

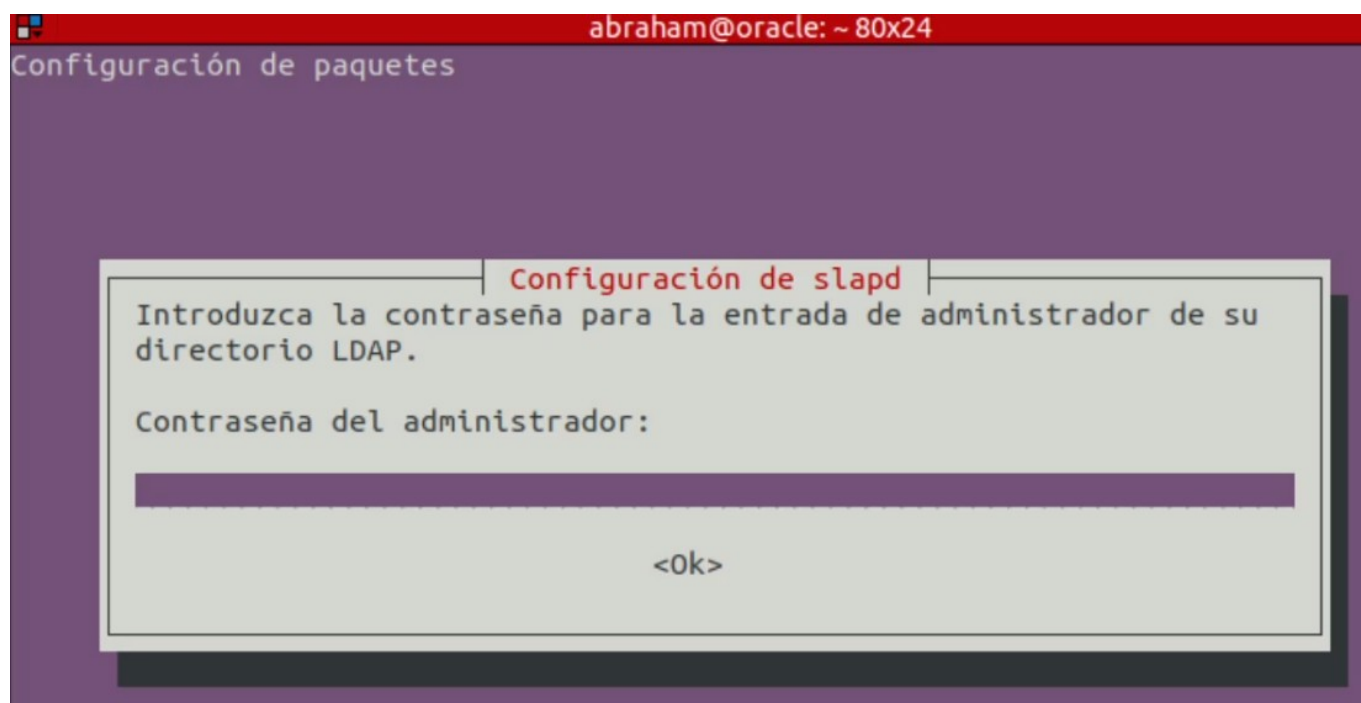
MPT2. Autenticación externa a traves de OpenLDAP

En esta practica vamos a instalar un servicio de OpenLDAP para podernos autenticarnos en la base de datos PostgreSQL. Lo primero que vamos a hacer es instalar el servicio OpenLDAP para ver la instalación de [Postgres mirar documentacion de Oracle Database Link](#)

1. OpenLDAP

El proceso de instalación es realmente sencillo. Vamos a instalar los siguientes paquetes **'sudo apt install slapd ldap-utils'**

Durante la instalación, aparece en la consola un mensaje que nos solicita la contraseña de administración para LDAP.



Al hacerlo, volveremos al aspecto normal de la consola y comprobaremos que la instalación sigue en curso. De forma predeterminada, slapd se configura con las mínimas opciones necesarias para que el demonio funcione de forma correcta.

Comenzaremos por modificar el contenido del archivo '/etc/hosts'. Dentro del archivo, añadimos una nueva línea que relacione la dirección IP estática del servidor con el nombre lógico que tenemos previsto utilizar.

```
abraham@ldap:~$ cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 ldap
192.168.122.22 ldapserver.local
```

Instalamos luego la librería libnss-ldap 'sudo apt install libnss-ldap'

Luego ponemos dpkg-reconfigure slapd:

La primera opción le damos a NO.

