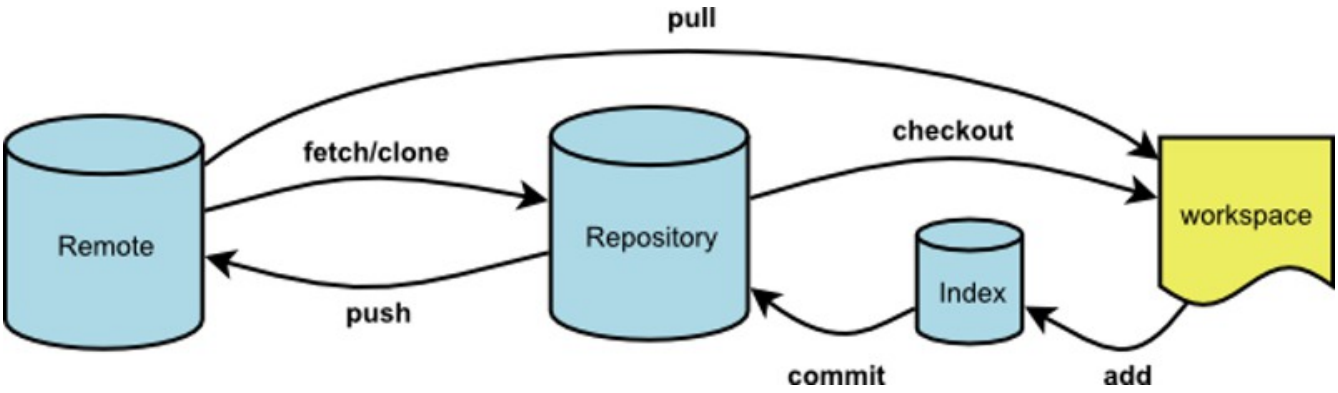
学习Git



【基本知识】

GUI Command-line(cli)

以管理员身份运行

Window与Linux 命令基本不一样

1：cmd/PowerShell

2:Terminal/iTerm

Path 告诉命令行去哪执行，这个变量记录了一组目录，它们用分号分隔，当我们在命令行里输入命令的时候，它会依次去找是否有这个名字的可执行文件。

添加就在我的电脑属性高级Path设置（底下的框）那里，把路径加在最后，用分号隔开

explainshell.com

注：在使用的任何的 Git 命令前，都要切换到 git 项目目录下

【技巧】

输入时可按Tab键（输出剩余文件名）

mkdir frank; mv frank frank2; 连成一行,都执行

mkdir frank && mv frank frank2; 必须前面的执行

!! 重新显示你上一次显示的命令（再运行）

alt +. 把你上一次的参数直接给抄下来

mkdir "a b" 如果名字有空格，需要加引号

cd a\ b/

cd "a b/"

选中-中键-复制粘贴

【清除】

clear 清空界面

【查询】

pwd 当前所处目录 process work directory

ls 列出当前目录的文件

ls -a a是all 还会显示以.开头的目录。

ls ../ 上层目录

ls -l 点出更长的信息 缩写:ll

ls --help | less 帮助

ls -h

xxx --help

xxx -h

git status 查看当前仓库的状态

git status --short 当前文件目录的文件状态

git status -b 显示分支

git status -sb

git remote -v 查看远程仓库

默认的第一个远程仓库叫origin

每一个本地仓库可有多个分支，默认的第一条路线叫master

本地的分支名可以与远程仓库的分支名可以不同

cd /c 看c盘

git log 看提交历史

【指定】

cd 换目录 change derectory

cd ~当前用户的根目录

cd ..

【文件操作】

mkdir 创建一个目录

mkdir -p a/b/c

touch a.html 创建

rm 删除 remove 默认只能删除文件不能删目录，因为危险

rm -r 删文件

rm -f 可删非空目录

rm -rf 就是 rm -r 与 rm -f 的合写

mv 重命名

cp copy

cp ~r ？

【输出】

echo 1 把1打出来

echo 1 > test.txt 把前面这个命令的结果打到后面的文件（覆盖）

cat test.txt 输出文件内容

cat test.txt | less 管道，把第一个命令的内容作为第二个命令的参数

head 显示前几行

head -n 3 显示前三行

tail -n 3 尾三行

du 显示目录的大小

du -hs

du -sh

【退出】

q 退出

ESC :wq 保存退出

quit! 不保存离开

【编辑】

vim是vi的升级版本

vim test.txt 进入后为不可编辑模式，然后按i

vim-adventures.com

【版本控制】

以前有svn,svn没有远程仓库你就无法上传。现在一般用git

【仓库】

git init 初始化为一个仓库 ls是看不见的，ls -a看得见

git add file 添加改动至缓存区

git add . 添加改动

git rm –cached file 更改 add 后的文件的状态为无状态(从缓存区移除)

