OpenLDAP Server

LDAP is an acronym for Lightweight Directory Access Protocol, it is a simplified version of the X.500 protocol. The directory setup in this section will be used for authentication. Nevertheless, LDAP can be used in numerous ways: authentication, shared directory (for mail clients), address book, etc.

To describe LDAP quickly, all information is stored in a tree structure. With **OpenLDAP** you have freedom to determine the directory arborescence (the Directory Information Tree: the DIT) yourself. We will begin with a basic tree containing two nodes below the root:

- "People" node where your users will be stored
- "Groups" node where your groups will be stored

Before beginning, you should determine what the root of your LDAP directory will be. By default, your tree will be determined by your Fully Qualified Domain Name (FQDN). If your domain is example.com (which we will use in this example), your root node will be dc=example,dc=com.

Installation

First, install the **OpenLDAP** server daemon **slapd** and **ldap-utils**, a package containing LDAP management utilities:

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

The installation process will prompt you for the LDAP directory admin password and confirmation.

By default the directory suffix will match the domain name of the server. For example, if the machine's Fully Qualified Domain Name (FQDN) is ldap.example.com, the default suffix will be *dc=example,dc=com*. If you require a different suffix, the directory can be reconfigured using **dpkg-reconfigure**. Enter the following in a terminal prompt:

```
sudo dpkg-reconfigure slapd
```

You will then be taken through a menu based configuration dialog, allowing you to configure various **slapd** options.

Configuration

OpenLDAP uses a separate database which contains the *cn=config* Directory Information Tree (DIT). The *cn=config* DIT is used to dynamically configure the **slapd** daemon, allowing the modification of schema definitions, indexes, ACLs, etc without stopping the service.

The *cn=config* tree can be manipulated using the utilities in the **ldap-utils** package. For example:

- Use **ldapsearch** to view the tree, entering the admin password set during installation or reconfiguration:

```
ldapsearch -xLLL -b cn=config -D cn=admin,cn=config -W olcDatabase={1}hdb
```

```
Enter LDAP Password:
dn: olcDatabase={1}hdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcHdbConfig
olcDatabase: {1}hdb
olcDbDirectory: /var/lib/ldap
olcSuffix: dc=example,dc=com
olcAccess: {0}to attrs=userPassword,shadowLastChange by dn="cn=admin,dc=exampl
 e,dc=com" write by anonymous auth by self write by * none
olcAccess: {1}to dn.base="" by * read
olcAccess: {2}to * by dn="cn=admin,dc=example,dc=com" write by * read
olcLastMod: TRUE
olcDbCheckpoint: 512 30
olcDbConfig: {0}set_cachesize 0 2097152 0
olcDbConfig: {1}set_lk_max_objects 1500
```

```
olcDbConfig: {2}set_lk_max_locks 1500
olcDbConfig: {3}set_lk_max_lockers 1500
olcDbIndex: objectClass eq
```

The output above is the current configuration options for the *hdb* backend database. Which in this case containes the *dc=example,dc=com* suffix.

- Refine the search by supplying a filter, in this case only show which attributes are indexed:

```
ldapsearch -xLLL -b cn=config -D cn=admin,cn=config -W olcDatabase={1}hdb olcDbIndex
```

```
Enter LDAP Password:
dn: olcDatabase={1}hdb,cn=config
olcDbIndex: objectClass eq
```

- As an example of modifying the *cn=config* tree, add another attribute to the index list using **ldapmodify**:

```
ldapmodify -x -D cn=admin,cn=config -W
```

```
Enter LDAP Password:
dn: olcDatabase={1}hdb,cn=config
add: olcDbIndex
olcDbIndex: cn eq,pres,sub

modifying entry "olcDatabase={1}hdb,cn=config"
```

Once the modification has completed, press *Ctrl+D* to exit the utility.

