Ubuntu 12.04 server版

1. 安装git-core

Sudo apt-get install git-core

2.新加用户git， 该用户将作为所有代码仓库和用户权限的管理者：  
[a@server:~$](mailto:a@server:%7E$)sudo useradd -m git

[a@server:~$](mailto:a@server:%7E$)sudo passwd git

建立一个git仓库的存储点：  
[a@server:~$](mailto:a@server:%7E$)sudo mkdir /home/repo

### 3. 安装配置gitosis

git clone  <https://github.com/res0nat0r/gitosis>

cd gitosis;

sudo python setup.py install ///安装gitosis

4.

如果你将作为git服务器的管理员，那么在你的电 脑上(另一台pc）生成ssh公钥：  
[usr@pc1:~$](mailto:a@server:%7E$)ssh-keygen -t rsa  
将公钥拷贝到服务器的/tmp下：  
[usr@pc1:~$](mailto:a@server:%7E$)scp .ssh/id\_rsa.pub [git@<server>:/tmp](mailto:git@192.168.1.39:/tmp)

回到git服务器上  
[a@server:/tmp/gitosis$](mailto:a@server:/tmp/gitosis$)sudo chmod a+r /tmp/id\_rsa.pub

让gitosis运行起来：  
[a@server:/tmp/gitosis$](mailto:a@server:/tmp/gitosis$)sudo -H -u git gitosis-init < /tmp/id\_rsa.pub  
Initialized empty Git repository in /home/repo/gitosis-admin.git/  
Reinitialized existing Git repository in /home/repo/gitosis-admin.git/

gitosis的有趣之处在于，它通过一个git仓库来管理配置文件，仓库就放在了/home/repo/gitosis- admin.git。我们需要为一个文件加上可执行权限：

[root@server:/home/git](mailto:root@server:/home/git)# cd repositories  
[root@server:/home/git/repositories](mailto:root@server:/home/git/repositories)# cd gitosis-admin.git/  
[root@server:/home/git/repositories/gitosis-admin.gi](mailto:root@server:/home/git/repositories/gitosis-admin.git) [t](mailto:root@server:/home/git/repositories/gitosis-admin.git)# sudo chmod 755 /home/repo/gitosis-admin.git/hooks/post-update

刚刚提到，gitosis本身的配置也是通过git来实现的。在你自己的开发机里，把gitosis-admin.git这个仓库clone下来，就可以以管理员的身份修改配置了。

在你的电脑里：  
git clone [git@192.168.0.112:gitosis-admin.git](mailto:git@192.168.0.112:gitosis-admin.git)

就可以下载gitosis-admin。

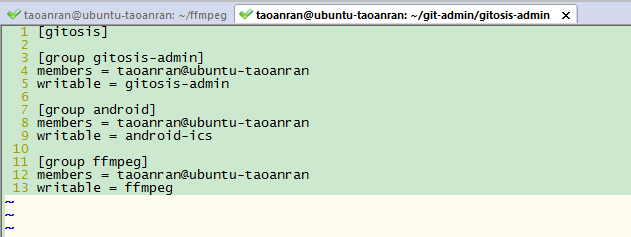
[usr@pc1:~/work$](mailto:a@server:%7E/work$)cd gitosis-admin/

该目录下的keydir目录是用来存放所有需要访问git服务器的用户的ssh公钥：

各个用户按照前面提到的办法生成各自的ssh公钥文件后，把所有人的 ssh公钥文件都拿来，按名字命名一下，比如b.pub, lz.pub等，统统拷贝到keydir下：

[usr@pc1:~/work/gitosis-admin$](mailto:a@server:%7E/work/gitosis-admin$)su root  
[root@server:/home/a/work/gitosis-admin](mailto:root@server:/home/a/work/gitosis-admin)# cp /path/to/.ssh/id\_rsa.pub ./keydir/b.pub   
[root@server:/home/a/work/gitosis-admin](mailto:root@server:/home/a/work/gitosis-admin)# exit

修改gitosis.conf文件，我的配置大致如下：



这个配置文件表达了如下含义：gitosis-admin组成员有陶安然，该组对gitosis-admin仓库有读写权限；

当然目前这些配置文件的修改只是在你的本地，你必须推送到远程的gitserver上才能真正生效。  
加入新文件、提交并push到git服务器：  
[usr@pc1:~/work/gitosis-admin$](mailto:a@server:%7E/work/gitosis-admin$)git add .  
[usr@pc1:~/work/gitosis-admin$](mailto:a@server:%7E/work/gitosis-admin$)git commit -am “add teamweok prj and users”  
[usr@pc1:~/work/gitosis-admin$](mailto:a@server:%7E/work/gitosis-admin$)git push origin master

