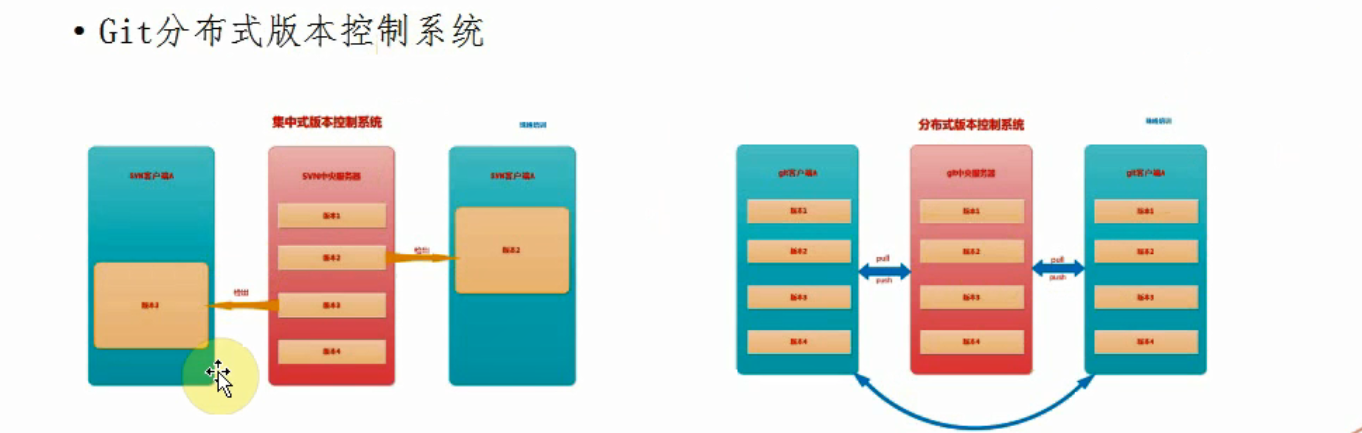
git: 分布式版本控制系统

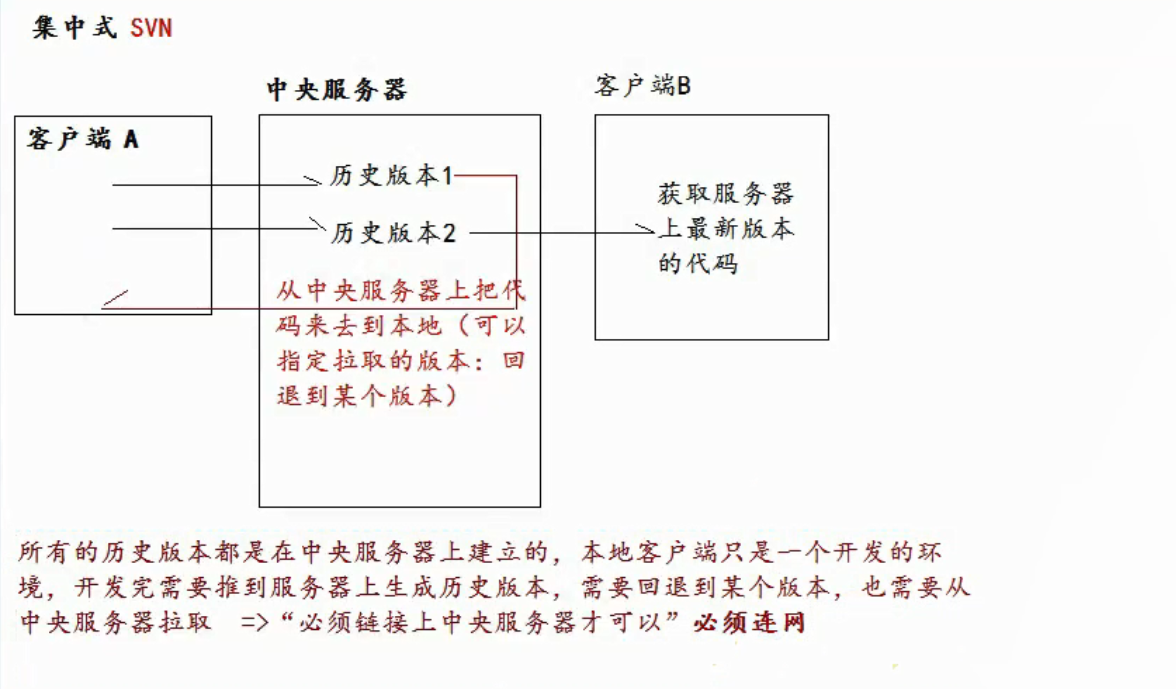
## 1 版本控制系统

开发中我们把每一次的修改都有效的进行记录(记录成一个版本),后期如果需要回退到原有的版本



实例图:集中式对比分布式

集中式:



