

# Capítulo 6

## Servicios de directorio, LDAP

### Introducción

Un **servicio de directorio** es una aplicación o un conjunto de aplicaciones que almacena y organiza la información sobre los usuarios de una red de ordenadores, sobre recursos de red, y permite a los administradores gestionar el acceso de usuarios a los recursos sobre dicha red. Además, los servicios de directorio actúan como una capa de abstracción entre los usuarios y los recursos compartidos.

**LDAP** (Lightweight Directory Access Protocol o Protocolo Ligero de Acceso a Directorios) es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también se considera una base de datos (aunque su sistema de almacenamiento puede ser diferente) a la que pueden realizarse consultas.

Gracias a los servicios de directorio podemos centralizar la **autenticación de usuarios** sobre un servidor: para la validación de los usuarios del sistema o de servicios como usuarios ftp, moodle o un proxy. Otra de las cosas que se puede implementar gracias a los servicios de directorio son los **perfiles móviles** (carga de los ficheros de un usuario independiente desde el ordenador de donde nos conectemos).

### Historia del LDAP

En la década de los 80 comenzaron a utilizarse los servicios de directorio como X.500, sobre todo por las compañías de telefonía para almacenar los datos de sus clientes, algo parecido a lo que podemos conocer como páginas amarillas.

Los servicios de X.500 se accedía mediante el protocolo **DAP** (Directory Access Protocol) que utilizaba la pila de protocolos **OSI**, lo que hacía que los clientes y servidores se comunicasen mediante utilizando el protocolo OSI lo que hacía que fuese muy pesado.

LDAP en lugar de utilizar todo el modelo OSI utilizó el modelo **TCP/IP**, en concreto el puerto **389/TCP** lo que hizo que fuese más ligero.

### LDAP frente a NIS

**NIS** (Network Information Service o Servicio de Información de Red) es un servicio que fue desarrollado por Sun Microsystems para sistemas Linux y se utiliza para la resolución de nombres y la distribución de los datos de configuración en sistemas distribuidos tales como nombres de usuarios y nombres de equipos. NIS permite distribuir la información contenida en los archivos `/etc/passwd` y `/etc/groups` de un servidor a todos los demás equipos de la red, haciendo que la misma parezca transparente para los usuarios; es decir, que toda la red se comporte como si fuera un solo equipo. Antiguamente se llamaba páginas amarillas que tubo que dejarse de utilizar por ser una marca registrada por British Telecom. Fue desarrollado exclusivamente para plataformas Unix.

LDAP además de Unix está dirigido a otras plataformas como Windows (a partir de la versión Windows 2000 server admite LDAP como servicio de directorio) y Novell.

## Funciones de LDAP

Estas son algunas de las funciones que podemos aplicar con LDAP:

- Empleo como sustituto para el servicio NIS.
- Autenticación de usuarios de aplicaciones web.
- Autenticación de usuarios de sistemas operativos.
- Autenticación de usuarios con NFS en redes Unix.
- Autenticación de usuarios con Samba en redes heterogéneas.
- Encaminamiento de correo (postfix, sendmail).
- Libretas de direcciones para clientes de correo como Mozilla, Evolution y Outlook.
- Administración de descripciones de zona para un servidor de nombres BIND9.

## Implementaciones de LDAP

LDAP se ha implementado en diferentes aplicaciones reales, las más conocidas y utilizadas son Active Directory y OpenLDAP.

**Active Directory:** Nombre utilizado por Microsoft, desde Windows Server 2000, para el almacenamiento centralizado de información. Es un servicio de directorio donde se almacena información de usuarios, recursos de la red, políticas de seguridad, configuración, asignación de permisos, etc.

**OpenLDAP:** Implementación del protocolo con licencia libre que incluye entre otros elementos un servidor y diferentes utilidades y herramientas para los clientes.

Además se ha implementado en otras aplicaciones como: Red Hat Directory Server, Apache Directory Server, Open DS, iPlanet (ahora Sun ONE Directory Server).

## El formato LDIF

LDIF (LDAP Data Interchange Format) es un formato que se utiliza para la importación y exportación de datos independientemente del servidor LDAP que se esté utilizando. Este formato es útil tanto para realizar copias de seguridad de los datos de un servidor LDAP, como para importar pequeños cambios que se necesiten realizar manualmente en los datos, siempre manteniendo la independencia de la implementación LDAP y de la plataforma donde esté instalada.

El formato LDIF es simplemente un formato de texto ASCII para entradas LDAP, que tiene la siguiente forma:

```
dn: <nombre distinguido>
<nombre_atributo>: <valor>
<nombre_atributo>: <valor>
<nombre_atributo>: <valor>
```

Ejemplo de una entrada para describir una cuenta de usuario en un servidor:

```
dn: uid=aitorla,ou=users,dc=profesordeinformatica,dc=com
uid: aitorla
cn: Aitor Lopez de Aberasturi
objectclass: account
objectclass: posixAccount
objectclass: top
loginshell: /bin/bash
uidnumber: 1001
gidnumber: 1001
homedirectory: /home/aitorla
gecos: Aitor Lopez de Aberasturi,,,,
userpassword: {crypt}eDgcded67Ddcawc
```

Estas son algunas de las reglas que hay que tener en cuenta en los LDIF:

- La primera entrada es siempre es el DN o nombre distinguido.
- Los atributos van a la izquierda y los valores a la derecha y están separados por : (dos puntos).
- El primer valor de DN debe estar en una línea posterior de la entrada en el ldif.
- El atributo debe tener su correspondiente entrada de atributo llamado objectClass que define el atributo y está relacionado con el schema.
- No se distinguen mayúsculas y minúsculas.
- El signo igual no tiene que tener espacios ni por delante ni por detrás.
- Los caracteres especiales se tienen que escapar (carácter \).
- En un archivo LDIF puede haber mas de una entrada definida, cada entrada se separa de las demás por una línea en blanco.

## Los atributos LDIF

Los atributos del LDAP pueden tener múltiples valores y pueden estar repetidos en el LDIF. Por ejemplo si tenemos varios números de teléfonos podríamos tener varios atributos **telephoneNumber**. Sin embargo algunos atributos solo pueden tener un único valor, esto lo define el esquema o schema. Como ejemplo de atributos con un único valor tendríamos **uidNumber** o **userPassword**.

El **esquema** o **schema** va a definir los atributos que podemos introducir en cada entrada LDAP y cuales se pueden repetir. En el directorio `/etc/ldap/schema/` tenemos unos esquemas predefinidos.

El atributo **objectClass** es uno de los atributos obligatorios y tiene que existir **por lo menos uno** en cada entrada. Cada atributo objectClass hace referencia al esquema y nos indica que atributos podemos introducir en cada entrada. En el ejemplo anterior el valor posixAccount hace referencia a usuarios Linux y por ello tenemos atributos como el directorio home, el uid, etc.

## Estructura del Arbol de Directorio (DIT)

**DIT** (Directory Information Tree ó Arbol de Información del Directorio) es la estructura de un servidor LDAP. Donde las ramas de arbol pueden ser contenedores o hojas.

Los **contenedores** pueden a su vez contener otros objetos. Tales clases de objetos son root (el elemento raíz del árbol de directorios, que no existe realmente), c (país), ou (unidad organizativa) y dc (componente de dominio). Este modelo es comparable con los directorios (carpetas) de un sistema de archivos.

Las **hojas** contienen la parte final de una rama y no contienen objetos. Algunos ejemplos serían person, InetOrgPerson o groupofNames.

Las clases de objeto que vamos a tener en nuestro arbol van a ser:

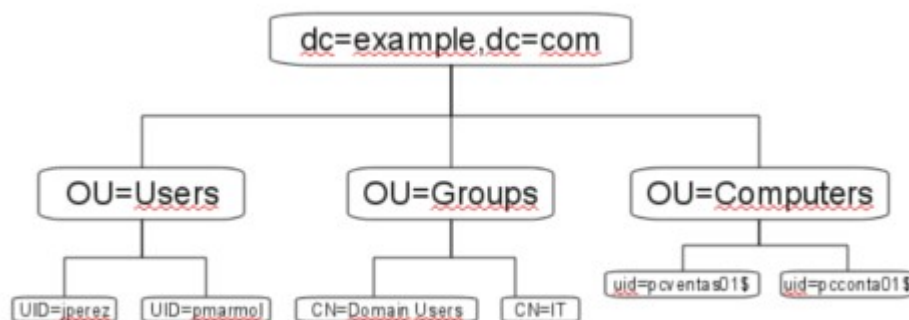
**dcObject:** Objeto domainComponent o componentes del nombre del dominio. Atributo obligatorio: dc

**organizationalUnit:** Unidad organizativa. Atributo ou.

**InetOrgPerson:** Datos relacionados con la persona para la intranet o Internet. Atributos sn y cn.

Cada entrada en el arbol posee un identificador único llamado Distinguished Name o DN. El primer paso para diseñar el DIT es definir el **Base DN**, el Base DN es el nivel más alto en el arbol de directorio, es decir, la base o raíz del directorio.

En nuestro caso el Base DN será definido utilizando los llamados **DC** ó Domain Components, similar a la estructura del Sistema de Nombres de Dominio (DNS).



En la figura de arriba se define:

**dc=profesordeinformatica,dc=com:** Raíz del directorio

**ou=Users:** Contenedor para almacenar cuentas de usuario para sistemas Linux/Unix y Windows

**ou=Computers:** Contenedor para las cuentas de Ordenadores para sistemas Windows (los de Linux serían Hosts).

**ou=Groups:** Contenedor para almacenar Grupos de sistema para sistemas Unix y Windows

## Funcionamiento del LDAP

Este es el funcionamiento del LDAP:

1. El cliente establece una sesión con el servidor LDAP. El cliente indica el servidor y el puerto en el que el servidor LDAP está escuchando, por defecto el **389**. El cliente puede proporcionar información de autenticación o establecer una sesión anónima con los accesos por defecto.

2. El cliente efectúa las operaciones sobre los datos. LDAP proporciona capacidades de búsqueda, lectura y actualización.

3. Una vez finalizadas las operaciones, el cliente cierra la sesión.

## Práctica: Instalación LDAP

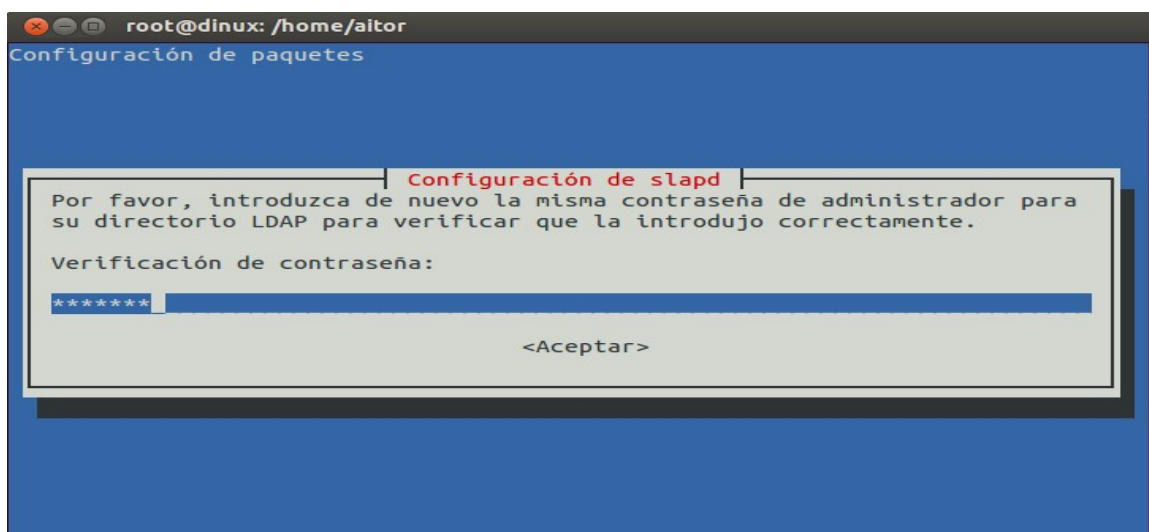
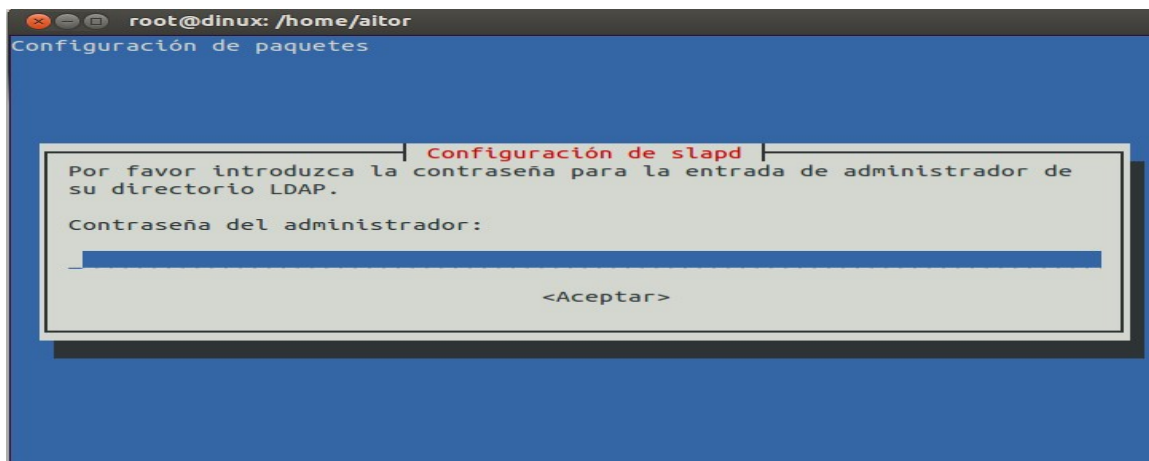
1.- Instalamos LDAP

```
$ sudo apt-get update
```

```
$ sudo apt-get install slapd ldap-utils
```

2.- Introducimos el password y confirmamos

En nuestro caso vamos a poner egibide



3.- Modificamos el fichero de configuración /etc/ldap/ldap.conf

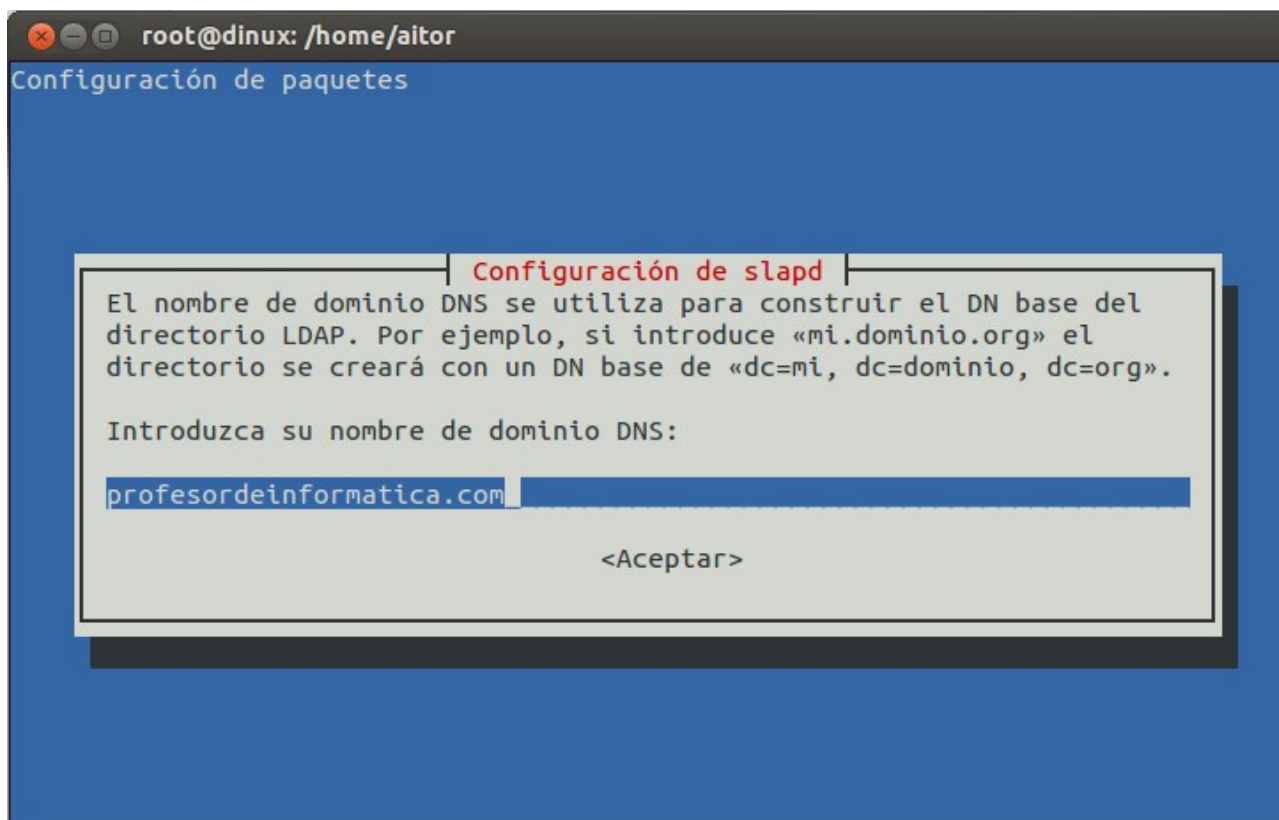
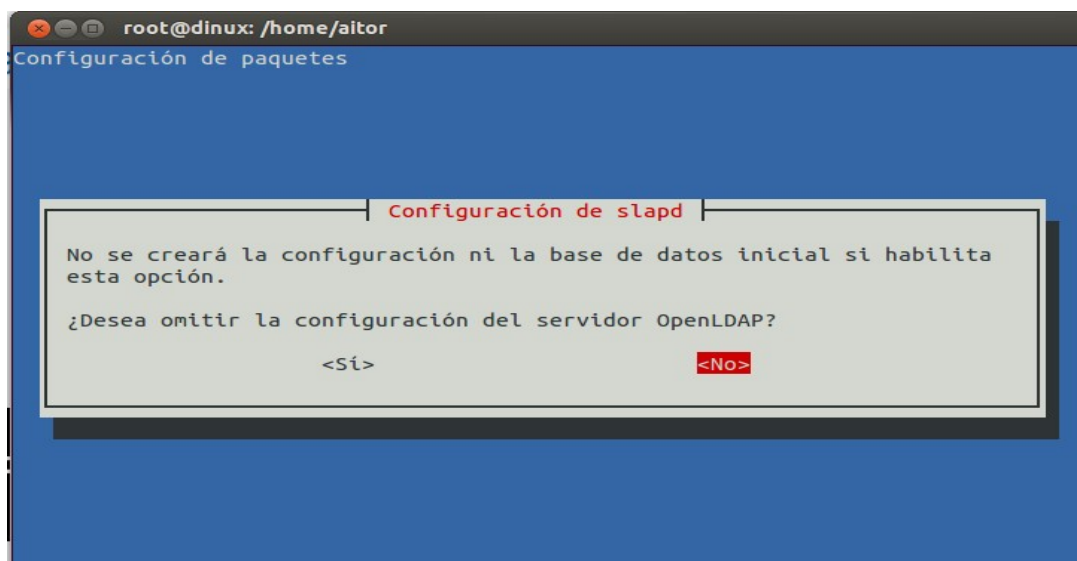
Descomentamos o añadimos la base y la URI: En la URI se puede poner el dominio o la dirección IP.

