

# Git第一讲


清华大学

电子系科协软件部

李宗洹

[lizonghu16@mails.tsinghua.edu.cn](mailto:lizonghu16@mails.tsinghua.edu.cn)

[kevinli606@gmail.com](mailto:kevinli606@gmail.com)



# 主要内容

- Git简介
- 本地仓库操作
- 关联远程仓库

# Git简介

- Git的产生和发展历史  
略过，感兴趣可以自己查
- **Git是当前最先进的分布式版本控制系统**
- 分布式vs集中式  
本地操作vs必须联网  
极大地提升工作效率和安全性

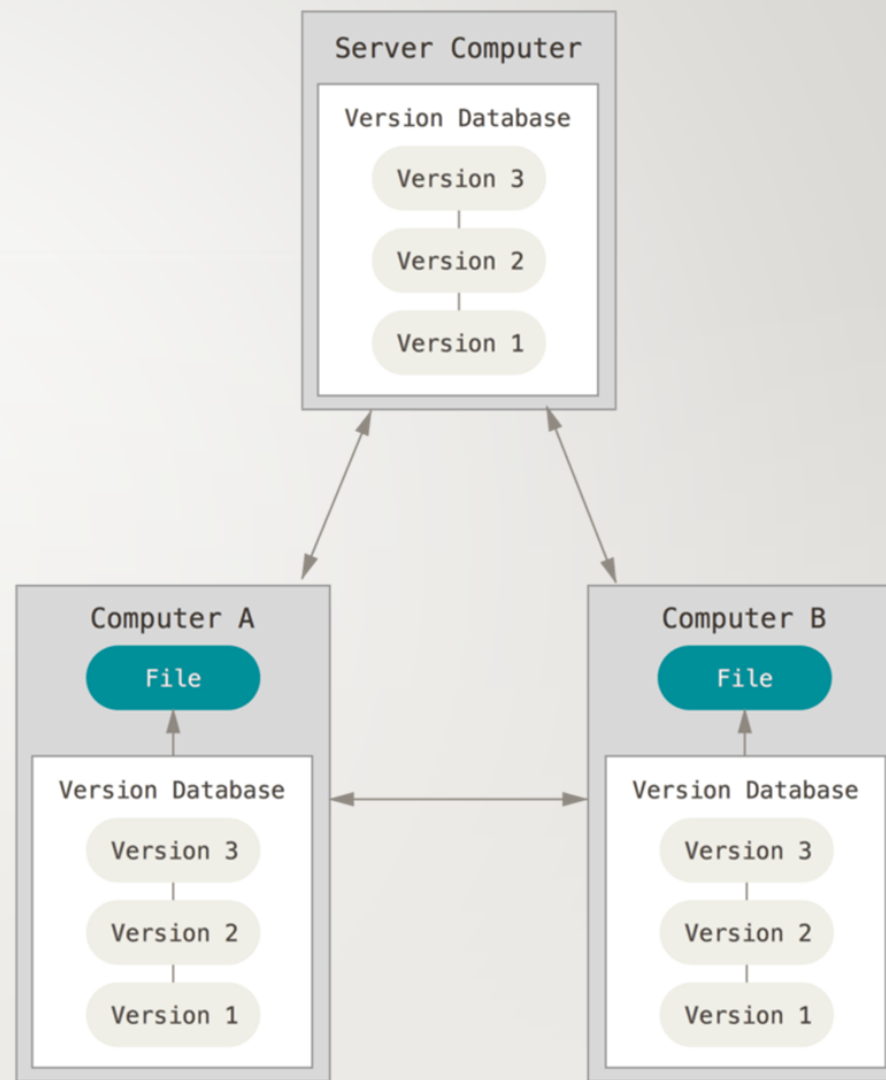
# Git简介

- 何为“版本控制”？
- **版本控制(Revision control)**是一种软件工程技巧，籍以在开发的过程中，确保由不同人所编辑的同一档案都得到更新
- 为何要“版本控制”？
- 大幅提升多人合作时的开发效率！

# 本地仓库操作

- Git在做什么?

Git负责管理文件的修改  
(包括文件的增加和删除)



# 本地仓库操作

- 普通备份：多个版本难以管理
- 施工图
- 施工图改1
- 施工图改2
- 施工图最终版
- 施工图最终版1
- 施工图最终版2
- .....

