**LDAP/TLS CONFIGURATION ON LINUX**

**192.168.0.180 ldapserver.hopto.org**

**192.168.0.181 ldapepas.hopto.org**

**192.168.0.182 ldapclient.hopto.org**

**yum -y install openldap-clients openldap-servers**

**systemctl start slapd**

**systemctl enable slapd**

**systemctl status slapd**

Once installed, we have to generate a password for the admin user. In this example, we use a simple password: “redhat”

**Slappasswd {SSHA}2qOMO8B2k5J3TtqbsrKLk+JOF/HwTPbu**

**New password:**

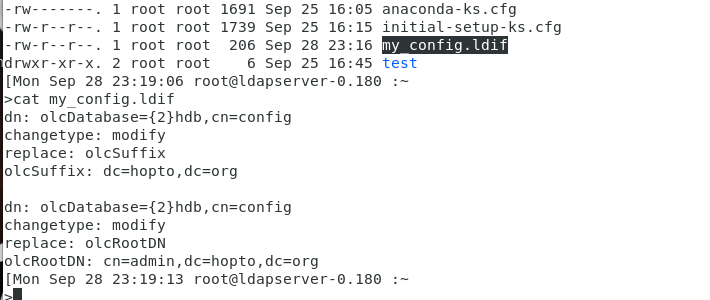
**Re-enter new password:**

**{SSHA}C5iN0R6Ra/CNyE+M0xNlYA9EPsgSu6HJ**

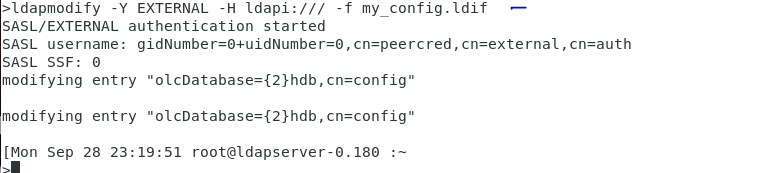
Another, and maybe better, way to identify the data we require to create the LDIF file could be to use the ldapsearchcommand.

**ldapsearch -Y EXTERNAL -H ldapi:/// -b cn=config olcDatabase=\\***

We save the LDIF file with an appropriate name, for example, my\_config.ldif, and we execute ldapmodify.



**ldapmodify -Y EXTERNAL -H ldapi:/// -f my\_config.ldif**



**We could now see that the file is changed**



Add olcRootPW attribute

To add a new attribute we use "add" and then the attribute name as shown in the below example.

Here we create another LDIF file (my\_config2.ldif) to add the olcRootPW attribute.

>cat my\_config2.ldif

dn: olcDatabase={2}hdb,cn=config

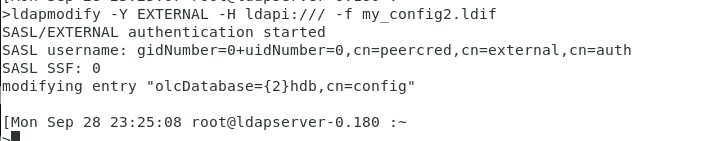
changeType: modify

add: olcRootPW

olcRootPW: {SSHA}C5iN0R6Ra/CNyE+M0xNlYA9EPsgSu6HJ

we the run

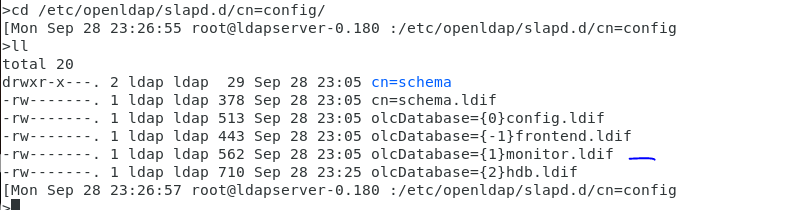
ldapmodify -Y EXTERNAL -H ldapi:/// -f my\_config2.ldif



### Replace olcAccess attribute

We also have to allow access to the LDAP database to the admin user we just specified before (cn=admin,dc=example,dc=com). If we take a look at the olcDatabase={1}monitor.ldif, file we’ll see the following line:

cd /etc/openldap/slapd.d/cn=config/



We’ll have to edit the file or use ldapmodify to change the entry. If we use ldapmodify, the LDIF file should be something like this:

cat my\_config3.ldif

dn: olcDatabase={1}monitor,cn=config

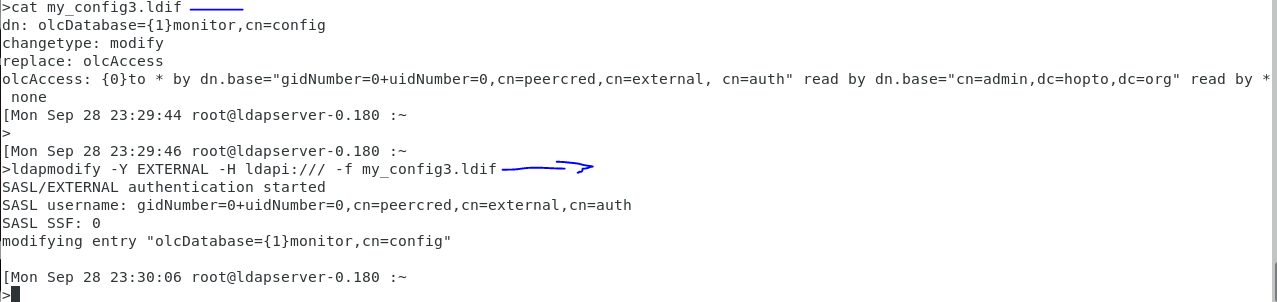
changetype: modify

replace: olcAccess

olcAccess: {0}to \* by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external, cn=auth" read by dn.base="cn=admin,dc=hopto,dc=org" read by \* none

Once again, we execute ldapmodify by passing the new LDIF file as a parameter.

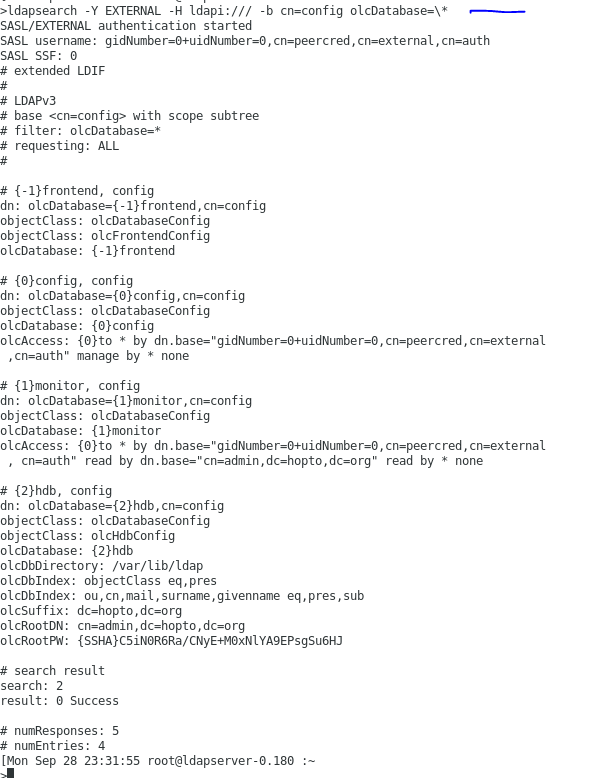
ldapmodify -Y EXTERNAL -H ldapi:/// -f my\_config3.ldif



### Validate the new attribute values

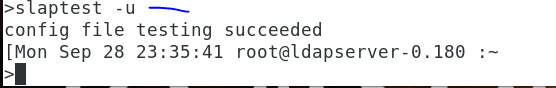
Now we can check with ldapsearch whether the value for the attribute was actually changed.

ldapsearch -Y EXTERNAL -H ldapi:/// -b cn=config olcDatabase=\\* we see all the modification



As we can see, the value was changed according to what we specified in the LDIF file.

Another tool we can use to check the configuration is the slaptest command.



## Adding Objects

Now we have to manually create an entry for dc=example,dc=com in our LDAP server. The easiest way to do this is to create an LDIF file for this entry and pass it to the ldapadd command.

So, we create a file named example.ldif, with the following content:

>cat example.ldif

dn: dc=hopto,dc=org

objectClass: dcObject

objectClass: organization

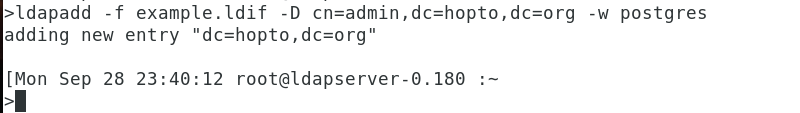
dc: hopto

o: hopto

We specify a series of attributes, such as distinguished name (dn), domain component (dc), and organization (o). We also define the new entry as an object of the type dcObject and organization.

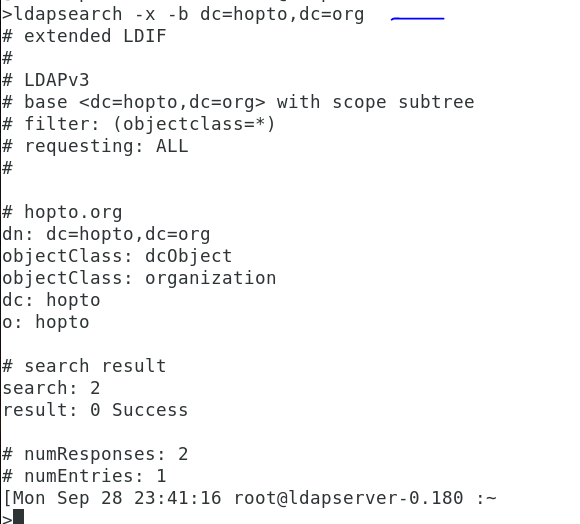
Now we execute ldapadd and pass it the example.ldif file as a parameter. We specify with (-f) the name of the file, the admin user (-D), and the password we defined for that admin user (-w).

ldapadd -f example.ldif -D cn=admin,dc=hopto,dc=org -w postgres



We can check whether the entry was created successfully by using the ldapsearch command.

ldapsearch -x -b dc=hopto,dc=org



You just saw how to add the object dc=example,dc=com to our LDAP. Now you’ll see how to add organizational units , groups, and users.