Luego nos saldrá esta ventana donde pondremos el dominio:

Configuración de slapd

El nombre de dominio DNS se utiliza para construir el DN base del directorio LDAP. Por ejemplo, si introduce «foo.example.org» el directorio se creará con un DN base de «dc=foo, dc=example, dc=org».

Introduzca el nombre de dominio DNS:

ldapserver.local

<Ok>

Configuración de slapd

Introduzca el nombre de la organización a utilizar en el DN base del directorio LDAP.

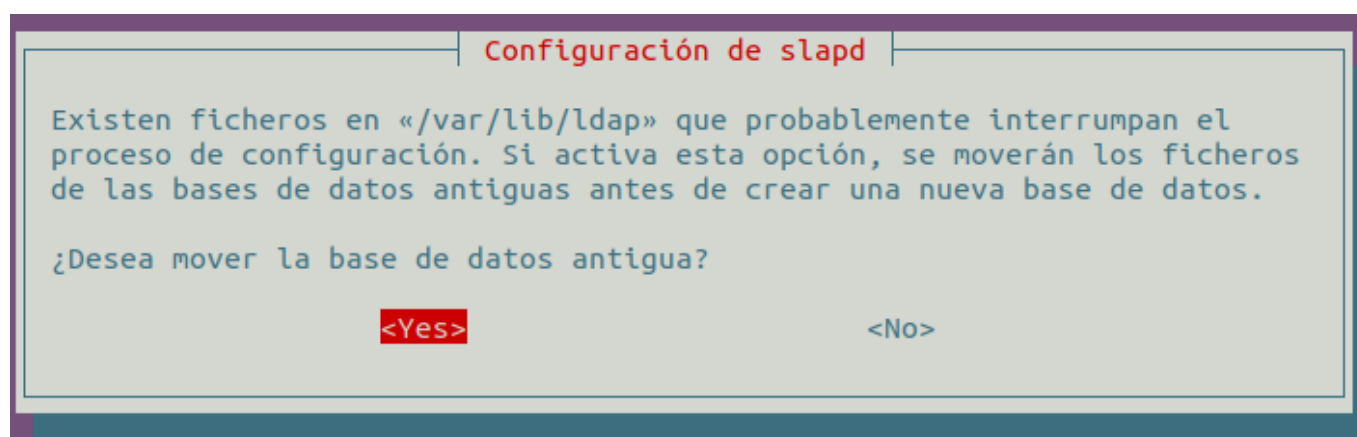
Nombre de la organización:

ldapserver.local

<Ok>

Pondremos la contraseña de administrador, y luego nos preguntara que si desea que borre la base de datos y se purge le decimos que NO.

Después no saldrá esto:



Luego nos saldrá para escoger la versión que sera la 3 la que elegiremos.

Ahora nos iremos al fichero de configuración de ldap y estamos el fichero poniendo de base nuestro dominio separado por dc= y la URI que en nuestro caso seria la dirección IP local + el puerto 389 que es el predeterminado.

```
GNU nano 4.8 /etc/ldap/ldap.conf
#
# LDAP Defaults
#
# See ldap.conf(5) for details
# This file should be world readable but not world writable.

BASE    dc=ldapserver,dc=local
URI     ldap://192.168.122.22:389

#SIZELIMIT    12
#TIMELIMIT    15
#DEREF        never
```

Hay que instalar apache, si lo tenemos instalado hay que reiniciar el servicio 'systemctl restart apache2'. Después abrimos los puertos 80 y 389 con ufw.

