Git
Servidores privados

Git. Servidores privados

Jesús Amieiro Becerra

Versión actual: 16/05/2014

Obtén la última versión disponible en http://www.fontelearn.com/es/git-servidores-privados/

Índice de contenido

1 Presentación

Este es uno de los libros de una serie sobre Git, un sistema de control de versiones desarrollado por Linux Torvalds en el año 2005 y que se ha hecho tremendamente popular gracias a servicios como GitHub y a su amplia aceptación en proyectos importantes como el Kernel de Linux, Android, Ruby on Rails, Eclipse, GNOME, KDE, Qt, Perl o PostgreSQL o por empresas como Google, Facebook, Microsoft, Twitter, LinkedIn o Netflix.

Si eres programador, desarrollador web, administrador de sistemas, diseñador, ... es muy probable que en algún momento de tu trabajo te encuentres con un proyecto en el que tengas que colaborar con otras personas usando Git. Puede que trabajes solo pero que te interese tener un seguimiento y control de tu trabajo. En estos dos casos y en muchos más un conocimiento más o menos profundo de Git te permitirá ser mucho más productivo en tu trabajo y, sobre todo, evitar muchos de los problemas con los que se encuentra a menudo la gente que no trabaja con un sistema de control de versiones.

Este libro supone que eres un usuario medio de Git, que eres capaz de gestionar los commits a lo largo del tiempo y que sabes cómo colaborar con otros usuarios a través de un servidor.

1.1 Contenido

Este libro está en versión **alfa**. Esto quiere decir que le faltan contenidos y, aunque he tratado de evitar cualquier tipo de error, puede que encuentres alguno. En ese caso te agradecería que lo notificaras a través de cualquiera de los métodos de contacto indicados en el capítulo *1.2 Contacto*.

¿Qué vas a encontrar en este libro?

En el capítulo *2 Introducción* describimos los equipos usados para realizar las pruebas, veremos las opciones que tenemos para centralizar la información en un servidor Git, veremos los protocolos de comunicaciones que usa Git y veremos cómo se crean las claves para autentificarnos ante otros equipos.

En el capítulo *3 Configuración de un servidor* veremos cómo centralizar Git mediante un servidor SSH, que instalaremos, y claves RSA, que generaremos.

En el capítulo *4 Gitolite* presentaremos Gitolite, un software que nos permite configurar un servidor Git en un servidor privado, con un meticuloso control de acceso, crear usuarios, repositorios y gestionar sus privilegios de acceso. Para finalizar veremos un ejemplo práctico de uso.

1.2 Contacto

Puedes ponerte en contacto con nosotros a través de cualquiera de los métodos de contacto indicados en la dirección web http://fontelearn.com/es/contacto/

2 Introducción

Git es un sistema de control de versiones distribuido, que nos permite trabajar en el equipo local sin necesidad de tener conexión a una red; pero también podemos colaborar con otros usuarios compartiendo los cambios que vamos realizando en un proyecto.

Técnicamente podríamos enviar (push) y obtener (pull) los cambios con cada repositorio individual, ubicado en un equipo de usuario, pero no es lo más adecuado, ya que necesitamos que los equipos estén disponibles en todo momento, y para ello es mejor un equipo con un perfil de servidor y no de usuario.

Lo que haremos será establecer un equipo como servidor y será ese equipo el que usemos para enviar los cambios (push) y recibirlos (pull).

Para poder realizar estas comunicaciones podremos usar varios protocolos, que vamos a ver en el capítulo 2.3.

Antes vamos a ver los equipos que se han usado en las pruebas, por si las quieres reproducir en algún momento.

A continuación veremos las opciones que tenemos para centralizar la información en un servidor Git: un servidor propio o uno externalizado.

Luego analizaremos los distintos protocolos y, para finalizar, veremos cómo se crean los pares de claves (pública y privada) de tipo RSA para poder autentificarnos ante otros equipos.

