【 Git 】

1. 参考网站

图解Git： https://marklodato.github.io/visual-git-guide/index-zh-cn.html

Git详解（七个部分）：http://www.cnblogs.com/BeginMan/p/3548139.html

2. Git多账号共存

描述：当我们同一个git客户端，想要使用多个账号来进行工作的时候，那么账号之间必须是独立的，每个账号分管自己的git仓库（git init初始化仓库），比如：办公使用的git账号，个人自己github使用的git账号等。如果是单用户那就很方便，默认生成的公私钥即可。多账号共存具体步骤如下：

1. 【生成公私钥】分别生成各个用户使用的公私钥【备注：一定要取自定义名称】在~/ssh目录下执行：ssh-keygen -t rsa -C "自己的邮箱"【第一个回车后输入名称】，随后一路回车即可，生成目录~/.ssh/下
2. 【添加公私钥】新秘钥添加到SSH agent中

   因为默认只读取id_rsa，为了让SSH识别新的私钥，需将其添加到SSH agent中：

   执行ssh-add ~/.ssh/id_rsa_work，

   如果出现Could not open a connection to your authentication agent的错误，就试用以下命令：

   ```
   ssh-agent bash
   ssh-add ~/.ssh/id_rsa_work
   ```
3. 【修改config文件配置】

   ~/.ssh/目录下需要配置config配置文件，没有的话需要创建：touch config即可，文件内容参考如下：

   Host和HostName 尽量保持一致！

   ```
   ----------begin----------
   #github配置
   Host github.com
   HostName github.com
   User Leyo
   Port 22
   IdentityFile ~/.ssh/github
   #leyo配置
   Host gitlab.leyo.net
   HostName gitlab.leyo.net
   User LIYAO-SZ
   Port 22
   IdentityFile ~/.ssh/leyo
   ----------end----------
   ```
4. 【github添加公钥】

   在github中添加SSH，打开~/.ssh/github.pub文件，将其内容拷贝到github的公钥中。
5. 【测试】样式【git@Host】

   ssh -T git@github.com 和 ssh -T git@gitlab.leyo.net

   会有：Hi LeeYoo! You've successfully authenticated, but GitHub does not provide shell access. 提示则代表本地和远程github绑定成功！

   如有出现类似如下提示，则说明config文件某一行有问题：

   ssh: connect to host github.com port 22: Bad file number

   备注，参考http://www.cnblogs.com/BeginMan/p/3548139.html

3. 常用命令

基本命令

初始化仓库： $ git init

克隆代码： $ git clone 代码地址

更新本地代码： $ git pull

抓取远程代码： $ git fetch

更新远程库到本地：$ git fetch origin

提交到缓冲区： $ git add test $ git add *

本地提交： $ git commit -a

推送代码： $ git push

内建的图形化git： $ gitk

对最近的一次commit进行修改（commit版本号不变）： git commit -a --amend

commit之后，如果想撤销最近一次提交(即退回到上一次版本)并本地保留代码：git reset HEAD^

分支相关

列出所有分支： $ git branch

创建分支：　　$ git branch mybranch
切换分支：　　$ git checkout mybranch
创建并切换分支：　　$ git checkout -b mybranch
推送分支到远程：　　$ git push origin mybranch

删除远程分支：　　$ git push origin :mybranch

远程分支更名：　　$ git branch -m <old name> <new name>

删除本地分支：　　$ git branch -d mybranch
强制删除分支：　　$ git branch -D mybranch
合并分支：（在当前分支操作合并其他分支过来）　$ git merge yjb-2.1.0
取远程分支合并到本地：　　$ git merge origin/mybranch
取远程分支并分化一个新分支：　　$ git checkout -b mybranch origin/mybranch
更新master主线上的东西到该分支上：　　$git rebase master

更新mybranch分支上的东西到master上：　$git rebase mybranch

合并分支：(merge from)　　$ git checkout master　$ git merge mybranch (merge from mybranch)
查看各个分支最后一次提交：　　$ git branch -v
查看哪些分支合并入当前分支：　　$ git branch --merged

查看哪些分支未合并入当前分支：　$ git branch --no-merged

标签相关

列出当前分支所有的tag：　　$git tag
搜索符合模式的标签：　　$git tag -| 'v0.1*'
切换标签：　　$git checkout V1.0.0（指向V1.0.0标签时的代码状态，但是在一个空的分支）
打轻量标签：　　$git tag V1.0.0　（无需参数）
打附注标签：　　$git tag -a V1.0.0 -m "V1.0.0版本"（-a指定标签类型，-m指定标签说明参数）
（备注：轻量标签是指向提交对象的的引用，而附注标签是仓库中的一个独立对象，建议使用附注标签）
删除标签：　　$git tag -d V1.0.0
给指定的commit打标签：　　$git tag -a V1.1.0 9bcfe2e1c　（最后的校验码是某次commit的，可以补打标签）
标签发布：（将制定标签提交到git服务器）　　$git push origin V1.0.0
标签发布：（将本地所有标签一次性提交到git服务器）　　$git push origin -tags

