LDAP managed mail server with Postfix and Dovecot for multiple domains

This article will describe how to set up and configure a secure mail system with Postfix and Dovecot as `SMTP` and `IMAP` server, and `OpenLDAP` as a backend for user authentication and mail routing.

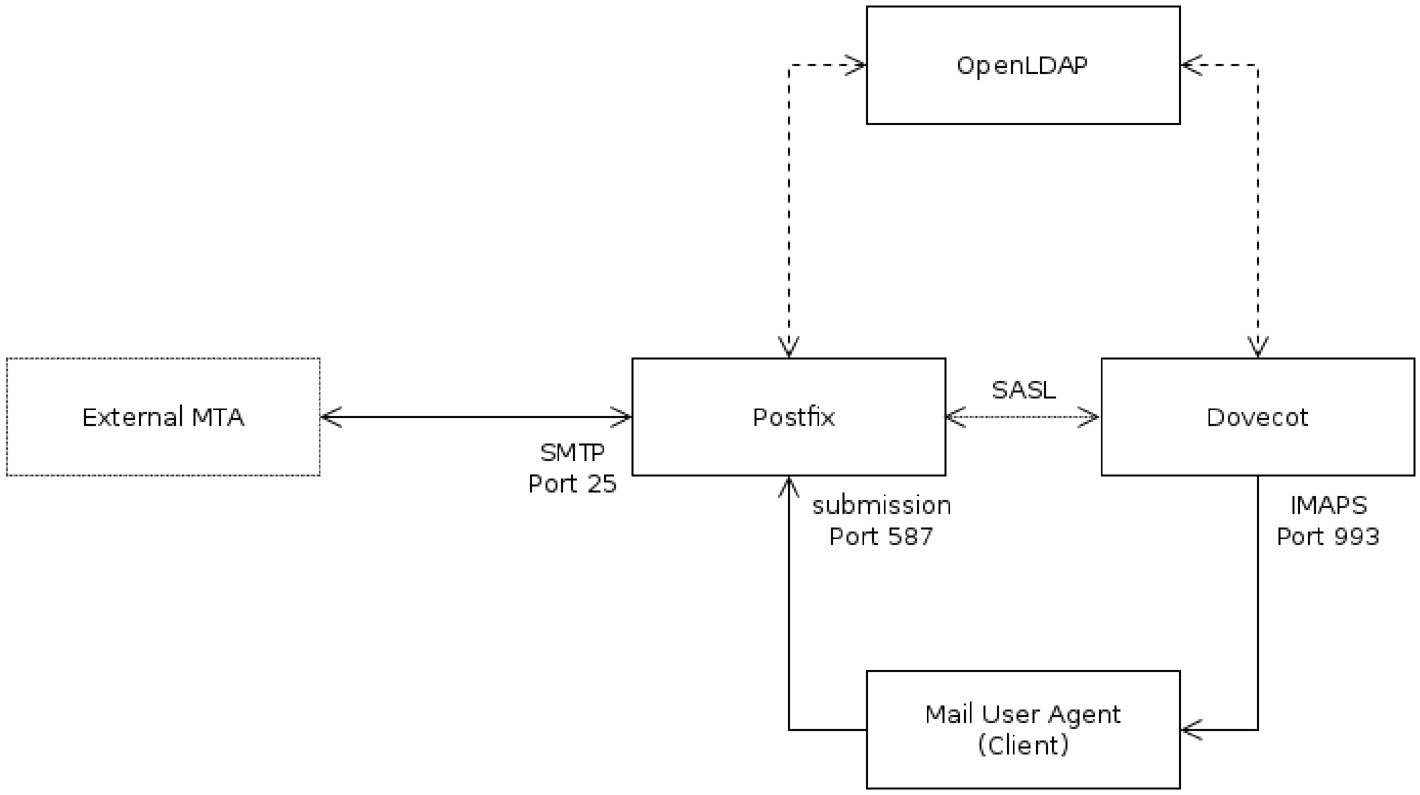
All services will be configured to use `TLS` by default to ensure transport layer security wherever possible. It's assumed that you already have a working `OpenLDAP` installation running with available `TLS` support as outlined in my OpenLDAP article.

