Gitolite

本节作为Gitolite的一个快速指南，指导基本的安装和设置。不能完全替代随Gitolite自带的大量文档。而且可能会随时改变本节内容，因此你也许想看看最新的[版本](http://sitaramc.github.com/gitolite/progit.html)。

Gitolite是在Git之上的一个授权层，依托sshd或者httpd来进行认证。（概括：认证是确定用户是谁，授权是决定该用户是否被允许做他想做的事情）。

Gitolite允许你定义访问许可而不只作用于仓库，而同样于仓库中的每个branch和tag name。你可以定义确切的人(或一组人)只能push特定的"refs"(或者branches或者tags)而不是其他人。

安装

安装Gitolite非常简单, 你甚至不用读自带的那一大堆文档。你需要一个unix服务器上的账户；许多linux变种和solaris 10都已经试过了。你不需要root访问，假设git，perl，和一个openssh兼容的ssh服务器已经装好了。在下面的例子里，我们会用git账户在gitserver进行。

Gitolite是不同于“服务”的软件 -- 其通过ssh访问, 而且每个在服务器上的userid都是一个潜在的“gitolite主机”。我们在这里描述最简单的安装方法，对于其他方法，请参考其文档。

开始，在你的服务器上创建一个名为git的用户，然后以这个用户登录。从你的工作站拷贝你的SSH公钥（也就是你用ssh-keygen默认生成的~/.ssh/id\_dsa.pub文件），重命名为<yourname>.pub（我们这里使用scott.pub作为例子）。然后执行下面的命令：

$ git clone git://github.com/sitaramc/gitolite

$ gitolite/install -ln

# assumes $HOME/bin exists and is in your $PATH

$ gitolite setup -pk $HOME/scott.pub

最后一个命令在服务器上创建了一个名为gitolite-admin的Git仓库。

最后，回到你的工作站，执行git clone git@gitserver:gitolite-admin。然后你就完成了！Gitolite现在已经安装在了服务器上，在你的工作站上，你也有一个名为gitolite-admin的新仓库。你可用通过更改这个仓库以及推送到服务器上来管理你的Gitolite配置。

定制安装

默认快速安装对大多数人都管用，还有一些定制安装方法如果你用的上的话。一些设置可以通过编辑rc文件来简单地改变，但是如果这个不够，有关于定制Gitolite的文档供参考。

配置文件和访问规则

安装结束后，你切换到gitolite-admin仓库（放在你的HOME目录）然后看看都有啥：

$ cd ~/gitolite-admin/

$ ls

conf/ keydir/

$ find conf keydir -type f

conf/gitolite.conf

keydir/scott.pub

$ cat conf/gitolite.conf

repo gitolite-admin

RW+ = scott

repo testing

RW+ = @all

注意 "scott" ( 之前用gl-setup 命令时候的 pubkey 名稱) 有读写权限而且在 gitolite-admin 仓库里有一个同名的公钥文件。

添加用户很简单。为了添加一个名为alice的用户，获取她的公钥，命名为alice.pub，然后放到在你工作站上的gitolite-admin克隆的keydir目录。添加，提交，然后推送更改。这样用户就被添加了。

gitolite配置文件的语法在conf/example.conf里，我们只会提到一些主要的。

你可以给用户或者仓库分组。分组名就像一些宏；定义的时候，无所谓他们是工程还是用户；区别在于你*使用*“宏”的时候

@oss\_repos = linux perl rakudo git gitolite

@secret\_repos = fenestra pear

@admins = scott

@interns = ashok

@engineers = sitaram dilbert wally alice

@staff = @admins @engineers @interns

你可以控制许可在”ref“级别。在下面的例子里，实习生可以push ”int“分支。工程师可以push任何有"eng-"开头的branch，还有refs/tags下面用"rc"开头的后面跟数字的。而且管理员可以随便更改(包括rewind)对任何参考名。

repo @oss\_repos

RW int$ = @interns

RW eng- = @engineers

RW refs/tags/rc[0-9] = @engineers

RW+ = @admins

在RWorRW+之后的表达式是正则表达式(regex)对应着后面的push用的参考名字(ref)。所以我们叫它”参考正则“（refex）！当然，一个refex可以比这里表现的更强大，所以如果你对perl的正则表达式不熟的话就不要改过头。

同样，你可能猜到了，Gitolite字头refs/heads/是一个便捷句法如果参考正则没有用refs/开头。

一个这个配置文件语法的重要功能是，所有的仓库的规则不需要在同一个位置。你能报所有普通的东西放在一起，就像上面的对所有oss\_repos的规则那样，然后建一个特殊的规则对后面的特殊案例，就像：

repo gitolite

RW+ = sitaram

那条规则刚刚加入规则集的 gitolite 仓库.

