Laboratorio de Sistemas Operativos y Redes.

Administrar un repositorio Git con Gogs

Alumnos:

Leandro Antunez

Mariano Verdecanna.

1er Cuatrimestre, 2018.

Sumario

Introducción

Gogs es un servicio de *Git* libre y de código abierto escrito en el lenguaje *Go*. Este servicio permite crear y ejecutar un servidor Git con un hardware de bajos recursos. Provee una interfaz web similar a *GitHub*, y ofrece soporte para bases de datos *MySQL*, *PostgreSQL* y *SQLite*.

En esta documentación, se mostrará paso a paso cómo instalar y configurar un servicio Git usando Gogs en *Ubuntu 16.04* (tambien valido para *Debian*). Cubrirá detalles que incluyen cómo instalar el sistema *Go* en *Ubuntu*, utilizando *apache2* y *MySQL*.

Instalación paso a paso

Requisitos

- Ubuntu 16.04 / Debian
- MySQL
- Apache2

Lo que queremos hacer es servir ‘Gogs’ desde un dominio virtual, utilizando Apache2 como servidor web y MySQL como servidor de base de datos.

Nuevo Usuario

Primero se crea un usuario para que corra la aplicación:

```
$ sudo adduser --disabled-login --gecos 'Gogs' git
```

Se va a instalar Go dentro del home del usuario que se acaba de crear para no interferir con ninguna otra versión de la distribución que se use.

Accedemos al usuario creado recientemente:

```
$ sudo su - git
```

Creamos un directorio:

```
$ cd ~
$ mkdir local
```

De esta manera, Se utilizará el directorio */home/git/local/go.*

Descarga e instalación de Go

Se descargara la versión disponible:

```
$ wget https://storage.googleapis.com/golang/go1.10.3.linux-amd64.tar.gz
```

Se descomprime:

```
$ tar -C /home/git/local -xzf go1.7.3.linux-amd64.tar.gz
```

Con el usuario git, se agregan unas cuantas rutas al fichero bashrc:

```
$ sudo su - git
$ cd ~
$ echo 'export GOROOT=$HOME/local/go' >> $HOME/.bashrc
$ echo 'export GOPATH=$HOME/go' >> $HOME/.bashrc
$ echo 'export PATH=$PATH:$GOROOT/bin:$GOPATH/bin' >> $HOME/.bashrc
$ source $HOME/.bashrc
```

Instalar Gogs

Se instalará Gogs rápidamente de esta manera:

```
$ go get -u github.com/gogs/gogs
$ cd $GOPATH/src/github.com/gogs/gogs
$ go build
```

Y una vez compilado, se puede probar con:

```
$ cd $GOPATH/src/github.com/gogs/gogs
$ ./gogs web
```

Configuración

La configuración por defecto está en conf/app.ini pero no se va a editar ese fichero. En lugar de eso Gogs nos permite crear una copia que será donde pondremos nuestras modificaciones.

Se empieza creando unos directorios:

```
mkdir -p $GOPATH/src/github.com/gogits/gogs/custom/conf
mkdir -p ~/gogs-repositories
sudo mkdir -p /var/log/gogs
sudo chown git:git /var/log/gogs
```

Hacemos una copia del fichero de configuración original, en donde haremos nuestros cambios:

```
cd $GOPATH/src/github.com/gogs/gogs
cp conf/app.ini custom/conf/
```

Cuando haya que actualizar la versión de Gogs, la configuración ‘custom’ no se modificará, ya que tienen un .gitignore para este fichero local. Por esta razón, se puede borrar la información por defecto.

Se ingresa al directorio del fichero:
`cd $GOPATH/src/github.com/gogs/gogs/conf/`

