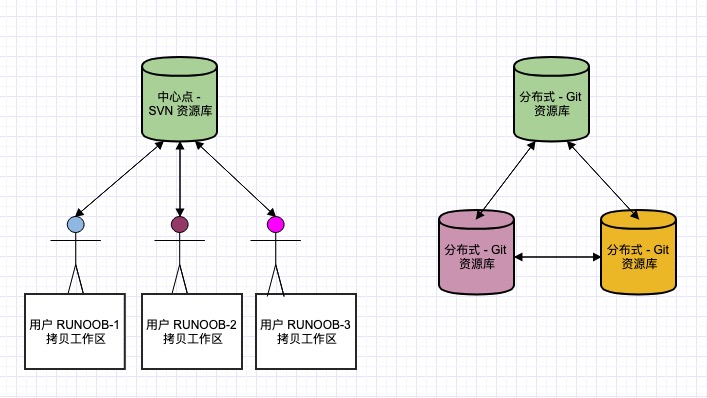
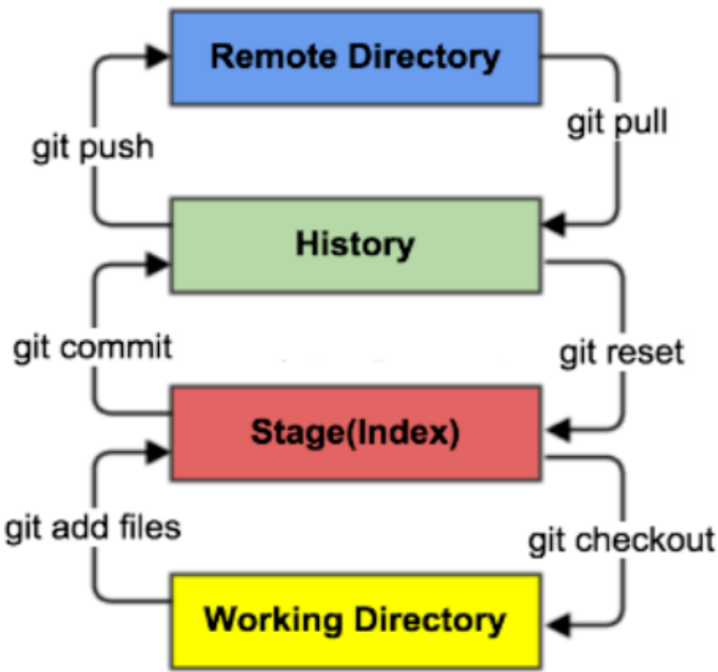
**Git 学习**