- **ldapmodify** can also read the changes from a file. Copy and paste the following into a file named `uid_index.ldif`:

```
dn: olcDatabase={1}hdb,cn=config
add: olcDbIndex
olcDbIndex: uid eq,pres,sub
```

Then execute **ldapmodify**:

```
ldapmodify -x -D cn=admin,cn=config -W -f uid_index.ldif
```

```
Enter LDAP Password:
modifying entry "olcDatabase={1}hdb,cn=config"
```

The file method is very useful for large changes.

- Adding additional *schemas* to **slapd** requires the schema to be converted to LDIF format. Fortunately, the **slapd** program can be used to automate the conversion. The following example will add the *misc.schema*:

  1. First, create a conversion `schema_convert.conf` file containing the following lines:

```
include /etc/ldap/schema/core.schema
include /etc/ldap/schema/collective.schema
include /etc/ldap/schema/corba.schema
include /etc/ldap/schema/cosine.schema
include /etc/ldap/schema/duaconf.schema
include /etc/ldap/schema/dyngroup.schema
include /etc/ldap/schema/inetorgperson.schema
include /etc/ldap/schema/java.schema
include /etc/ldap/schema/misc.schema
include /etc/ldap/schema/nis.schema
include /etc/ldap/schema/openldap.schema
include /etc/ldap/schema/ppolicy.schema
```

  2. Next, create a temporary directory to hold the output:

```
mkdir /tmp/ldif_output
```

3. Now using **slaptest** convert the schema files to LDIF:

```
slaptest -f schema_convert.conf -F /tmp/ldif_output
```

Adjust the configuration file name and temporary directory names if yours are different. Also, it may be worthwhile to keep the `ldif_output` directory around in case you want to add additional schemas in the future.

4. Edit the `/tmp/ldif_output/cn=config/cn=schema/cn={8}misc.ldif` file, changing the following attributes:

```
dn: cn=misc,cn=schema,cn=config
...
cn: misc
```

And remove the following lines from the bottom of the file:

```
structuralObjectClass: olcSchemaConfig
entryUUID: 10dae0ea-0760-102d-80d3-f9366b7f7757
creatorsName: cn=config
createTimestamp: 20080826021140Z
entryCSN: 20080826021140.791425Z#000000#000#000000
modifiersName: cn=config
modifyTimestamp: 20080826021140Z
```


The attribute values will vary, just be sure the attributes are removed.

5. Finally, using the **ldapadd** utility, add the new schema to the directory:

```
ldapadd -x -D cn=admin,cn=config -W -f /tmp/ldif_output/cn\=config/cn\=schema/cn\=\{8\
```

There should now be a *dn: cn={4}misc,cn=schema,cn=config* entry in the cn=config tree.

Populating LDAP

The directory has been created during installation and reconfiguration, and now it is time to populate it. It will be populated with a "classical" scheme that will be compatible with address book applications and with Unix Posix accounts. Posix accounts will allow authentication to various applications, such as web applications, email Mail Transfer Agent (MTA) applications, etc.


For external applications to authenticate using LDAP they will each need to be specifically configured to do so. Refer to the individual application documentation for details.

LDAP directories can be populated with LDIF (LDAP Directory Interchange Format) files. Copy the following example LDIF file, naming it `example.com.ldif`, somewhere on your system:

```
dn: ou=people,dc=example,dc=com
objectClass: organizationalUnit
ou: people

dn: ou=groups,dc=example,dc=com
objectClass: organizationalUnit
ou: groups

dn: uid=john,ou=people,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: john
sn: Doe
givenName: John
cn: John Doe
displayName: John Doe
uidNumber: 1000
gidNumber: 10000
userPassword: password
gecos: John Doe
```

```
loginShell: /bin/bash
homeDirectory: /home/john
shadowExpire: -1
shadowFlag: 0
shadowWarning: 7
shadowMin: 8
shadowMax: 999999
shadowLastChange: 10877
mail: john.doe@example.com
postalCode: 31000
l: Toulouse
o: Example
mobile: +33 (0)6 xx xx xx xx
homePhone: +33 (0)5 xx xx xx xx
title: System Administrator
postalAddress:
initials: JD

dn: cn=example,ou=groups,dc=example,dc=com
objectClass: posixGroup
cn: example
gidNumber: 10000
```

In this example the directory structure, a user, and a group have been setup. In other examples you might see the *objectClass: top* added in every entry, but that is the default behaviour so you do not have to add it explicitly.