回滚相关

1. 还没执行 add 可以执行删除文件，已经add 的时候但没commit，

   回滚要用 ：**$git checkout xxxx**

2. 已经add 并且已经commit，回滚要用 ：**$git reset HEAD……**

3. 已经commit 就是版本回滚了，**$git reset 版本**

4. 已经push到远程，oh no！只能从删掉然后本地删掉了 然后提交上去，不能用回滚操作

当change id未生成时，使用：$**git commit --amend** 再次生成方可提交。

其他命令

1　git init 初始化一个Git仓库。

2　添加文件到Git仓库，分两步：

　　git add <filename>，该命令可以多次使用，添加多个文件；
　　git commit -m "commit提交注释说明"，完成。

3　git status 查看当前工作区的状态。
4　git diff <filename> 查看修改的内容，红色为删除部分；绿色为新加部分。

5 git log 查看系统中历史commit提交记录。
如：git log或者git log --pretty=oneline，只输出一行附加参数。
其中pretty指定打印提交记录内容的格式，可选值由：oneline，short，medium，full等等。
补充，--graph可以查看分支合并图，--abbrev-commit打印简短commit_id。
6 git reset --hard <commit_id> 将head指针设定为指向指定的commit_id。
HEAD表示当前版本id
HEAD^表示上一个版本id
HEAD^^表示上上个版本id
HEAD~100表示上上100个版本id
版本id也可以通过git log命令查看，commit就是了。
7 git reflog 记录每一次有关head的历史命令记录
8 **工作区、暂存区、版本库**三者概念。
工作区，英文名叫做working directory就是当前工作目录；
暂存区，英文名叫做stage，已添加但是还未提交到仓库的区域；
版本库，英文名叫做repository，对应为在工作区下隐藏目录.git/
补充，版本库里边又细分分支、HEAD指针等。
git add添加文件命令，将文件从工作区放到暂存stage区；
git commit提交命令就是把stage区的所有内容提交到当前分支中。
9 git checkout --<filename> 把工作区中对filename文件的修改撤销掉，分两种情况：
其一为修改后还没放到暂存区（即还没使用git add命令）现在撤销修改就是回到版本库状态；
其二为已经添加到暂存区后，又做了修改，现在撤销修改就是回到添加到暂存区后的状态。
总之，就是让该文件回到最近一次git add或git commit时的状态。
请注意，git checkout <name> 是切换到对应的分支上。
10 git reset HEAD <filename> 把暂存区中对filename的修改撤销掉。
此处为把修改从暂存区撤销到工作区，要想撤销工作区的修改，要进一步使用：
git checkout --<filename>命令处理。
若修改已经提交到版本库中，在没有把本地版本推送到远程仓库情况下可以使用：
git reset --hard <commit_id>命令进行版本回退处理。
11 git rm <filename> 删除暂存区中的文件filename，使用git commit提交删除申请，从版本库中删除文件。
12 ssh-keygen -t rsa -C "huge@163.com" 在用户主目录下生成密钥对。-t rsa指定加密方式为rsa加密方式。
13 git config --global user.name "leyo" 设定用户名，全局有效。
14 git config --global user.email "leyo@leyo.com" 设定邮箱，全局有效。
15 git remote add origin git@github.com:<github用户名>/<仓库名>.git
将本地仓库和github仓库关联起来。
16 git push -u origin master 将本地仓库的所有内容推送到远程仓库上，只在第一次推送的时候加上-u参数。
git push origin master可能会遇到远程有添加文件，但本地没有，故先执行git pull将远程库拉到本地库中。
git pull处理完毕后，再执行git push origin master推送本地仓库内容到远程仓库。
17 git clone git@github.com:<github用户名>/<仓库名>.git 从远程仓库克隆到本地仓库。
18 git branch 查看分支；git branch <name> 创建分支；git checkout <name> 切换分支。
git branch --set-upstream <branchname> origin/<branchname> 将本机分支和远程分支关联起来。
git checkout -b <name> 创建并切换到分支name，
相当于是git branch <name>和git checkout <name>两步操作。
19 git merge <name>合并分支name到当前分支，若合并发生冲突，直接要解决冲突
（通过git status查看冲突文件）
必须手动处理冲突文件，处理完毕后经过git add/commit提交过程。
--no-ff 参数表示禁止使用 Fast forward 模式合并，应为本次合并创建一个新的提交commit，
加入-m注释说明信息。
20 git branch -d <name> 删除分支name，大D参数为git branch -D <name>强行删除name分支。
21 git stash 将当前工作现场储藏起来，等以后恢复现场后继续工作。
22 git stash list 查看当前现场情况，恢复方式有：
git stash apply 恢复，但是恢复后，stash内容并不会删除，需要通过git stash drop删除。
git stash pop 恢复的同时将stash内容也删除掉，相当于是出栈操作。
23 git remote 查看远程仓库信息，加v参数为git remote -v展示抓取地址和push推送地址。
24 git pull 把别人最新的提交从远程仓库抓取下来。
25 git tag <name>[commit_id] 在commit_id处创建一个标签，commit_id默认为HEAD。
26 git show <tagname> 查看标签详细信息。
27 .gitignore文件，忽略特殊文件，参考http://github.com/github/gitignore 即可。
28 见下方更多内容为：
a、多人协作的工作模式为：
第一步：尝试用git push origin branch-name推送自己的修改。
第二步：如果推送失败，则因为远程分支比你的本地分支更新，需要先用git pull试图合并
若合并有冲突，则解决冲突，并在本地提交。
反之没有冲突或者解决冲突后，再用git push origin branch-name推送就能成功了。
另外一种异常流程为：
若git pull提示no tracking information则说明本地分支和远程分支的链接关系没有创建。
git branch --set-upstream branch-name origin/branch-name
关联上本地分支和远程分支的关系即可。
b、标签管理：在发布上线后，打一个标签处理。
命令为：git tag tag_name 打标签的名称，默认打标签是打在最新提交的commit上的。
推送标签到远程仓库为：git push origin tagname
或者 git push origin --tags 推送全部标签到远程仓库。
git tag tag_name commit_id 还能对历史的commit打标签处理。

