地完成，不需要联网

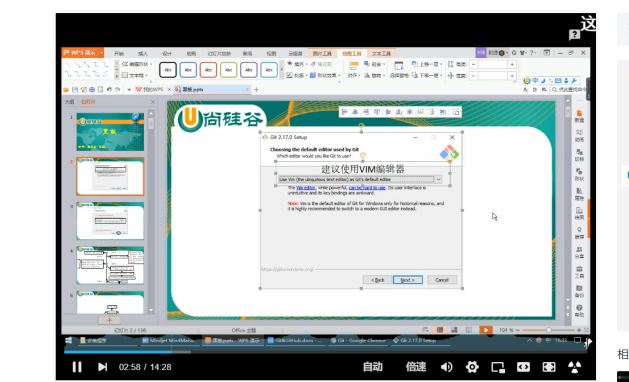
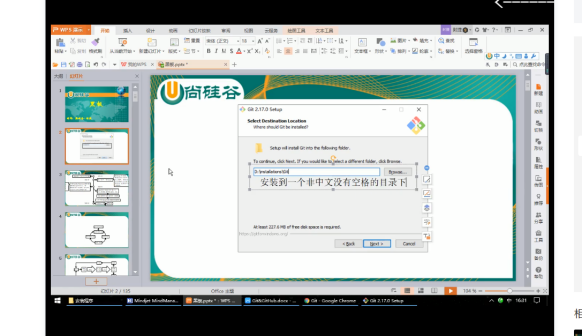
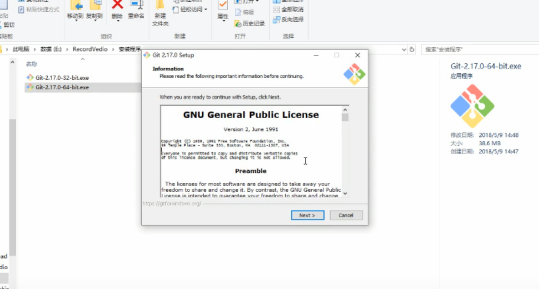
完整性保证

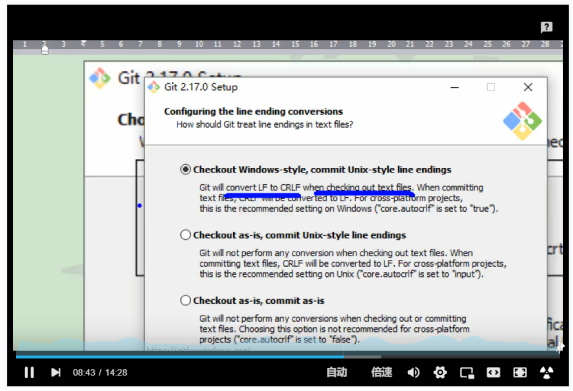
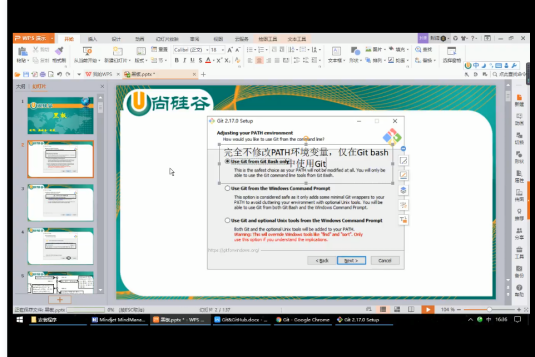
尽可能添加数据而不是删除或修改数据

