设置更新 仓库地址

git remote set-url origin 仓库地址

# 拷贝项目

我们使用 **git clone** 从现有 Git 仓库中拷贝项目（类似 **svn checkout**）

克隆仓库的命令格式为：

git clone <repo>

如果我们需要克隆到指定的目录，可以使用以下命令格式：

git clone <repo> <directory>

**参数说明：**

**repo:** Git 仓库。

**directory:** 本地目录。

几种效果等价的git clone写法：

git clone http://github.com/CosmosHua/locate new

git clone http://github.com/CosmosHua/locate.git new

git clone git://github.com/CosmosHua/locate new

git clone git://github.com/CosmosHua/locate.git new

## 克隆分支

git clone -b <branch> <**repo**> 例如： git clone -b 指定的分支名字  
git clone <**repo**> -b <branch> 这样就是某个分支呀  
简单 git clone -b <branch> <**repo**>

# 配置

（用户名和密码）

git config --global user.name "eaglezsx"

git config --global user.email ["15732621728@163.com"](mailto:\"15732621728@163.com\")

# 初始化

可以指定一个目录

git init <directory>

# 常用指令

git show # 显示某次提交的内容 git show $id

git add <file> # 将工作文件修改提交到本地暂存区

git add . # 将所有修改过的工作文件提交暂存区

git commit -m ‘’ 提交已经被add进来的改动.

git pull 抓取远程仓库所有分支更新并合并到本地; 会首先执行git fetch,然后执行git merge,把取来的分支的head merge到当前分支.这个merge操作会产生一个新的commit.

git push 将本地主分支推到远程主分支;将会把当前分支merge到alias上的[branch]分支.如果分支已经存在,将会更新,如果不存在,将会添加这个分支.

git pull --no-ff # 抓取远程仓库所有分支更新并合并到本地，不要快进合并

git fetch origin # 抓取远程仓库更新

git merge origin/master # 将远程主分支合并到本地当前分支

git push # push所有分支

git push origin master # 将本地主分支推到远程主分支

git push -u origin master # 将本地主分支推到远程(如无远程主分支则创建，用于初始化远程仓库)

git push origin <local\_branch> # 创建远程分支， origin是远程仓库名

git push origin <local\_branch>:<remote\_branch> # 创建远程分支

git push origin :<remote\_branch> #先删除本地分支(git br -d <branch>)，然后再push删除远程分支

## Git暂存管理

git stash # 暂存

git stash list # 列所有stash

git stash apply # 恢复暂存的内容

git stash drop # 删除暂存区

## 查看提交记录

git log git log <file> # 查看该文件每次提交记录

git log -p <file> # 查看每次详细修改内容的diff

git log -p -2 # 查看最近两次详细修改内容的diff

git log --stat #查看提交统计信息

## 查看文件diff

git diff <file> # 比较当前文件和暂存区文件差异 git diff

git diff <id1><id1><id2> # 比较两次提交之间的差异

git diff <branch1>..<branch2> # 在两个分支之间比较

git diff --staged # 比较暂存区和版本库差异

git diff --cached # 比较暂存区和版本库差异

git rm <file> # 从版本库中删除文件

git rm <file> --cached # 从版本库中删除文件，但不删除文件

git reset <file> # 从暂存区恢复到工作文件

git reset -- . # 从暂存区恢复到工作文件

git reset --hard # 恢复最近一次提交过的状态，即放弃上次提交后的所有本次修改

git revert <$id> # 恢复某次提交的状态，恢复动作本身也创建次提交对象

git revert HEAD # 恢复最后一次提交的状态

git status // 查看当前文件状态

git merge dev // 合并当前分支到dev分支

## 版本回退

$ git log // 从近到远查看提交日志

$ git reset --hard HEAD^ // 回退到上一个版本，^^两个箭头就表示上上个版本，依次类推

$ git reset --hard 3628164 // 回退到指定版本号

$ git reflog // 查看最近几次提交记录的版本号

在某些情况下，比如说因为某些原因，我从2.0版本回退到了1.0版本，这个操作可以通过git reset --hard HEAD^命令实现，但是这时我又想回到2.0版本怎么办呢？这时候去查看git log命令，发现最新的版本是1.0，git log查看的是提交历史，似乎是没有什么办法回到2.0版本了。答案是 git reflog命令，该命令可以查看命令历史，可以通过找到2.0的版本号，再通过git reset --hard回到2.0版本。

