链滴

搭建基于 Git&Gitolite 的分布式版本控制系统

作者：honglan

原文链接：https://ld246.com/article/1517997840155

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

Git

Git简介

Linus在1991年创建了开源的Linux，从此，Linux系统不断发展，已经成为最大的服务器系统软件了Linus虽然创建了Linux，但Linux的壮大是靠全世界热心的志愿者参与的，这么多人在世界各地为Linu编写代码。2005年，为了解决Linux系统的源码开发管理这一问题，Linus花了两周时间自己用C写了个分布式版本控制系统，这就是Git，目前世界上最先进的分布式版本控制系统。2008年，GitHub网上线了，它为开源项目免费提供Git存储，无数开源项目开始迁移至GitHub，包括jQuery，PHP，Ruy等等，Git迅速成为最流行的分布式版本控制系统。

Git安装

最早Git是在Linux上开发的，很长一段时间内，Git也只能在Linux和Unix系统上跑。不过，慢慢地有把它移植到了Windows上。现在，Git可以在Linux、Unix、Mac和Windows这几大平台上正常运行。

大多是Linux发行版都装有Git。msysgit是Windows版的Git，从http://msysgit.github.io/下载，然按默认选项安装即可。安装完成后，在开始菜单里找到“Git” -> “Git Bash”，蹦出一个类似命令窗口的东西，就说明Git安装成功！

安装完成后，还需要最后一步设置，在命令行输入：

```
git config --global user.name "Your Name"
git config --global user.email "email@example.com"
```

至此，windows下的Git环境配置成功。关于Git的使用，安利廖雪峰老师的Git教程。

为了后面配置方便，在此生成SSH-key为后面搭建Git服务器提供认证，如果你只是单机使用则无需成。生产方法为：

```
ssh-keygen -t rsa -C "youremail@example.com"
```

然后一路回车，使用默认值即可，无需设置密码。

对于Git新手，庞杂的Git命令可能会让你很烦恼，那么试试Git可视化管理工具SourceTree，默认安即可。

为了方便大家，将上述安装包上传至百度云。

Gitolite

Gitolite简介

如果不是要和他人协同开发，Git 根本就不需要架设服务器。Git 在本地可以直接使用本地版本库的路完成 git 版本库间的操作。但是如果需要和他人分享版本库、协作开发，就需要能够通过特定的网络议操作 Git 库。Git 支持的协议很丰富，架设服务器的选择也很多，不同的方案有着各自的优缺点。Git服务管理工具这个领域，主要有三种流行的方案，它们分别是

- Gitosis - 轻量级， 开源项目，使用SSH公钥认证，只能做到库级的权限控制。目前项目已经停止开，不再维护。
- Gitolite - 轻量级，开源项目，使用SSH公钥认证，能做到分支级的权限控制。
- Git + Repo + Gerrit - 超级重量级，集版本控制，库管理和代码审核为一身。可管理大型及超大型目。

大名鼎鼎的Android平台就是使用的 Git + Repo + Gerrit。对于个人，中小型企业及一些开源项目言，如果没有特殊的要求,其实没有必要去架设上面第三种方案Git服务器。Gitolite提供的服务已经足用。下面将详细讲解怎么搭建Gitolite服务器。

安装基础包

```
$ yum install perl openssh git
```

创建git用户

```
$ adduser git
$ passwd git
```

安装gitolite

切换到git用户

```
$ su – git
```

创建文件夹bin

```
$ mkdir bin
```

从github克隆gitolite的源码

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

安装gitolite

```
$ ./gitolite/install -to /home/git/bin/
```

至此gitolite的安装安装，可以查看bin目录里的内容。

配置gitolite管理员

在管理员本机复制公钥到服务器

```
 cp .ssh/id_rsa.pub /tmp/admin.pub
```

在服务器切换回git用户，为gitolite配置管理员

```
$ /home/git/bin/gitolite setup -pk /tmp/admin.pub
```

配置SSH

修改sshd配置文件(/etc/ssh/sshd_config)，找到以下内容，并去掉注释符”#“

```
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile  /home/git/.ssh/authorized_keys
```

重启ssh服务

```
service sshd restart
```

使用git-shell来限制用户ssh登陆

出于安全的考虑，我们最好限制用户只能进行git push/pull，但无法登陆。这可以使用git-shell来完成
查看一下git-shell的位置

```
$ which git-shell
```

将git-shell的路径添加到 /etc/shells文件中，然后修改git用户的shell：

```
sudo chsh git
```

设置为git-shell的路径，这样，如果再使用ssh方式登陆，则会报错。

Gitolite权限管理

克隆 gitolite-admin 管理库

管理员在本机克隆Gitolite管理库

```
git clone git@yourServerIP:gitolite-admin.git
```

你可以看到在管理库里，有两个目录， conf/和keydir/，其中conf/下面有个名为gitolite.conf的配文件。

- conf/gitolite.conf 用于Git项目配置，访问权限设置。
- keydir/ 用于存储用户的SSH public key(公钥）。

增加新用户

增加新用户，就是允许新用户能够通过其公钥访问 Git 服务。只要将新用户的公钥添加到 gitolite-admin 版本库的 keydir 目录下，即完成新用户的添加。管理员从用户获取公钥，并将公钥按照 username.pub 格式进行重命名，然后进入 gitolite-admin 本地克隆版本库中，复制新用户公钥到 keydir 目录，更新到远程版本库即可。

```
git add.
git commit -m "add user XXX"
git push origin master
```

增加版本库

在conf/gitolite.conf文件中添加两行，然后更新到远程版本库即可。

```
repo gitolite-admin
    RW+                 = jiangxin
```

权限配置

下面我们看一个不那么简单的授权文件:

```
1   @admin = jiangxin wangsheng
2
3   repo gitolite-admin
4       RW+                 = jiangxin
5
6   repo ossxp/.+
7       C                   = @admin
8       RW                  = @all
9
10  repo testing
11      RW+                     =   @admin
12      RW      master          =   junio
13      RW+     pu              =   junio
14      RW      cogito$         =   pasky
15      RW      bw/             =   linus
16      -                       =   somebody
17      RW      tmp/            =   @all
18      RW      refs/tags/v[0-9]    =   junio
```

在上面的示例中，我们演示了很多授权指令。

- 第1行，定义了用户组 @admin，包含两个用户 jiangxin 和 wangsheng。
- 第3-4行，定义了版本库 gitolite-admin。并指定只有用户 jiangxin 才能够访问，并拥有读(R)写(W)和强制更新(+)的权限。
- 第6行，通过正则表达式定义了一组版本库，即在 ossxp/ 目录下的所有版本库。
- 第7行，用户组 @admin 中的用户，可以在 ossxp/ 目录下创建版本库。创建版本库的用户，具有版本库操作的所有权限。
- 第8行，所有用户都可以读写 ossxp 目录下的版本库，但不能强制更新。

- 第9行开始，定义的 testing 版本库授权使用了引用授权语法。
- 第11行，用户组 @admin 对所有的分支和里程碑拥有读写、重置、添加和删除的授权。
- 第12行，用户 junio 可以读写 master 分支。（还包括名字以 master 开头的其他分支，如果有的）。
- 第13行，用户 junio 可以读写、强制更新、创建以及删除 pu 开头的分支。
- 第14行，用户 pasky 可以读写 cogito 分支。（仅此分支，精确匹配）。

详细的权限设置请查阅官方文档。

作者：Gutierrez
链接：https://www.jianshu.com/p/41512a5fb63f
來源：简书
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。