192.168.50.150

This is the one I followed using 'g_admin' instead of 'gitolite' for the 'admin.

Phil Whelan's
BIG FAST BLOG
Vancouver, BC
Canada
Contact me :
+1 (778) 233-4935

Gitolite Installation Step-By-Step

By Phil Whelan on July 23, 2012

In this post I will walk you through how to setup your own gitolite server and start hosting your own git repositories.

Gitolite is a great tool for doing this and works in a very git-like way for managing users, access-rights and setting up new repositories for collaboration between your team. A common reason to setup Gitolite is to remove reliance on services like GitHub.

Let's begin…

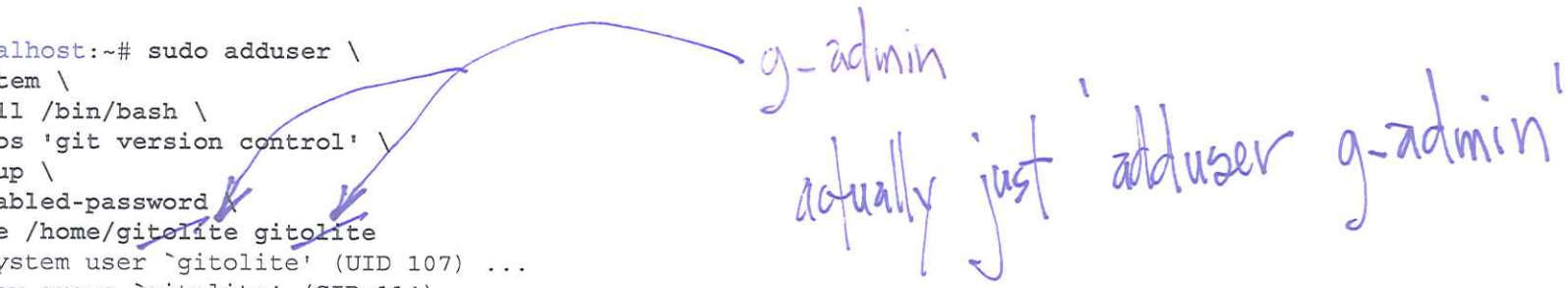
We need hardware!

First, I fired a new machine on Linode. I'm not going to go into the details of that, but it was a fresh install of "Ubuntu 12.04 LTS 64bit" with root user setup and ssh access to the root user via password. This is simply how Linode does it. I know that many people will try this on Amazon Web Services, so I've prefixed all appropriate root commands with "sudo" and you will only have replace "root" with "ubuntu" (or whatever the default username is on the AMI you fire-up).

Mr. Gitolite

First, we are going to login to our remote machine, which will be hosting gitolite on, and we are going to create the "gitolite" user.

```
phil@air:~
$ ssh root@66.175.220.39
The authenticity of host '66.175.220.39 (66.175.220.39)' can't be established.
RSA key fingerprint is 0c:9a:e3:6d:03:21:ce:1b:8c:bb:1d:20:69:cd:98:75.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '66.175.220.39' (RSA) to the list of known hosts.
root@66.175.220.39's password:
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.4.2-x86_64-linode25 x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Tue Jul 24 02:57:20 UTC 2012

  System load:  0.07              Processes:           77
  Usage of /:   2.4% of 19.48GB   Users logged in:     0
  Memory usage: 7%                IP address for eth0: 66.175.220.39
  Swap usage:   0%

  Graph this data and manage this system at https://landscape.canonical.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@localhost:~# sudo adduser \
   --system \
   --shell /bin/bash \
   --gecos 'git version control' \
   --group \
   --disabled-password \
   --home /home/gitolite gitolite
Adding system user `gitolite' (UID 107) ...
Adding new group `gitolite' (GID 114) ...
Adding new user `gitolite' (UID 107) with group `gitolite' ...
Creating home directory `/home/gitolite' ...
```



Now we can return to our local machine.

```
root@localhost:~# exit
logout
Connection to 66.175.220.39 closed.
```

Anyone seen my keys?

In order to provide easy access to the gitolite user on the remote machine, we need to create a key-pair. This is the public and private keys used for ssh.