git commit a1 -m"message" 托管，提交缓存区内容并附带提交信息

git config user.name yourname 配置用户名

git config user.email youremail 配置邮箱

git commit a1 -am"message" am = add+ -m 的组合技，前提是被改动文件已经是tracked

git add* 提交所有文件

git add.* 提交所有带点的文件

git add a/b/c/1.txt

【生成SSH公钥】

ssh-keygen -t rsa -C username@example.com

cat pub那一段，回车,或

$ cat ~/.ssh/id_rsa.pub（或者直接打开.SSH文件）

然后复制出来的那一段到github上

【clone】

git clone git@github.com:jirengu-inc/jrg-renwu10.git

git config --global user.name 你的名字 注意，在公司里时候不要写global，会泄露个人信息

git config --global user.email 你的邮箱

$ git clone -o jQuery https://github.com/jquery/jquery.git

$ git remote

jQuery

上面命令表示，克隆的时候，指定远程主机叫做jQuery

【push与pull】

git pull命令的作用是，取回远程主机某个分支的更新，再与本地的指定分支合并。它的完整格式稍稍有点复杂。

$ git pull <远程主机名> <远程分支名>:<本地分支名>

比如，取回origin主机的next分支，与本地的master分支合并，需要写成下面这样。

$ git pull origin next:master 如果远程分支是与当前分支合并，则冒号后面的部分可以省略。

$ git pull origin next

上面命令表示，取回origin/next分支，再与当前分支合并。实质上，这等同于先做git fetch，再做git merge。

$ git fetch origin

$ git merge origin/next

git pull origin master 我把我本地的代码更新到最新的(*),以在你的工作目录中 获取（fetch） 并 合并（merge） 远端的改动。

git push origin master:master

还有一种情况，就是不管是否存在对应的远程分支，将本地的所有分支都推送到远程主机，这时需要使用--all选项。

$ git push --all origin

注意，分支推送顺序的写法是<来源地>:<目的地>，所以git pull是<远程分支>:<本地分支>，而git push是<本地分支>:<远程分支>。

如果远程主机的版本比本地版本更新，推送时Git会报错，要求先在本地做git pull合并差异，然后再推送到远程主机。这时，如果你一定要推送，可以使用--force选项。

$ git push --force origin

上面命令使用--force选项，结果导致远程主机上更新的版本被覆盖。除非你很确定要这样做，否则应该尽量避免使用--force选项。

【追踪】

在某些场合，Git会自动在本地分支与远程分支之间，建立一种追踪关系（tracking）。比如，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也允许手动建立追踪关系。

git branch --set-upstream master origin/next

上面命令指定master分支追踪origin/next分支。 如果当前分支与远程分支存在追踪关系，git pull就可以省略远程分支名。

$ git pull origin

上面命令表示，本地的当前分支自动与对应的origin主机"追踪分支"（remote-tracking branch）进行合并。

如果当前分支只有一个追踪分支，连远程主机名都可以省略。

$ git pull

上面命令表示，当前分支自动与唯一一个追踪分支进行合并。

如果合并需要采用rebase模式，可以使用--rebase选项。

$ git pull --rebase <远程主机名> <远程分支名>:<本地分支名>

如果远程主机删除了某个分支，默认情况下，git pull 不会在拉取远程分支的时候，删除对应的本地分支。这是为了防止，由于其他人操作了远程主机，导致git pull不知不觉删除了本地分支。

但是，你可以改变这个行为，加上参数 -p 就会在本地删除远程已经删除的分支。

$ git pull -p # 等同于下面的命令

$ git fetch --prune origin $ git fetch -p

【fetch】

$ git fetch <远程主机名> 将某个远程主机的更新，全部取回本地.git fetch命令通常用来查看其他人的进程，因为它取回的代码对你本地的开发代码没有影响。

$ git fetch <远程主机名> <分支名> 默认情况下，git fetch取回所有分支（branch）的更新。如果只想取回特定分支的更新，可以指定分支名。

$ git checkout -b newBrach origin/master 取回远程主机的更新以后，可以在它的基础上，使用git checkout命令创建一个新的分支。

此外，也可以使用git merge命令或者git rebase命令，在本地分支上合并远程分支。

$ git merge origin/master # 或者 $ git rebase origin/master 表示在当前分支上，合并origin/master