### Adding an Organizational Unit

Maybe we’d like to have an organizational unit (OU) called users in which to store all LDAP users. To do so, we’ll create a new LDIF file named users.ldif, with the following content:

>cat users.ldif

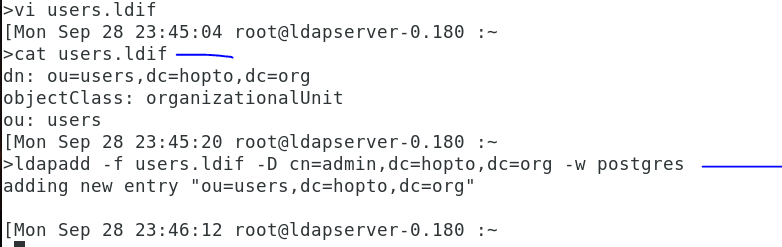
dn: ou=users,dc=hopto,dc=org

objectClass: organizationalUnit

ou: users

We execute ldapadd again to create the OU.

ldapadd -f users.ldif -D cn=admin,dc=hopto,dc=org -w postgres



### Adding a User

We can now include a user inside the organizational unit. The procedure is quite similar to what we have seen so far. First, we create a file named benson.ldif, with the following content:

>cat benson.ldif

dn: cn=benson of edb,ou=users,dc=hopto,dc=org

cn: benson

sn: edb

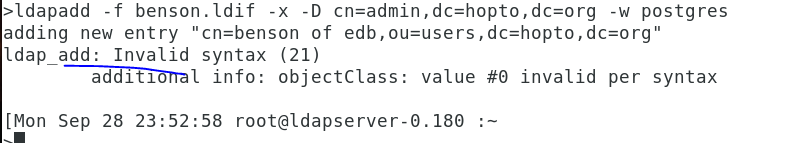
objectClass: inetOrgPerson

userPassword: postgres

uid: benson

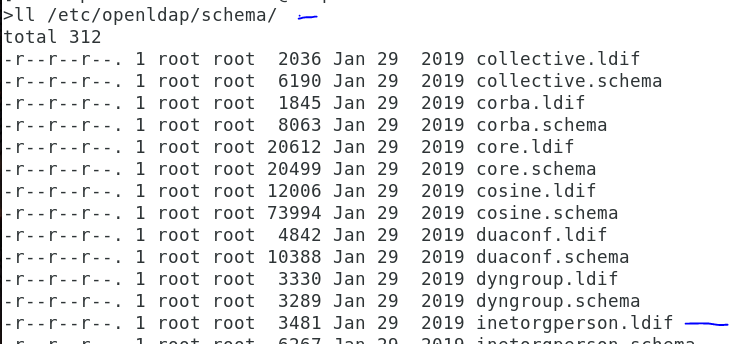
Then we execute ldapadd again.

ldapadd -f benson.ldif -x -D cn=admin,dc=hopto,dc=org -w postgres

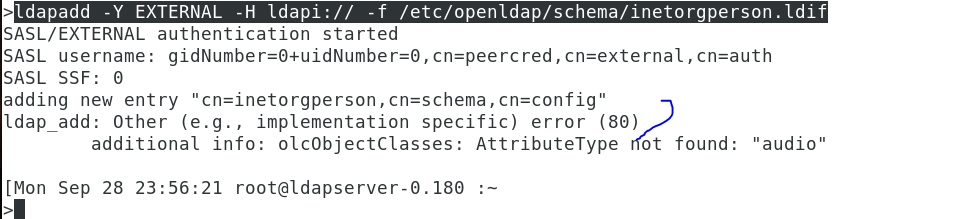


What this message means is that the object inetOrgPerson isn’t loaded in the core schema, so we’ll have to include it. In the /etc/openldap/schema folders, there are many LDIF files to extend the schema when we need it. We can see there is an inetorgperson.ldif file, which contains the schema definition for the inetOrgPerson object.

The schema itself is contained in the LDAP database, so we can add new definitions to it with the ldapadd command. As we’re going to modify the configuration itself, instead of the data, we’ll authenticate ourselves as the external root user (-Y EXTERNAL).



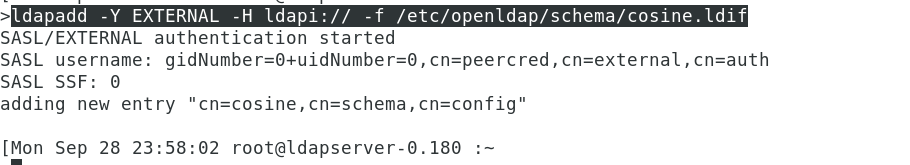
ldapadd -Y EXTERNAL -H ldapi:// -f /etc/openldap/schema/inetorgperson.ldif



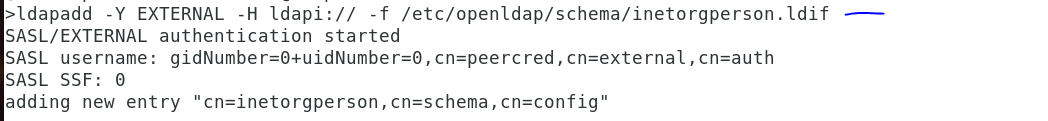
As we can see, we get an error, because the attribute type audio isn’t defined. So, we have to include this definition in the schema too .

If we perform a search of the string audio in the files located in the /etc/openldap/schema/ folder, we’ll see that the attribute audio is defined in the cosine.ldif file. So, we extend the schema with this LDIF file first.

ldapadd -Y EXTERNAL -H ldapi:// -f /etc/openldap/schema/cosine.ldif

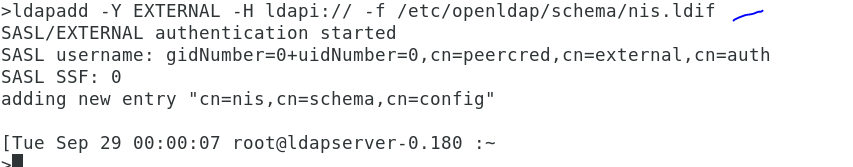


ldapadd -Y EXTERNAL -H ldapi:// -f /etc/openldap/schema/inetorgperson.ldif

\

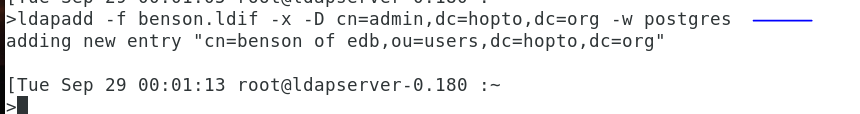
Next we also add nis.ldif file

ldapadd -Y EXTERNAL -H ldapi:// -f /etc/openldap/schema/nis.ldif



Now we can add the user with the benson.ldif file we created before.

ldapadd -f benson.ldif -x -D cn=admin,dc=hopto,dc=org -w postgres



If at some point we have to take a look at the currently used schema, we can use the slapcat command like this:

slapcat -b "cn=schema,cn=config" the output of this is large but look at the beginning and at the end.

### Adding a Group

To add a group, we repeat the same process . First we create the group.ldif file with the following content:

>cat group.ldif

dn: cn=edbsupport,ou=users,dc=hopto,dc=org

cn: edbsupport

objectClass: groupOfNames

member: cn=benson of edb,ou=users,dc=hopto,dc=org

And we add the group with ldapadd.

ldapadd -f group.ldif -x -D cn=admin,dc=hopto,dc=org -w postgres



**Now since our ldap server is configured, next we will configure openldap with TLS certificates**

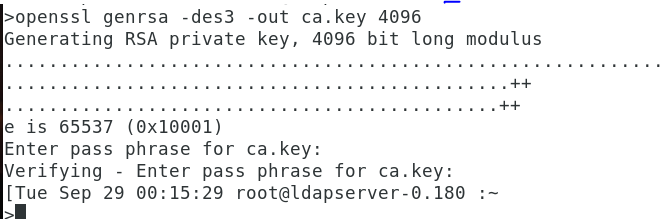
To configure OpenLDAP with TLS certificates we need openssl package. This will give us a directory hierarchy for creating the certificates to configure OpenLDAP with TLS certificates

yum -y install openssl

## Create private key for CA certificate

First we need a private key to generate our own CA certificate. If you do not wish to have an encrypted private key then you can ignore -des3 in the below command:

openssl genrsa -des3 -out ca.key 4096



## Generate CA Certificate

Next we will use our CA key to generate CA certificate. We will use this CA ceriticate later to sign the ldap client certificates

openssl req -new -x509 -days 365 -key ca.key -out ca.cert.pem



It is important that you provide a proper CN or else you will face problems with [TLS handshake later as I have demonstrated in earlier articles](https://www.golinuxcloud.com/openssl-create-client-server-certificate/#Verify_TCP_Handshake_using_Client_Server_Certificates). Here I am using the CN of my ldapserver.hopto.org.

## Configure openssl x509 extension to create SAN certificate

We will create SAN certificate to avoid creating multiple certificates for each of our ldap client. You can learn more about SAN certificates at [Create san certificate](https://www.golinuxcloud.com/openssl-generate-csr-create-san-certificate/)

**>cat server\_cert\_ext.cnf**

basicConstraints = CA:FALSE

nsCertType = server

nsComment = "OpenSSL Generated Server Certificate"

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid,issuer:always

keyUsage = critical, digitalSignature, keyEncipherment

extendedKeyUsage = serverAuth

subjectAltName = @alt\_names

[alt\_names]

IP.1 = 192.168.0.180

IP.2 = 192.168.0.182

IP.3 = 192.168.0.181

IP.3 = 192.168.0.183

DNS.1 = ldapclient.hopto.org

DNS.2 = ldapepas.hopto.org

DNS.3 = ldapserver.hopto.org

Under [alt\_names], I will provide the complete list of IP Address and DNS name which the ldap client certificate should resolve when validating a client request.