First, we will create the key pair on our local machine and we will call this key-pair "gitolite".

```
phil@air:~
$ cd ~/.ssh
phil@air:~/.ssh
$ ssh-keygen -t rsa -f gitolite
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in gitolite.
Your public key has been saved in gitolite.pub.
The key fingerprint is:
ca:31:15:88:c1:0a:c7:42:09:f7:b5:dc:22:d2:52:62 phil@air.local
The key's randomart image is:
+--[ RSA 2048]----+
|+oE.oo...        |
|o+o=oo.o .       |
| +o.+ + o        |
|  .o . o         |
|      o S        |
|     . +         |
|      o          |
|                 |
|                 |
+-----------------+
```

We can see that the 2 keys have been created.

```
phil@air:~/.ssh
$ ls -l ~/.ssh/gitolite*
```

```
-rw-------  1 phil  staff  1679 23 Jul 19:58 /Users/phil/.ssh/gitolite
-rw-r--r--  1 phil  staff   396 23 Jul 19:58 /Users/phil/.ssh/gitolite.pub
```

g_admin

"gitolite" is the private key and "gitolite.pub" is the public key.

Upload the public key to the root user's account on the remote machine.

g_admin.pub

```
phil@air:~/.ssh
$ scp ~/.ssh/gitolite.pub root@66.175.220.39:
root@66.175.220.39's password:
gitolite.pub
```

Login to the root users account on the remote machine…

```
phil@air:~/.ssh
$ ssh root@66.175.220.39
root@66.175.220.39's password:
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.4.2-x86_64-linode25 x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Tue Jul 24 03:00:05 UTC 2012

  System load:  0.02              Processes:           75
  Usage of /:   2.4% of 19.48GB   Users logged in:     0
  Memory usage: 7%                IP address for eth0: 66.175.220.39
  Swap usage:   0%

  Graph this data and manage this system at https://landscape.canonical.com/

Last login: Tue Jul 24 02:57:20 2012 from s12079106cffd1777.vc.shawcable.net
```

…and move the key to the gitolite user home directory, being sure to also change the permissions.

g_admin.pub g_admin

```
root@localhost:~# sudo mv gitolite.pub /home/gitolite
root@localhost:~# sudo chown gitolite:gitolite /home/gitolite/gitolite.pub
```

g_admin: g_admin g_admin g_admin.pub

Let's git it on

As I mentioned at top of this post, the remote machine is a fresh machine, so we need to install the "git" command. btw, this command is specific to Ubuntu.

```
root@localhost:~# sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run git-daemon-sysvinit git-doc git-el git-arch git-cvs git-svn git-email git-gui gitk gitweb
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 6,741 kB of archives.
After this operation, 15.2 MB of additional disk space will be used.
Do you want to continue [Y/n]?
Get:1 http://us.archive.ubuntu.com/ubuntu/ precise/main liberror-perl all 0.17-1 [23.8 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu/ precise/main git-man all 1:1.7.9.5-1 [630 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu/ precise/main git amd64 1:1.7.9.5-1 [6,087 kB]
Fetched 6,741 kB in 3s (1,846 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 21183 files and directories currently installed.)
Unpacking liberror-perl (from .../liberror-perl_0.17-1_all.deb) ...
Selecting previously unselected package git-man.
Unpacking git-man (from .../git-man_1%3a1.7.9.5-1_all.deb) ...
Selecting previously unselected package git.
Unpacking git (from .../git_1%3a1.7.9.5-1_amd64.deb) ...
Processing triggers for man-db ...
Setting up liberror-perl (0.17-1) ...
Setting up git-man (1:1.7.9.5-1) ...
Setting up git (1:1.7.9.5-1) ...
```

Now that "git" is installed, let's login as the gitolite user, via the root account….

```
root@localhost:~# sudo su - gitolite
```

g_admin

…and download the "gitolite" installation files.

g_admin @ 192.168.50.150

```
gitolite@localhost:~$ git clone git://github.com/sitaramc/gitolite
Cloning into 'gitolite'...
remote: Counting objects: 7472, done.
remote: Compressing objects: 100% (2464/2464), done.
remote: Total 7472 (delta 5120), reused 7200 (delta 4883)
Receiving objects: 100% (7472/7472), 1.76 MiB | 1.23 MiB/s, done.
Resolving deltas: 100% (5120/5120), done.
```

We will install it in the gitolite user's home directory under "~/bin", so we first create this directory…

```
gitolite@localhost:~$ mkdir bin
```

Now run the install…

```
gitolite@localhost:~$ gitolite/install -to /home/gitolite/bin
```

We're not there yet. This "install" actually only installed the command-line tool that we will use to setup our gitolite server.