2.1 Equipo de pruebas

Las pruebas se han realizado en dos máquinas virtuales ejecutadas con VMware Player 6.0.2 y con las VMware tools instaladas. Se puede utilizar cualquier otro sistema de virtualización (Microsoft Hyper-V, Citrix XenServer, Oracle VirtualBox, Proxmox, KVM, ...) o máquinas físicas, a elección del administrador de los sistemas o del responsable de TIC en la empresa. Lo único que tenemos que conseguir es que las máquinas tengan visibilidad entre ellas a nivel IP.

2.1.1 Máquina 1

Debian 7 32 bits

Esta máquina tiene los siguientes usuarios:

- Usuario/contraseña: root/root
- Usuario/contraseña:fontelearn/fontelearn, con permisos de administración.

Su configuración de red es la siguiente:

- IP versión 4 estática.
- Dirección IP: 192.168.0.235
- Máscara de red: 255.255.255.0
- Puerta de enlace: 192.168.0.1
- DNS: 8.8.8.8, 8.8.4.4, 208.67.222.222, 208.67.220.220 (Google y OpenDNS)
- Configuración del adaptador de red de Vmware en modo “Bridged”, no en modo “NAT”.

Software instalado: Git, paquete build-essential.

2.1.2 Máquina 2

Fedora 19 32 bits

Esta máquina tiene los siguientes usuarios:

- Usuario/contraseña: root/root
- Usuario/contraseña:fontelearn/fontelearn, con permisos de administración.

Su configuración de red es la siguiente:

- IP versión 4 estática.
- Dirección IP: 192.168.0.236
- Máscara de red: 255.255.255.0
- Puerta de enlace: 192.168.0.1
- DNS: 8.8.8.8, 8.8.4.4, 208.67.222.222, 208.67.220.220 (Google y OpenDNS)
- Configuración del adaptador de red de Vmware en modo “Bridged”, no en modo “NAT”.

Software instalado: Git

2.2 Servidor propio o externalizado

A la hora de utilizar un servidor tenemos que decidir si vamos a usar una solución alojada en un servidor administrado por nosotros o una solución gestionada por otra empresa.

En el caso del servidor administrado por nosotros, que podemos tener alojado en nuestra oficina, en casa, en un centro de datos en el que alquilamos el servidor, ... tenemos un control total sobre los datos que gestionamos, pero tenemos que invertir más tiempo en configurar el sistema, en mantenerlo actualizado, ... Entre las soluciones software que encajan en esta tipología encontramos Gitolite, Gerrit, GitLab, Gitorius, ...

En el caso de servicios externalizados, entre los que podemos gestionar repositorios, usuarios, permisos, ... se encuentran GitHub, Bitbucket, Gitorius, GitLab,...

2.3 Protocolos

Git soporta cuatro protocolos:

- Sistema de archivos.
- HTTP(S).
- SSH.
- Git.

2.4 Generación de claves SSH

Para poder acceder a una máquina SSH utilizando archivos de claves (par clave pública/clave privada), lo primero que tenemos que hacer es generarlas.

El conjunto de clave pública/privada se suele almacenar en el directorio ".ssh" en el directorio raíz de cada usuario. Por ejemplo, si estamos autentificados con el usuario "*fontelearn*", las claves, si existen, estarán almacenadas en el directorio "*/home/fontelearn/.ssh/*".

Para comprobar que si existen simplemente ejecutamos

```
ls /home/fontelearn/.ssh/
```

Si aparece un mensaje como el siguiente

```
ls: no se puede acceder a /home/fontelearn/.ssh/: No existe el fichero o el
directorio
```

Es que aún no tenemos ningún fichero de claves en el directorio.

Para generarlas ejecutamos el comando

```
ssh-keygen
```

Nos va a preguntar el archivo donde queremos guardar las claves (*/home/fontelearn/.ssh/id_rsa_fontelearn*) y nos pedirá una contraseña de acceso a la clave.