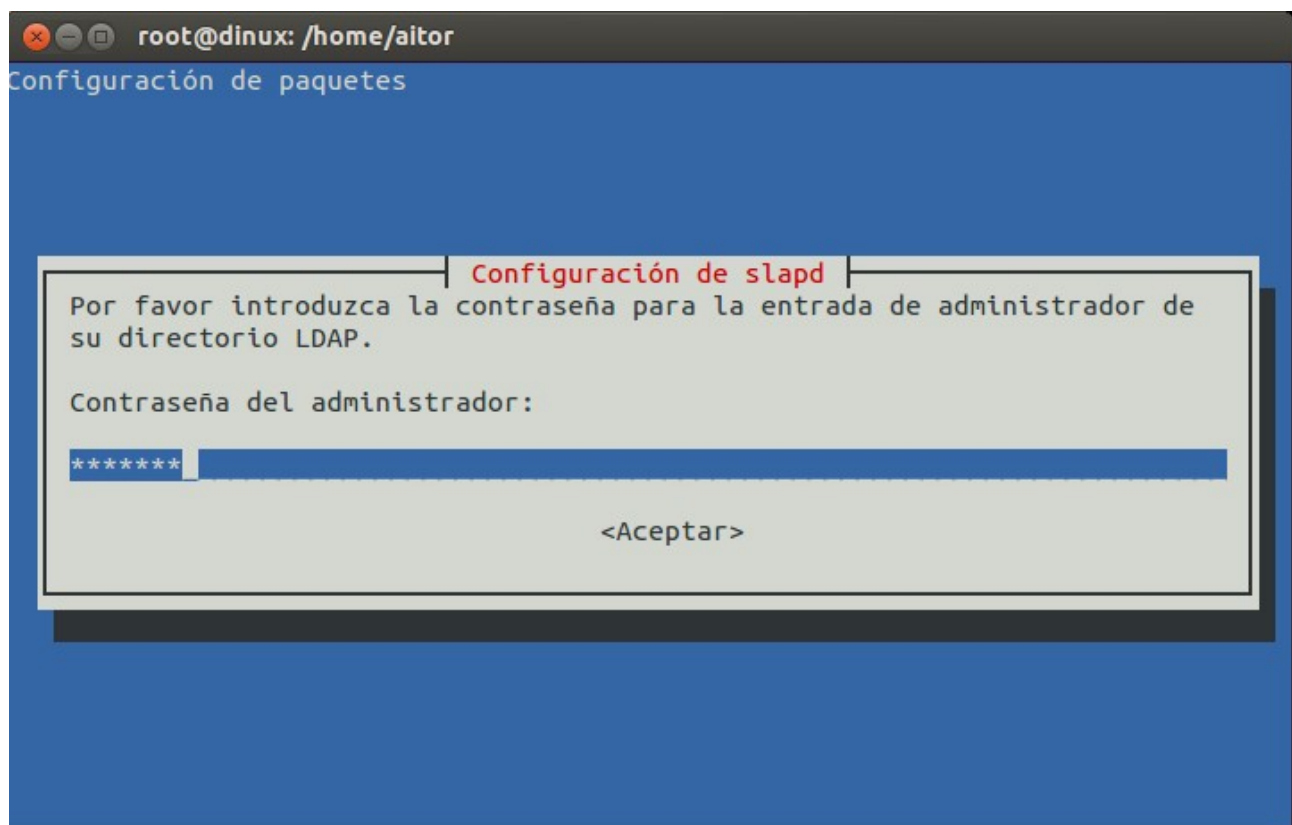
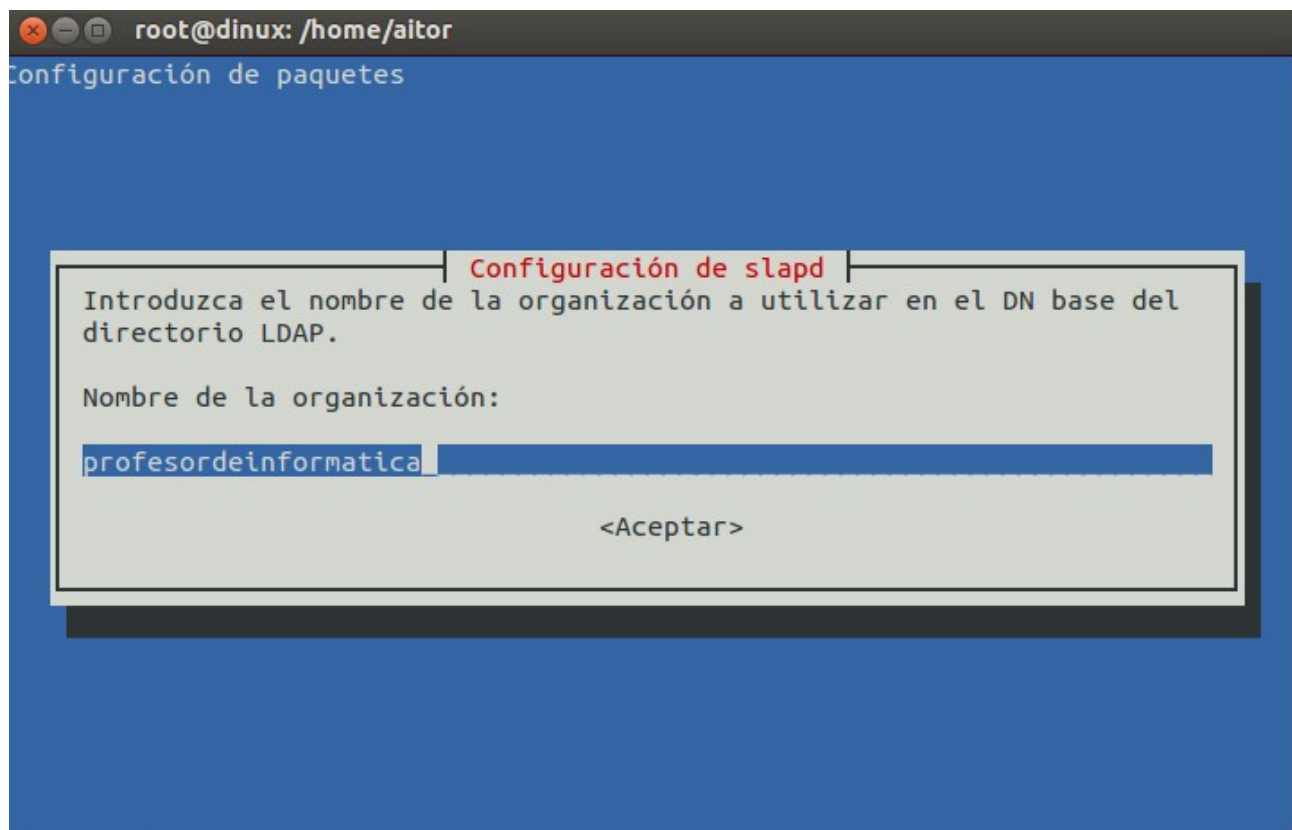
BASE dc=profesordeinformatica,dc=com

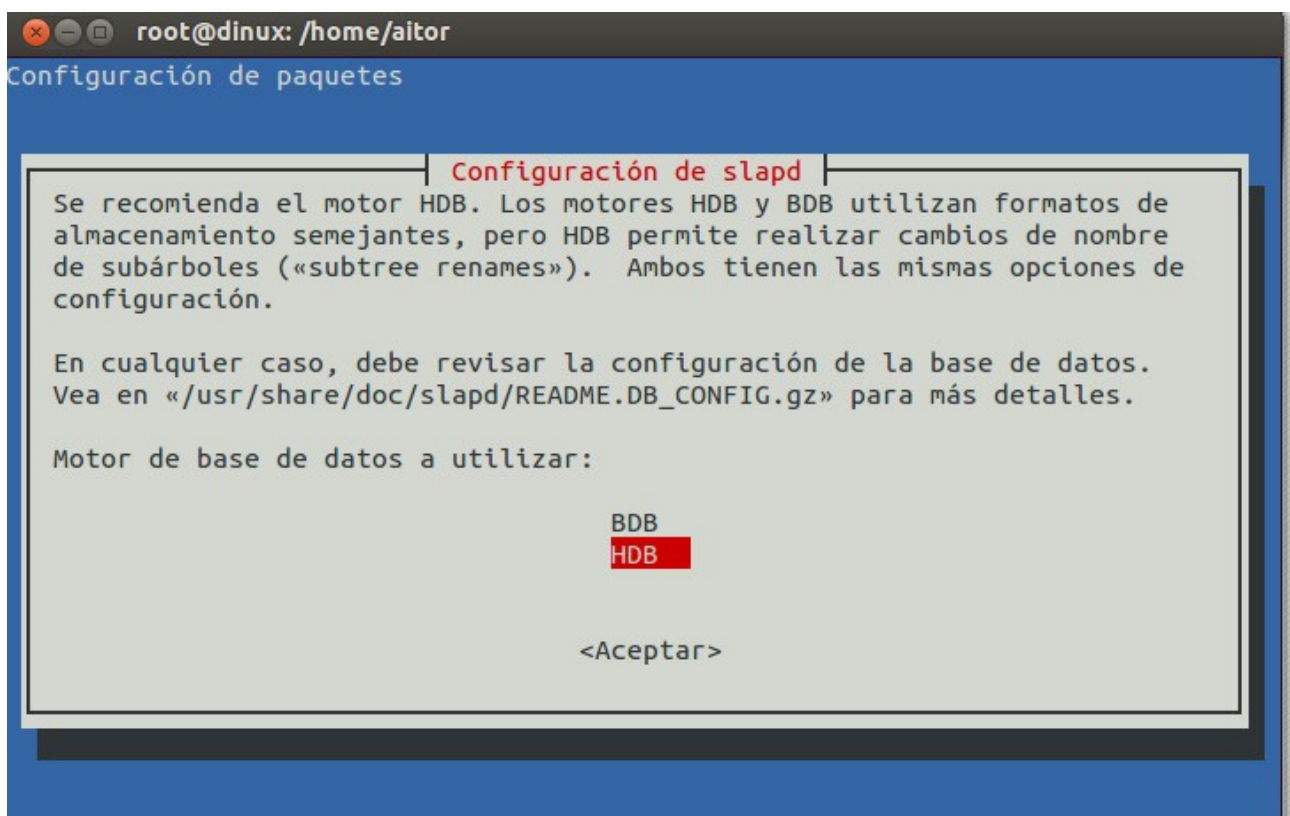
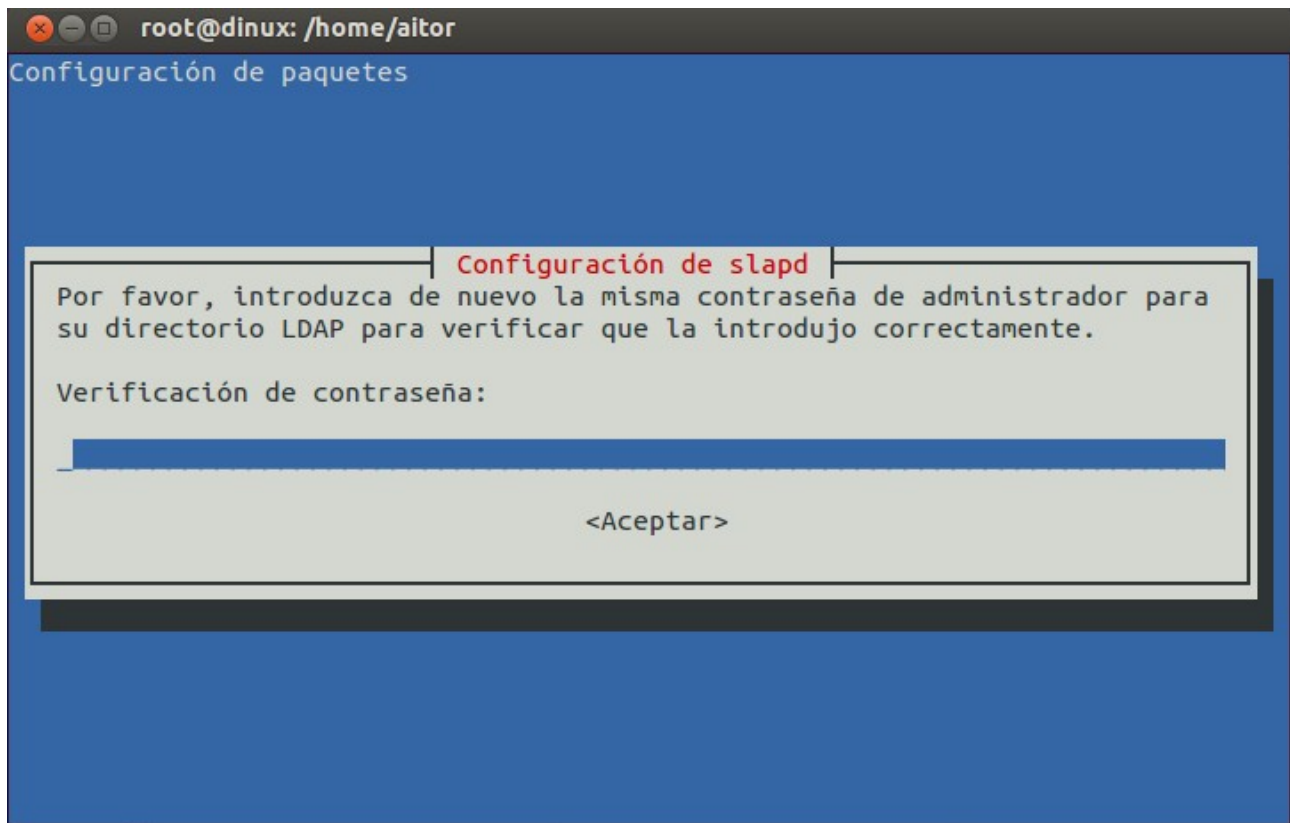
URI ldap://ldap.profesordeinformatica.com ldap://ldap-master.profesordeinformatica.com:666

#### 4.- Reconfiguramos el servicio slapd

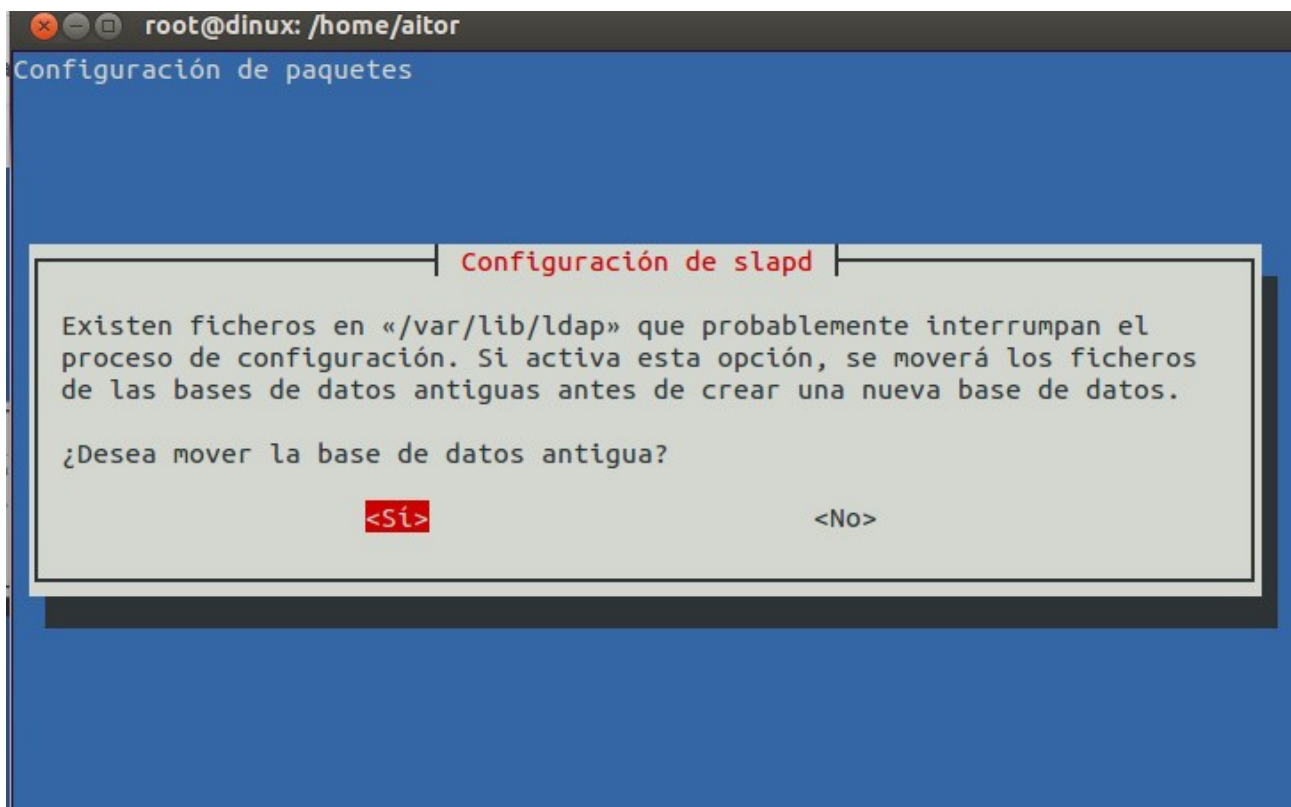
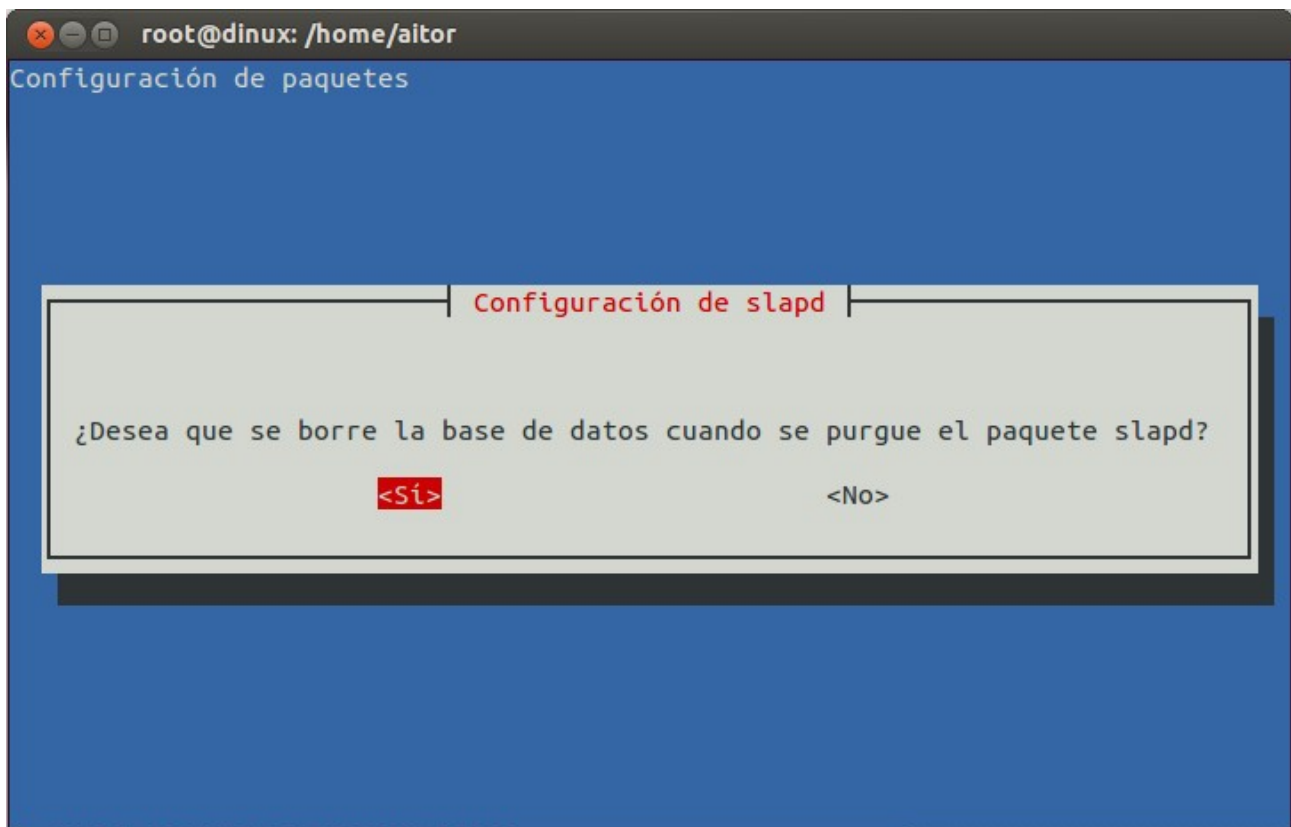
dpkg-reconfigure slapd

