1. **创建一个 SSH key**

$ ssh-keygen -t rsa -C "your\_email@example.com"

代码参数含义：

-t 指定密钥类型，默认是 rsa ，可以省略。  
-C 设置注释文字，比如邮箱。  
-f 指定密钥文件存储文件名。

以上代码省略了 -f 参数，因此，运行上面那条命令后会让你输入一个文件名，用于保存刚才生成的 SSH key 代码，如：

Generating public/private rsa key pair.

# Enter file in which to save the key (/c/Users/you/.ssh/id\_rsa): [Press enter]

当然，你也可以不输入文件名，使用默认文件名（推荐），那么就会生成 id\_rsa 和 id\_rsa.pub 两个秘钥文件。

接着又会提示你输入两次密码（该密码是你push文件的时候要输入的密码，而不是github管理者的密码），

当然，你也可以不输入密码，直接按回车。那么push的时候就不需要输入密码，直接提交到github上了，如：

Enter passphrase (empty for no passphrase):

# Enter same passphrase again:

接下来，就会显示如下代码提示，如：

Your identification has been saved in /c/Users/you/.ssh/id\_rsa.

# Your public key has been saved in /c/Users/you/.ssh/id\_rsa.pub.

# The key fingerprint is:

# 01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your\_email@example.com

当你看到上面这段代码的收，那就说明，你的 SSH key 已经创建成功，你只需要添加到github的SSH key上就可以了。

**Git分支建立及分支合并**

1. 创建分支

$ git checkout -b <branch-name>

它是下面两条命令的简写：

$ git branch <branch-name> //创建分支

$ git checkout <branch-name>

提交该分支到远程仓库

$ git push origin <branch-name>

1. 切换分支

$ git checkout master

1. 删除分支

$ git branch -d <branch-name>

1. 合并分支

$ git merge <branch-name>

合并之后要push到远程仓库，不然只是合并了本地仓库

git push orgin <branch-name>

git pull origin master(相当于是从远程获取最新版本并merge到本地)

1. 冲突解决

任何因包含合并冲突而有待解决的文件，都会以未合并状态标识出来。 Git 会在有冲突的文件中加入标准的冲突解决标记，这样你可以打开这些包含冲突的文件然后手动解决冲突。 出现冲突的文件会包含一些特殊区段，看起来像下面这个样子：

<<<<<<< HEAD:index.html

<div id="footer">contact : email.support@github.com</div>

=======

<div id="footer">

please contact us at support@github.com

</div>

>>>>>>> iss53:index.html

这表示 HEAD 所指示的版本（也就是你的 master 分支所在的位置，因为你在运行 merge 命令的时候已经检出到了这个分支）在这个区段的上半部分（======= 的上半部分），而 iss53 分支所指示的版本在======= 的下半部分。 为了解决冲突，你必须选择使用由 ======= 分割的两部分中的一个，或者你也可以自行合并这些内容

如果你想使用图形化工具来解决冲突，你可以运行 git mergetool，该命令会为你启动一个合适的可视化合并工具，并带领你一步一步解决这些冲突：

$ git mergetool

Git内容暂存 git stash

### 1. stash当前修改

git stash会把所有未提交的修改（包括暂存的和非暂存的）都保存起来，用于后续恢复当前工作目录。  
比如下面的中间状态，通过git stash命令推送一个新的储藏，当前的工作目录就干净了。

需要说明一点，stash是本地的，不会通过git push命令上传到git server上。  
实际应用中推荐给每个stash加一个message，用于记录版本，使用git stash save取代git stash命令。示例如下：

$ git stash save "test-cmd-stash"

### 2. 重新应用缓存的stash

可以通过git stash pop命令恢复之前缓存的工作目录，输出如下：

这个指令将缓存堆栈中的第一个stash删除，并将对应修改应用到当前的工作目录下。  
你也可以使用git stash apply命令，将缓存堆栈中的stash多次应用到工作目录中，但并不删除stash拷贝。命令输出如下：