1. **分布式版本控制**

**Git是分布式版本控制系统，没有中央服务器，每个人的电脑就是一个完整的版本库。**



1. **三个区域**



远程仓库：托管代码的服务器如github等

本地仓库：安全存放数据（包含所有版本数据）

提交的文件）

暂存区：临时存放（保存即将

提交的文件）

工作区：平时存放代码

**工作流程：**

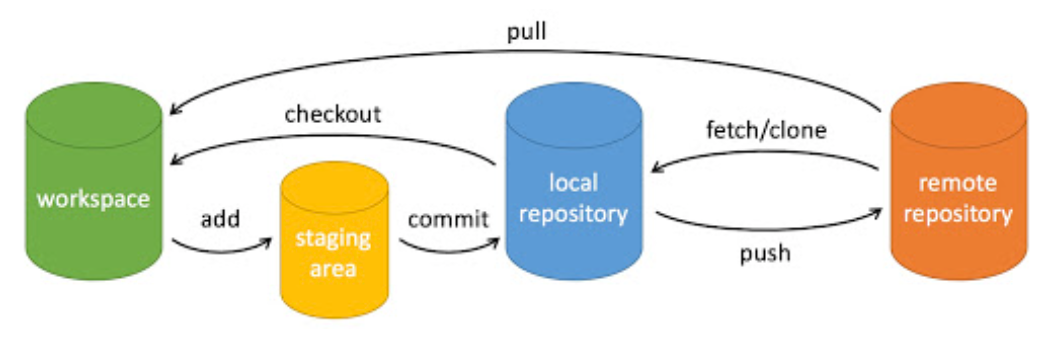
１、在工作目录中添加、修改文件；

２、将需要进行版本管理的文件放入暂存区域；

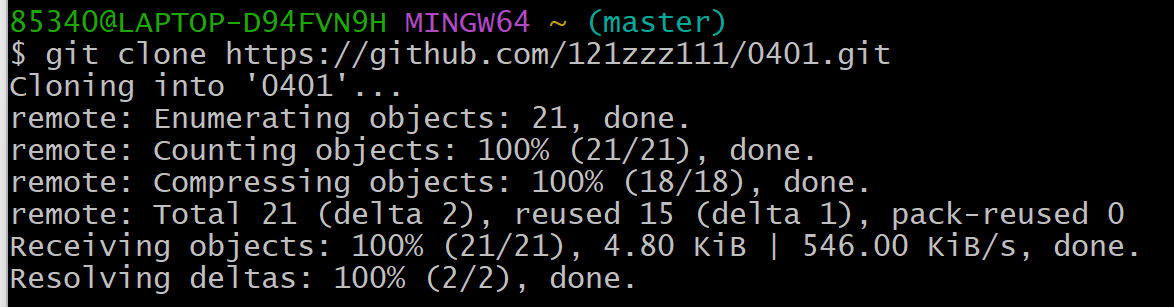
３、将暂存区域的文件提交到git仓库。

git管理的文件有三种状态：已修改（modified）,已暂存（staged）,已提交(committed)

