A guide to setting up Postgres-backed OpenLDAP with the back-sql backend

OpenLDAP is typically backed by a LMDB database (mdb). However it's backing datastore is actually pluggable. One of the most intriguing backends is back-sql, which allows LDAP data to be fetched from an ODBC (SQL) datasource. This lets us add a LDAP facade on top of database-backed applications, like JIRA and Confluence.

This guide is specifically for getting OpenLDAP connecting to **PostgreSQL** on **Ubuntu** (20.04). Following these instructions should leave you with an OpenLDAP directory whose tree members are read directly from equivalent database tables.

- Other Guides

Recompile OpenLDAP with the SQL backend (back-sql)
Create a sample database
Connect with ODBC
Set up OpenLDAP
- Plain mdb-backed LDAP
- Add a SQL backend
  - Tell AppArmor to let slapd access odbc.ini

Testing your SQL-backed LDAP
Startup dependency?

Other Guides

The best guide I could find at time of writing (04 Jun 2020) is the OpenLDAP-POSTGRESQL HOWTO, written in 2001 and last updated in 2012. The HOWTO still fares well, given its age. It is weakest in the beginning, giving (I think) too many odbc options, outdated odbc advice, and too much compiling-from-source. Its strongest section is the last, a live example from a production system.

If you are using Ubuntu or Debian I suggest following this guide first, then refer back to the HOWTO's last section for a real example.

Recompile OpenLDAP with the SQL backend (back-sql)

First, get the OpenLDAP package source:

```
apt install dpkg-dev devscripts                    # Requirements for fetching and building source packages
vim /etc/apt/sources.list                          # Uncomment the deb-src line to enable fetching source
packages
apt update                                         # Refresh package cache
mkdir openldap
cd openldap
apt source openldap
cd openldap-2.4.49+dfsg
apt build-dep slapd                                # Get build-time dependencies of OpenLDAP
echo '--enable-sql' >> debian/configure.options    # Enable SQL backend, which is why we're recompiling
DEB_BUILD_OPTIONS='nostrip noopt parallel=4 nocheck' DFSG_NONFREE=true debuild --no-lintian -i -us -uc -b
```

The final build command needs a bit of explanation:

- **-us** is --unsigned-source – do not sign the source package
  -uc is --unsigned-changes
  -b is --build=binary
  -i is passed to dpkg-source
  --no-lintian prevents a slow 'lint' operation
- **DFSG_NONFREE=true** prevents the build breaking (can't remember why - check debian/rules)
- Within DEB_BUILD_OPTIONS:
  nostrip noopt leaves debug symbols in the resulting binaries. This is needed for later running slapd under 'gdb', which is something you're unfortunately likely to need.
  parallel=4 speed up the build a bit
  nocheck avoid running the tests on every build. For slapd they are *very* slow.

Now install the built packages:

```
apt install psmisc                            # This is a runtime dependency of slapd. Normally when you 'apt
install slapd' this dependency is fetched automatically; however since we're installing directly ('dpkg -i') we
need to install it manually.
dpkg -i ../*.deb                                # Install OpenLDAP packages we just built.
```

Ensure that slapd was built with 'sql' support:

```
$ slapd -VVV
@(#) $OpenLDAP: slapd  (Ubuntu) (Mar  6 2020 14:39:12) $
        Debian OpenLDAP Maintainers <pkg-openldap-devel@lists.alioth.debian.org>

Included static backends:
    config
    ldif
    sql
```

Create a sample database

Install Postgres and create a sample 'ldapsql' database:

```
apt install postgresql-12 postgresql-client-12
su - postgres                                 # Switch from root to 'postgres'
createuser -P ldapsql                         # Create 'ldapsql' postgres user. Enter 'ldapsql' for the
password.
createdb -O ldapsql ldapsql                   # Create 'ldapsql' database owned by 'ldapsql'
logout                                        # Switch from 'postgres' back to root
```