【本地-远程】

git remote 列出远程主机

git remote -v 可以参看远程主机的网址

$ git remote show <主机名> git remote show命令加上主机名，可以查看该主机的详细信息

git remote add <主机名> 地址 远程添加

$ git remote rm <主机名> 删除远程主机

$ git remote rename <原主机名> <新主机名> 给远程主机改名

git config --global push.default matching

git push 还有问题，发现远程仓库那边没有master分支

git push --set-upstream origin master

git push -u origin master(简写，第一次需要)

【多人合作】

添加合作者——在setting；或者是访问team里边的仓库

【本地冲突问题】

0.git diff 在合并改动之前，也可以使用此命令查看

1.git pull

2.UU

3. 解决冲突 ==== <<<< >>>>

4.git add.;git commit.

5.git push

git fetch 只把仓库给同步下来，index里的东西都不变，无冲突，不改工作目

录，等完成了再去合并代码,后接git pull 就更新至本地了

【分支查询】

git branch 列出当前分支，前面有星号的为当前所在分支

git branch -r 查看远程分支

git branch -a 查看本地与远程所有分支

【创建本地分支】

git branch branchname 建立分支.新建分支是基于当前所在的分支基础上进行的，即以上命令是基于 master 分支新建了一个叫 develop 的分支，此时 develop 分支跟 master 分支的内容完全一样

git checkout branchname 切换到目标分支

git checkout -b branchname 可以把上面两个步骤和合并，新建并自动切换到 develop 分支

git push origin branchname 把 branchname 分支推送到远程仓库

git push origin branchname:branchname 把 branchname 分支推送到远程仓库，如果远程仓库没有该分支则会创建

【创建远程分支】

git push -u origin newbranchname

git checkout -b release origin/release 如果远程有个分支 release ,而本地没有，你想把远程的 release 分支更新到本地

【删除分支】

git branch -d branchname1 branchname2 不合并就删除会警报（除非原分支什么变动都没有），但确认删除的话

git branch -D branchname

git push origin :branchname 删除一条远程分支

【合并分支】

git merge a 和 git rebase a 把 a 分支的内容合并到 master 分支，前提要先切换到 master 分支，两个做法的区别是： merge 是把 a 中的内容全部粗暴的合并进来，rebase 会先比较内容改变的顺序，再按顺序合并

【撤销】

git checkout --<filename> 此命令会使用 HEAD 中的最新内容替换掉你的工作目录中的文件。已添加到缓存区的改动，以及新文件，都不受影响。

git fetch origin;git reset --hard origin/master 假如你想要丢弃你所有的本地改动与提交，可以到服务器上获取最新的版本并将你本地主分支指向到它

git reset 56883c（先用git log查)

git reset --hard 56883c 连硬盘都改

原则：只要coomit过就不会丢失。即使reset --hard

git reflog

git checkout **SHA1**切换到某次 commit ，这个**SHA1**是每次 commit 的 **SHA1** 值，可以使用 git log 查看

git checkout test.txt 撤销 test.txt 的改变，注 checkout 命令只能撤销还没有 add 进暂存区的文件

【stash】

git stash 把当前分支没有 commit 的代码先暂存起来，这时使用 git status 会发现分支很干净

使用场景，如你正在一个分支 a 上修改着代码什么的，但是这时候，有一个紧急的任务，需要你切换到另一个分支 b 去做些工作， 而 a 上的代码还是个半吊子，不想去 commit 甚至不想去 add ，这时候 stash 命令就大有用处了，前提是没有执行 commit

git stash list 可以看到此时暂存区多了一条记录

git stash apply重回原来的分支

git stash drop 就是把最近的一条 stash 记录删除， drop 后还可以跟 stash_id 来删除指定的记录

git stash pop 这个命令相当于同时执行了 apply 和 drop ，谨慎起见，使用 git stash list 查看一下是否确实删除了该记录

git stash clear 清空暂存区的所有记录，drop 只是删除一条，drop 后可以跟 stash_id 来删除指定的记录，不跟就是删除最近的

git diff可以查看冲突,解决冲突要做的就是修改冲突的文件 test.txt 为最终想提交的内容，并去掉其中的标识符 (标识符: ++<<<<<<< HEAD；++=======； ++>>>>>>>>>) 修改然后 add 和 commit

【标签】

git tag 1.0.0 1b2e1d63ff

git push不会推送标签（tag），除非使用--tags选项。 $ git push origin --tags