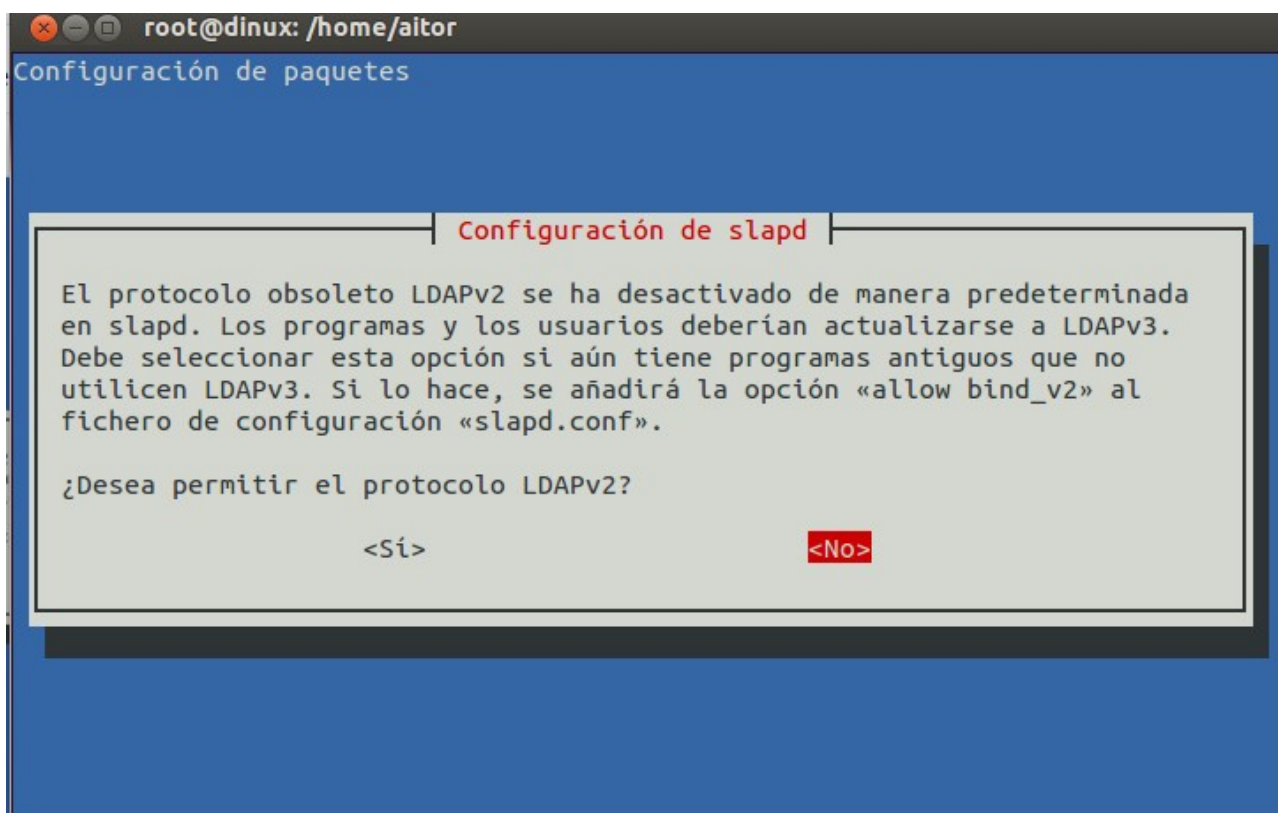












5.- Si no tenemos un servidor dns con la url ldap.profesordeinformatica.com editamos el fichero /etc/hosts

```
127.0.0.1 localhost ldap.profesordeinformatica.com
```

6.- Probamos

```
ldapsearch -x
# extended LDIF
#
# LDAPv3
# base <dc=profesordeinformatica,dc=com> (default) with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# profesordeinformatica.com
dn: dc=profesordeinformatica,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: profesordeinformatica
dc: profesordeinformatica
```

```
# admin, profesordeinformatica.com
dn: cn=admin,dc=profesordeinformatica,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
```

```
# search result
search: 2
result: 0 Success
```

```
# numResponses: 3
# numEntries: 2
```

## **Práctica: Agregar objetos al LDAP: Usuarios y Grupos.**

Algunos de los objetos más comunes que se administran en un servidor LDAP son grupos y personas. En estos ejemplos veremos cómo crear un par de unidades organizacionales (*OU*) y agregar un usuario a una *OU*:

### 1.- Añadimos unidades organizativas

Vamos añadir dos unidades organizacionales los usuarios y los grupos. Editamos el fichero **UO.Idif** con el siguiente contenido:

```
dn: ou=users,dc=profesordeinformatica,dc=com
objectClass: top
objectClass: organizationalUnit
ou: users
description: Usuarios
```

```
dn: ou=groups,dc=profesordeinformatica,dc=com
objectClass: top
objectClass: organizationalUnit
ou: groups
description: Grupos
```

Detenemos el servidor ldap  
/etc/init.d/slapd stop

Añadimos las unidades organizativas  
sudo slapadd -v -l UO.Idif

y nos devuelve

```
added: "ou=users,dc=profesordeinformatica,dc=com" (00000003)
added: "ou=groups,dc=profesordeinformatica,dc=com" (00000004)
```

```
_##### 100.00% eta  none elapsed      none fast!
```

## 2.- Añadimos un objeto persona

Editamos el fichero Usuario.ldif

```
dn: cn=test,ou=users,dc=profesordeinformatica,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
sn: apellido
cn: test
```

Detenemos el servidor ldap  
/etc/init.d/slaped stop

Añadimos las unidades organizativas  
sudo slapadd -v -l Usuario.ldif

y nos devuelve

```
added: "cn=test,ou=users,dc=profesordeinformatica,dc=com" (000000005)
```

```
_##### 100.00% eta  none elapsed      none fast!
```

## 3.- Reiniciamos el servidores

/etc/init.d/slaped restart

## 4.- Vemos que hemos añadido las unidades organizativas y el usuario concepto ldapsearch -x

# Práctica: Administración gráfica de LDAP desde phpldapadmin

Montamos el entorno LAMP

```
sudo apt-get install apache2 php5 php5-mysql
```

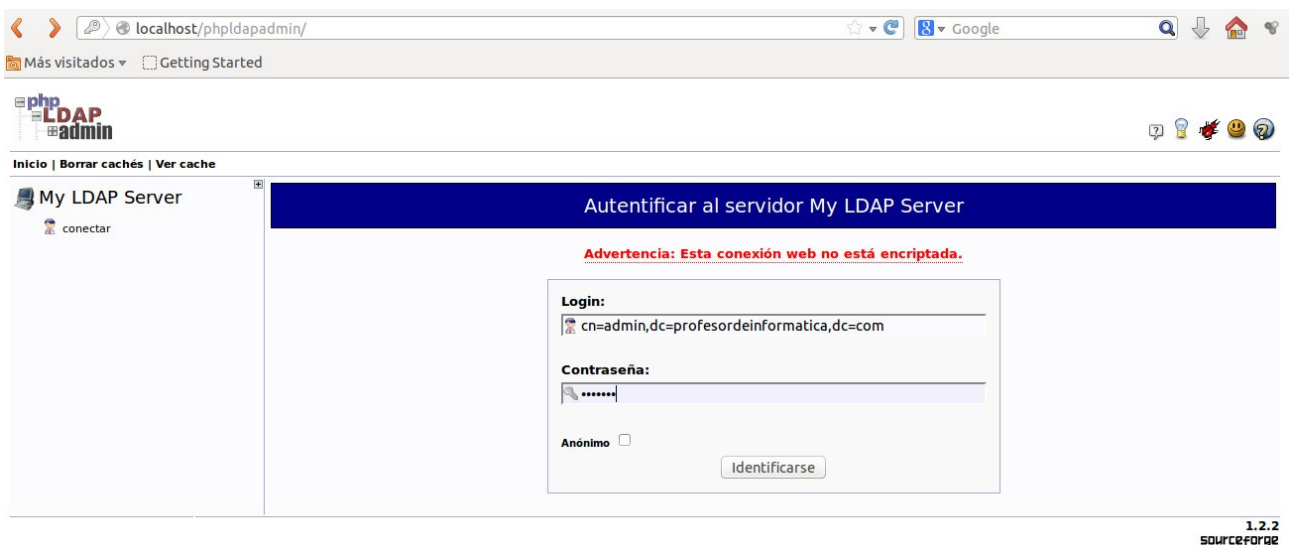
Instalamos el phpldapamin

```
sudo apt-get install phpldapadmin
```

Configuramos el fichero **/usr/share/phpldapadmin/config/config.php** con los siguientes valores en las líneas 300 y 326:

```
$servers->setValue('server','base',array('dc=profesordeinformatica,dc=com'));
$servers->setValue('login','bind_id','cn=admin,dc=profesordeinformatica,dc=com');
```

Accedemos



## Práctica creación de usuarios con la línea de comandos

1.- Instalamos el paquete Idapscripts

```
sudo apt-get install Idapscripts
```

2.- Añadimos el password al fichero

```
echo -n "egibide" > /etc/ldapscripts/ldapscripts.passwd  
chmod 400 /etc/ldapscripts/ldapscripts.passwd
```

Donde egibide es el password de nuestro servidor ldap.

**Nota:** Editando el fichero directamente da problemas.

Para comprobar que lo hemos configurado bien podemos probar estos comandos:

```
ldapwhoami -x -D cn=admin,dc=mydomain,dc=com -y /etc/ldapscripts/ldapscripts.passwd.  
ldapwhoami -x -D cn=admin,dc=mydomain,dc=com -w egibide
```

3.- modificamos /etc/ldapscripts/ldapscripts.conf dejamos con el siguiente contenido (borramos el contenido anterior):

```
SERVER=localhost  
BINDDN='cn=admin,dc=profesordeinformatica,dc=com'  
BINDPWDFILE="/etc/ldapscripts/ldapscripts.passwd"  
SUFFIX='dc=profesordeinformatica,dc=com'  
GSUFFIX='ou=groups'  
USUFFIX='ou=users'
```

```

MSUFFIX='ou=machines'
GIDSTART=10000
UIDSTART=10000
MIDSTART=10000
# User properties
USHELL="/bin/bash"
UHOMES="/home/users/%u"
CREATEHOMES="yes"
HOMESKEL="/etc/skel"
HOMEPERMS="700"
GCLASS="posixGroup"

```

**Nota:** groups, machines y users se han creado con phpmymadmin o exportando los ficheros OU.ldif.

4.- Modificamos /usr/share/ldapscripsts/runtime.debian con el siguiente contenido:

```

### This file predefine some ldapscripsts variables for Debian boxes.
#
# Copyright (c) 2005 Ganaoui LAPLANCHE - Linagora
# Copyright (c) 2005-2007 Pierre Habouzit
# Copyright (c) 2009 Alexander GQ Gerasiov
#
# This program is free software; you can redistribute it and/or
# modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation; either version 2
# of the License, or (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307,
# USA.

##### Beginning of ldapscripsts configuration #####

getfield() {
    local field="$1"
    local nssconf=/etc/libnss-ldap.conf
    if [ -f "$nssconf" ];then
        local value=$(awk "/^s*$field/ {print \$2}" /etc/libnss-ldap.conf)
    else
        local value="$2"
    fi
    echo ${value:-$2}
}

getsuffix() {
    field="$1"
    value=$(getfield "$1" | sed -e "s/.*$/")
    echo ${value:-$2}
}

# LDAP Configuration
SERVER=$(getfield uri "$(getfield host)")
BINDDN=$(getfield rootbinddn)
if [ -f /etc/libnss-ldap.secret ];then
    BINDPWDFILE=/etc/libnss-ldap.secret

```

```

elif [ -f /etc/ldap.secret ];then
    BINDPWDFILE=/etc/ldap.secret
fi

SUFFIX=`getfield base`
GSUFFIX=`getsuffix nss_base_group 'ou=Group'`
USUFFIX=`getsuffix nss_base_passwd 'ou=People'`
MSUFFIX=`getsuffix nss_base_hosts 'ou=Hosts'`

# User properties
[ -f /etc/adduser.conf ] && . /etc/adduser.conf
USHELL=${DSHELL:-"/bin/bash"}
UHOMES=${DHOME:-"/home"}"/%u"
HOMESKEL=${SKEL:-"/etc/skel"}
HOMEPERMS=${DIR_MODE:-"0755"}

# Where to log
LOGFILE="/var/log/ldapscripts.log"

# Various binaries used within scripts
LDAPSEARCHBIN=`which ldapsearch`
LDAPADDBIN=`which ldapadd`
LDAPDELETEBIN=`which ldapdelete`
LDAPMODIFYBIN=`which ldapmodify`
LDAPMODRDNBIN=`which ldapmodrdn`
LDAPPASSWDBIN=`which ldappasswd`

# Getent command to use - choose the ones used on your system. Leave blank or comment for auto-guess.
# GNU/Linux
GETENTPWCMD="getent passwd"
GETENTGRCMD="getent group"

TMPDIR="/tmp"
##### End of configuration #####

```

5.- Añadimos usuarios y grupos con los comandos:

```

ldapaddgroup usuariosldap
ldapadduser aitorla usuariosldap

```

6.- Asignamos un password al usuario:

```

ldapsetpasswd aitorla
Changing password for user uid=aitorla,ou=users,dc=profesordeinformatica,dc=com
New Password:
Retype New Password:
Successfully set password for user uid=aitorla,ou=users,dc=profesordeinformatica,dc=com

```

## Práctica: Administración gráfica de LDAP desde LAM

1.- Descargamos e instalamos Ldap Account Manager:

Descargar la versión .deb de <https://www.ldap-account-manager.org/lamcms/releases>

o apt-get install ldap-account-manager-lamdaemon

2.- Editamos el fichero **/usr/share/ldap-account-manager/config/lam.conf**

# server address (e.g. ldap://localhost:389 or ldaps://localhost:636)

serverURL: ldap://localhost:389

# list of users who are allowed to use LDAP Account Manager

# names have to be separated by semicolons

# e.g. admins: cn=admin,dc=yourdomain,dc=org;cn=root,dc=yourdomain,dc=org

admins: cn=admin,dc=**profesordeinformatica**,dc=com

# password to change these preferences via webfrontend (default: lam)

passwd: {SSHA}RjBruJcTxZEdcBjPQdRBkDaSQeY= iueleA==

# suffix of tree view

# e.g. dc=yourdomain,dc=org

treesuffix: dc=**profesordeinformatica**,dc=com

.....

