Archive Mail Server using Dovecot

We have run across this a few times, and thought it might be good to document. A client uses some service which severely limits the amount of e-mail which can be retained. I have seen anywhere from 2G to 10G recently (2016-2018). For some clients, it is a requirement to save e-mail for years, perhaps decades. One common example is the Legal field, where something you did 10 years ago can end up in court.

Additionally, storing a large number of e-mails on a server can severely impact performance, especially when using e-mail clients which continuously index the Inbox (like Microsoft Outlook) and users who do not store mail in separate folders. Generally speaking, 1000 e-mails per folder is the limit for efficiency except in mail servers which index and store internally like Zimbra (https://www.zimbra.com/) or Microsoft Exchange.

Many e-mail clients allow archival of e-mail, but they store the information locally, on your workstation, where it is subject to hardware failure, theft, or natural disaster. Additionally, Microsoft Outlook stores all of its e-mail in one huge file, so backups require copying one huge file each time instead of just looking for new/changed files and copying them. Simply opening Outlook can cause the file to be modified, even if you do not do anything with it.

A much better solution in many cases is to create an IMAP store specifically for archival purposes. In the last cases we had involving actual e-mail limits, the clients were connecting to an Exchange server and already had an internal Unix file server which had an automated/monitored off site backup set up, so it was straight forward to set up an IMAP store for archival purposes.

This article covers building a Dovecot IMAP server on Linux, manually setting up the users (and space for them). If you want a pretty GUI (actually WebUI) you might look at installing ISPConfig on a new machine or virtual, but we'll cover doing everything manually here. It is mainly taken from the article at https://wiki2.dovecot.org/HowTo/SimpleVirtualInstall.

NOTE: ISPconfig has a handy thing in their webui that allows you to disable POP and SMTP for an account. I did not look into how this worked, but it is very handy to ensure you don't accidentally send from your archive account. I'm guessing it will work with the below procedure since we are A) only installing IMAP and B) using a separate credentials file for Dovecot than what an SMTP server would use.

Setup and Install Dovecot Server

This is pretty straight forward; allow the operating systems package manager to install Dovecot, then override the default configuration file.

Install Dovecot

```
apt -y install dovecot-core dovecot-imapd # devuan/debian
yum install dovecot # CentOS
```

Create a user and store for the e-mail

We should use a different user/group for this and all mail will be owned by that user/group. Additionally, we don't want a login, so we'll set the shell to /bin/false. We'll also tell the adduser script to not create the home directory (we'll create it ourselves),

Message store (ie, home directory) can be anyplace. I'm going to set it up in /srv/vmail. This will be the head of a tree of subdirectories for individual users. Note, I use useradd (vs Debian's adduser) for simplicity.

```
useradd --home-dir /srv/vmail --no-create-home --shell /bin/false --user-
group --comment 'Used for vmail only' vmail
mkdir -p /srv/vmail
chmod 755 /srv/vmail
chown vmail:vmail /srv/vmail
```

Modifying dovecot

Create a configuration file for dovecot. I generally back up the original, then create a completely new file, so:

```
mv /etc/dovecot/dovecot.conf /etc/dovecot/dovecot.conf.original
edit /etc/dovecot/dovecot.conf
```

dovecot.conf

```
protocols = imap

# It's nice to have separate log files for Dovecot. You could do this
# by changing syslog configuration also, but this is easier.
log_path = /var/log/dovecot.log
info_log_path = /var/log/dovecot-info.log

# Disable SSL for now.
ssl = no
disable_plaintext_auth = no

# We're using Maildir format
mail_location = maildir:~/Maildir

# Authentication configuration:
auth_verbose = yes
auth_mechanisms = plain
passdb {
  driver = passwd-file
  args = /srv/vmail/passwd
}
userdb {
```

```
  driver = static
  args = uid=vmail gid=vmail home=/srv/vmail/%u
}
```

