**Chapter 1 :- LDAP Server**

**Introduction of LDAP**

* LDAP stands for Lightweight Directory Access Protocol. It is an Internet protocol for accessing distributed directory services. Using LDAP organizations can organize their users and resources into a hierarchical directory (or tree) structure. Benefits of this organization is that it can act as a centralized source of information for the organization there by providing following services
* Organize contact information of users into directories. Think of this as an address book.
* Resources like printers, scanners, meeting rooms , etc can also be organized in LDAP.
* LDAP supports ACLs i.e. you can control what can a user access in the directory.
* LDAP can store login credentials of users, there by acting as a company wide authentication and authorization database. And as such a lot of organizations use LDAP with their Single Sign On (SSO) integration.

**LDAP schema**

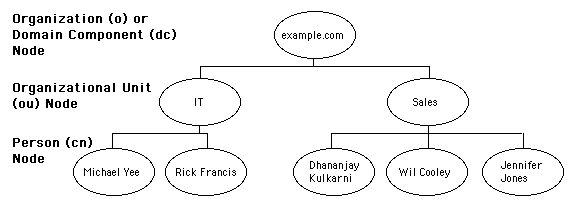
A relational database schema contains information about the structure of the database, including information about the tables, about the columns of each of those tables, and the data types and constraints of each of those columns. In LDAP, the schema provides much the same kind of information, but because information is arranged very differently in a directory server than a relational database, the expression of those schema elements is substantially different.

There are a number of different types of elements that may comprise an LDAP schema. Every LDAP schema must include the following elements:

* Attribute syntaxes define the types of data that can be represented in a directory server.
* Matching rules define the kinds of comparisons that can be performed against LDAP data.
* Attribute types define named units of information that may be stored in entries.
* Object classes define named collections of attribute types which may be used in entries containing that class, and which of those attribute types will be required rather than optional.

LDAP schemas may also include additional elements that can impose further restrictions on the structure of the data in the server. These include:

* Name forms may be used to restrict the kinds of attributes which may be used as the naming attributes for entries of a particular type.
* DIT content rules may be used to augment object class definitions and further indicate the kinds of attributes that must, may, and must not appear in entries of a particular type.
* DIT structure rules may be used to define information about hierarchical relationships that are allowed to exist in the server.
* Matching rule uses may be used to impose restrictions on the kinds of attributes with which particular matching rules may be used.



**Features of LDAP**

1. **Increases security :** The current LDAP version supports SASL, which is an important internet standard responsible for allowing clients to select the specific authentication protocols they wish to utilize. In addition, LDAP also supports TLS, which encrypts the entire data passing between the server and client. This protects the information from any security breach.
2. **Lightweight :** Many programmers consider LDAP as lightweight in comparison to X.500 system that was mainly built on OSI layers. The reason is that LDAP carries fewer operations than the X.500. Even though LDAP is largely based on the X.500, it is more simplified and adjusted to custom requirements. This benefits the programmers as they can easily implement and also connect to numerous services.
3. **One directory** : An essential LDAP benefit is that it enables users to make use of one information directory containing user data. It also offers an access to that directory from various services. For instance, an information directory can be utilized for [email](http://benefitof.net/benefits-of-email/)and web browsers among other services. Furthermore, time savings achieved from offering one central directory for various services is also a huge benefit.
4. **Simple to use** : LDAP is very simple to use and users can add extra information to the directory as required by the organization. This information includes things like phone numbers, [office](http://benefitof.net/benefits-of-open-office/)location, addresses and other vital data.

**Configuring LDAP Server Creating of LDAP Domain Database**

**Install LDAP**

The OpenLDAP server is in Ubuntu's default repositories under the package "slapd", so we can install it easily with apt-get. We will also install some additional utilities:

# sudo apt-get update

# sudo apt-get install slapd ldap-utils

You will be asked to enter and confirm an administrator password for the administrator LDAP account.

**Reconfigure slapd**

When the installation is complete, we actually need to reconfigure the LDAP package. Type the following to bring up the package configuration tool:

# sudo dpkg-reconfigure slapd

You will be asked a series of questions about how you'd like to configure the software.