Para verificar la instalación de OpenLDAP podemos poner `ldapsearch -x`:

Nos saldra algo como esto

```
abraham@ldap:~$ ldapsearch -x
# extended LDIF
#
# LDAPv3
# base <dc=ldapserver,dc=local> (default) with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# ldapserver.local
dn: dc=ldapserver,dc=local
objectClass: top
objectClass: dcObject
objectClass: organization
o: ldapserver.local
dc: ldapserver
# admin, ldapserver.local
dn: cn=admin,dc=ldapserver,dc=local
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
```

Finalmente ya estaria el servicio de LDAP instalado. Ahora procedemos a instalar el gestor de LDAP como un dashboard en mi caso utilizo **phpLDAPadmin**

2. phpLDAPadmin

La instalación es muy sencilla ponemos `'sudo apt install phpldapadmin -y'` una vez se haya acabado la instalación del paquete creamos un link simbólico con el siguiente comando `'ln -s /usr/share/phpldapadmin/ /var/www/html/phpldapadmin'`

Ahora vamos a configurar el phpLDAPadmin en `'nano /etc/phpldapadmin/config.php'` y tenemos que descomentar `config->custom->appearance['timezone'] = 'America/Los_Angeles';` y poner nuestra zona:

```
// $config->custom->appearance['timezone'] = null;  
$config->custom->appearance['timezone'] = 'Europe/Madrid';
```

Y configurar las siguientes líneas:

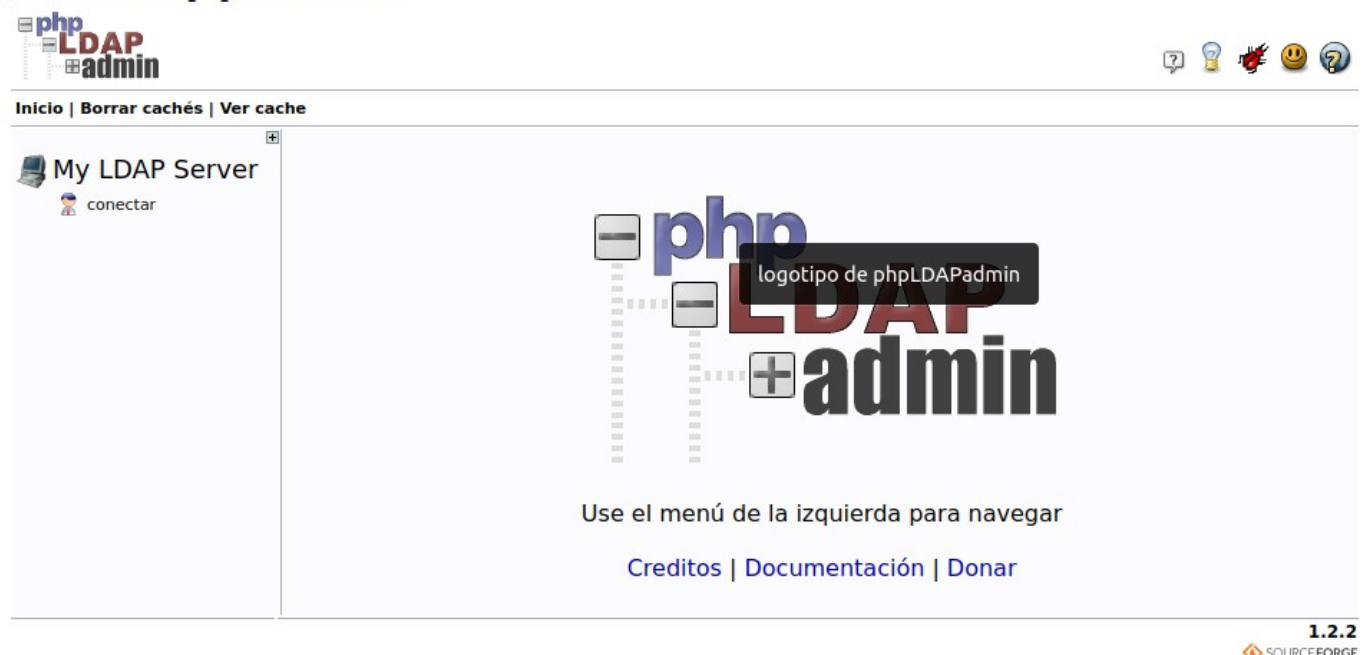
```
(Unix socket at /usr/local/var/run/ldap) */  
$servers->setValue('server','host','192.168.122.22');
```

```
$servers->setValue('server','base',array('dc=ldapserver,dc=local'));
```

```
$servers->setValue('login','bind_id','cn=admin,dc=ldapserver,dc=local');
```

Una vez hecho esto reiniciamos el servicio apache de nuevo y ya estaria.

