神勇小白鼠的博客

使用git管理github项目

中文教程

http://progit.org/book/zh/

Git是一个分布式的版本控制系统，最初由Linus Torvalds编写，用作Linux内核代码的管理。在推出后，Git在其它项目中也取得了很大成功，尤其是在Ruby社区中。目前，包括Rubinius和Merb在内的很多知名项目都使用了Git。Git同样可以被诸如Capistrano和Vlad the Deployer这样的部署工具所使用。

目前大部分的版本控制都需要一個服务器端，commit时提交到服务器端。git虽然是分布式的管理工具，但它也可以使用集中管理的方式。举例来说，billy创建了一个新项目，tom觉得这个项目比较有意思，他想玩玩看，他可以把整个repo(reoisitories)都clone到本地，并且在他clone下来的项目中自己有一个repo。tom可以自由的对这个repo做各种提交修改，直到他满意为止。这时tom可以告诉billy，他改了一些什么，若billy觉得不错，可以从tom那边fetch并merge到本地。

如前所述，作为一个分布式的版本控制系统，在Git中并不存在主库这样的概念，每一份复制出的库都可以独立使用，任何两个库之间的不一致之处都可以进行合并。正因为如此，所以有了Github这个网站，github是基于ruby的并揉合多种语言进行开发的，上面的ruby项目也非常活跃。在github 中，每个人都可以有多个repo，这些repo都是与用户绑定在一起的。user之间可以互相clone repo、fork repo、watch repo或是 follow其他user，就好像twitter或是plurk等社交网站一样。

不过若使用免费的github则所有的repo都要是public且有300M容量的限制。若希望有private的repo，则需要跟付费才可。github另外还有剪贴簿的功能，可将代码贴到其中并自由的发展各种branch。

GitHub可以托管各种git库，并提供一个web界面，但与其它像 SourceForge或Google Code这样的服务不同，GitHub的独特卖点在于从另外一个项目进行分支的简易性。为一个项目贡献代码非常简单：首先点击项目站点的“fork”的按 钮，然后将代码检出并将修改加入到刚才分出的代码库中，最后通过内建的“pull request”机制向项目负责人申请代码合并。已经有人将GitHub称为代码玩家的MySpace：

> 在GitHub进行分支就像在Myspace（或Facebook [...]）进行交友一样，在社会关系图的节点中不断的连线。

GitHub项目本身自然而然的也在GitHub上就行托管，只不过在一个私有的，公共视图不可见的库中。开源项目可以免费托管，但私有库则并不如此。Chris Wanstrath，GitHub的开发者之一，肯定了通过付费的私有库来在财务上支持免费库的托管这一计划。

> 是的，我们正是这么计划的。通过与客户的接洽，开发FamSpam，甚至是开发GitHub本身，GitHub的私有库已经被证明了物有所值。任何希望节省时间并希望和团队其它成员一样远离页面频繁转换之苦的人士都会从GitHub中获得他们真正想要的价值。

Chris Wanstrath还向InfoQ分享了关于GitHub的一些内幕信息：

> GitHub主要用Rails实现。我们目前在进行的post-commit集成小应用完全使用Merb编写。我们使用了Python的Pygments来做格式高亮显示，另外还用了Ara T. Howard's Bj加上一些Ruby脚本来做我们的排队系统。当然，我们用了Ruby Grit库来和Git进行交互。

GitHub已经有了一组引人注目的特性，除了命令式的库浏览器和一个项目Wik，GitHub甚至还包括了一个GitHub gem，以使通过shell方式使用GitHub更为方便。更多的未来特性已经在计划中：

> 许多人都希望能有一个条目系统，因此一个简单的条目系统已经在开发中。此外，正如我前面所言，我们尚在进行RubyGems服务器和一些之前留出的post-commit钩子方面的工作。如果你不能或就是不想托管一个你自己的守护进程，你可以使用我们所提供的。
>
> 我们还在开发一些特性来帮助公司在使用Github时可以停留在sync之上。
>
> 最后，我们也在进行API发布方面的工作。我们很快就会发布一些只读性的API，随后是一些很强大的“写”集成。你可以使用API将新的事件发布到新闻feed中，发消息和做其他许多很酷的事情。

最新博文
- 在shell脚本中调用另一个脚本的三种不同方法(fork, exec, source)
- wget 使用技巧
- Linux 启动顺序
- debian 6（squeeze）源列表
- linux中的定时任务crontab
- nginx虚拟目录
- Python 各进制间的转换
- 解决TNSLSNR占用8080端口
- 如何彻底的删除Oracle表
- PAC Manager: Ubuntu 上强大的 SSH 帐号管理工具，可取代 SecureCRT

查看所有博文»

相关博文
- Ubuntu9.10安装Git，Git和Github初次使用
- GIT分支管理是一门艺术
- GIT学习笔记2--GIT的优势
- GIT学习笔记1--基本使用
- git autocrlf
- Git一些小经验
- 在Windows环境中使用版本管理工具Git
- 如何安装 Vim 的 Erlang 差件 vimerl（github，debian）
- 《Git权威指南》
- 研发管理系统选型必读

