

Git基本使用

一、GIT基本知识

工作区 --> 暂存区 --> 版本库 --> 远程版本库

工作区：文件的增加，修改，删除操作都在工作区执行

暂存区：文件修改后且add后，到暂存区

版本库：文件commit后，到版本库

远程仓库：本地版本库的文件push到远程仓库，从远程仓库pull/fetch文件到本地

HEAD保存的是最后一次提交点（当前），指向当前工作的分支

HEAD^ 上一个版本

HEAD^^ 上上个版本

HEAD~10 上10个版本

二、配置

安装git后执行以下配置

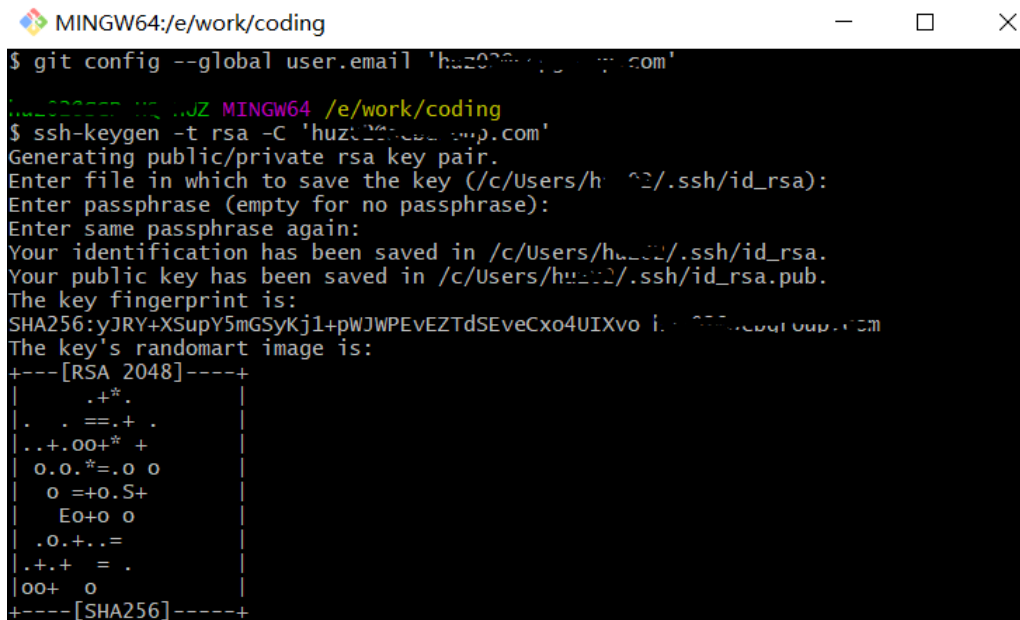
1.配置用户名及邮箱

git config --global user.name 'huzhong' 使用域账号

git config --global user.email 'hu123@qq.com' 使用公司邮箱

2.生成ssh key

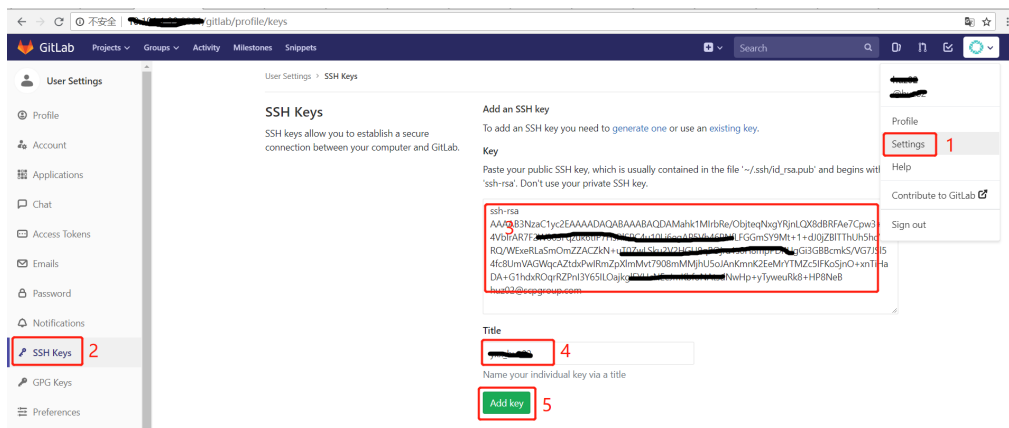
(1)在bash中执行以下命令：ssh-keygen -t rsa -C 'xxx@xxx.com'，然后一路按回车



```
MINGW64:/e/work/coding
$ git config --global user.email 'huzhong@xxx.com'
$ ssh-keygen -t rsa -C 'huzhong@xxx.com'
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/huzhong/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/huzhong/.ssh/id_rsa.
Your public key has been saved in /c/Users/huzhong/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:yJRY+XSupY5mGSyKj1+pWJWPEvEZTdSEveCxo4UIXvo i@xxx.com
The key's randomart image is:
+---[RSA 2048]---+
| .+* |
| . . ==.+ |
| ..+.00+*+ |
| 0.0.*=.0 0 |
| o =+0.S+ |
| Eo+0 0 |
| .O.+..= |
| .+. + =. |
| oo+ 0 |
+---[SHA256]-----+
```

