所有的版本控制系统，其实只能跟踪文本文件的改动，比如TXT文件，网页，所有的程序代码等

Git安装后：开始->Git Bash

进入命令行窗口配置用户名和e-mail(--global:加此参数表示这台机器上所有的git仓库都使用这个配置)

git config --global user.name “your name”

git config --global user.email “your email”

1. 创建版本库（repository）

1.mkdir “目录名” : 目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪,目录里确保不包括中文

2.git init:把目录变成git可以管理的仓库（pwd:git命令，用于显示当前目录）

3. git add “文件名”:在当前目录下创建一个文件，用git add “文件名”命令把一个文件添加到仓库缓存区

git add .:将当前目录下修改的所有文件添加到缓存区

git add 文件名1 文件名2:同时添加多个文件到缓存区

4.git commit –m “文件说明”:将缓存区的文件提交到本地仓库

5.git status:查看仓库当前状态

6.git diff <HEAD> 文件名:查看文件的具体修改内容

二. 版本回退

1.git log: 显示从最近到最远的提交日志

git log --pretty=oneline:简要输出提交日志

2.git reset --hard HEAD^:回退到上一个版本.(HEAD^^:上上一个版本,HEAD~100:往上100个版本)

git reset --hard 版本号(版本号可不写完整):回退到指定版本.

3.git reflog:记录你的每一次操作指令.

三. 工作区和缓存区:

工作区:创建的git目录.

版本库: .git 文件夹.

注:创建版本库时,git为我们自动创建一个唯一分支master.

四. 撤消修改.

1.git checkout -- 文件名: 撤消未add的修改

2.git reset HEAD 文件名:撤消此文件的add操作,HEAD表示最新的版本.

五. 删除操作.

若删除本地工作区的文件后要将删除提交,可如下操作:

git rm 文件名->git commit –m “”

或 git add 文件名->git commit –m “”(将删除操作提交)

六、git账号

用户名：caiysdc

密码：yuanmeidjtc2815

Email：1757325938@qq.com

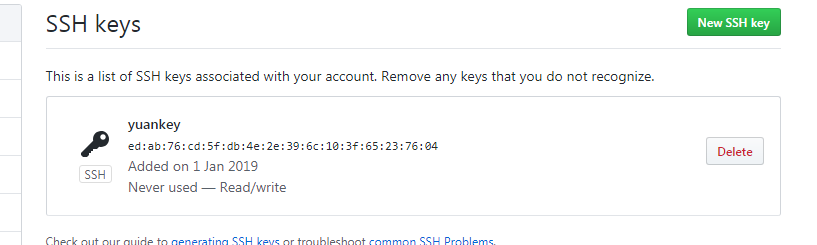
七、远程仓库

1.创建SSH Key(在git Bash内):

ssh-keygen –t rsa -C “email”(有双引号)

创建成功后，在/c/Users/Administrator或c:/users目录下会有.ssh文件夹，里面有id\_rsa（私钥）和id\_rsa.pub（公钥）两个文件。（创建前可先查看此目录下是否有.ssh文件夹和其中两个文件，若有则无需创建）

2. 登陆GitHub，打开“Account settings”，“SSH Keys”页面，添加add Key,结果如下：



2.添加远程仓库

a.登录github，右上角->（+）new repository

b.输入repository名，点击Create repository按钮

c.git remote add origin [git@github.com](mailto:git@github.com):caiysdc/git\_repository.git:将本地仓库与远程仓库关联(origin为远程库名，可更改，我的远程库名现在为myorigin)

d.git push -u myorigin master:第一次推送master分支的所有内容

git push: 将本地库的内容推送到远程。由于远程库是空的，我们第一次推送master分支时，加上了-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令

3.从远程库克隆

A、在github上创建远程库（右上角+ -> New Repository -> 输入Repository name -> 勾选Initialize this repository with a README (GitHub会自动为我们创建一个README.md文件) –> 点击 Create Repository按钮）

B、用git clone克隆一个本地库

git clone [git@github.com:caiysdc/gitSkills.git](mailto:git@github.com:caiysdc/gitSkills.git)

注意：GitHub给出的地址不止一个，还可以用https://github.com/caiysdc/gitSkills.git这样的地址。实际上，Git支持多种协议，默认的git://使用ssh，但也可以使用https等其他协议。

使用https除了速度慢以外，还有个最大的麻烦是每次推送都必须输入口令，但是在某些只开放http端口的公司内部就无法使用ssh协议而只能用https。

八、分支管理

1、创建与合并分支（HEAD指向分支->分支指向提交）

A、创建分支: git checkout –b dev(分支名)

-b参数表示创建并切换=git branch dev（创建分支） -> git checkout dev（切换分支）

B、git branch: 查看所有分支， \* 表示当前分支

C、git merge 分支名： 合并此分支到当前分支

注意： Fast-forword信息说明，此次合并是快进模式，即直接把master指向dev的当前提交，合并速度非常快

D、git branch –d 分支名： 删除分支

2、解决冲突

A．冲突时，Git用<<<<<<<，=======，>>>>>>>标记出不同分支的内容，我们修改后保存

B、git log --graph --pretty=oneline --abbrev-commit: 用带参数的git log可看到分支的合并情况

3、分支管理策略：

git merge --no-ff –m “提交注释说明” 分支名：用--no-ff 强制禁用Fast-forward提交，此时git在merge时会生成一个新的commit,但删除分支前还能从分支历史上看出分支信息。

分支管理原则：master分支应该是非常稳定的，仅用来发布新版本，平时不在上面干活，一般在dev上干活，在发布版本时，再把dev分支合并到master上。

九、bug分支：开发中需要修复bug,但当前dev工作还未完成，不便提交时可用。

1、git stash: 把当前工作现场储藏起来

2、git stash list: 查看储藏起来的所有工作现场

3、git stash pop(==git stash apply -> git stash drop):恢复工作现场并删除，若储藏了多个工作现场，每次只能恢复一个，默认为第一个，可在命令后面加上每一个工作现场的标识在指定要恢复的工作现场。

修复bug时，我们会通过创建新的bug分支进行修复，然后合并，最后删除；

当手头工作没有完成时，先把工作现场git stash一下，然后去修复bug，修复后，再git stash pop，回到工作现场。

十、feature分支：

每添加一个新功能，最好新建一个feature分支，在上面开发，完成后，合并，最后，删除该feature分支。

git branch –D 分支名： 强制删除此分支，若此分支有提交未合并时，用-d删除时会报错并提示，用-D才能删除, 若要丢弃一个没有被合并过的分支，可以通过git branch -D <name>强行删除。

十一、多人协作

git remote ： 查看远程库信息

git remote –v : 查看远程库详细信息

git push <远程库名 分支名> ：把该分支的所有本地提交推送到指定远程库

git branch --set-upstream-to <branch-name> origin/<branch-name>:建立本地分支和远程分支的链接关系。（？？git branch --set-upstream-to=origin/dev dev (or git branch --set-upstream branch-name origin/branch-name)）