To add the entries to the LDAP directory use the **ldapadd** utility:

```
ldapadd -x -D cn=admin,dc=example,dc=com -W -f example.com.ldif
```

We can check that the content has been correctly added with the tools from the **ldap-utils** package. In order to execute a search of the LDAP directory:

```
ldapsearch -xLLL -b "dc=example,dc=com" uid=john sn givenName cn

dn: uid=john,ou=people,dc=example,dc=com
cn: John Doe
sn: Doe
givenName: John
```

Just a quick explanation:

- *-x:* will not use SASL authentication method, which is the default.
- *-LLL:* disable printing LDIF schema information.

LDAP replication

LDAP often quickly becomes a highly critical service to the network. Multiple systems will come to depend on LDAP for authentication, authorization, configuration, etc. It is a good idea to setup a redundant system through replication.

Replication is achieved using the *Syncrepl* engine. Syncrepl allows the directory to be synced using either a *push* or *pull* based system. In a push based configuration a “primary” server will push directory updates to “secondary” servers, while a pull based approach allows replication servers to sync on a time based interval.

The following is an example of a *Multi-Master* configuration. In this configuration each OpenLDAP server is configured for both *push* and *pull* replication.

1. First, configure the server to sync the *cn=config* database. Copy the following to a file named `syncrepl_cn-config.ldif`:

   ```
   dn: cn=module{0},cn=config
   changetype: modify
   add: olcModuleLoad
   olcModuleLoad: syncprov

   dn: cn=config
   changetype: modify
   replace: olcServerID
   olcServerID: 1 ldap://ldap01.example.com
   olcServerID: 2 ldap://ldap02.example.com
   ```

```
dn: olcOverlay=syncprov,olcDatabase={0}config,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov

dn: olcDatabase={0}config,cn=config
changetype: modify
add: olcSyncRepl
olcSyncRepl: rid=001 provider=ldap://ldap01.example.com binddn="cn=admin,cn=config" bindmet
  credentials=secret searchbase="cn=config" type=refreshAndPersist
  retry="5 5 300 5" timeout=1
olcSyncRepl: rid=002 provider=ldap://ldap02.example.com binddn="cn=admin,cn=config" bindmet
  credentials=secret searchbase="cn=config" type=refreshAndPersist
  retry="5 5 300 5" timeout=1
-
add: olcMirrorMode
olcMirrorMode: TRUE
```

2. Edit the file changing:

   - *ldap://ldap01.example.com* and *ldap://ldap02.example.com* to the hostnames of your LDAP servers.

     

     You can have more than two LDAP servers, and when a change is made to one of them it will by synced to the rest. Be sure to increment the *olcServerID* for each server, and the *rid* for each *olcSyncRepl* entry.

   - And adjust *credentials=secret* to match your admin password.

3. Next, add the LDIF file using the **ldapmodify** utility:

```
ldapmodify -x -D cn=admin,cn=config -W -f syncrepl_cn-config.ldif
```

4. Copy the `syncrepl_cn-config.ldif` file to the next LDAP server and repeat the **ldapmodify** command above.

5. Because a new module has been added, the **slapd** daemon, on all replicated servers, needs to be restarted:

```
sudo /etc/init.d/slapd restart
```