1. **常用指令**



1. 本地仓库搭建
2. 创建新的：git init
3. 克隆：



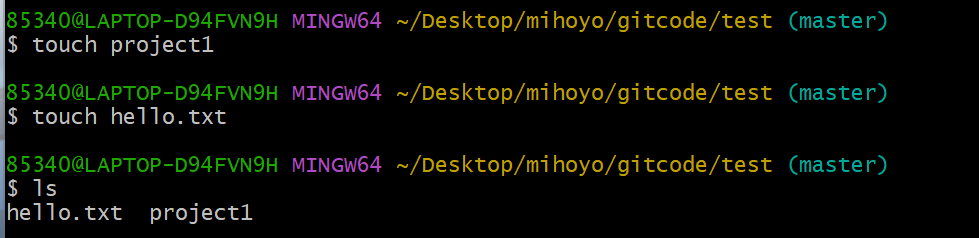
1. 提交与修改



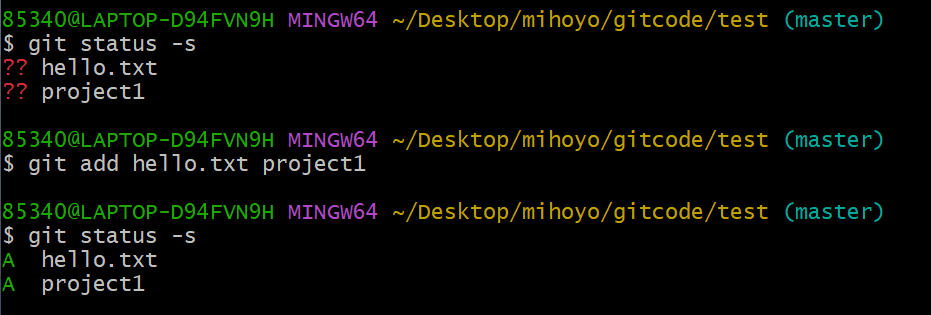
1. git\_add



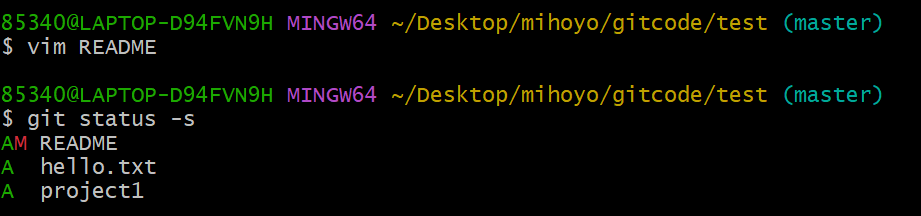
* 新建两个文件



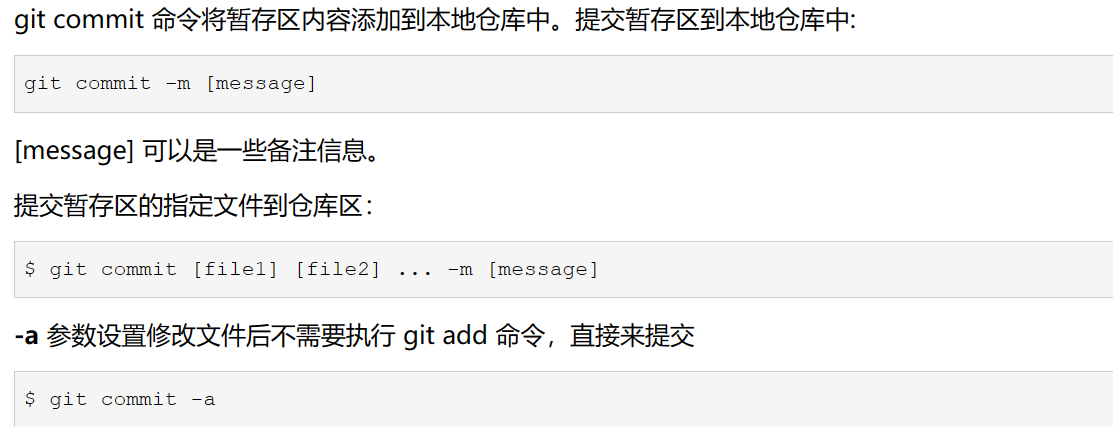
* 查看状态--添加至暂存区--查看状态

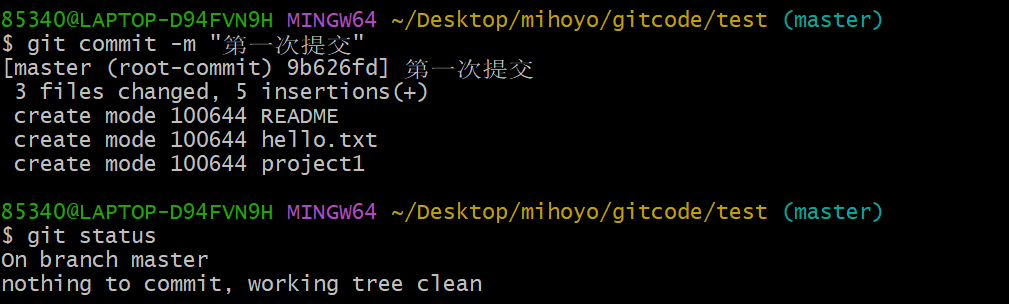


* 使用 vim README进行修改--查看状态



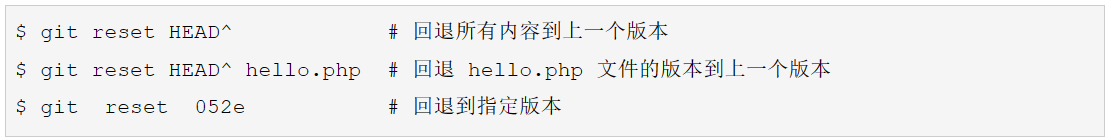
1. git\_commit





1. git\_reset [--soft | --mixed| --hard] [HEAD] ：修改HEAD的位置，即将HEAD指向的位置改变为之前存在的某个版本

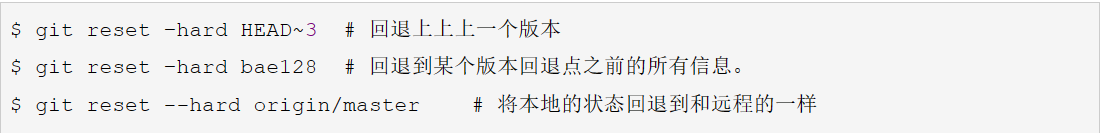
* --mixed 为默认，用于重置暂存区的文件与上一次的commit保持一致，工作区文件内容保持不变。