## Generate LDAP client key

Next we will need a ldap client key which we will name as "ldap.hopto.org.key"

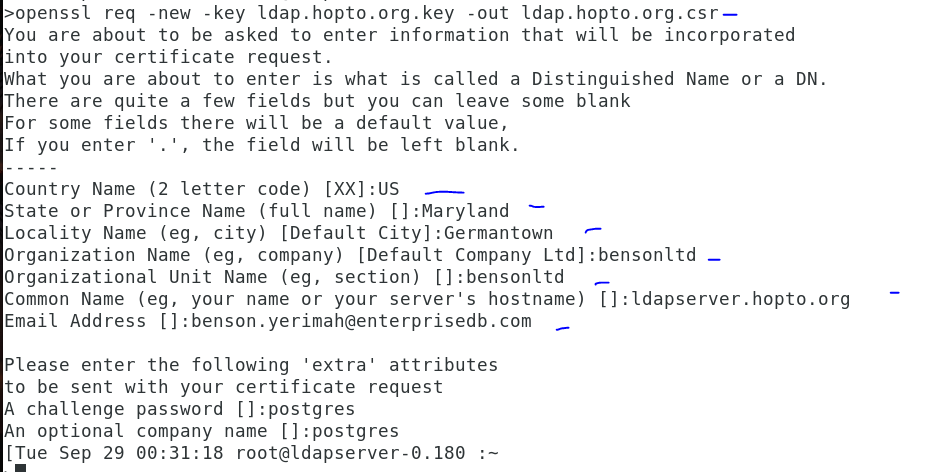
openssl genrsa -out ldap.hopto.org.key 4096



## Create Certificate Signing request (CSR)

Next we need a CSR to sign our ldap client certificate. Here we will use our server configuration file to provide the list of IP and DNS. These values will be used as CN so all the clients with these IP or DNS will be allowed to establish TLS communication with the LDAP server

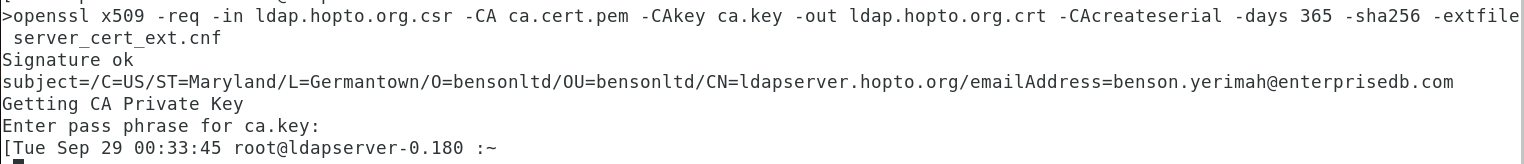
openssl req -new -key ldap.hopto.org.key -out ldap.hopto.org.csr



## Create LDAP client certificate

Next we will create our ldap client certificate (ldap.hopto.org.crt) using the CSR, CA key and CA certificate we created earlier. This certificate will be valid for 365 days and is encrypted with sha256 algorithm.

openssl x509 -req -in ldap.hopto.org.csr -CA ca.cert.pem -CAkey ca.key -out ldap.hopto.org.crt -CAcreateserial -days 365 -sha256 -extfile server\_cert\_ext.cnf



## Verify the ldap client certificate

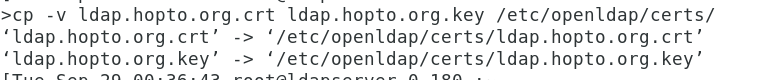
Next check the content of your ldap client certificate to make sure it contains the list of IP and DNS which we provided earlier.

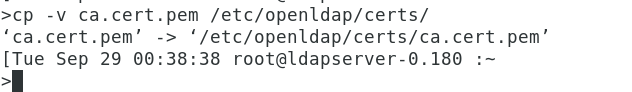
openssl x509 -noout -text -in ldap.hopto.org.crt ---large ouput

After signing the certificate, we copy both the certificate and the key file to /etc/openldap/certs/. We also copy the CA certificate to /etc/openldap/cacerts/. Later, we’ll have to modify the openldap configuration accordingly.

cp -v ldap.hopto.org.crt ldap.hopto.org.key /etc/openldap/certs/

cp -v ca.cert.pem /etc/openldap/certs/

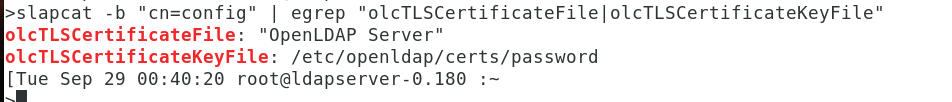




## Securing the LDAP protocol

In CentOS 7, there are already default values for the TLS related attributes. We can see these values with slapcat.

slapcat -b "cn=config" | egrep "olcTLSCertificateFile|olcTLSCertificateKeyFile"



We have to modify the values of the olcTLSCertificateFile and olcTLSCertificateKeyFile attributes. So, we create the following LDIF file:

>vi tls7.ldif

[Tue Sep 29 00:43:00 root@ldapserver-0.180 :~

>cat tls7.ldif

dn: cn=config

changetype: modify

replace: olcTLSCertificateFile

olcTLSCertificateFile: /etc/openldap/certs/ldap.hopto.org.crt

-

replace: olcTLSCertificateKeyFile

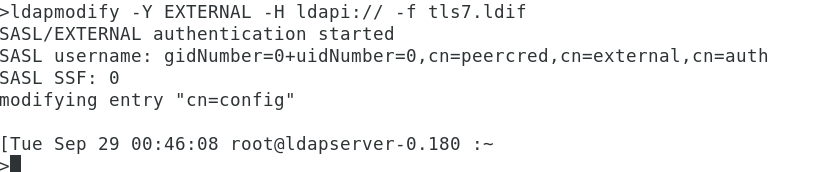
olcTLSCertificateKeyFile: /etc/openldap/certs/ldap.hopto.org.key

Change the ownership of /etc/openldap/certs and and /etc/openldap/cacerts directory

chown -R ldap:ldap /etc/openldap/certs

And we run the ldapmodify command with this LDIF file .

ldapmodify -Y EXTERNAL -H ldapi:// -f tls7.ldif



Next add a new attribute olcTLSCACertificateFile for CA certificate file. For this we will create another ldiff file

>vi tls7\_1.ldif

[Tue Sep 29 00:48:01 root@ldapserver-0.180 :~

>cat tls7\_1.ldif

dn: cn=config

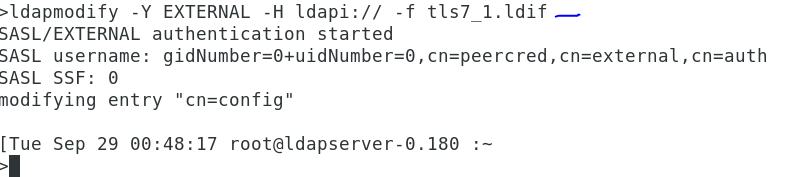
changetype: modify

add: olcTLSCACertificateFile

olcTLSCACertificateFile: /etc/openldap/certs/ca.cert.pem

Then we run

ldapmodify -Y EXTERNAL -H ldapi:// -f tls7\_1.ldif



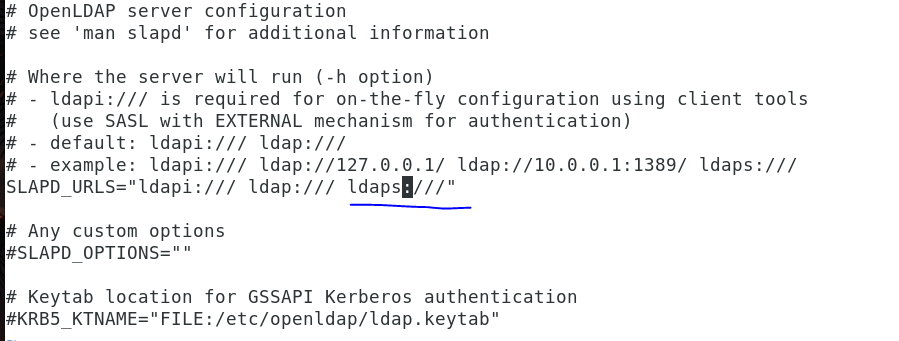
Validate the new values using slapchat.

slapcat -b "cn=config" | egrep "olcTLSCertificateFile|olcTLSCertificateKeyFile|olcTLSCACertificateFile"



Now we edit the /etc/sysconfig/slapd file to add ldaps:/// to the SLAPD\_URLS parameter

vi /etc/sysconfig/slapd

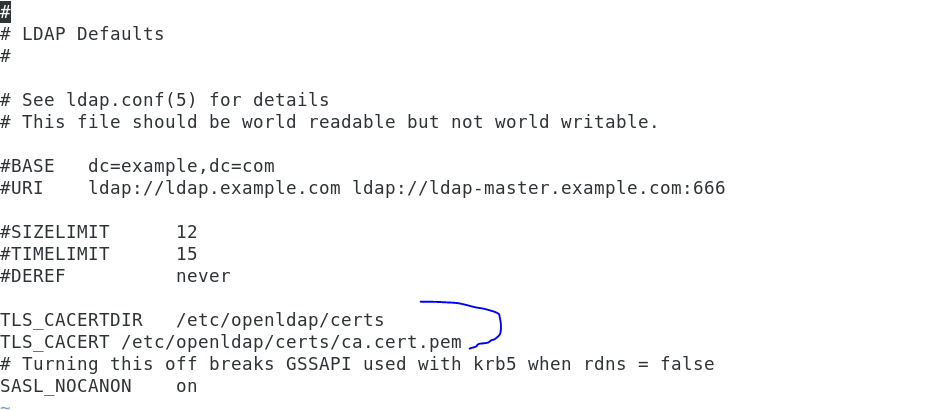


vi /etc/openldap/ldap.conf

TLS\_CACERTDIR /etc/openldap/certs

TLS\_CACERT /etc/openldap/certs/ca.cert.pem

Add the above in the file as below



NOTE: Since we are using self signed certificate it is important to specify the CA certificate using TLS\_CACERT or else you may get "TLS negotiation failure" during ldap client authentication. Alternatively you can choose to use TLS\_REQCERT never for insecure communication and ignore any certificate checks

Then we restart the service to activate our changes

systemctl restart slapd

systemctl status slapd

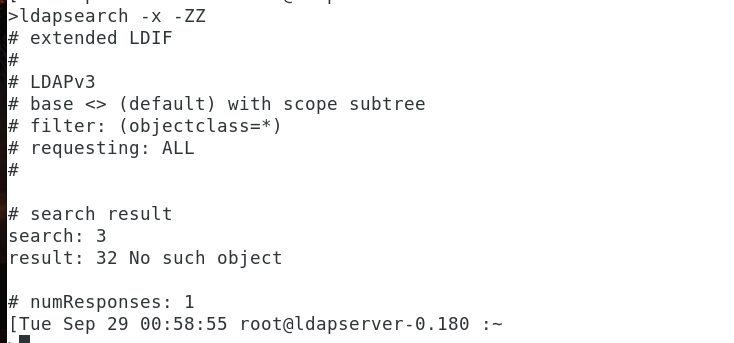


## Validate TLS connectivity for LDAP

To make sure that TLS for LDAP is working properly, we can check it by passing the -ZZ option to ldapsearch.