搜索更多»

更多关于GitHub的信息可以参见GitHub官方网站或GitHub博客。目前通过GitHub进行代码管理的开源项目列表也已经可以查阅。

下面先來介紹如何将环境配置好：配置环境分为三个步骤，安装与设置git，安装与设置ssh-key，以及在github上注册并建立repo。

安装与设置git
Linux:只要用yum，apt-get等安装即可，或是下载之后编译安装。
Mac OS X:从这里下载并安装。
Windows:先安装putty，然后从这里下载并安装。
安装与设置ssh-key
git使用ssh tunnel来提交源码，这个加密通道可以避免源码的封包被拦截而截取。因此要先产生并上传ssh key到github，方便之後与服务器之间的通迅。
Mac OS X与Linux，只要输入ssh-keygen -t rsa并根据指示操作即可：

```
[~/.ssh]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/tom/.ssh/id_rsa): <enter>
Enter passphrase (empty for no passphrase): <输入key的密碼，或直接按下enter使用空密码>
Enter same passphrase again: <再输入一次密码>
Your identification has been saved in /home/tom/.ssh/id_rsa.
Your public key has been saved in /home/tom/.ssh/id_rsa.pub.
The key fingerprint is:
50:43:77:c6:97:af:61:82:dc:ea:9b:6b:67:d4:1b:61 tom@volcano
```

其中id_rsa.pub是公钥，而id_rsa则是私钥，请妥善保存以免遺失，它们都存放于~/.ssh目录中。将公钥粘贴到你github帐号中的SSH Public Keys的位置。注意小心不要复制到空格。

Windows，执行git-bash并输入：

```
ssh-keygen -C “username@email.com” -t rsa
```

github注册
在github你的帐号右上角可以看到一个Your Repositories，选择Create one。输入Project name后，可以看到它有教你如何创建一个新的项目。

这里以一个名为test的项目为例：

```
# Global setup:

# Download and install Git
git config --global user.email <your email>
git config --global user.name <your name>

# Next steps:
mkdir test # cd test
git init
# github会自动读取你的README内容并显示在项目简介中，因此先创建README
touch README
# 将README加到index中
git add README
 # 提交到版本库中
git commit -m 'first commit'
# 把github的repo加入为远程的repo
git remote add origin git@github.com:<你的ID>/test.git
# 把目前的commit状态push并同步到github上面
git push origin master

# Existing Git Repo?
cd existing_git_repo
git remote add origin git@github.com:<你的ID>/test.git
git push origin master
```

把你的程序push上github后就可以有自己的repo了。

若您喜欢别人的repo，只要他是public的，就可以点击上面的fork，把整个repo复制到你的帐户底下并修改提交后再请求原作者进行pull。

若你fork了一个repo(名为test)到github中之后，只要输入：

```
git clone git@github.com:<ID>/test.git test
```

就可以将整个repo复制下来并存放在test目录中。若你的ssh key有加上密码保护，每次与github通信的过程中都需要输入密码以存取ssh key。

git的简单使用：

```
# 创建一个版本库
git init
# 每次修改好了后，可以先将修改存入stage(快照/索引)中
git add <modified files>
# 修改了大量文件则使用下面这个命令批量存入
git add .
# 使用commit将快照/索引中的内容提交到版本库中
git commit -m "msg"
# 也可以将git add与git commit用一个指令完成
git commit -a -m "msg"
# 将本地的git档案与github(远程)上的同步
git push
# 将github(远程)的git档案与本地的同步(即更新本地端的repo)
git pull
# 例如,pull指令其实包含了fetch(将变更复制回来)以及merge(合并)操作
git pull git://github.com/tom/test.git

# 另外版本控制系统的branch功能也很有意思，若同时修改bug，又要加入新功能，可以fork出一个branch：
一个专门修bug，一个专门加入新功能，等到稳定后再merge合并
git branch bug_fix # 建立branch，名为bug_fix
git checkout bug_fix # 切换到bug_fix
git checkout master #切换到主要的repo
git merge bug_fix #把bug_fix这个branch和现在的branch合并

# 若有remote的branch，想要查看并checkout
git branch -r # 查看远程branch
git checkout -b bug_fix_local bug_fix_remote #把本地端切换为远程的bug_fix_remote
branch并命名为bug_fix_local

# 还有其它可以查看repo状态的工具
git log #可以查看每次commit的改变
git diff #可以查看最近一次改变的内容，加上参数可以看其它的改变并互相比较
git show #可以看某次的变更

# 若想知道目前工作树的状态，可以输入
git status
```

上篇 (5个月前)：如何删除bdump下的.trc文件
下篇 (5个月前)：MySQL 'Access denied for user 'root'@'localhost' 的问题
原文地址：http://linglong117.blog.163.com/blog/static/277145472010926583310/

发表评论 文明上网，理性发言

回到顶部 回到评论列表