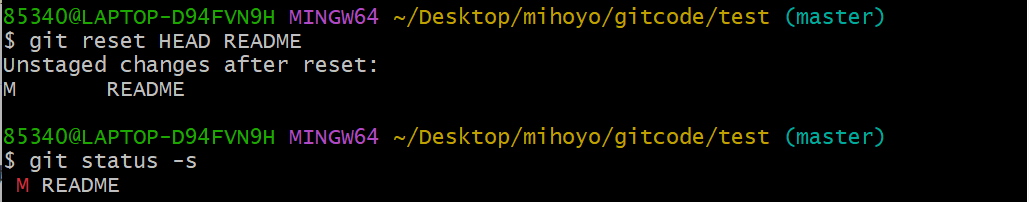
* --soft用于回退到某个版本



* --hard  撤销工作区中所有未提交的修改内容，将暂存区与工作区都回到上一次版本，并删除之前的所有信息提交



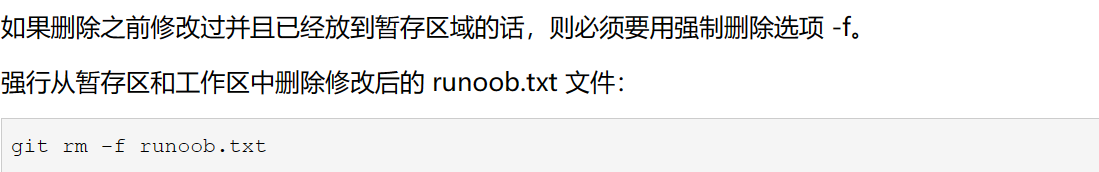
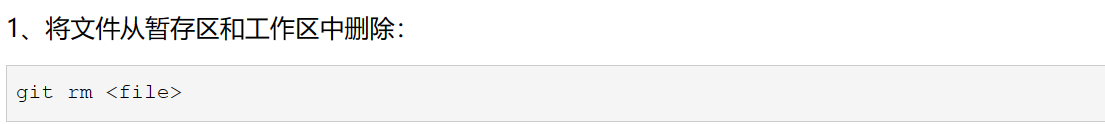
* git reset HEAD 命令用于取消已缓存的内容

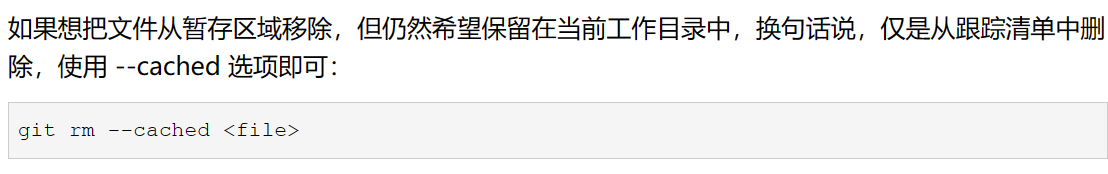


具体操作：

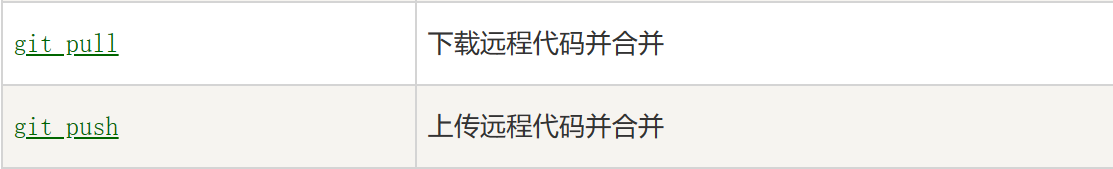
* 查看版本号：git log （也可以通过 git tag打标签）
* 使用git reset --hard 目标版本号命令将版本回退
* 再用git log查看版本信息，此时本地的HEAD已经指向之前的版本
* 使用git push -f提交更改