* Omit OpenLDAP server configuration? No
* DNS domain name?
* This will create the base structure of your directory path. Read the message to understand how it works.
* There are no set rules for how to configure this. If you have an actual domain name on this server, you can use that. Otherwise, use whatever you'd like.
* In this article, we will call it test.com
* Organization name?
* Again, this is up to you
* We will use example in this guide.
* Administrator password?
* Use the password you configured during installation, or choose another one
* Database backend to use? HDB
* Remove the database when slapd is purged? No
* Move old database? Yes
* Allow LDAPv2 protocol? No

**Install PHPldapadmin**

We will be administering LDAP through a web interface called PHPldapadmin. This is also available in Ubuntu's default repositories.

Install it with this command:

# sudo apt-get install phpldapadmin

That will install all of the required web server and PHP dependencies.

**Configure PHPldapadmin**

We need to configure some values within the web interface configuration files before trying it out.

Open the configuration file with root privileges:

# sudo nano /etc/phpldapadmin/config.php

Search for the following sections and modify them accordingly.

Change the red value to the way you will be referencing your server, either through domain name or IP address.

$servers->setValue('server','host','domain\_nam\_or\_IP\_address');

For the next part, you will need to reflect the same value you gave when asked for the DNS domain name when we reconfigured "slapd".

You will have to convert it into a format that LDAP understands by separating each domain component. Domain components are anything that is separated by a dot.

These components are then given as values to the "dc" attribute.

For instance, if your DNS domain name entry was "imaginary.lalala.com", LDAP would need to see "dc=imaginary,dc=lalala,dc=com". Edit the following entry to reflect the name you selected (ours is "test.com" as you recall):

$servers->setValue('server','base',array('dc=test,dc=com'));

The next value to modify will use the same domain components that you just set up in the last entry. Add these after the "cn=admin" in the entry below:

$servers->setValue('login','bind\_id','cn=admin,dc=test,dc=com');

Search for the following section about the "hidetemplatewarning" attribute. We want to uncomment this line and set the value to "true" to avoid some annoying warnings that are unimportant.

$config->custom->appearance['hide\_template\_warning'] = true;

Save and close the file.

**Log Into the Web Interface**

You can access by going to your domain name or IP address followed by "/phpldapadmin" in your web browser:



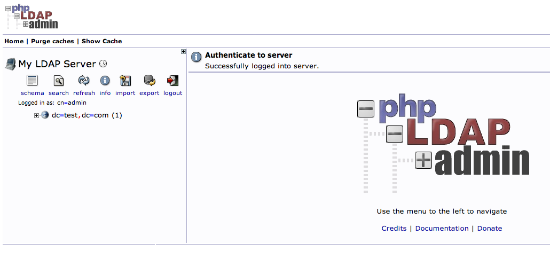
Click on the "login" link on the left-hand side.

You will receive a login prompt. The correct Login DN (distinguished name) should be pre-populated if you configured PHPldapadmin correctly. In our case, this would be "cn=admin,dc=test,dc=com".

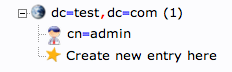


Enter the password you selected during our slapd configuration.

You will be presented with a rather sparse interface initially.



If you click on the "plus" next to the domain components (dc=test,dc=com), you will see the admin login we are using.



**Add Organizational Units, Groups, and Users**

LDAP is very flexible. You can create hierarchies and relationships in many different ways, depending on what kind of information you need accessible and what kind of use case you have.

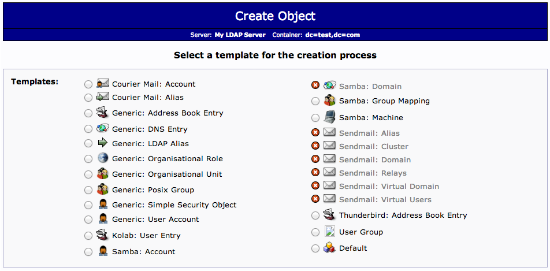
We will create some basic structure to our information and then populate it with information.

**Create Organizational Units**

First, we will create some categories of information where we will place the later information. Because this is a basic setup, we will only need two categories: groups and users.

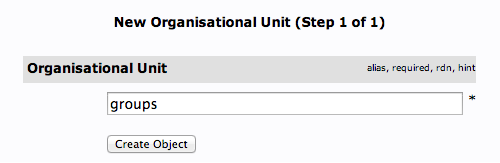
Click on the "Create new entry here" link on the left-hand side.

Here, we can see the different kinds of entries we can create.

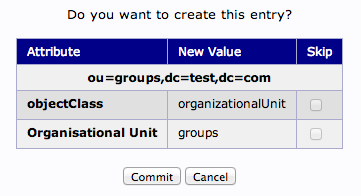


Because we are only using this as an organizational structure, rather than an information-heavy entry, we will use the "Generic: Organizational Unit" template.