Thus, we’re telling ldapsearch to establish a TLS connection.

ldapsearch -x –ZZ



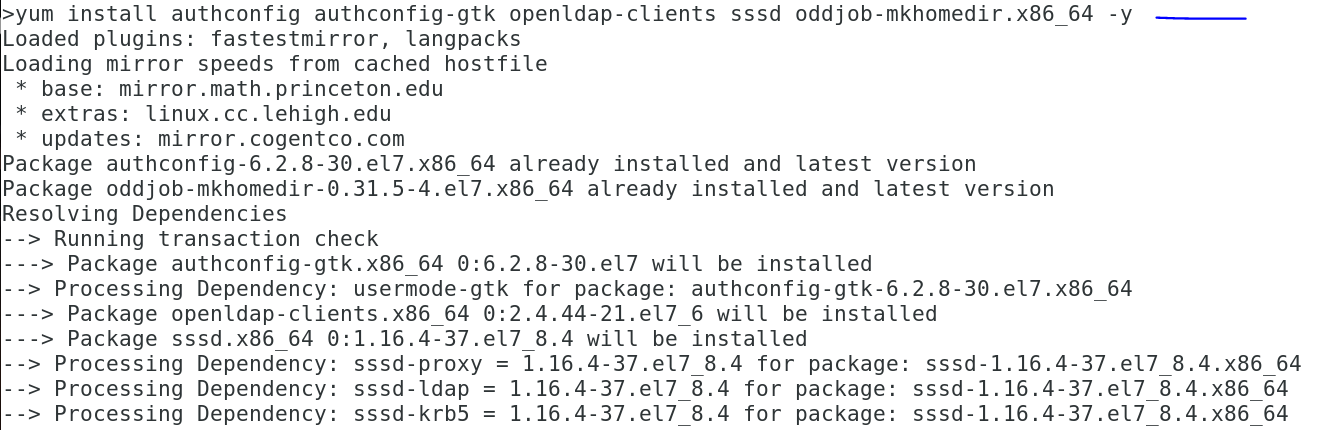
**Now since we have our LDAP server is ready with TLS certificates, next we will configure ldap client to authenticate with ldap server**

## Install Openldap client packages on client or database server

First let us install all the required openldap client and dependent packages

Since we plan to use authconfig to configure ldap client for our RHEL/CentOS 7 Linux node, we only install SSSD and authconfig packages. oddjob-mkhomedir is required to be able to create active directory user's home directory automatically.

**yum install authconfig authconfig-gtk openldap-clients sssd oddjob-mkhomedir.x86\_64 –y**

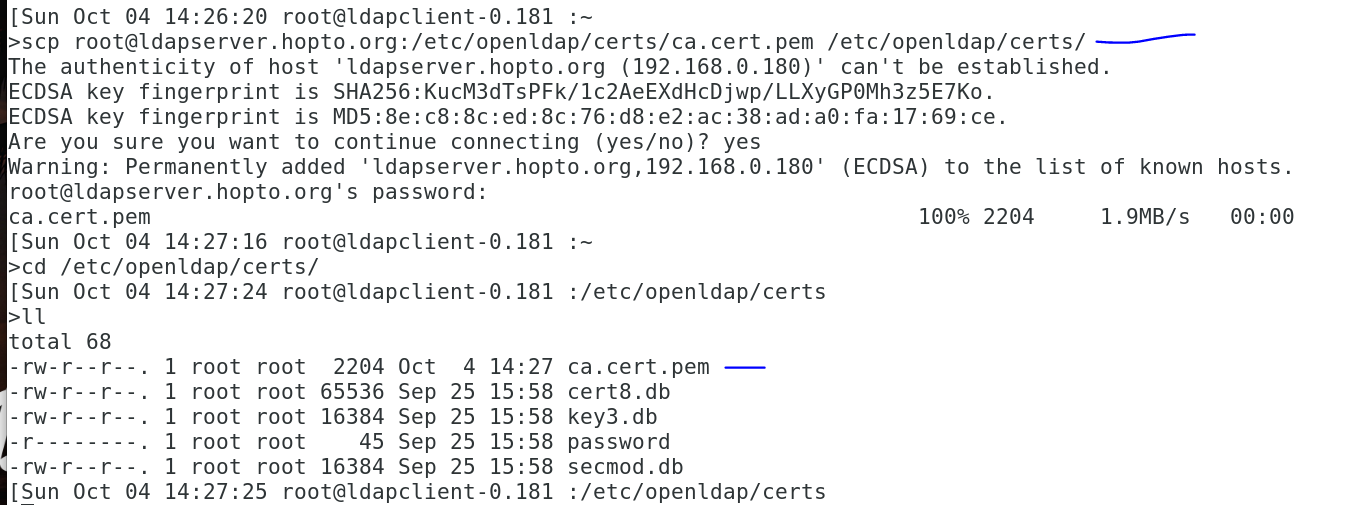


**If you wish to configure LDAP client using nslcd, then you must install nss-pam-ldapd, pam\_ldap. We will only use SSSD hence will not cover this part in our tutorial**

## Configure LDAP client or database server to authenticate with LDAP server. But in this case here, I will configure in server where database is running.

Before you start make sure you copy /etc/openldap/cacerts/ca.cert.pem from the **ldap-server** to **ldap-client** in the same location under /etc/openldap/cacerts/ca.cert.pem. This key will be referred by the authconfig tool

**scp** [**root@ldapserver.hopto.org:/etc/openldap/certs/ca.cert.pem /etc/openldap/cacerts/**](mailto:root@ldapserver.hopto.org:/etc/openldap/certs/ca.cert.pem%20/etc/openldap/cacerts/)

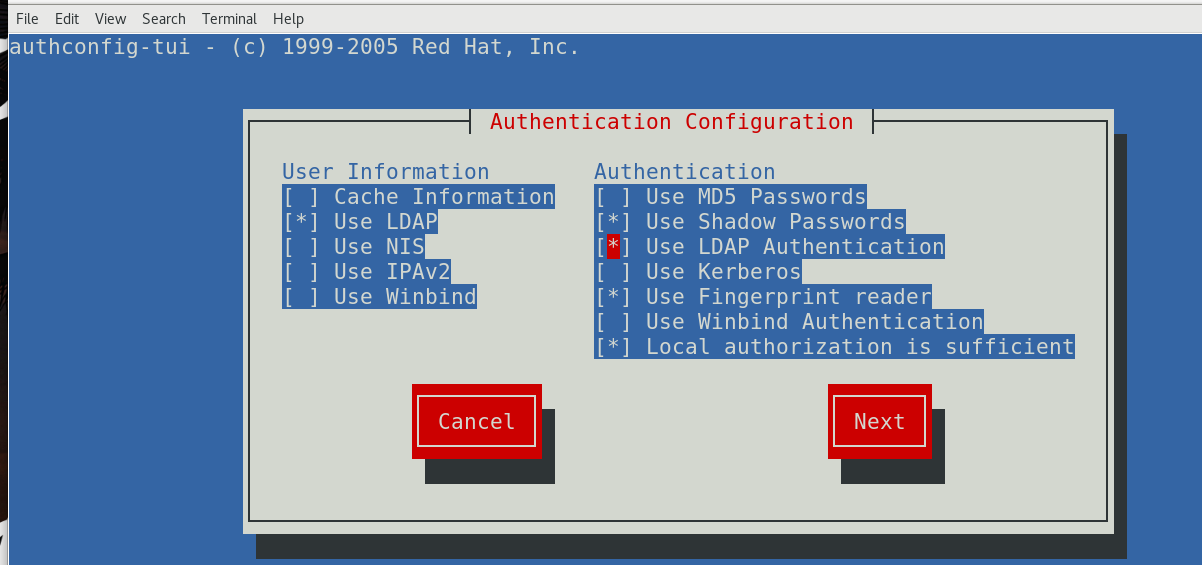


### Method 1: Using authconfig-tui

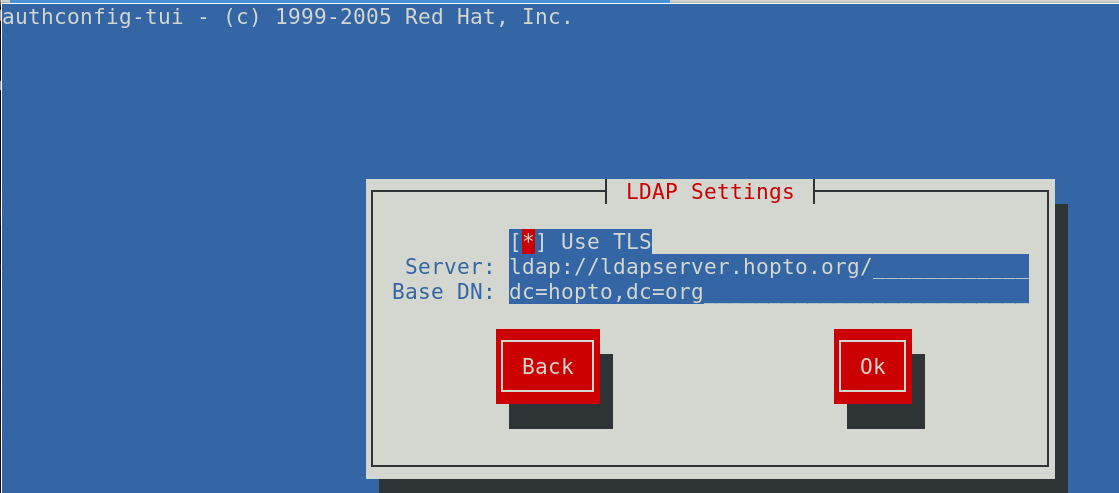
Configuring a client system to use an LDAP directory for user authentication is as easy as pie on a Fedora or RHEL system. Fedora has command-line utilities as well as GUI tools (for example, system-config-authentication, authconfig-gtk) that make it easy.

One of the command-line tools is provided by the package authconfig. To launch the tool from the command line, type the following:

authconfig-tui



* In the Authentication Configuration screen, navigate to (using TAB on your keyboard) and then select (using SPACEBAR on your keyboard) the following:  
  **Use LDAP** → Under the User Information section  
  **Use LDAP Authentication** → Under the Authentication section
* Navigate to the **Next** button and press ENTER to select it.



* Use the following information to complete the fields in the ensuing LDAP Settings screen:  
  **As seen above, make sure you put the hostname. If you decide to use the ip address, tls will not work because the certificates did not define that.**

Here 192.168.0.180 is the IP address of my ldap-server, replace it with your server details

* When we click **OK**, this will automatically change a series of files that otherwise would have to be changed by hand. For example, it will add the following lines to the /etc/openldap/ldap.conf file:

URI ldap:// ldapserver.hopto.org/

BASE dc=hopto,dc=org

TLS\_CACERTDIR /etc/openldap/certs



**Open the file as shown and make sure you modify** /etc/openldap/cacerts to /etc/openldap/certs as per my case

Add the following **TLS\_REQCERT nerver as seen below**



### Method 2: Using authconfig CLI

You can also configure ldap on the client using authconfig as shown below

**[root@ldap-client ~]# authconfig --enableldap --enableldapauth --ldapserver=192.168.0.180 --ldapbasedn="dc=hopto,dc=org" --enableldaptls –update**

*Since we have already configured client using authconfig-tui, we will skip the configuration using authconfig.*

If you get "Could not start TLS encryption. TLS: hostname does not match CN in peer certificate" error in /var/log/messages after restarting sssd then it means your client's hostname is not configured to use the CA certificate. In such case you must have a proper certificate generated for this client of use SAN certificate on the ldap server. Alternatively you can disable TLS check using TLS\_REQCERT never in /etc/openldap/ldap.conf and also ldap\_id\_use\_start\_tls = False in /etc/sssd/sssd.conf. But this will disable the TLS communication hence making the system insecure.