6. Now that the configuration database is synced between servers, the *backend* database needs to be synced as well. Copy and paste the following into another LDIF file named `syncrepl_backend.ldif`:

```
dn: olcDatabase={1}hdb,cn=config
changetype: modify
add: olcRootDN
olcRootDN: cn=admin,dc=example,dc=edu
-
add: olcSyncRepl
olcSyncRepl: rid=003 provider=ldap://ldap01.example.com binddn="cn=admin,dc=example,dc=com"
 bindmethod=simple credentials=secret searchbase="dc=example,dc=com" type=refreshOnly
 interval=00:00:00:10 retry="5 5 300 5" timeout=1
olcSyncRepl: rid=004 provider=ldap://ldap02.example.com binddn="cn=admin,dc=example,dc=com"
 bindmethod=simple credentials=secret searchbase="dc=example,dc=com" type=refreshOnly
 interval=00:00:00:10 retry="5 5 300 5" timeout=1
-
add: olcMirrorMode
olcMirrorMode: TRUE

dn: olcOverlay=syncprov,olcDatabase={1}hdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov
```

7. Like the previous LDIF file, edit this one changing:

   - *searchbase="dc=example,dc=com"* to your directory's searchbase.
   - If you use a different admin user, change *binddn="cn=admin,dc=example,dc=com"*.
   - Also, replace *credentials=secret* with your admin password.

8. Add the LDIF file:

```
ldapmodify -x -D cn=admin,cn=config -W -f syncrepl_backend.ldif
```

Because the servers' configuration is already synced there is no need to copy this LDIF file to the other servers.

The configuration and backend databases should now sycnc to the other servers. You can add additional servers using the **ldapmodify** utility as the need arises. See the section called “Configuration” for details.

The **slapd** daemon will send log information to `/var/log/syslog` by default. So if all does *not* go well check there for errors and other troubleshooting information.

Setting up ACL

Authentication requires access to the password field, that should be not accessible by default. Also, in order for users to change their own password, using **passwd** or other utilities, *shadowLastChange* needs to be accessible once a user has authenticated.

To view the Access Control List (ACL), use the **ldapsearch** utility:

```
ldapsearch -xLLL -b cn=config -D cn=admin,cn=config -W olcDatabase=hdb olcAccess
```

```
Enter LDAP Password:
dn: olcDatabase={1}hdb,cn=config
olcAccess: {0}to attrs=userPassword,shadowLastChange by dn="cn=admin,dc=exampl
 e,dc=com" write by anonymous auth by self write by * none
olcAccess: {1}to dn.base="" by * read
olcAccess: {2}to * by dn="cn=admin,dc=example,dc=com" write by * read
```

TLS and SSL

When authenticating to an OpenLDAP server it is best to do so using an encrypted session. This can be accomplished using Transport Layer Security (TLS) and/or Secure Sockets Layer (SSL).

The first step in the process is to obtain or create a *certificate*. See the section called “Certificates” and the section called “Certification Authority” for details.

Once you have a certificate, key, and CA cert installed, use **ldapmodify** to add the new configuration options:

```
ldapmodify -x -D cn=admin,cn=config -W
```

```
Enter LDAP Password:
dn: cn=config
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/ssl/certs/cacert.pem
-
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ssl/certs/server.crt
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ssl/private/server.key

modifying entry "cn=config"
```

Adjust the `server.crt`, `server.key`, and `cacert.pem` names if yours are different.

Next, edit `/etc/default/slapd` uncomment the *SLAPD_SERVICES* option:

```
SLAPD_SERVICES="ldap://127.0.0.1:389/ ldaps:/// ldapi:///"
```

Now the *openldap* user needs access to the certificate:

```
sudo adduser openldap ssl-cert
sudo chgrp ssl-cert /etc/ssl/private/server.key
```

If the `/etc/ssl/private` and `/etc/ssl/private/server.key` have different permissions, adjust the commands appropriately.

Finally, restart **slapd**:

```
sudo /etc/init.d/slapd restart
```

The **slapd** daemon should now be listening for LDAPS connections and be able to use STARTTLS during authentication.

TLS Replication

If you have setup **Syncrepl** between servers, it is prudent to encrypt the replication traffic using *Transport Layer Security (TLS)*. For details on setting up replication see the section called “LDAP replication”.

After setting up replication, and following the instructions in the section called “TLS and SSL”, there are a couple of consequences that should be kept in mind:

- The configuration only needs to be modified on *one* server.
- The path names for the *certificate* and *key* must be the same on all servers.