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/home/fontelearn/.ssh/id_rsa):
/home/fontelearn/.ssh/id_rsa_fontelearn
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/fontelearn/.ssh/id_rsa_fontelearn.
Your public key has been saved in /home/fontelearn/.ssh/id_rsa_fontelearn.pub.
The key fingerprint is:
21:e9:98:3f:a2:85:66:2d:f7:7e:b5:6e:c1:69:13:16 fontelearn@debian
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|       .  E      |
|      o .  .     |
|     + . .o      |
|    o . So o     |
|   o .    B      |
|  = = o  o +     |
| o = o .. o      |
|  .  .o. o.      |
+-----------------+
```

Si ahora vemos el contenido del directorio

```
ls -la ~/.ssh
```

Aparecen los dos archivos de claves

```
id_rsa_fontelearn
id_rsa_fontelearn.pub
```

El que tiene la extensión ".pub" es la clave pública, la que podemos compartir, y el que no tiene extensión es la clave privada, la que debemos guardar del resto de usuarios.

Si vemos el contenido de los archivos

```
cat .ssh/id_rsa_fontelearn
```

La clave privada tiene un aspecto como el siguiente

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA6DfLJmqyUBoAyKeMgt0wQdbtLe3NmPVf1XwJyNQad9HJwJLt
J/b+eghZLEPjP7WMbIyzAlDoetReO1C46VRkWrtAj/Y4SOOeta+o2T6G+fVx5Nuo
SCQa0eSzW+/U9aP3OBwCSSUhrDStPeH17YsZAds1rw+BvLHJSx+yd5Cl2zk/t5S8
HcNFFBZ3yG9hJ1TwR9LZdCJmGAXQ7sq3/xeFdl+xnMU/1+OdAiFzqNt0/Mf9cHE1
RFQIAl/+yMP/durL+Qgi7oZh6qvExT4R5yDWsm+FsrkNSOBmeNiUbmfpYj2PFTVp
6jjMBcnrQjJLx83IjoLIAMfaSuHdhfVVjs4Z5QIDAQABAoIBAHazbmioyZUgdZ4b
5ztNjUlab8q2cA3r/5zE1suNgCoXVqXVdNyk6NfARLTXzuXWC33BPg/FA9EllTWE
wcCjntiyNwJfvc2kbJA6xnil+aDz0dBGZ/LYYc1Atl4cbiNSOSC5UIHLNTcthhDH
3EocfB/yybI+nvaxek+oWDW9/snbtEOdWcS5fPhvkGuv41w1ZTlMpT1gBd+5mcX0
/MqRAW+nH1kmqz4KqlQTPKVfSQkORMDiugwW07RBA3nNQb+v5Yxrd4iZQ2B1Y7aa
ycmfEtYzUsjfALZO0eiLCxnNgVwIRV3Gjg47Zjp7TA2QQyzoayD/MqjAIH0Hxw/+
Ja5kD1ECgYEA/OTQtFMIr093fyt90ujKd3OJVB3z0yZNv4Y0U26QosxPS6rDCOMo
G5rjhXoQSkkllPmvVvk9OD9wpDT171jPL2O//oa9RYRdbxDHbHRjyTCc4QVn5rVy
QAfuHG7iRuvEbXRLV+Tz4qtsNaa/IClE0HqKkRCEcQzL1pJ0A/CUXlsCgYEA6xH3
X2mAGNsdRKOqfogztiBaxZqctpZjaoPktykS6d+Fkd5HBLK5LfbtGvWC4zup4hjb
4579Qk28J+fzY48KfA6oAE4aY6N3DIb13SNw0JO3+y48UoV+xuxuN0fuVWQmpoTa
skc41YAiqjL4CkrIu1xdjCEuGKicTtjPjtcJXL8CgYAn8T/ha0fw4H1AMa9S03++
9BmZlo1/KQqYmGrFTq47eEeB7PuW+MHJ8mHDIC9YfEsaR4FhHEaGDscMuGyrm/xF
QNz/J2rXgvt64kyNHajxCuOMSt1JPBlKcbYy+XR1vSEuJHMN89RtVaPmqWqAeuQq
hWBDXzd734iGYGVAhgv5BwKBgGqLEhvkrXKWnYVmyPfgs89wirPViAIZ16WHcjqy
Gyh1bwzWOkoe11/F82GstU5Y1H7t1Xwtq4xJv2Y281HEm3PvEKZSwBM5G9RJt1Yr
og5nZZQrOJwOdY+jybC0oYmCgyImRRlYvqHGGfAaeeHwU1rYWm4Q+9uhUHdMPYb4
X/3xAoGBAJXDdQJL9P/Ntyp20IyFqooyaP9gWlPMmjAAXWN4brvR5hl40B4IGxY4
```

```
aoBdW3qLDhqDX4y6CK+fOOaF5wB4VyLPTOYRK9yy8QK+aEnGUv0dRuZ+tiMAJTRU
+hFb+SRz/MCCypPZf+FcJO3rCFE5YeUadsmLKtJGFbNBsNn0m9W5
-----END RSA PRIVATE KEY-----
```