这次你可能会想要知道访问控制规则是如何应用的，我们简要介绍一下。

在gitolite里有两级访问控制。第一是在仓库级别；如果你已经读或者写访问过了任何在仓库里的参考，那么你已经读或者写访问仓库了。

第二级，应用只能写访问，通过在仓库里的branch或者tag。用户名如果尝试过访问 (W或+)，参考名被更新为已知。访问规则检查是否出现在配置文件里，为这个联合寻找匹配 (但是记得参考名是正则匹配的，不是字符串匹配的)。如果匹配被找到了，push就成功了。不匹配的访问会被拒绝。

带'拒绝'的高级访问控制

目前，我们只看过了许可是R,RW, 或者RW+这样子的。但是gitolite还允许另外一种许可：-，代表 ”拒绝“。这个给了你更多的能力，当然也有一点复杂，因为不匹配并不是唯一的拒绝访问的方法，因此规则的顺序变得无关了！

这么说好了，在前面的情况中，我们想要工程师可以rewind任意branch除了master和integ。 这里是如何做到的

RW master integ = @engineers

- master integ = @engineers

RW+ = @engineers

你再一次简单跟随规则从上至下知道你找到一个匹配你的访问模式的，或者拒绝。非rewind push到master或者integ 被第一条规则允许。一个rewind push到那些refs不匹配第一条规则，掉到第二条，因此被拒绝。任何push(rewind或非rewind)到参考或者其他master或者integ不会被前两条规则匹配，即被第三条规则允许。

通过改变文件限制 push

此外限制用户push改变到哪条branch的，你也可以限制哪个文件他们可以碰的到。比如, 可能Makefile (或者其他哪些程序) 真的不能被任何人做任何改动，因为好多东西都靠着它呢，或者如果某些改变刚好不对就会崩溃。你可以告诉 gitolite:

repo foo

RW = @junior\_devs @senior\_devs

- VREF/NAME/Makefile = @junior\_devs

这是一个强力的公能写在 conf/example.conf里。

个人分支

Gitolite也支持一个叫”个人分支“的功能 (或者叫, ”个人分支命名空间“) 在合作环境里非常有用。

在 git世界里许多代码交换通过”pull“请求发生。然而在合作环境里，委任制的访问是‘绝不’，一个开发者工作站不能认证，你必须push到中心服务器并且叫其他人从那里pull。

这个通常会引起一些branch名称簇变成像 VCS里一样集中化，加上设置许可变成管理员的苦差事。

Gitolite让你定义一个”个人的“或者”乱七八糟的”命名空间字首给每个开发人员(比如，refs/personal/<devname>/\*)；看在doc/3-faq-tips-etc.mkd里的"personal branches"一段获取细节。

"通配符" 仓库

Gitolite 允许你定义带通配符的仓库(其实还是perl正则式), 比如随便整个例子的话assignments/s[0-9][0-9]/a[0-9][0-9]。 这是一个非常有用的功能，需要通过设置$GL\_WILDREPOS = 1; 在 rc文件中启用。允许你安排一个新许可模式("C")允许用户创建仓库基于通配符，自动分配拥有权对特定用户 - 创建者，允许他交出 R和 RW许可给其他合作用户等等。这个功能在doc/4-wildcard-repositories.mkd文档里

其他功能

我们用一些其他功能的例子结束这段讨论，这些以及其他功能都在 "faqs, tips, etc" 和其他文档里。

**记录**: Gitolite 记录所有成功的访问。如果你太放松给了别人 rewind许可 (RW+) 和其他孩子弄没了 "master"， 记录文件会救你的命，如果其他简单快速的找到SHA都不管用。

**访问权报告**: 另一个方便的功能是你尝试用ssh连接到服务器的时候发生了什么。Gitolite告诉你哪个 repos你访问过，那个访问可能是什么。这里是例子：

hello scott, this is git@git running gitolite3 v3.01-18-g9609868 on git 1.7.4.4

R anu-wsd

R entrans

R W git-notes

R W gitolite

R W gitolite-admin

R indic\_web\_input

R shreelipi\_converter

**委托**：真正的大安装，你可以把责任委托给一组仓库给不同的人然后让他们独立管理那些部分。这个减少了主管理者的负担，让他瓶颈更小。这个功能在他自己的文档目录里的 doc/下面。

**镜像**: Gitolite可以帮助你维护多个镜像，如果主服务器挂掉的话在他们之间很容易切换。