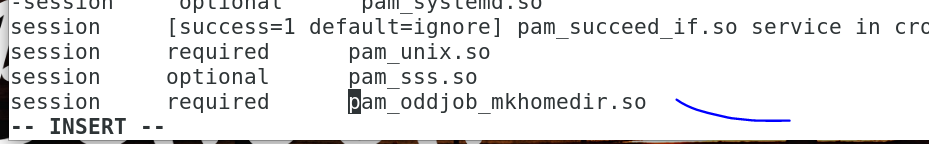
## Configure oddjob-mkhomedir to auto create home directories

We can use oddjob-mkhomedir to create home directories **automatically** for any ldap user which logs into the client node. This requires pam\_mkhomedir.so provided by ddjob-mkhomedir which we had already installed earlier.

Modify the PAM configuration to use pam\_oddjob\_mkhomedir. For example, add this line at the **bottom** of /etc/pam.d/system-auth:

**vi /etc/pam.d/system-auth**

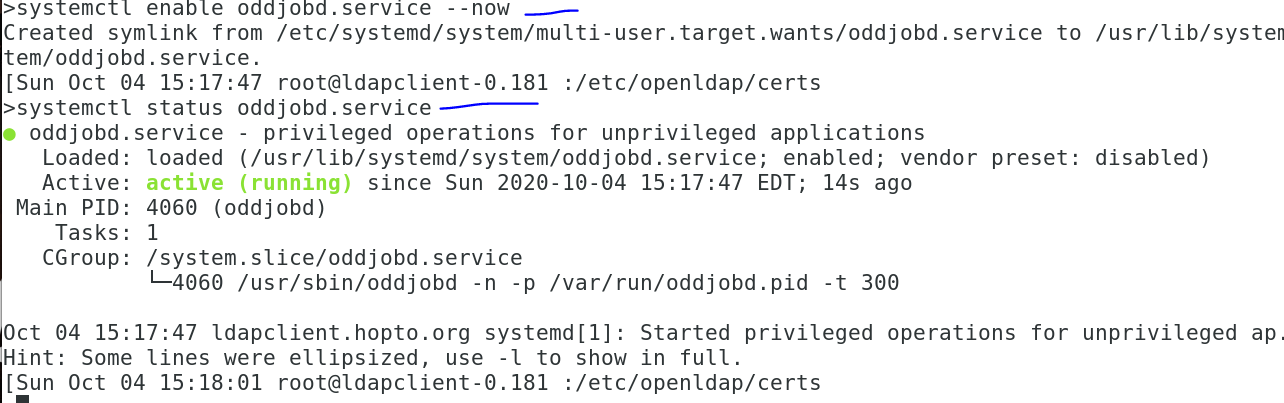
**session required pam\_oddjob\_mkhomedir.so**



Next enable and start oddjobd.service

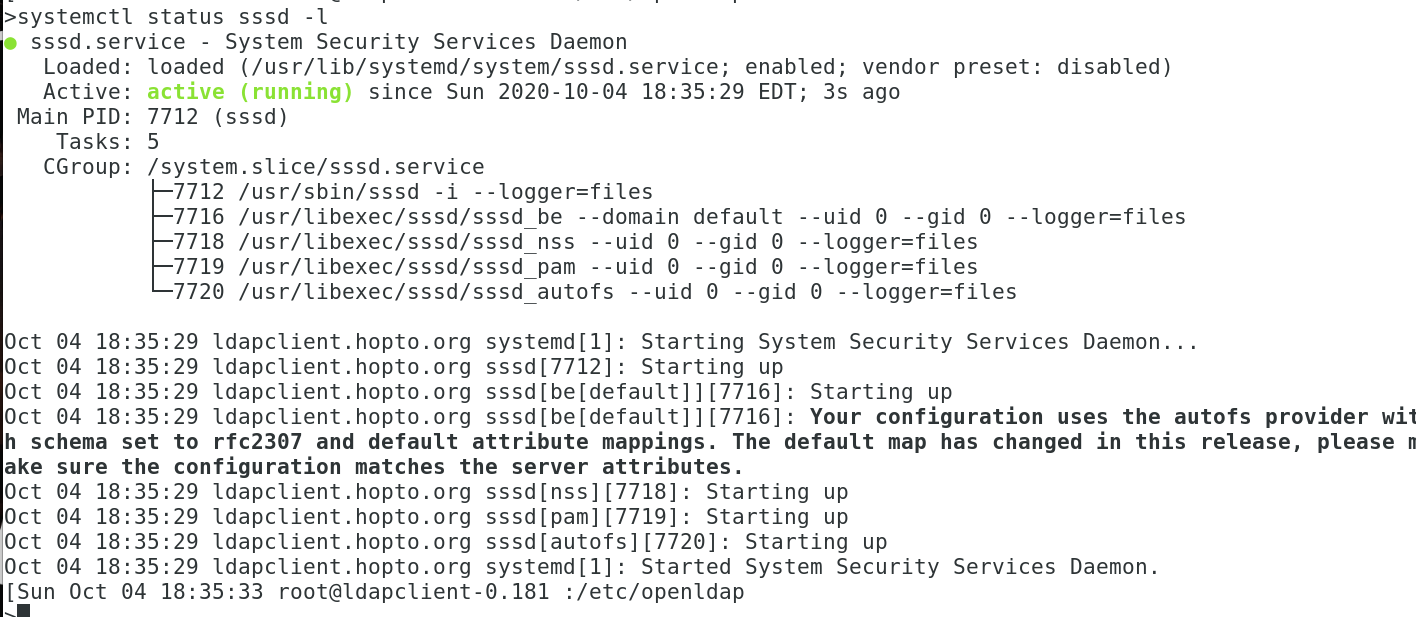
**systemctl enable oddjobd.service –now**

**systemctl status oddjobd.service**



**systemctl restart sssd.service**

**systemctl status sssd.service -l**



**Make sure you run the status with the –l to make sure you don’t have errors**

## Create LDAP user (Optional)

You can ignore this step if you already a ldap user.

In order to authenticate as an LDAP user, when we create the user, we have to include a series of fields, such as shell, uid, gid, etc. As an example, let’s add the user edbuser. We begin by creating the edbuser.ldif file, with the following content:

**ON MY LDAP SERVER, I DID THE FOLLOWING.**

**>user add edbuser**

**[Sun Oct 04 15:40:18 root@ldapserver-0.180 :~**

**[Sun Oct 04 15:40:21 root@ldapserver-0.180 :~**

**>su - edbuser**

**[edbuser@ldapserver ~]$ slappasswd**

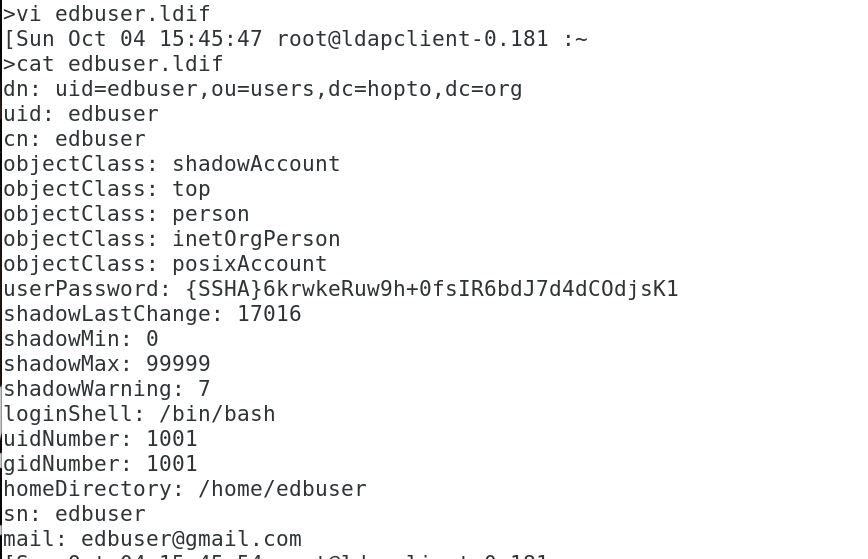
**New password:**

**Re-enter new password:**

**{SSHA}6krwkeRuw9h+0fsIR6bdJ7d4dCOdjsK1**

**[edbuser@ldapserver ~]$**

**ON MY CLIENT, I CREATE THE FILES BELOW FOR THE USER**



**dn: uid=edbuser,ou=users,dc=hopto,dc=org**

**uid: edbuser**

**cn: edbuser**

**objectClass: shadowAccount**

**objectClass: top**

**objectClass: person**

**objectClass: inetOrgPerson**

**objectClass: posixAccount**

**userPassword: {SSHA}6krwkeRuw9h+0fsIR6bdJ7d4dCOdjsK1**

**shadowLastChange: 17016**

**shadowMin: 0**

**shadowMax: 99999**

**shadowWarning: 7**

**loginShell: /bin/bash**

**uidNumber: 1001**

**gidNumber: 1001**

**homeDirectory: /home/edbuser**

**sn: edbuser**

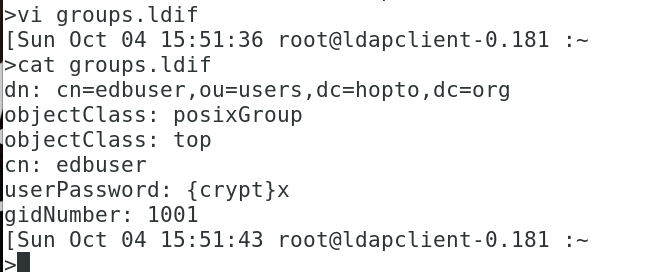
**mail: edbuser@gmail.com**

As see above, I already created an encrypted password for edbuser using slappasswd on **ldap-server** node. My password for edbuser is postgres

## Create LDAP group (Optional)

You can ignore this step if you already a ldap group.

