Git

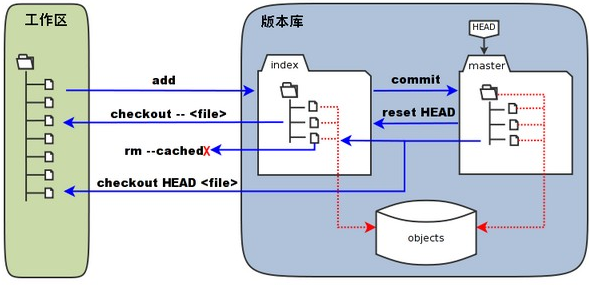
1. Git是一个开源的分布式版本控制系统
2. Git的工作原理
   1. 给他的工作流

* 一般工作流程如下：
* 克隆 Git 资源作为工作目录。
* 在克隆的资源上添加或修改文件。
* 如果其他人修改了，你可以更新资源。
* 在提交前查看修改。
* 提交修改。
* 在修改完成后，如果发现错误，可以撤回提交并再次修改并提交。

# 2.2工作区、暂存区和版本库

* **工作区：**就是你在电脑里能看到的目录。
* **暂存区：**英文叫stage, 或index。一般存放在"git目录"下的index文件（.git/index）中，所以我们把暂存区有时也叫作索引（index）。
* **版本库：**工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

"HEAD" 实际是指向 master 分支的一个"游标"。所以图示的命令中出现 HEAD 的地方可以用 master 来替换。



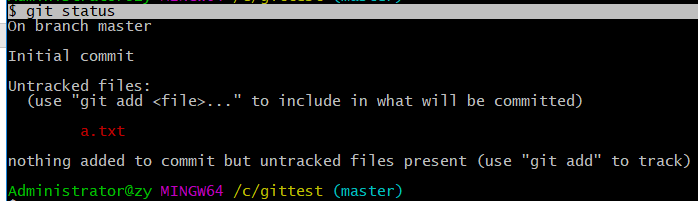
2.3基本命令操作

在一个文件夹下，创建git仓库 ，ls –a该目录下会多一个.git的目录

git init

查看项目状态$ git status 红色a.txt表示还未加入暂存区

git status –s 简略显示,左M修改了放入了暂存区，右M修改了未放入暂存区，A新增文件未放入暂存区

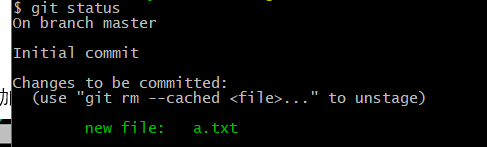


echo “hello”>a.txt

工作区加入指定文件暂存区 git add a.txt

工作区全部加入暂存区 git add .