Mientras que si vemos la clave pública

```
cat .ssh/id_rsa_fontelearn.pub
```

Tiene una apariencia similar a

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDoN8smarJQGgDIp4yC3TBB1u0t7c2Y9V/VfAnI1Bp30cnAku
0n9v56CFksQ+M/tYxsjLMCUOh61F47ULjpVGRau0CP9jhI4561r6jZPob59XHk26hIJBrR5LNb79T1
o/c4HAJJJSGsNK094fXtixkB2zWvD4G8sclLH7J3kKXbOT+3lLwdw0UUFnfIb2EnVPBH0tl0ImYYBd
Duyrf/F4V2X7GcxT/X450CIXOo23T8x/1wcTVEVAgCX/7Iw/926sv5CCLuhmHqq8TFPhHnINayb4Wy
uQ1I4GZ42JRuZ+liPY8VNWnqOMwFyetCMkvHzciOgsgAx9pK4d2F9VWOzhnl fontelearn@debian
```

3 Configuración de un servidor

Lo primero que vamos a hacer es instalar un servidor SSH (máquina 1) para que los clientes se puedan conectar a la máquina

```
fontelearn@servidor$ sudo apt-get install openssh-server
```

La instalación genera una serie de archivos de configuración almacenados en */etc/ssh/*:

- sshd_config : archivo de configuración del servidor
- ssh_host_dsa_key : clave privada DSA
- ssh_host_dsa_key.pub : clave pública DSA
- ssh_host_ecdsa_key : clave privada ECDSA
- ssh_host_ecdsa_key.pub : clave pública ECDSA
- ssh_host_rsa_key : clave privada RSA
- ssh_host_rsa_key.pub : clave pública RSA

Para más información sobre la configuración de un servidor SSH podemos consultar la wiki del proyecto Debian https://wiki.debian.org/SSH

Lo siguiente que vamos a hacer es crear un usuario git en el servidor y configurar su fichero de claves autorizadas

```
fontelearn@servidor$ sudo adduser git
fontelearn@servidor$ sudo su - git
fontelearn@servidor$ mkdir .ssh
```

Ahora, en ese directorio vamos a añadir las claves públicas de los usuarios a los que le queremos dar acceso. Para ello vamos a crear en la máquina 2 los usuarios “cliente_01” y “cliente_02”.

Creamos el “cliente_01”, creamos sus claves RSA y las copiamos a la carpeta /home/git/.ssh/ de la máquina 1, del servidor.

```
fontelearn@pc$ sudo adduser cliente_01
fontelearn@pc$ sudo su - cliente_01
fontelearn@pc$ cd ~/.ssh/
fontelearn@pc$ ssh-keygen -t rsa -f cliente_01
fontelearn@pc$ scp /home/cliente_01/.ssh/cliente_01.pub
git@192.168.0.235:/home/git/.ssh/
```

Hacemos la misma operación con el “cliente_02”.