- 版本控制：版本树清晰明了



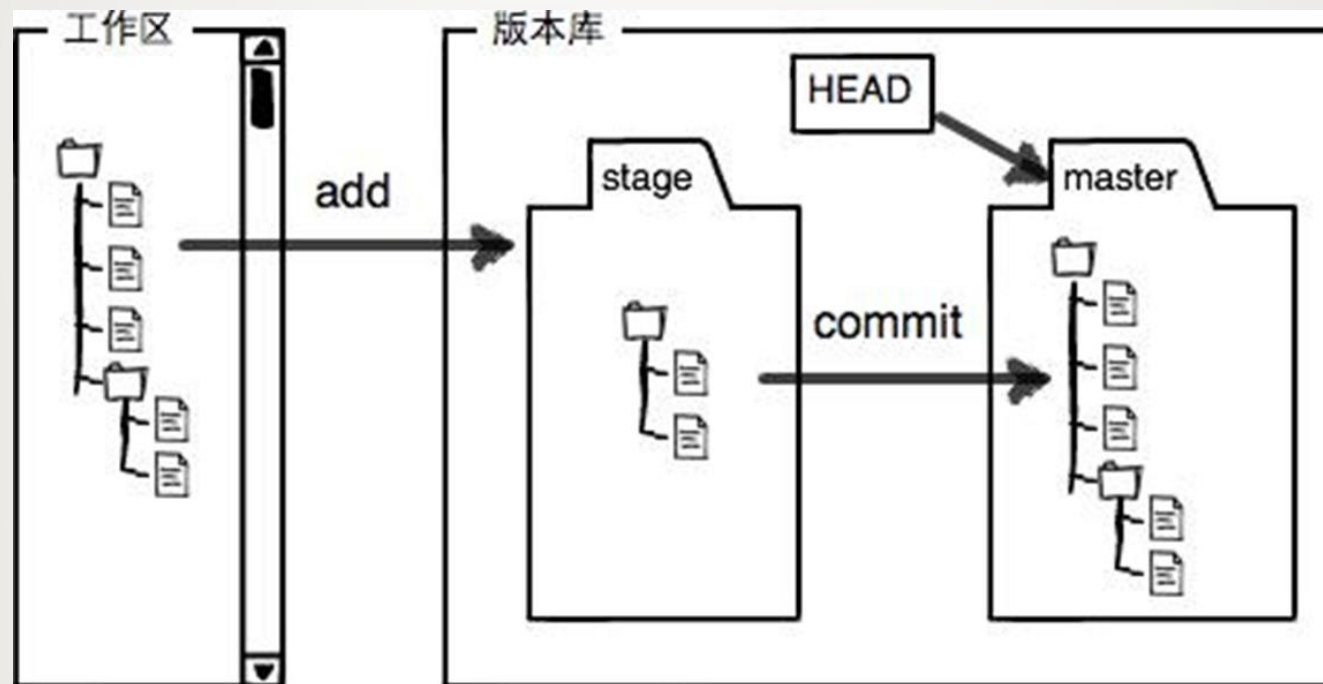


## 本地仓库操作

- 这些版本形成了**版本库**，也叫**仓库(repository)**
- 仓库在本地可以理解成文件夹，只是可以由git管理
- 想让一个文件夹由git管理？非常简单：
- `$ git init`
- 自动创建.git目录（不要修改里面的内容！）