1. git\_rm



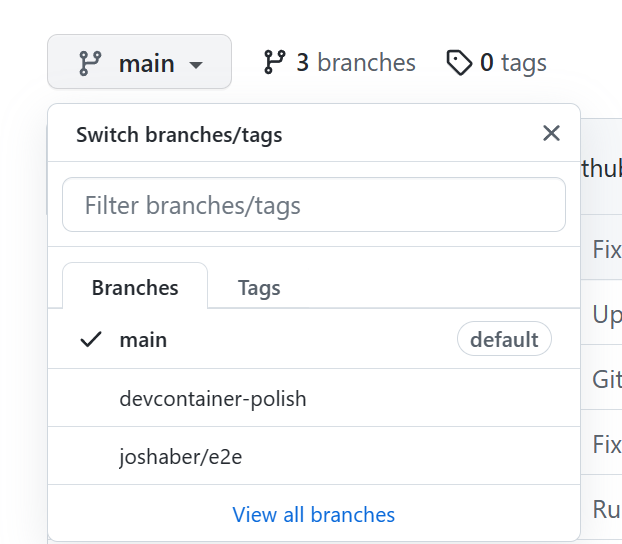


1. 上传至远程仓库



1. 分支

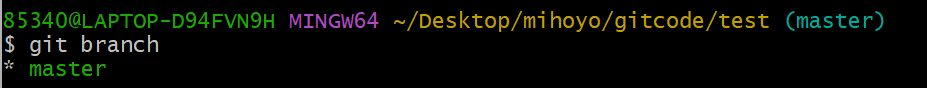
1.一个分支代表一条独立的开发线，在不影响main的情况下同时运行。

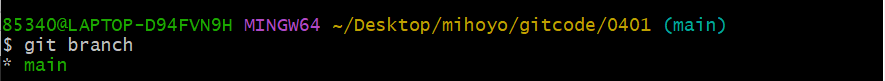


1. 常用命令



（1）查看分支 本地git branch 远程 git branch -r





（init创建后，主分支对应master，但如果我clone下来就对应main）

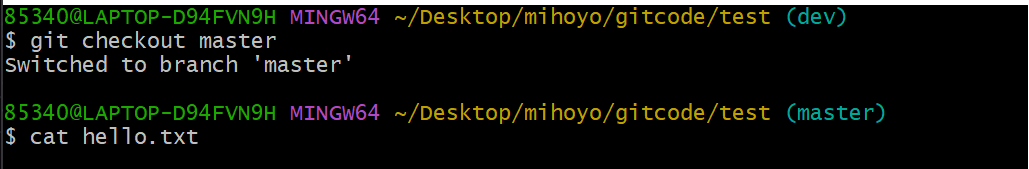
（2）新建分支 停留在当前 git branch [分支名] 切换到新分支 git checkout -b [分支名]