```
fontelearn@pc$ sudo adduser cliente_02
```

```
fontelearn@pc$ sudo su - cliente_02
cliente_02@pc$ cd ~/.ssh/
cliente_02@pc$ ssh-keygen -t rsa -f cliente_02
cliente_02@pc$ scp /home/cliente_02/.ssh/cliente_02.pub
git@192.168.0.235:/home/git/.ssh/
```

En el servidor lo que vamos a hacer es añadir esas dos claves públicas al archivo “authorized_keys” del usuario “git”.

```
fontelearn@servidor$ sudo su - git
git@servidor$ cat ~/.ssh/cliente_01.pub >> ~/.ssh/authorized_keys
git@servidor$ cat ~/.ssh/cliente_02.pub >> ~/.ssh/authorized_keys
```

Ahora inicializamos el repositorio con el comando “git init” con la opción *--bare*, que inicializa el repositorio sin un directorio de trabajo, ya que estamos en un servidor.

```
git@servidor$ ls /opt
git@servidor$ mkdir /opt/git
git@servidor$ cd /opt/git
git@servidor$ mkdir proyecto.git
git@servidor$ cd proyecto.git
git@servidor$ git --bare init
```

Y ya podemos empezar con el trabajo habitual en la máquina 2 o en otras más que tengan bien configuradas las claves RSA.

```
fontelearn@pc$ sudo su - cliente_01
cliente_01@pc$ git config --global user.name “Cliente 01”
cliente_01@pc$ git config --global user.email
cliente_01@fontelearn.com
cliente_01@pc$ mkdir proyectos
cliente_01@pc$ cd ~/proyectos/
cliente_01@pc$ git init
cliente_01@pc$ echo “Línea 1” >> archivo_01.txt
cliente_01@pc$ git add .
cliente_01@pc$ git commit -m “Commit inicial. Añado el
archivo_01.txt”
cliente_01@pc$ git remote add origin
git@192.168.0.235:/opt/git/proyecto.git
```

```
cliente_01@pc$ $ git push origin master
```

Ahora cambiamos al cliente 2

```
cliente_01@pc$ sudo su - cliente_02
cliente_02@pc$ git config --global user.name “Cliente 02”
cliente_02@pc$ git config --global user.email
cliente_02@fontelearn.com
cliente_02@pc$ mkdir proyectos
cliente_02@pc$ cd ~/proyectos/
cliente_02@pc$ git clone git@192.168.0.235:/opt/git/proyecto.git
cliente_02@pc$ echo “Línea 2” >> archivo_02.txt
cliente_02@pc$ git add .
cliente_02@pc$ git commit -m “Añado el archivo_02.txt”
cliente_02@pc$ git remote add origin
git@192.168.0.235:/opt/git/proyecto.git
cliente_02@pc$ $ git push origin master
```

Ahora obtengo los cambios con el cliente 1

```
cliente_02@pc$ sudo su - cliente_01
cliente_02@pc$ cd ~/proyectos/proyecto
cliente_02@pc$ git pull
cliente_02@pc$ ls -la
```

4 Gitolite

Gitolite es un software que nos permite configurar un servidor Git en un servidor privado, con un meticuloso control de acceso y muchas características más potentes.

4.1 ¿Qué es gitolite?

Gitolite es una capa de control de acceso ubicada por encima de Git. Algunas de sus características fundamentales son:

- Utiliza un usuario unix (usuario "real") en el servidor.
- Proporciona acceso a muchos usuarios gitolite:
    - No son los usuarios "reales" del sistema.
    - No tienen acceso a una consola.
- Controla el acceso a muchos repositorios Git:
    - Acceso de lectura controlado a nivel de repositorio.
    - Acceso de escritura controlado a nivel de rama / etiqueta / archivo / directorio, incluyendo quién puede rebobinar, crear y eliminar ramas / etiquetas.
- Se puede instalar sin acceso de root, asumiendo que tenemos instalado git y perl.
- La autentificación se lleva a cabo mediante sshd, pero también se puede usar http.

4.2 Requisitos

4.2.1 Servidor

- Un sistema Unix compatible con POSIX.
- Git 1.6.6 o superior.
- Perl 5.8.8 o superior.
- Openssh.
- Un usuario Unix para gitolite. Habitulamente es el usuario "*git*", pero puede ser cualquier usuario, incluso uno existente. (Si estás utilizando un sistema RPM/DEB la instalación probablemente cree uno llamado "*gitolite*").

4.2.2 Cliente

- Un cliente Openssh.
- Git 1.6.6 o superior.

4.3 Instalación

4.3.1 Usuario y claves

Máquina 1

Lo primero que vamos a hacer es acceder al servidor donde se va a alojar Gitolite y crear el usuario “gitolite”.