## 本地仓库操作

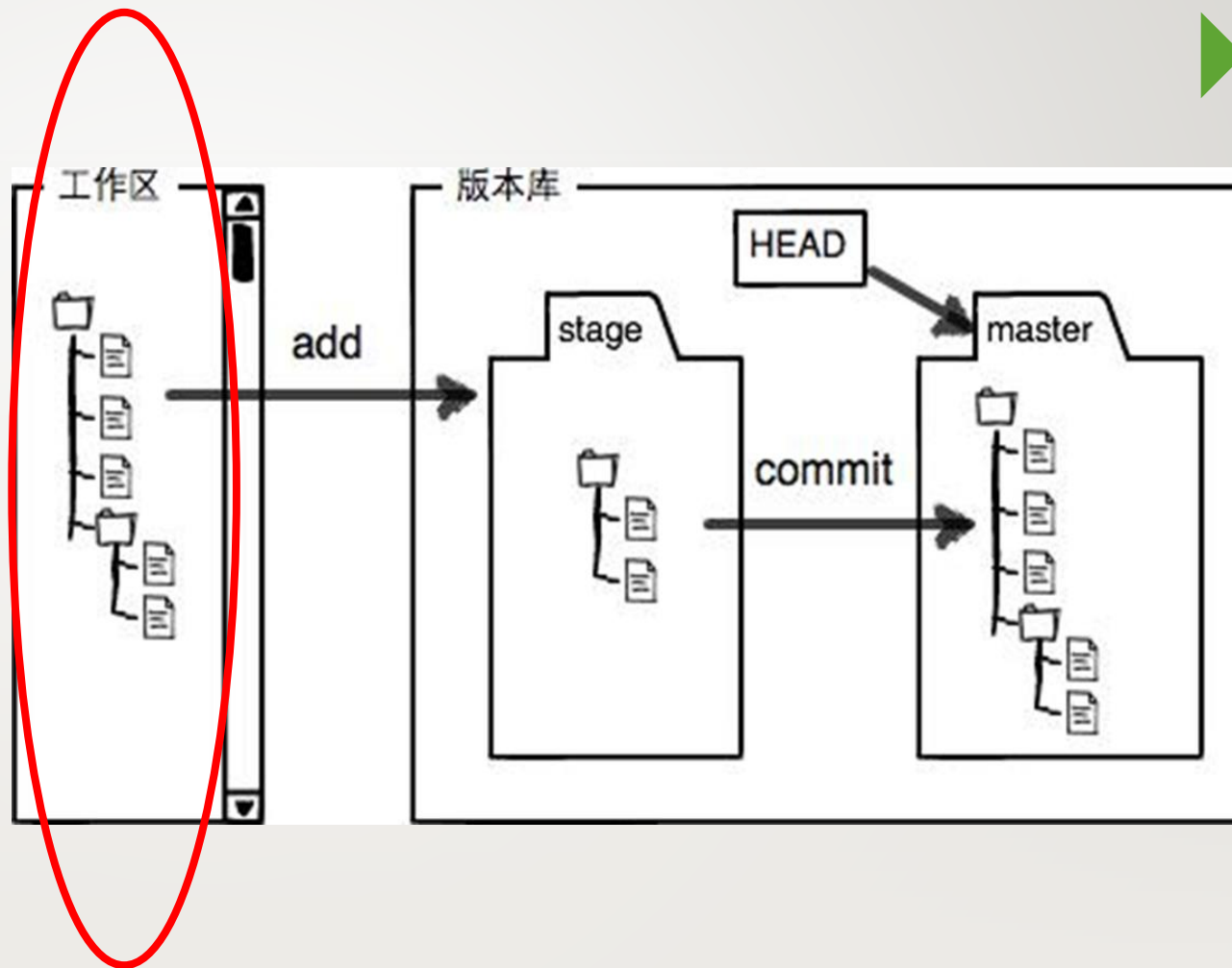
- Git仓库的文件管理结构:





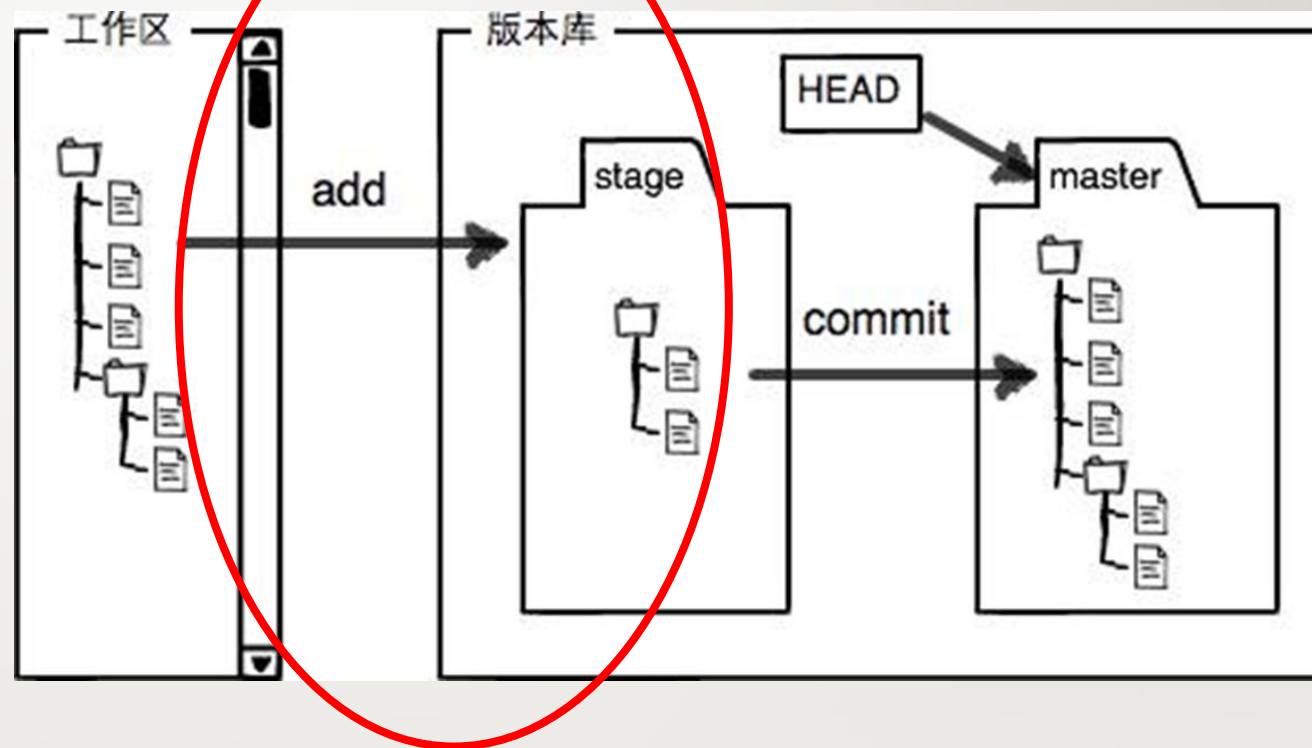
## 本地仓库操作

- 1. 文件修改
- 使用你自己的编辑器，保存即可
- 不建议使用记事本，可用Notepad++代替



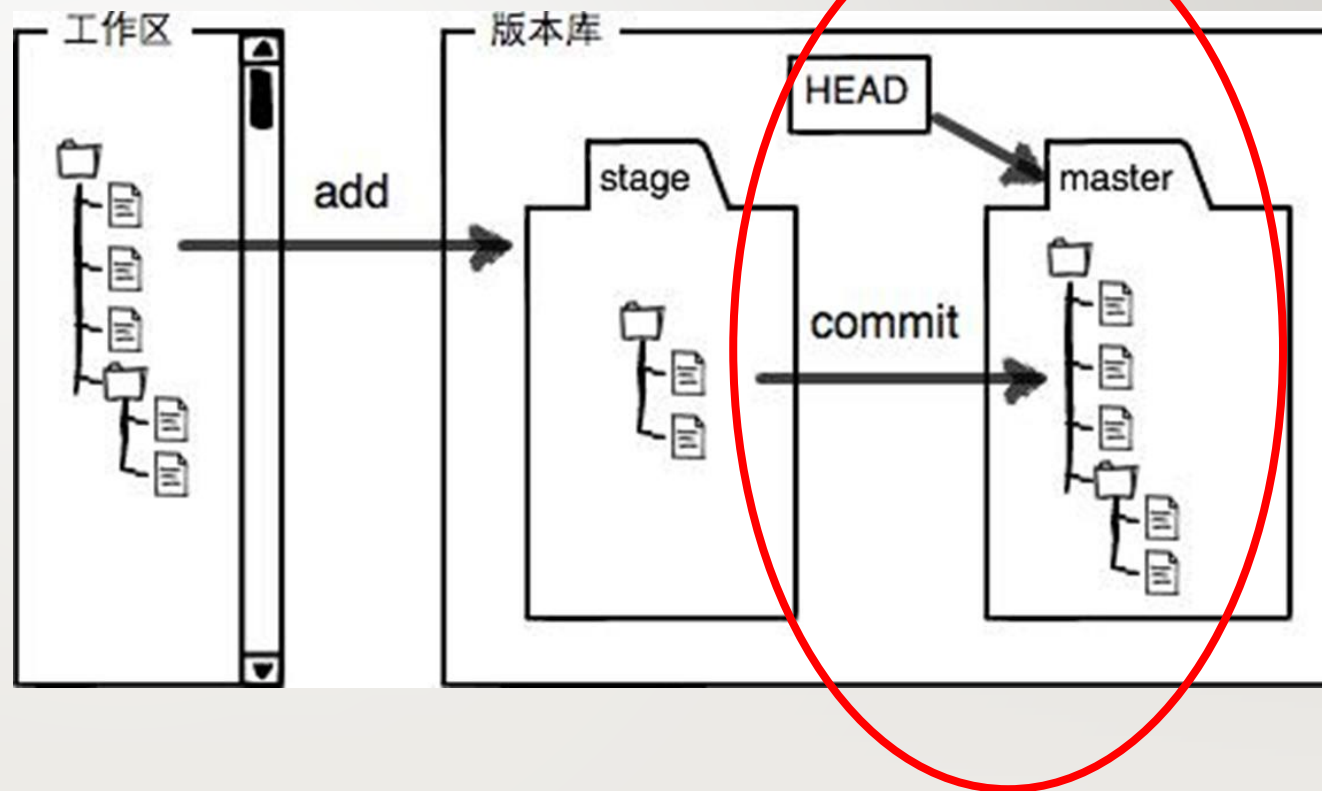
## 本地仓库操作

- 2. 提交到暂存区(stage)
- `$ git add new.txt`
- `$ git add new1.txt new2.txt .....`
- 添加所有修改和未跟踪文件:
- `$ git add .`



## 本地仓库操作

- 3. 提交到版本库(repo)
- `git commit -m "....."`
- 一定要加 -m
- 可对本地所有变更的已跟踪文件进行提交，包括修改和删除，但不包括未跟踪文件
- HEAD指向的就是目前最新的版本