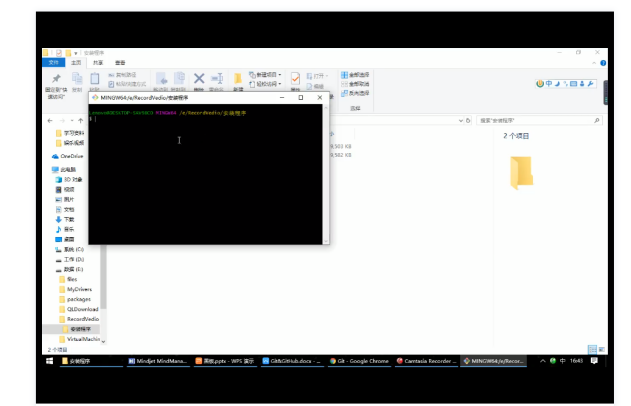
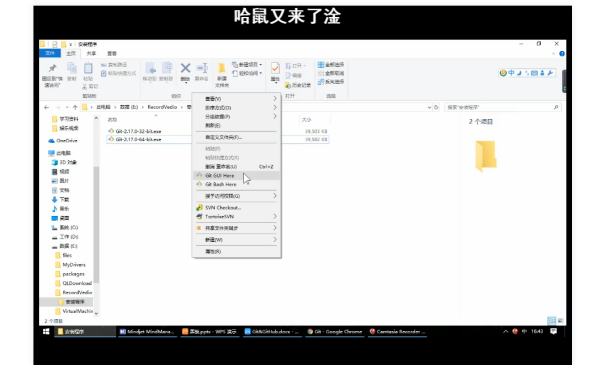
分支操作非常快捷流畅

与Linux命令全面兼容

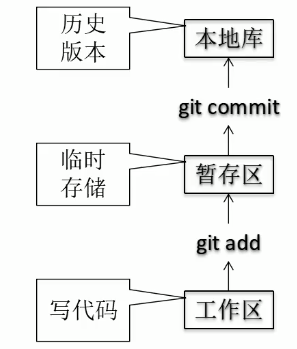
## Git的安装







## Git的结构



## Git和代码托管中心

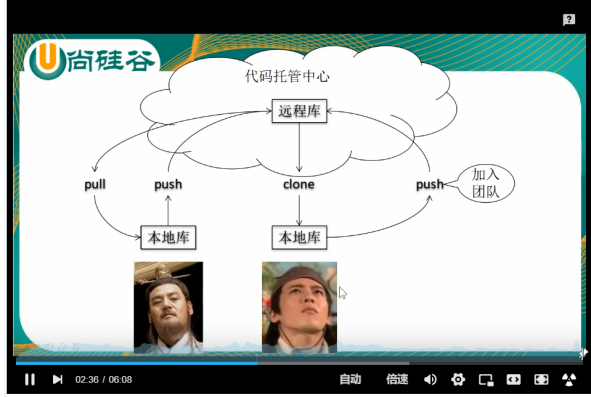
代码托管中心的任务：维护远程库

局域网环境：GitLab服务器

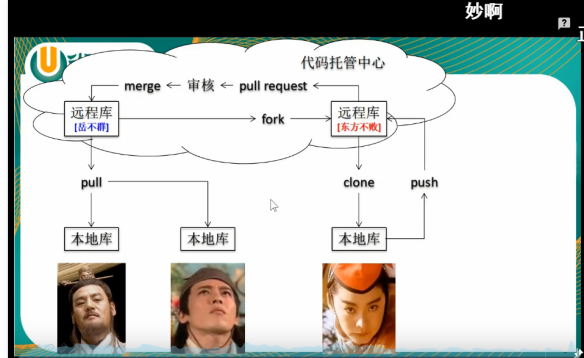
外网环境：GitHub、码云

## Git本地库和远程库的交互方式

团队内协作



跨团队协作



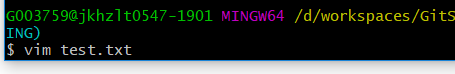
# Git命令行操作

## 本地库操作

### linx相关基础操作

vim [file name]