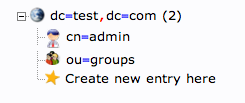
We will be asked to create a name for our organizational unit. Type "groups":



We will then need to commit the changes.



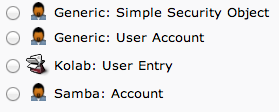
When this is complete, we can see a new entry on the left-hand side.



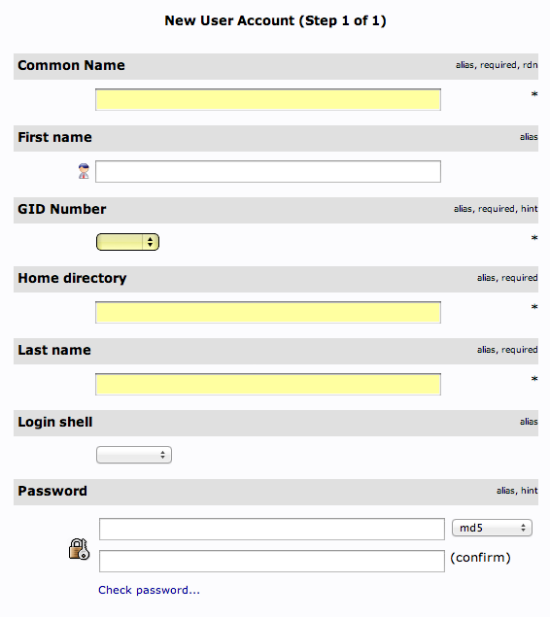
### Create Users

Next, we will create users to put in these groups. Start by clicking the "ou=users" category. Click on "Create a child entry".

We will choose "Generic: User Account" for these entries.



We will be given a lot of fields to fill out:



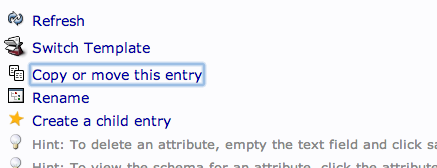
Fill in all of the entries with information that makes sense for your user.

Something to keep in mind is that the "Common Name" needs to be unique for each entry in a category. So you may want to use a username format instead of the default "FirstName LastName" that is auto-populated.

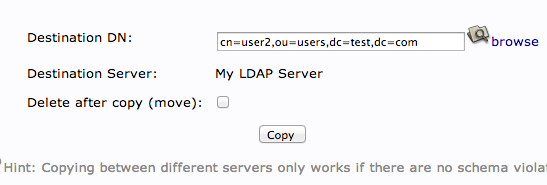
Click "Create Object" at the bottom and confirm on the following page.

To create additional users, we will take advantage of the ability to copy entries.

Click on the user you just created in the left-hand panel. In the main pane, click "Copy or move this entry":



Adjust the "cn=user" portion of the entry to point it to the common name you'd like to use for the new entry. Click "Copy" at the bottom:

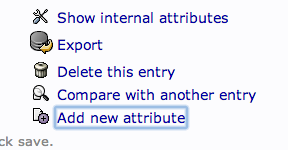


You will be given the next page populated with your first users data. You will need to adjust it to match the new users information.

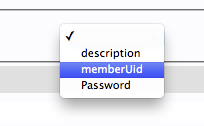
Be sure to adjust the uidNumber. Click the "Create Object" button at the bottom.

**Add Users to Groups**

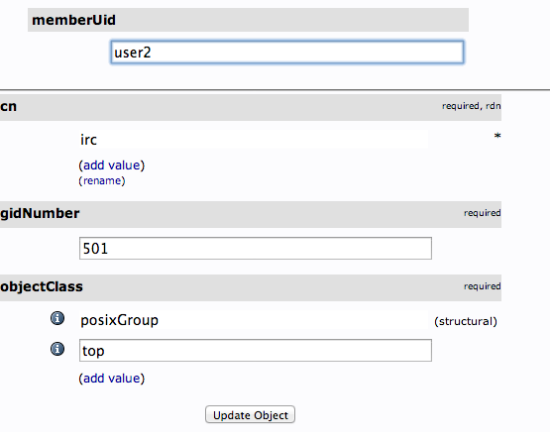
We can add users to various groups by clicking on the group in question. In the main pane, select "Add new attribute":