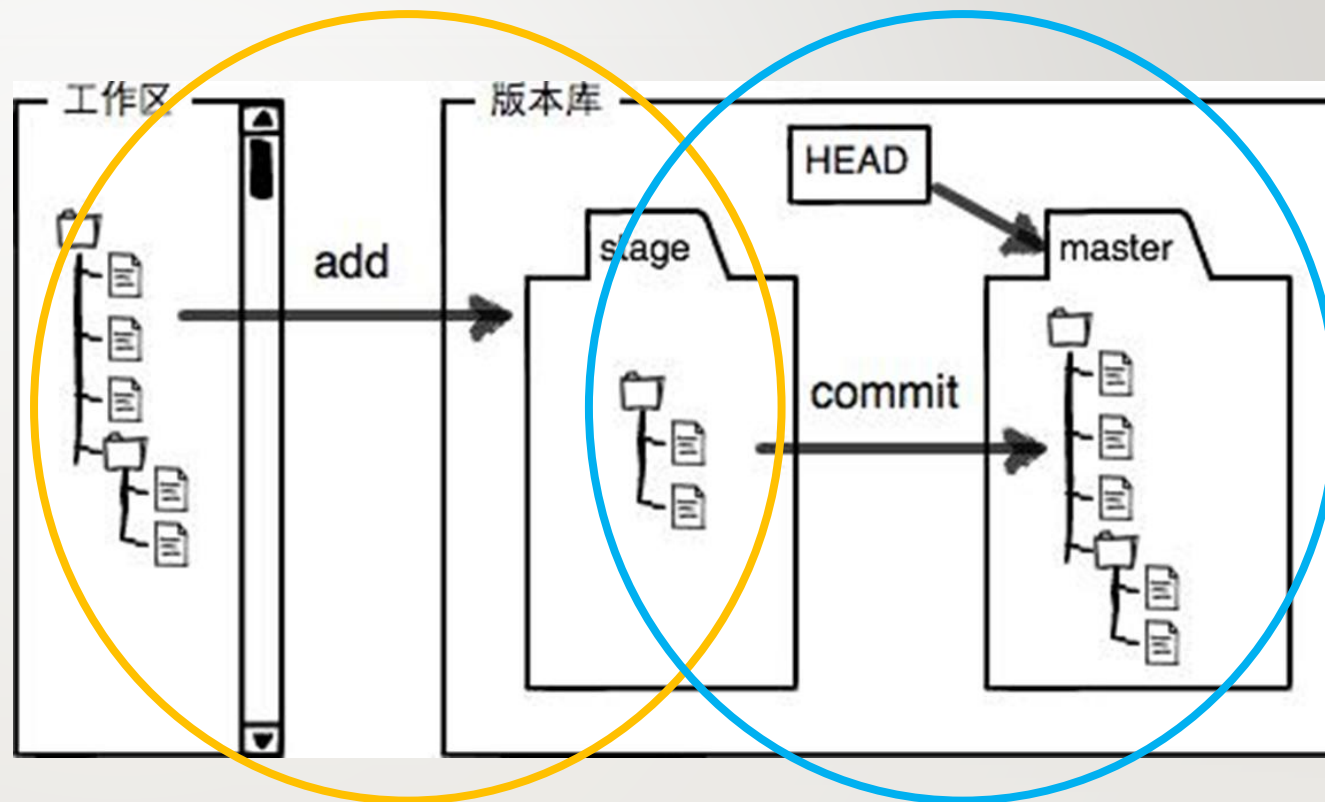


## 本地仓库操作

- 如何查看以前提交的版本?
- `$ git log`
- 查看提交历史, 可看到对应版本的提交说明和版本号
- 每一个版本对应一个独特的版本号(commit id)
- 版本号是一个SHA1计算出的十六进制数, 例如:
- 3628164fb26d48395383f8f31179f24e0882e1e0
- 很复杂, 所以最好清楚写明每一次commit的提交信息, 以便分辨

## 本地仓库操作

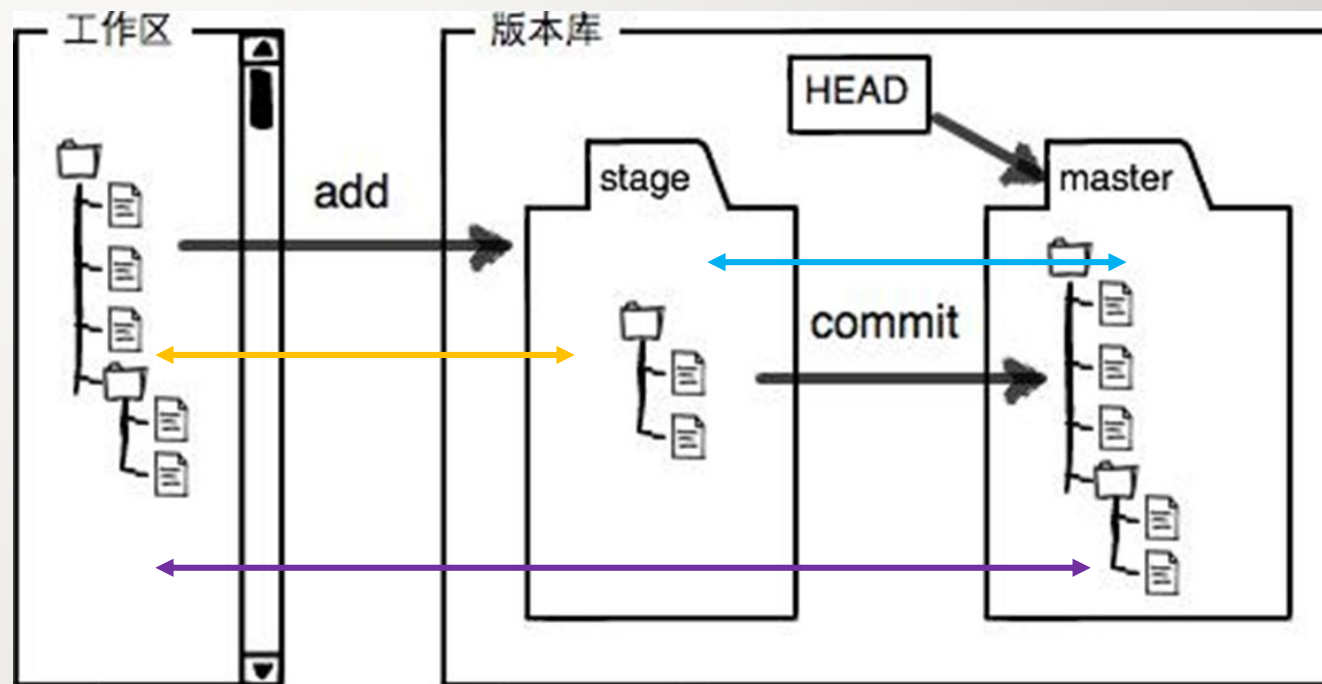
- 如何查看仓库当前的工作状态?
- `$ git status`
- 能看到什么?
- 已修改但未添加到暂存区的文件  
(已修改未add)
- 已添加但未提交到仓库的文件  
(已add未commit)