types: suffix\_user: ou=People,dc=**profesordeinformatica**,dc=com

types: attr\_user: #uid;#givenName;#sn;#uidNumber;#gidNumber

types: modules\_user: inetOrgPerson,posixAccount,shadowAccount,sambaSamAccount

types: suffix\_group: ou=group,dc=**profesordeinformatica**,dc=com

types: attr\_group: #cn;#gidNumber;#memberUID;#description

types: modules\_group: posixGroup,sambaGroupMapping

types: suffix\_host: ou=machines,dc=**profesordeinformatica**,dc=com

types: attr\_host: #cn;#description;#uidNumber;#gidNumber

types: modules\_host: account,posixAccount,sambaSamAccount

types: suffix\_smbDomain: dc=**profesordeinformatica**,dc=com

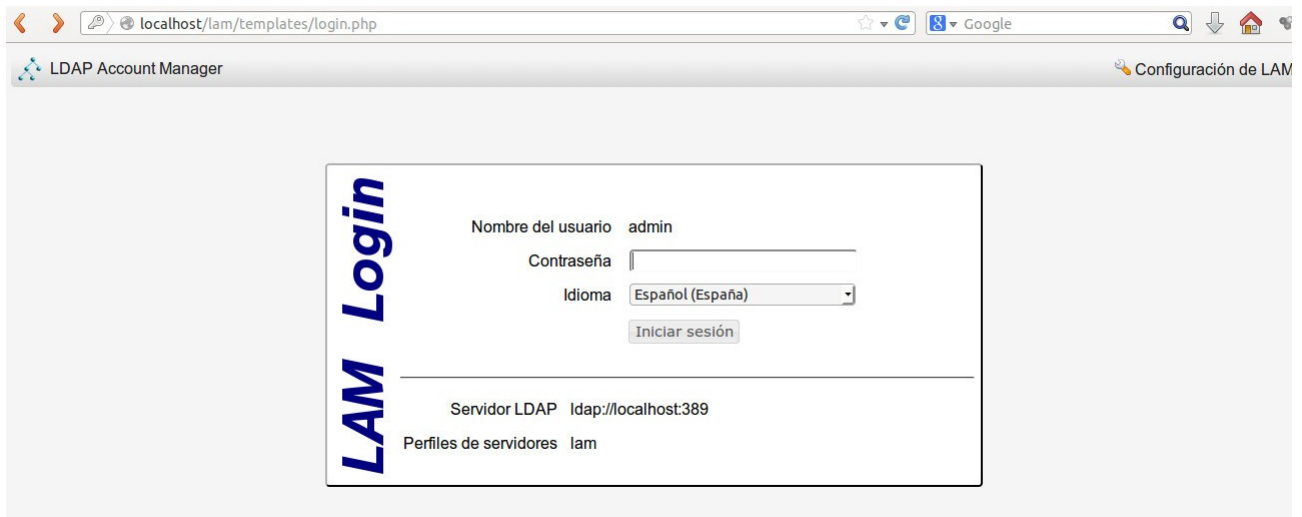
types: attr\_smbDomain: sambaDomainName:Domain name;sambaSID:Domain SID

types: modules\_smbDomain: sambaDomain

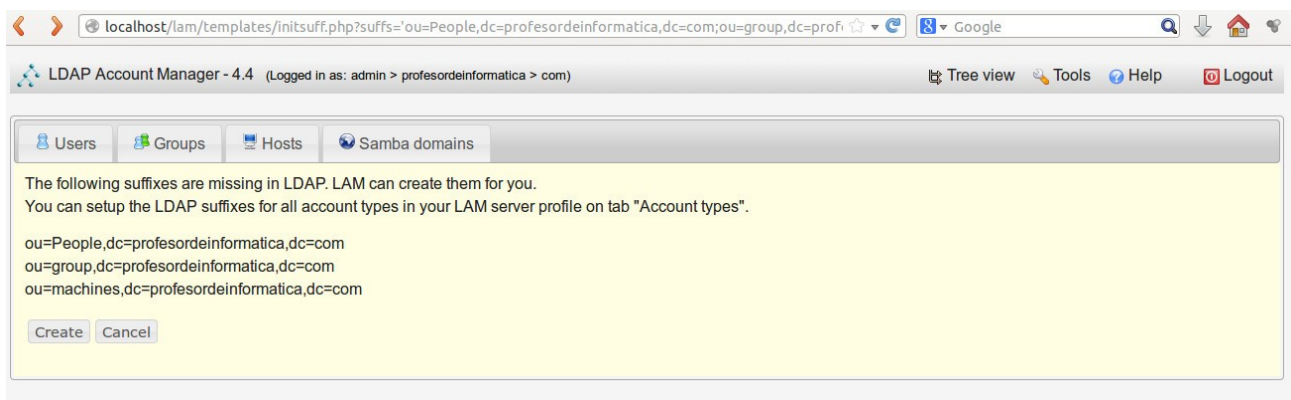
**Nota:** Básicamente cambiamos example por profesordeinformatica. Podemos cambiar las unidades organizativas por las que hemos puesto anteriormente: Users en lugar de People y Groups en lugar de Group.

3.- Accedemos a <http://localhost/lam/>





4.- Creamos la estructura, si no la hemos creado anteriormente.



5.- Creamos grupos y usuarios

## Otras herramientas administrativas

Jxplorer

apt-get install jxplorer

## Configuración del cliente LDAP

La idea es poder acceder desde una máquina virtual o cualquier ordenador con los usuarios de nuestro servidor LDAP. Vamos a considerar que la IP de nuestro servidor LDAP es la 192.168.1.128. Estas son las configuraciones que tenemos que hacer en nuestro cliente Ubuntu:

1.- Instalamos las librerías ldap

sudo apt-get install libnss-ldap libpam-ldap ldap-utils

LDAP server Uniform Resource Identifier: **ldap://192.168.1.128/**  
Distinguished name of the search base: **dc=profesordeinformatica,dc=com**  
ldap://ldap.tuxnetworks.com **3**  
Make local root Database admin: **Yes**  
Does the LDAP database require login? **No**  
LDAP account for root: **cn=admin,dc=profesordeinformatica,dc=com**  
LDAP root password: (Introducimos el password de root de LDAP)

Si nos hemos confundido en alguna configuración podemos reconfigurar el paquete de la siguiente forma:

```
sudo dpkg-reconfigure libnss-ldap
```

Nota: Si no nos deja lo hacemos directamente sobre el fichero /etc/ldap.conf

2.- Modificamos el fichero /etc/nsswitch.conf

```
sudo gedit /etc/nsswitch.conf
```

# Añadimos a la línea 7,8 y 9 del fichero la palabra **ldap**:

```
passwd:      compat      ldap
group:       compat      ldap
shadow:      compat      ldap
```

# Cambiamos la línea del 19 del fichero

```
netgroup:    ldap
```

3.-Modificamos el fichero /etc/pam.d/common-password

```
sudo gedit /etc/pam.d/common-password
```

# Cambiamos la línea 26 ( quitamos 'use\_authtok' ) y lo dejamos de la siguiente forma:

```
password [success=1 user_unknown=ignore default=die] pam_ldap.so try_first_pass
```

4.- Cambiamos el fichero /etc/pam.d/common-session

```
sudo gedit /etc/pam.d/common-session
```

# añadimos al final del fichero para que nos cree un directorio automáticamente la primera vez que entre el usuario.

```
session optional pam_mkhomedir.so skel=/etc/skel umask=077
```

5.- Verificamos el fichero /etc/ldap/ldap.conf y /etc/hosts utiliza ldap.profesordeinformatica.com

6.- Instalamos sysv-rc-conf y habilitamos la librería libnss-ldap

```
sudo apt-get install sysv-rc-conf
```

```
sudo sysv-rc-conf libnss-ldap on
```

7.- Probamos con un usuario del servidor ldap

```
su - aitorla
```

## Práctica: Acceso restringido al servidor Apache con usuarios LDAP

Se trata que validemos el acceso restringido de un directorio privado con los usuarios de un servidor LDAP.

1.- Instalamos los paquetes necesarios para la validación:

```
sudo apt-get install install libapache-mod-ldap libapache-authnetldap-perl libldap-2.3-0 ldap-utils
```

2.- Cargar los módulos de apache:

```
sudo a2enmod ldap a2enmod authnz_ldap
```

3.- Comprobamos que tenemos el "AllowOverride All" en el fichero para que tenga efecto el .htaccess.

```
/etc/apache2/sites-enabled/000-default
```

4.- Creamos el fichero /var/www/accesorestringido/.htaccess

```
AuthName "Acceso con usuario LDAP"
```

```
AuthType Basic
```

```
AuthBasicProvider ldap
```

```
AuthLDAPURL ldap://localhost:389/ou=users,dc=profesordeinformatica,dc=com?uid?sub
```

```
AuthLDAPBindDN "cn=admin,dc=profesordeinformatica,dc=com"
```

```
AuthLDAPBindPassword "egibide"
```

```
AuthzLDAPAuthoritative off
```

```
Require valid-user
```

5.- Reiniciamos el servidor

```
/etc/init.d/apache2 restart
```

6.- probamos en un navegador:

<http://localhost/accesorestringido>

## Práctica Acceso FTP con usuarios LDAP (\*No funciona\*)

La idea es que podamos acceder a un servidor FTP accediendo con los usuarios de un servidor LDAP.

1.- Instalamos el módulo:

```
apt-get install proftpd-mod-ldap
```

2.- Editamos el fichero /etc/proftpd/modules.conf y descomentamos la línea:

```
#LoadModule mod_ldap.c
```

3.- Editamos el fichero /etc/proftpd/proftpd.conf y buscamos la línea:

```
#Include /etc/proftpd/ldap.conf
```

4.- Editamos el fichero /etc/proftpd/ldap.conf y añadimos las siguientes líneas:

```
<IfModule mod_ldap.c>
#LDAPServer ldap://localhost/??sub
#LDAPDNInfo "cn=admin,dc=profesordeinformatica,dc=com" "egibide"
#LDAPDoAuth on "ou=users,dc=profesordeinformatica,dc=com"
LDAPServer ldap://localhost/??sub
LDAPBindDN "cn=admin,dc=profesordeinformatica,dc=com" "egibide"
LDAPUsers dc=users,dc=profesordeinformatica,dc=com (uid=%u) (uidNumber=%u)
</IfModule>
```

Nota: egibide es el password del servidor LDAP.

Donde especificamos el servidor LDAP, el nombre distinguido del usuario que se conecta, su contraseña y la ubicación de la rama que contiene los usuarios.

5.- Para que puedan autenticarse usuarios con la shell (/bin/false) hay que descomentar la siguiente l

```
#RequireValidShell off
```

6.- Reiniciamos el servidor de FTP

```
/etc/init.d/proftpd restart
```

7.- Probamos con un usuario y password LDAP

```
ftp localhost
```

## Prácticas propuestas

### Ejercicio 1

Realiza una instalación de la plataforma Moodle con la validación de usuarios de un servidor LDAP.

### Ejercicio 2

Realiza una instalación de la plataforma ownCloud con la validación de usuarios de un servidor LDAP.

### Ejercicio 3

Despliega la siguiente aplicación en un servidor LDAP que te permite modificar tu cuenta ldap.

<https://gist.github.com/mattrude/657334>

## Configuración de LDAP con NFS

El Network File System (Sistema de archivos de red), o NFS, es un protocolo de nivel de aplicación, según el Modelo OSI. Es utilizado para sistemas de archivos distribuido en un entorno de red de computadoras de área local. Posibilita que distintos sistemas conectados a una misma red accedan a ficheros remotos como si se tratara de locales. Originalmente fue desarrollado en 1984 por Sun Microsystems, con el objetivo de que sea independiente de la máquina, el sistema operativo y el protocolo de transporte, esto fue posible gracias a que está implementado sobre los protocolos XDR (presentación) y ONC RPC (sesión). El protocolo NFS está incluido por defecto en los Sistemas Operativos UNIX y la mayoría de distribuciones Linux.

Para poder configurar un servidor NFS primero deberemos instalar los paquetes necesarios:

## **Servidor**

```
sudo apt-get install nfs-common nfs-kernel-server portmap
```

Ahora hemos de modificar el archivo `/etc/exports` para indicar que carpeta queremos exportar, y bajo que condiciones. En el caso de los homes que exportaremos para el servicio LDAP quedará así:

```
# gedit /etc/exports
/home/users 192.168.1.0/24(rw,async,no_root_squash,subtree_check)
```

Además de esto habremos de modificar los archivos `/etc/hosts.allow` y `/etc/hosts.deny`, indicando que servicios permitimos y bloqueamos, por temas de seguridad:

```
# gedit /etc/hosts.allow
portmap: 192.168.1.0/255.255.255.0
nfs: 192.168.1.0/255.255.255.0
```

```
# gedit /etc/hosts.deny
portmap: ALL
```

Reiniciamos los servicios

```
# /etc/init.d/nfs-kernel-server restart
# /etc/init.d/portmap restart
```

Creamos el directorio por si no existe

```
# mkdir /home/users
```

## **Cliente**

Ahora hemos de configurar la parte del cliente, lo primero será instalar el paquete `nfs-common`, en caso de que no esté instalado:

```
# apt-get install nfs-common
```

Y modificaremos el archivo `/etc/fstab` que es el archivo que indica al Sistema Operativo que debe montar durante el arranque del sistema:

```
# vim /etc/fstab
192.168.1.128:/home/users /home/users nfs rsize=8192,wsiz=8192,timeo=14,intr
```

Creamos el directorio por si no existe

```
# mkdir /home/users
```

Con esto el cliente queda configurado, para que la configuración surta efecto tenemos dos opciones. La primera es reiniciar el PC, ya que el sistema cargará al arranque lo indicado en /etc/fstab y la segunda es ejecutar el siguiente comando:

```
# mount -a
```

## Otras formas de autenticación de usuarios

Hemos visto que LDAP se suele utilizar para autenticación de usuarios en diferentes aplicaciones como correo electrónico, moodle, o como usuario del sistema centralizado. Hoy en día la mayoría de usuarios están ya registrados en Facebook, Twitter o Google. Podemos utilizar ese registro para hacer una autenticación en nuestras aplicaciones.

### Práctica autenticar usuarios Facebook con php

0.- Vamos a suponer que tenemos ya un LAMP (o Wamp) instalado

**Nota importante:** Tenemos que tener tb curl instalado  
sudo apt-get install php5-curl

1.- Creamos la BBDD y tabla

Creamos la base de datos "miaplicacion" con phpmyadmin

Creamos la tabla de usuarios:

```
CREATE TABLE `users` (  
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,  
  `oauth_provider` varchar(10),  
  `oauth_uid` text,  
  `username` text,  
  PRIMARY KEY (`id`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

2.- Creamos la API de la zona de desarrolladores de Facebook. Para ello accedemos a <https://developers.facebook.com/> pinchamos en "Apps" (del menú superior) y pinchamos en "Crear una nueva aplicación". Elegimos "Sitio web con inicio de sesión en Facebook" y ponemos profesordeinformatica.com.

Descargamos la api desde  
<https://developers.facebook.com/docs/php/gettingstarted/>  
o  
<https://github.com/facebook/facebook-php-sdk>

Descargamos el zip del lateral izquierdo download zip

y lo descomprimos en /var/www/profesrodeinformatica