Deprecated: Array and string offset access syntax with curly braces is deprecated in `/usr/share/phpldapadmin/lib/functions.php` on line 1614



Nos conectamos con admin y contraseña

Aquí vemos la organizaciones y grupos y usuarios que tenemos:



Para crear una organización le damos a crear nuevo objeto y luego le damos a unidad organizacional. Ponemos nombre a la organización

Luego le damos de nuevo a crear nuevo objeto dentro de nuestra organización y le damos a grupo posix y le damos un nombre:

The screenshot shows a web form with three main sections: 'Número GID', 'Grupo', and 'Usuarios'. The 'Número GID' section has a text input field containing '501'. The 'Grupo' section has a text input field containing 'grupoprueba'. The 'Usuarios' section has a checkbox next to the text 'Rocio Vegas (rveg)'. At the bottom of the form is a button labeled 'Crear objeto'. The form is styled with a light gray background and rounded corners. The 'Número GID' and 'Grupo' sections have a header bar with the label and a list of attributes (alias, requerido, nota, rdn). The 'Usuarios' section has a header bar with the label and a list of attributes (alias, nota). The 'Rocio Vegas (rveg)' text is preceded by a small square checkbox.

Número GID alias, requerido, nota, rdn

501

Grupo alias, requerido, rdn

grupoprueba *

Usuarios alias, nota

☐ Rocio Vegas (rveg)

Crear objeto

Despues le damos dentro del grupo a crear objeto en este hilo y ya rellenamos este formulario:

Crear objeto

Servidor: **My LDAP Server** Contenedor: **cn=grupoprueba,ou=prueba,dc=ldapserver,dc=local**
plantilla: **Generic: User Account (posixAccount)**

Nueva cuenta de usuario (Paso 1 de 1)


Nombre común

alias, requerido, rdn

*

Nombre propio

alias



Número GID

alias, requerido, nota

*

Directorio personal

alias, requerido

*

Apellido

alias, requerido


*

Consola de Login

alias

Contraseña

alias, nota




md5

(confirmar)

[Revisar clave...](#)

Número UID

alias, requerido, nota, ro

 1001

ID del Usuario

alias, requerido

*

Nos quedaria algo como esto:

cn	requerido, rdn
Rocio Vegas *	
(añadir valor)	
(renombrar)	
gidNumber	requerido
<input type="text" value="500"/>	
usuarios ()	
givenName	
<input type="text" value="Rocio"/>	
(añadir valor)	
homeDirectory	requerido
<input type="text" value="/home/users/rvegas"/>	
loginShell	
<input type="text" value="/bin/sh"/>	
objectClass	requerido
i inetOrgPerson (estructural)	
i <input type="text" value="posixAccount"/>	
i <input type="text" value="top"/>	
(añadir valor)	
Contraseña	alias
<input type="password" value="....."/>	
<input type="button" value="md5"/>	
Revisar clave...	
(añadir valor)	
sn	requerido
<input type="text" value="Vegas"/>	
(añadir valor)	
uidNumber	requerido
<input type="text"/>	

Para ver que funciona tendremos que poner esto en nuestra terminal:

```
abraham@ldap:~$ su - rvegas
Password:
su: warning: cannot change directory to /home/users/rvegas: No such file
ctory
$
```

Vemos que conecta con el usuario creado de LDAP en consola sh. Ya estaría todo finalmente realizado.

3. Autenticación LDAP en PostgreSQL

Para añadir la autenticación en postgresql tendremos que modificar el archivo pg_hba.conf

Y añadir la línea de conexión de postgresql con ldap.

```
# Database administrative login by Unix domain socket
local    all             postgres                                peer
host     all             all 0.0.0.0/0 ldap ldapserver=ldapserver.local ldapbasedn="dc=ldapserver, dc=local" ldapsearchattribute=uid
# TYPE  DATABASE  USER  ADDRESS  METHOD
```

Una vez hecho esto la conexión se ha realizado reiniciamos postgresql. Ahora tendremos que crear un rol por cada usuario de LDAP con el mismo nombre de usuario, es decir, si mi usuario es rvegas el rol se llamara rvegas.

```
abraham@ldap:~$ sudo -u postgres psql
psql (12.6 (Ubuntu 12.6-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# create role rvegas login;
```

Una vez creado el rol, podemos entrar con el usuario al esquema postgres ya que ese usuario no tiene asignado ningun esquema:

```
abraham@ldap:~$ psql -h 192.168.122.22 -U rvegas -d postgres
Password for user rvegas:
psql (12.6 (Ubuntu 12.6-0ubuntu0.20.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, c
ompression: off)
Type "help" for help.

postgres=> █
```

4. pg-ldap-sync

Vamos a utilizar este programa de github para no tener que ir creando rol por cada usuario.