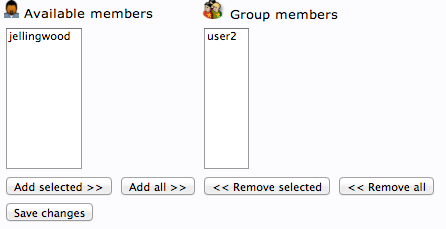
Select "memberUid" from the drop down menu:



In the text field that populates, enter the first user you'd like to add. Click "Update Object" at the bottom:



You can then add more members by clicking "modify group members" and selecting them from the available choices:



**Modifying and Deleting Users from LDAP Database**

**Deleting Entries from the DIT**

We had our first glimpse of the changetype option in the last section. This option provides the method for specifying the high-level type of modification we wish to make. For an entry deletion, the value of this option is "delete".

Entry deletion is actually the most straight-forward change that you can perform because the only piece of information needed is the DN.

For instance, if we wanted to remove the ou=othergroup entry from our DIT, our LDIF file would only need to contain this:

**dn: ou=othergroup,dc=example,dc=com**

**changetype: delete**

To process the change, you can use the exact format used with ldapmodify above. If we call the file with the deletion request rmothergroup.ldif, we would apply it like this:

$ ldapmodify -x -D "cn=admin,dc=example,dc=com" -w password -H ldap:// -f rmothergroup.ldif

This will remove the ou=othergroup entry from the system immediately.

**Modifying an Entry's Attributes**

Modifying an entry's attributes is a very common change to make and is made possible by specifying changetype: modify after the DN of the entry. The types of modifications you can make to attributes mostly mirror the modifications you can make to an entry itself. Because of this, the details of the type of requested attribute change are specified afterwards using additional directives.

**Adding an Attribute to an Entry**

For instance, you can add an attribute by using the add: command after changetype: modify. This should specify the attribute you wish to add. You would then set the value of the attribute like normal. So the basic format would be:

dn: entry\_to\_add\_attribute

changetype: modify

add: attribute\_type

attribute\_type: value\_to\_set

For instance, to add some email addresses to our accounts, we could have an LDIF file that looks like this:

dn : uid=sbrown20,ou=People,dc=example,dc=com

changetype: modify

add: mail

mail: sbrown@example.com

dn: uid=jsmith1,ou=People,dc=example,dc=com

changetype: modify

add: mail

mail: jsmith1@example.com

mail: [johnsmith@example.com](mailto:johnsmith@example.com)

As you can see from the second entry, you can specify multiple additions at the same time. The mail attribute allows for multiple values, so this is permissible.

You can process this with ldapmodify as normal. If the change is in the file sbrownaddmail.ldif, you could type:

$ ldapmodify -x -D "cn=admin,dc=example,dc=com" -w password -H ldap:// -f sbrownaddmail.ldif

**Replacing the Value of an Attribute in an Entry**

Another common change is to modify the existing value for an attribute. We can do this using the replace: option below changetype: modify.

This operates in almost the same way as the add: command, but by default, removes every existing occurrence of the attribute from the entry and replaces it with the values defined afterwards. For instance, if we notice that our last add: command had an incorrect email, we could modify it with the replace command like this:

dn: uid=sbrown20,ou=People,dc=example,dc=com

changetype: modify

replace: mail

mail: sbrown2@example.com

Keep in mind that this will replace every instance of mail in the entry. This is important for multi-value attributes that can be defined more than once per-entry (like mail). If you wish to replace only a single occurrence of an attribute, you should use the attribute delete: option (described below) in combination with the attribute add: option (described above)

If this change was stored in a file called sbrownchangemail.ldif, we can replace Sally's email by typing:

$ ldapmodify -x -D "cn=admin,dc=example,dc=com" -w password -H ldap:// -f sbrownchangemail.ldif

### Delete Attributes from an Entry

If you wish to remove an attribute from an entry, you can use the delete:command. You will specify the attribute you wish to delete as the value of the option. If you want to delete a specific instance of the attribute, you can specify the specific key-value attribute occurrence on the following line. Otherwise, every occurrence of that attribute in the entry will be removed.

For instance, this would delete every description attribute in John Smith's entry:

dn: uid=jsmith1,ou=People,dc=example,dc=com

changetype: modify

delete: description

However, this would delete only the email specified:

dn: uid=jsmith1,ou=People,dc=example,dc=com

changetype: modify

delete: mail

mail: jsmith1@example.com