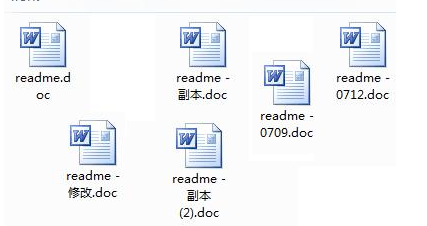
# 版本控制系统

版本控制系统可以对同一个文件的多个历史版本进行管理。

## 为什么需要版本控制系统

如果没有版本控制，对于同一个文件的多个历史版本，就只能创建多个不同名字的副本文件。

图1



而用了版本控制系统，同一个文件只保留一个文件，多个历史版本可以找回。

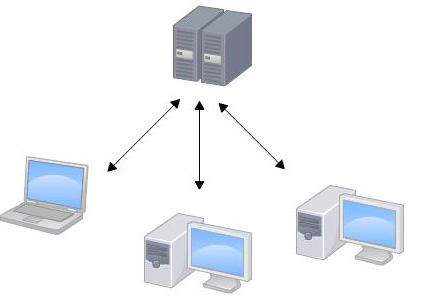
图2



## 集中式版本控制系统

版本库（多个历史版本的文件）放在中央服务器，本机要先从中央服务器获取该文件的最新版本，然后修改，再把自己的修改版本推送给中央服务器。

图1



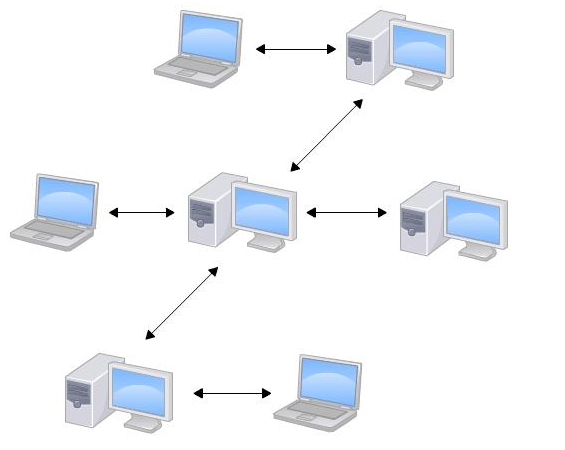
### 1.2.1 缺点

1. 必须联网才能工作，带宽慢时上传文件特别慢（最大的缺点）
2. 因为版本库只在中央服务器有一份，如果中央服务器挂机，所有本机不能工作

## 分布式版本控制系统

每个本机从中央服务器克隆一份版本库，本机版本库的修改推送到中央服务器版本库。

图1



### 1.3.1 优点

1. 不需要每次都联网，可以将本机的修改先提交到本机版本库，有网络的时候再一起推送到中央版本库。
2. 每个本机都有一个本机版本库，如果某个本机挂机了，可以从另一个本机拷贝。

# Git操作

## 2.1 git init 初始化

git init将普通的文件目录变成git仓库。

.git目录中管理了版本库有关的信息。

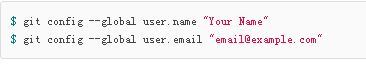
图1



## 2.2 git config 配置参数

--global表示本机的所有git仓库都使用这个配置。

图1



## 2.3 git add添加到暂存区

git add将文件从工作区添加到暂存区。

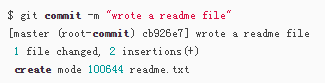
图1



## 2.4 git commit添加到本机版本库当前分支

git commit将文件从暂存区添加到本机版本库的当前分支上。

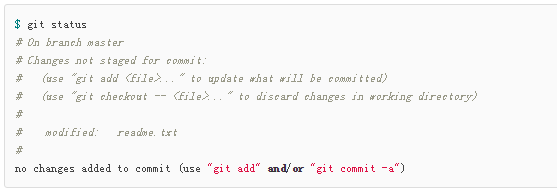
图1



## 2.5 git status查看文件当前状态

Git status可以查看文件当前是在工作区，还是在暂存区，还是已提交（在本机版本库）。

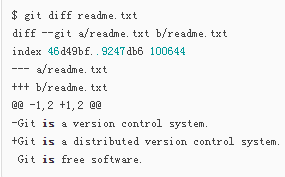
图1



## 2.6 git diff查看文件修改的内容

git diff查看文件修改的内容。

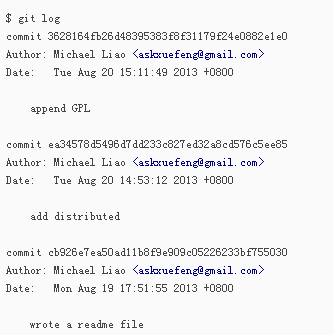
图1



## 2.7 git log查看提交日志

git log查看提交日志。

图1



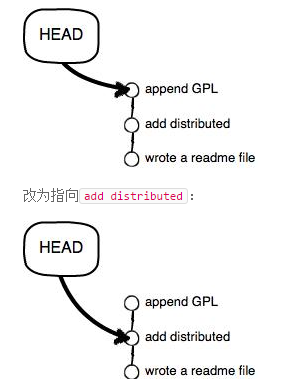
## 2.8 git reset --hard HEAD^ 回退到上一个版本

HEAD指的是当前版本，HEAD^是上一个版本，HEAD^^是上上个版本。

图1



图2



## 2.9 git reset --hard 版本号 切换到任意版本

版本号只要前面几位就可以，只要前面几位可以确认唯一的版本号。

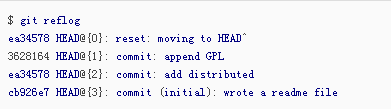
图1



## 2.10 git relog 查看命令日志

git relog查看命令日志。

图1



## 2.11 git checkout -- file撤销工作区的修改

git checkout -- readme.txt可以撤销工作区的修改。

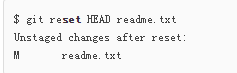
图1



## 2.12 git reset HEAD file撤销暂存区的修改并返回工作区

git reset HEAD readme.txt撤销暂存区的修改，把这个修改返回到工作区中。

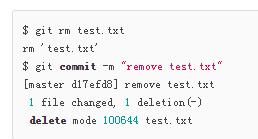
图1



## 2.13 git rm将删除信息添加到暂存区

从本机文件目录删掉，只是删除了工作区的文件；git rm可以将这个删除的操作添加到暂存区。

图1



# 工作区，暂存区，版本库的关系

工作区就是本机文件目录。

版本库是.git目录。

版本库中包含了暂存区和各个分支。工作区的内容先add到暂存区，然后commit会将暂存区的内容添加到当前分支上。

图1