（没有文件名时新建文件）编辑文件



i为编辑模式

编辑结束后esc退出编辑模式 ：输入指令 wq ，w为写入q为退出

### 本地库初始化

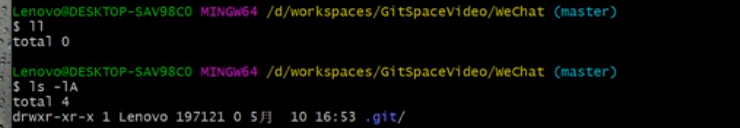
命令：git init

效果：初始化git仓库



命令：ls -la

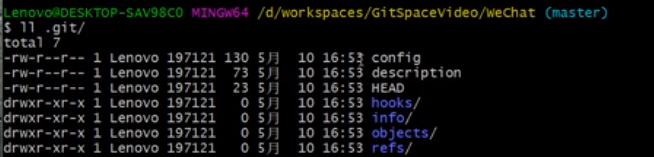
效果：查看隐藏的资源



命令：ll .git

效果：显示.git隐藏文件中内容

注意：.git目录里存放的是本地库相关子目录和文件，不用删除或随意修改。



效果：设置签名

形式： 用户名：XXX

Email地址：XXXX@XXcom

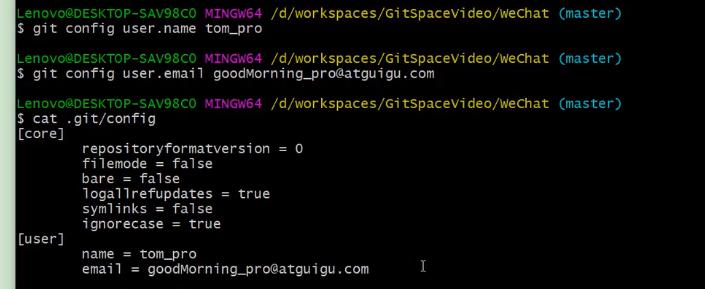
作用：区分不同开发人员的身份

命令： 1、项目级别：仅在当前本地库范围内有效

git config user.name bigDragon

git config user.email [183502947@qq.com](mailto:183502947@qq.com)

信息保存位置 .git/config



1. 系统用户级别：登录当前操作系统的用户范围

git config --global user.name bigDragon

git config --global user.email [183502947@qq.com](mailto:183502947@qq.com)

信息保存位置：~/.gitconfig



1. 优先级

就近原则：项目级别优秀与系统系统级别，二者都有采用项目级别

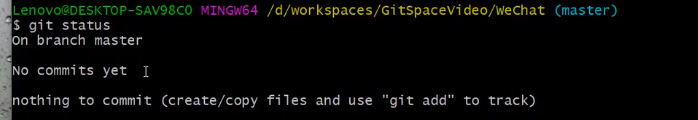
如果只有系统用户级别就以系统用户级别为准

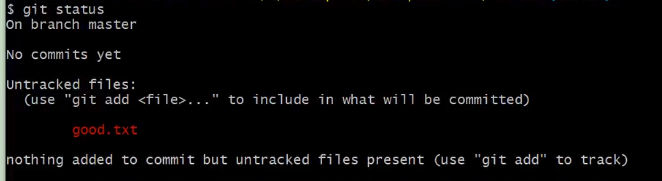
二者都没有不允许

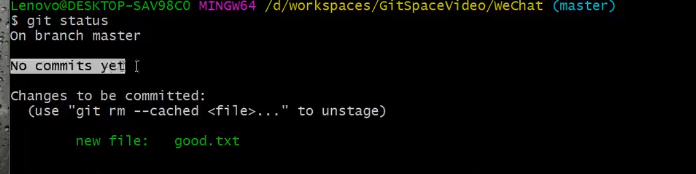
### 基本操作

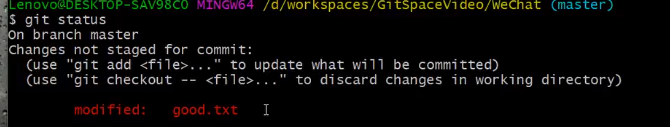
git status

查看工作区、暂存区状态



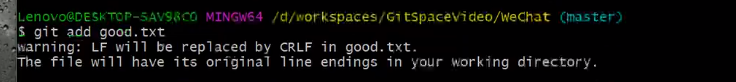






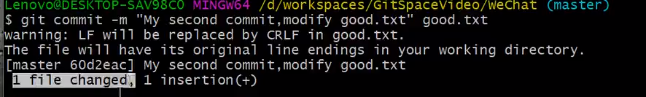
git add [file name]/[.]

将工作区的“新建/修改”添加到暂存区



git commit -m “commit message” [file name]/[.]