Also we will need a group for this edbuser so I will add it to our existing OU "users"



**cat groups.ldif**

dn: cn=edbuser,ou=users,dc=hopto,dc=org

objectClass: posixGroup

objectClass: top

cn: edbuser

userPassword: {crypt}x

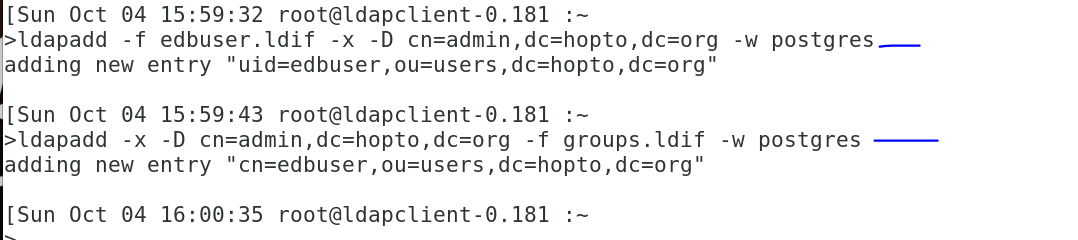
gidNumber: 1001

## Add user and group to LDAP database (Optional)

Add the users and groups to the ldap directory using the below commands

**ldapadd -f edbuser.ldif -x -D cn=admin,dc=hopto,dc=org -w postgres**

**ldapadd -x -D cn=admin,dc=hopto,dc=org -f groups.ldif -w postgres**



If you are getting ldap\_add: Invalid syntax (21); additional info: objectClass: value #0 invalid per syntax while executing above command then you must load nis.ldif using

# ldapadd -Y EXTERNAL -H ldapi:// -f /etc/openldap/schema/nis.ldif

SASL/EXTERNAL authentication started

SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth

SASL SSF: 0

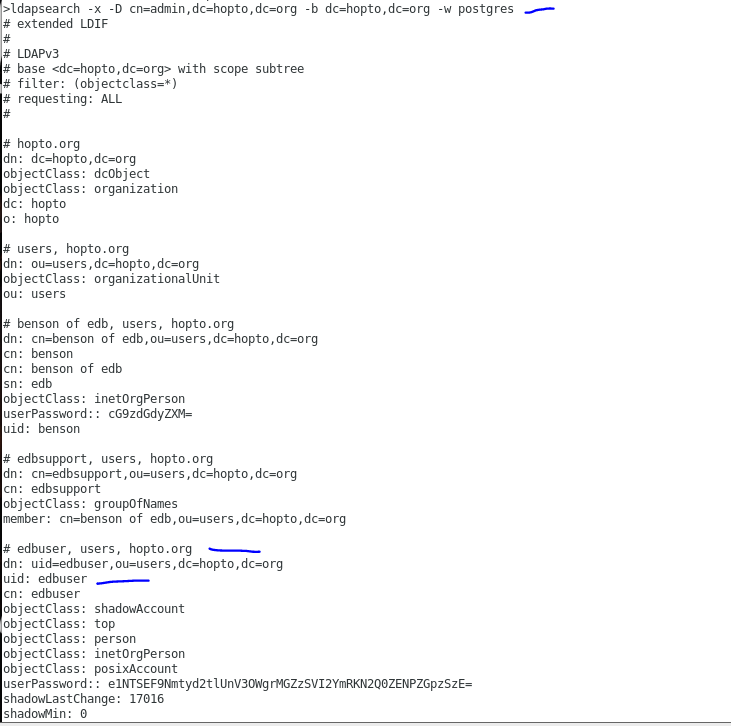
adding new entry "cn=nis,cn=schema,cn=config"

and then re-try the ldapadd command for user/group.

## Validate the new user and group (Optional)

Run the ldapsearch command again and verify users and groups are listed under the base DN to complete the configuration.

**ldapsearch -x -D cn=admin,dc=hopto,dc=org -b dc=hopto,dc=org -w postgres**

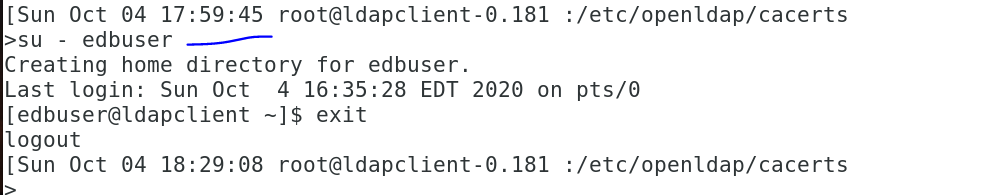


## Connect via LDAP User

Since we have installed oddjob, so we don't need to create ldap user's home directory manually and it will created during the first login of the user. Now we’ll be able to authenticate with an LDAP user

**su – edbuser**

**whoami**



**If your not able to su as the user, you can restart sssd**

**Next is the installation of postgres**

yum -y install <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>

yum -y install https://yum.enterprisedb.com/edb-repo-rpms/edb-repo-latest.noarch.rpm

**sed -i 's/<username>/benson.yerimah/g' /etc/yum.repos.d/edb.repo**

**sed -i 's/<password>/FDQsxz8UnheMvcUti/g' /etc/yum.repos.d/edb.repo**

y

-y

/usr/edb/as12/bin/edb-as-12-setup initdb

systemctl start edb-as-12

su – enterprisedb

:> .bash\_profile

vi .bash\_profile

export PGDATA=/var/lib/edb/as12/data

export PATH=/usr/edb/as12/bin:$PATH

export PGDATABASE=edb

export PGUSER=enterprisedb

export PGPORT=5444

export PGHOME=/var/lib/edb

export PGLOCALEDIR=/usr/edb/as12/share/locale

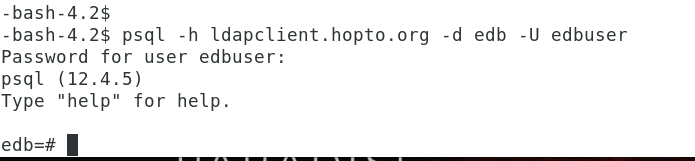
vi pg\_hba.conf

host all all 0.0.0.0/0 ldap ldapserver=192.168.0.180 ldapport=389 ldaptls=1 ldapprefix="uid=" ldapsuffix=",ou=users,dc=hopto,dc=org"

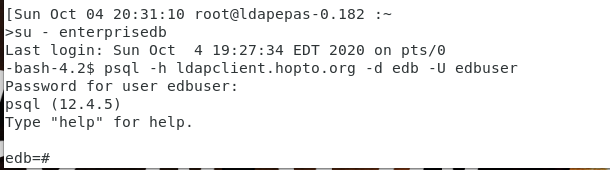


Test connection from same hosts..

**psql -h ldapclient.hopto.org -d edb -U edbuser**



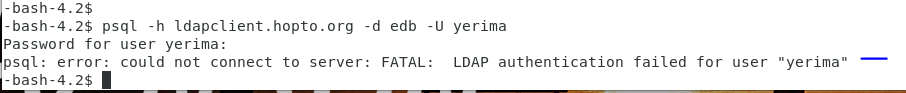
**TESTING THIS SAME CONNECTION FROM ANOTHER CLIENT SERVER WITH NO SETUP.. JUST psql utility installed.**



**This works fine.**

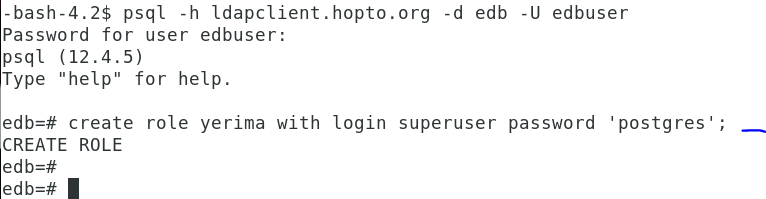
**TEST CASE AND ADDING USER.**

**psql -h ldapclient.hopto.org -d edb -U yerima**

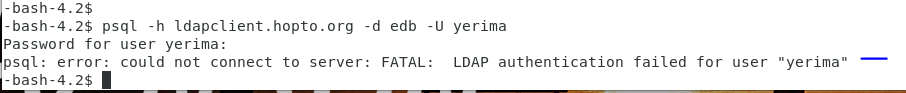


**Inorder to get the above user to login, first I add the user in the postgres server (database)**

**create role yerima with login superuser password 'postgres';**



**Note that even after creating the user, I cant login.**



**Next step will be to add the user in ldap server**

**On the ldap server, I added the user and created the password.**

[Sun Oct 04 21:01:22 root@ldapserver-0.180 :~

>useradd yerima

[Sun Oct 04 21:01:28 root@ldapserver-0.180 :~

>su - yerima

[yerima@ldapserver ~]$ slappasswd

New password:

Re-enter new password:

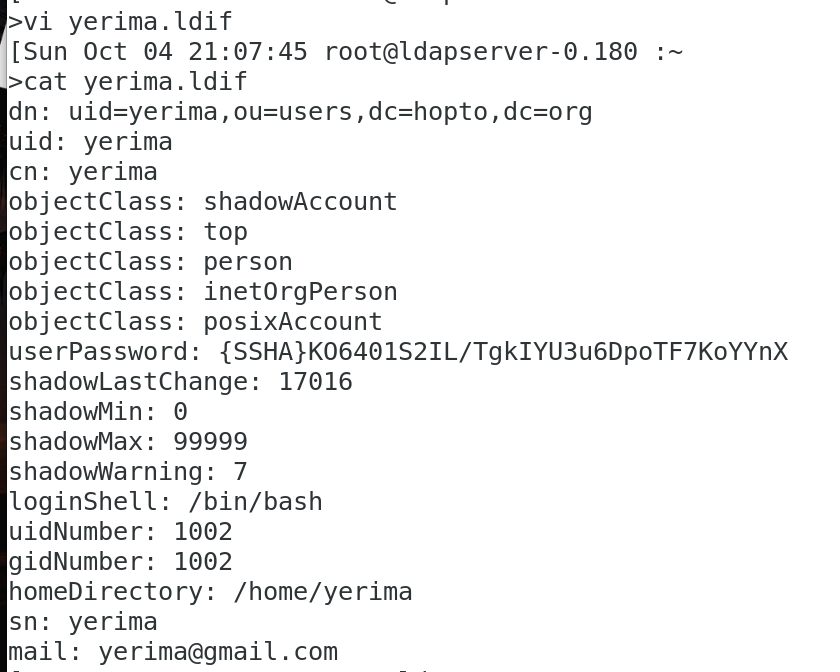
{SSHA}KO6401S2IL/TgkIYU3u6DpoTF7KoYYnX

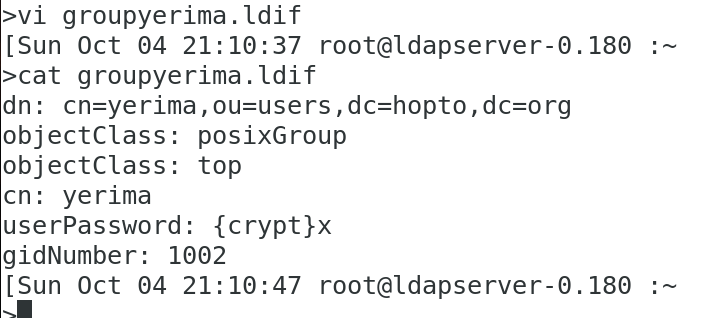
[yerima@ldapserver ~]$ id

uid=1002(yerima) gid=1002(yerima) groups=1002(yerima) context=unconfined\_u:unconfined\_r:unconfined\_t:s0-s0:c0.c1023

