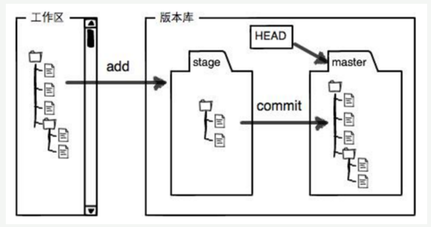
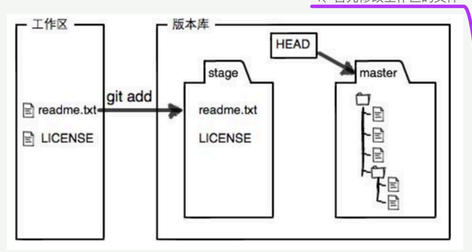
- Git
  - 什么是版本控制系统?
    - 没有版本控制系统之前。。比如写小说、论文的痛苦，需要手动的去记录每次的改动。
    - 能记录每次文件的改动（包括每次改动的时间、作者、内容等），还可以协同其他人协作编辑的一个工具（系统）。
    - 集中式版本控制系统（代表：CVS、SVN） 版本库放在中央服务器，必须联网才能工作（记录每次的改动）
    - 分布式版本控制系统（代表Git） 每个人的电脑就是一个完整的版本库，没有网络也可以工作（记录每次的改动）
  - Git的历史
    - 开发Linux系统的那个人用C语言写的Git，最开始Git只能运行在Linux和Unix系统上，后来移植到了Windows上，现在Git可以在Linux、Unix、Mac、Windows等几大平台上运行。
  - Git安装
    - Linux、Mac这里忽略。。日后去公司需要安装的话百度一下。
    - Windows安装也比较麻烦，不过还好有一个Git面向Windows的打包好的安装程序。（Git-2.10.2-64-bit.exe）
    - 默认安装好后，在开始菜单中找到‘Git’-》‘Git Bash’即可通过Linux命令来操作电脑，从而使用Git。
  - Git命令
    - 在终端下执行：git - -version查看git的版本号
    - 配置全局的用户和邮箱(让别人知道这个是谁提交的，配置一次即可)
      - git config --global user.name ‘你的用户名（英文）’
      - git config --global user.email ‘你的邮箱’
    - git init 创建版本库，在要使用Git管理的目录下执行这个命令，会把当前目录变为Git可以管理的仓库
    - git status 查看Git仓库的状态（是否有文件被改动、是否有未提交等）
    - git add <文件>
      - 添加【工作区的修改】到暂存区
      - git add . 添加工作区所有修改到暂存区
    - git commit -m ‘本次提交的描述内容’ 将暂存区文件的修改提交到某个分支中（默认是master主分支）
    - git diff <文件> 查看和版本库中的不同（在未提交之前使用，方便对比自这次做了什么修改，以免提交错误）
    - git log
      - 查看从最近到最远的日志信息列表
      - git log --pretty=oneline 省略日志其他信息，在一行显示
      - git reflog 获取操作的版本号（一般用于回退到某个版本使用）
    - git reset --hard
      - 版本回退（把已经提交过的修改，回退到指定的某一个版本）
      - git reset - -hard HEAD^ 回退到上个版本
      - git reset - -hard HEAD^^ 回退到上上个版本
      - git reset - -hard~100 回退到上100个版本
      - git reset - -hard ‘提交ID(版本号)’ 回退到指定版本
    - git checkout -- <文件>（注意-- 两边有空格，其中 -- 不能省略， 如果没有 -- 就变成切换分支的命令了）
      - 撤销修改分两种情况（也可以通过该命令找回工作区已删除的文件，因为删除也是修改的操作）
        - 1、没有add到暂存区的时候直接使用 通过git checkout - - <文件>丢弃工作区的修改
        - 2、已经add到暂存区的时候，但还没有提交
          - 先通过git reset HEAD <文件> 命令把暂存区的修改撤销掉，重新放回工作区
          - 再通过上一条命令丢弃工作区的修改 git checkout - - <文件> 【注意】如果文件中没有内容修改，那么可能出现撤销失败的情况
    - 分支相关
      - git branch 查看分支列表以及当前所在分支
      - git branch <分支名字> 创建新分支
      - git checkout <分支名字> 切换分支
      - git checkout -b <分支名字> 创建并且切换分支（以上两部操作合二为一）
      - git branch -d <分支名字> 删除分支
        - 如果在删除一个没有完全合并的分支的时候，可能会提示删除不成功，你确定确实要删除的话，使用：`git branch -D cart` 进行删除即可(即使用D)
      - git merge <分支名字>