效将暂存区的内容提交到本地库



### 日志管理

git log

显示提交日志

哈希值

Head是指针

多屏控制方式：

空格向下翻页

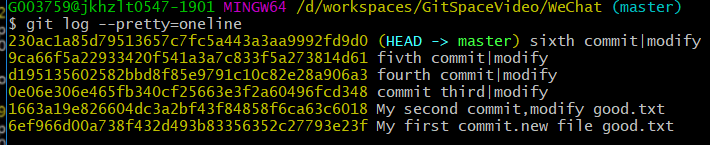
b向上返回

q退出



git log --pretty=oneline

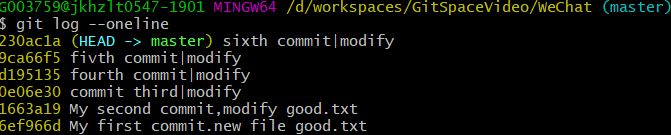
单行显示提交日志



git log --oneline

单行显示提交日志（简）

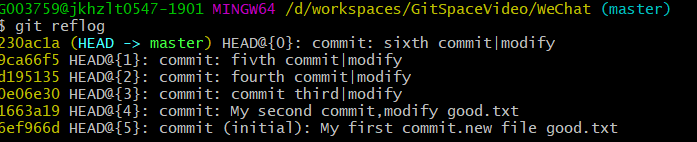
只显示之前的 不显示之后的

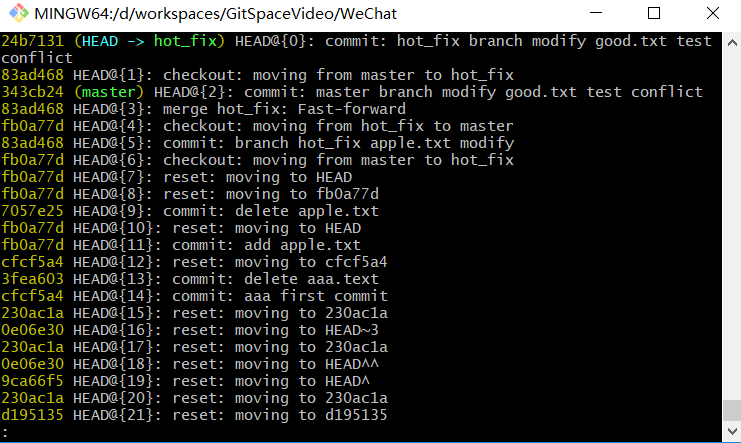


git reflog

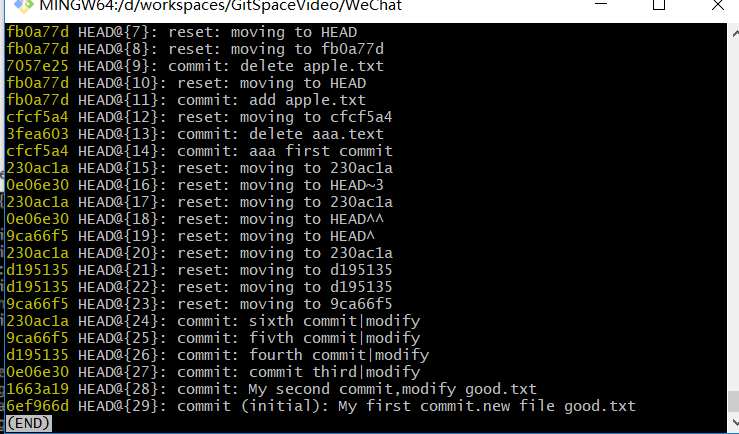
单行显示提交日志（带指针）

HEAD@{移动到当前版本需要多少步}



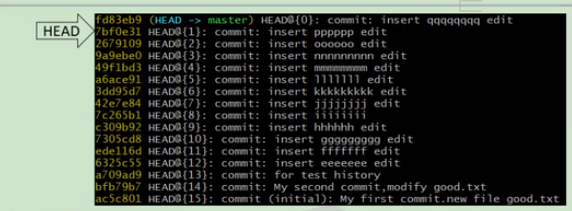


下一页为空格键 上一页为b键 退出为q



### 前进和后退

本质



基于索引值操作[推荐]

git reset --hard [索引值，head的位置hash值]

git reset --hard 230ac1a

想到哪一步就选哪个索引

使用^符号：只能后退

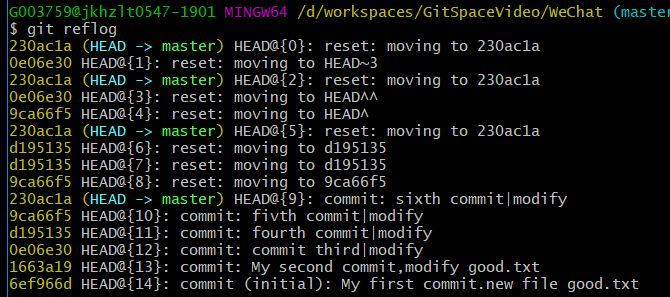
git reset --hard HEAD[^……]