Save this in /etc/dovecot, then set it with the correct ownership

```
chown root:root /etc/dovecot/dovecot.conf
chmod 644 /etc/dovecot/dovecot.conf
```

Add a user

First, let's create the file /srv/vmail/passwd and set its permissions:

```
touch /srv/vmail/passwd
chown vmail:vmail /srv/vmail/passwd
chmod 644 /srv/vmail/passwd
```

To add a user, simply make an entry as follows. The extra colons are to conform to the dovecot format, but they can be ignored if you like; the format can be simply username:password. The difference is, if you want to add additional flags later (like quotas), you'll want the extra colons.

```
test:{PLAIN}test::::::
```

is the same as

```
test:{PLAIN}test
```

The text inside the curly braces tells what encryption is used on the password (which immediately follows, no space or anything). Obviously, you don't want plaintext in most situations. `doveadm` has a pw function which will calculate the hash for you and it gives you the correct format for the entire passwd file format.

Here's an example from the dovecot article, where '> ' shows you are typing something at the prompt:

```
> doveadm pw -s ssha256
Enter new password: foo
Retype new password: foo
{SSHA256}ZpgszeowIcHdoxe3BNqvUTtPxFd6fMsyQxEWyY0Qlobaacjk
>
```

We could change the entry for the test user above using the line emitted by doveadm

```
test:{SSHA256}ZpgszeowIcHdoxe3BNqvUTtPxFd6fMsyQxEWyY0Qlobaacjk::::::
```

which would give us greater security.

If you want, here is a little utility written in Perl that will do all of it for you. It takes a username and a

password, then either updates or adds that information to the password file. It is NOT very friendly, and could use some cleanup, and the username/password are left in your history file, so it is insecure. Call it with

```
./updatePasswd username 'password'
```

updatePasswd

```
#! /usr/bin/env perl

# WARNING: This is insecure as it will leave the users password in the
# bash history file

use strict;
use warnings;

my $pwfile = '/srv/vmail/passwd'; # location of the password file
my $user = shift;
my $password = shift;

die "Usage: $0 username password\n" unless $user && $password;

my $found = 0; # determines if user already exists

# call doveadm to get the hash
my $key = `/usr/bin/doveadm pw -s ssha256 -u '$user' -p '$password'`;

# read the password file
open PW,"<$pwfile" or die "Could not open the password file: $!\n";
my @data = <PW>;
close PW;

# go through it and see if the user already exists
my $newLine = "$user:$key";
for ( my $line = 0; $line < @data; $line++ ) {
   my ($thisUser,$thisPass) = split( ':', $data[$line] );
   if ( $thisUser eq $user ) { # yes, so replace the line and mark
found
      $data[$line] = $newLine;
      $found = 1;
      last;
   } # if statement
} # for loop
push @data, $newLine unless $found; # we did not find them, so add
chomp @data; # remove all line endings

# write the file back out
open PW,">$pwfile" or die "Could not write the password file: $!\n";
print PW join( "\n", @data ) . "\n";
close PW;
```

```
# tell user what we did
print "User $user ";
print $found ? "modified\n" : "added\n";

1;
```

Setting up mail client

All of the email clients I know do not allow a stand alone IMAP server; they want an SMTP server to be associated with it. I just use one of my other SMTP servers for this purpose; you won't be sending mail from this account, but if you accidentally do, you'd at least send from your main mail account.

Note, after this is done, you can move mail into the archive account manually, or through rules.

Following are some of the clients we have run into, and the fixes we made to get them to work.

OS X Mac Mail

Actually, Mac Mail (aka 'mail') does allow you to say “we have no SMTP server for this” but, you have to modify it after the creation of the account.

Do the standard create new account, and it will fail with a verification error. At that point, click the Continue button below. The account will be created, but disabled.

Now, follow these instructions to set up the account correctly.