[yerima@ldapserver ~]$

**Created the ldif file for yerima**

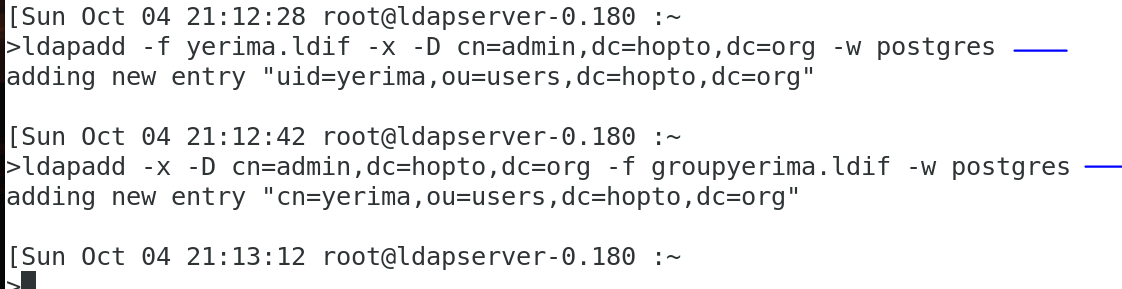




**Now I add the user and group to the ldap database**

ldapadd -f yerima.ldif -x -D cn=admin,dc=hopto,dc=org -w postgres

ldapadd -x -D cn=admin,dc=hopto,dc=org -f groupyerima.ldif -w postgres



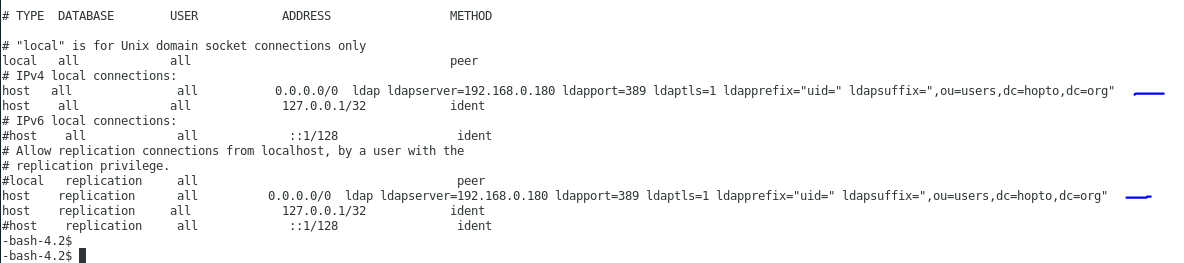
**Now let’s test the connection again from another host.**

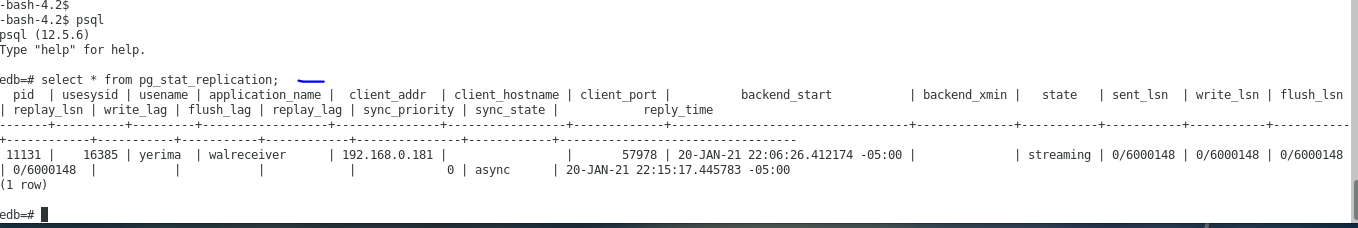
**psql -h ldapclient.hopto.org -d edb -U yerima**



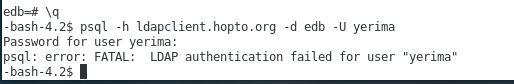
**What if we setup streaming replication for the servers?**

I was able to setup streaming replication with the following files setup on the master..





SO we could see that streaming replication is working fine but when we try to connect to the standby server, we get the error as seen below..



And this is because the standby server as needs to be setup to accept encrypted connections. So we need to setup and also copy the certs files to this server.

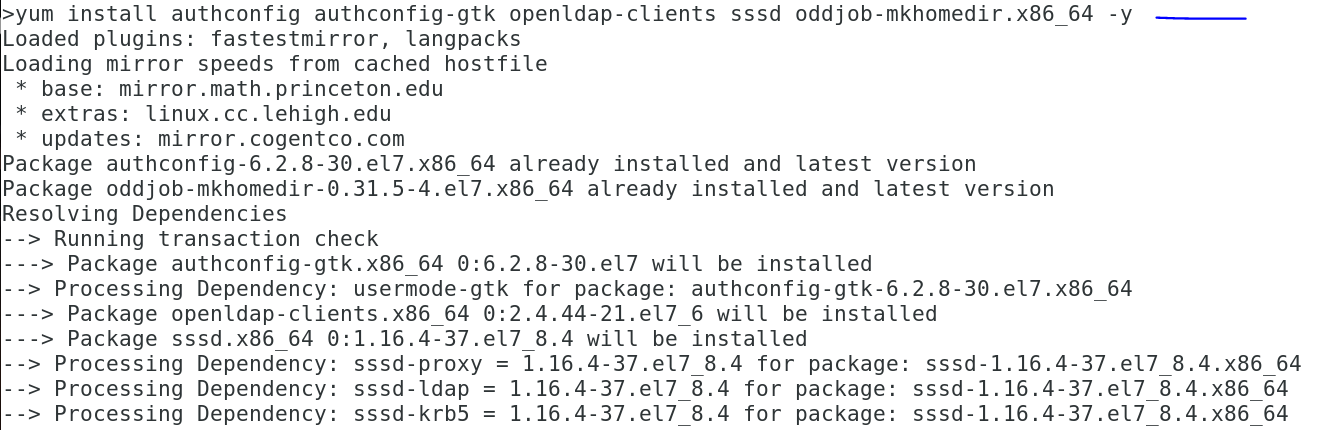
So we will need to repeat the following steps on the standby server.

## Install Openldap client packages for standby

First let us install all the required openldap client and dependent packages

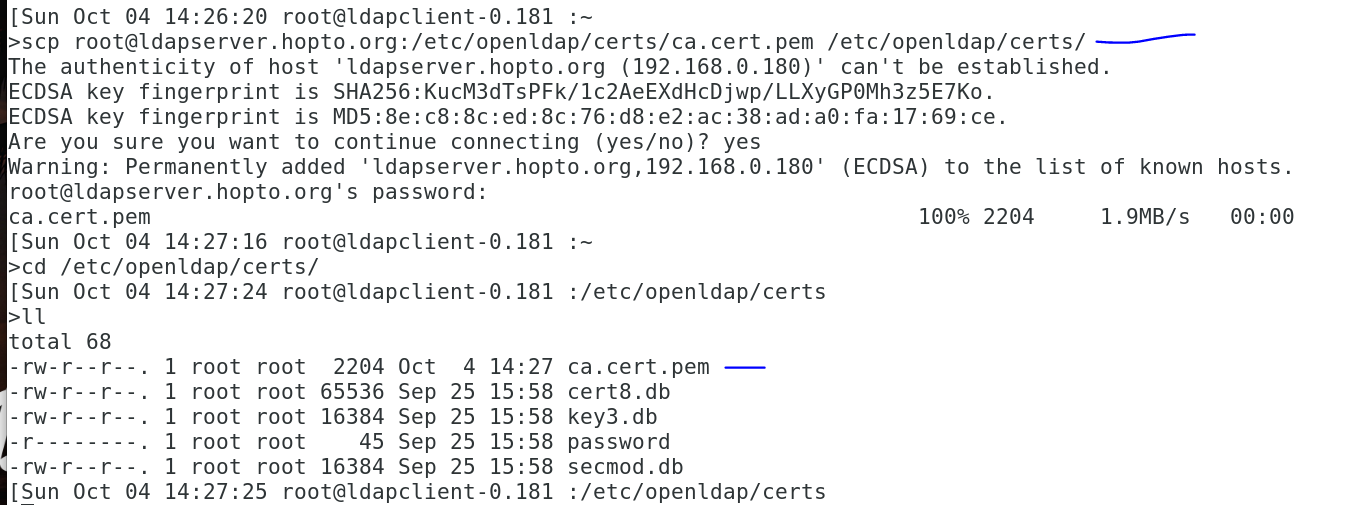
Since we plan to use authconfig to configure ldap client for our RHEL/CentOS 7 Linux node, we only install SSSD and authconfig packages. oddjob-mkhomedir is required to be able to create active directory user's home directory automatically.

**yum install authconfig authconfig-gtk openldap-clients sssd oddjob-mkhomedir.x86\_64 –y**



## Configure LDAP client or database server to authenticate with LDAP server. But in this case here, I will configure in server where database is running.