一个^移动一格，n个表示后退n格

使用~符号

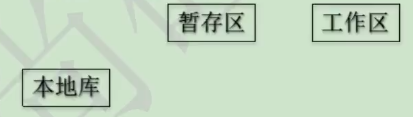
git reset --hard HEAD~n

~后的n数字表示后退n步



### reset命令的三个参数对比

soft 仅仅在本地库移动指针



mixed 本地库HEAD移动指针

重置暂存区



hard 本地库HEAD移动指针

重置暂存区

重置工作区



### 删除文件并找回

前提：删除钱，文件存在时的状态提交的了本地库

rm [file name]删除文件并提交到本地库

仍可通过git reset --hard [索引值（指针位置）]的方式找回版本

删除操作已经提交到本地库：指针位置纸箱历史记录

删除操作尚未提交到本地库：指针位置使用HEAD(最近一次提交的指针位置)

git所有文件所有记录都不不可磨灭的

### 比较文件差异

git diff [file name]/[.]

将工作区的文件与暂存区文件相比较

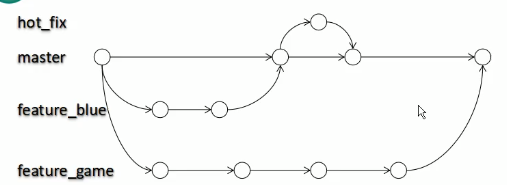
git different [本地库中历史版本] [file name]/[.]

将工作区的文件和本地库历史记录比较

不带文件名将比较多个文件

### 分支管理

概念：在版本控制过程中，使用多条线同时推进多个任务。



优点： 同时并行多个功能开发，提高开发效率

各个分支开发过程中，如果某一个分支开发失败，不会对其他分支有任何影响。失败的分支删除重来即可。

查看当前所有分支 git branch -v



创建分支 git branch [分支名]

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）

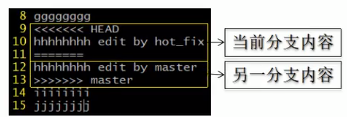
git checkout [被合并分支名]

第二步：执行merge命令

git merge [有新内容分支名]

合并分支时产生冲突

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：把文件修改到蚂蚁的程度，保存退出

第三部：git add [文件名]



第四部：git commit -m “日志信息”