Load the back-sql sample database:

```
cd ~/openldap/openldap-2.4.49+dfsg/servers/slapd/back-sql/rdbms_depend/pgsql
export PGHOST=localhost PGUSER=ldapsql PGPASSWORD=ldapsql PGDATABASE=ldapsql
cat testdb_create.sql testdb_data.sql backsql_create.sql testdb_metadata.sql | psql
```

Verify that things look correct:

```
$ psql
ldapsql=> select * from ldap_entries;
 id |                  dn                   | oc_map_id | parent | keyval
----+---------------------------------------+-----------+--------+--------
  1 | dc=example,dc=com                     |         3 |      0 |      1
  2 | cn=Mitya Kovalev,dc=example,dc=com    |         1 |      1 |      1
  3 | cn=Torvlobnor Puzdoy,dc=example,dc=com |         1 |      1 |      2
  4 | cn=Akakiy Zinberstein,dc=example,dc=com |         1 |      1 |      3
  5 | documentTitle=book1,dc=example,dc=com |         2 |      1 |      1
  6 | documentTitle=book2,dc=example,dc=com |         2 |      1 |      2
  7 | ou=Referral,dc=example,dc=com         |         4 |      1 |      1
(7 rows)
ldapsql=> select * from persons;
 id |    name    |   surname   | password
----+------------+-------------+----------
  1 | Mitya      | Kovalev     | mit
  2 | Torvlobnor | Puzdoy      |
  3 | Akakiy     | Zinberstein |
(3 rows)
```

Connect with ODBC

```
apt install unixodbc odbc-postgresql
cat - <<EOF >> /etc/odbc.ini
[ldapsql]
Description         = Example for OpenLDAP's back-sql
Driver              = PostgreSQL ANSI
Trace               = No
Database            = ldapsql
Servername          = localhost
UserName            = ldapsql
Password            = ldapsql
Port                = 5432
;Protocol            = 6.4
ReadOnly            = No
RowVersioning       = No
ShowSystemTables    = No
ShowOidColumn       = No
FakeOidIndex        = No
ConnSettings        =
EOF
```

Connect with 'isql' to validate the odbc connection:

```
root@openldap2:/# isql -m10 ldapsql <<< 'select * from persons'
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL> select * from persons
+-----------+-----------+-----------+-----------+
| id        | name      | surname   | password  |
+-----------+-----------+-----------+-----------+
| 1         | Mitya     | Kovalev   | mit       |
| 2         | Torvlobnor| Puzdoy    |           |
| 3         | Akakiy    | Zinberstei|           |
+-----------+-----------+-----------+-----------+
SQLRowCount returns 3
3 rows fetched
```

Set up OpenLDAP

OpenLDAP has traditionally been configured in a slapd.conf(5) file. Since 2.3 OpenLDAP has adopted the newer slapd-config(5) format, where the configuration is itself managed as a directory tree, managed by LDAP.

The new format is horrible; instead of just editing a file, every change now needs to be translated to LDIF.

Fortunately you don't need to use it:

```
rm -r /etc/ldap/slapd.d/                # Begone, new format!
vim /etc/default/slapd                  # Set SLAPD_CONF=/etc/ldap/slapd.conf
systemctl restart slapd
```

Create a new slapd.conf from scratch.

Plain mdb-backed LDAP

First we'll get a normal mdb-backed LDAP working before adding a SQL backend:

```
cat - <<EOF > /etc/ldap/slapd.conf
include         /etc/ldap/schema/core.
schema
include         /etc/ldap/schema/cosine.
schema
include         /etc/ldap/schema/inetorgperson.
schema


pidfile         /var/run/slapd/slapd.pid
argsfile        /var/run/slapd/slapd.args

modulepath      /usr/lib/ldap
moduleload      back_mdb

database        mdb
suffix          "dc=test,dc=com"
rootdn          "cn=admin,dc=test,dc=com"
rootpw          secret
# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory       /var/lib/ldap
# Indices to maintain
index   objectClass     eq
EOF

systemctl restart slapd                                 # Restart; 'journalctl -fu slapd' if you have problems.

cat - <<EOF > /tmp/init.ldif
dn: dc=test,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: Test Organization
dc: Test

dn: cn=admin,dc=test,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword: $(slappasswd  -h {SSHA} -s hunter2)
EOF
ldapadd -f /tmp/init.ldif -x -D 'cn=admin,dc=test,dc=com' -w secret
```