## 撤销修改

$ git checkout -- readme.txt // 把readme.txt在工作区的修改全部撤销

$ git reset HEAD readme.txt // 把readme.txt暂存区的修改撤销，重新放回工作区

修改文件的时候，突然发现文件修改错了，但是这时候已经修改了很多，无法手动恢复，趁还没有git add的时候把它恢复，可以执行git checkout -- filename命令。如果当你发现的时候已经执行了git add操作，也就是文件的修改已经被提交到了暂存区怎么办呢？这时候就应该先执行git reset HEAD filename命令，把缓存区的修改放回工作区，再执行git checkout -- filename撤销工作区修改的文件。当你删除文件时候，也可以通过git checkout -- filename命令恢复，因为对git来说，删除也是一种修改。

## 解决冲突

### 第一种方法，stash：

**那怎么stash本地的更新呢？直接执行：**

**git stash**

**git pull**

**git stash pop**

**接下来diff一下此文件看看自动合并的情况，并作出相应的修改。**

**git stash：备份当前的工作区，从最近一次提交中读取相关内容，让工作区保持和上一次提交的内容一致。同时，将工作区的内容保存到git栈中。**

**git stash pop：从git栈中读取最近一次保存的内容，恢复工作区的相关内容。由于可能存在多个stash的内容，所以用栈来管理，pop会从最近一个stash中读取内容并恢复到工作区。**

### 第二种方法：放弃本地的修改，直接覆盖

**git reset --hard**

**git pull origin develop:develop**

**git commit --amend 撤销上一次提交 并讲暂存区文件重新提交**

**git checkout -- <file> 拉取暂存区文件 并将其替换成工作区文件**

**git reset HEAD -- <file> 拉取最近一次提交到版本库的文件到暂存区 改操作不影响工作区**

## [git 放弃本地修改](https://www.cnblogs.com/qufanblog/p/7606105.html)

### 未使用 git add 缓存代码时

 git checkout -- filepathname

可以使用 git checkout -- filepathname (比如： git checkout -- readme.md  ，不要忘记中间的 “--” ，不写就成了检出分支了！！)。放弃所有的文件修改可以使用 git checkout .  命令。

此命令用来放弃掉所有还没有加入到缓存区（就是 git add 命令）的修改：内容修改与整个文件删除。但是此命令不会删除掉刚新建的文件。因为刚新建的文件还没已有加入到 git 的管理系统中。所以对于git是未知的。自己手动删除就好了。

### 已经使用了git add 缓存了代码。

 git reset HEAD filepathname

可以使用  git reset HEAD filepathname （比如： git reset HEAD readme.md）来放弃指定文件的缓存，放弃所以的缓存可以使用 git reset HEAD . 命令。

此命令用来清除 git  对于文件修改的缓存。相当于撤销 git add 命令所在的工作。在使用本命令后，本地的修改并不会消失，而是回到了如（一）所示的状态。继续用（一）中的操作，就可以放弃本地的修改。

### 已经用 git commit提交了代码。

 git reset --hard HEAD^

可以使用 git reset --hard HEAD^ 来回退到上一次commit的状态。此命令可以用来回退到任意版本：git reset --hard  commitid

你可以使用 git log 命令来查看git的提交历史。git log 的输出如下,之一这里可以看到第一行就是 commitid：

commit cf0d692e982d8e372a07aaa6901c395eec73e356 (HEAD -> master)  
Author: toyflivver <2440659688@qq.com>  
Date: Thu Sep 28 14:07:14 2017 +0800

可以看出现在的状态在 commitid 为 cf0d692e982d8e372a07aaa6901c395eec73e356 的提交上（有 HEAD -> master 标记）。

如果在此之间有其他人提交过代码，必须执行一次

git pull 拉取最新

### **[码云](https://www.cnblogs.com/ycyzharry/p/10782860.html)**

### [报错Incorrect username or password ( access token )](https://www.cnblogs.com/ycyzharry/p/10782860.html)

使用码云将仓库clone到本地，报错信息如下：

D:\123>git clone https://gitee.com/ycyzharry/helloworld.git

Cloning into 'helloworld'...

remote: Incorrect username or password ( access token )

fatal: Authentication failed for 'https://gitee.com/ycyzharry/helloworld.git/'

后面继续clone时候以为会弹出登录窗口，结果直接提示用户名或密码错误。

解决办法：

打开电脑的控制面板–>用户账户–>凭据管理器

找到普通凭据git:https://gitee.com这一栏，选择编辑，填入正确的用户名和密码，最后点击保存即可。