Y se puede modificar para que quede de una manera similar a esta:

```
; Change it if you run locally
RUN_USER = git
; Either "dev", "prod" or "test", default is "dev"
RUN_MODE = prod


[repository]
ROOT = /home/git/gogs-repositories

[server]
PROTOCOL                         = http
DOMAIN                           = localhost
ROOT_URL                         = %(PROTOCOL)s://%(DOMAIN)s:%(HTTP_PORT)s/
HTTP_ADDR                        = 0.0.0.0
HTTP_PORT                        = 4000
; Permission for unix socket
UNIX_SOCKET_PERMISSION           = 666
```

```
[database]
; Either "mysql", "postgres" or "sqlite3", it's your choice
DB_TYPE  = mysql
HOST     = 127.0.0.1:3306
NAME     = gogs
USER     = root
PASSWD   = admin
; For "postgres" only, either "disable", "require" or "verify-full"
SSL_MODE = disable
; For "sqlite3" and "tidb", use absolute path when you start as service
PATH     = data/gogs.db
```

```
[security]
INSTALL_LOCK                       = true
; !!CHANGE THIS TO KEEP YOUR USER DATA SAFE!!
; ; #@FDEWREWR&*(
SECRET_KEY                         = 1SWoEM1I8hvpcI3
; Auto-login remember days
LOGIN_REMEMBER_DAYS                = 7
COOKIE_USERNAME                    = gogs_awesome
COOKIE_REMEMBER_NAME               = gogs_incredible
COOKIE_SECURE                      = false
; Reverse proxy authentication header name of user name
REVERSE_PROXY_AUTHENTICATION_USER = X-WEBAUTH-USER
; Enable to set cookie to indicate user login status
ENABLE_LOGIN_STATUS_COOKIE         = false
LOGIN_STATUS_COOKIE_NAME           = login_status

[service]
ACTIVE_CODE_LIVE_MINUTES       = 180
RESET_PASSWD_CODE_LIVE_MINUTES = 180
; User need to confirm e-mail for registration
REGISTER_EMAIL_CONFIRM         = false
; Does not allow register and admin create account only
DISABLE_REGISTRATION           = true
; User must sign in to view anything.
REQUIRE_SIGNIN_VIEW            = false
; Mail notification
ENABLE_NOTIFY_MAIL             = false
ENABLE_CAPTCHA                 = true
```

```
[log]
MODE      = file
LEVEL     = Info
ROOT_PATH = /home/git/go/src/github.com/gogs/gogs/log
```

Configurar Apache2 con Gogs

En Debian podemos instalar Apache2 con el siguiente comando:
`sudo apt-get install apache2`

En nuestro caso, ya lo teníamos instalado de la cursada.

La instalación de Gogs que acabamos de hacer corre en el puerto 3000 por defecto.
Nosotros lo que hicimos fue configurarlo en el puerto 4000.

Lo que queremos hacer es utilizar un servidor web como proxy para poder servir Gogs desde el puerto 80, como un subdominio de nuestro dominio, por ejemplo en este caso:

- http://gitlabo.unq

La mejor forma de hacer esto es crear un fichero de configuración para hacer un VirtualHost y usar mod_proxy para poder servir Gogs en el puerto 80 con Apache.

```
<VirtualHost *:80>
  ServerName gitlabo.unq

  ProxyPreserveHost On
  ProxyPass         / http://localhost:4000/
  ProxyPassReverse / http://localhost:4000/

  ErrorLog ${APACHE_LOG_DIR}/gogs-error.log
  CustomLog ${APACHE_LOG_DIR}/gogs-access.log combined

</VirtualHost>
```

El nombre del dominio que usemos acá será el mismo que pondremos en la configuración de Gogs. Falta activar este VirtualHost:

```
$ sudo ln -s /etc/apache2/sites-available/080-gogs.conf /etc/apache2/sites-enabled/
```

Ahora activaremos el módulo mod_proxy de Apache. Podemos probar con a2enmod:

```
$ sudo a2enmod proxy
```

Esto debería crear los dos ficheros necesarios dentro de */etc/apache2/mods-enabled/ : proxy.conf y proxy.load*

Reiniciar Apache.

```
$ sudo service apache2 restart
```

Configurar MySQL

Ya teníamos MySQL de la cursada, o incluso de antes. En principio alcanza con crear una base de datos nueva y asignar todos los permisos a un usuario para Gogs:

```
CREATE DATABASE gogs CHARACTER SET utf8 COLLATE
utf8_general_ci;
GRANT ALL PRIVILEGES ON gogs.* To 'gogs'@'localhost'
IDENTIFIED BY 'password';
```

Ejecutar el instalador

```
$ cd $GOPATH/src/github.com/gogs/gogs
```

```
~/go/src/github.com/gogs/gogs $
~/go/src/github.com/gogs/gogs $ ./gogs web
```

Si ahora usamos un navegador para acceder a la dirección que hayamos configurado en nuestro Apache, deberíamos acceder a la primera pantalla del instalador, que nos guiará por la configuración básica. En nuestro caso sería:

- http://gitlabo.unq

Añadir Gogs a init.d

Llegados a este punto todo debería estar funcionando, sin embargo necesitamos arrancar Gogs a mano. Lo que necesitamos es poder levantar y parar Gogs como un servicio más de nuestra máquina, y además, que este arranque automáticamente al levantar o reiniciar el servidor.