此时commit一定不能带具体文件名



### Git基本原理

哈希

哈希是一系列的加密算法，各个不同的哈希短发虽然强度不同，单有一下几个共同点：

不管输入数据量有多大，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。

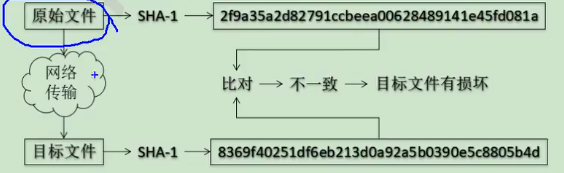
哈希算法确定，输入数据有变化，输出数据一定有变化，而且通常变化很大

哈希算法不可逆

Git底层采用的是SHA-1算法。

哈希算法中有一种算法MD5，无论传入文件类型大小输出都为16字节的密文。

哈希算法可以用来验证文件。



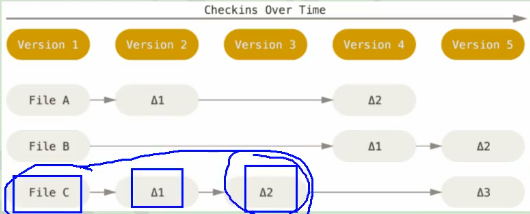
### 保存版本的机制

集中式版本控制工具的文件管理机制：

以文件变更的方式存储信息。这类系统将它们保存的信息看作是一种基本文件和每个文件随时间逐步累积的差异。

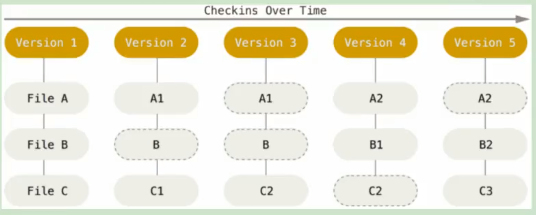
特点：保存每个版本做出的修改，不会保存重复的数据

优点：可以节约服务器存储的空间

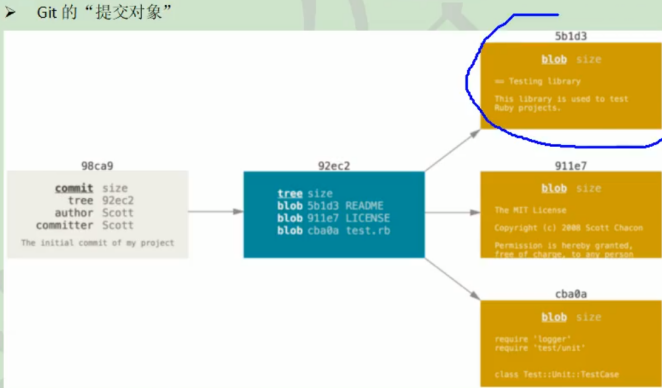


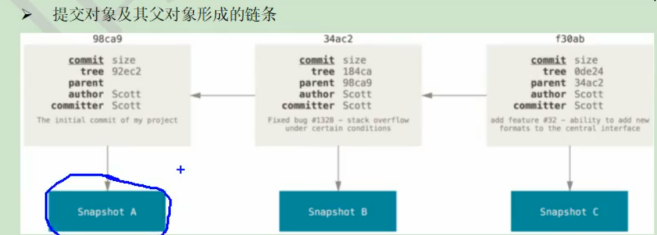
Git的文件管理机制：

Git把数据看作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不在重新存储该文件，而是只保留一个链接指向之前存储的文件。所以Git的工作方式可以称之为快照流。



Git文件管理机制细节

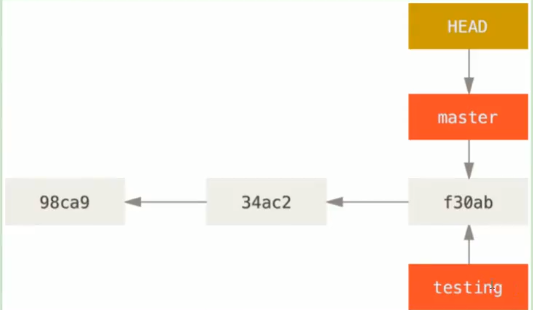




### Git分支管理机制

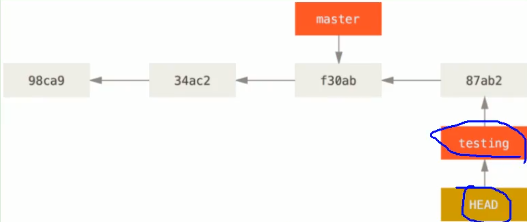
分支的创建

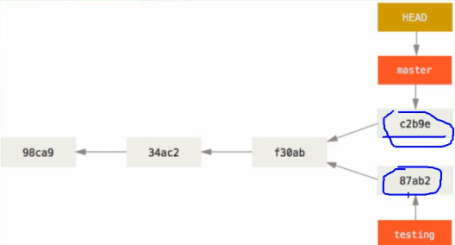
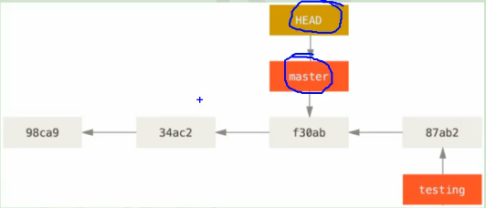
不同分支指向同一个版本