The OS in use is Debian 8 but the conceptual design should work independently from that.

If you find any issues with the setup described here, or you have suggestions for improvements, please contact me.

Table of contents

Overview



The setup consists of three different software components. `Postfix` will relay mails submitted by clients to other MTAs and receive mails from other MTAs to be stored in the user's `maildir`. `Dovecot` will serve the user's `maildir` via `IMAPS` so it can be read and managed by client software like Thunderbird. `Dovecot` also acts as an `SASL` authentication provider for `Postfix`. Information about user accounts and mail aliases is stored in the `LDAP` directory and queried by `Postfix` and `Dovecot`.

The whole setup won't involve any “real” accounts on the OS. For `Postfix` we can utilize virtual mailboxes to create mailboxes with any `UID` and `GID` we want, and `Dovecot` supports virtual users that don't need to exist in the OS's context.

LDAP

The LDAP directory needs to keep several information needed by `Postfix` and `Dovecot` to work.

- Username & Password for authentication
- `UID` and `GID` for managing permissions of the user's `maildir`
- Location of the `maildir`
- List of mail aliases for a given user to allow one user to have several mail adresses.

The account information can be stored with the `posixAccount` objectClass. To store the aliases a new `LDAP` scheme needs to be installed. It will feature a objectClass `postfixUser` with the two attributes

- `mailacceptinggeneralid` to store a list of mail aliases
- `maildrop` to set one or more final destinations for mail sent to the given aliases

To install the `postfixUser` scheme, download the `LDIF` to your LDAP host and add it to `cn=schema,cn=config`.

```
root@ldaphost:~# wget -O postfix.ldif https://raw.githubusercontent.com/68b32/postfix-lda
root@ldaphost:~# ldapadd -ZZ -x -W -D cn=admin,cn=config -H ldap://ldap.example.com -f po
```

To keep mail users seperate from others and allow privilege seperation in your directory, you might want to store them within an `organizationalUnit` e.g. `ou=Mail,dc=example,dc=com`.

```
dn: ou=Mail,dc=example,dc=com
objectclass: organizationalUnit
objectclass: top
ou: Mail
```

Create a mail account that can be used for testing purposes when configuring postfix and dovecot. To create a hashed password, use `slappasswd`. Make sure not to use any UID or GID used by another user or process.

```
dn: uid=mail000,ou=Mail,dc=example,dc=com
cn: mail000
gidnumber: 20000
homedirectory: /home/mail/mail000
mailacceptinggeneralid: test0@hosted-domain.com
mailacceptinggeneralid: test1@hosted-domain.com
maildrop: mail000@mail.example.com
objectclass: account
objectclass: posixAccount
objectclass: postfixUser
objectclass: top
uid: mail000
uidnumber: 20000
userpassword: {SSHA}ncdXGXXD6zaG76dSCVKVAQLH4vPcHaTa
```

The aliases listed with `mailacceptinggeneralid` can contain any domain thats `MX` record points to your mail server. If you are not sure what domain to use for `maildrop`, don't worry, it will be explained when postfix is set up.

To speed up the dircetory lookups, indices should be created for the `mailacceptinggeneralid` and `maildrop` attribute. Add the following `LDIF` to your database definition in `cn=config` to activate indices.

```
dn: olcDatabase={1}mdb,cn=config
objectclass: olcDatabaseConfig
objectclass: olcMdbConfig
olcdbindex: mailacceptinggeneralid eq,sub
olcdbindex: maildrop eq
```

Since `postfix` and `dovecot` will query the directory, a seperate user should be created along with sufficient permissions to read the `ou=Mail,dc=example,dc=com` subtree.

```
dn: cn=mailAccountReader,ou=Manager,dc=example,dc=com
cn: mailAccountReader
objectclass: organizationalRole
objectclass: simpleSecurityObject
objectclass: top
userpassword: {SSHA}GY6RPlSGKI7SPKnnT7i/ktb/3JGmCHLT
```

```
to attrs=userPassword
  by self =xw
  by anonymous auth
  by * none
```

```
to dn.subtree="ou=Mail,dc=example,dc=com"
  by dn.base="cn=mailAccountReader,ou=Manager,dc=example,dc=com" read
  by * none
```

Make sure that anonymous LDAP accounts are allowed to authenticate against their `userPassword`, since this mechanism will be used by `Dovecot` to authenticate clients.

