# git工具用途简介

Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。

开源的分布式版本控制系统，命令行

图形界面程序：GitExtensions 、TortoiseGit

应用广泛性：

gitHub源代码管理平台，nodeJs也可以直接从gitHub获取插件

OpenShift用git作发布版本管理，gitBook文档版本管理

最新xcode默认使用git做版本管理

## 特点

基本特点：

分布式，每个人本地都有所有的历史信息，查看一下历史记录，svn也要向远程请求数据。离线可以修改文件、比较文件差异

擅长基于分支的管理，擅长处理复杂业务，每人一个或多个分支，提交的代码都合并到主分支。

基元信息的记录方式，占用存储空间少。

场景：

有冲突的提交，开发时改旧bug，测试时做新功能。

开源项目贡献代码的差异。

不依赖服务器，就可以简单的在本地做版本管理

方便推送到多个中央档案库。本地可以拷贝多份同个档案库。

有很多免费的在线档案库平台可以使用，多种开源项目可以快速创建自己的在线档案库平台。

方便下载开源源码，并可以看到提交记录。是一种方便的代码交流手段。

每人可以拥有多个分支，应对多版本共存问题，理想情况下，在一个版本里的更改也可以合并到新的版本里。

使用账号或SSH密钥验证权限，

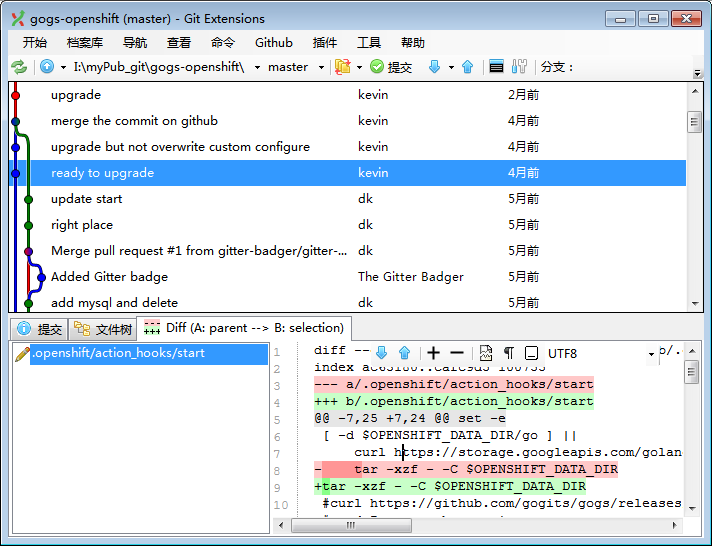
tfs一般要配合vs、域用户使用

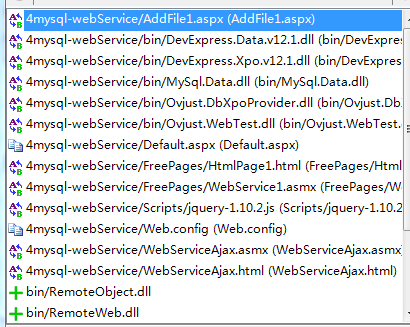
svn偏轻量

开源，容易实现源代码管理网站、发布管理

非常方便的分支切换、提交记录查看、文件比较

适用场景：代码、数据库、发布网站、前端样式设计等的版本管理。





# 基本操作

## 安装、查看文件历史

客户端推荐：

Git for Windows: MsysGit(Git-1.7.4-preview20XX0204.exe)

GitExtensions /**TortoiseGit**

https://github.com/gitextensions/gitextensions/releases

[**GitExtensions-2.48.05-SetupComplete.msi**](https://github.com/gitextensions/gitextensions/releases/download/v2.48.05/GitExtensions-2.48.05-SetupComplete.msi)（包含了git）

[**GitExtensions-2.48.05-Mono.zip**](https://github.com/gitextensions/gitextensions/releases/download/v2.48.05/GitExtensions-2.48.05-Mono.zip)

http://git-extensions-documentation.readthedocs.org/en/latest/getting\_started.html

服务器端：

Git for Windows: MsysGit(Git-1.7.4-preview20XX0204.exe)

OpenSSH for Windows: CopSSH(Copssh\_5.X.4\_Installer.exe)/Gitblit等网站平台

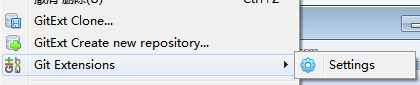
客户端安装后效果：

桌面图标

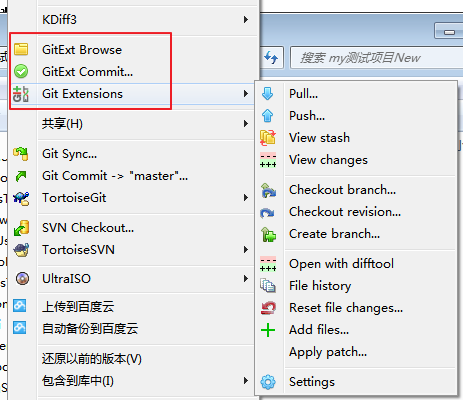


文件夹上右键菜单，与svn类似

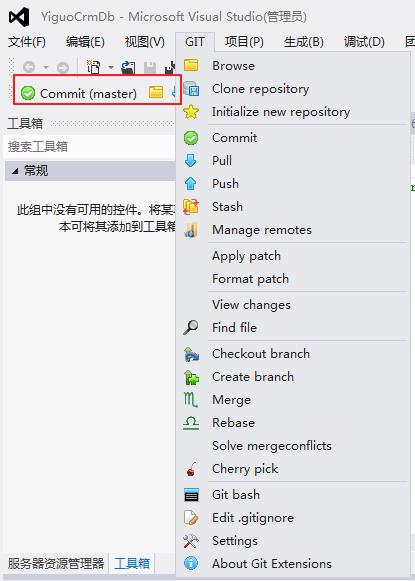
非git项目的文件夹



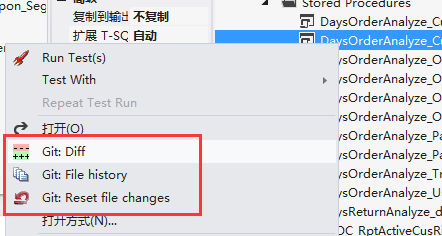
git项目文件夹的右键



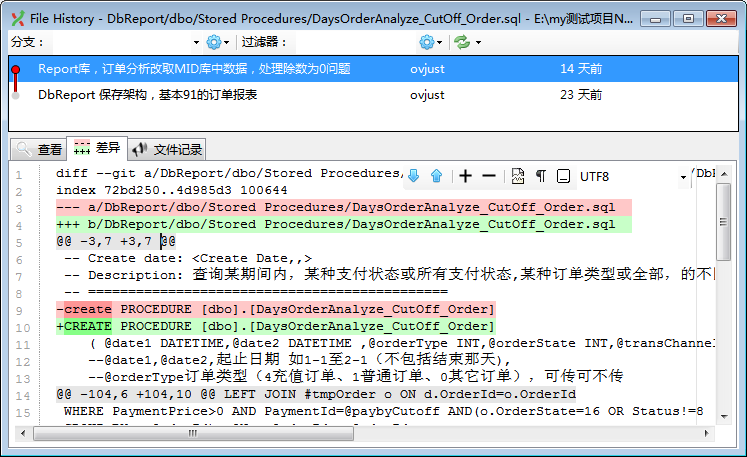
vs的git菜单



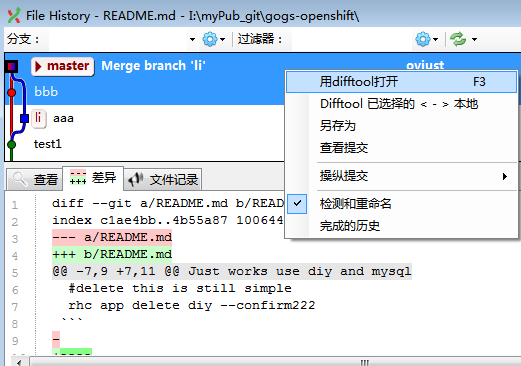
vs中文件右键菜单，文件上右键菜单与此类似。



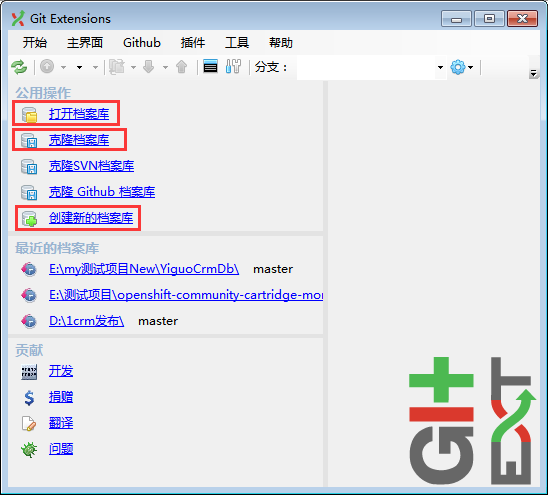
查看历史记录



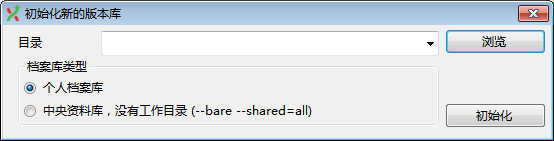
选中两行，可以右键比较



## 档案库初始化、打开档案库

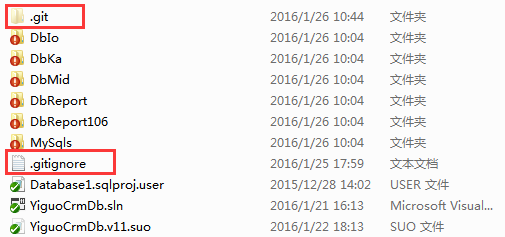


初始化档案库，文件夹是否为空均可



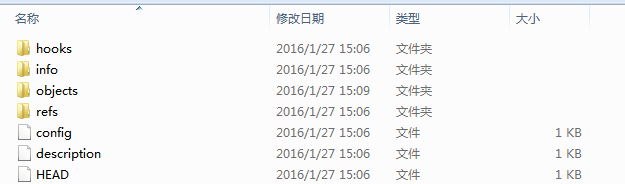
### 文件结构

个人档案库-非裸



中央档案库-裸

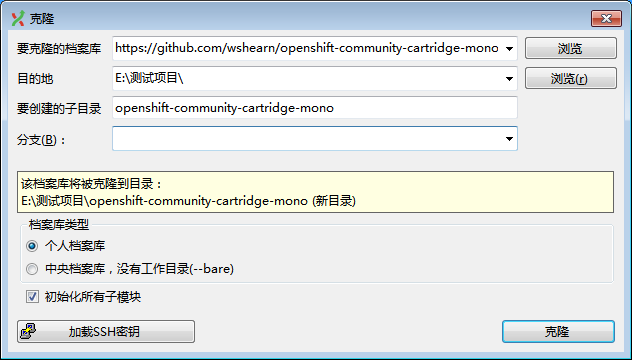
文件目录同上图中的".git"文件夹



### 克隆档案库

git://git.kernel.org/pub/scm/git/git.git

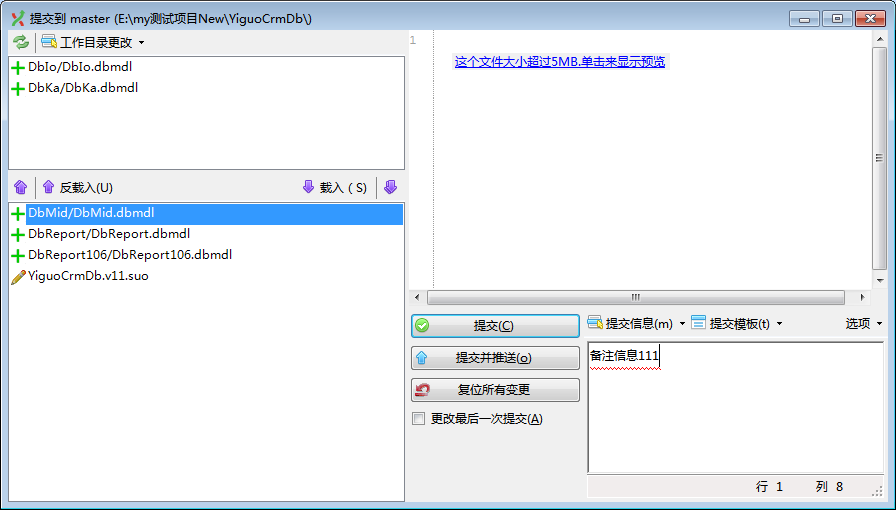
http(s):// [name:password@]www.kernel.org/pub/scm/git/git.git