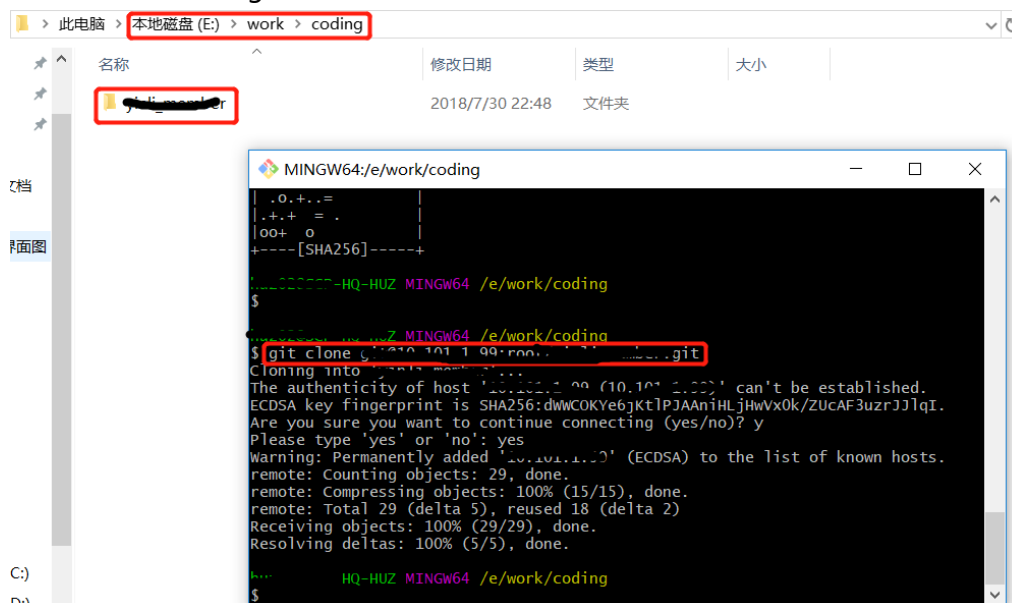
(2)然后打开~/.ssh/id_rsa.pub文件(~表示用户目录，比如我的windows就是C:\Users\huz02\.ssh)，复制其中的内容

(2)打开gitlab，找到Profile Settings-->SSH Keys--->Add SSH Key,并把上一步中复制的内容粘贴到Key所对应的文本框，在Title对应的文本框中给这个sshkey设置一个名字，点击Add key按钮



然后复制项目中的代码地址，如：

进入指定目录，执行git clone命令就可以把代码克隆下来



3.提交代码时每次都需要输入密码，解决方法

(1)在~/下，touch创建文件 .git-credentials, 用vim编辑此文件，输入内容格式：

创建文件：touch .git-credentials

编辑文件：vim .git-credentials

文件内容：https://{username}:{password}@github.com

(2)在终端下执行 git config --global credential.helper store

(3)可以看到 ~/.gitconfig 文件，会多了一项：

```
[credential]
    helper = store
```

三、基本操作

1.查看配置信息

git config -l

2.初始化仓库（本地仓库）

git init

3.克隆远程代码

git clone url

3.拉取远程代码

git pull 相当于 git fetch 和git merge

4.从其他分支合并代码到当前分支

git merge branch-name

4.比较文件

git diff [filename]

4.添加文件

git add [.]filename]

5.提交文件

git commit -[a]m '备注信息'

6.查看仓库状态

git status

\$ git status

On branch master 处于master分支

Initial commit 初始化提交

Untracked files: 未跟踪的文件

(use "git add <file>..." to include in what will be committed) 使用add命令来添加文件

test.txt 为跟踪的文件名

nothing added to commit but untracked files present (use "git add" to track) 没有提交但未添加文件