到这里，git服务器已经可以使用了。

下面开始配置gitWeb

## 配置gitweb

1. 安装gitweb

   sudo apt-get install gitweb

2. 安装apache2

  sudo apt-get install apache2

3. 配置gitweb   
(1)默认没有 css 加载，把 gitweb 要用的静态文件连接到 DocumentRoot 下：  
   cd /var/ www/   
   sudo ln -s / usr/ share/ gitweb/\* .

   (注意后面的点)

(2)修改配置：

sudo vi /etc/ gitweb.conf

   将 $projectroot 改为gitosis-admin.git所在目录： /home/git/repositories

 (3)修改 /home/git/repositories权限，默认情况下，gitosis将 repositories权限设置为不可读的

    sudo chmod 777 -R /home/git/repositories

重新启动apache：sudo /etc/init.d/apache2 restart，访问<http://localhost/cgi-bin/gitweb.cgi>

即<http://192.168.10.103/cgi-bin/gitweb.cgi>

GIT服务器管理

Git 服务器管理

1. 我们使用gitosis作为git服务器管理工具，官方source code 在<https://github.com/res0nat0r/gitosis.git>

但是由于这个的管理功能比较单一，所有我使用的是蒋鑫的<https://github.com/ossxp-com/gitosis>，他的gitosis已经做了一些修改，比较完善.

获取并安装gitosis

git clone <https://github.com/ossxp-com/gitosis>

cd gitosis

sudo python setup.py install

配置gitosis

cp ~/.ssh/id\_rsa.pub /tmp

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

sudo chmod 755 /home/git/repositories/gitosis-admin.git/hooks/post-update

管理gitosis配置

cd ~

git clone git@hostname:用户名/gitosis-admin.git

cd gitosis-admin/

gitosis-admin.conf文件配置(权限管理)

[gitosis]

[group gitosis-admin]

writable = gitosis-admin

members = taoanran

[group all-admin]

members = taoanran

admin = amlogic-ics/\*\*

[group all-coders]

members = sunke yiyuan

write = amlogic-ics/\*\*

[group my]

members = taoanran

admin = \*\*

write = amlogic-ics.bak/\*\*

members = taoanran

admin=amlogic-ics/\*\*

指的是该用户(taoanran)拥有对amlogic-ics目录里所有仓库或是目录的admin(读写，创建仓库) 的权限.

[group my]

members = taoanran

admin = \*\*

write = amlogic-ics.bak/\*\*

“admin = \*\*” 指该用户对/home/git/ repositories 目录了的仓库(不包括目录)有权限，如果在/home/git/repositories 目录里有下级目录，下级目录里有git仓库， 该用户没有对这些目录的admin权限!!!!

当如果有用户需要加入到git中，使之有相应的权限，需要该用户把自己的public key传到git server上，然后改名为类似于zhangsan.pub, 放在gitosis-admin/keydir 目录里。然后在gitosis.conf文件里相应的项目里添加member=zhangsan(只能写.pub前面的字符串),然后git commit , git push 到server上就行

具体的可以参考《Git权威指南》----蒋鑫著

1.repo

repo是一个脚本。用来管理由多个git仓库组成项目的脚本工具。

通过一个标有远端地址，分支信息，仓库名等信息的xml文件去管理多个git仓库组成的项目。

2.repo下载

摘自google android官网

To install, initialize, and configure Repo, follow these steps:

Make sure you have a bin/ directory in your home directory, and that it is included in your path:

$ mkdir ~/bin  
$ PATH=~/bin:$PATH

Download the Repo script and ensure it is executable:

$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo  
$ chmod a+x ~/bin/repo

3.核心是manifest文件，让我们看一下

<?xml version="1.0" encoding="UTF-8"?>

<manifest>

<remote name="aosp"

fetch=".." /> //这里声明了一个叫aosp的remote，地址是'..’，意思是这个menifest的地址的上一级，我们看下这个menifest的地址

<https://android.googlesource.com/platform/manifest>，所以fetch地址就是https://android.googlesource.com