```
fontelearn@servidor$ sudo adduser \
   --system \
   --shell /bin/bash \
   --gecos 'git version control' \
   --group \
   --disabled-password \
   --home /home/gitolite gitolite
```

A continuación añadimos este usuario a los permitidos para acceder al equipo por ssh

```
vim /etc/ssh/sshd_config
```

Y le añadimos la siguiente línea, que permite el acceso por ssh al usuario gitolite

```
AllowUsers gitolite
```

Guardamos los cambios y reiniciamos el servidor SSH

```
service ssh restart
```

Máquina 2

Para poder acceder de una forma sencilla al servidor con el usuario "gitolite" vamos a crear el par de claves (pública y privada).

```
fontelearn@pc$ sudo su - fontelearn
fontelearn@pc$ cd ~/.ssh
fontelearn@pc$ ssh-keygen -t rsa -f gitolite
```

Podemos ver las dos claves (pública y privada) que acabamos de crear

```
fontelearn@pc$ ls -l ~/.ssh/gitolite*
```

“gitolite” es la clave privada y“gitolite.pub” es la clave pública.

Ahora vamos a copiar la clave pública al servidor, al directorio raíz del usuario "fontelearn".

```
fontelearn@pc$ scp ~/.ssh/gitolite.pub fontelearn@192.168.0.235:
```

Máquina 1

En el servidor vamos a mover la clave al directorio raíz del usuario "gitolite" y vamos a cambiarle los permisos.

```
root@servidor$ sudo su - fontelearn
fontelearn@servidor$ sudo mv gitolite.pub /home/gitolite
fontelearn@servidor$ sudo chown gitolite:gitolite
/home/gitolite/gitolite.pub
```

4.3.2 Instalar Git

A continuación vamos a comprobar si Git está instalado en las dos máquinas, y en caso de que no esté instalado lo vamos a instalar.

Primero comprobamos que no esté instalado, ejecutando el comando

```
git
```

Si la salida es del tipo

```
bash: git: no se encontró la orden
```

Tenemos que instalar Git. Para ello ejecutamos

```
fontelearn@servidor$ sudo apt-get install git
```

O

```
fontelearn@pc$ sudo yum install git
```

4.3.3 Clonar el código

A continuación, en la máquina 1 vamos a acceder con el usuario "gitolite"

```
fontelearn@servidor$ sudo su - gitolite
```

Y descargar el código

```
gitolite@servidor$ git clone git://github.com/sitaramc/gitolite
```

Vamos a instalar el software “gitolite” en el directorio "bin" de la raíz del usuario "gitolite", por lo que tenemos que crearlo

```
gitolite@servidor$ mkdir bin
```

Y ejecutamos el instalador

```
gitolite@servidor$ gitolite/install -to /home/gitolite/bin
```

Ahora vamos a ejecutar la configuración del servidor gitolite, pasándole como parámetro la clave pública que creamos en el otro equipo y que transferimos por scp.