So, let's run that command-line tool to setup our gitolite server, giving it the public key that we uploaded, called "gitolite.pub".

```
gitolite@localhost:~$ /home/gitolite/bin/gitolite setup -pk gitolite.pub
Initialized empty Git repository in /home/gitolite/repositories/gitolite-admin.git/
Initialized empty Git repository in /home/gitolite/repositories/testing.git/
WARNING: /home/gitolite/.ssh missing; creating a new one
WARNING: /home/gitolite/.ssh/authorized_keys missing; creating a new one
```

Logout of the gitolite user….

```
gitolite@localhost:~$ exit
logout
```

…and exit the machine.

```
root@localhost:~# exit
logout
Connection to 66.175.220.39 closed.
```

Reduce carpal tunnel by 20%!

Just to make it easy to we are going to give our machine an ssh alias, by adding some config under our ~/.ssh directory.

Edit (or create) ~/.ssh/config

```
phil@air:~
$ vim ~/.ssh/config
```

and add the following lines…

```
Host gitbox
   User gitolite
   Hostname 66.175.220.39
   Port 22
   IdentityFile ~/.ssh/gitolite
```

The IP is specific to my install, so you will want to change that.

Take control

With gitolite, the way we manage repositories, users and access-rights is via a git repository. I love this fact. Git controlling git.

Let's download the gitolite-admin from our newly created git repository.

```
phil@air:~
$ git clone gitbox:gitolite-admin
Cloning into gitolite-admin...
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (6/6), done.
```

What do we have here?

Let's take a look in the repository…

```
phil@air:~
$ cd gitolite-admin/

phil@air:~/gitolite-admin (master)
$ ls -l
total 0
drwxr-xr-x  3 phil  staff  102 23 Jul 20:22 keydir
drwxr-xr-x  3 phil  staff  102 23 Jul 20:22 conf
```

Everyone gets a key

"keydir" is where store our user's keys. Currently, you will see a gitolite.pub file in there. That is the "gitolite" user we use to administrate

our gitolite installation. tester

Let's create a new gitolite user called "phil" by copying phil's public key into the "keydir" directory. You can create the key-pair for this user, just as you did for the gitolite user at the top of this post.

```
phil@air:~/gitolite-admin (master)
$ cp ~/.ssh/phil_rsa.pub keydir/phil.pub
phil@air:~/gitolite-admin (master)
$ git add keydir/phil.pub
phil@air:~/gitolite-admin (master)
$ git commit -m 'added user "phil" to gitolite'
[master d595439] added user "phil" to gitolite
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 keydir/phil.pub
phil@air:~/gitolite-admin (master)
```

Pushing to our remote machine will create that user.

```
$ git push
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 714 bytes, done.
Total 4 (delta 0), reused 0 (delta 0)
To gitbox:gitolite-admin
   7f266e0..d595439  master -> master
```

Privileges

Under the "conf" directory you will find "conf/gitolite.conf" which controls which repositories are available on the system and who has which rights to those repositories.

Let's edit that file with vim…

```
phil@air:~/gitolite-admin (master)
$ vim conf/gitolite.conf
```

@admin @develop @process

…and add the groups "@admin" and "@staff" and a new repository called "bigfastblog".

```
@admin      = phil
```

```
@staff      = phil

repo gitolite-admin
    RW+     = gitolite @admin

repo testing
    RW+     = @all

repo bigfastblog
    RW+     = phil @admin
    R       = @staff
```

As you can see, anyone in our @staff group has read-only privileges to the new "bigfastblog" git repository, but only "phil" and users we add to the @admin group will be able to git-push to that repository.

```
phil@air:~/gitolite-admin (master)
$ git commit conf/gitolite.conf -m 'added "bigfastblog" repo'
[master 357bbc8] added "bigfastblog" repo
 1 files changed, 9 insertions(+), 1 deletions(-)
```

When we "git push" our changes to this configuration back to our gitolite server, we can actually see in the dialog that the "bigfastblog" git repository is being created.

```
phil@air:~/gitolite-admin (master)
$ git push
Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 428 bytes, done.
Total 4 (delta 0), reused 0 (delta 0)
remote: Initialized empty Git repository in /home/gitolite/repositories/bigfastblog.git/
To gitbox:gitolite-admin
   d595439..357bbc8  master -> master
```

I hope that gives you a good overview of how to install gitolite. If you found this helpful, please subscribe to this blog or follow me on twitter @philwhln.

Posted in Infrastrucutre, Software Development | Tagged git, github, gitolite, ssh-keygen | 47 Responses