git tag 强烈建议提供描述说明，增加上git tag -a v0.1 -m "version v0.1 released"
-m 增加上描述信息，查询tag详细信息命令为：git show v0.1
-d 删除标签，若标签已经推送到远程，要删除远程标签两个步骤为：
第一步：删除本地标签。git tag -d v0.9
第二步：删除远程标签。git push origin :refs/tags/v0.9

c、git配置文件为：

git配置文件分 **全局配置文件** 以及 **仓库私有配置文件** ，全局配置设置--global参数。
比如，git config --global alias.co checkout 设定checkout别名为co。
全局配置文件默认放在：C:\Users\Administrator\目录下的.gitconfig文件中。
仓库私有的配置文件放在仓库目录下的.git/目录下的config文件。

d、Windows中大小写不敏感

Windows系统对大小写不敏感，所以大部分客户端会设置一个默认的参数core.ignorecase=true，这样的话，git就不会以文件名称的大小写差异来区别文件。
在Windows下，执行git config --global core.ignorecase=false后，git才能区分文件名称的大小写。

3. 注意事项

git本地做了很多commit，如何修改其中某次的commit

第一种方式： 利用分支

首先记录下想要修改的commit 哈希值${commit_id}，重新建立一个分支rewrite_branch

git checkout -b rewrite_branch ${commit_id}

修改。。。

git add -u

git commit --amend

把原来分支${commit_id}之后的提交，再一个个的cherry-pick回来。

第二种方式：rebase

首先执行 git rebase -i ${commit_id}^后，在页面中修改 ${commit_id}一行中的pick为edit，保存离开

然后修改代码，git add -> git commit --amend -> git rebase --continue

如果还不熟悉rebase，可以借助分支；熟悉之后，可以大胆的用rebase。

如何把本地的多次commit合并为一次

例如我本地分支上针对一个编号为1024的缺陷做了多次commit，怎么把这几个commit作为一次commit推送出去？

```
$ git branch -av
* hotfix_1024 6323854 [ahead 3] fixed bug 1014

$ git lg

* 6323854 - (HEAD, hotfix_1024) fixed bug 1014 (2015-02-02 20:03:33 +0800) <yanfeng-sz>
* a9aaa39 - fixed bug 1014 (2015-02-02 20:02:09 +0800) <yanfeng-sz>
* 59b28a0 - fixed bug 1014 (2015-02-02 20:00:37 +0800) <yanfeng-sz>
* 2feb727 - (origin/master, origin/HEAD, master)
```

【解决方法】回到master分支，把hotfix分支上的修改squash为一次commit

第一步：切换到master分支

```
$ git checkout master
```

第二步：压缩合并hotfix分支

```
lava@lava-pc /cygdrive/d/workaround/testing
$ git merge -n --squash hotfix_1024
更新 2feb727..6323854
Fast-forward
压缩提交 -- 未更新 HEAD
all_project_list | 66
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1 file changed, 66 insertions(+)
create mode 100644 all_project_list
```

第三步：提交，编辑commit-msg

```
$ git commit
[master b33e3a1] fixed bug 1014
1 file changed, 66 insertions(+)
create mode 100644 all_project_list
```