Postfix

Postfix can be installed from the Debian repository.

```
root@mailhost:~# apt-get install postfix postfix-ldap
```

Select *Internet site* as initial type of configuration. Then enter the `FQDN` of your mail host. This is probably the same as configured for `myhostname` in `/etc/postfix/main.cf` (see next section). This will leave you with a basic `/etc/postfix/main.cf` to get started with the postfix configuration.

The hostname

The very first parameter to configure is the `myhostname` directive. This is the hostname of the mail server, and should be the same as the `MX` record of the domains this server will receive mails for. It also should be set as PTR record for the IP address this hostname resolves to, since it is common to block mail from hosts where this is not the case. To check this out perform a reverse lookup with `dig -x <IP address>`.

main.cf

```
myhostname = mail.example.com
```

LDAP mapping

Before configuring postfix to deliver and receive mail, we create some LDAP lookup tables postfix will use to query the direcotry. The next table gives an overview on how information is queried.

	Key	Value
virtual_alias_domains	Domain part of an address name@hosted-domain.com	Anything if this is a hosted domain, nothing otherwise
virtual_alias_maps	Address in form name@hosted-domain.com	Address in form user@mail.example.com
virtual_mailbox_maps	Address in form user@mail.example.com	Path to mail directory
virtual_uid_maps	Address in from user@mail.example.com	Numeric UID to be used for mail directory
smtpd_sender_login_maps	Address in form name@hosted-domain.com	Login username allowed to use this sender address

Create a directory `/etc/postfix/ldap` to keep all map definitions in one place. Since the files in this directory will contain your LDAP credentials, set its owner to `postfix:postfix` and its mode to `0100`.

The connection information for your LDAP directory is common to all maps and can be copied to the top of all of them.

```
server_host = ldap://ldap.example.com
start_tls = yes
version = 3
tls_ca_cert_file = /etc/ldap/tls/CA.pem
tls_require_cert = yes

bind = yes
bind_dn = cn=mailAccountReader,ou=Manager,dc=example,dc=com
bind_pw = <Password for bind_dn>

search_base = ou=Mail,dc=example,dc=com
scope = sub
```

This will connect to the LDAP directory using TLS and check the validity of the certificate provided by the peer. This needs to be included into all files in `/etc/postfix/ldap` listed next.

virtual_alias_domains

```
query_filter = mailacceptinggeneralid=*@%s
result_attribute = mailacceptinggeneralid
result_format = %d
```

virtual_alias_maps

```
query_filter = mailacceptinggeneralid=%s
result_attribute = maildrop
```

virtual_mailbox_maps

```
query_filter = maildrop=%s
result_attribute = homeDirectory
result_format = %s/mailbox/
```

virtual_uid_maps

```
query_filter = maildrop=%s
result_attribute = uidNumber
```

smtpd_sender_login_maps

```
query_filter = (|(mailacceptinggeneralid=%s)(maildrop=%s))
result_attribute = uid
```

You can test the mapping with the `postmap` command. Here is some sample output for the user created for testing.

```
root@mailhost:~# postmap -q hosted-domain.com ldap:/etc/postfix/ldap/virtual_alias_domain
hosted-domain.com, hosted-domain.com

root@mailhost:~# postmap -q mail000@mail.example.com ldap:/etc/postfix/ldap/virtual_mailb
/home/mail/mail000/mailbox/

root@mailhost:~# postmap -q mail000@mail.example.com ldap:/etc/postfix/ldap/virtual_uid_m
20000

root@mailhost:~# postmap -q test0@hosted-domain.com ldap:/etc/postfix/ldap/smtpd_sender_l
mail000
```

If lookups don't work as expected, set `debuglevel = -1` in the map definition and review the output when running `postmap`.