```
cd /var/www/profesrodeinformatica  
cp /home/aitor/Descargas/facebook-php-sdk-master.zip .
```

# Capítulo 7

## Sistema de control de versiones

### 1.- Introducción

Un **sistema de control de versiones** o SVC (System Version Control) es una herramienta que registra los cambios realizados sobre un archivo o conjunto de archivos de un proyecto a lo largo del tiempo, de modo que puedas recuperar la versión de esos archivos en un estado anterior en el tiempo.

El sistema de control de versiones es de gran utilidad para entornos de **desarrollo colaborativo** donde varios diseñadores, maquetadores y/o programadores trabajan sobre un mismo proyecto. En todo momento tenemos los cambios que se realizan sobre cada uno de los ficheros por los diferentes programadores y podemos recuperar un fichero en un estado anterior.

En este capítulo nos vamos a centrar en la herramientas **Git y Subversión**, pero existen muchas otras como CVS, Mercurial, Bazaar, SourceSafe, etc.

### 2.- Definición y características de los SCV

Es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que se pueda recuperar versiones específicas de los mismos en un determinado momento.

Un SVC posee tres capacidades importantes:

- **Reversibilidad:** retornar a un estado anterior del proyecto en caso de fallos.
- **Concurrencia:** Muchas personas modificando el mismo código o documento.
- **Anotación:** Adjuntar información relevante de los cambios realizados.

Un sistema de control de versiones debe proporcionar:

- Mecanismo de almacenamiento de los elementos que deba gestionar (ej. archivos de texto, imágenes, documentación...).
- Posibilidad de realizar cambios sobre los elementos almacenados (ej. modificaciones parciales, añadir, borrar, renombrar o mover elementos).
- Registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente pudiendo volver o extraer un estado anterior del producto).

### 3.- Conceptos Básicos en los Sistemas de control de versiones



**Repositorio:** lugar en el que se almacenan los datos actualizados e históricos de cambios (sistema de archivos en un disco duro, un banco de datos, etc).

**Revisión:** Versión determinada de la información que se gestiona.

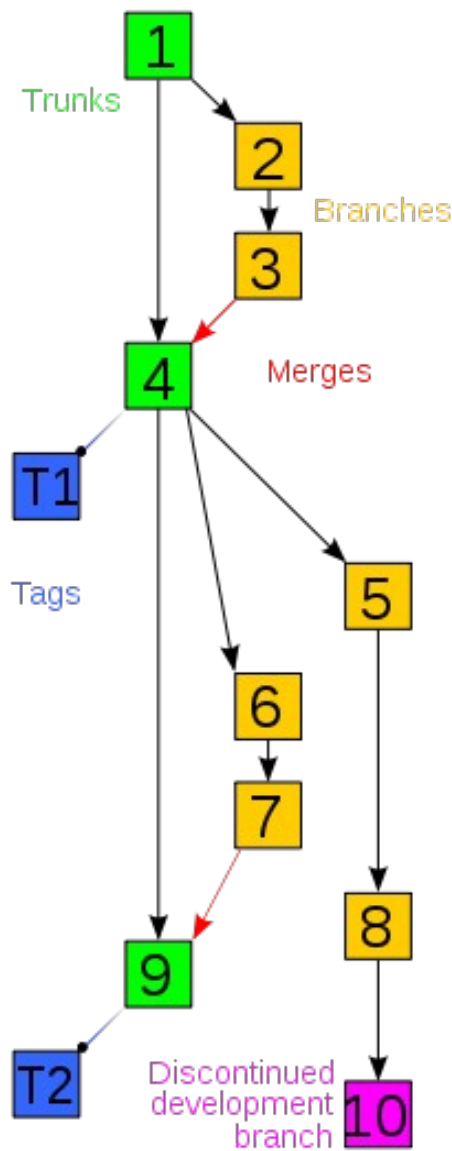
**Tags:** Permiten identificar de forma fácil revisiones importantes en el proyecto.

**Módulo:** Conjunto de directorios y/o archivos dentro del repositorio que pertenecen a un proyecto común.

**Branch:** Es una copia del proyecto aislada, de forma que los cambios realizados no afecten al resto del proyecto y vice versa, excepto cuando los cambios sean "unidos" de un lado al otro.

**Baseline:** Una revisión aprobada de un documento o fichero fuente, a partir del cual se pueden realizar cambios subsiguientes.

**Checkout:** crea una copia de trabajo local desde el repositorio.

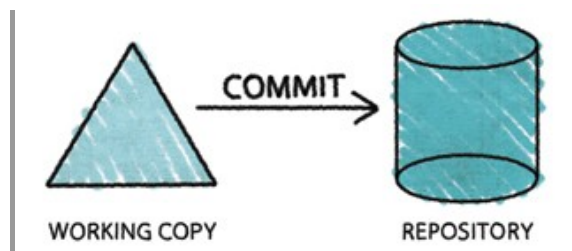


**Merge:** Une dos grupos de cambios en un archivo (o grupo de archivos), generando una revisión unificada.

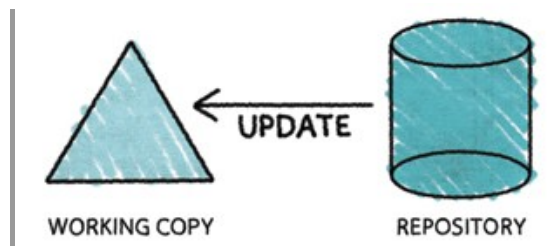
**Conflicto:** Sucede cuando dos o más personas intentan realizar diferentes cambios en la misma porción de código.

**Commit:** Consiste en realizar un cambio local en el proyecto y luego almacenar dicho cambio en el repositorio.

**Change set:** Conjunto de cambios realizados en un único commit.



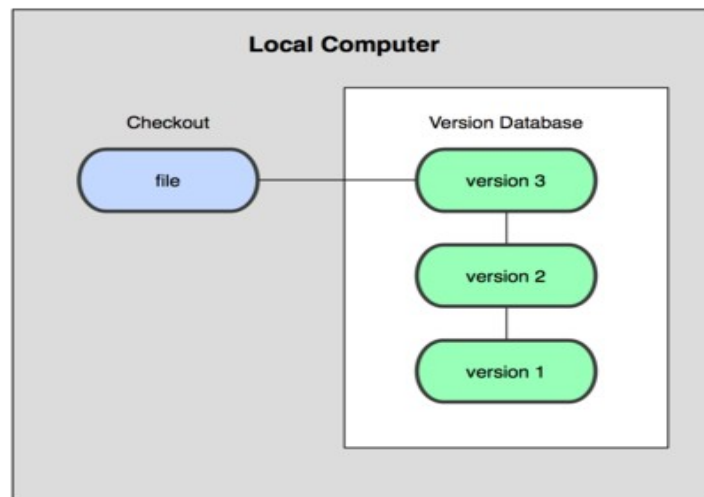
**Update:** Integra los cambios que han sido realizados en el repositorio en la copia de trabajo local.



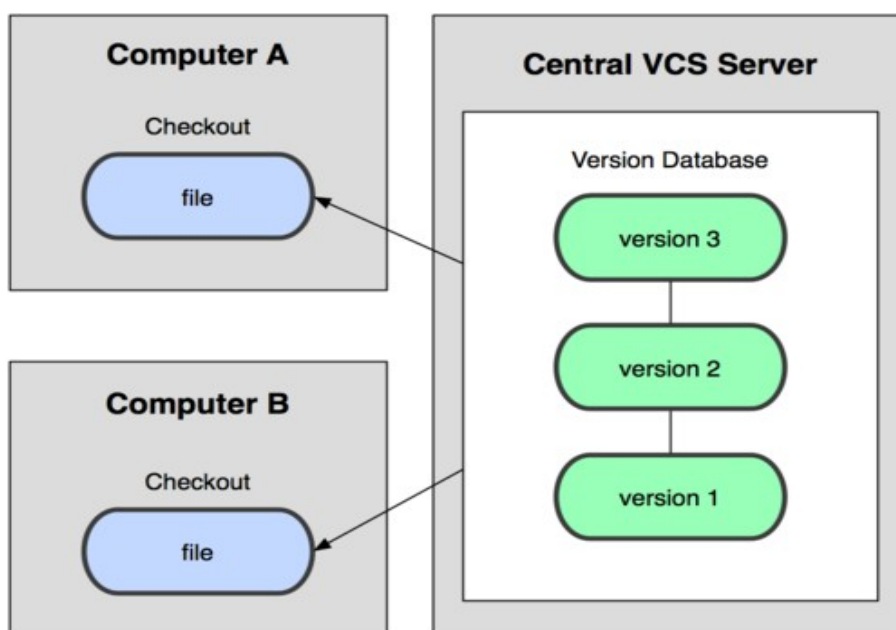
#### 4.- Clasificación de los SCV

Podemos clasificar los sistemas de control de versiones según la arquitectura para almacenar la información en locales, centralizados o distribuidos.

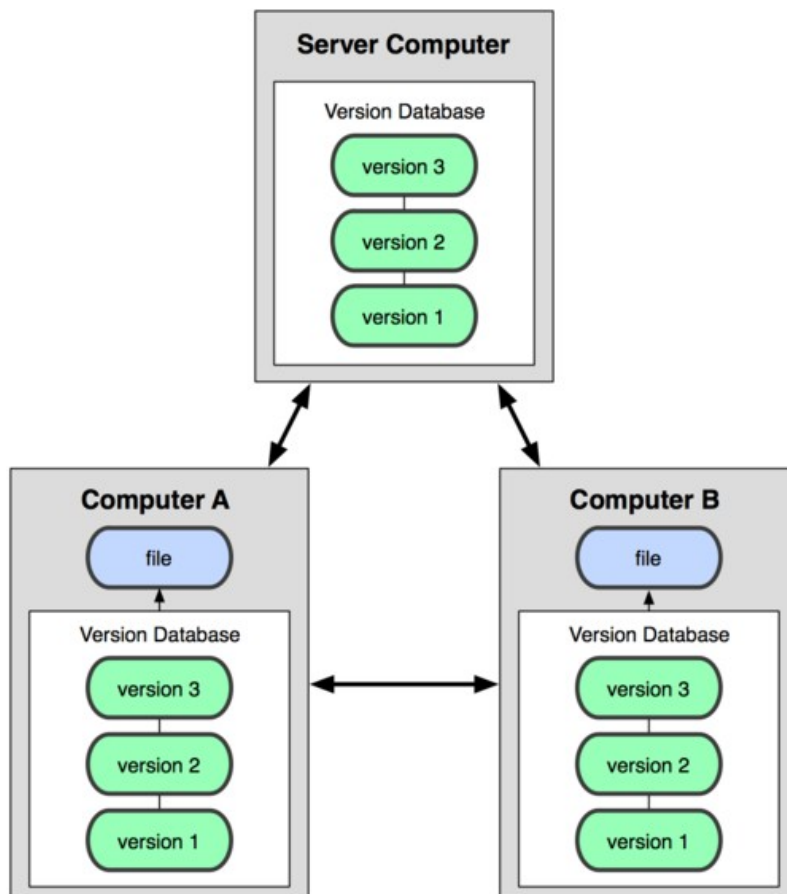
- **Locales:** La información se guarda en un ordenador o repositorio local con lo que no sirve para trabajar en forma colaborativa. Ejemplo: RCS (Revision Control System).



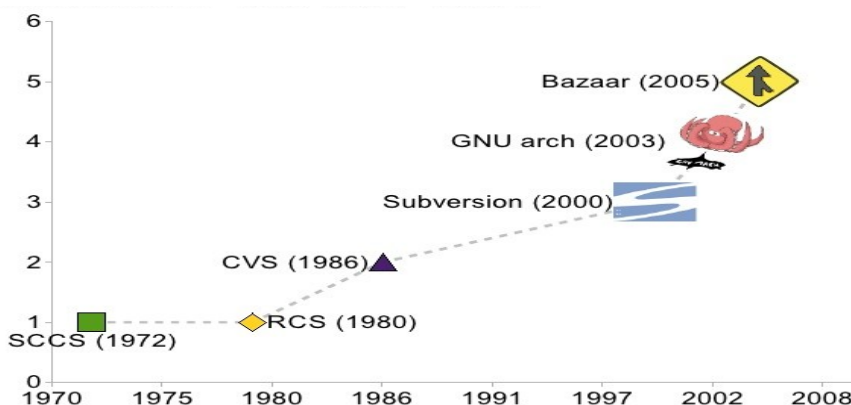
- **Centralizados o CVCS (Centralized Version Control System):** La información se guarda en un servidor dentro de un repositorio centralizado. Existe un usuario o usuarios responsables con capacidad de realizar tareas administrativas a cambio de reducir flexibilidad, necesitan la aprobación del responsable para realizar acciones, como crear una rama nueva. Ejemplos: Subversión y CVS.



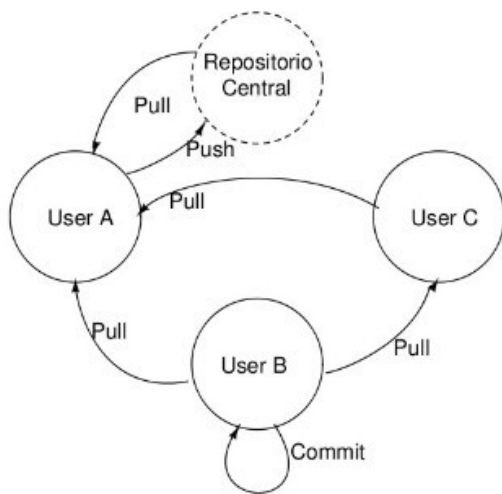
- **Distribuidos o DVCS (Distributed Version Control System):** Cada usuario tiene su propio repositorio. Los distintos repositorios pueden intercambiar y mezclar revisiones entre ellos. Es frecuente el uso de un repositorio, que está normalmente disponible, que sirve de punto de sincronización de los distintos repositorios locales. Ejemplos: Git y Mercurial, Bazaar y Darcs.



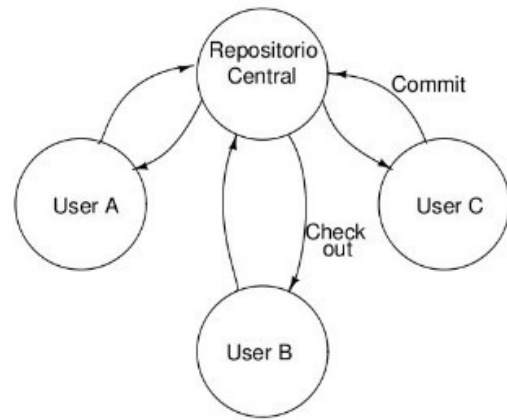
Generation	Networking	Operations	Concurrency	Examples
First	None	One file at a time	Locks	RCS, SCCS
Second	Centralized	Multi-file	Merge before commit	CVS, SourceSafe, Subversion, Team Foundation Server
Third	Distributed	Changesets	Commit before merge	Bazaar, Git, Mercurial



## Distribuidos



## Centralizados

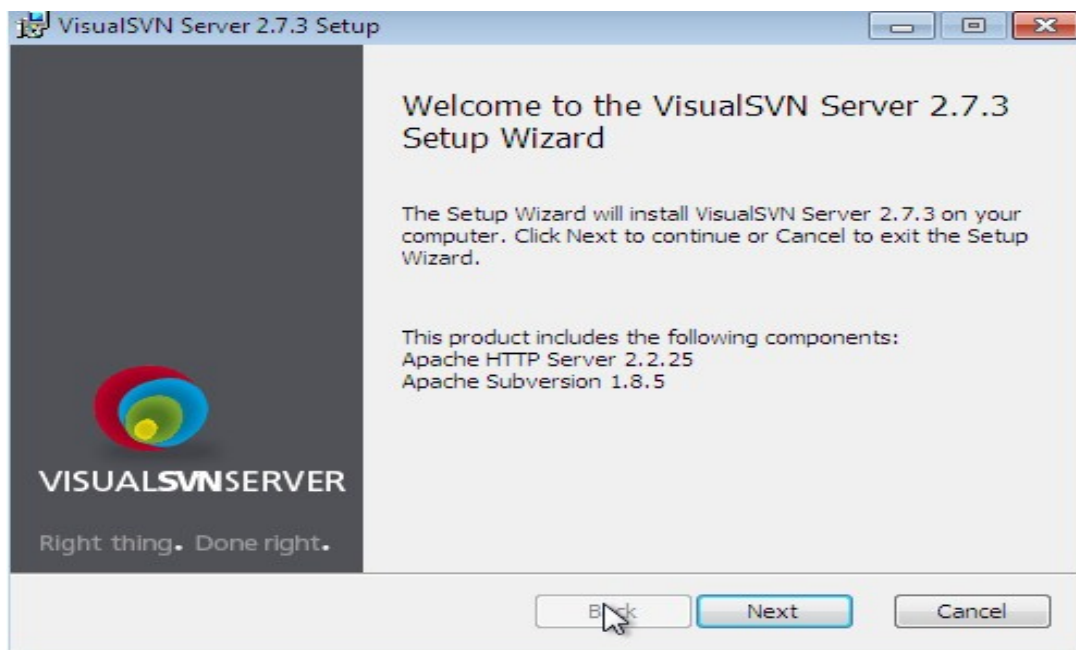


## 5.- Subversión

## 6.- Instalación y configuración de subversión en modo gráfico Windows

1.- Descargamos el servidor y lo instalamos

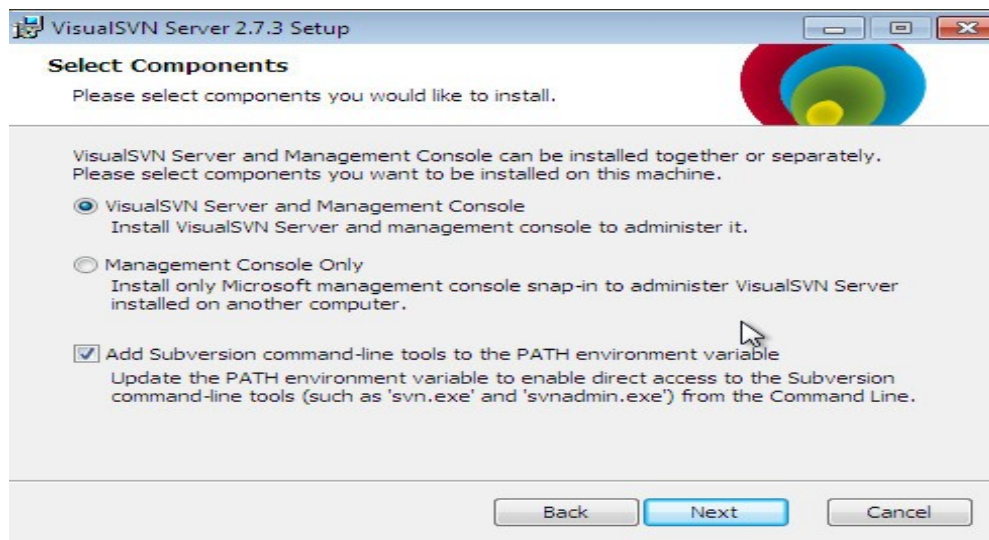
<http://www.visualsvn.com/server/download/>



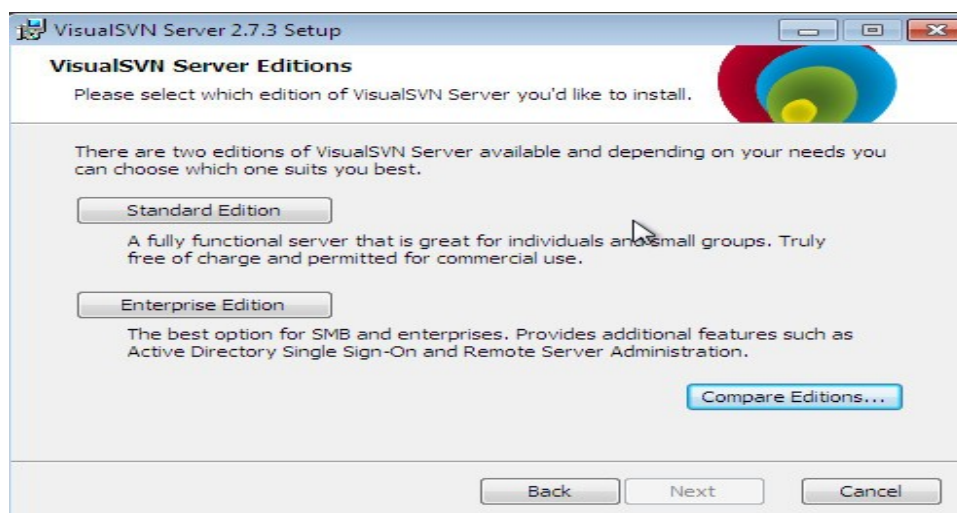
Aceptamos la licencia:



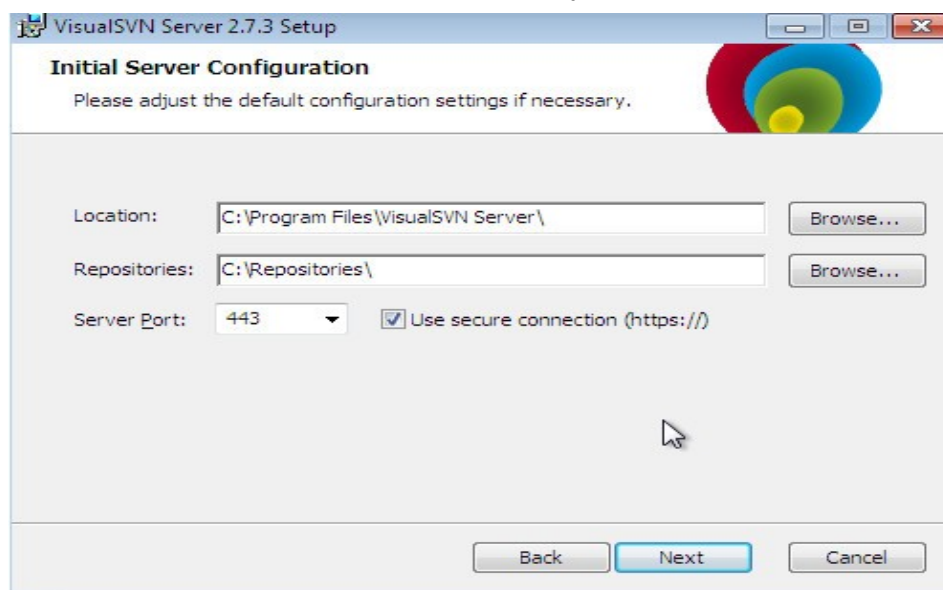
Instalamos el servidor y la consola:



Instalamos la edición estandar:

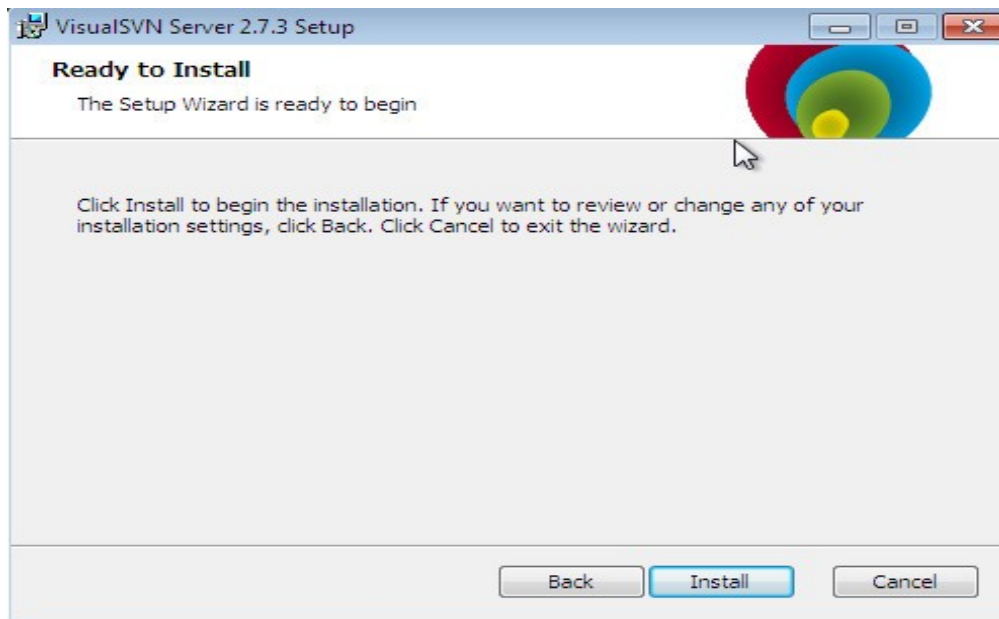


Seleccionamos el directorio donde lo instalamos y el directorio del los repositorios:

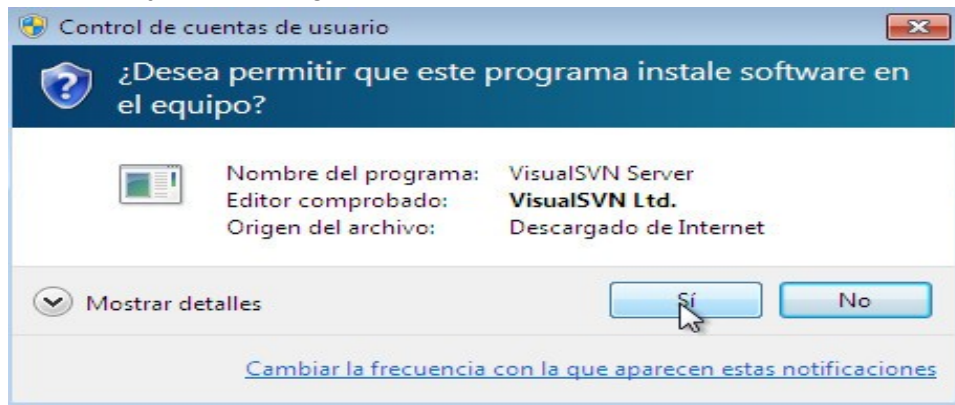


Comenzamos la instalación:

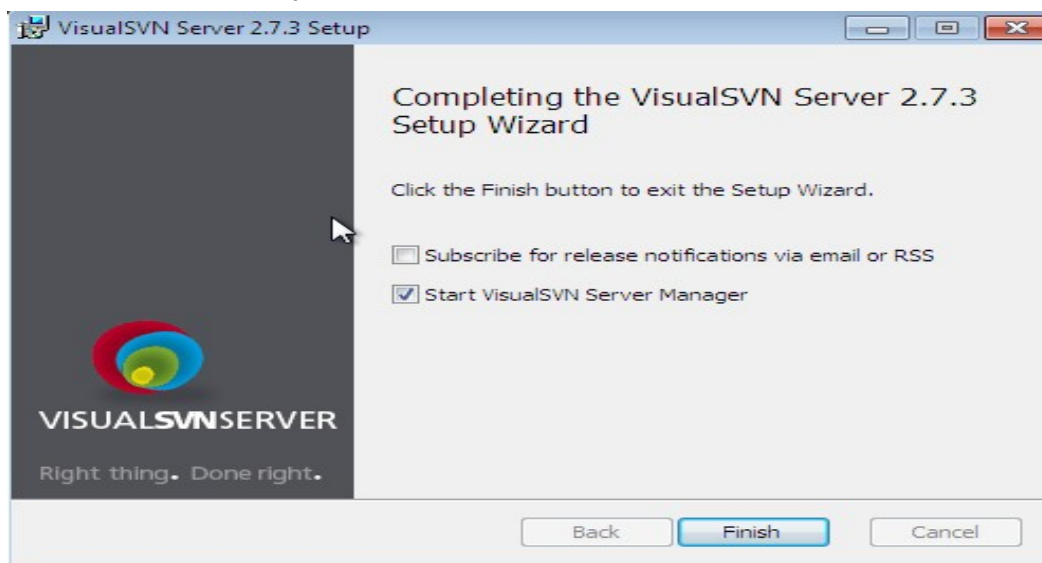




Permitimos se ejecute el programa de instalación:

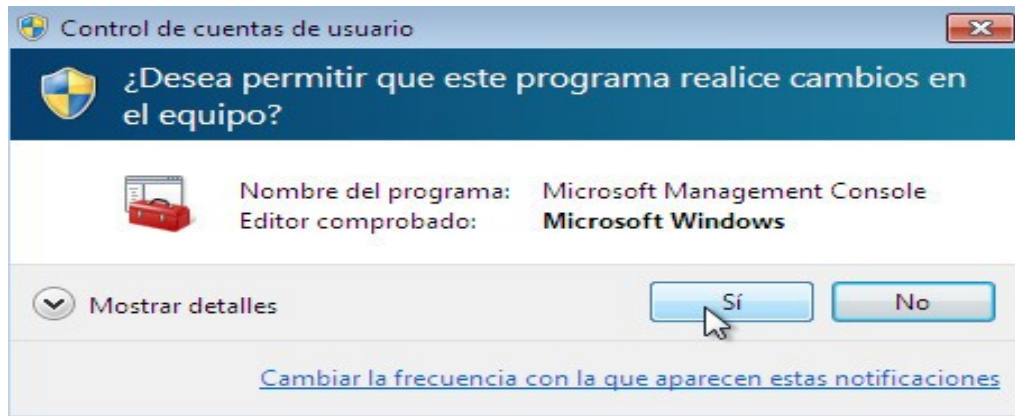


Finalizamos el asistente y arrancamos Visual SVN:

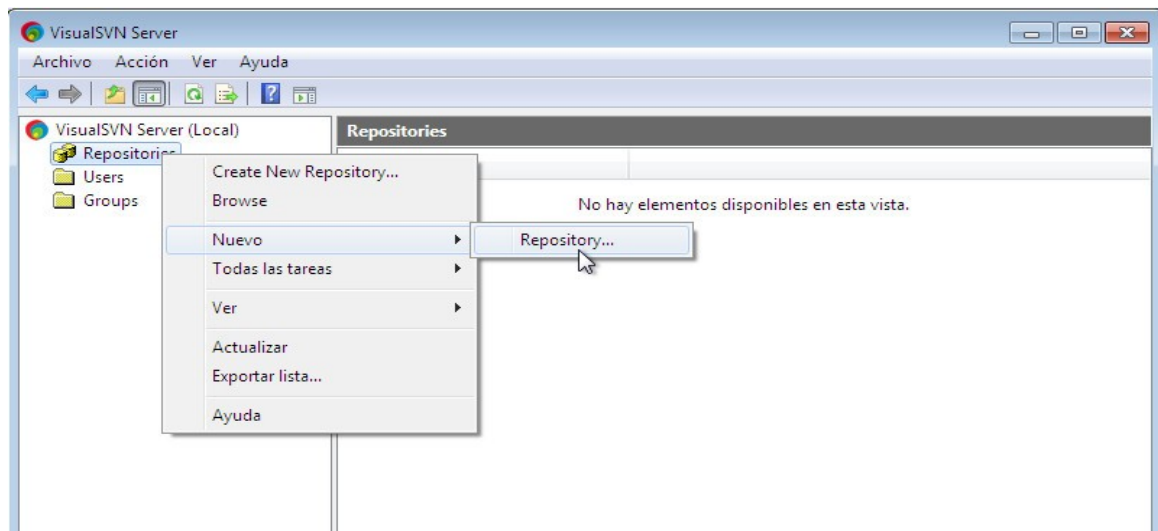


Permitimos la ejecución de VisualSVN:

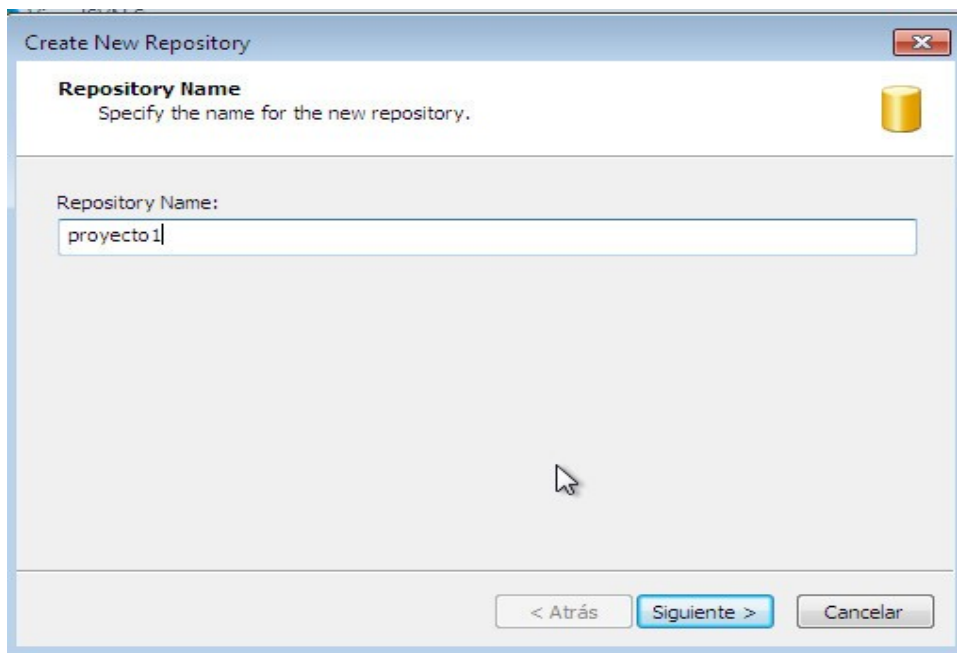




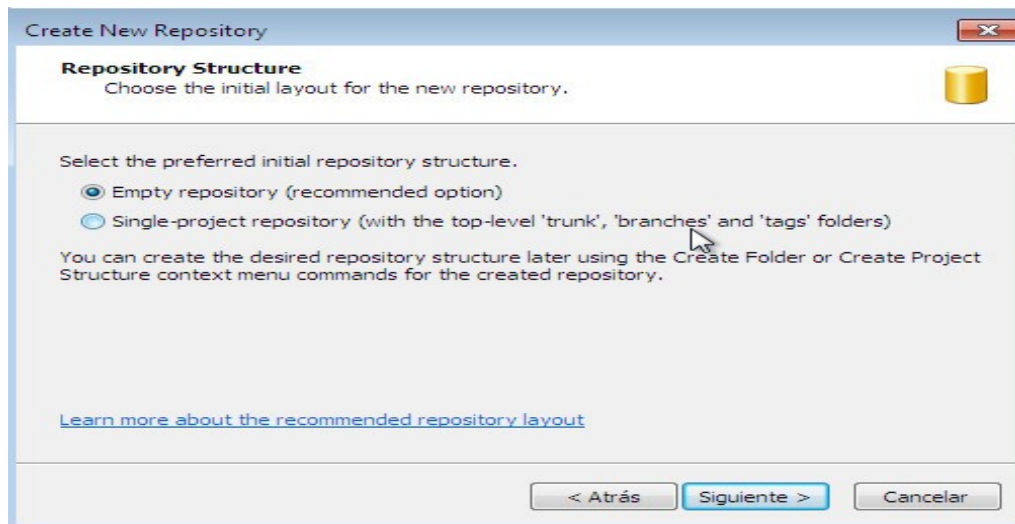
Creamos un nuevo repositorio:



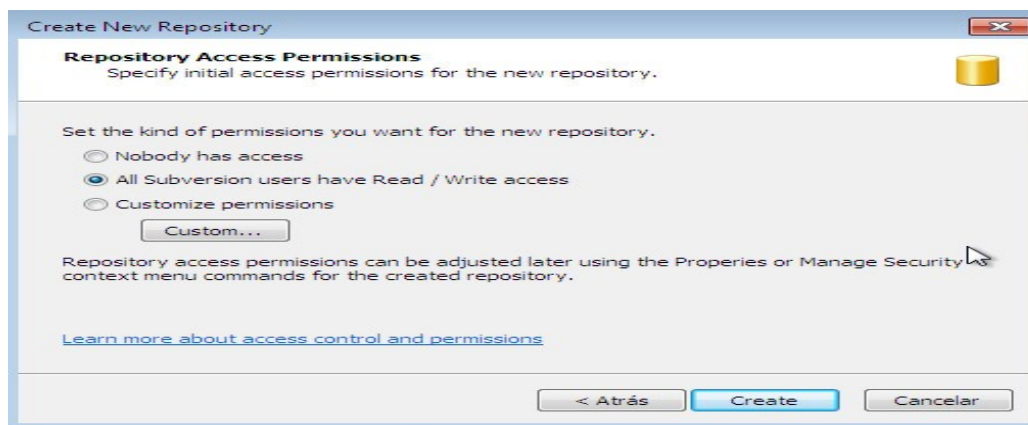
Le damos nombre:



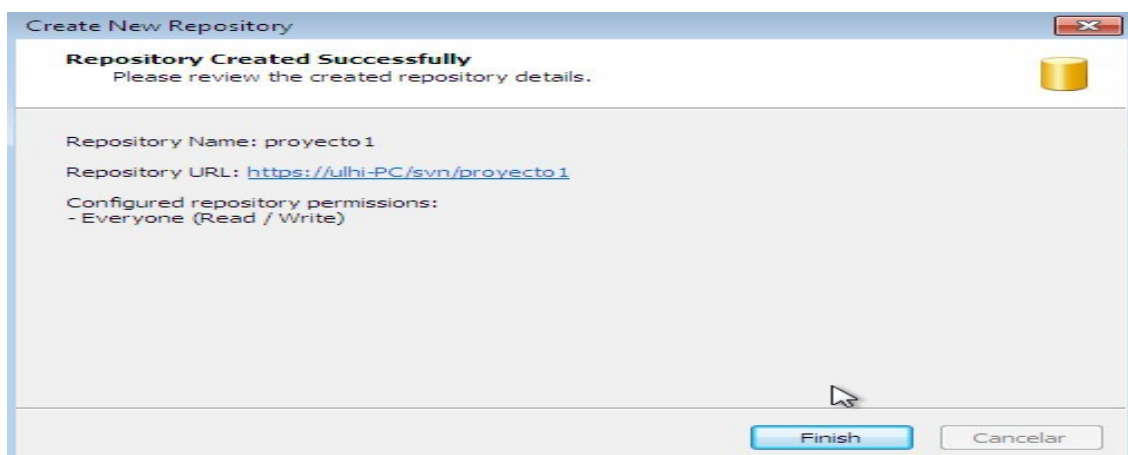
Creamos un repositorio vacío:



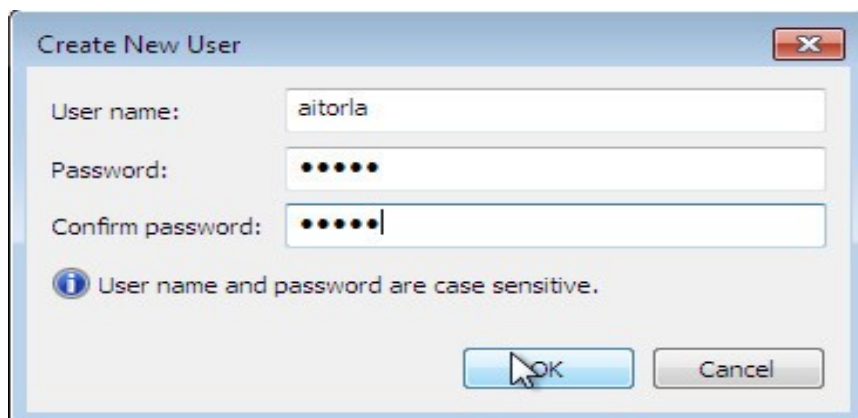
Damos permiso de lectura y escritura a todos los usuarios:



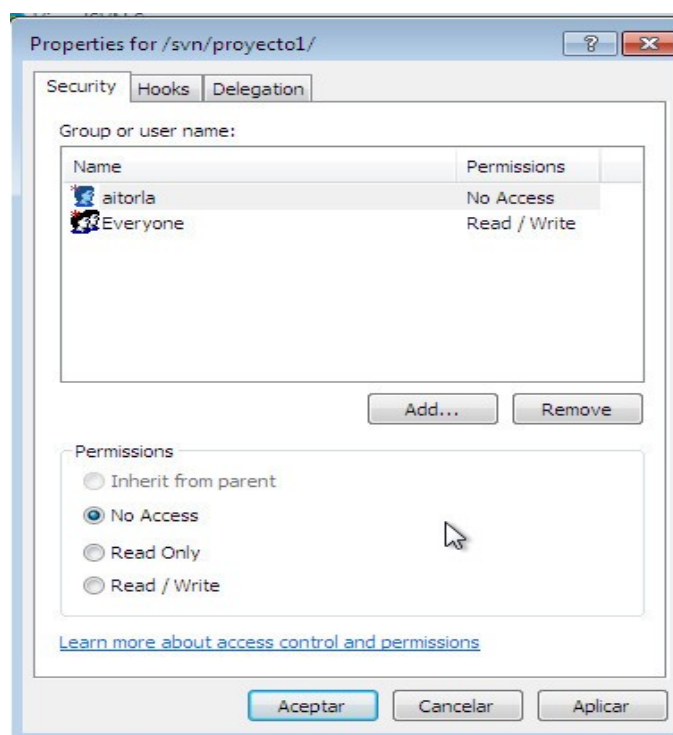
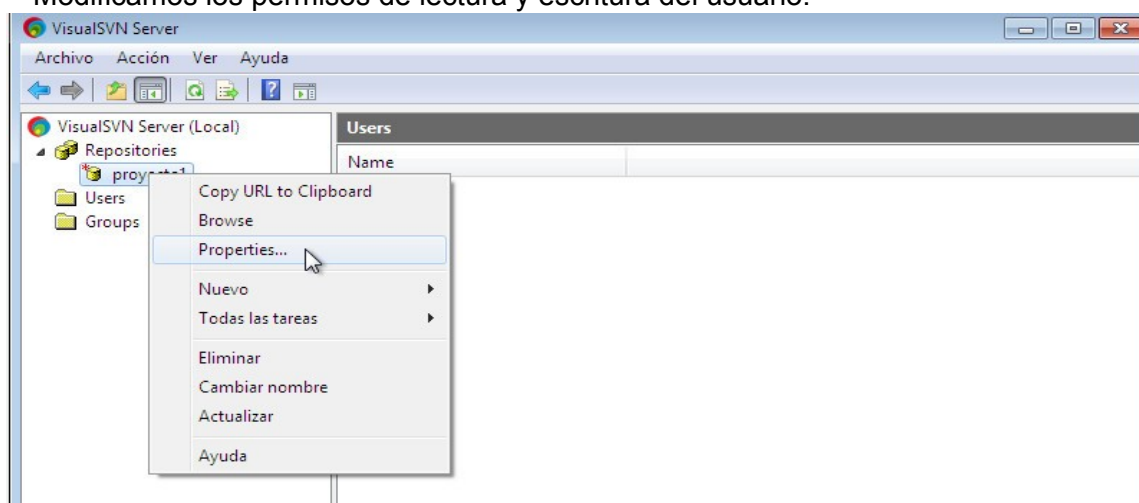
Finalizamos la creación:



Creamos un nuevo usuario:



Modificamos los permisos de lectura y escritura del usuario:



## Instalamos cliente TortoiseSVN

**TortoiseSVN** es un cliente subversion con licencia GNU GPL. Se integra al explorador de Windows y puede incorporar los comandos de subversión. Lo descargamos e instalamos.

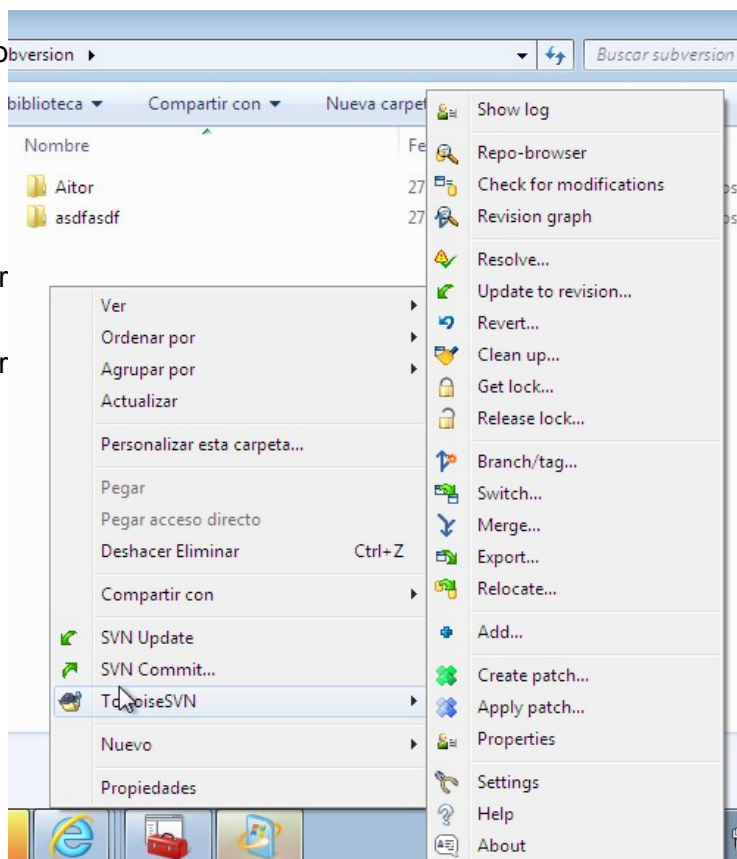
SVN Checkout: Primero nos conectamos al servidor.

<https://ip/svn/repositorio>

Posteriormente:

SVN Update: Descargar repositorio

SVN Commit: Actualizar repositorio



## Instalamos Subversión en ubuntu

<http://proyectosbeta.net/2013/07/instalar-un-servidor-svn-en-ubuntu-13-04/>

## Herramienta gráfica cliente SVN para ubuntu