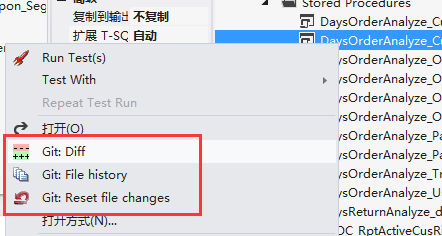
## 提交（本地）、推送push（远程）、撤销更改

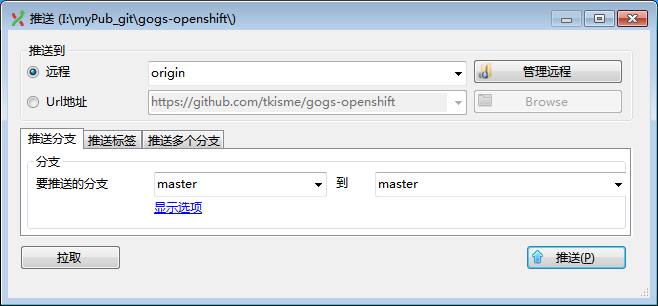
提交，保存在本地

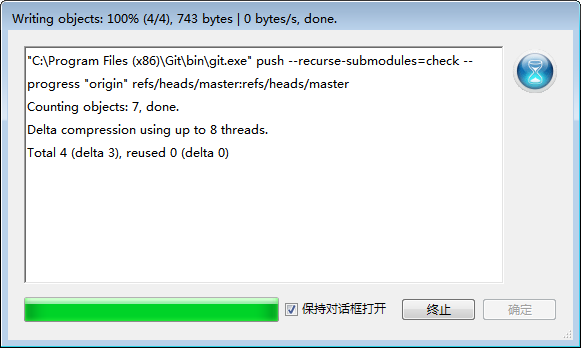
推送，更新到远程



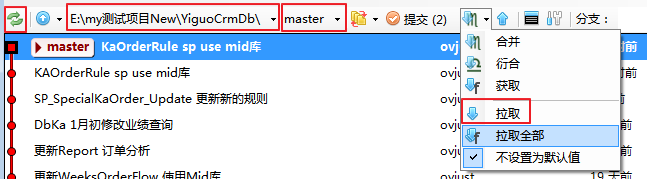
上图及下图里都可以做撤销操作，基本每个操作都有三个入口可以选择（文件夹右键、vs中git菜单、gitExt浏览器）。