Since the files contain credentials for binding to the directory make sure to set proper file permissions.

```
root@mailhost:~# chown postfix:postfix /etc/postfix/ldap/*
root@mailhost:~# chmod 400 /etc/postfix/ldap/*
```

Another problem to solve is that postfix will run in a chrooted environment (`/var/spool/postfix`) and will not be conscious about the CA certificate `/etc/ldap/tls/CA.pem` needed to validate the server's certificate. To counter this problem, create `/var/spool/postfix/etc/ldap/tls/` and either copy `/etc/ldap/tls/CA.pem` to this directory, or create a bind mount to make the CA certificate available in both locations.

```
root@mailhost:~# mkdir -p /var/spool/postfix/etc/ldap/tls
root@mailhost:~# touch /var/spool/postfix/etc/ldap/tls/CA.pem
root@mailhost:~# mount --bind /etc/ldap/tls/CA.pem /var/spool/postfix/etc/ldap/tls/CA.pen
```

Be aware that this won't persist through reboots, so make sure to bind the directory before postfix is started. This can be achieved with a `systemd` mount unit. Create `/etc/systemd/system/var-spool-postfix-etc-ldap-tls-CA.pem.mount` with following content.

var-spool-postfix-etc-ldap-tls-CA.pem.mount

```
[Unit]
Description=Bind /etc/ldap/tls/CA.pem to /var/spool/postfix/etc/ldap/tls/CA.pem
Before=postfix.service

[Mount]
What=/etc/ldap/tls/CA.pem
Where=/var/spool/postfix/etc/ldap/tls/CA.pem
Type=none
Options=bind

[Install]
WantedBy=postfix.service
```

Reload `systemd` and start the unit.

```
root@mailhost:~# systemctl daemon-reload
root@mailhost:~# systemctl start var-spool-postfix-etc-ldap-tls-CA.pem.mount
root@mailhost:~# mount | grep CA
```

Check the output of `mount` to see if the bind was successful and enable the bind mount to be set up before postfix on boot.

```
root@mailhost:~# systemctl enable var-spool-postfix-etc-ldap-tls-CA.pem.mount
Created symlink from /etc/systemd/system/postfix.service.wants/var-spool-postfix-etc-ldap
```

Reboot the host to see if the bind is created during boot.

The Mailboxes

Postfix implements different methods to deliver mail to the recipient. In this setup we will make use of the *virtual alias domain class* and the *virtual mailbox domain class*. Each final mailbox is associated with a definite address `<username>@$myhostname`. `$myhostname` contains the hostname set in `/etc/postfix/main.cf` and will be listed in `virtual_mailbox_domains`. All hosted domains are treated as *virtual alias domains*, and will be listed in `virtual_alias_domains` with the LDAP map defined earlier and mapped to the address of the final mailbox.

The testuser `mail000` as created in the LDAP section contains the adresses `test0@hosted-domain.com` and `test1@hosted-domain.com` as `mailacceptinggeneralid`. When a mail is received for one of these addresses, postfix will look for the corresponding `maildrop` attribute, which could either be a remote address (e.g. to just forward to a Gmail address), or an address with a domain listed in `virtual_mailbox_domains`, e.g. `mail000@mail.example.com`. Mails to domains listed in `virtual_mailbox_domains` will be stored in the user's mailbox. Information on where and how to create the mailbox is also retrieved from LDAP, using the mailbox address.