```
sudo apt-get install rapidsvn
```

## Conceptos Subversión

### Repositorio (repository)

Es el almacén donde se guardan todos los ficheros y sus versiones. Todas las operaciones registran un punto de referencia en el repositorio: por ejemplo, hacer un check-out (extraer un fichero del repositorio) o un commit (incorporar un fichero al repositorio).

### Copia de trabajo (working copy)

Inicialmente es el directorio que se obtiene al hacer el check-out de un proyecto almacenado en

un repositorio. La copia de trabajo contiene información para que Subversion pueda saber qué ficheros están sincronizados y cuáles no. Es el directorio donde después de hacer check-out editaremos nuestros ficheros y los iremos incorporando en el repositorio. La copia de trabajo también contiene ficheros extras que ayudan a mantener la sincronía con el repositorio y facilitan la implementación de las órdenes. Estos ficheros de administración se encuentran en la carpeta .svn, y no se deben borrar.

### **Directorio para inicializar el repositorio**

Es el directorio que contiene los ficheros de la práctica o proyecto que queremos poner bajo control de revisiones. Si ejecutamos la orden svn import, Subversion incorporará todos los ficheros al repositorio. **IMPORTANTE:** Una vez finalizada la importación de ficheros, este directorio no se convierte en una copia de trabajo. Por lo tanto, para seguir trabajando con los ficheros que contenía este directorio, necesitaremos ejecutar svn check-out para obtener una copia de trabajo limpia y ubicada, para mayor seguridad, en un nuevo directorio. Entonces podremos empezar a modificar el código de la práctica que hay en el directorio donde se ha hecho el check-out. El directorio que hemos usado para inicializar el repositorio se puede borrar.

### **Sincronización**

Es la clave para realizar el control de versiones. Debe ser nuestro objetivo en todo momento: tener un repositorio y una copia de trabajo en sincronía. Si partimos de una copia de trabajo sincronizada, nos pueden pasar dos cosas:

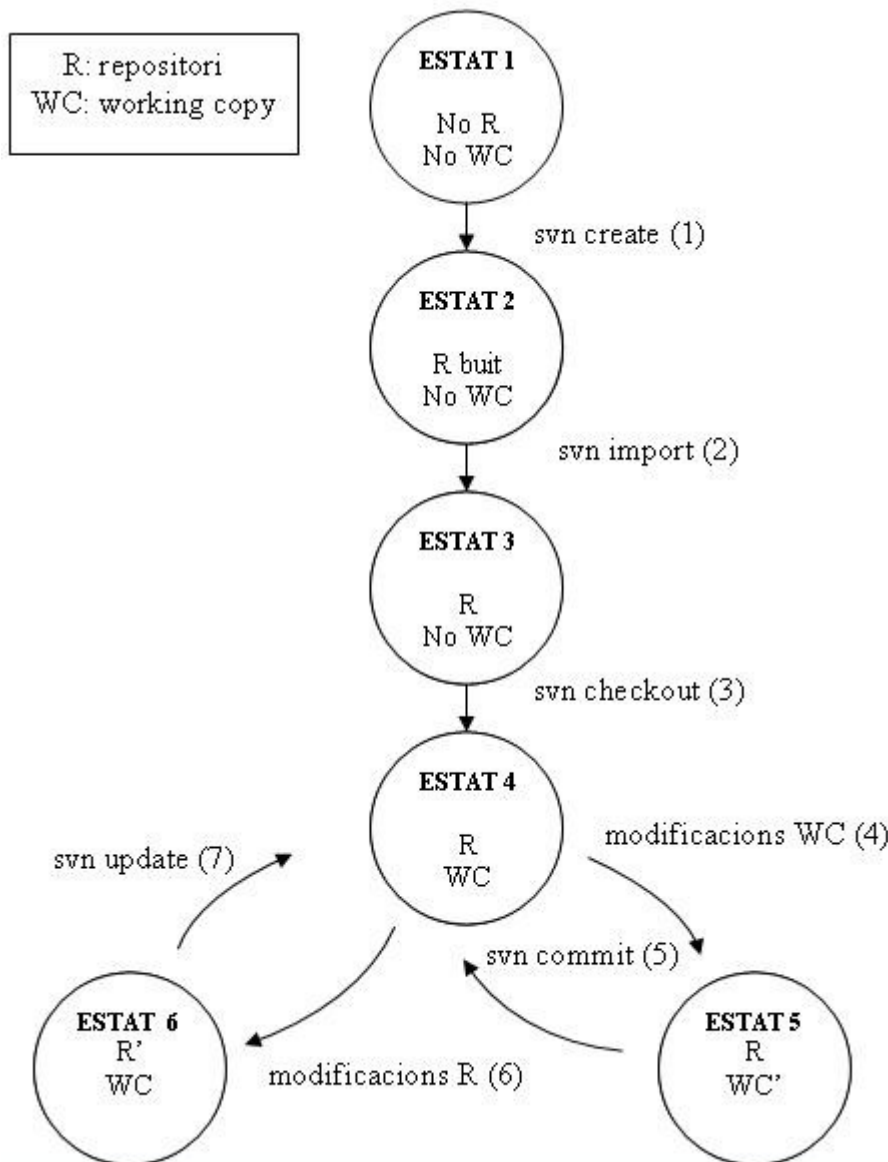
1. La copia de trabajo se ha modificado y no está sincronizada con el repositorio.
2. El repositorio ha sido actualizado por otro miembro del grupo de trabajo, en el caso de que haya un grupo de programadores, y el repositorio no está sincronizado con la copia de trabajo.

Para conseguir la sincronía en cada una de las dos situaciones anteriores, habrá que ejecutar:

- 1.svn commit, para actualizar el repositorio.
- 2.svn update, para descargarnos los cambios a nuestra copia de trabajo.

### **Ciclo habitual de trabajo**

Podemos ver el ciclo habitual de trabajo en el siguiente gráfico que explicaremos a continuación:



- Estado 1

No existe ni un repositorio ni una copia de trabajo. En este punto, podemos ejecutar la orden **svn create** para crear un repositorio.

**svn create**

- Estado 2

Disponemos de un repositorio, pero está vacío. Con la orden

**svn import**

cargamos el repositorio con los ficheros del directorio.

- Estado 3

Disponemos de un repositorio con datos, pero todavía no podemos modificar sus ficheros. Hay que crear la copia de trabajo:

## **svn checkout**

- Estado 4

El repositorio contiene datos y existe una copia de trabajo sincronizada. Ahora podemos modificar nuestra copia de trabajo:

- Modificamos ficheros con un editor de texto, o
- Añadimos, borramos, copiamos o renombramos ficheros:
  - **svn add**
  - **svn delete**
  - **svn copy**
  - **svn move**

- Estado 5

Tenemos una copia de trabajo modificada con respecto al repositorio. Ahora necesitamos guardar los cambios en el repositorio:

## **svn commit**

Con esta operación volvemos al estado 4.

- Estado 6 se puede llegar a este estado cuando se trabaja en grupo sobre un repositorio, y alguien lo modifica. En este caso, tendremos que incorporar estos cambios a nuestra copia de trabajo. Ejecutamos la orden:

## **svn update**

Con esta operación volvemos al estado 4.

En cualquier momento del desarrollo podemos consultar las diferencias entre el repositorio y nuestra copia de trabajo, mediante las órdenes: **svn list**, **svn estatus**, **svn diff**.

Cuando se trabaja con repositorios, se suele utilizar una estructura formada por tres subdirectorios: trunk, branches y tags:

- Trunk:

Es el directorio que contiene la última versión del proyecto que se está desarrollando, la versión común a todos los desarrolladores.

- Branches:

Este directorio contiene varios subdirectorios, uno para cada versión de "Trunk" que se quiera hacer. Es normal que cada desarrollador trabaje sobre su propio subdirectorio en "Branches", y que después ponga todo en común en el directorio "Trunk" con la orden **svn merge**. Cada subdirectorio de "Branches" normalmente se inicializará haciendo una copia de "Trunk" en el subdirectorio mediante **svn copy**.

- Tags:

En "Tags" se guardan copias de seguridad (snapshots) del proyecto, ejecutando la

orden **svn copy**. La única diferencia entre los directorios "Tags" y "Branches" es que nunca se modifican ficheros pertenecientes al directorio "Tags".

## Introducción Git

### Práctica Control de versiones Git en local

En esta práctica vamos a realizar una instalación de versiones básica con Git. Vamos a crear un repositorio y vamos a añadir y modificar un fichero en el.

1.- Instalamos git