So on each replicated server: install a certificate, edit `/etc/default/slapd`, and restart **slapd**.

Once *TLS* has been setup on each server, modify the *cn=config* replication by entering the following in a terminal:

```
ldapmodify -x -D cn=admin,cn=config -W
```

```
Enter LDAP Password:
dn: olcDatabase={0}config,cn=config
replace: olcSyncrepl
olcSyncrepl: {0}rid=001 provider=ldap://ldap01.example.com binddn="cn=admin,cn
 =config" bindmethod=simple credentials=secret searchbase="cn=config" type=refre
 shAndPersist retry="5 5 300 5" timeout=1 starttls=yes
olcSyncrepl: {1}rid=002 provider=ldap://ldap02.example.com binddn="cn=admin,cn
 =config" bindmethod=simple credentials=secret searchbase="cn=config" type=refre
 shAndPersist retry="5 5 300 5" timeout=1 starttls=yes

modifying entry "olcDatabase={0}config,cn=config"
```

Now adjust the *backend* database replication:

```
ldapmodify -x -D cn=admin,cn=config -W
```

```
Enter LDAP Password:
dn: olcDatabase={1}hdb,cn=config
replace: olcSyncrepl
olcSyncrepl: {0}rid=003 provider=ldap://ldap01.example.com binddn="cn=admin,dc=example,dc=
 com" bindmethod=simple credentials=secret searchbase="dc=example,dc=com" type=r
 efreshOnly interval=00:00:00:10 retry="5 5 300 5" timeout=1 starttls=yes
olcSyncrepl: {1}rid=004 provider=ldap://ldap02.example.com binddn="cn=admin,dc=example,dc=
 com" bindmethod=simple credentials=secret searchbase="dc=example,dc=com" type=r
 efreshOnly interval=00:00:00:10 retry="5 5 300 5" timeout=1 starttls=yes

modifying entry "olcDatabase={1}hdb,cn=config"
```

If the LDAP server hostname does not match the Fully Qualified Domain Name (FQDN) in the certificate, you may have to edit `/etc/ldap/ldap.conf` and add the following TLS options:

```
TLS_CERT /etc/ssl/certs/server.crt
TLS_KEY /etc/ssl/private/server.key
TLS_CACERT /etc/ssl/certs/cacert.pem
```

Finally, restart **slapd** on each of the servers:

```
sudo /etc/init.d/slapd restart
```

LDAP Authentication

Once you have a working LDAP server, the **auth-client-config** and **libnss-ldap** packages take the pain out of configuring an Ubuntu client to authenticate using LDAP. To install the packages from, a terminal prompt enter:

```
sudo apt-get install libnss-ldap
```

During the install a menu dialog will ask you connection details about your LDAP server.

If you make a mistake when entering your information you can execute the dialog again using:

```
sudo dpkg-reconfigure ldap-auth-config
```

The results of the dialog can be seen in `/etc/ldap.conf`. If your server requires options not covered in the menu edit this file accordingly.

Now that **libnss-ldap** is configured enable the **auth-client-config** LDAP profile by entering:

```
sudo auth-client-config -a -p lac_ldap
```

- *-a:* applies the specified profile.
- *-p:* name of the profile to enable, disable, etc.
- *lac_ldap:* the **auth-client-config** profile that is part of the **ldap-auth-config** package.

You should now be able to login using user credentials stored in the LDAP directory.

User and Group Management

The **ldap-utils** package comes with multiple utilities to manage the directory, but the long string of options needed, can make them a burden to use. The **ldapscripts** package contains configurable scripts to easily manage LDAP users and groups.