1. Go to Mail then Preferences
2. Select Server Settings from the popup window
3. Select the Archive account on the left panel
4. Add the username to the account (it “failed” so Mail removed it)
5. Click the Save button which appears at the bottom and wait for verification to complete
6. On the Outgoing server, either select None or select a different account
    1. If you choose *none*, you will get an error message if you accidently try to send from that account
    2. If you choose a different SMTP server, it will automagically use that account anytime you have the archive account selected and try to send a message
7. Click on the SMTP dropdown again and choose Edit SMTP Server List
8. Locate and highlight the SMTP server the wizard created. It should have nothing in the “In use by this account” column
9. Delete the server definition by clicking the minus sign (-)
10. Close all windows to return to main screen.

Thunderbird

Thunderbird changes fairly regularly, so these instructions cover v52.9.1, but they should at least get you close. Thunderbird requires an SMTP server be associated so, unlike OSX Mail, you have to point

the SMTP server to some pre-existing account.

1. File | New Existing Mail Account
2. fill in information and click Continue
3. Select Manual Config
4. Click Advanced Config
5. Click on the account name itself
6. Click on Outgoing Server (SMTP) (bottom of window) and choose a different SMTP server
7. Click 'Manage Identities' (optional)
    1. Click Edit on the default
    2. fill in the information *as if you were sending from other mail server*
    3. Click Ok, then close
8. Go to bottom of account list and select Outgoing Server (SMTP)
9. Locate and select SMTP server that was automatically created
10. Click Remove
11. Click the Ok button.

End result

Once you have added the username/password, the user then sets up their e-mail client to access it. Dovecot will create the correct directory tree on first login, so you don't have to do anything else on that. The user can then create folders and use the archives IMAP store to save e-mail which does not need to be on the main server. Additionally, e-mail is stored in a known location, in IMAP format, so it is very easy to back up (it is just text files).

Tuning

Several things are possible with tuning. If you are using ZFS, setting dedup and compress on will result in very high efficiency as far as disk space is concerned. Or, you can use one of the many scripts available to strip attachments from e-mail messages and stored them (in binary format) in a separate web accessible location for retrieval. Turning mime encoded attachments into standard binary files will result in a space savings of about 25% by itself, and if you use ZFS dedup and/or ext hard links, you can save even more as the fifteen copies of the cute kitty video your friend sent you will only be stored once.

Automation

I have written a script we use for ourselves and our clients which will automate archiving an active e-mail account to the archive you created.

```
svn co http://svn.dailydata.net/svn/sysadmin_scripts/trunk/archiveIMAP
archiveIMAP
```

Installation consists of placing the script someplace, then installing some Perl libraries (see comment in archiveIMAP script, which has Debian installation line). It is released under the GNU Public License, though I'm thinking of changing it to one of the BSD's.

Basically, it opens an IMAP connection to the *source* account, then finds any messages over a predefined relative date, copies or moves them to the archive server. I used that script to connect to a standard IMAP server and to a Microsoft Exchange server (via IMAP) to auto-clean older mail from the active storage, while maintaining the directory structure.

The script can be configured to preserve the directory structure, or modify it as you like. It also has test runs, etc... The configuration file is YAML. The script can be run from the source, the target, or even a third server.

The script could also be set up to remove MIME attachments and store them. A good place to start on that would be in the article http://www.perlmonks.org/bare/?node_id=525036 where they describe how to pull a MIME attachment out and store it as a file. The script could then replace the MIME attachment code in the e-mail with a link to the extracted file.

Errors

Some e-mail, especially older ones or spam, have malformed dates, or dates which can not be processed by the Perl libraries. In this case, you may receive an error similar to

```
Use of uninitialized value $t[4] in addition (+) at ./archiveIMAP line 234,
<GEN3> line 720660.
```

and the e-mail in question will **not** be processed. If you find lines like this in your logs, or if you see some older e-mail not being moved, you will need to move manually or delete them.