/etc/postfix/main.cf

```
myhostname = mail.example.com

virtual_alias_domains = ldap:/etc/postfix/ldap/virtual_alias_domains
virtual_mailbox_domains = $myhostname

virtual_alias_maps = ldap:/etc/postfix/ldap/virtual_alias_maps
virtual_mailbox_base = /
virtual_mailbox_maps = ldap:/etc/postfix/ldap/virtual_mailbox_maps
virtual_uid_maps = ldap:/etc/postfix/ldap/virtual_uid_maps
virtual_gid_maps = ldap:/etc/postfix/ldap/virtual_uid_maps

smtpd_sender_login_maps = ldap:/etc/postfix/ldap/smtpd_sender_login_maps
```

It is very important to list each domain only once. E.g. do not list `$myhostname` in `mydestination` or `virtual_alias_domains` when already listed in `virtual_mailbox_domains`.

`virtual_alias_domains` will expand to all domains used in addresses given in any `mailacceptinggeneralid` attribute in the `ou=Mail,dc=example,dc=com` subtree.

The mailboxes will be stored in maildir format if the value returned by the `virtual_mailbox_maps` map ends with `/`. The value returned will be appended to `virtual_mailbox_base`. The map used in this setup will return the path stored in the user's `homedirectory` attribute, with `/mailbox/` appended. So for the testuser created earlier, mail to `test0@hosted-domain.com` and `test1@hosted-domain.com` will first be redirected to `mail000@mail.example.com` and then stored to `/home/mail/mail000/mailbox/` in *maildir* format.

The directory `/home/mail` containing the user homes must be created in advance and must be writeable by all potential mail users. To achieve this, either use the same `GID` across all mail users and set group ownership to that `GID`, or make the directory world writable.

```
root@mailhost:~# mkdir -p /home/mail
root@mailhost:~# chmod o+w /home/mail                                 # Make world wri
root@mailhost:~# chown :<common GID> /home/mail && chmod g+w /home/mail  # use common grc
```

The user specific directories will be created with the user's `UID` and `GID` and minimal permissions by postix when mail is received.

Reload postfix and try to send a test mail to your new server.

```
root@mailhost:~# systemctl reload postfix
root@mailhost:~# tail -f /var/log/mail.info
```

When the mail is received, the logs should contain something like

```
.../smtpd[6693]: connect from unknown[xxxx:...]
.../smtpd[6693]: A7D1665808DE: client=unknown[xxxx:...]
.../cleanup[6697]: A7D1665808DE: message-id=<56E9374F.8020607@external.example.com>
.../qmgr[5561]: A7D1665808DE: from=<snip@external.example.com>, size=892, nrcpt=1 (queue
.../smtpd[6693]: disconnect from unknown[xxx:...]
.../virtual[6698]: A7D1665808DE: to=<mail000@mail.example.com>, orig_to=<test0@hosted-dor
.../qmgr[5561]: A7D1665808DE: removed
```

then check if the maildir got created

```
root@mailhost:~# tree -pug /home/mail/mail000/
/home/mail/mail000/
└── [drwx------ 20000    20000    ]  mailbox
    ├── [drwx------ 20000    20000    ]  cur
    ├── [drwx------ 20000    20000    ]  new
    │   └── [-rw------- 20000    20000    ]  1458124575.V902I1b800feM892045.mx0
    └── [drwx------ 20000    20000    ]  tmp
```

TLS and submission service

To avoid sniffing of the data exchanged with postfix, `TLS` needs to be enabled and configured properly. It is recommended to at least use a keylength of 2048 bit for the RSA key, and to provide a valid certificate with a correct *common name* (`CN`), signed by a well-known certificate authority. If you don't want to pay any money, consider Let's Encrypt as a free alternative. It is also advised to generate your own Diffie-Hellman groups with at least 2048 bit due to some weaknesses found with the commonly and widely used Diffie-Hellman groups.

To generate a group with 4096 bit, run

```
root@mailhost:~# openssl dhparam 4096 > /etc/postfix/dhparam/dh4096.pem
```

Generating a new group once in a while will increase security.