分布式:GIT

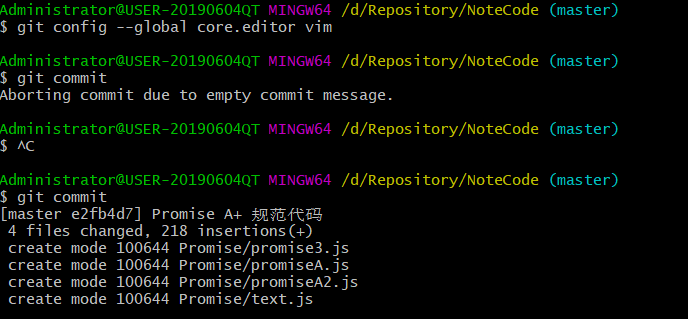
切换到本地所在仓库目录

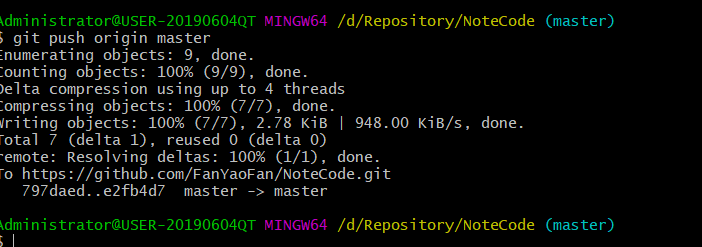
git add . 或者git add 文件名

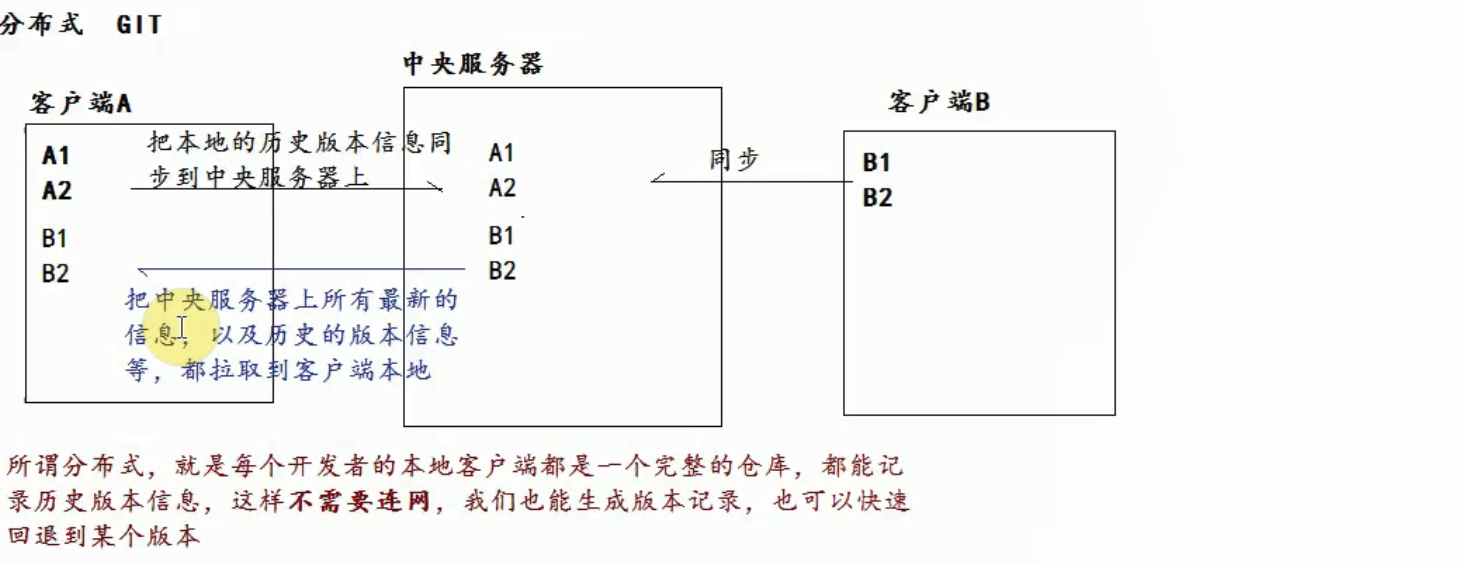
git commit 这里可能出现一个默认编辑器,使用 git config global core.eritor vim 来设置