**scp** [**root@ldapserver.hopto.org:/etc/openldap/certs/ca.cert.pem /etc/openldap/cacerts/**](mailto:root@ldapserver.hopto.org:/etc/openldap/certs/ca.cert.pem%20/etc/openldap/cacerts/)

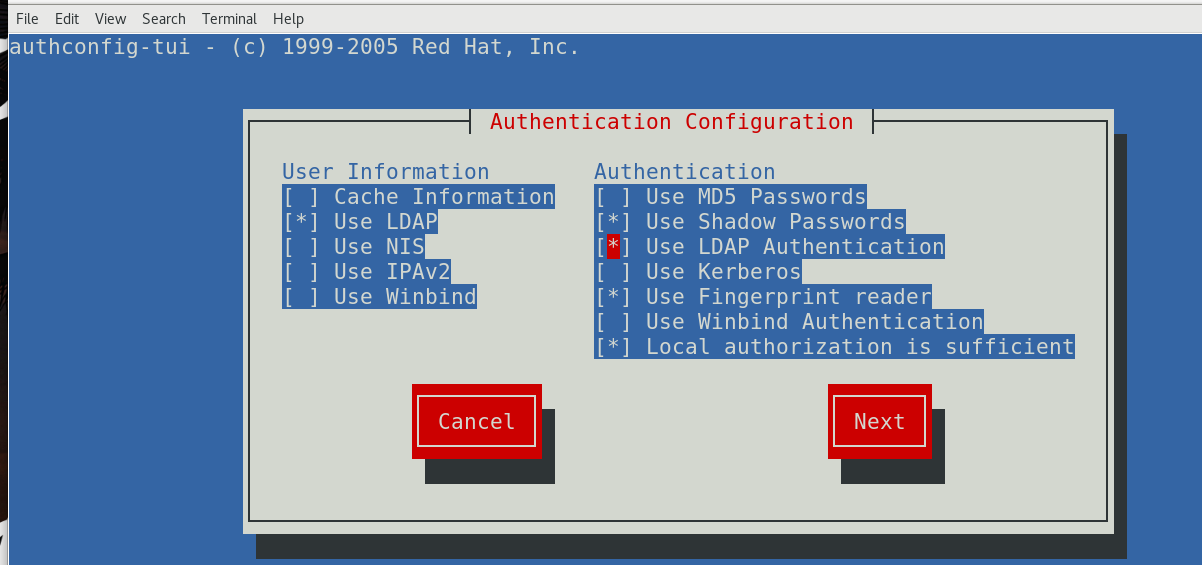


### Method 1: Using authconfig-tui

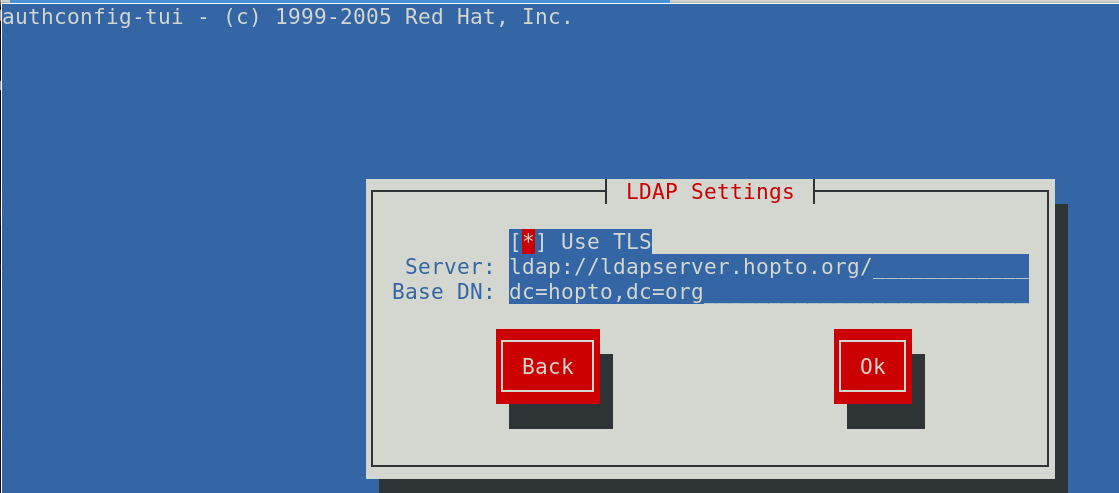
Configuring a client system to use an LDAP directory for user authentication is as easy as pie on a Fedora or RHEL system. Fedora has command-line utilities as well as GUI tools (for example, system-config-authentication, authconfig-gtk) that make it easy.

One of the command-line tools is provided by the package authconfig. To launch the tool from the command line, type the following:

authconfig-tui



* In the Authentication Configuration screen, navigate to (using TAB on your keyboard) and then select (using SPACEBAR on your keyboard) the following:  
  **Use LDAP** → Under the User Information section  
  **Use LDAP Authentication** → Under the Authentication section
* Navigate to the **Next** button and press ENTER to select it.



* Use the following information to complete the fields in the ensuing LDAP Settings screen:  
  **As seen above, make sure you put the hostname. If you decide to use the ip address, tls will not work because the certificates did not define that.**

Here 192.168.0.180 is the IP address of my ldap-server, replace it with your server details

* When we click **OK**, this will automatically change a series of files that otherwise would have to be changed by hand. For example, it will add the following lines to the /etc/openldap/ldap.conf file:

URI ldap:// ldapserver.hopto.org/

BASE dc=hopto,dc=org

TLS\_CACERTDIR /etc/openldap/certs



**Open the file as shown and make sure you modify** /etc/openldap/cacerts to /etc/openldap/certs as per my case

Add the following **TLS\_REQCERT nerver as seen below**



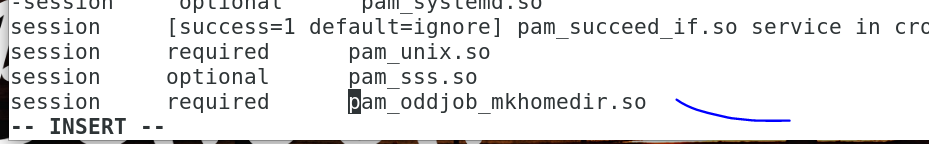
## Configure oddjob-mkhomedir to auto create home directories

We can use oddjob-mkhomedir to create home directories **automatically** for any ldap user which logs into the client node. This requires pam\_mkhomedir.so provided by ddjob-mkhomedir which we had already installed earlier.

Modify the PAM configuration to use pam\_oddjob\_mkhomedir. For example, add this line at the **bottom** of /etc/pam.d/system-auth:

**vi /etc/pam.d/system-auth**

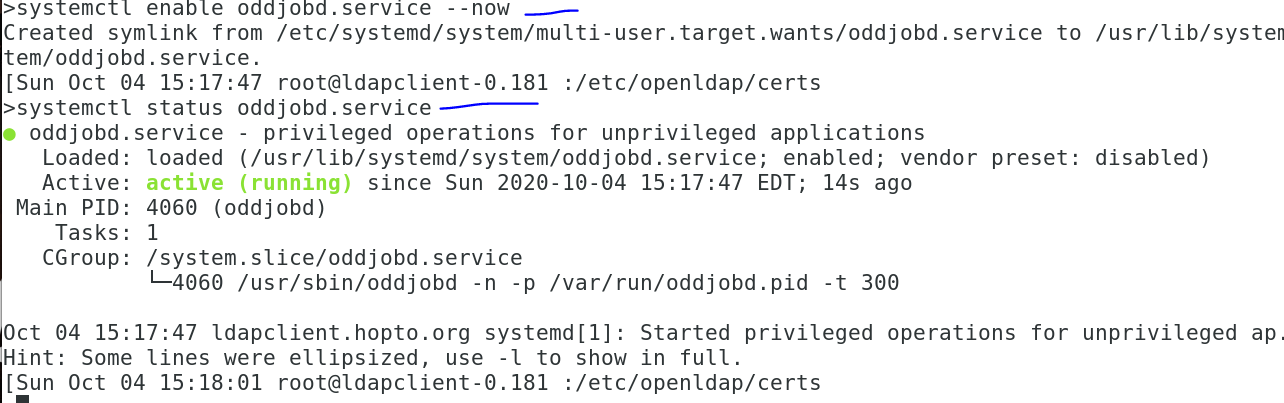
**session required pam\_oddjob\_mkhomedir.so**

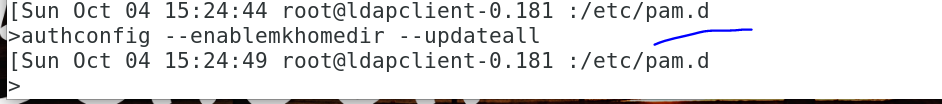


Next enable and start oddjobd.service

**systemctl enable oddjobd.service –now**

**systemctl status oddjobd.service**

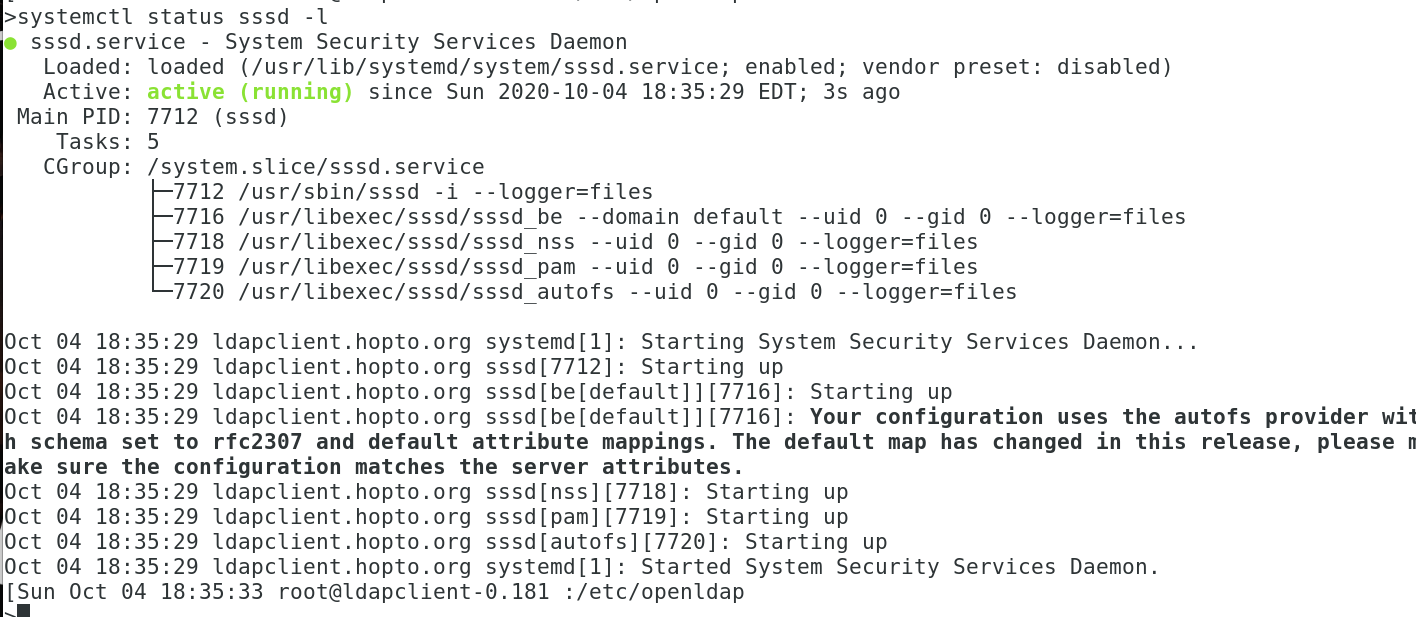




Also restart sssd.service

**systemctl restart sssd.service**

**systemctl status sssd.service -l**



**Make sure you run the status with the –l to make sure you don’t have errors**

**After the above configuration on the standby, everything works fine now..**