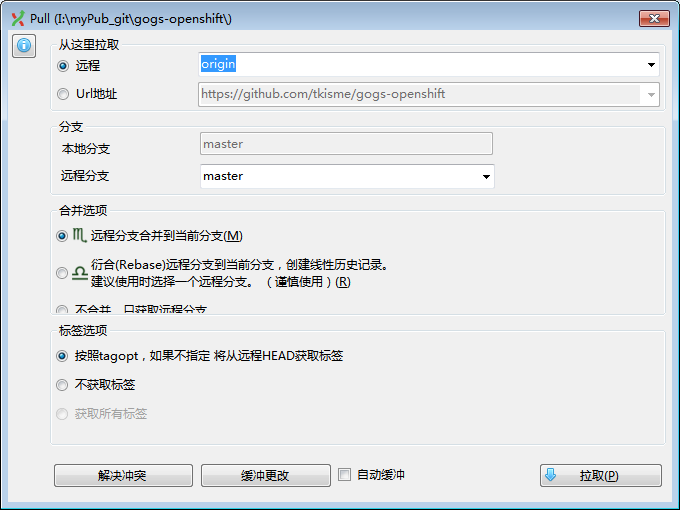




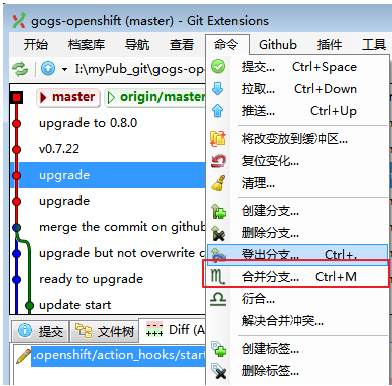


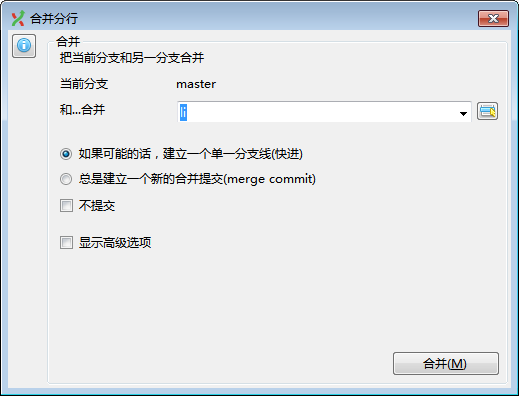
## 拉取pull，拉取全部fetch --all

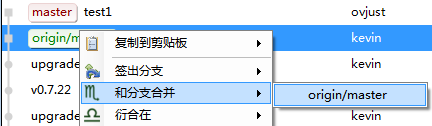




## 合并merge 、解决冲突



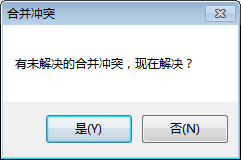


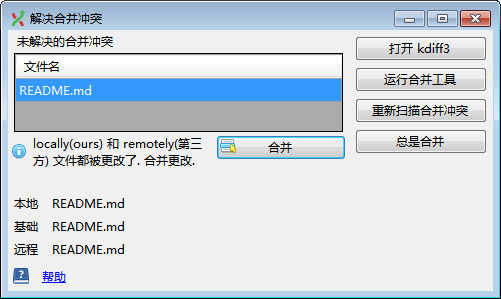


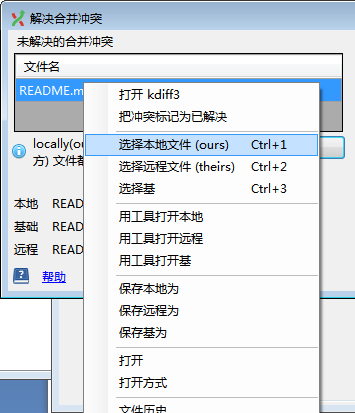
不同点：此时是和分支的此次提交合并，上面的是和分支的最后一次提交合并

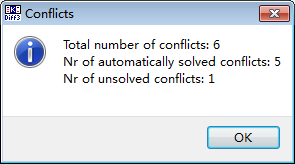
合并遇到冲突

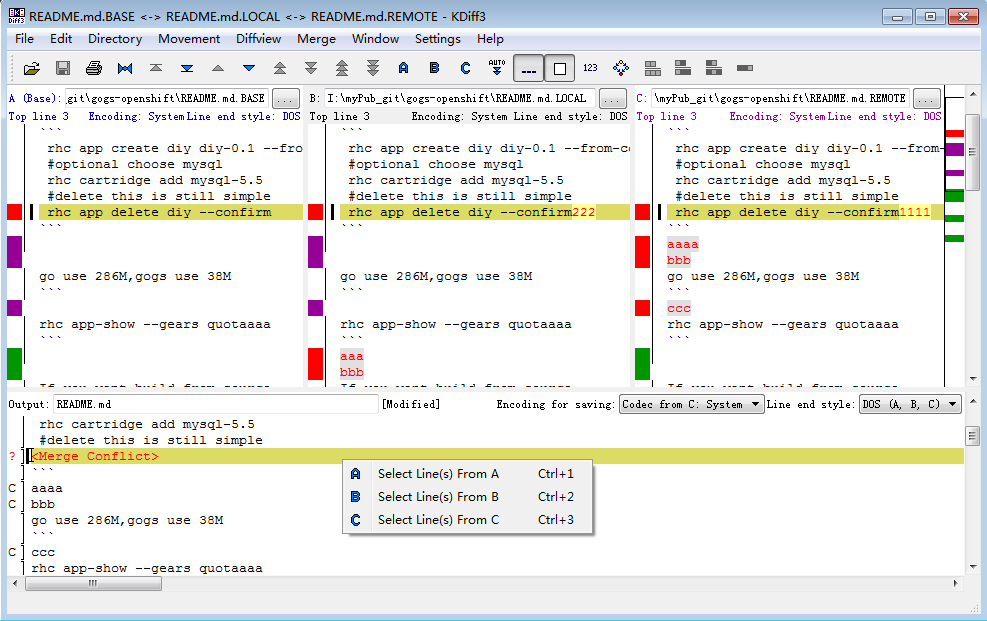




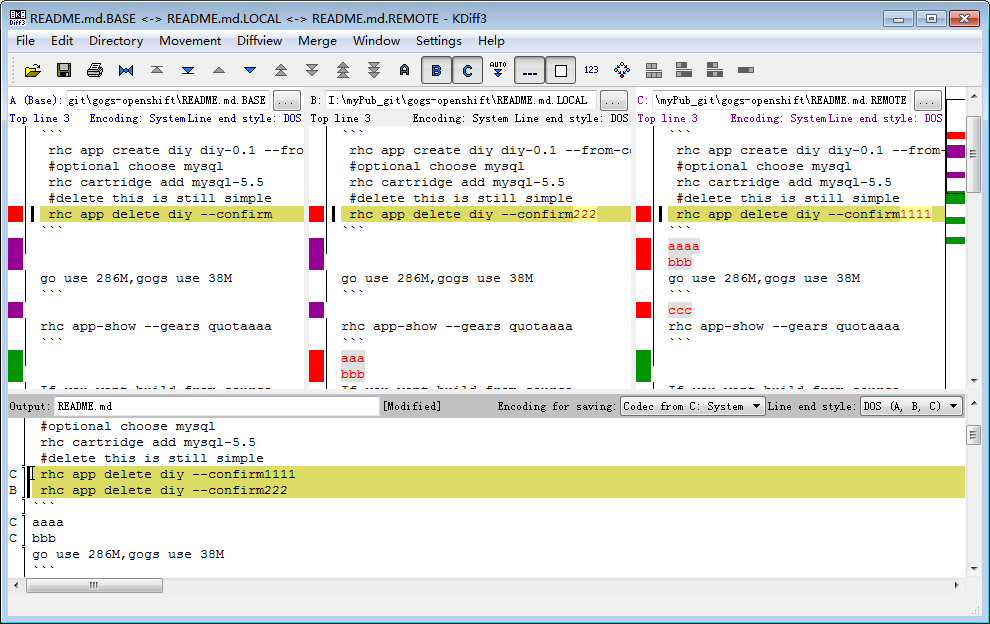




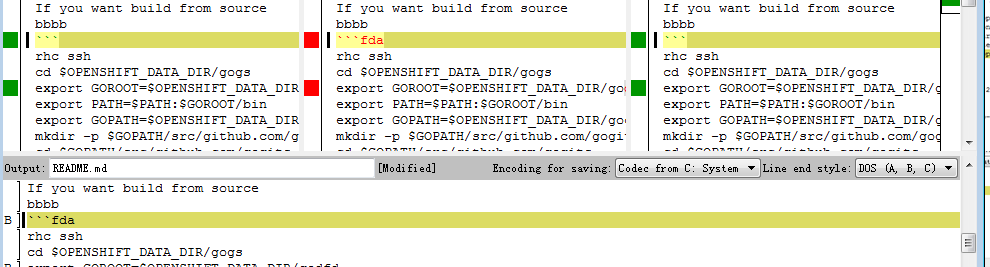




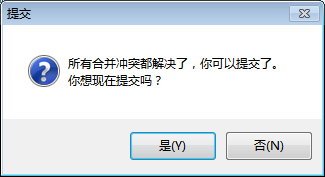
选择使用哪个版本的

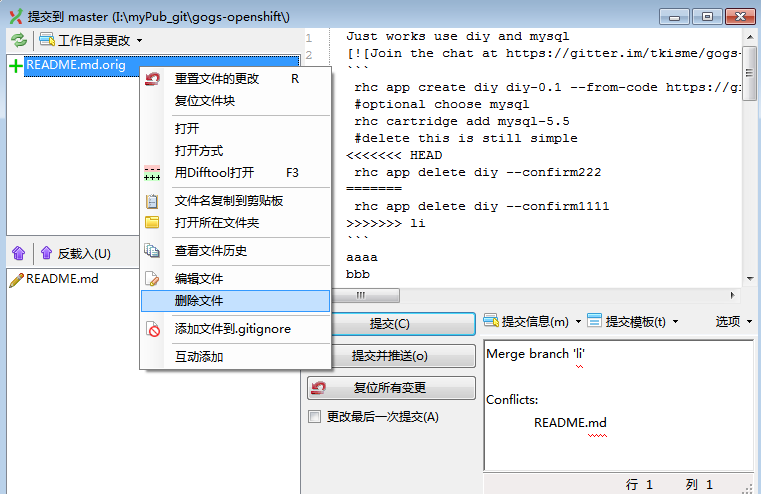


绿色的标头的行表示：已自动处理的冲突

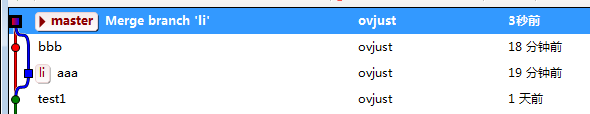


保存后





再合并到master时 就不会有冲突。



## 获取代码步骤

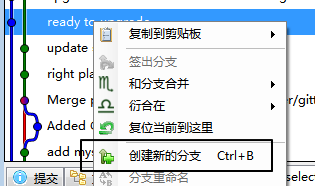
1. 拉取远程master到本地master
2. 合并本地master到本地li
3. 如果有冲突，解决冲突，本地运行测试，无问题再提交到本地li

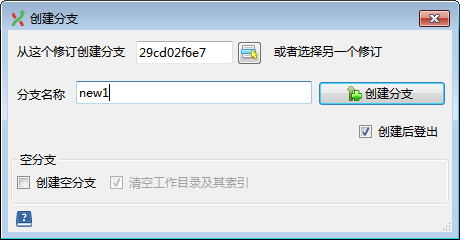
## 提交代码步骤

1. 提交到本地li。
2. 如果master有更新，执行“获取代码步骤”
3. 合并本地li到本地master，此时不会存在冲突。
4. 推送本地li、master到远程

如果程序不能正常生成，不要合并到主分支。当可能有冲突时，不推荐直接把个人分支合并到主分支、远程分支，不利于解决冲突，容易误将有问题代码提交到主分支。

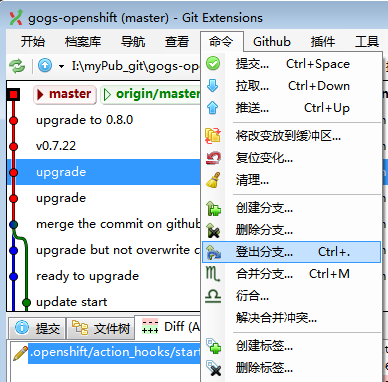
## 创建分支

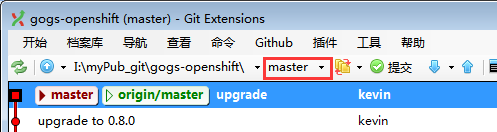


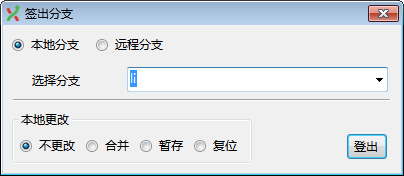


## 切换分支

本地文件都会更新为新分支的。



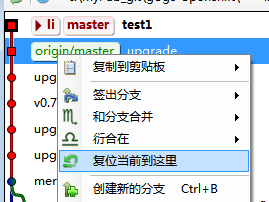


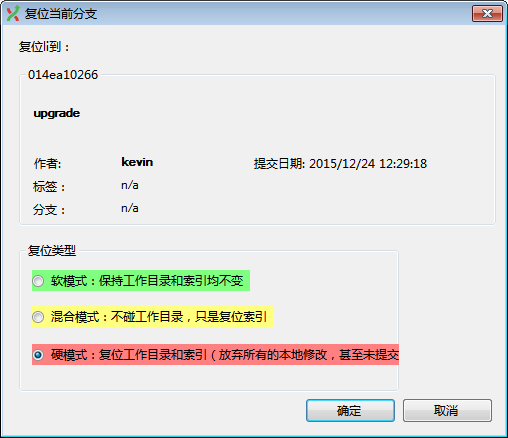


# 特殊操作

## 回退以前版本

复位只在本地修改了，注意要推送到远程

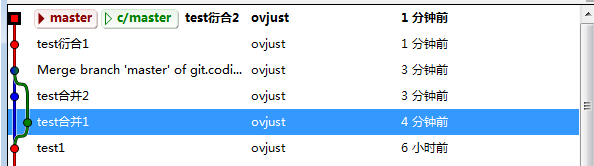




## 推送冲突的解决

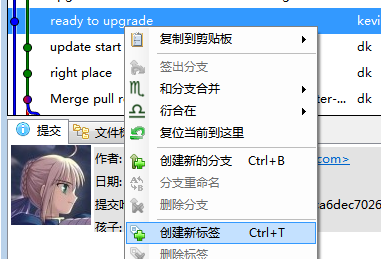


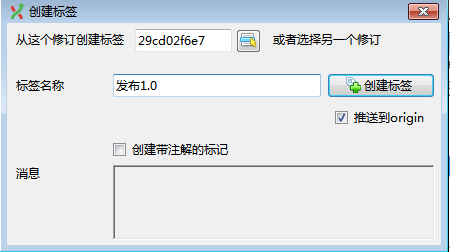
一般推荐“合并拉取”

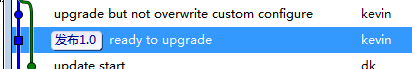


## 创建标签

一般用于版本标记，允许打包下载等。注意最后要推送。







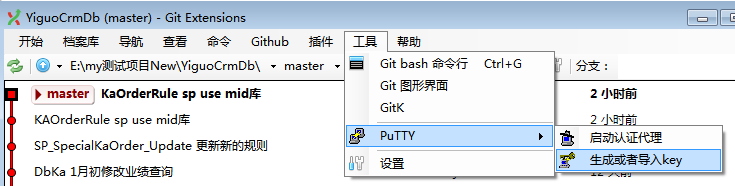
## 生成SSH（RSA）密钥（安全外壳协议）

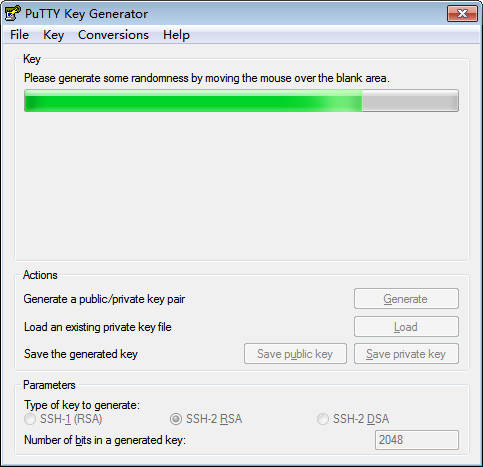
SSH 为建立在应用层和传输层基础上的安全协议。SSH 是目前较可靠，专为[远程登录](http://baike.baidu.com/view/59099.htm)会话和其他网络服务提供安全性的协议。利用 SSH 协议可以有效防止远程管理过程中的信息泄露问题。

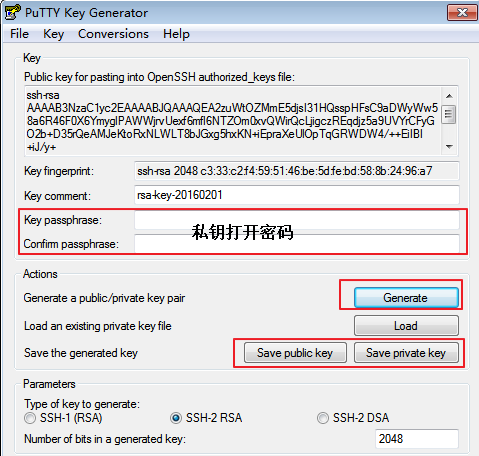
SSH公钥、私钥的概念，公钥给对方

加密（原文，公钥）、解密（私钥），账号登录

签名（取哈希，私钥）、验证（取哈希，签名，公钥），私钥登录，推送代码，保证信息来源和完整。

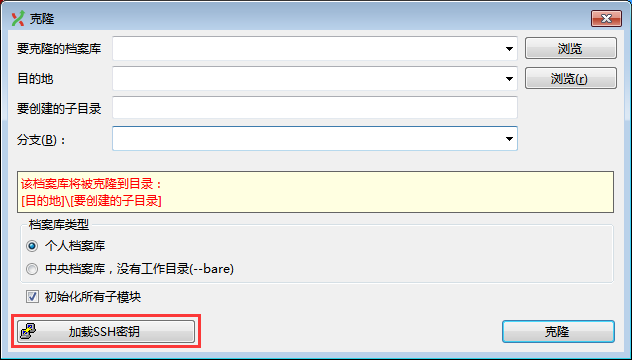


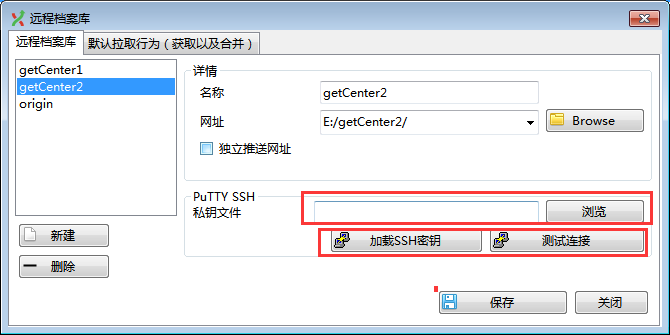




网上让填写公钥时，有的要格式“ssh-rsa AAAAB3NzaC...",而有的要“ssh-rsa \n AAAAB3NzaC..."。

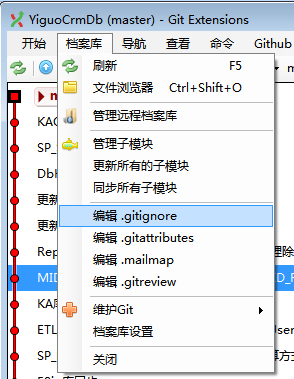
密钥的使用：

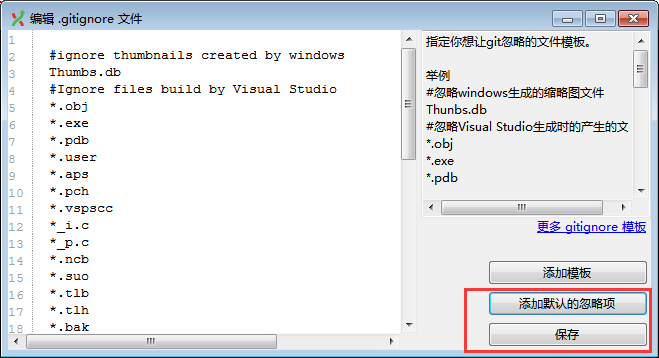




在C#中也可以实现rsa生成密钥（是xml格式）、加解密、签名-验证。

## ignore





gitHub里有更好的模板，ignore文件还可以加在每个子目录里，作用域同Web.config

ReadMe.dm、**授权许可**

**ignore示例：**

\*.suo

# Build results

[Dd]ebug/

[Rr]elease/

x64/

build/

[Bb]in/

[Oo]bj/

#MsOffice临时文件

~$\*

## Git命令行

Windows命令行（C:\Program Files (x86)\Git\bin\加入环境变量），gitHub命令行工具，Cygwin（Windows里的Unix环境）