使用 git commit 命令的时候要写 备注信息,不然会报错 Aborting commit due to empty commit message. //因为提交消息为空而终止

git push origin master









## 2 git的常规流程

每一个git出口都有三个区域

工作区:写代码

暂存区:临时存放每一次修改的代码,但是并没有生成历史版本

历史区:存放所有历史版本的地方(提交到历史区就会生产法历史版本)

# 2.1 Git 基础 - 获取 Git 仓库

## 获取 Git 仓库

有两种取得 Git 项目仓库的方法。

第一种是在现有项目或目录下导入所有文件到 Git 中； 已经有的导入到git中

第二种是从一个服务器克隆一个现有的 Git 仓库。 克隆一个from 服务器

### 在现有目录中初始化仓库

1 git init

.git很重要,没有它不能叫做一个仓库

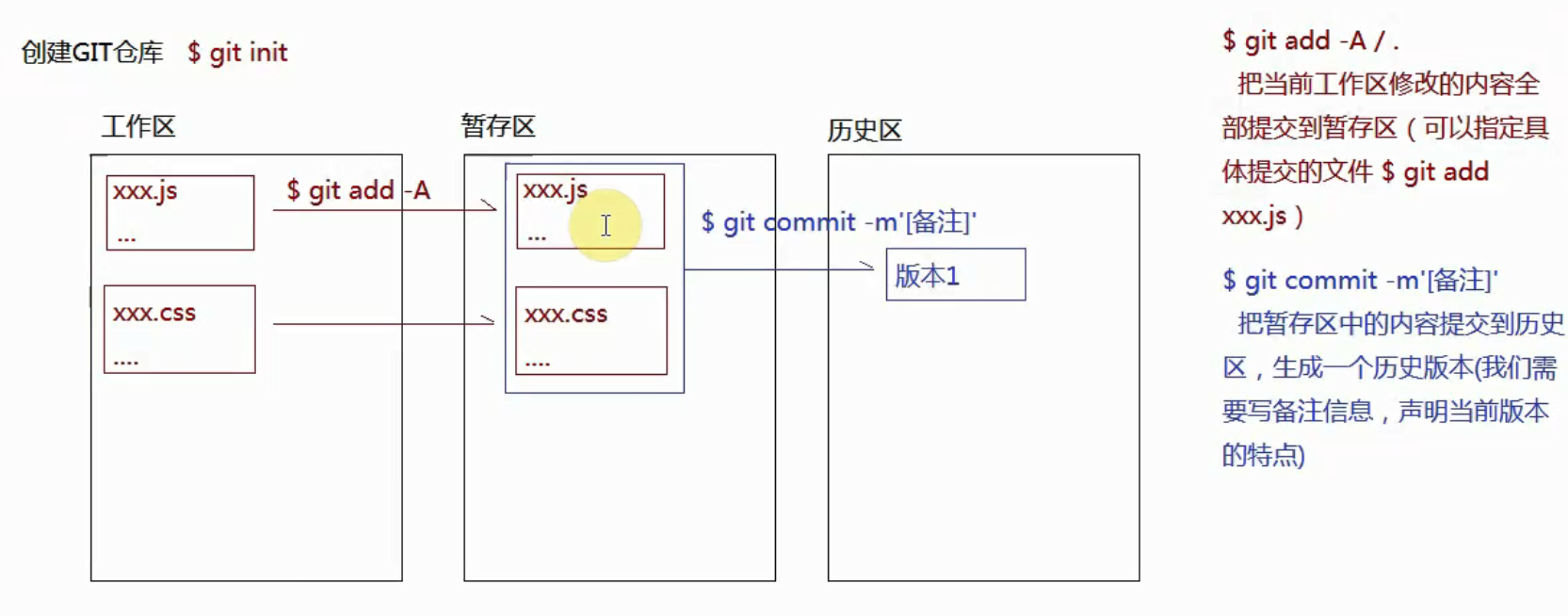
该命令将创建一个名为 .git 的子目录，这个子目录含有你初始化的 Git 仓库中所有的必须文件，这些文件是 Git 仓库的骨干。 但是，在这个时候，我们仅仅是做了一个初始化的操作，你的项目里的文件还没有被跟踪。