$ git stash apply

### 3. 查看现有stash

可以使用git stash list命令，一个典型的输出如下：

在使用git stash apply命令时可以通过名字指定使用哪个stash，默认使用最近的stash（即stash@{0}）。

### 4. 移除stash

可以使用git stash drop命令，后面可以跟着stash名字。下面是一个示例：

### 5. 查看指定stash的diff

可以使用git stash show命令，后面可以跟着stash名字。示例如下：

$ git stash show

在该命令后面添加-p或--patch可以查看特定stash的全部diff，如下：

$ git stash show -p

### 6. 从stash创建分支

如果你储藏了一些工作，暂时不去理会，然后继续在你储藏工作的分支上工作，你在重新应用工作时可能会碰到一些问题。如果尝试应用的变更是针对一个你那之后修改过的文件，你会碰到一个归并冲突并且必须去化解它。如果你想用更方便的方法来重新检验你储藏的变更，你可以运行 git stash branch，这会创建一个新的分支，检出你储藏工作时的所处的提交，重新应用你的工作，如果成功，将会丢弃储藏。

### 7. 暂存未跟踪或忽略的文件

默认情况下，git stash会缓存下列文件：

* 添加到暂存区的修改（staged changes）
* Git跟踪的但并未添加到暂存区的修改（unstaged changes）

但不会缓存一下文件：

* 在工作目录中新的文件（untracked files）
* 被忽略的文件（ignored files）

git stash命令提供了参数用于缓存上面两种类型的文件。使用-u或者--include-untracked可以stash untracked文件。使用-a或者--all命令可以stash当前目录下的所有修改。

# git如何删除已经 add 的文件 (如何撤销已放入缓存区文件的修改)

使用 git rm 命令即可，有两种选择,

一种是 git rm –cached “文件路径”，不删除物理文件，仅将该文件从缓存中删除；

一种是 git rm –f “文件路径”，不仅将该文件从缓存中删除，还会将物理文件删除（不会回收到垃圾桶）。

## 在git中出现中文乱码的解决方案

* 在cygwin中，使用git add添加要提交的文件的时候，如果文件名是中文，会显示形如 274\232\350\256\256\346\200\273\347\273\223.png 的乱码。
  + 解决方案：在bash提示符下输入： git config --global core.quotepath false
  + core.quotepath设为false的话，就不会对0x80以上的字符进行quote。中文显示正常。
* 在MsysGit中，使用git log显示提交的中文log乱码。   
  解决方案：
  + 设置git gui的界面编码 git config --global gui.encoding utf-8
  + 设置 commit log 提交时使用 utf-8 编码，可避免服务器上乱码，同时与linux上的提交保持一致！ git config --global i18n.commitencoding utf-8
  + 使得在 $ git log 时将 utf-8 编码转换成 gbk 编码，解决Msys bash中git log 乱码。git config --global i18n.logoutputencoding gbk
  + 使得 git log 可以正常显示中文（配合i18n.logoutputencoding = gbk)，在 /etc/profile 中添加：export LESSCHARSET=utf-8
* 在MsysGit自带的bash中，使用ls命令查看中文文件名乱码。cygwin没有这个问题。
  + 解决方案：
    1. 使用 ls --show-control-chars 命令来强制使用控制台字符编码显示文件名，即可查看中文文件名。
    2. 为了方便使用，可以编辑 /etc/git-completion.bash ，新增一行 alias ls=”ls –show-control-chars”

Gitignore无效时配置

把某些目录或文件加入忽略规则，按照上述方法定义后发现并未生效，原因是.gitignore只能忽略那些原来没有被追踪的文件，如果某些文件已经被纳入了版本管理中，则修改.gitignore是无效的。那么解决方法就是先把本地缓存删除（改变成未被追踪状态），然后再提交：

git rm -r --cached .  
git add .  
git commit -m 'update .gitignore'

**修改仓库远程地址**

git remote set-url origin git@192.168.0.150:/gitFile/oa

**查看现在的仓库地址**

git remote -v

# git丢弃本地修改的所有文件（新增、删除、修改）

git checkout . #本地所有修改的。没有的提交的，都返回到原来的状态

git stash #把所有没有提交的修改暂存到stash里面。可用git stash pop回复。

git reset --hard HASH #返回到某个节点，不保留修改。

git reset --soft HASH #返回到某个节点。保留修改

git clean -df #返回到某个节点

git clean 参数

-n 显示 将要 删除的 文件 和 目录

-f 删除 文件

-df 删除 文件 和 目录

### 撤销提交信息

**首先，通过git log查看提交信息，以便获取需要回退至的版本号：**

git log

git reset --soft aa909cff2239536df14820fe086d96305b24e9f1

git push origin master --force