**git init** [--bare]

**git clone** git://git.kernel.org/pub/scm/  [project\_path]

创建目录mkdir aaa 转到目录cd aaa

创建文本文件 echo "HTML files generated by VimWiki" > README.txt

**添加新文件**：git add dir/files

**提交**：git commit -q -m "my blog site"

**推送**：git push orig

### 导出一套纯代码 git archive

git archive v0.1 | gzip > site.tgz

git archive –format zip -o site-$(git log –pretty=format:”%h” -1).zip HEAD

site.gz site-87424a7.zip

git archive --format zip --output "./output.zip" master -0

不压缩：git archive master | tar -x -C /somewhere/else

### 设置用户的姓名和邮件，使用以下命令

git config --global user.name "name"

git config --global user.email "email"

### “个人档案库”转成“中央档案库”

1. **git clone -bare git://git.kernel.org/pub/scm/  [project\_path]**
2. **git fetch --all**

**拷出.get目录下内容，改config** core.bare=true

git config --bool core.bare true

### 多地址推送

编辑本地仓库的.git/config文件即可达到目的。

|  |  |
| --- | --- |
|  | [remote "all"]      url = git@github.com:luoshupeng/commonuseppa.git      url = git@gitcafe.com:chinesedragon/commonuseppa.git      url = chinesedragon@gitcd.com:/commonuseppa |

这样，使用git push all即可一键Push到多个远程仓库中。

### 权限验证

https： 用户名、密码，可以在当前档案库.git\config中设置记住密码[credential] helper = [store](http://cpro.baidu.com/cpro/ui/uijs.php?adclass=0&app_id=0&c=news&cf=1001&ch=0&di=128&fv=19&is_app=0&jk=20688eaf2ba64ad3&k=store&k0=store&kdi0=0&luki=2&mcpm=0&n=10&p=baidu&q=smileking_cpr&rb=0&rs=1&seller_id=1&sid=d34aa62baf8e6820&ssp2=1&stid=9&t=tpclicked3_hc&td=1682280&tu=u1682280&u=http://www.th7.cn/system/win/201511/142298.shtml&urlid=0)

git@。。。： 是使用SSH私钥

git命令行，不识别PuTTY生成的私钥，且要求私钥在特定的位置。

可能需要设置set HOME=C:\Users\当前用户

ssh-keygen -t rsa -C "ovjust@gmail.com"，生成后自动保存在C:\Users\当前用户\.ssh\id\_rsa、id\_rsa.pub

PuTTY可以将ssh-keygen生成私钥加载，再另存为它自己的格式。

### 让ssh-agent管理密钥

目的：加载非特定位置的密钥、多个密钥、保存密钥的密码。

$ ssh-agent.exe -s

SSH\_AUTH\_SOCK=/tmp/ssh-TWD7NYSMxENy/agent.4120; export SSH\_AUTH\_SOCK;

SSH\_AGENT\_PID=5948; export SSH\_AGENT\_PID;

echo Agent pid 5948;

$ ssh-agent.exe bash

ssh-add.exe ~/.ssh/id\_rsa

# Enter passphrase for /home/ganiks.liu/.ssh/id\_rsa:

Identity added: /home/ganiks.liu/.ssh/id\_rsa (/home/ganiks.liu/.ssh/id\_rsa)

git bash和[putty](http://files.cnblogs.com/mumutou/PuTTY.rar)集成

* 1. 打开puttygen.exe软件,点击load加载一个private key,因为我的private key并不是puttygen产生的，所以需要进行一个转换，加载完成之后，点击save private key，保存为tangk.ppk,这个是pageant.exe承认的key。
  2. 打开pageant.exe,在托盘下面会看到一个图标，点击add key，加载刚才生成的tangke.ppk文件。
  3. 设置一个环境变量GIT\_SSH,设置为plink.exe的全路径地址。
  4. 直接进入git bash,git pull, git push都可以操作了。

### Bundle、压缩档案库

Bundle文件是在packfile文件的基础上增加了代码库的元信息。通俗的说bundle文件就是一个便携版的代码库，里面可以包含代码库所必须包含的信息。

服务器端：

git bundle create repo.bundle master

git bundle create repo.**bundle** master~10..master

客户端：

git clone repo.bundle

git fetch repo.bundle

档案库太臃肿了，可以用命令行清理较久远的信息。

# 扩展

## 源代码管理平台

github 私有项目收费，第三方扩展支持

coding.net，多种语言代码发布演示

gitbook.com

服务器端：

OpenSSH for Windows: CopSSH(Copssh\_5.X.4\_Installer.exe)/Gitblit

平台：

GitLab、Gogs（Go语言）、GitBlit（java环境）、GitStack（一键安装）、GitCandy（C#）等仿gitHub项目管理平台搭建。

### Gogs、GitBlit

Gogs也有OpenShift安装项目。





## git服务端的形式

1. 共享的文件夹
2. ssh、https等通讯服务
3. 仿gitHub源代码管理平台

## 中央档案库的初始化方式

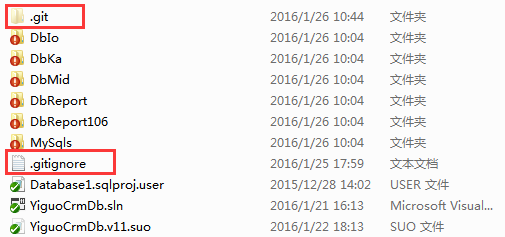
1. 从其它中央档案库导入
2. 创建未初始化的中央档案库，首次使用个人档案库推送。
3. 创建中央档案库，并初始化为空的，克隆生成个人档案库。

## git实现网站自动发布的形式

### git推送发布的原理

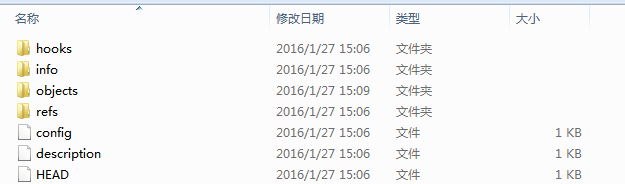
中央档案库-裸，没有显式的一套源码，

个人档案库-非裸



中央档案库-裸

文件目录同上图中的".git"文件夹



中央档案库master分支被推送时，触发发布环境的个人档案库git拉取行为。

### 手动pull

### hook checkout

服务端修改hooks/post-receive，手动修改或在cmd中修改

$ cat > hooks/post-receive

。。。。。。。ctrl+D退出编辑

在其中写入要执行的命令：

#!/bin/bash

git archive master | tar -x -C e:\gitCenter1Out

或

cd /gitTest

git pull

### webHook url pull

在网页中执行cmd命令“d: && cd git1\pub && git pull”。

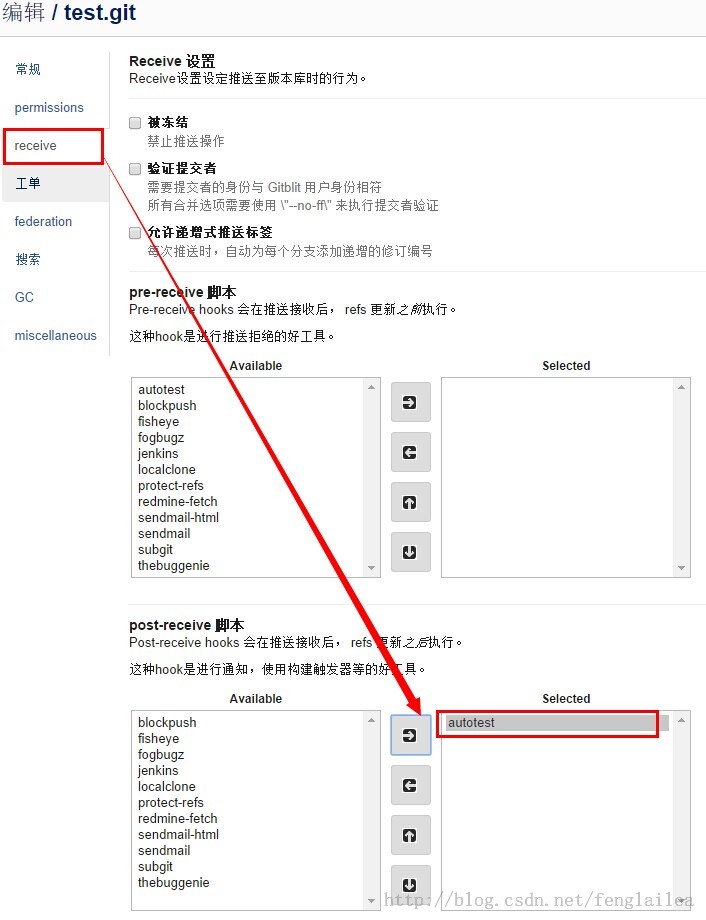
php： exec("git.exe pull", $output);

注意点：

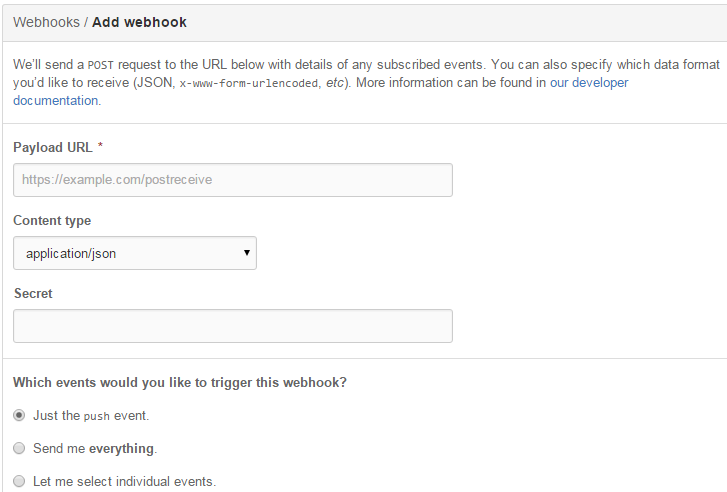
使用恰当的计算机用户执行。

**gitblit hook**

gitblit 所有钩子都放在这个目录 /gitblit/[data](http://cpro.baidu.com/cpro/ui/uijs.php?adclass=0&app_id=0&c=news&cf=1001&ch=0&di=128&fv=19&is_app=0&jk=a12e8bd2e627af7&k=data&k0=data&kdi0=0&luki=2&mcpm=0&n=10&p=baidu&q=smileking_cpr&rb=0&rs=1&seller_id=1&sid=f77a622ebde8120a&ssp2=1&stid=9&t=tpclicked3_hc&td=1682280&tu=u1682280&u=http%3A%2F%2Fwww%2Eth7%2Ecn%2Fsystem%2Fwin%2F201409%2F70022%2Eshtml&urlid=0)/groovy  
把localclone.groovy复制另存为 autotest.troovy