$ git add \*.c

$ git add LICENSE

$ git commit -m 'initial project version'

你是在一个已经存在文件的文件夹（而不是空文件夹）中初始化 Git 仓库来进行版本控制的话，你应该开始跟踪这些文件并提交。 你可通过 git add 命令来实现对指定文件的跟踪，然后执行 git commit 提交：



每一次区域间信息的提交都不会删除原有区域信息;下一次提交,只是把各个区域对比出来不一样的内容提交

如果你想获得一份已经存在了的 Git 仓库的拷贝，比如说，你想为某个开源项目贡献自己的一份力，这时就要用到 git clone 命令

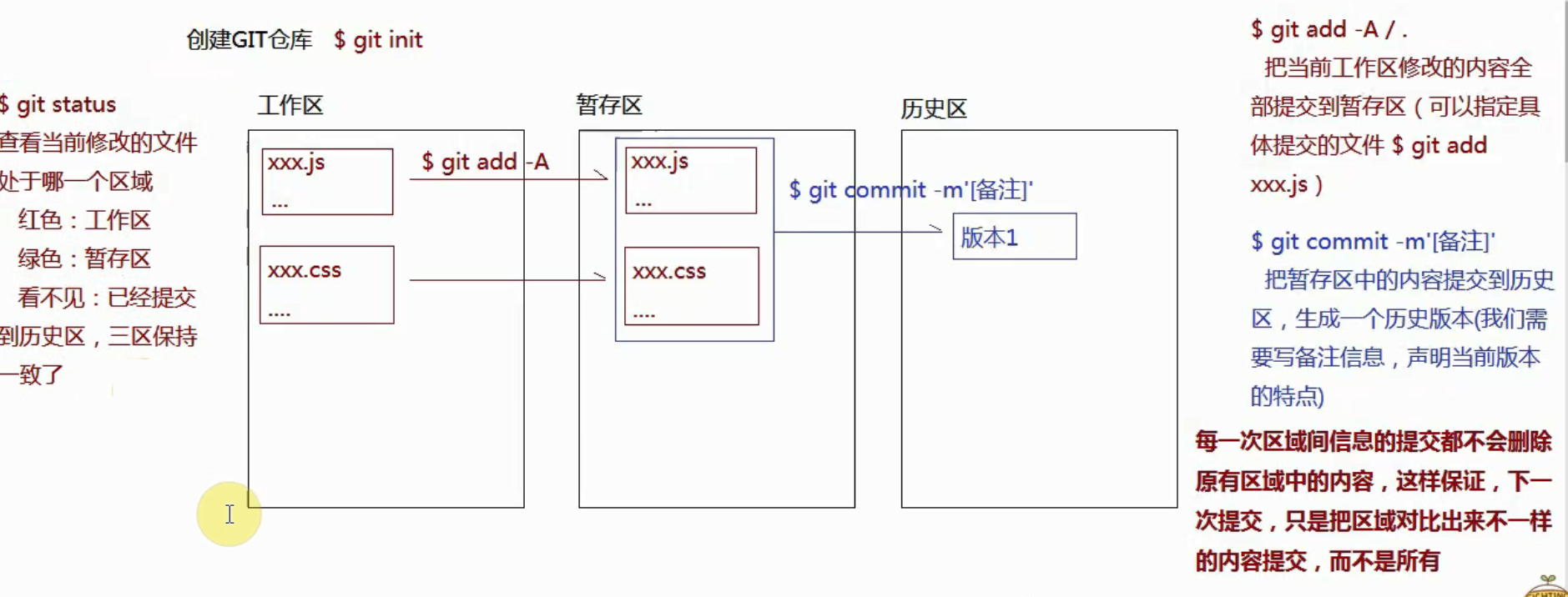
克隆仓库的命令格式是 git clone [url] 。 比如，要克隆 Git 的可链接库 libgit2，可以用下面的命令：