We should now be able to bind as cn=admin using either 'secret' or 'hunter2':

```
# ldapsearch -x -b 'dc=test,dc=com' -D 'cn=admin,dc=test,dc=com' -w hunter2
# extended LDIF
#
# LDAPv3
# base <dc=test,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# test.com
dn: dc=test,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: Test Organization
dc: Test

# admin, test.com
dn: cn=admin,dc=test,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword:: e1NTSEF9czRHRHZ4TlNTMWk1eTI0K2cyd3pnTVpFclY4TGpzN2s=

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

Add a SQL backend

Now let's add a SQL backend. OpenLDAP can have multiple backends - we'll leave our `dc=test,dc=com` backend configured, and add a new SQL backend rooted at **dc=example,dc=com** :

```
sed -e 's/^ //g' - <<EOF >> /etc/ldap/slapd.conf

# SQL Backend
database        sql
suffix          "dc=example,dc=com"
rootdn          "cn=admin,dc=example,dc=com"
rootpw          secret
dbname          ldapsql
dbuser          ldapsql
dbpasswd        ldapsql
insentry_stmt   "insert into ldap_entries (id,dn,oc_map_id,parent,keyval) values ((select max(id)+1 from
ldap_entries),?,?,?,?)"
upper_func      "upper"
strcast_func    "text"
concat_pattern  "?||?"
#subtree_cond    "ldap_entries.dn LIKE CONCAT('%',?)"
has_ldapinfo_dn_ru      no
EOF
systemctl restart slapd                     # If unsuccessful, 'journalctl -fu slapd &' and try again
```

⚠ The `sed` command is to trip the blank space ' ' that Confluence is adding when this block is copied and pasted :/

Tell AppArmor to let slapd access odbc.ini

Initially when I added the SQL backend, my slapd refused to start with an unhelpful error:

```
backend_startup_one (type=sql, suffix="dc=jira"): bi_db_open failed! (1)
```

More verbose logs can be obtained by editing `/etc/default/slapd` and setting SLAPD_OPTIONS="-s7". That yielded:

```
 ==>backsql_get_db_conn()
 ==>backsql_open_db_handle()
 backsql_open_db_handle(): SQLConnect() to database "ldapsql" failed.
 Return code: -1
    nativeErrCode=0 SQLengineState=IM002[unixODBC][Driver Manager]Data source name not found, and no default
driver specified msg="[unixODBC][Driver Manager]Data source name not found, and no default driver specified"
 backsql_db_open(): connection failed, exiting
 backend_startup_one (type=sql, suffix="dc=example,dc=com"): bi_db_open failed! (1)
 slapd shutdown: initiated
 ==>backsql_db_close()
 <==backsql_db_close()
 slapd destroy: freeing system resources.
 ==>backsql_db_destroy()
 ==>backsql_free_db_env()
 <==backsql_free_db_env()
 ==>destroy_schema_map()
 <==destroy_schema_map()
 <==backsql_db_destroy()
```

I eventually stumbled upon this post, which identified the problem: Ubuntu's AppArmor is preventing slapd from accessing `/etc/odbc.ini`

The fix is:

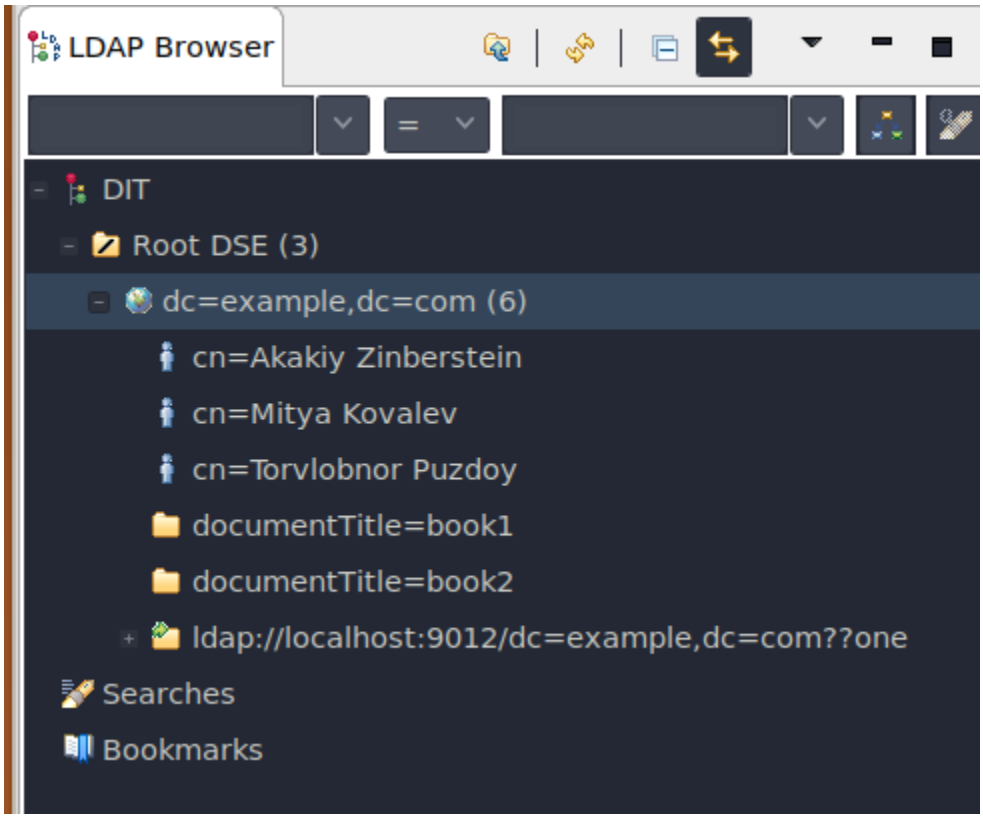
```
cat - <<EOF > /etc/apparmor.d/local/usr.sbin.slapd
# Let slapd access odbc config files, and (for MySQL) the mysql unix socket.
# https://ubuntuforums.org/showthread.php?p=8248430#post8248430
/etc/odbc.ini r,
/etc/odbcinst.ini r,
/var/run/mysqld/mysqld.sock w,
EOF
systemctl restart apparmor
systemctl restart slapd
```

Testing your SQL-backed LDAP

If everything went correctly, you should now be able to query your database-backed directory tree:

```
root@openldap2:/# ldapsearch -x -b 'dc=example,dc=com' '(objectclass=inetOrgPerson)' objectclass cn sn  -D
'cn=admin,dc=example,dc=com' -w secret
# extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> with scope subtree
# filter: (objectclass=inetOrgPerson)
# requesting: objectclass cn sn
#

# Akakiy Zinberstein, example.com
dn: cn=Akakiy Zinberstein,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: pkiUser
cn: Akakiy Zinberstein
sn: Zinberstein

# Mitya Kovalev, example.com
dn: cn=Mitya Kovalev,dc=example,dc=com
objectClass: inetOrgPerson
cn: Mitya Kovalev
sn: Kovalev

# Torvlobnor Puzdoy, example.com
dn: cn=Torvlobnor Puzdoy,dc=example,dc=com
objectClass: inetOrgPerson
cn: Torvlobnor Puzdoy
sn: Puzdoy

# search result
search: 2
result: 0 Success

# numResponses: 4
# numEntries: 3
```

At this point you might like to download a LDAP browser like Apache Directory Studio. Point it at `localhost` with your `dc=example,dc=com` base DN:



Startup dependency?

We have now introduced a startup dependency between Postgres and OpenLDAP. If OpenLDAP happens to start before Postgres it up it will fail. I know we need:

```
After=postgresql@12-main.service
Wants=postgresql@12-main.service
```

but I'm not sure how to add this, given slapd is still started as a sysvinit startup file.