//之前老版本的repo，fetch是不支持相对路径的。

<default revision="refs/tags/android-4.0.3\_r1"

remote="aosp"

sync-j="4" /> //声明了一些默认参数，分支，remote等，sync-j参数是后来新加的支持，应该是repo sync的默认线程数，可以加快sync的速度

<project path="build" name="platform/build">

<copyfile src="core/root.mk" dest="Makefile" />

</project>//声明了一个project,path="build" name="platform/build"表示会从https://android.googlesource.com/platform/build.git仓库clone到本地，checkout后会建立一个build目录，copyfile参数表示会在checkout的过程结束后，把src制定的文件拷贝到dest制定的地方，这里是把build/core/root.mk文件拷贝到源码根目录，取名Makefile。可以思考下为什么这样做。

<project path="abi/cpp" name="platform/abi/cpp" />//剩下的就与上面类似了，只不过目录结构会多一些，这里就是把https://android.googlesource.com/platform/abi/cpp.git仓库clone到本地，并checkout到“本地android源码根目录/abi/cpp”中。

<project path="bionic" name="platform/bionic" />

<project path="bootable/bootloader/legacy" name="platform/bootable/bootloader/legacy" />

<project path="bootable/diskinstaller" name="platform/bootable/diskinstaller" />

<project path="bootable/recovery" name="platform/bootable/recovery" />

<project path="cts" name="platform/cts" />

<project path="dalvik" name="platform/dalvik" />

<project path="development" name="platform/development" />

<project path="device/common" name="device/common" />

?xml version="1.0" encoding="UTF-8"?>

…...............

后面省略若干行。。。

3.小例子

自己搭建一个由三个仓库组成的项目，用repo管理起来。

$mkdir source.git

$cd source.git

$mkdir manifest.git //建立我们manifest文件存放的仓库目录

$mkdir project1.git

$mkdir dir1

$cd dir1

$mkdir project2.git

$mkdir project3.git

//建立第一个仓库

$cd ../project1.git

$touch project1.cpp

$git init

$git checkout -b branch-test

$git add .

$git commit -am “init commit” //git add . 与commit -am不是什么太好的习惯，add .很容易把不该跟踪的东西加入到版本库中，commit -am 则让你失去最后一次检查到这个错误的机会，好习惯是挨个git add你想要提交的文件，然后commit,尽量不要使用-m那样说明你的提交注释非常少。

//建立第二个与第三个仓库

$cd ../dir1/project2.git

$touch project2.cpp

$git init

$git checkout -b branch-test

$git add .

$git commit -am “init commit”

$cd ../project3.git

$touch project3.cpp

$git checkout -b branch-test

$git add .

$git commit -am “init commit”

//建立manifest仓库

$cd ../../manifest.git

$gedit default.xml //额，我故意没有用vi或者emacs，这算不算增进用户体验？

//添加如下内容

<?xml version="1.0" encoding="UTF-8"?>

<manifest>

<remote name="test-remote"

fetch="." />

<default revision="branch-test"

remote="test-remote"

sync-j="4" />

<project path="project1" name="project1" />

<project path="dir1/project2" name="dir1/project2" />

<project path="dir1/project3" name="dir1/project3" />

</manifest>

//提交

$git init

$git checkout -b branch-test

$git add .

$git commit -am “init commit”

这里我们就把三个仓库与manifest仓库都建好了，然后我们来使用repo

新建一个目录来保存我们的项目

$mkdir repo\_test

$cd repo\_test

$repo init -u ~/source.git/manifest.git -b branch-test

//确认配置

//执行完毕后开始同步代码

$repo sync

这样大家就可以从这个小例子看出如何使用repo去管理你自己的代码了。

下面是我的仓库的tree:

/home/git/repositories/android-ics/

Platform ------- abi/cpp.git

bootable/

cts.git/

development.git/

bionic.git/

manifests.git ----🡪这个是repo的default.xml的仓库

…………………..（等等）

Device ---------- common.git/

generic/

google/

sample.git/

samsung/

ti/

我们首先需要建立manifests.git仓库，这里是repo脚本的XML文档，这个XML会去管理android的所有git仓库。这里我按照google的思路把XML放在了platform目录下，然后利用repo\_create.sh和default.xml(可以从google官网上的android代码里找，位置在根目录里.repo文件夹中)去建立android仓库群。