查看项目状态 git status 绿色表示已加入暂存区



暂存区加入仓库 git commit –m“注释信息“ –m后写的注释

git commit –a –m”注释信息” 把所有修改或删除了的都放进暂存区，并commit，未被跟踪的新建文件不受影响。

git log 查看commit的提交日志d

git diff文件对比

git diff 工作区和暂存区的不同，对比的是工作区和暂存区都有的文件的不同，还未加入暂存区的文件不对比。

git diff –cached 暂存区和暂存区上次版本的不同：

git diff HEAD **显示工作目录(已track但未add文件)和暂存区(已add但未commit文件)与仓库的所有不相同**

**git diff branch1 branch2 --stat   两个分支上最后 commit 的内容的差别（--stat表示不详细显示）**

**git reset命令**

git reset --soft HEAD~1 仓库回退到上个版本，即head指向master的上个节点，工作区和暂存区不变

git reset HEAD~1 默认—mix 仓库和暂存区回退到上个版本，工作区不变

git reset --hard HEAD~1 仓库回退到上个版本，工作区和暂存区清空，仓库回退到的上个版本代替工作区和暂存区，工作区新创建未索引的文件不变。

git reset 版本号 或者git reset head~num 回退几个版本。

git checkout filename 把缓存区的文件更新到工作区，起到撤回的作用。

git tag tagName 给仓库历史中的提交打上标签

git rm 删除

git rm a.txt 删除工作区和暂存区的该文件，但是当工作区修改了，不论加入暂存区与否，使用该命令都报错，即要求工作区和仓库文件一致。

git commit –m’注释’ 提交到仓库，也删除了仓库的文件

git rm –f a.txt 删除工作区和暂存区文件

git rm –cached a.txt 只删除暂存区文件，工作区文件不变。

git mv a.txt c.txt 改名文件并提交到暂存区，该命令相当于以下三个命名

mv a.txt c.txt 改名

git rm test.txt 删除工作区和暂存区源文件

git add mydir 加入改名后文件

分支管理

git branch 查看本地分支

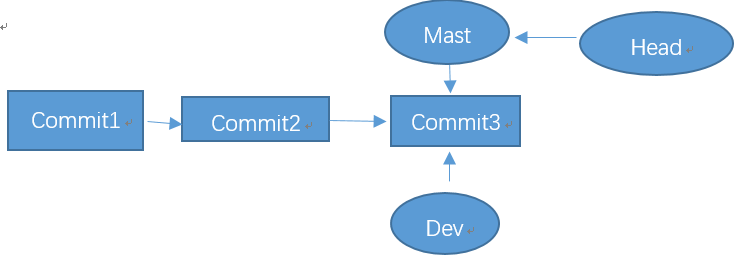
git branch -a 查看本地和远程分支

git branch dev 创建dev分支

git branch –d dev 删除分支

git branch –m dev newdev 重命名分支

git checkout dev 切换到dev分支，比如当前head指向master，会把dev指向master的commitid，head指向dev， 把仓库的文件替换缓存区和工作区，但是工作区和缓存区commit之前的更改不会改变，如果checkout分支的时候，替换的过程有文件冲突或者有未commit的文件就会显示错误不能切换分支。



git merge dev 把dev分支合并到当前master分支上。

远程仓库，github上创建仓库，方法百度，简单。

git clone <https://github.com/zhouyanger/test1.git> 为克隆的存储库中的每个分支创建远程跟踪分支(使用git branch -r可见)，并从克隆检出的存储库作为当前活动分支的初始分支。

在git clone的时候，所有本地分支默认与远程主机的同名分支，建立追踪关系，也就是说，本地的master分支自动”追踪”origin/master分支。

git branch --set-upstream master origin/next 本地master分支和远程origin/next分支建立追踪关系

git remote 查看远程分支，clone之后有一个git默认命名的origin 的远程库-v可以查看远程仓库id

git init 初始化仓库

git remote add origin <https://github.com/zhouyanger/test1.git> 添加一个新的远程仓库,可以指定一个简单的名字,以便将来引用运行

git fetch origin 更新远程所有跟踪分支，

git fetch origin master 更新远程master跟踪分支

git merge origin/master 把远端master分支合并到本地该分支

git pull origin master 远程的master分支拉回合并到本地当前分支

git pull origin master:master 远程的master分支拉回合并到本地master分支

git pull 如果本地当前分支和远程分支建立了追踪关系，可以直接pull

git push origin 本地分支:远端分支 把commit的push到远端仓库

git push origin master 把当前本地分支push到远端master分支

git push 如果本地分支和远端建立了追踪关系，可以省略。

当切换分支的时候，如果有未提交的变更记录，此时系统会阻止你切换分支，你只有commit该更改或者使用git stash，stash只保存工作区和暂存区有跟踪的数据，工作区新建的文件不会保存

git stash

git stash save “stash” 把所有未提交的修改（包括暂存的和非暂存的）都保存起来

### git stash list 查看现有stash

git stash pop命令恢复之前缓存的工作目录。这个指令将缓存堆栈中的第一个stash删除，并将对应修改应用到当前的工作目录下。

git stash apply 命令恢复之前缓存的工作目录。这个不会删除stash

git stash branch bname 把stash的内容通过新建一个分支显示出来