**gitHub WebHook**



## 注意事项

允许多人同时修改一个文件，但注意解决冲突时的风险，应适当避免风险。

vs等项目中增删文件、修改可能多人使用的文件时，应先获取最新，处理后尽快提交推送。

权限控制，避免可以推送到别人的分支。

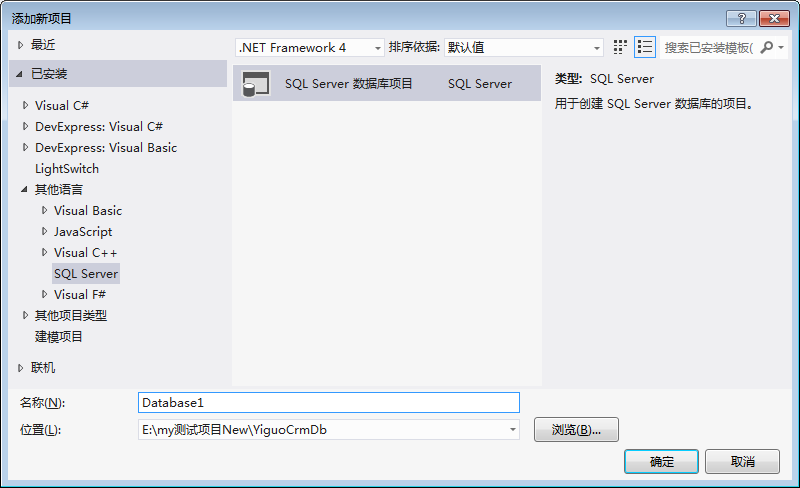
由于经常切换分支，注意不要误在公共或别人分支上直接编辑。

在个人分支上合并好，能成功生成再合并提交。

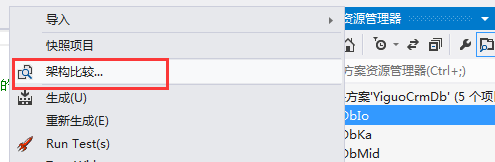
如果你有两个自己的分支，经常在这两个分支间切换工作，那建议：本地保留两份个人档案库，分别处理一个分支。

## 数据库脚本版本管理

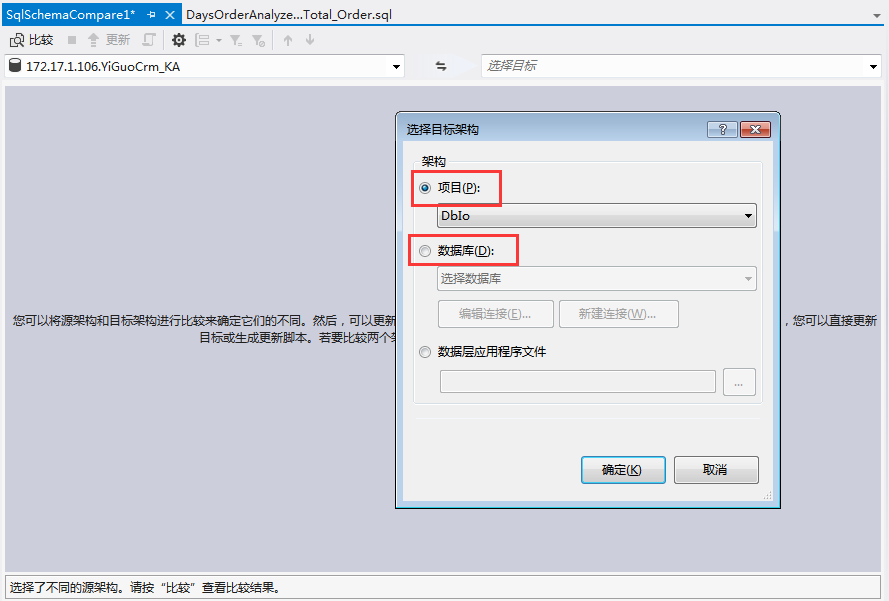
vs 数据库项目



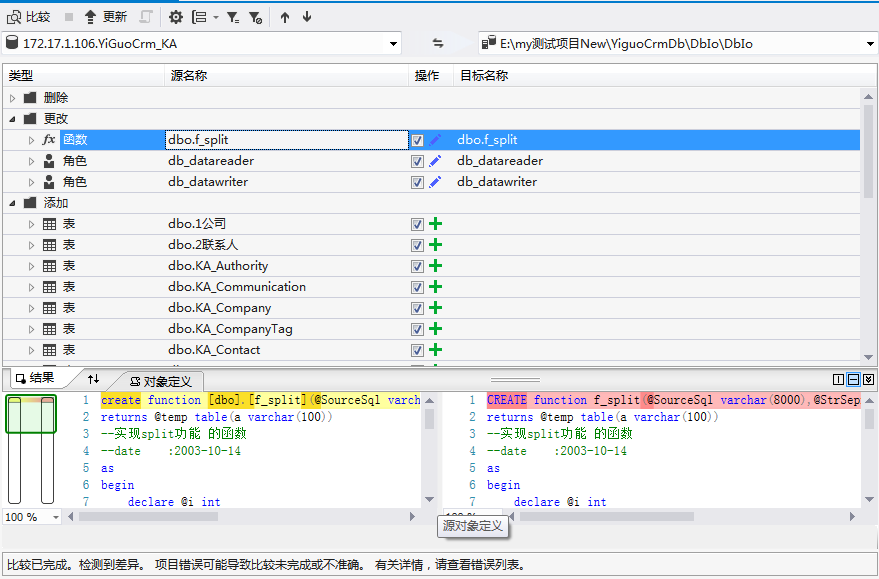
在项目上右键架构比较



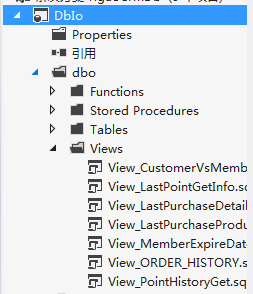
数据库与数据库、数据库与项目之间都可以比较



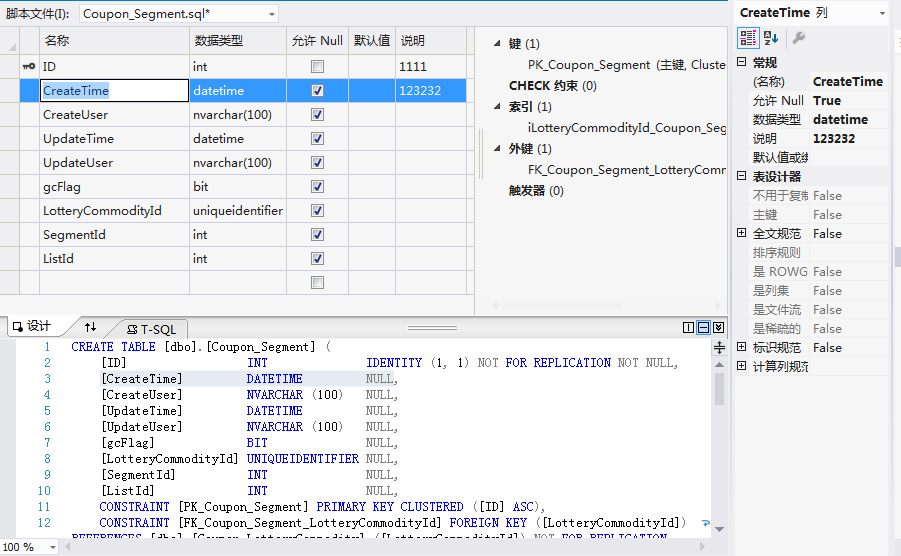
比较结果



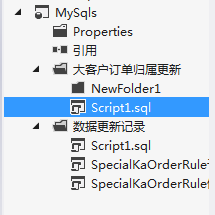
更新后



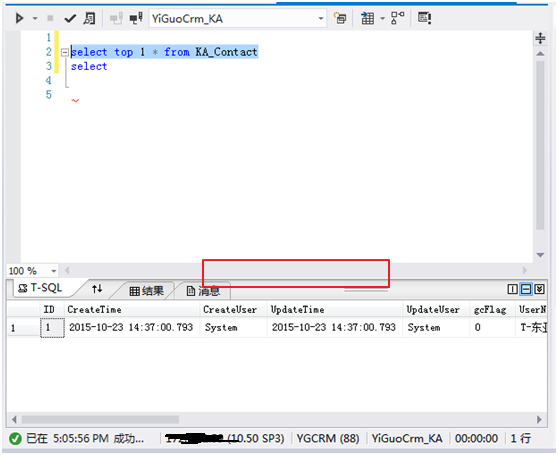
vs里修改表结构等，理想情况可以反向更新到数据库



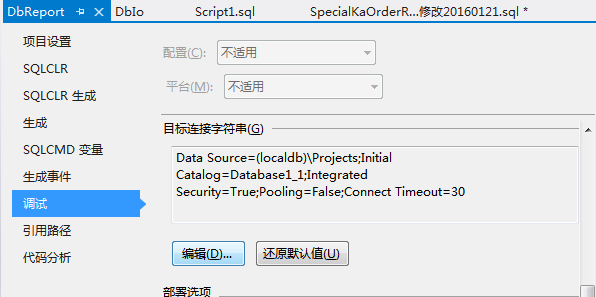
建一个空的SqlServer项目，可以分类整理自己的脚本、数据处理笔记

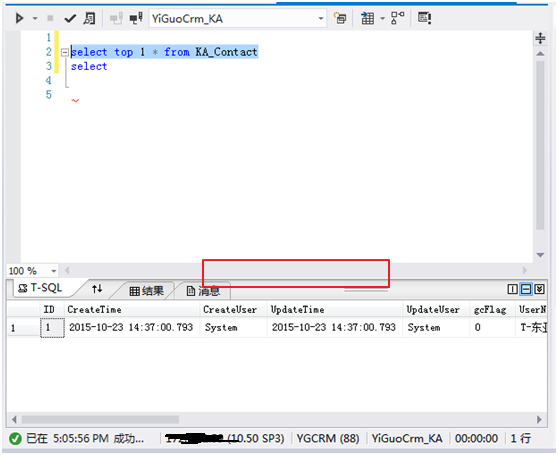


项目里就可以执行脚本



项目属性里，可以设置默认的数据库连接，上图中断开后，可以再连接别的库。





配合git，你现在拥有了一套完善的脚本收集、版本管理、架构比较、快速执行的工具。

## 免费空间openshift

红帽openshift.com 使用git做版本发布控制，支持多种开发语言 多种数据库，三个免费1g空间，可绑域名，可交费升级，WebSocket，https，测试外网可访站点