To install the package, from a terminal enter:

```
sudo apt-get install ldapscripts
```

Next, edit the config file `/etc/ldapscripts/ldapscripts.conf` uncommenting and changing the following to match your environment:

```
SERVER=localhost
BINDDN='cn=admin,dc=example,dc=com'
BINDPWDFILE="/etc/ldapscripts/ldapscripts.passwd"
SUFFIX='dc=example,dc=com'
GSUFFIX='ou=Groups'
USUFFIX='ou=People'
MSUFFIX='ou=Computers'
GIDSTART=10000
UIDSTART=10000
MIDSTART=10000
```

Now, create the `ldapscripts.passwd` file to allow authenticated access to the directory:

```
sudo sh -c "echo -n 'secret' > /etc/ldapscripts/ldapscripts.passwd"
sudo chmod 400 /etc/ldapscripts/ldapscripts.passwd
```

Replace “secret” with the actual password for your LDAP admin user.

The **ldapscripts** are now ready to help manage your directory. The following are some examples of how to use the scripts:

- Create a new user:

  ```
  sudo ldapadduser george example
  ```

  This will create a user with uid george and set the user's primary group (gid) to example

- Change a user's password:

```
sudo ldapsetpasswd george
Changing password for user uid=george,ou=People,dc=example,dc=com
New Password:
New Password (verify):
```

- Delete a user:

```
sudo ldapdeleteuser george
```

- Add a group:

```
sudo ldapaddgroup qa
```

- Delete a group:

```
sudo ldapdeletegroup qa
```

- Add a user to a group:

```
sudo ldapaddusertogroup george qa
```

You should now see a *memberUid* attribute for the qa group with a value of george.

- Remove a user from a group:

```
sudo ldapdeleteuserfromgroup george qa
```

The *memberUid* attribute should now be removed from the qa group.

- The **ldapmodifyuser** script allows you to add, remove, or replace a user's attributes. The script uses the same syntax as the **ldapmodify** utility. For example:

```
sudo ldapmodifyuser george
# About to modify the following entry :
dn: uid=george,ou=People,dc=example,dc=com
objectClass: account
objectClass: posixAccount
cn: george
uid: george
uidNumber: 1001
gidNumber: 1001
homeDirectory: /home/george
loginShell: /bin/bash
gecos: george
description: User account
userPassword:: e1NTSEF9eXFsTFcyWlhwWkF1eGUybVdFWHZKRzJVMjFTSG9vcHk=

# Enter your modifications here, end with CTRL-D.
dn: uid=george,ou=People,dc=example,dc=com
replace: gecos
gecos: George Carlin
```

The user's *gecos* should now be “George Carlin”.

- Another great feature of **ldapscripts**, is the template system. Templates allow you to customize the attributes of user, group, and machine objectes. For example, to enable the *user* template edit `/etc/ldapscripts/ldapscripts.conf` changing:

```
UTEMPLATE="/etc/ldapscripts/ldapadduser.template"
```

There are sample templates in the `/etc/ldapscripts` directory. Copy or rename the `ldapadduser.template.sample` file to `/etc/ldapscripts/ldapadduser.template`:

```
sudo cp /etc/ldapscripts/ldapadduser.template.sample /etc/ldapscripts/ldapadduser.template
```

Edit the new template to add the desired attributes. The following will create new user's as with an *objectClass* of *inetOrgPerson*:

```
dn: uid=<user>,<usuffix>,<suffix>
objectClass: inetOrgPerson
objectClass: posixAccount
cn: <user>
sn: <ask>
uid: <user>
uidNumber: <uid>
gidNumber: <gid>
homeDirectory: <home>
loginShell: <shell>
gecos: <user>
description: User account
title: Employee
```

Notice the <*ask*> option used for the *cn* value. Using <ask> will configure **ldapadduser** to prompt you for the attribute value during user creation.

There are more useful scripts in the package, to see a full list enter: **dpkg -L ldapscripts | grep bin**

Resources

- For more information see OpenLDAP Home Page
- Though starting to show it's age, a great source for in depth LDAP information is O'Reilly's LDAP System Administration
- Packt's Mastering OpenLDAP is a great reference covering newer versions of OpenLDAP.
- For more information on **auth-client-config** see the man page: **man auth-client-config**.
- For more details regarding the **ldapscripts** package see the man pages: **man ldapscripts**, **man ldapadduser**, **man ldapaddgroup**, etc.

Chapter 6. Network Authentication Samba and LDAP