分支的切换

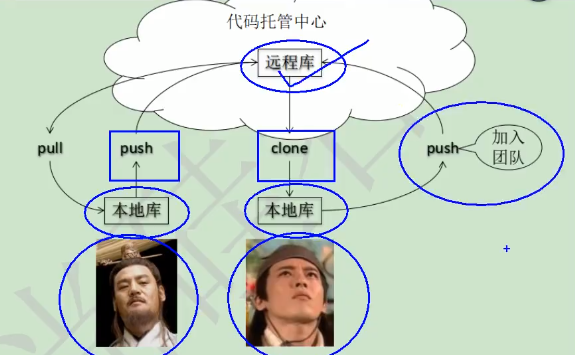
只是切换指针，不存在文件的复制和替换



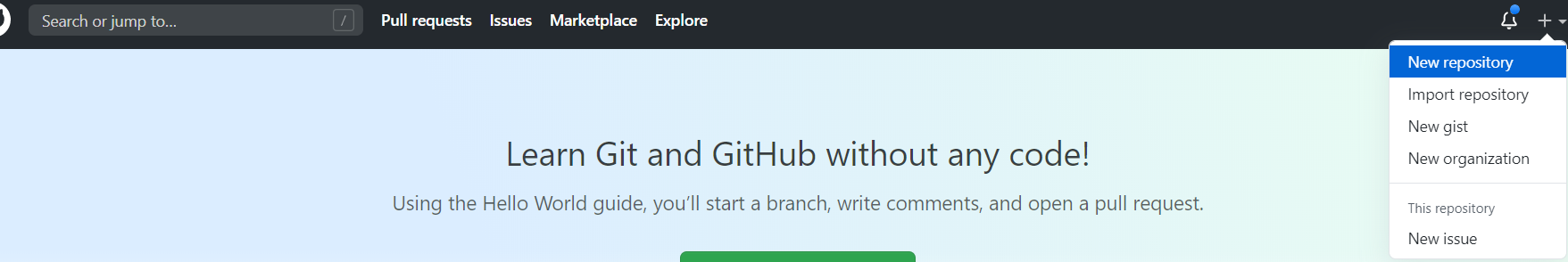


## 远程库操作

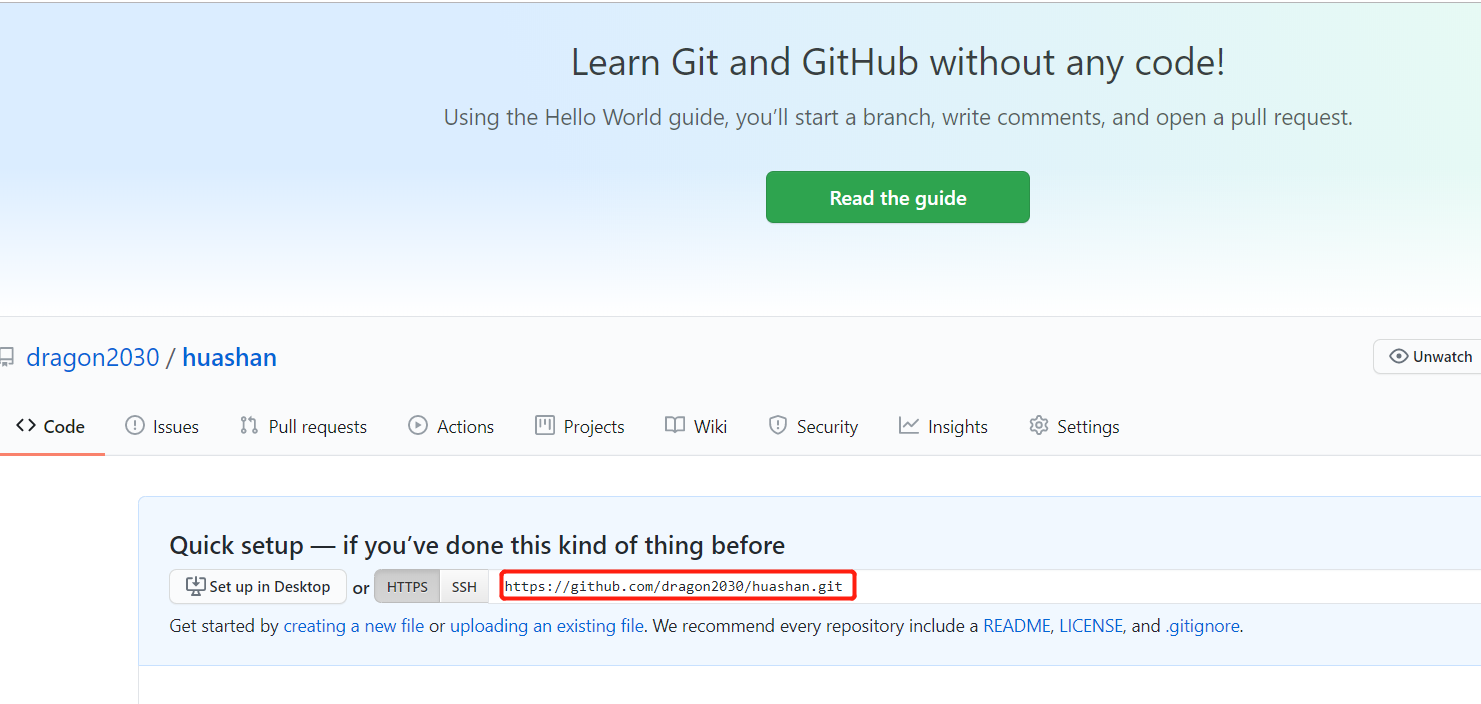
流程图：



### 创建远程库



### 远程库地址



### 为远程库地址设置别名

git remote add [别名] [远程库地址]



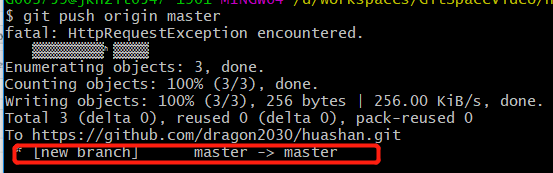