Gogs viene con una plantilla que nos facilita mucho las cosas (para Debian / Ubuntu).

Está aquí:

```
$GOPATH/src/github.com/gogs/gogs/scripts/init/debian/gogs
```

La copiamos al directorio de init.d:

```
$ sudo cp
$GOPATH/src/github.com/gogs/gogs/scripts/init/debian/gogs
/etc/init.d/gogs
```

Y ahora modificamos un par de cosas. Editaremos las dos primeras líneas:

```
# Required-Start:    $syslog $network
# Required-Stop:     $syslog
```

De esta manera:

```
# Required-Start:    $syslog $network $local_fs apache2
mysql
# Required-Stop:     $syslog $local_fs
```

Y nos aseguramos de que la variable WORKINGDIR apunte al directorio correcto:

```
WORKINGDIR=/home/git/go/src/github.com/gogs/gogs
```

El script necesita permisos de ejecución, y lo añadimos a la configuración de inicio:

```
$ sudo chmod ug+x /etc/init.d/gogs
$ sudo update-rc.d gogs defaults 30 70
```

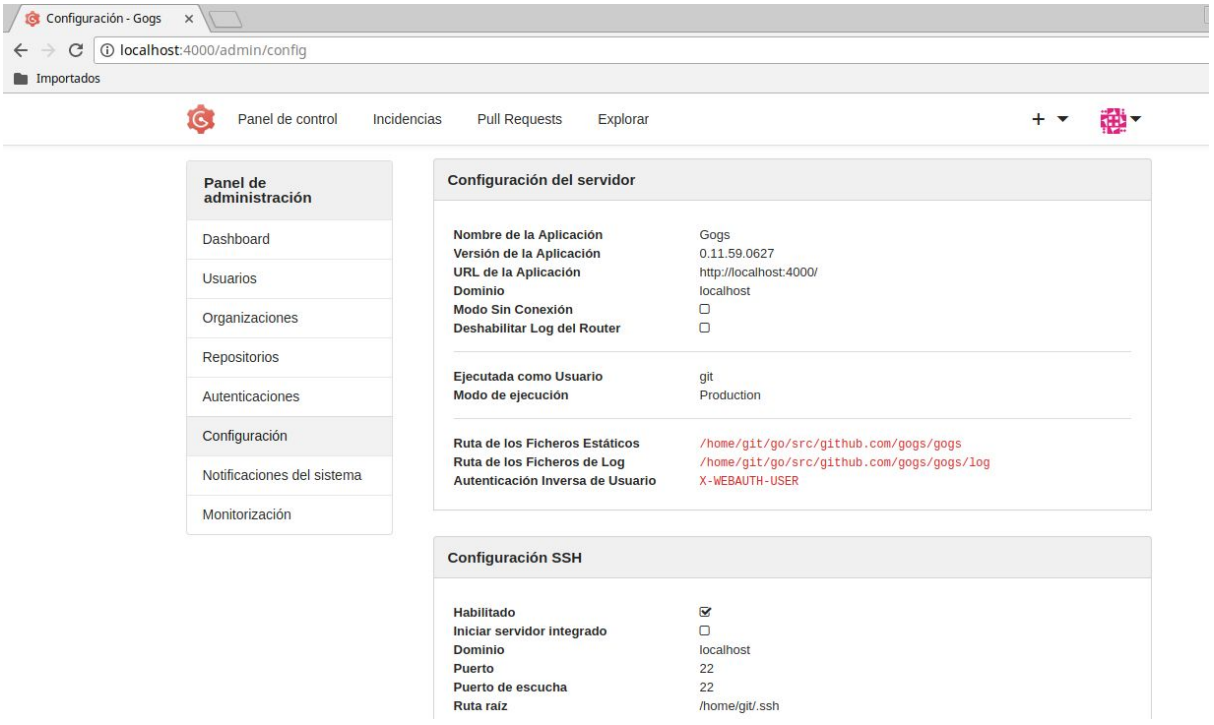
Ahora podemos probar nuestra configuración y levantar el servicio gogs:

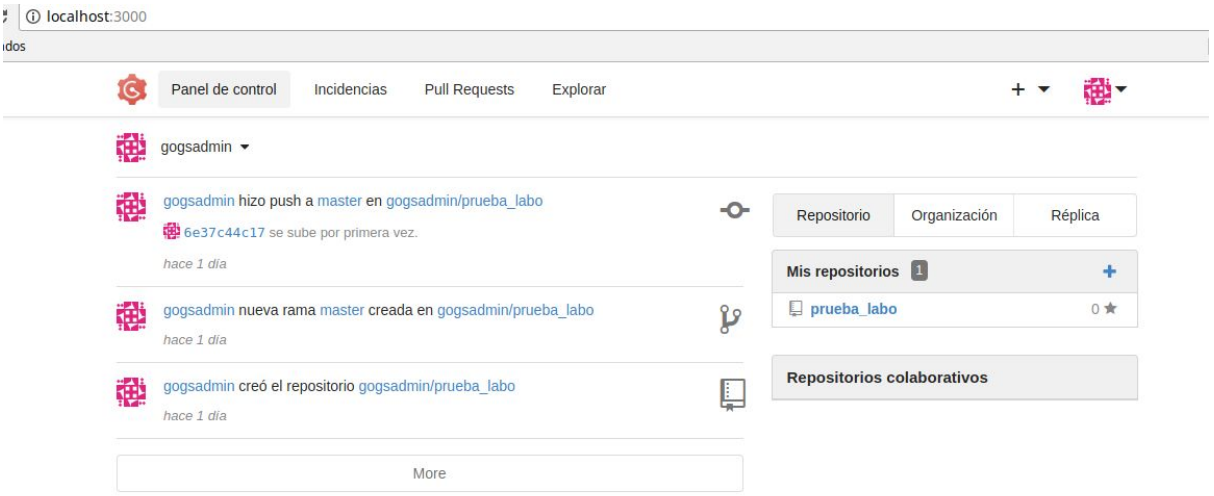
```
~/go/src/github.com/gogs/gogs $
~/go/src/github.com/gogs/gogs $ sudo service gogs start
```

```
git@mariano-Inspiron-3421 ~/go/src/github.com/gogs/gogs $ sudo service gogs status
[sudo] password for git:
● gogs.service - LSB: A self-hosted Git service written in Go.
   Loaded: loaded (/etc/init.d/gogs; bad; vendor preset: enabled)
   Active: active (running) since jue 2018-07-12 02:08:38 -03; 10h ago
     Docs: man:systemd-sysv-generator(8)
  Process: 7856 ExecStop=/etc/init.d/gogs stop (code=exited, status=0/SUCCESS)
  Process: 7866 ExecStart=/etc/init.d/gogs start (code=exited, status=0/SUCCESS)
    Tasks: 7 (limit: 512)
   CGroup: /system.slice/gogs.service
           └─7878 /home/git/go/src/github.com/gogs/gogs/gogs web

jul 12 02:08:38 mariano-Inspiron-3421 systemd[1]: Starting LSB: A self-hosted Git service written in Go....
jul 12 02:08:38 mariano-Inspiron-3421 gogs[7866]:  * Starting Gogs gogs
jul 12 02:08:38 mariano-Inspiron-3421 gogs[7866]:    ...done.
jul 12 02:08:38 mariano-Inspiron-3421 systemd[1]: Started LSB: A self-hosted Git service written in Go..
git@mariano-Inspiron-3421 ~/go/src/github.com/gogs/gogs $
```

Así quedó la funcionando:





Problemáticas

A lo largo del trabajo nos encontramos con distintas problemáticas:

- Tuvimos que darle permisos de administrador al nuevo usuario “git”, ya que no nos permitía hacer algunas cosas del tutorial por este motivo.
- Nuestro mayor problema y con lo que perdimos mucho tiempo, fue con poder configurar el VirtualHost, nunca pudimos levantar la aplicación desde el navegador con un dominio personalizado, como se ve en las capturas lo hicimos con localhost y con el puerto 4000 (esto si lo pudimos cambiar, ya que por defecto lo levantaba en el 3000)

Links

- https://gogs.io
- https://www.howtoforge.com/tutorial/how-to-install-gogs-go-git-service-on-ubuntu-1604/
- https://www.carloscarrascal.com/blog/servidor-de-git-privado-con-gogs
- https://geekytheory.com/como-configurar-un-virtual-host-de-apache-en-linux