## 本地仓库操作

- 如何查看文件的修改?
- `$ git diff`
- 看工作区和暂存区文件的区别
- `$ git diff --cached`
- 看暂存区和HEAD的区别
- `$ git diff HEAD`
- 看工作区和HEAD的区别
- 注意: git保存的是文件的修改, 不是文件本身





## 本地仓库操作

- 如何进行版本回退?
- 前面讲的: `$ git log` 查看历史版本, 可以看到版本号
- 退回历史版本:
- `$ git reset [--hard/soft/mixed] 版本号`
- []可以不写, 默认为mixed
- 版本号可以不写全, 但也不能太短, 要能唯一确定

# 本地仓库操作

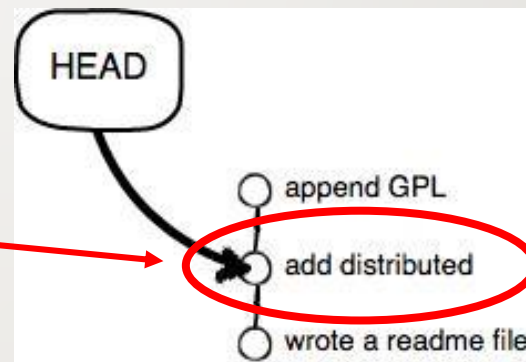
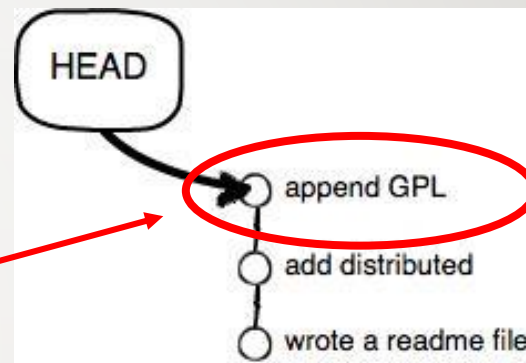
- reset后接不同命令的作用
- 首先都可以让HEAD指向指定的版本
- 默认情况下为mixed，只回退暂存区不回退工作区
- hard重设工作区和暂存区
- soft不改变工作区和暂存区，自指定版本号以来的修改都会成为待commit的修改

## 本地仓库操作

- reset和revert有区别
- reset**退回到**某一次修改，revert**撤销**某一次修改
- reset**移动**HEAD指针，revert**更改**相应的版本
- revert带有记录

## 本地仓库操作

- HEAD指针指向当前的版本
- `$ git reset` 也可以转到更新的未来版本，只要有确定的版本号
- 如果当前在某个历史版本，此时用 `$ git log` 命令会怎样？
- 只能看到更早的版本！



## 本地仓库操作

- 想撤销文件的修改?
- 第一种, 还没有用 `$ git add` 添加到暂存区:
- `$ git checkout -- filename`
- 作用: 恢复到上一次commit或add的版本
- 第二种, 已经添加到暂存区, 还没有commit:
- `$ git reset HEAD filename`
- 作用: 把暂存区的修改回退到工作区