查看设置的远程库推送别名

git remote -v



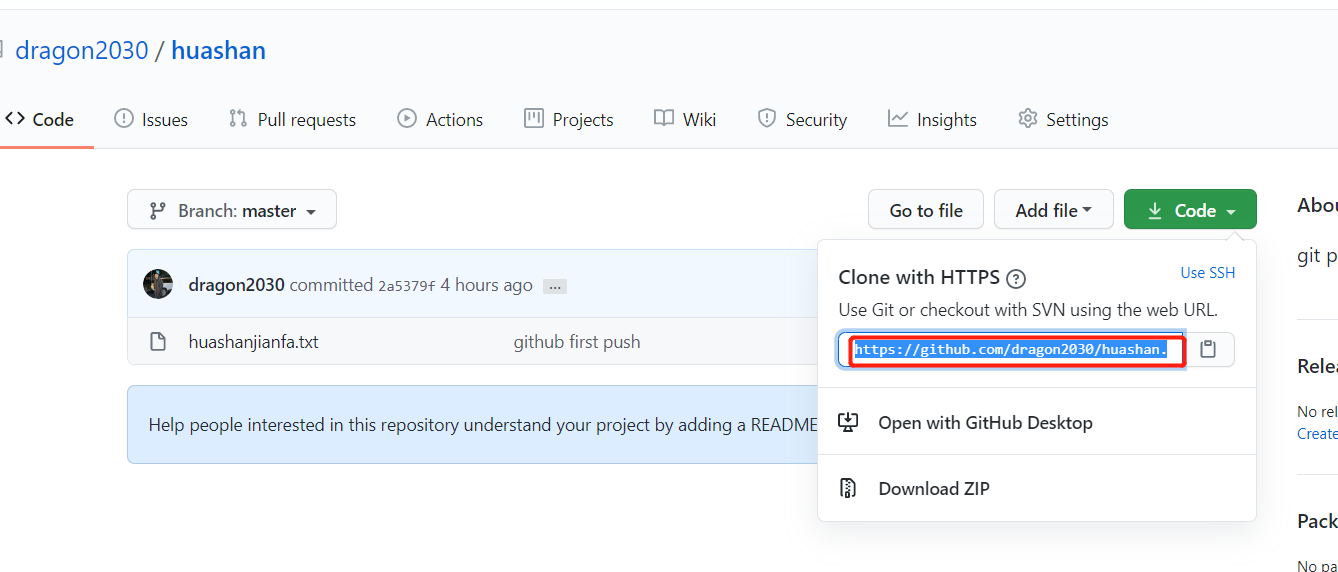
### 本地库推送到远程库

git push [别名] [分支名称]



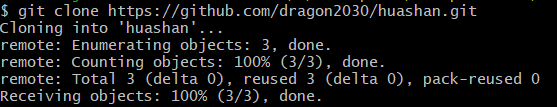
### 克隆操作

获取远程库克隆地址



创建一个新的文件夹，进行克隆操作

命令:git clone [克隆地址]



克隆特点和效果：

1.完整的把远程库下载到本地

2.创建origin远程地址别名

3.初始化本地库

# Git图形化界面操作

# Gitlab服务器环境搭建