注册时的google验证码、自动分配的二级域名打开时，需要翻墙

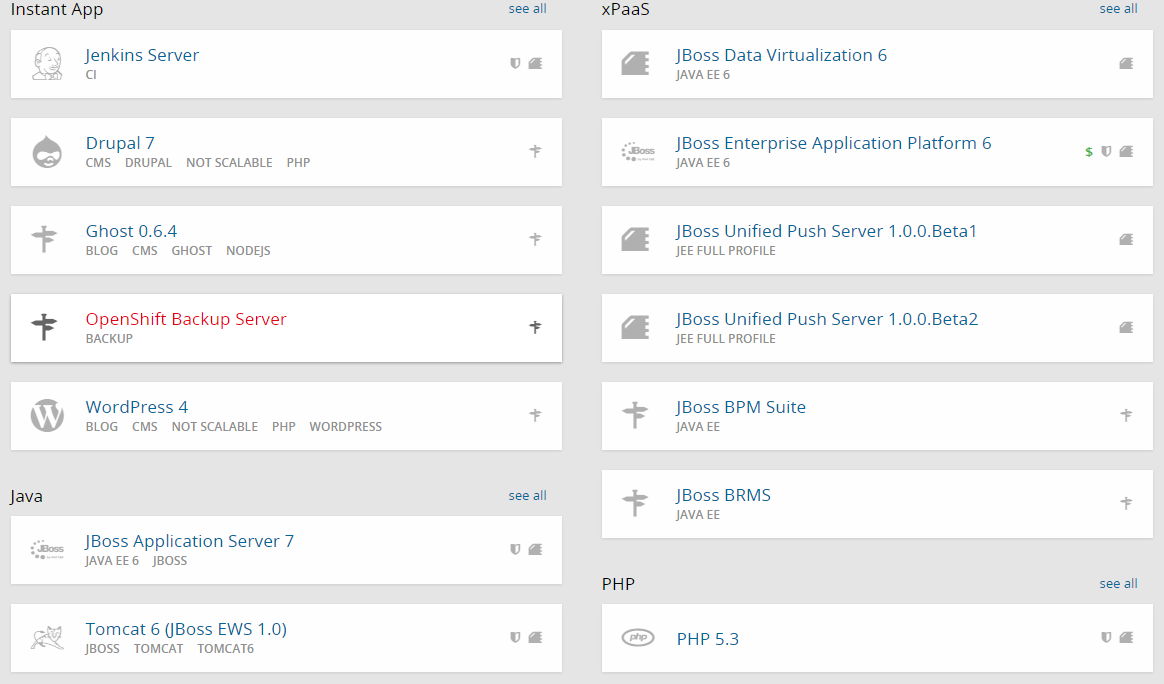
rhc命令行管理工具，Xshell+SSH验证或配合免费空间，可以做翻墙用途

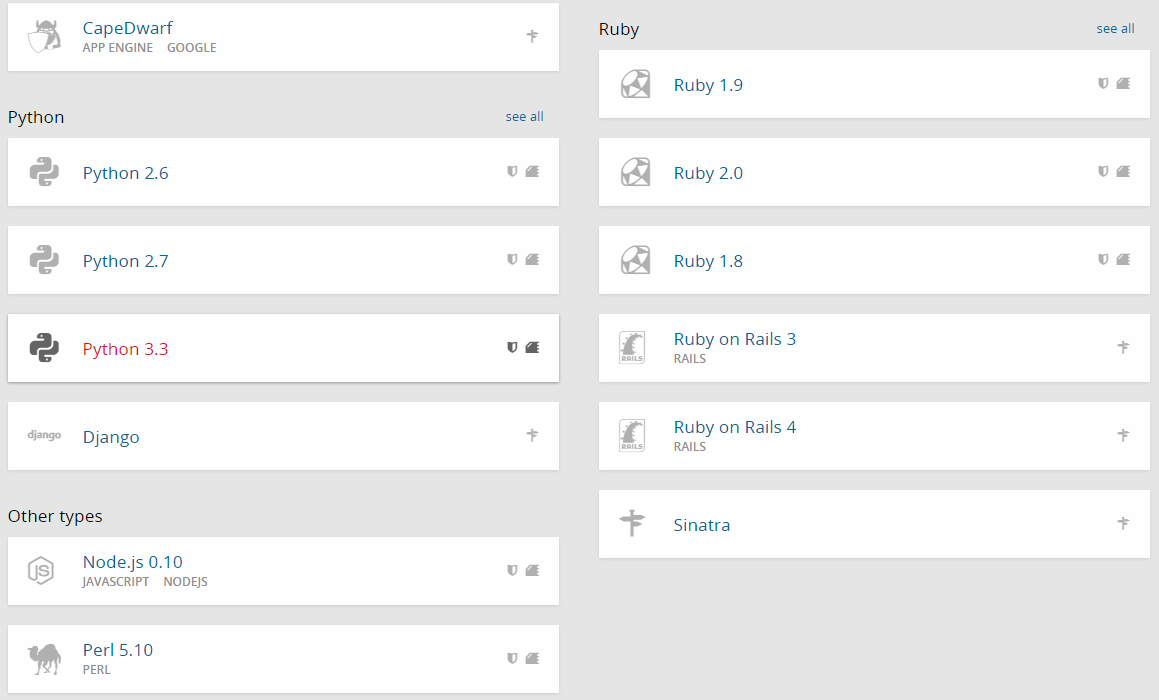
Databases

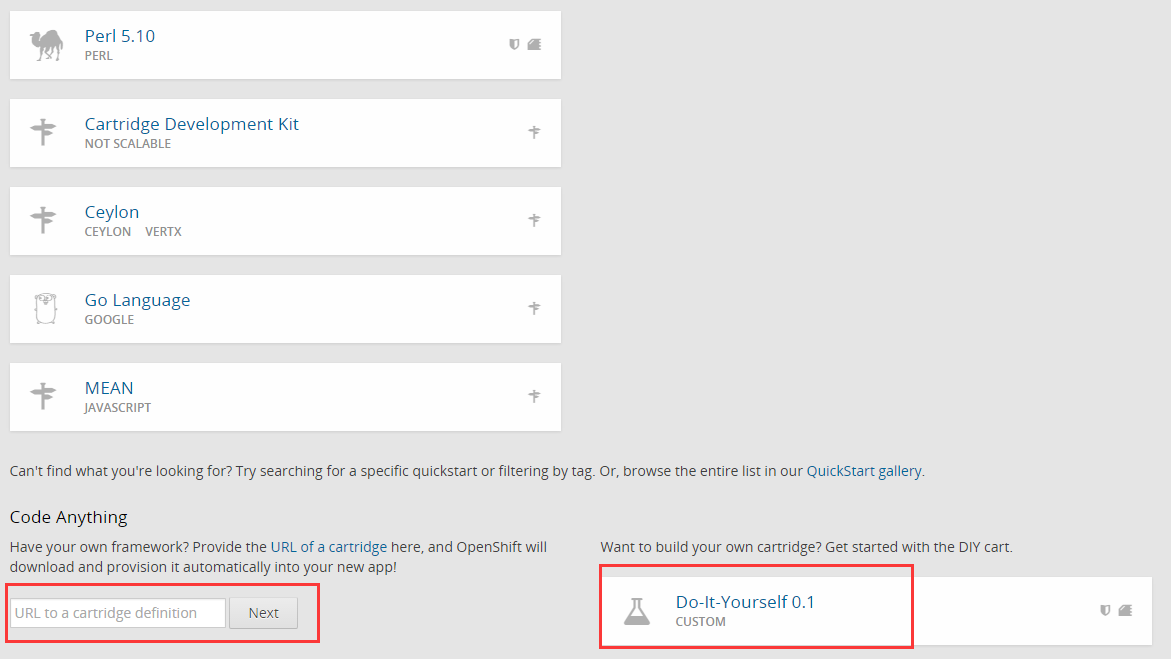
[Add MongoDB 2.4](https://openshift.redhat.com/app/console/application/5576c21be0b8cd2af0000024-pinche/cartridge_type/mongodb-2.4?direct=true)

[Add MySQL 5.5](https://openshift.redhat.com/app/console/application/5576c21be0b8cd2af0000024-pinche/cartridge_type/mysql-5.5?direct=true)

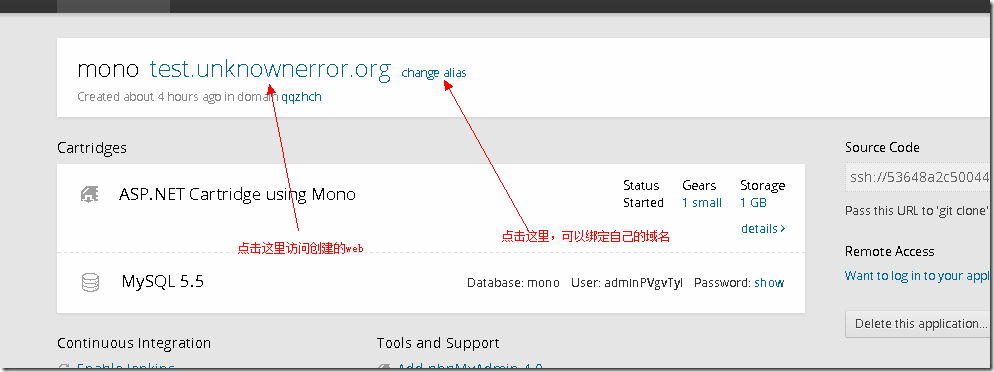
[Add PostgreSQL 9.2](https://openshift.redhat.com/app/console/application/5576c21be0b8cd2af0000024-pinche/cartridge_type/postgresql-9.2?direct=true)







开源项目支持C#，WebForm、MVC、.Net3.5、4.0都支持  
https://raw.github.com/wshearn/openshift-community-cartridge-mono/master/metadata/manifest.yml