TLS support needs to be configured for the server side (`smtpd_tls_*`) of postfix, to encrypt and authenticate the connection when receiving mail as well as the client side (`smtp_tls_*`) for securing delivery to external MTAs.

main.cf

```
smtpd_tls_cert_file = /etc/postfix/tls/server.crt
smtpd_tls_key_file = /etc/postfix/tls/server.key
smtpd_tls_loglevel = 1
smtpd_tls_received_header = yes
smtpd_tls_security_level = may
smtpd_tls_auth_only = yes
smtpd_tls_mandatory_protocols = !SSLv2, !SSLv3
#tls_preempt_cipherlist = yes
tls_disable_workarounds = 0xFFFFFFFFFFFFFFFF
```

```
smtpd_tls_mandatory_ciphers = high
smtpd_tls_exclude_ciphers = aNULL, eNULL, EXPORT, DES, RC4, MD5, PSK, aECDH, EDH-DSS-DES-
smtpd_tls_dh1024_param_file = /etc/postfix/dhparam/dh4096.pem
smtpd_tls_eecdh_grade = ultra
```

This configuration for the `smtpd` side is quite secure but restrctive and might not work with all MTAs.

`smtpd_tls_mandatory_ciphers` sets the ciphersuite allowed to be used and is set to `high`. To see what ciphers belong to this group run `postconf tls_high_cipherlist`.

`smtpd_tls_exclude_ciphers` contains a list of insecure ciphers and is recommended by the research group mentioned above.

`tls_preempt_cipherlist` enables server cipher preferences when set to `yes`. This means that instead of the client, postfix will select the cipher used for communication. This might cause some interoperability issues, see the postfix documentation for more information.

`tls_disable_workarounds` set to `0xFFFFFFFFFFFFFFFF` disables all OpenSSL bug work-arounds on a 64 bit system since they can create unexpected security issues.

`smtpd_tls_security_level` is set to `may` which means that encryption is optional. Changing this to `encrypt` will enforce the use of `TLS` but has the side effect that MTAs without TLS capability won't be able to deliver mail to your server.

`smtpd_tls_auth_only` enforces encryption during authentication independently from `smtpd_tls_security_level`.

`smtpd_tls_dh1024_param_file` sets the path to the Diffie-Hellman group and despite the name can (and should!) be larger than 1024 bit.

`smtpd_tls_eecdh_grade` selects the curves used by postfix for ephemeral ECDH key exchange. `ultra` selects the curve set in `tls_eecdh_ultra_curve` (see `postconf tls_eecdh_ultra_curve`) and is the strongest setting available, but needs approximately twice the computational cost of the `strong` setting.

To debug TLS connections `smtpd_tls_loglevel` is set to at least `1`.

`smtpd_tls_received_header` set to `yes` will add information about the ciphers used during transfer to the message headers.

main.cf

```
smtp_tls_security_level = verify
smtp_tls_CApath = /etc/ssl/certs
smtp_tls_loglevel = $smtpd_tls_loglevel
smtp_tls_mandatory_protocols = $smtpd_tls_mandatory_protocols
smtp_tls_mandatory_ciphers = $smtpd_tls_mandatory_ciphers
smtp_tls_exclude_ciphers = $smtpd_tls_exclude_ciphers
```

For the `smtp` side, `smtp_tls_loglevel`, `smtp_tls_mandatory_protocols`, `smtp_tls_mandatory_ciphers` and `smtp_tls_exclude_ciphers` are set to the same values as their `smtpd` counterparts. `smtp_tls_security_level` set to `verify` enforces encryption when delivering mail. The connection to the external MTA will fail if TLS is not supported or the certificate of the remote site can't be verified. To verify the certificate the CA certificates in `smtp_tls_CApath` are used. This can cause mail to be deferred if the peer certificate was self signed or expired. To handle this issue, either use the less secure `encrypt` option or set `delay_warning_time` to a suitable value to get a notification if your mail was deferred.

If you set `smtpd_tls_security_level` to `may` or `tls_preempt_cipherlist` to `no`, it is good practice to provide a second instance of `smtpd` listening on port `587` with `smtpd_tls_security_level` set to `encrypt` and `tls_preempt_cipherlist` set to `yes`. This instance should then be used by users to submit mails for delivery (outgoing SMTP server).