\*对比一下前面回退版本的  
`$ git reset`  
本质上是一样的

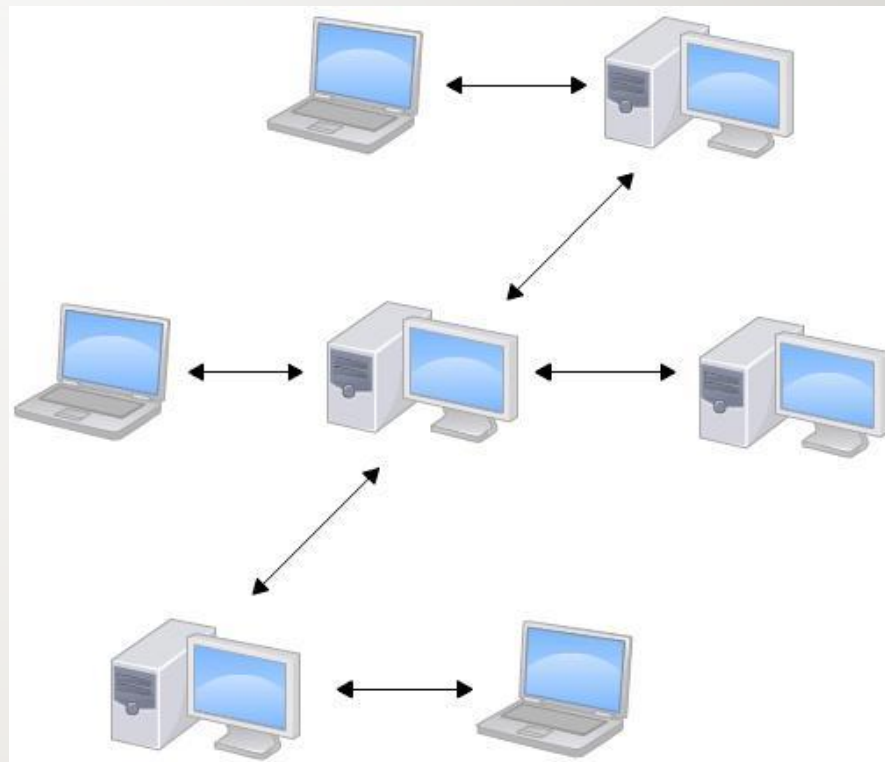
## 本地仓库操作

- 第三种：已经add且已经commit
- 如果尚未push，可以用 `$ git reset` 版本回退
- 如果已经push，git就无能为力了
- 小结：
- 大部分情况下，有后悔药可以吃
- 只要没有推送到远程仓库，都可以返回修改



## 关联远程仓库

- 还记得**分布式**版本控制系统吗？
- 需要有一台电脑“充当”中央服务器，否则版本库交换不方便
- GitHub就扮演了这个角色
- 我们的**远程仓库**，一般都位于GitHub上



## 关联远程仓库

- 首先，注册GitHub账号
- 新建仓库，然后与本地仓库关联：
- `$ git remote add [name] [url]`
- 例如：
- `$ git remote add origin git@github.com:EESAST/test.git`
- 作用：将本地仓库与远程仓库建立**关联**，以后通过pull和push操作就可以实现本地和远程的版本库同步！

## 关联远程仓库

- 如果先有了远程仓库，想复制到本地并关联，怎么办？
- 更简单！
- `$ git clone [name] [url]`
- 例如：
- `$ git clone git@github.com:EESAST/test.git`
- Git会自动创建版本库并关联

## 关联远程仓库

- 想查看本地仓库和远程的关联情况:
- `$ git remote`
- 或者 `$ git remote -v` 查看详细信息

```
kevinli@DESKTOP-SRRT18C MINGW64 ~/Desktop/EESAST-2018-2019/eesast_recruitment (ma
ster)
$ git remote
origin

kevinli@DESKTOP-SRRT18C MINGW64 ~/Desktop/EESAST-2018-2019/eesast_recruitment (ma
ster)
$ git remote -v
origin  git@github.com:AlbertHuyb/eesast_recruitment.git (fetch)
origin  git@github.com:AlbertHuyb/eesast_recruitment.git (push)
```

## 关联远程仓库

- 基本的pull和push
- `$ git pull`
- **前提：**本地仓库和远程仓库都已存在并关联
- **作用：**将远程仓库的内容拉取到本地
- pull过程伴随merge操作，可能冲突，下节再讲

## 关联远程仓库

- `$ git push`
- 前提：本地仓库和远程仓库都已存在并关联
- 作用：将本地仓库的内容推送到远程
- 后面可以加命令：`$ git push origin master` 推送相应分支的修改
- 推送所有修改，使用 `$ git push` 即可
- 和pull一样可能会冲突，下节再讲



## 下节预告

- 远程仓库操作

pull或者push时可能遇到问题，怎么办？手动解决vs暴力删库.....

- 分支管理

为什么需要分支？如何进行分支的创建、合并？ git merge, git rebase.....

- 标签管理


标签是什么？如何设置、管理标签？ .....

- GitHub使用

多人开发的正确方式是什么？什么是pull request？ .....

## 参考资料

- [Git教程 - 廖雪峰的官方网站]  
<https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000>
- [官方git教程] <https://try.github.io/>
- [去年的ppt, 内容差不多]
- <https://github.com/eesast/Training/tree/master/git>



Git第一讲结束