Este programa ayuda a resolver el problema sincronizando usuarios, grupos y sus membresías desde LDAP a PostgreSQL. El acceso a LDAP se utiliza de sólo lectura. pg_ldap_sync emite los comandos CREATE ROLE, DROP ROLE, GRANT y REVOKE adecuados para sincronizar usuarios y grupos.

Para ello necesitamos varios paquetes antes de instalarlo:

El primero es instalar ruby, no esta en los repositorios por lo tanto lo instalamos con snap:

```
sudo snap install ruby --classic
```

Luego necesitamos jruby para ello hacemos 'sudo apt update' y luego 'sudo apt install jruby'.

Cuando instalemos todo lo anterior deberemos de instalar bundler con ruby utilizando 'sudo gem install bundler'

Ahora nos descargamos pg-ldap-sync de github <https://github.com/larskanis/pg-ldap-sync>

```
git clone https://github.com/larskanis/pg-ldap-sync.git
```

Y ejecutamos el comando bundle install, nos saldra un error de versiones no compatibles para ellos editamos el fichero pg-ldap-sync.gemspec y editamos la linea siguiente:

```
spec.add_development_dependency "bundler", "~> 2.2.17"
```

Esto indica que vamos a utilizar la version actual de bundler. Una vez hecho esto tiramos el comando de bundle install y nos saldra otro error:

```
Fetching pg 1.2.3
Installing pg 1.2.3 with native extensions
Gem::Ext::BuildError: ERROR: Failed to build gem native extension.

current directory:
/tmp/bundler20210521-31272-ozqwmpg-1.2.3/gems/pg-1.2.3/ext
/usr/bin/ruby2.7 -I /usr/lib/ruby/2.7.0 -r ./siteconf20210521-31272-18lricn.rb
extconf.rb
mkmf.rb can't find header files for ruby at /usr/lib/ruby/include/ruby.h

You might have to install separate package for the ruby development
environment, ruby-dev or ruby-devel for example.

extconf failed, exit code 1

Gem files will remain installed in
/tmp/bundler20210521-31272-ozqwmpg-1.2.3/gems/pg-1.2.3 for inspection.
Results logged to
```

```
An error occurred while installing pg (1.2.3), and Bundler cannot
continue.
Make sure that `gem install pg -v '1.2.3' --source 'https://rubygems.org/'`
succeeds before bundling.
```

Hacemos caso a lo que nos dice y metemos el comando ‘sudo gem install pg -v ‘1.2.3’ -source [‘https://rubygems.org’](https://rubygems.org)”

Nos saldrá un error el cual tendremos que instalar unas dependencias que son estas:

```
sudo apt install libpq-dev
```

```
sudo apt install ruby-dev
```

```
sudo apt install build-essential
```

Una vez instalado todo ya nos dejara tirar el comando de instalacion anterior de ruby.

Procedemos a hacer lo mismo que con bundler editar el fichero pg-ldap-sync.gemspec y poner nuestra version de pg (prevenir antes que curar).

```
spec.add_runtime_dependency "pg", "1.2.3"
```

Y ahora si podremos hacer bundle install:

```
Fetching ruby-ldapserver 0.5.3
Installing ruby-ldapserver 0.5.3
Bundle complete! 6 Gemfile dependencies, 9 gems now installed.
Use `bundle info [gemname]` to see where a bundled gem is installed.
abraham@ldap:~/pg-ldap-sync$
```

ahora ejecutamos `bundle exec rake install` para la instalacion de `pg-ldap-sync`

```
abraham@ldap:~/pg-ldap-sync$ bundle exec rake install
pg-ldap-sync 0.2.0 built to pkg/pg-ldap-sync-0.2.0.gem.
rake aborted!
Couldn't install gem, run `gem install /home/abraham/pg-ldap-sync/pkg/pg-ldap-sync-0.2.0.gem` for more detailed output
/var/lib/gems/2.7.0/gems/bundler-2.2.17/lib/bundler/gem_helper.rb:103:in `install_gem'
/var/lib/gems/2.7.0/gems/bundler-2.2.17/lib/bundler/gem_helper.rb:57:in `block in install'
/usr/share/rubygems-integration/all/gems/rake-13.0.1/exe/rake:27:in `'
Tasks: TOP => install
(See full trace by running task with --trace)
```