```
sudo apt-get install git git-core
```

2.- Creamos un usuario

```
git config --global user.name "Aitor LA"  
aitor@linux:~$ git config --global user.email "aitor@kaixo.com"
```

**Nota:** El usuario va entre comillas

3.- Creamos una carpeta

```
mkdir /home/aitor/git
```

4.- Nos posicionamos en la carpeta e iniciamos hit

```
cd /home/aitor/hit
```

```
git init
```

```
Initialized empty Git repository in /home/aitor/git/.git/
```

5.- Vemos que no tenemos nada cacheado

```
git status
```

5.- Creamos fichero.txt

```
vi fichero.txt en un principio vacío
```

6.- Vemos que el fichero se ha creado pero no se ha agregado nada



```
git status
```

```
# En la rama master
```

```
#
```

```
# Commit inicial
```

```
#
```

```
# Archivos sin seguimiento:
```

```
# (use «git add <archivo>...» para incluir lo que se ha de ejecutar)
```

```
#
```

```
#      fichero.txt
```

no se ha agregado nada al commit pero existen archivos sin seguimiento (use «git add» para darle seguimiento)

## 7.- Añadimos el fichero

```
git add fichero.txt
```

**Nota:** podemos añadir todos los archivos haciendo git add .  
y vemos el estado:

```
git status
```

```
# En la rama master
```

```
#
```

```
# Commit inicial
```

```
#
```

```
# Cambios para hacer commit:
```

```
# (use «git rm --cached <archivo>...» para eliminar stage)
```

```
#
```

```
#      archivo nuevo: fichero.txt
```

```
#
```

## 8.- Hacemos un commit para hacer válidos los cambios

```
git commit -m "Primera prueba con git"
```

```
[master (root-commit) ec3b174] Primera prueba con git
```

```
1 file changed, 0 insertions(+), 0 deletions(-)
```

```
create mode 100644 fichero.txt
```

## 9.- Modificamos el fichero.txt y hacemos un commit