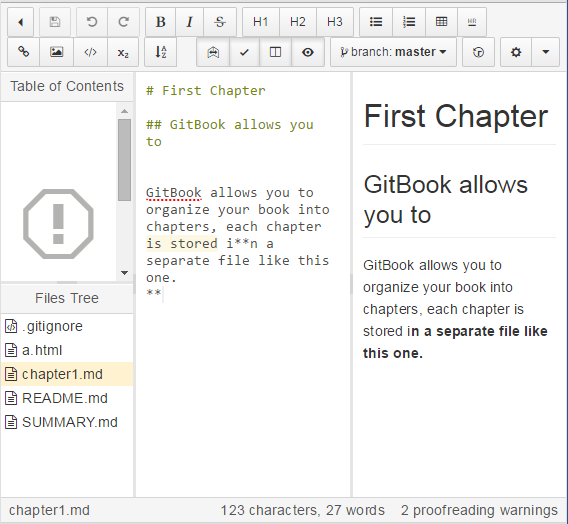


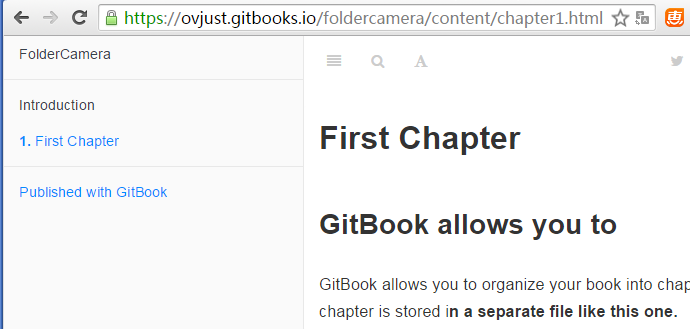
## gitBook

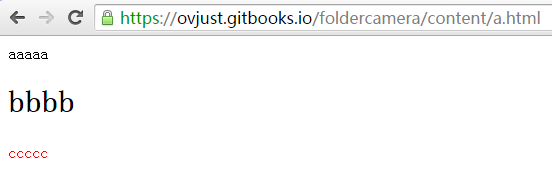
编辑模式、预览模式，支持在别的网页引用、普通Html页、导出pdf。

全称：GitHub Flavored Markdown，可以用CSS控制整体样式。

优点：写好了使用说明文档，提交到主分支，别人直接就能看到，免除要分享的麻烦。



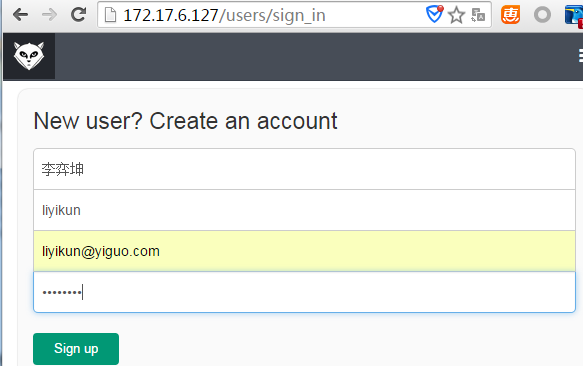




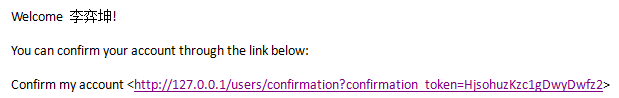
# Yiguo-GitLab

## 注册

http://172.17.6.127/

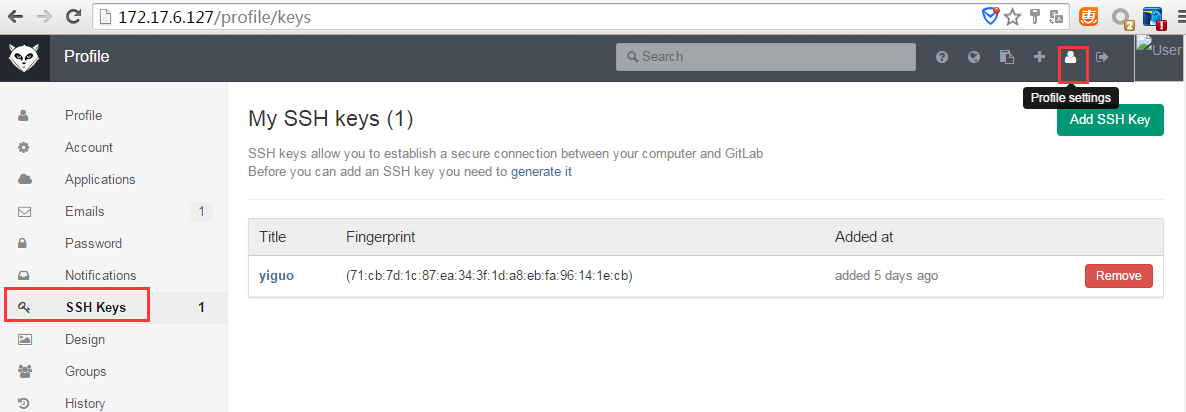


邮箱激活（垃圾邮件），以后项目的变动会收到邮件。把激活链接中的“127.0.0.1”替换成“172.17.6.127”。

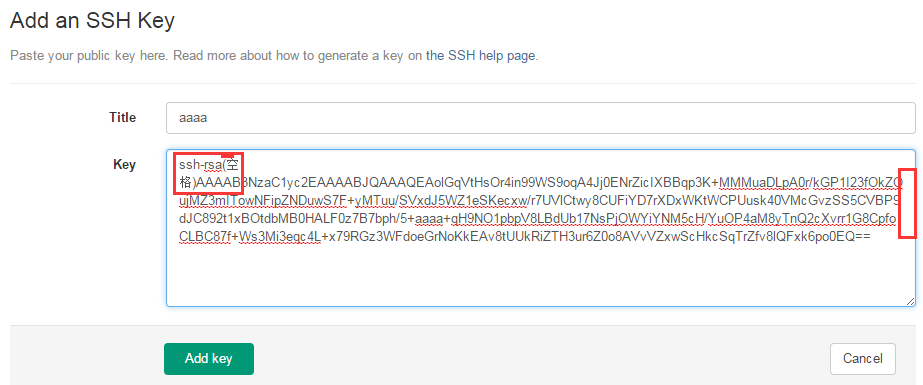


## 添加SSH Key

每个人的要不同



ssh-rsa(空格)AAAAAAAAAA。。。，中间有一个空格，各行不要回车。



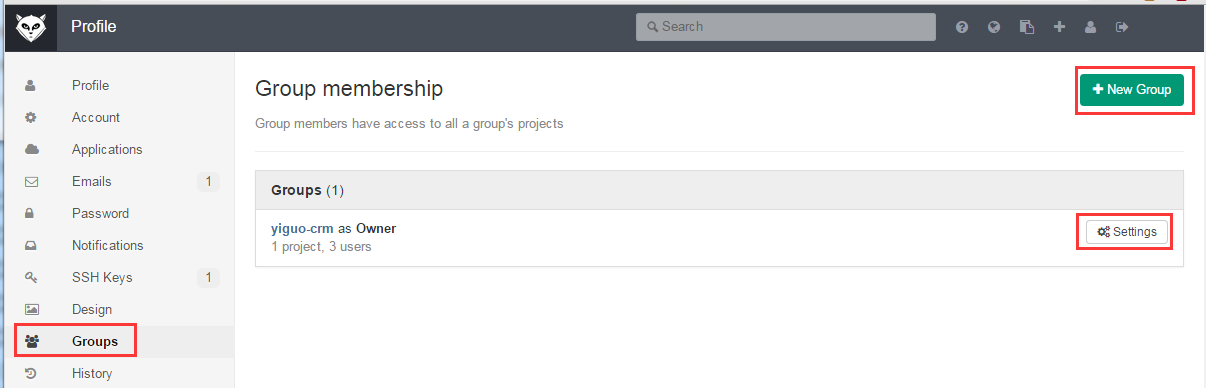
回首页

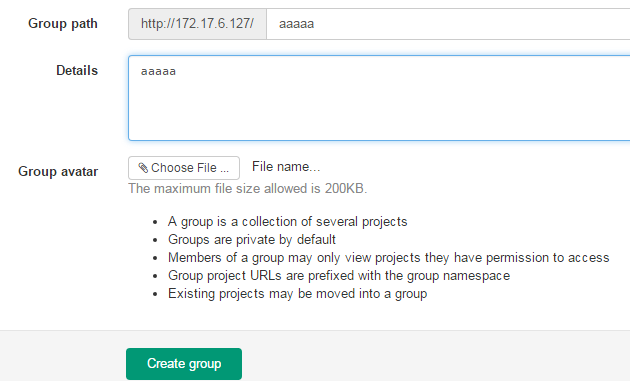


## 群组

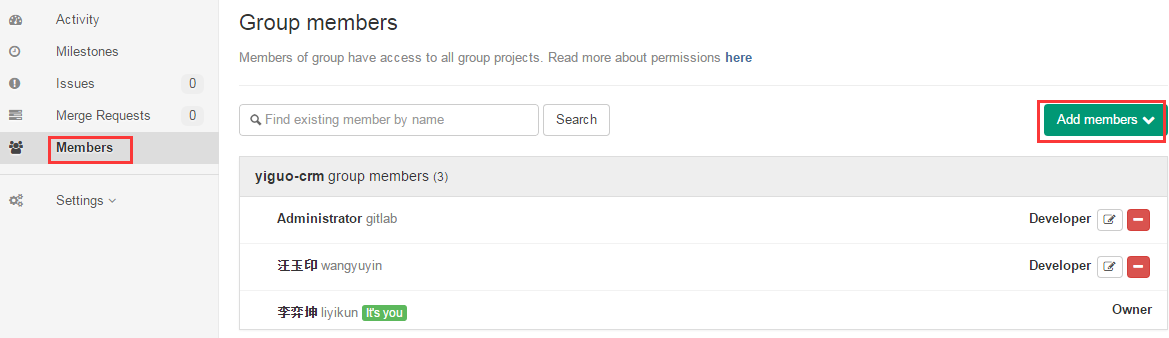
两个入口



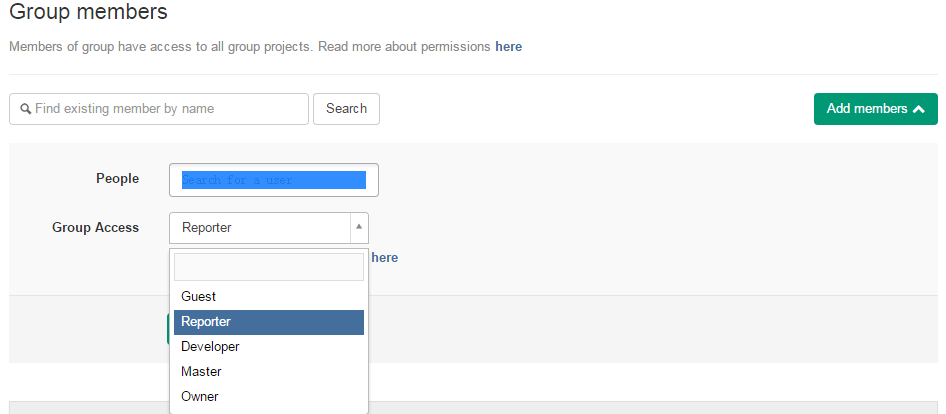




组员管理

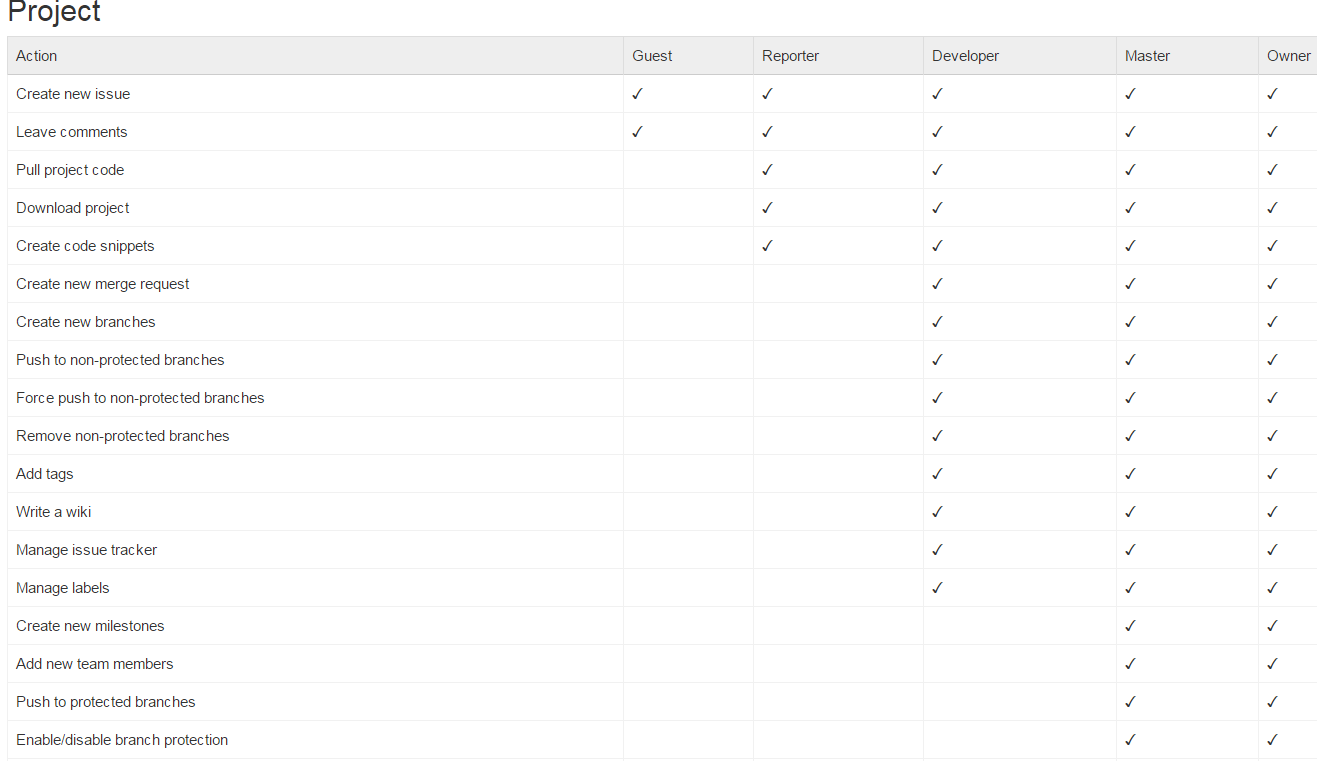
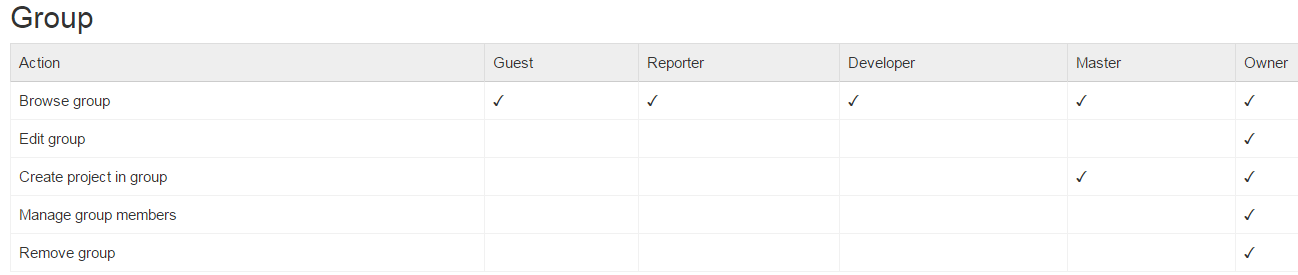