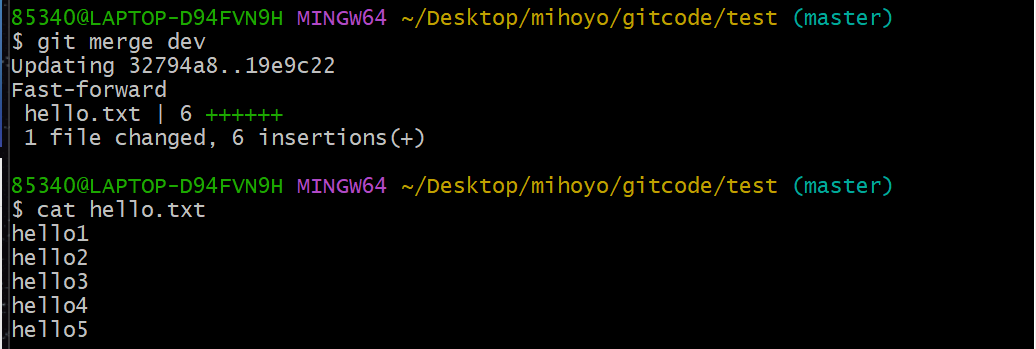
* 切换分支--修改文件--add--commit--查看



1. 合并分支（正常合并） 切换至主分支 hello.txt并未改变

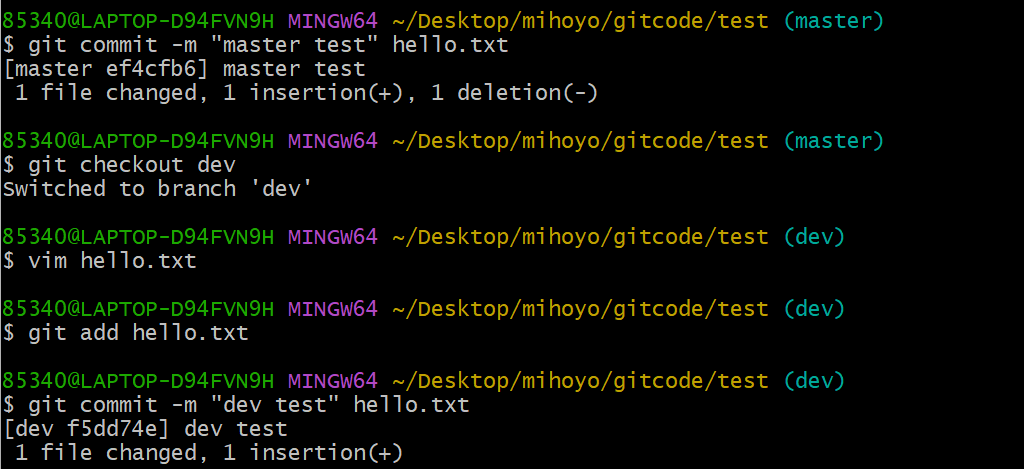


* 将dev分支合并到master，即hello.txt文件被改变



（冲突合并）：合并分支时，两个分支在同一个文件的同一个位置有两套完全不同的修改，Git无法替我们决定使用哪一个，必需人为决定新代码内容。

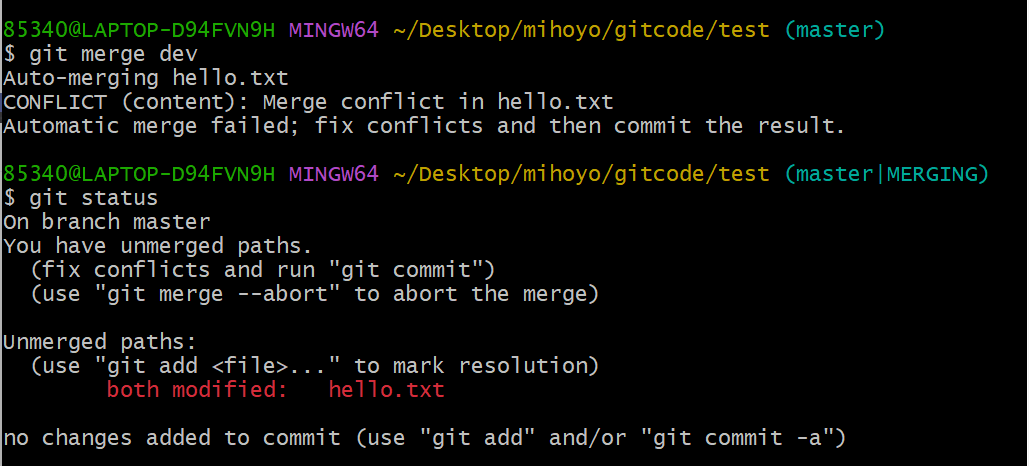
* 分别修改位于master和dev中的hello.txt



修改dev分支的文件

修改master分支的文件

* 合并冲突，自动合并失败



* 查看文件，进行修改



五、github提交与拉取

1.提交



2.查看当前远程库git remote

1. 拉取远程库

* 在github中修改README文件
* 利用git fetch从远程库下载新分支与数据
* 将更新同步至本地 git merge

