Gitosis

把所有用户的公钥保存在 authorized\_keys 文件的做法，只能凑和一阵子，当用户数量达到几百人的规模时，管理起来就会十分痛苦。每次改删用户都必须登录服务器不去说，这种做法还缺少必要的权限管理 — 每个人都对所有项目拥有完整的读写权限。

幸好我们还可以选择应用广泛的 Gitosis 项目。简单地说，Gitosis 就是一套用来管理 authorized\_keys 文件和实现简单连接限制的脚本。有趣的是，用来添加用户和设定权限的并非通过网页程序，而只是管理一个特殊的 Git 仓库。你只需要在这个特殊仓库内做好相应的设定，然后推送到服务器上，Gitosis 就会随之改变运行策略，听起来就很酷，对吧？

Gitosis 的安装算不上傻瓜化，但也不算太难。用 Linux 服务器架设起来最简单 — 以下例子中，我们使用装有 Ubuntu 8.10 系统的服务器。

Gitosis 的工作依赖于某些 Python 工具，所以首先要安装 Python 的 setuptools 包，在 Ubuntu 上称为 python-setuptools：

$ apt-get install python-setuptools

接下来，从 Gitosis 项目主页克隆并安装：

$ git clone https://github.com/tv42/gitosis.git

$ cd gitosis

$ sudo python setup.py install

这会安装几个供 Gitosis 使用的工具。默认 Gitosis 会把 /home/git 作为存储所有 Git 仓库的根目录，这没什么不好，不过我们之前已经把项目仓库都放在 /opt/git 里面了，所以为方便起见，我们可以做一个符号连接，直接划转过去，而不必重新配置：

$ ln -s /opt/git /home/git/repositories

Gitosis 将会帮我们管理用户公钥，所以先把当前控制文件改名备份，以便稍后重新添加，准备好让 Gitosis 自动管理 authorized\_keys 文件：

$ mv /home/git/.ssh/authorized\_keys /home/git/.ssh/ak.bak

接下来，如果之前把 git 用户的登录 shell 改为 git-shell 命令的话，先恢复 'git' 用户的登录 shell。改过之后，大家仍然无法通过该帐号登录（译注：因为 authorized\_keys 文件已经没有了。），不过不用担心，这会交给 Gitosis 来实现。所以现在先打开 /etc/passwd 文件，把这行：

git:x:1000:1000::/home/git:/usr/bin/git-shell

改回:

git:x:1000:1000::/home/git:/bin/sh

好了，现在可以初始化 Gitosis 了。你可以用自己的公钥执行 gitosis-init 命令，要是公钥不在服务器上，先临时复制一份：

$ sudo -H -u git gitosis-init < /tmp/id\_dsa.pub

Initialized empty Git repository in /opt/git/gitosis-admin.git/

Reinitialized existing Git repository in /opt/git/gitosis-admin.git/

这样该公钥的拥有者就能修改用于配置 Gitosis 的那个特殊 Git 仓库了。接下来，需要手工对该仓库中的 post-update 脚本加上可执行权限：

$ sudo chmod 755 /opt/git/gitosis-admin.git/hooks/post-update

基本上就算是好了。如果设定过程没出什么差错，现在可以试一下用初始化 Gitosis 的公钥的拥有者身份 SSH 登录服务器，应该会看到类似下面这样：

$ ssh git@gitserver

PTY allocation request failed on channel 0

ERROR:gitosis.serve.main:Need SSH\_ORIGINAL\_COMMAND in environment.

Connection to gitserver closed.

说明 Gitosis 认出了该用户的身份，但由于没有运行任何 Git 命令，所以它切断了连接。那么，现在运行一个实际的 Git 命令 — 克隆 Gitosis 的控制仓库：

# 在你本地计算机上

$ git clone git@gitserver:gitosis-admin.git

这会得到一个名为 gitosis-admin 的工作目录，主要由两部分组成：

$ cd gitosis-admin

$ find .

