# Git clone

git clone [git@github.com:qipingzhiji/Hello-world.git](mailto:git@github.com:qipingzhiji/Hello-world.git) 目录

将仓库克隆至仓库名后的目录中

# 编写代码

进入该目录，在该目录下编写你的代码

# 提交

git status 查看状态

git add 你的文件 将文件加入暂存区

git commit 提交

# 进行push

git push 只要执行了这步，GitHub上的仓库就会更新

# 设置ssh key

**Github连接已有的仓库是通过使用了ssh的公开密钥的认证方式进行的**

**操作：**

**ssh-keygen –t rsa –C”1213462884@qq.com”**

**Enter passphrase(empty for no passphrase):wly0099zht**

**Enter same passphrase again:wly0099zht**

# 创建分支

git checked –b

# 合并分支

git merge --no-ff 要合并的分支

# 用图表形式查看分支

git log –graph

# 回溯历史版本

git reset -–hard 文件操作的hash码

# 查看当前仓库的操作的日志

git reflog

# 修改提交信息

git commit –amend

# 压缩历史

git rebase ~i

# 添加远程仓库

git remote add 别名 [git@github.com:用户名/](mailto:git@github.com:用户名/)仓库名

例如：git remote add origin [git@github.com:qipingzhiji/git-tutorial.git](mailto:git@github.com:qipingzhiji/git-tutorial.git)

# 推送至远程仓库

git push –u 别名 分支

-u参数：添加了这个参数后可以在推送的同时，将origin仓库的master分支设置为本地仓库当前分支的upstream（上游）。添加了这个参数，将来运行git pull命令从远程仓库获取内容时，省去了另外添加参数的麻烦。

# 从远程仓库获取

git clone

# 获取本地仓库分支的信息

git branch –a

# 获取远程的分支

git checkout –b 本地分支名 远程仓库别名/远程分支名

# 推送分支

git push

# 获取最新的远程分支

git pull

# Github中版本的比较

版本之间：例如：

<https://github.com/rails/rails/compare/4-0-stable..3-2-stable>

时间：例如

[https://github.com/rails/rails/master@{7.day.ago}...master](https://github.com/rails/rails/master@%7b7.day.ago%7d...master)

查看与指定日期之间的差别，例如：

https://github.com/rails/rails/compare/master@{2013-01-12}...master

# GitHub中Tasklist语法的使用

例如:

#本用要做的任务

* [ ] 完成图片
* [x] 完成部署工作的设置
* [ ] 实现抽签功能

# GitHub中获取diff格式或者是patch格式的文件

例如：<https://github.com/用户名/仓库名/pull/28.diff>

<https://github.com/用户名/仓库名/pull/28.patch>

# GitHub引用别人的评论

选中要引用的评论按r键就可以

评论中表情可以自动补全，只需要输入冒号及需要的前几个字母就可以。

# GitHub仓库维护

从别人clone克隆出来的仓库与原仓库是没有关系的。需要将原仓库设置为远程仓库。

做法：git remote add 别名 git://github.com/用户名/仓库名

git fetch 你设置的别名

通过git fetch 来获取最新的源码

# Github在本地环境中反应pull request中的内容

1.将接收方的本地仓库更新至最新状态

git clone [git@github.com:接收方姓名/](mailto:git@github.com:接收方姓名/)仓库名

git pull 更新至最新状态

2.获取发送方的远程仓库

git remote add 发送方的姓名 [git@github.com:发送方/](mailto:git@github.com:发送方/)发送方的仓库

git fetch 发送方的姓名

3.创建用于检查的分支

git checkout –b pr1

4.合并

git merge pr发送者/分支名

5.检查并删除分支

git branch –D pr1

# 不Fork仓库的方法

1.在GitHub上进行Fork

2.将1处的仓库clone至本地开发环境

3.在本地环境中创建牲分支

4.对特性分支时行代码修改并提交

5.将特性分支push到1处的仓库中

6.在Github上对Fork来源仓库发送Pull Request

Github Flow的流程：

1.令master分支时常保持可以部署的状态

2.进行新的作业时要从master分支创建新分支，新分支名称要具有描述性

3.在2处新的本地仓库中进行提交

4.在githug端创建创建同名分支，定期push

5.需要帮助或反馈时创建pull request,以pull request进行交流。

6.让其他开发者进行审查，确认作业完成后与master分支进行合并。

7.与master分支合并后立刻部署。

# Git Flow 以发布为中心的开发模式

1.从开发版的分支（develop）分支中创建工作分支（feature branches），进行功能的实现和修改。

2.工作分支（feature branches）的修改结束后，与开发版的分支进行合并

3.重复上述工作，直到可以发布新的功能。

4.创建用于发布的分支（release branches），处理发布的各种工作

5.发布工作完成后，与master分支进行合并，打上版本标签（Tag）进行发布

6.如果发布的软件出现bug后，以打了标签的版本为基础进行修正（hotfixes）

# Git删除远程端的仓库上的文件

git rm –cached 文件名

git commit –m “delete file”

git push

如果是删除文件夹的话，要加上-r参数

# Git 标签

git tag 1.0.0 需要提交的id

# Git 替换本地改动

git checkout -- <filename> :此命令会将HEAD中的最新的内容替换掉工作目录中的文件。