```
gitolite@servidor$ /home/gitolite/bin/gitolite setup -pk
gitolite.pub
```

Máquina 2

Regresamos a la máquina 2, y con el usuario "fontelearn" creamos un alias ssh, añadiendo un archivo "config" en la carpeta "~/ssh/"

```
root@pc$ sudo su - fontelearn
fontelearn@pc$ vim ~/.ssh/config
```

En ese archivo introducimos

```
Host gitbox
   User gitolite
   Hostname 192.168.0.235
   Port 22
   IdentityFile ~/.ssh/gitolite
```

Donde especificamos que vamos a usar el usuario “gitolite”, la IP de la máquina 1 y la clave privada “~/.ssh/gitolite” para acceder a ese equipo.

4.3.4 Crear usuarios

La forma que usar gitolite para gestionar repositorios, usuarios y permisos de acceso es a través de un repositorio Git llamado "gitolite-admin".

Lo que vamos a hacer es clonarlo, realizar cambios, ejecutar los commits correspondientes y los push para que la configuración tenga efecto.

Empezamos clonando el repositorio

```
fontelearn@pc$ cd ~/proyectos/
fontelearn@pc$ git clone gitbox:gitolite-admin
```

Si echamos un vistazo en lo que tiene dentro

```
fontelearn@pc$ cd gitolite-admin/
fontelearn@pc$ ls -l
```

Podemos ver que tiene un directorio donde va a guardar las claves públicas de acceso (keydir) y otro donde va a almacenar la configuración (conf).

4.3.4.1 Claves de acceso

A continuación creamos los usuarios "usuario" y "usuario2" y sus pares de claves.

Máquina 2

Empezamos por el usuario "usuario"

```
fontelearn@pc$ sudo adduser usuario
fontelearn@pc$ sudo su - usuario
usuario@pc$ mkdir ~/.ssh
usuario@pc$ cd ~/.ssh
usuario@pc$ ssh-keygen -t rsa -f usuario
usuario@pc$ exit
```

Luego hacemos lo mismo para el "usuario2"

```
fontelearn@pc$ sudo adduser usuario
fontelearn@pc$ sudo su - usuario2
usuario2@pc$ mkdir ~/.ssh
usuario2@pc$ cd ~/.ssh
usuario2@pc$ ssh-keygen -t rsa -f usuario2
usuario2@pc$ exit
```

A continuación copiamos las dos claves públicas que acabamos de crear al directorio de claves del repositorio de administración

```
fontelearn@pc$ sudo cp /home/usuario/usuario.pub
/home/fontelearn/gitolite-admin/keydir/
fontelearn@pc$ sudo cp /home/usuario2/usuario2.pub
/home/fontelearn/gitolite-admin/keydir/
```

Luego las añadimos al repositorio

```
fontelearn@pc$ cd /home/fontelearn/gitolite-admin/
fontelearn@pc$ git add .
```

Ejecutamos el commit

```
fontelearn@pc$ git commit -m "Añado las claves de dos usuarios"
```

Y llevamos los cambios al servidor, que está en la "máquina 1".

```
fontelearn@pc$ git push
```

4.3.5 Repositorios y privilegios

Una vez que tenemos creados los usuarios vamos a ver cómo gestionar los repositorios existentes y como gestionar los permisos.

Para ello accedemos a la carpeta "conf" del repositorio

```
fontelearn@pc$ cd /home/fontelearn/gitolite-admin/conf/
```

Y vemos el contenido del archivo de configuración "gitolite.conf"

```
fontelearn@pc$ cat gitolite.conf
```

```
repo gitolite-admin
    RW+     =   gitolite

repo testing
    RW+     =   @all
```

Tenemos 2 repositorios:

- gitolite-admin, en el que el usuario "gitolite" tiene acceso completo.
- testing, al que tienen acceso completo todos los usuarios.

Las iniciales de los permisos significan:

- R, para solo lectura.
- RW, permite el push en ref existentes o crear nuevas ref.
- RW+, permite "push -f" o borrado de ref.
- - (el signo menos), para denegar el acceso.

Editamos el archivo para crear dos repositorios más, "pruebas" y "web_wordpress"