        - 合并某分支到当前分支(分清楚到底是谁合并谁，合并的内容在哪个分支)
      - 总结：以后开发，我们做某个功能的时候尽可能的在分支上去做，然后在分支上进行版本管理，最后做完后，合并到主分支，并且【推送到服务器中】
      - 如果通过git merge <分支名字>合并两个分支的时候，因为其中一个分支中(比如主分支master)已经有一个文件了，比如user.c文件。那么合并的时候就会因为两个文件冲突而合并失败。这个时候假设你在子分支中，那么你无法删除这个子分支，因为你的合并操作还没执行完；同时你也不能切换到master分支，同样是因为你的合并操作没有执行完。那么可以做的是：先在子分支中删除冲突的文件或者修改文件名 删除:rm user.c 修改文件名mv user.c xxx.c 这个操作其实就是一个修改(删除也是一种修改）然后再提交：git add xxx.c 再git commit -m ‘修改了文件名’ 之后再合并。如果因为冲突而知道了并不需要该文件，那么解决办法就是先删除，再提交删除操作。最后可以自由选择合并或者不合并。冲突一旦被解决，那么切换分支或者进行合并操作都是被允许的啦。如果别人的更新和自己的更新都有可取之处，那么就应该进行【手动合并】保留两人各自的可取之处。
  - 关联远程仓库GitHub
    - 生成ssh key ssh-keygen -t rsa -C ‘youremail@example.com’
    - 找到生成的密钥
      - window： C: Users/Adminstroter/.ssh 打开id_rsa.pub
      - linux&mac open ~/.ssh 打开id_rsa.pub
    - 登录github账号
      - 点击头像下的Settings 然后在左侧找到SSH and GPGKey，点击进入
      - 新建立一个ssh key，输入title，把生成的密钥拷贝进来，保存
    - 创建远程仓库 点头像左侧的小+号，New repository
    - 关联远程仓库(关联成功一次后就不需要在关联了) git remote add origin git@github.com:yourname/gitTest.git
    - 推送到远程仓库 git push -u origin master
    - 以后每次在本地操作后，都循环去执行以上的git相关操作（add 、commit等），最后push到远程仓库
    - 可以通过git clone + 远程仓库地址 把代码下载(拷贝)下来并且生成git环境（将来去公司后，一般来说公司已经有存在的项目仓库了，来公司后只需要git clone 下代码就可以了）
    - 更新代码 git pull 把远程仓库的代码更新下来（多人合作的时候经常使用，把别人提交上去的代码先pull下来）【注意】这种更新是互补性更新，即将别人commit的修改部分更新到我的项目中，相同的部分没有任何影响，更不会整个项目覆盖(也就是说我自己更新的内容不受到影响，不会被覆盖)。所以，每次提交代码之前，需要先进行git pull操作，以保证提交的时候不会发生冲突。
    - 总结：
      - 1、以后去公司开发，养成好习惯，每天在早上上班之后先要git pull（更新服务器上最新的代码），然后在进行开发。
      - 2、在开发过程中，尽量避免两个人做相同的模块，如果做相同模块的话，注意代码不要乱写，每个人只写在自己的功能那里，这样以后互相更新代码的时候不会造成麻烦。
    - 解除关联 git remote remove origin
  - Git所管理的目录结构(我们可以叫Git仓库)
    - 工作区（Working Directory） 电脑上的目录
    - 版本库（Reponsitory） 工作区中有一个隐藏的.git目录，这个不属于工作区，这个是【版本库】（注意：这个隐藏的.git目录我们不要手动去改动它），版本库包含了【暂存区】和【其他分支】两部分。
      - 暂存区（Stage） git add 操作后就会把【工作区的修改】添加到暂存区 git commit 操作后就会把暂存区的文件提交到版本库的当前分支上
      - 主分支（Master Branch） Git会为我们自动创建一个默认主分支master，以及一个指向master主分支的指针HEAD
      - 其他分支。。。

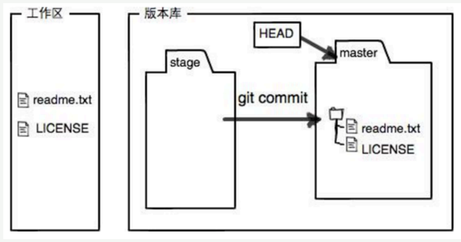


Git管理结构图

  - Git内部操作的流程（面试重点）
    - 1、首先修改工作区的文件



    - 2、通过git add <文件> 把工作区文件的修改添加到暂存区



    - 3、通过git commit -m ‘注释’ 就可以一次性的把暂存区的文件所有修改提交到master分支上