$ git clone <https://github.com/libgit2/libgit2>

这会在当前目录下创建一个名为 “libgit2” 的目录，并在这个目录下初始化一个 .git 文件夹，从远程仓库拉取下所有数据放入 .git 文件夹，然后从中读取最新版本的文件的拷贝。 如果你进入到这个新建的 libgit2 文件夹，你会发现所有的项目文件已经在里面了，准备就绪等待后续的开发和使用。 如果你想在克隆远程仓库的时候，自定义本地仓库的名字，你可以使用如下命令：

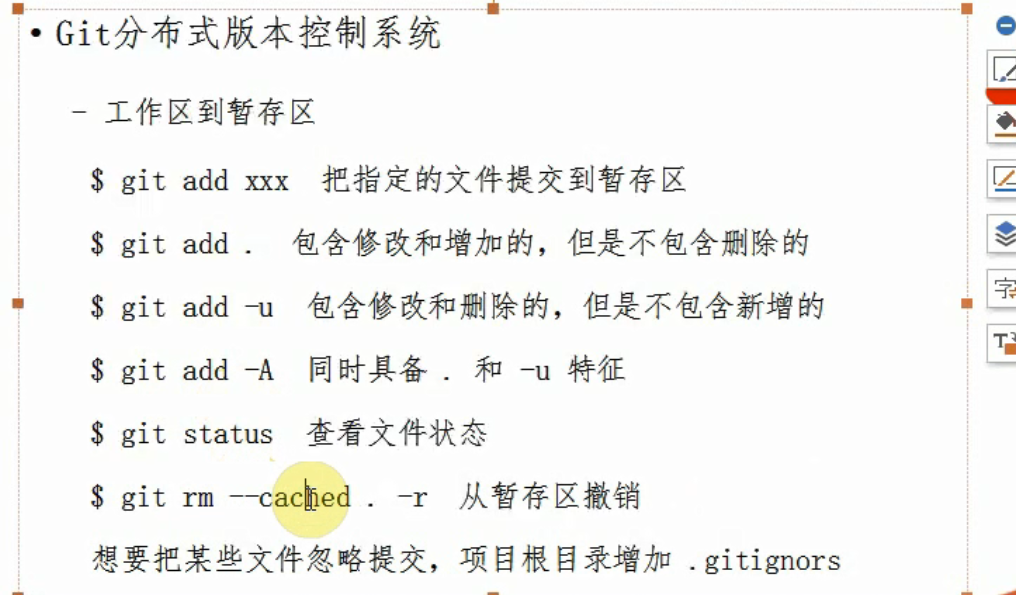
$ git clone https://github.com/libgit2/libgit2 mylibgit

这将执行与上一个命令相同的操作，不过在本地创建的仓库名字变为 mylibgit。



git add file 工作区 =>暂存区

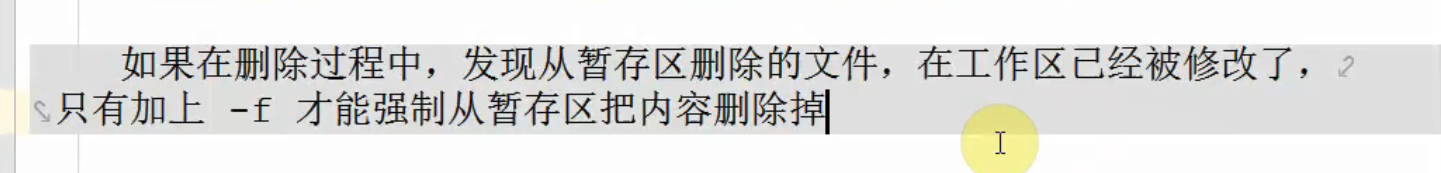
git rm –cached file 可以撤回来了 从暂存区撤回到工作区

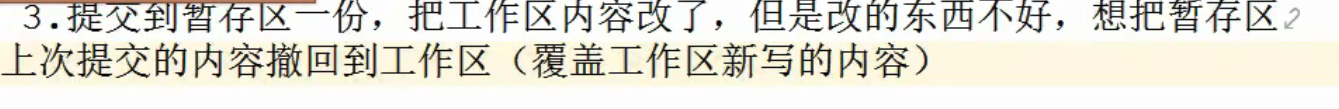


提交1.txt , git add 1.txt (status就变绿) 到了暂存区

然后 修改工作区的1.txt, status为 modified

将暂存区的1.txt 回撤下来,暂存区里面的1.txt是什么样就是什么样,不会和modified的合并的 (要加 -f )





git checkout 从暂存区复制一份到工作区,把工作区的覆盖了(暂存区的还在)