./gitosis.conf

./keydir

./keydir/scott.pub

gitosis.conf 文件是用来设置用户、仓库和权限的控制文件。keydir 目录则是保存所有具有访问权限用户公钥的地方— 每人一个。在 keydir 里的文件名（比如上面的 scott.pub）应该跟你的不一样 — Gitosis 会自动从使用gitosis-init 脚本导入的公钥尾部的描述中获取该名字。

看一下 gitosis.conf 文件的内容，它应该只包含与刚刚克隆的 gitosis-admin 相关的信息：

$ cat gitosis.conf

[gitosis]

[group gitosis-admin]

members = scott

writable = gitosis-admin

它显示用户 scott — 初始化 Gitosis 公钥的拥有者 — 是唯一能管理 gitosis-admin 项目的人。

现在我们来添加一个新项目。为此我们要建立一个名为 mobile 的新段落，在其中罗列手机开发团队的开发者，以及他们拥有写权限的项目。由于 'scott' 是系统中的唯一用户，我们把他设为唯一用户，并允许他读写名为iphone\_project 的新项目：

[group mobile]

members = scott

writable = iphone\_project

修改完之后，提交 gitosis-admin 里的改动，并推送到服务器使其生效：

$ git commit -am 'add iphone\_project and mobile group'

[master 8962da8] add iphone\_project and mobile group

1 file changed, 4 insertions(+)

$ git push origin master

Counting objects: 5, done.

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 272 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To git@gitserver:gitosis-admin.git

fb27aec..8962da8 master -> master

在新工程 iphone\_project 里首次推送数据到服务器前，得先设定该服务器地址为远程仓库。但你不用事先到服务器上手工创建该项目的裸仓库— Gitosis 会在第一次遇到推送时自动创建：

$ git remote add origin git@gitserver:iphone\_project.git

$ git push origin master

Initialized empty Git repository in /opt/git/iphone\_project.git/

Counting objects: 3, done.

Writing objects: 100% (3/3), 230 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To git@gitserver:iphone\_project.git

\* [new branch] master -> master

请注意，这里不用指明完整路径（实际上，如果加上反而没用），只需要一个冒号加项目名字即可 — Gitosis 会自动帮你映射到实际位置。

要和朋友们在一个项目上协同工作，就得重新添加他们的公钥。不过这次不用在服务器上一个一个手工添加到~/.ssh/authorized\_keys 文件末端，而只需管理 keydir 目录中的公钥文件。文件的命名将决定在 gitosis.conf中对用户的标识。现在我们为 John，Josie 和 Jessica 添加公钥：

$ cp /tmp/id\_rsa.john.pub keydir/john.pub

$ cp /tmp/id\_rsa.josie.pub keydir/josie.pub

$ cp /tmp/id\_rsa.jessica.pub keydir/jessica.pub

然后把他们都加进 'mobile' 团队，让他们对 iphone\_project 具有读写权限：

[group mobile]

members = scott john josie jessica

writable = iphone\_project

如果你提交并推送这个修改，四个用户将同时具有该项目的读写权限。

Gitosis 也具有简单的访问控制功能。如果想让 John 只有读权限，可以这样做：

[group mobile]

members = scott josie jessica

writable = iphone\_project

[group mobile\_ro]

members = john

readonly = iphone\_project

现在 John 可以克隆和获取更新，但 Gitosis 不会允许他向项目推送任何内容。像这样的组可以随意创建，多少不限，每个都可以包含若干不同的用户和项目。甚至还可以指定某个组为成员之一（在组名前加上 @ 前缀），自动继承该组的成员：

[group mobile\_committers]

members = scott josie jessica

[group mobile]

members = @mobile\_committers

writable = iphone\_project

[group mobile\_2]

members = @mobile\_committers john

writable = another\_iphone\_project

如果遇到意外问题，试试看把 loglevel=DEBUG 加到 [gitosis] 的段落（译注：把日志设置为调试级别，记录更详细的运行信息。）。如果一不小心搞错了配置，失去了推送权限，也可以手工修改服务器上的/home/git/.gitosis.conf 文件 — Gitosis 实际是从该文件读取信息的。它在得到推送数据时，会把新的gitosis.conf 存到该路径上。所以如果你手工编辑该文件的话，它会一直保持到下次向 gitosis-admin 推送新版本的配置内容为止。