To enable the submission service edit `/etc/postfix/master.cf` and uncomment the relevant lines.

master.cf

```
...
submission inet n       -       -       -       -       smtpd
  -o syslog_name=postfix/submission
  -o smtpd_tls_security_level=encrypt
  -o tls_preempt_cipherlist=yes
```

Reload postfix, and check `/var/log/mail.*` for any errors. Also check if the submission service is listening on port 587.

```
root@mailhost:~# systemctl reload postfix
root@mailhost:~# netstat -pltn | grep master
```

Dovecot

While Postfix serves as an `smtp` server and relay, `Dovecot` will serve as `IMAP` server to retrieve the messages stored on the mail host. Dovecot also includes a SASL implementation which can be used by postfix to authenticate users.

Dovecot packages can be installed from the Debian repository.

```
root@mailhost:~# apt-get install dovecot-core dovecot-imapd dovecot-ldap
```

Configuration

The configuration is spread over several files in `/etc/dovecot/` and `/etc/dovecot/conf.d` and contains some basic configuration blocks, which can be included by uncommenting them. `doveconf -n` will print the final configuraton build up out of all snippets.

Maildir location

The location of the Maildir is set in `/etc/dovecot/conf.d/10-mail.conf`. The user's homedirectory will be read from LDAP, and '/mailbox' will be appended with the following configuration.

```
maildir:~/mailbox
```

IMAPS

To only activate `IMAPS` support on port 993 and deactivate potentially insecure login attempts to the normal IMAP port 143 the `port` of the `imap` listener must be set to `0` and the lines for `IMAPS` needs to be uncommented in `/etc/dovecot/conf.d/10-master.conf`.

10-master.conf

```
service imap-login {
    inet_listener imap {
        port = 0
    }
    inet_listener imaps {
        port = 993
        ssl = yes
    }

    service_count = 1
    process_min_avail = 1
}
```

LDAP backend

Dovecot can use LDAP as a password database for authentication as well as a user database for information like the maildir location, and the `UID` and `GID` of the user. An overview is given in the Dovecot wiki.

To activate LDAP as a password and user database, enable it in `/etc/dovecot/conf.d/10-auth.conf` and disable `auth-system.conf.ext`.

10-auth.conf

```
#!include auth-system.conf.ext
!include auth-ldap.conf.ext
```

`/etc/dovecot/conf.d/auth-ldap.conf.ext` should contain the declaration for `passdb` and `userdb`.

auth-ldap.conf.ext

```
passdb {
        driver = ldap
        args = /etc/dovecot/dovecot-ldap.conf.ext
}
userdb {
        driver = ldap
        args = /etc/dovecot/dovecot-ldap.conf.ext
}
```

The file `/etc/dovecot/dovecot-ldap.conf.ext` is used by both, `passdb` and `userdb` to configure the connection parameters to the LDAP directory.

dovecot-ldap.conf.ext

```
uris = ldap://ldap.example.com
dn = cn=mailAccountReader,ou=Manager,dc=example,dc=com
dnpass = <dn bind password>
tls = yes
tls_ca_cert_file = /etc/ldap/tls/CA.pem
tls_require_cert = hard
debug_level = 0
auth_bind = yes
auth_bind_userdn = uid=%u,ou=Mail,dc=example,dc=com
ldap_version = 3
```

```
base = ou=Mail,dc=example,dc=com
scope = subtree
user_attrs = homeDirectory=home,uidNumber=uid,gidNumber=gid
user_filter = (&(objectClass=posixAccount)(uid=%u))
```

This configuration `auth_bind = yes` tries to bind to OpenLDAP with the DN of the authenticating IMAP user instead of checking the password directly. The advantage of this configuration is that the password stored in the directory does not need to be readable by Dovecot. When the `auth_bind_userdn` template is defined, `pass_attr` can be omitted. `user_filter` sets the filter to find the LDAP entry using the login username. `user_attrs` maps the LDAP attributes to Dovecot's internal attributes. When retrieving the user information, Dovecot connects to the directory with the credentials set with the `dn` and `dnpass` directives. The other parameters are used to connect to OpenLDAP with TLS and are the same as in the postfix configuration.