(用 "git add" 追踪)

7.查看日志

git log

commit cbc220915fa1039e475b7865cc05bc42c6a5e826 提交的编号

Author: huz02 <huz02@vanke.com> 作者

Date: Tue Nov 28 14:23:10 2017 +0800 提交日期

add test.txt 提交的内容 (添加test.txt文件)

git log --pretty=oneline 格式化查看日志

8.查看某个提交修改的内容

git show commitID

12.查看某个文件修改记录

git log -p filename

四、分支管理

查看分支

git branch -a // all 全部分支
git branch -r // remote 远程分支
git branch -l // local 本地分支

创建分支

git branch branch_name

切换分支

git checkout branch_name

创建分支并切换到分支

git checkout -b new_branch exists_branch (默认为空, 从master分支拉取代码)

删除分支, 如果正在当前分支, 则不能删除

git branch -d branch_name

删除远程分支

git push origin --delete branch_name

合并自分支代码, 先切换到master分支

git checkout master 切换到主分支

git merge sub_branch 合并sub分支代码到主分支

```
$ git merge sub
```

```
Updating 5ce93f7..b59cc48
```

```
Fast-forward
```

```
hahaha.txt | 17 ++-----
```

```
1 file changed, 2 insertions(+), 15 deletions(-)
```

注意: Fast-forward 表示快速合并 (将master的指针指向brh), 不会产生新的commit id, 只是利用了子分支的commit id继续操作

注意: 如果在子分支修改了代码, 未commit就切换到master, master也会显示文件被修改了

推送分支代码到远程分支

git push origin master

git push origin sub_branch 推送sub分支

推送本地分支到远程, 如果远程分支没有, 则使用下面命令

git push --set-upstream origin invoice

五、代码回退

1.修改本地文件, 还未add操作, 注意: 所有修改将丢失

git checkout -- filename

2.添加新文件且执行了add，想返回未add状态，保留修改的把内容

git reset HEAD filename

3.如果文件被删除后，想要恢复源文件

git checkout HEAD -- filename

4.文件执行了commit后，想回到上一个版本（log会被删除）

git reset --soft commit_id 回退到制定版本，回到add后，commit前的代码

git reset commit_id 回退到指定版本修改后，回到修改后，add前的代码（默认--mixed，可省略）

git reset --hard commit_id 回退到指定版本，回到修改前的代码

如a.txt

版本6文件内容：66666666666666666666666666666666

修改文件为：77777777777777777777777777777777

提交到版本7

如果执行 git reset HEAD^ 返回版本6，但文件内容为77777777777777777777777777777777，可执行add，commit

如果执行 git reset --hard HEAD^ 返回版本6，文件内容为66666666666666666666666666666666

六、代码开发

进行功能开发前不允许直接在开发分支上开发，必须新建一个任务分支，开发完成后合并到开发分支

步骤	命令	含义
1	Git checkout 153_develop	切换到当前公共开发分支
2	Git pull	拉取公共开发分支最新代码
3	Git checkout -b 23434	创建23434对应的本地私人开发分支
4	开发	
5	Git add .	将需要添加到23434提交清单的文件添加进提交列表
6	Git commit -m "提交描述"	提交修改记录到23434 - 提交规范例子：v153-develop-张三-禅道号23434，喜盈佳发 票返回时，判断状态不正常时等待时间 - （此步需要在本地开发测试结束后进行）

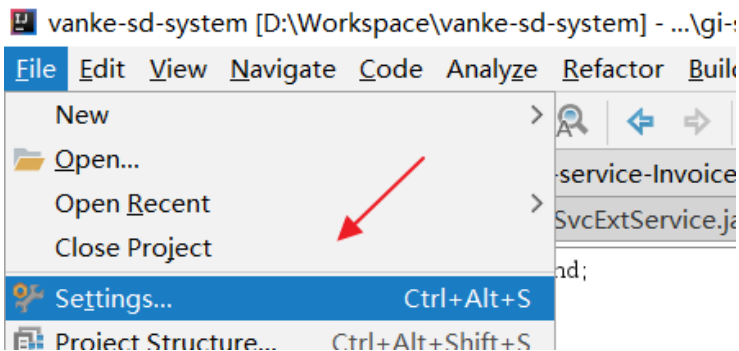
私有分支合入共有分支操作步骤

步骤	命令	含义
1	Git checkout 153_develop	切换到公用开发分支
2	Git pull	在开发本地开发期间，可能其他开发已经提交了很多代码，需要把这些提交拉取
3	Git checkout 23434	切换到私人禅道开发分支
4	Git rebase -i 153_develop	将公用开发分支其他开发的代码压入私人禅道分支
5	Git checkout 153_develop	切换到公用开发分支
6	Git pull	如果发现公共开发分支其他人又有提交，重复上面1到5的步骤
7	Git merge 23434	将禅道分支提交记录合并到公共分支
8	Git fetch	探测远程是否有人有新的提交，如果有联系管理人员处理
9	Git push	如果第8步探测到远程没有更新，则可以直接推送