```
vi fichero.txt
```

```
git commit -a -m "Modificamos el fichero"
```

[master 8c0d578] Modificamos el fichero  
1 file changed, 1 insertion(+)

**Nota:** Para no tener que hacer un add podemos pasar el parámetro -a

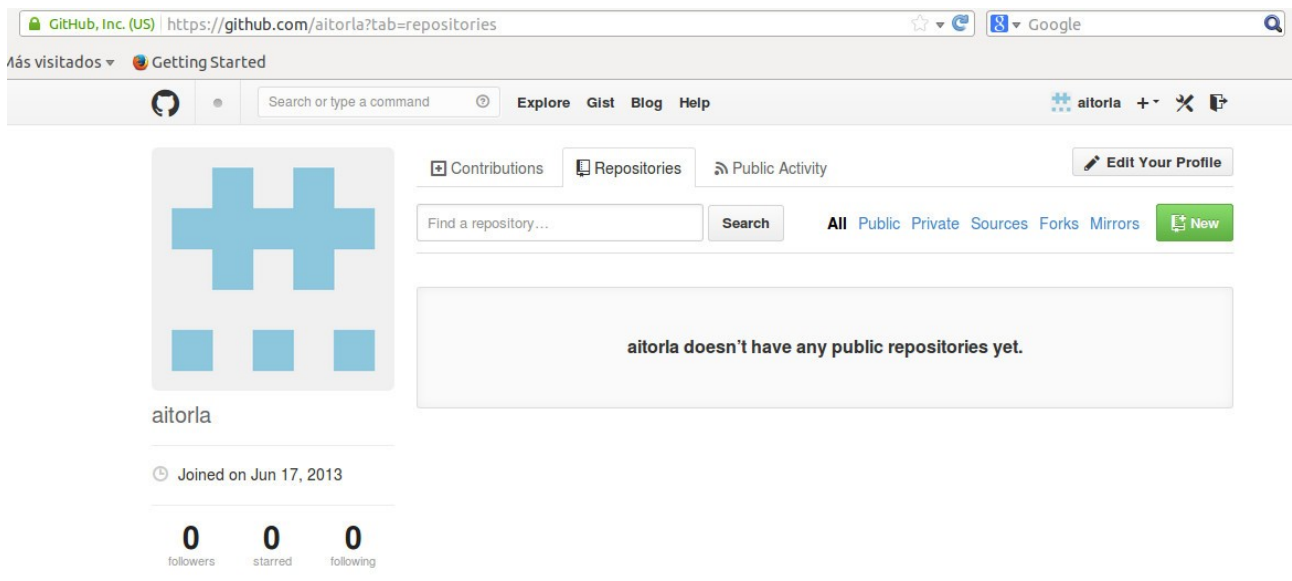
9.- Instalamos gitk que nos permite ver los cambios en modo gráfico  
apt-get install gitk  
y lo ejecutamos  
gitk

## Práctica Control de versiones Git en la nube

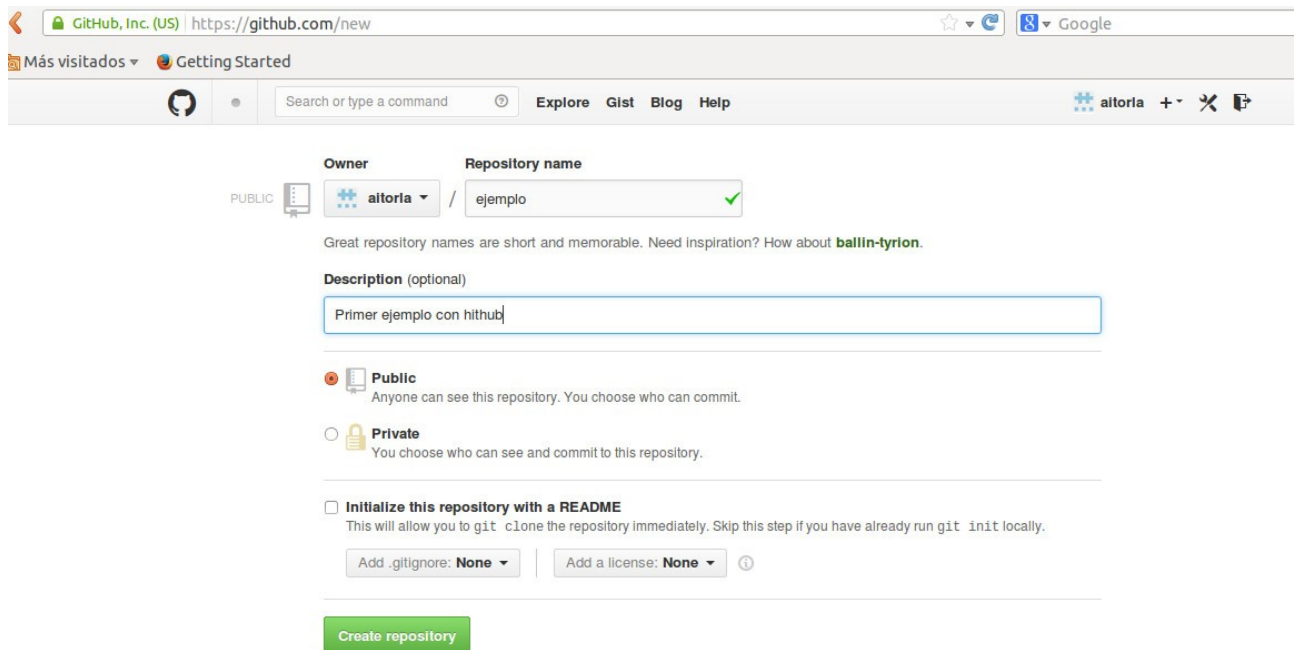
La idea es utilizar github.com para tener un repositorio en la nube.

1.- Nos registramos en hithub.com. Yo lo he hecho con el usuario aitorla.

Pinchamos dentro de nuestro usuario y dentro de la pestaña repositorios pinchamos en New:



Damos el nombre y la descripción a nuestro repositorio. Pinchamos en crear repositorio:



Owner: aitorla / Repository name: ejemplo

Great repository names are short and memorable. Need inspiration? How about **ballin-tyrion**.

Description (optional): Primer ejemplo con hithub

☒ Public  
Anyone can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

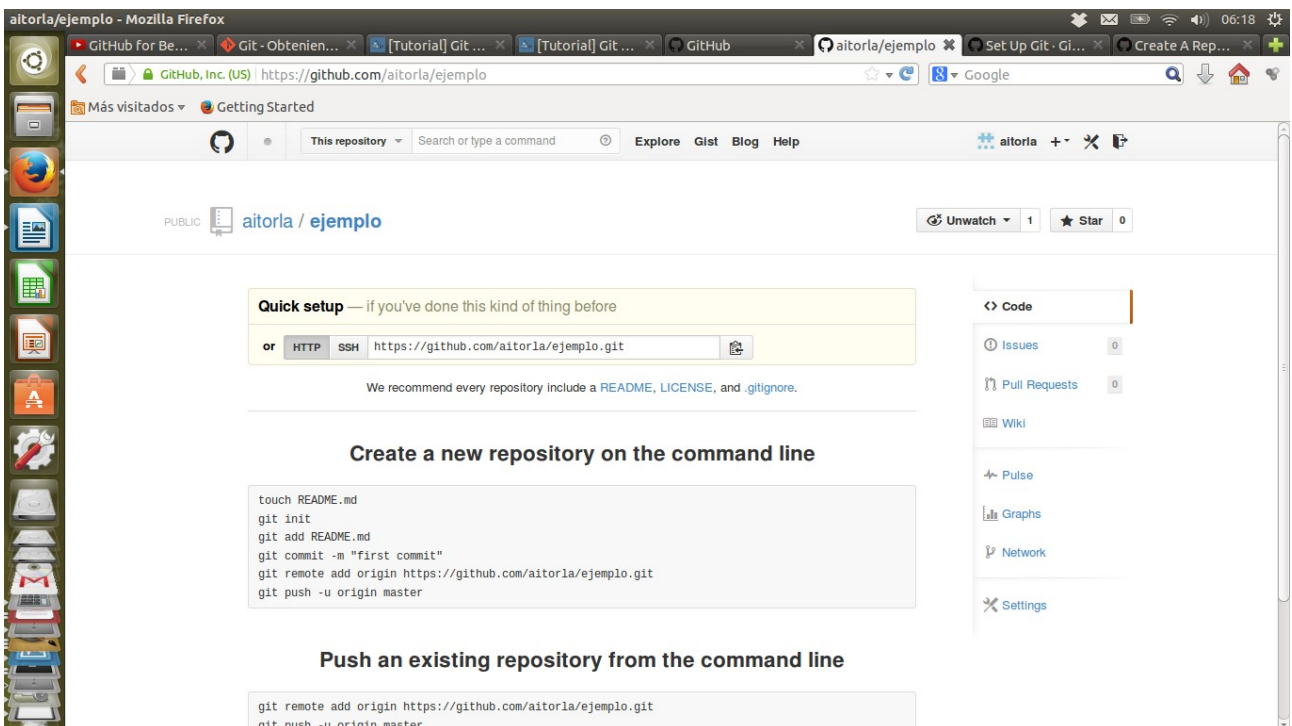
☐ Initialize this repository with a README  
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: None Add a license: None

Create repository

Vemos que nuestro repositorio se encuentra en la url:

<https://github.com/aitorla/ejemplo.git>



aitorla/ejemplo - Mozilla Firefox

GitHub, Inc. (US) | <https://github.com/aitorla/ejemplo>

Public aitorla / ejemplo

Unwatch 1 Star 0

**Quick setup** — if you've done this kind of thing before

or HTTP SSH `https://github.com/aitorla/ejemplo.git`

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**Create a new repository on the command line**

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/aitorla/ejemplo.git
git push -u origin master
```

**Push an existing repository from the command line**

```
git remote add origin https://github.com/aitorla/ejemplo.git
git push -u origin master
```

Code  
Issues 0  
Pull Requests 0  
Wiki  
Pulse  
Graphs  
Network  
Settings

Creamos un fichero README.me y lo enviamos a nuestro repositorio:

`touch README.md`

`git init`

```
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/aitorla/ejemplo.git
git push -u origin master
```

Username for 'https://github.com': aitorla

Password for 'https://aitorla@github.com':

Counting objects: 3, done.

Writing objects: 100% (3/3), 208 bytes, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/aitorla/ejemplo.git

\* [new branch] master -> master

Branch master set up to track remote branch master from origin.

### **Enviamos el repositorio**

```
git remote add origin https://github.com/aitorla/ejemplo.git
git push -u origin master
```

## **Git: Branch y Merges**

1.- Creamos una rama:

```
git checkout -b Rama1
```

Ahora hacemos cambios sobre la rama

```
git checkout Rama1
```

2.- Unimos las dos ramas:

```
git checkout master
```

```
git checkout Rama1
```

3.- Borramos la rama

```
git branch -d Rama1
```

Nota: para borrar una rama que no ha sido mezclada utilizaremos -D en lugar de d.

### **Ejercicios 1**

Prueba los sistemas de repositorios online BitBucket y Gitorious.

### **Ejercicios 2**

Integración de subversión con eclipse

<http://www.karmany.net/index.php/programacion-web/36-eclipse/182-integrar-subversion-svn-en-eclipse>

<http://proyectosbeta.net/2012/12/como-instalar-un-cliente-de-svn-en-eclipse-juno/>

### **Ejercicio 3**

Integración github con eclipse

<http://www.iesgrancapitan.org/dawblog/?p=425>

### **Ejercicio 4**

Instalación de un servidor git

<http://codehero.co/git-desde-cero-instalando-git-en-un-servidor/>

## **Forjas**

En la actualidad, existe un variado catalogo de paquetes software cuyo principal cometido es el de integrar diferentes servicios de soporte para el desarrollo de software, de manera que sean accesibles desde una interfaz web común. Estos paquetes, denominados forjas, han permitido que los desarrolladores de software puedan concentrarse en su principal cometido (la atención al proceso de desarrollo de sus programas), delegando la gestión de todos los servicios que soportan dicho proceso en la empresa o grupo que controla la forja y ofrece dicho servicio de alojamiento de proyectos.

La mayoría de las forjas cuentan en la actualidad con un conjunto básico de servicios que proporcionan a los grupos interesados en el desarrollo de software:

- Sistema de control de versiones para el código fuente (SVN, CVS, Git, etc.)
- Listas de correo para la comunicación entre los miembros del proyecto (usuarios, desarrolladores, etc.)
- Sistemas de seguimiento de fallos (bug-tracking).
- Wikis, que permiten el desarrollo colaborativo de documentación.
- Sistemas de publicación y descarga de archivos.

Sin embargo, el grado de madurez de las diferentes soluciones para el despliegue de forjas, así como la cohesión que consiguen entre los distintos elementos que la componen es en la actualidad muy dispar. Por todo ello, en primer lugar presentaremos un breve compendio comparativo de las soluciones software existentes para el despliegue de forjas, detallando las características más sobresalientes de cada una de ellas. A continuación, efectuaremos un resumen comparativo de estas soluciones, tratando de identificar la más adecuada en términos de madurez de la aplicación, estabilidad en la integración de servicios que ofrece y proyección futura, de cara a su mejora con la inclusión de nuevas funcionalidades.

Gforje, Sourceforge, collab.net, Savannah, BerliOS, Google Code, Launchpad, Novaforge, Oforge, Kforge, TiddlyForge.

## Sourceforge

SourceForge, fundado en 1999 por VA Software, es un sitio web de colaboración para proyectos de software. SourceForge es comercializado por Dice Holdings, desde el 18 de Septiembre de 2013. Provee una portada para un amplio rango de servicios útiles para los proceso de desarrollo de software e integra un amplio número de aplicaciones de software libre (tales como PostgreSQL y CVS).

SourceForge es una central de desarrollos de software que controla y gestiona varios proyectos de software libre y actúa como un repositorio de código fuente. SourceForge.net es hospedado por VA Software y corre en una versión del software SourceForge.

Ofrece alojamiento a proyectos tales como **Ares Galaxy** (cliente p2p), **FileZilla** (servidor y cliente ftp), **7-Zip** (descompresor), **phpMyAdmin** (gestor web de base de datos), etc.

<http://www.sourceforge.net>

## GForge

GForge es un software libre basado en la Web para la gestión colaborativa de proyectos de software creada originalmente para SourceForge y luego escindido en una rama (fork) distinta. GForge está licenciado bajo GPL (GNU General Public License).

GForge ofrece alojamiento de proyectos, control de versiones (CVS y Subversion), seguimiento de fallos y mensajería.

En febrero de 2009 algunos de los desarrolladores de GForge continuaron el desarrollo del anterior código abierto bajo el nuevo nombre de FusionForge luego de que el GForge Group se enfocara en GForge Advanced Server.

<https://gforgegroup.com/>

## Google Code

Google Code es un sitio de Google para desarrolladores interesados en el desarrollo Google-related/open-source. El sitio contiene códigos fuente abiertos, una lista de sus servicios de apoyo público y API.

Entre los proyectos albergados tenemos **Digg** (indexación de contenidos), **jquery** (librería javascript Ajax), **Firebug** (plugin Firefox para desarrolladores web), etc.

<http://code.google.com/intl/es-ES/>

## **Savannah**

GNU Savannah es un proyecto de la Free Software Foundation, que consiste en un sistema de gestión de proyectos de software libre, también llamado forja. Savannah ofrece servicios de CVS, GNU arch, Subversion, Git, Mercurial, 1 Bazaar, listas de correo, hospedaje web, hospedaje de archivos y seguimiento de bugs. Savannah ejecuta Savane, que está basado en el mismo software que utiliza el portal SourceForge.

El sitio web de Savannah está dividido en dos dominios: savannah.gnu.org para software oficial del proyecto GNU, y savannah.nongnu.org para todo software libre no perteneciente al proyecto.

A diferencia de SourceForge, Savannah se centra en el alojamiento de proyectos de software totalmente libre, es decir, libre de componentes no libres, como Flash; y para ello se muestra muy estricto en sus políticas de publicación, de manera que se asegure de que solo es alojado software libre. Al momento de registrar un proyecto, los colaboradores de éste han de especificar qué licencia de software libre usa.

En 2004, hubo una controversia acerca de la migración del proyecto GNU Savannah de software Savannah (ahora Savane) a GForge, debido a desacuerdos entre los propios colaboradores en relación a la falta de seguridad y de mantenibilidad del código.

## **Launchpad**

Launchpad es una plataforma de desarrollo colaborativo de software, en particular el de software libre a través de un sitio web como un servicio gratuito. Está desarrollada y mantenida por Canonical Ltd. El registro solo es necesario si se desea comentar, o subir nuevos reportes de errores.

El 9 de julio del 2007, Canonical anunció<sup>1</sup> la liberación de Storm con la licencia libre LGPL 2.1. Storm, primer componente de Launchpad liberado, es un mapeador para Python de objetos a sistemas relacionales (ORM), que ayuda a simplificar el desarrollo de aplicaciones que funcionen sobre una base de datos. Este desarrollo está basado en Zope.

El 21 de julio de 2009 Launchpad pasó a ser completamente libre, bajo la versión 3 de la licencia GNU Affero General Public license.<sup>2</sup>

Consta de varias partes:

Code: un sitio de alojamiento de código fuente que utiliza el sistema de control de versiones Bazaar.

Bugs: un sistema de seguimiento de errores para informar sobre bugs en diferentes distribuciones y productos.

Blueprints: un sistema de seguimiento para especificaciones y nuevas características.

Translations: un sitio para traducir aplicaciones a múltiples idiomas.

Answers: un sitio de ayuda para la comunidad.

Soyuz: una herramienta para llevar una pequeña parte del mantenimiento de las distribuciones. Abarca el sistema de construcción, el mantenimiento de paquetes y la publicación de archivos.

Launchpad es usada primordialmente para el desarrollo de Ubuntu y sus derivados oficiales, aunque también contempla otras distribuciones y proyectos independientes. Launchpad utiliza el servidor de aplicaciones web gratuito y de código abierto Zope.

## Fases del desarrollo del software: alpha, beta, release, candidate, estable.

En la ingeniería del software el término fases de desarrollo expresa cómo ha progresado el desarrollo de un software y cuánto desarrollo puede requerir. Cada versión importante de un producto pasa generalmente a través de una etapa en la que se agregan las nuevas características (**etapa alfa**), después una etapa donde se eliminan errores activamente (**etapa beta**), y finalmente una etapa en donde se han quitado todos los bugs importantes (**etapa estable**). Las etapas intermedias pueden también ser reconocidas. Las etapas se pueden anunciar y regular formalmente por los desarrolladores del producto, pero los términos se utilizan a veces de manera informal para describir el estado de un producto. Normalmente muchas compañías usan nombres en clave para las versiones antes del lanzamiento de un producto, aunque el producto y las características reales son raramente secretas.

- **Versión ALFA:** Es la primera versión del programa. El producto todavía es inestable y se está a la espera de que se eliminen los errores o a la puesta en práctica completa de toda su funcionalidad, pero satisface la mayoría de los requisitos.
- **Versión BETA:** Representa la primera versión completa del programa. Los desarrolladores las lanzan a un grupo de probadores, a veces al público en general, para que lo prueben e informen de cualquier error que encuentren y propongan características que quisieran encontrar en la versión final.
- **Versión candidata a definitiva o RC:** Comprende un producto final preparado para lanzarse como versión definitiva a menos que aparezcan errores que lo impidan.
- **Versión RTM ó de disponibilidad general:** Es la versión final del programa. Normalmente es casi idéntica a la versión RC, con sólo correcciones de último momento. Esta versión es considerada muy estable, relativamente libre de errores y con una calidad adecuada para una distribución amplia y para ser ya utilizada por los usuarios finales.

En la programación de código abierto los números de las versiones, o los términos estable e inestable, normalmente distinguen las fases del desarrollo. En el pasado, el núcleo Linux usaba el número de versión para denotar si una versión era **estable o inestable**. En efecto, las versiones estaban formada por cuatro números, separados por un punto. Una cifra impar en el segundo número de la versión indicaba una **versión inestable**. Hoy en día ya no se usa esta convención, y todas las versiones son estables independientemente del número de versión. En la práctica el uso de números pares e impares para indicar la estabilidad de un producto ha sido usado por otros muchos proyectos de software libre. Este concepto también se aplica al software empaquetado en algunas distribuciones Linux como Debian, de modo que existe una rama o conjunto de paquetes considerados estables y otra rama considerada inestable. Esta última rama



aporta versiones de programas más recientes que la estable pero que no están tan probados.

## Generadores de documentación

Un generador de documentación es una herramienta de programación que genera documentación destinada a los programadores (documentación de API) o a usuarios finales, o a ambos, a partir de un conjunto de código fuente especialmente documentado, y en algunos casos, archivos binarios.

Si estamos desarrollando un proyecto medio-grande este tiene que ser modificado no sólo para otros programadores sino muchas veces para nosotros mismos.

Entre las herramientas de documentación más utilizadas tenemos PhpDocumentor, Docblox o Doxygen.

### PhpDocumentor

phpDocumentor es un generador de documentación de código abierto escrito en PHP. Automáticamente analiza el código fuente PHP y produce la API de lectura y documentación del código fuente en una variedad de formatos. phpDocumentor genera la documentación en base al estándar formal PHPDoc. Es compatible con la documentación del código orientado a objetos y programación procedural, además es capaz de crear documentos HTML, PDF, CHM y formatos Docbook. Se puede utilizar desde la línea de comandos o mediante una interfaz web. Tiene soporte para la vinculación entre la documentación, la incorporación de documentos a nivel de usuario como tutoriales, y la creación de código fuente resaltado con referencias cruzadas a la documentación en general de PHP. phpDocumentor es capaz de analizar toda la sintaxis de PHP y apoya PHP4 y PHP5. Se trata de un proyecto de código abierto y se distribuye bajo la licencia LGPL.

```
sudo apt-get install libapache2-mod-php5  
sudo /etc/init.d/apache2 restart
```

```
Instalamos phpDocumentor con phpPear  
sudo apt-get install php-pear  
sudo pear install --alldeps PhpDocumentor
```

```
sudo mkdir /var/www/PhpDocumentor  
sudo mkdir /var/www/PhpDocumentor/Documentacion
```

```
sudo chown www-data:www-data /var/www/PhpDocumentor/Documentacion  
sudo chmod 755 /var/www/PhpDocumentor/Documentacion
```

Documentamos squirrelmail (lo instalamos antes en /var/www/squirrelmail)

```
phpdoc -d /var/www/squirrelmail/ -t /var/www/PhpDocumentor/Documentacion
```

Podemos acceder con un navegador a:

<http://localhost/PhpDocumentor/Documentacion>

Podemos desinstalar Documentor así:

```
sudo pear uninstall PhpDocumentor
```

Comparativa de generadores de documentación

[http://es.wikipedia.org/wiki/Anexo:Comparativa\\_de\\_generadores\\_de\\_documentaci%C3%B3n](http://es.wikipedia.org/wiki/Anexo:Comparativa_de_generadores_de_documentaci%C3%B3n)

Documentación

<http://trucosinformaticos.wordpress.com/tag/documentacion/>

Introducción a la documentación

<http://blog.rhiss.net/introduccion-documentacion-php.html>