```
fontelearn@pc$ vim gitolite.conf
```

```
@admin      = usuario
 @devel      = usuario usuario2

 repo gitolite-admin
     RW+     = gitolite @admin

 repo testing
     RW+     = @all

 repo pruebas
     RW+     = gitolite @admin
```

```
        R       = @devel


    repo web_wordpress
        RW+     = @devel
```

Podemos ver que hemos creado dos grupos, que empiezan con el carácter "@":

- @admin, con el usuario "usuario".
- @devel, con los usuarios "usuario" y "usuario2".

Tenemos 2 repositorios más:

- Pruebas, en el que:
  - El usuario "gitolite" y el grupo "admin" tienen permiso completo.
  - El grupo "devel" tiene permiso de lectura.
- web_wordpress, en el que:
  - El grupo "devel" tiene permiso completo.

Además hemos dado al grupo @admin permiso completo al repositorio “gitolite-admin”.

Solo nos queda por añadir los cambios al repositorio y ejecutar el commit

```
fontelearn@pc$ git commit -am "Añado el repositorio pruebas y hago
cambios en los permisos"
```

Y llevamos los cambios al servidor, que está en la "máquina 1".

```
fontelearn@pc$ git push
```

4.4 Usando los repositorios

Ahora ya podemos empezar a trabajar con los repositorios. Para ello, en la máquina 1 accedemos con el usuario "usuario", clonamos el repositorio "web_wordpress", añadimos un elemento y enviamos los cambios al servidor:

```
fontelearn@pc$ sudo su - usuario
```

Editamos el archivo "~/.ssh/config"

```
usuario@pc$ vi ~/.ssh/config
```

E introducimos la siguiente información (solo hay que hacerlo la primera vez).

```
Host gitbox
Hostname 192.168.0.235
User gitolite
Port 22
```

```
    IdentityFile ~/.ssh/usuario
```

Ahora podemos clonar el repositorio "web_wordpress"

```
    usuario@pc$ git clone gitbox:web_wordpress
    usuario@pc$ cd web_wordpress
```

Creamos dos archivos

```
    usuario@pc$ touch index.php
    usuario@pc$ touch style.css
```

Los añadimos al repositorio

```
    usuario@pc$ git add .
```

Realizamos el commit

```
    usuario@pc$ git commit -m "commit inicial"
```

Y enviamos los cambios al servidor

```
    usuario@pc$ git push
    usuario@pc$ exit
```

Hago algo similar con el usuario "usuario2".

```
    fontelearn@pc$ sudo su - usuario2
```

Editamos el archivo "~/.ssh/config"

```
    usuario2@pc$ vi ~/.ssh/config
```

E introducimos la siguiente información (solo hay que hacerlo la primera vez).

```
    Host gitbox
    Hostname 192.168.0.235
    User gitolite
    Port 22
    IdentityFile ~/.ssh/usuario2
```

Ahora podemos clonar el repositorio "web_wordpress"

```
usuario2@pc$ git clone gitbox:web_wordpress
usuario2@pc$ cd web_wordpress
```

Vemos los archivos que hay

```
usuario2@pc$ ls -la
```

Introducimos un par de cambios en los archivos

```
usuario2@pc$ echo "Hola" >> index.php
usuario2@pc$ echo "Archivo CSS" >> style.css
```

Hacemos el commit

```
usuario2@pc$ git commit -am "Añado una línea a index.php y a
style.css"
```

Y llevo los cambios al servidor remoto

```
usuario2@pc$ git push
usuario2@pc$ exit
```

Volvemos a acceder con el usuario “usuario” y traemos los últimos cambios

```
fontelearn@pc$ sudo su - usuario
usuario@pc$ cd web_wordpress
usuario@pc$ git pull
usuario@pc$ ls index.php
usuario@pc$ ls style.css
```