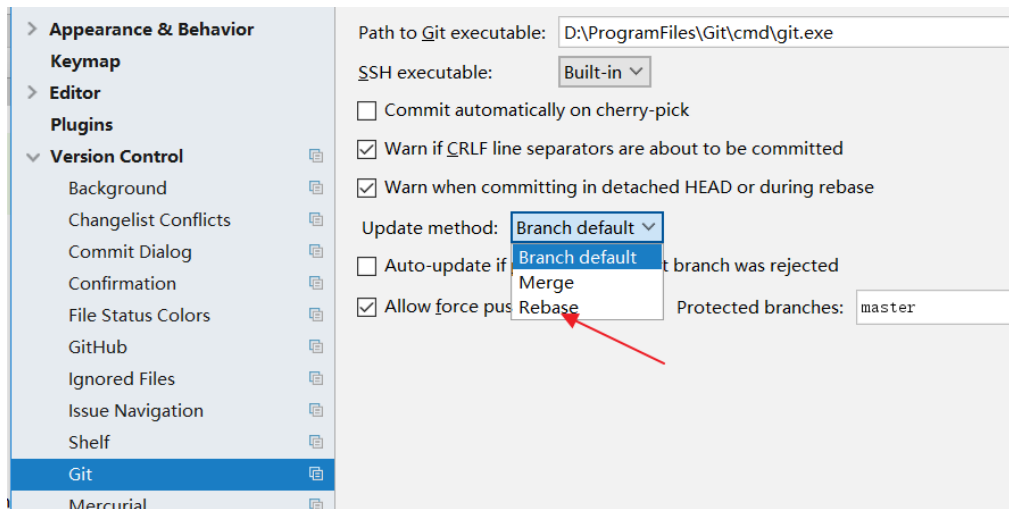
习惯用IntelliJ开发的，做好下面设置，可以，忽略前面PPT提出的要求，用IntelliJ做代码管理

使用IntelliJ和JetBrain WebStorm开发和做代码管理，需要设置对应IDE的代码管理默认设置，以IntelliJ为例：

1，打开IDE设置



2, 设置更新方法默认为 “Rebase”



七、冲突管理

手动解决, 可以使用sourcetree, Tortoise GIT

八、sourceTree安装

1. 下载软件

2. 安装, 先要安装git, 跳过注册安装

在C:\Users\huz02\AppData\Local\Atlassian\SourceTree目录增加文件accounts.json

内容如下:

```
[
  {
    "$id": "1",
    "$type": "SourceTree.Api.Host.Identity.Model.IdentityAccount, SourceTree.Api.Host.Identity",
    "Authenticate": true,
    "HostInstance": {
      "$id": "2",
      "$type": "SourceTree.Host.Atlassianaccount.AtlassianAccountInstance, SourceTree.Host.AtlassianAccount",
      "Host": {
        "$id": "3",
        "$type": "SourceTree.Host.Atlassianaccount.AtlassianAccountHost, SourceTree.Host.AtlassianAccount",
        "Id": "atlassian account"
      }
    }
  }
]
```

```
    },  
    "BaseUrl": "https://id.atlassian.com/"  
  },  
  "Credentials": {  
    "$id": "4",  
    "$type": "SourceTree.Model.BasicAuthCredentials, SourceTree.Api.Account",  
    "Username": "",  
    "Email": null  
  },  
  "IsDefault": false  
}  
]
```

3.使用操作: https://blog.csdn.net/baidu_33570760/article/details/72764339

其他参考资料

分支的管理和创建, 可查看网址

<https://git-scm.com/book/zh/v1/Git-%E5%88%86%E6%94%AF-%E5%88%86%E6%94%AF%E7%9A%84%E6%96%B0%E5%BB%BA%E4%B8%8E%E5%90%88%E5%B9%B6>

<https://www.cnblogs.com/zhangxiaoyong/p/6000084.html>