TLS

TLS can be configured in `/etc/dovecot/conf.d/10-ssl.conf`. Either reuse the key and certificate used for postfix if the hostname for the IMAP server will be the same, or generate a new key and certificate for Dovecot. The Diffie-Hellman group is generated and managed automatically by Dovecot, the size can be set with `ssl_dh_parameters_length` and should at least be set to 2048 bit. The ciphersuite was taken from weakdh.org

10-ssl.conf

```
ssl = required
ssl_cert = </etc/dovecot/tls/server.crt
ssl_key = </etc/dovecot/tls/server.key
ssl_dh_parameters_length = 4096
ssl_protocols = !SSLv2 !SSLv3
ssl_cipher_list = ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES
ssl_prefer_server_ciphers = yes
verbose_ssl = yes
```

SASL Authentication

Dovecot can be configured to provide a SASL interface for postfix so that authentication for postfix can be done through Dovecot, which then uses LDAP as a backend.

To activate the SASL interface enable the `auth` service in `/etc/dovecot/conf.d/10-master.conf`.

10-master.conf

```
service auth {
    ...
    unix_listener /var/spool/postfix/private/auth {
        mode = 0660
        user = postfix
        group = postfix
    }
}
```

This will create a unix socket in `/var/spool/postfix/private/auth` owned and only accessible by the postfix user.

To activate SASL authentication in postfix, enable it in `/etc/postfix/main.cf`.

main.cf

```
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_auth_enable = yes
broken_sasl_auth_clients = yes
```

To allow relaying mail from authenticated users to remote destinations, add `permit_sasl_authenticated` to `smtpd_relay_restriction`. And to allow only `FROM` addresses that belong to the authenticated user, set `reject_sender_login_mismatch` in `smtpd_recipient_restrictions`. The allowed addresses are looked up from the map defined in `smtpd_sender_login_maps`.

main.cf

```
smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_desti
smtpd_recipient_restrictions = reject_sender_login_mismatch
```

Test authentication

Postfix authentication can be tested easily from the commandline. First connect to the server on port 25 or port 587.

```
you@workspace:~$ openssl s_client -starttls smtp -connect mail.example.com:25
```

You will get a lot of useful information about the TLS connection established. The next step is to introduce yourself with the `EHLO` command.

```
EHLO foo.example.com
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-AUTH PLAIN
250-AUTH=PLAIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
```

The `250-AUTH` line indicates that authentication is enabled. To test authentication generate a base64 encoded string containing the username and the password prefixed by NULL bytes.

```
you@workstation:~$ echo -ne "\0username\0password" | base64
```

Then perform the authentication, send the string to the server.

```
AUTH PLAIN AHVzZXJuYW1lAHBhc3N3b3Jk
235 2.7.0 Authentication successful
```

You will either get `235` or `535` depending on if authentication was successful. If authentication does not work as expected, see `/var/log/mail.*` for errors. If this does not help, set `debug_level = -1` in `/etc/dovecot/dovecot-ldap.conf.ext` and check the logs again when testing the authentication.

Links

- Postfix Documentation
- Postfix Virtual Domain Hosting Howto
- http://wiki.dovecot.org/HowTo/PostfixAndDovecotSASL
- https://easyengine.io/tutorials/mail/server/testing/smtp/
- https://github.com/credativ/postfix-ldap-schema
- http://www.openldap.org/faq/data/cache/1442.html
- http://www.postfix.org/LDAP_README.html#example_virtual
- http://wiki2.dovecot.org/AuthDatabase/LDAP/Userdb
- http://wiki2.dovecot.org/SSL/DovecotConfiguration
- http://www.postfix.org/SASL_README.html#smtpd_sasl_security_options