Nos sale que rake abortó pero que ejecutemos un comando, lo hacemos:

```
abraham@ldap:~/pg-ldap-sync$ sudo gem install /home/abraham/pg-ldap-sync/pkg/pg-ldap-sync-0.2.0.gem
Successfully installed pg-ldap-sync-0.2.0
Parsing documentation for pg-ldap-sync-0.2.0
Installing ri documentation for pg-ldap-sync-0.2.0
Done installing documentation for pg-ldap-sync after 0 seconds
1 gem installed
abraham@ldap:~/pg-ldap-sync$ which pg_ldap_sync
/usr/local/bin/pg_ldap_sync
```

Y ya funciona. Ahora tenemos que tener la configuracion en '/etc' la configuración es tal que asi:

```
GNU nano 4.8 /etc/pg_ldap_s
# With this sample config the distinction between LDAP-synchronized
# groups/users from is done by the membership to ldap_user and
# ldap_group. These two roles has to be defined manually before
# pg_ldap_sync can run.

# Connection parameters to LDAP server
# see also: http://net-ldap.rubyforge.org/Net/LDAP.html#method-c-new
ldap_connection:
  host: 192.168.122.22
  port: 389
  auth:
    method: :simple
    username: CN=admin,DC=ldapserver,DC=local
    password: 123456
  encryption:
    method: :simple_tls

# Search parameters for LDAP users which should be synchronized
ldap_users:
  base: OU=postgres,DC=ldapserver,DC=local
  # LDAP filter (according to RFC 2254)
  # defines to users in LDAP to be synchronized
  filter: (&(objectClass=person)(objectClass=organizationalPerson)(givenName=*)(sn=*)(sAMAccountName=*))
  # this attribute is used as PG role name
  name_attribute: sAMAccountName
  # lowercase name for use as PG role name
  lowercase_name: true

# Search parameters for LDAP groups which should be synchronized
ldap_groups:
  base: OU=postgres,DC=ldapserver,DC=local
  filter: (cn=usuarios)
  # this attribute is used as PG role name
  name_attribute: cn
  # lowercase name for use as PG role name
  lowercase_name: false
  # this attribute must reference to all member DN's of the given group
  member_attribute: member

# Connection parameters to PostgreSQL server
# see also: http://rubydoc.info/gems/pg/PG/Connection#initialize-instance_method
pg_connection:
  host: 192.168.122.22
  dbname: postgres
  user: postgres
  password: 123456
```

```
pg_users:
  # Filter for identifying LDAP generated users in the database.
  # It's the WHERE-condition to "SELECT rolname, oid FROM pg_roles"
  filter: oid IN (SELECT pam.member FROM pg_auth_members pam JOIN pg_roles pr ON pr.oid=pam.roleid WHERE pr.rolname='ldap_users')
  # Options for CREATE RULE statements
  create_options: LOGIN IN ROLE ldap_users

pg_groups:
  # Filter for identifying LDAP generated groups in the database.
  # It's the WHERE-condition to "SELECT rolname, oid FROM pg_roles"
  filter: oid IN (SELECT pam.member FROM pg_auth_members pam JOIN pg_roles pr ON pr.oid=pam.roleid WHERE pr.rolname='ldap_groups')
  # Options for CREATE RULE statements
  create_options: NOLOGIN IN ROLE ldap_groups
  grant_options:
```

Ahora solo tendríamos que ejecutar este comando 'pg_ldap_sync -c /etc/pg_ldap_sync -vv'

En mi caso nos sale un error de SSL en el cual me he documentado que es de versiones incompatibles, he intentado de actualizar a la version mas reciente de SSL en Ruby pero sigue saliendo el error. Aun asi en el fichero anterior utilizo en method la opcion de plain para no usar conexión SSL y me sigue dando este error:

```
n socket'  
' : unsupported encryption method plain (Net::LDAP::Error)
```