添加组员，不建议群组太多。可以指定多个Owner。



GitLib有五种身份权限，分别是：

* Owner [项目](http://www.07net01.com/tags-%E9%A1%B9%E7%9B%AE-0.html)所有者，拥有所有的操作权限
* Master 项目的管理者，除更改、删除项目元信息外其它操作均可
* Developer 项目的开发人员，做一些开发工作，对受保护内容无权限
* Reporter 项目的报告者，只有项目的读权限，可以创建代码片断
* Guest 项目的游客，只能提交问题和评论内容

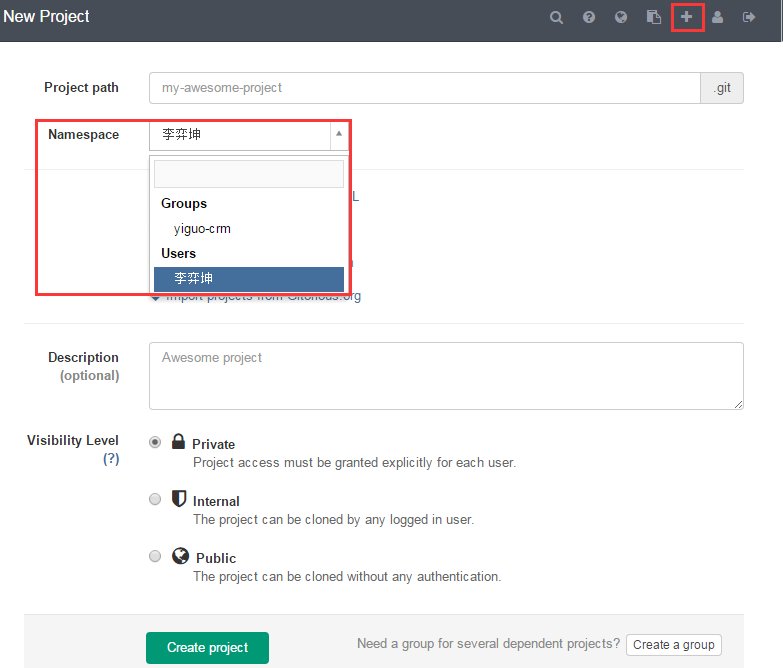
 

## 创建项目

添加群组项目



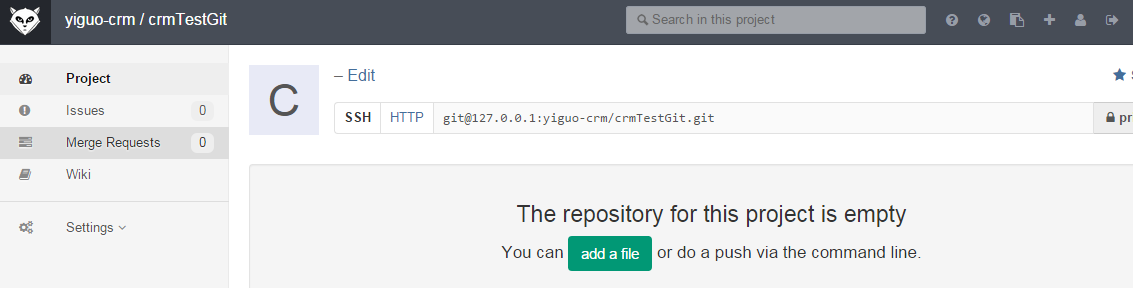
个人或群组项目



项目列表

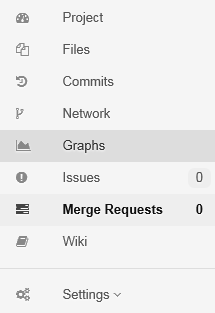


新建的项目首页



1. 本地初始化一个项目，推送到远程。
2. “add a file”,完成服务端档案库初始化。
3. 其他人“克隆档案库”。

单个项目的菜单



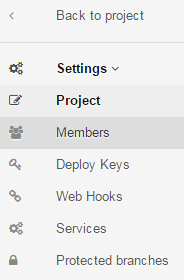
贡献代码，pushRequest



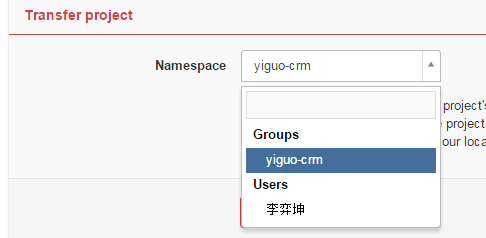
MergeRequests

## 项目设置

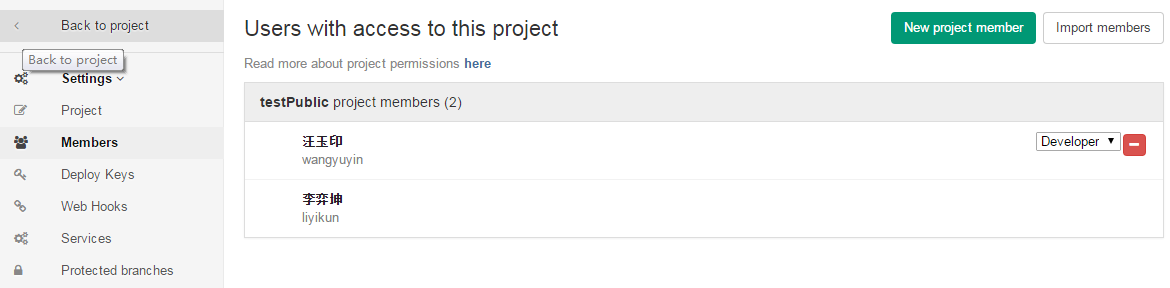
单个项目的设置菜单



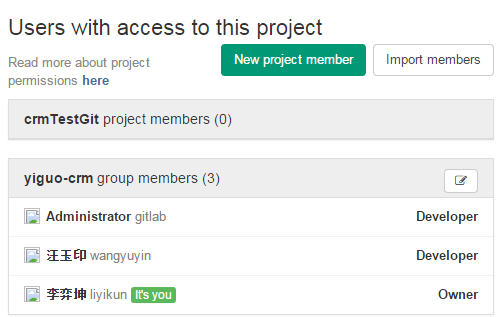
项目属性修改，改所属的群组



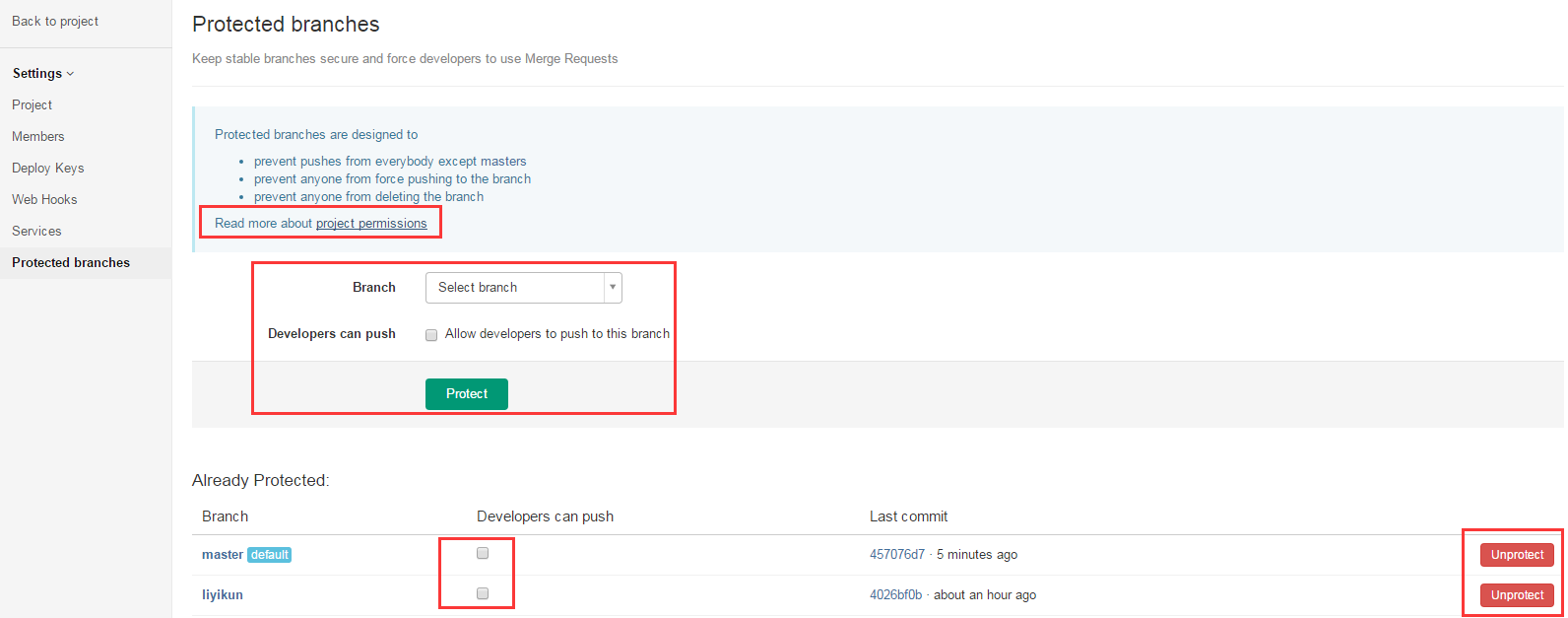
加项目Member

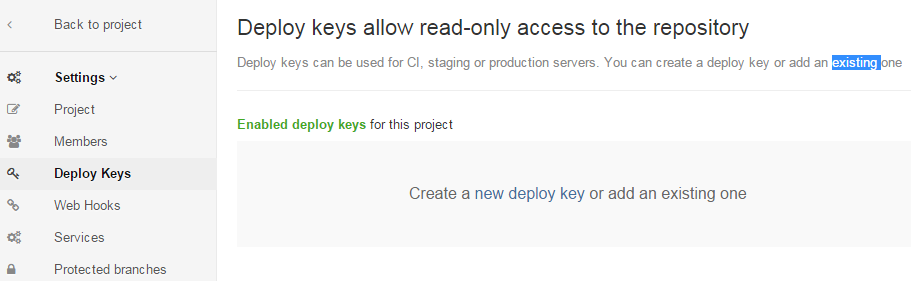


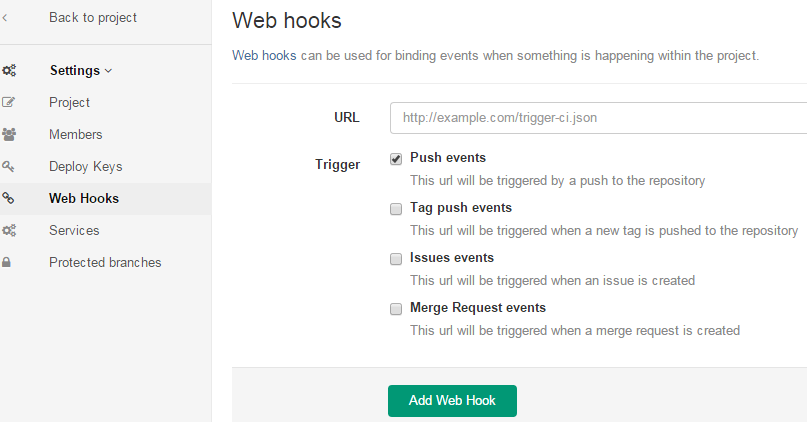
群组项目也可以再加项目Member



Protected branches







## git分支管理示例



# 分享资料

本文档、安装文件、ignore示例文件。