

PigeonDeliver and Postfix HOWTO

Carlo Contavalli

ccontavalli@masobit.net

Revision History

Revision 1.0.0 2006/01/24
First document revision

This document describes how to install PigeonDeliver with Postfix, describing and detailing each and every step.

For a complete reference about compilation options and how to install PigeonDeliver, please refer to %%TODO%%.

1. Before starting

This document was written as part of the documentation of the PigeonAir Project to provide help and support to users, system administrators or developers.

While every effort has been made to ensure that the information is accurate at the time of publication, this document may contain errors, omissions, incongruences or wrong technical details. No liability for damages is accepted by the Author/Authors, the publishers or any other organization or person providing the information, arising from any errors or omissions that may appear, however caused.

In case you find an error, you would like to propose better solutions than those discussed in this document or you would like to discuss an idea regarding this document or its content, we would be glad to hear from you and please contact us by writing directly to the author of this document or to the <pigeon-dev at ml.pigeonair.net> mailing list.

1.1. Intended Audience

This document was meant to provide an easy to follow, step by step guide at installing and setting up PigeonDeliver with Postfix on any linux system, even for non-professional users.

For that purpose, many details and technical descriptions were made using an overway simplified model, and often by using improper terms and definitions. The hope was to make the document more intuitive and easier to read and understand, even for a non-technical reader.

Hope that you, as a reader, will understand my good will and will not regret nor blame me for the improper use of some terms and definitions, and for the simplified or even wrong models sometimes described or used just for the sake of creating a more intuitive and simplified environment.

This document will not discuss details regarding configuration files, the structure of the mail system, or the integration with other mail servers or systems.

1.2. Copyright Notice

This document was written by Carlo Contavalli <ccontavalli at masobit.net> and is thus Copyright (C) 2003,2004,2005,2006 Carlo Contavalli.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, no Back-Cover Texts, and no Front-Cover Text.

Any example of program code available in this document should be considered Copyright (C) 2003,2004,2005,2006 Carlo Contavalli, protected by the terms of the GNU General Public License, version 2.00.

You should have received a copy of the GNU General Public License along with this document; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Trademarks are owned by their respective owners.

2. Introduction

Ok, the procedure that will be described in the following sections has been used to compile and install PigeonDeliver and Postfix on a wide range of Linux distributions, including Slackware and Debian, using OpenLDAP and slapd as a backend database.

Trying to outline the installation process, this is what I did:

1. installed OpenLDAP and slapd, and configured
2. downloaded, and configured Postfix

3. downloaded, configured and installed PigeonDeliver
4. installed and configured postfix
5. a few tests :)

Anyway, before starting, you should download the latest version of PigeonDeliver from <http://www.pigeonair.net> and unpack it, with something like:

```
$ cd /usr/src
$ zcat pigeondeliver-x.y.z.tar.gz |tar xvf -
```

Note that in the course of the document, all examples containing commands to be run will have a \$ in front of every command you can run as a normal user, and a # for every command you need to run as root (super user). Lines outputted by those commands will not have any prefix.

Note also that along this document, we will try to setup the mail server for the virtual provider called "foobar.org", which will serve domains and services for all its own customers.

3. Installing OpenLDAP

To install OpenLDAP and slapd, I both tried with the packages provided by my own linux distribution, and by compiling and installing the OpenLDAP .tar.gz available on the internet site.

There is nothing strange to do, any standard version of OpenLDAP should work out of the box with PigeonDeliver. I personally suggest using version > 2.2.26, but not yet 2.3.x (Mon Jan 16 04:03:20 CET 2006), even if 2.3 is considered the stable branch. If you plan to use PigeonAdmin, make also sure the version of OpenLDAP you are using has support for HDB (hdb, hierarchical database).

To install OpenLDAP from the .tar.gz, I just downloaded the sources from <http://www.openldap.org> and compiled and installed them after carefully reading the instructions provided on the internet site.

Before even attempting to compile OpenLDAP, make sure to have BDB, "Berkeley DB", installed on your system (libbdb). Since this library is a very important component of most linux distribution, it is almost always already installed.

On Debian, and on many other distributions, you may need to install the packages "libbdb4.2" and "libbdb4.2-dev".

If you need/want to compile and install libbdb4.2 by yourself, keep in mind that this is a very important component of OpenLDAP, and probably the trickiest to install: all your data will be stored in a BDB database, which means that the reliability, speed and storage limits of your LDAP directory will be completely based on this library. Currently (Thu Jan 26 12:09:38 CET 2006), OpenLDAP gurus suggest to use BDB 4.2.52 with a couple patches to increase reliability and performance. Without some of those patches, OpenLDAP will sometimes hang and not work properly, so, make sure to install them.

Since libbdb4.2 was already installed on my Linux system, to compile and install OpenLDAP 2.2.30 I just had to run:

```
$ zcat openldap-2.2.30.tgz |tar xvf -
$ cd openldap-2.2.30
$ mkdir compile
$ cd compile
$ ../configure --prefix=/usr/local/openldap/ --with-cyrus-sasl=no \
  --enable-hdb --enable-slurpd
$ make depend
$ make
$ make install
```

With, 2.3.x, I used:

```
$ zcat openldap-2.3.x.tgz |tar xvf -
$ cd openldap-2.3.x
$ mkdir compile
$ cd compile
$ ../configure --prefix=/usr/local/ldap-2.3.x/ \
  --with-cyrus-sasl=no
$ make depend
$ make
$ make install
```

IMPORTANT: if you install from the .tar.gz, remember to add the bin and sbin directory of the newly installed OpenLDAP database to your path, with something like:

```
export PATH="/usr/local/openldap/bin:/usr/local/openldap/sbin:$PATH"
```

In all the examples in the document, I will assume that running commands like "slapcat" or "slapadd" will run the binaries you just compiled and installed. If you do not change the PATH, you may end up running the "slapadd" and "slapcat" commands made available by your linux distribution, possibly screwing up any database already installed on your system, and loosing all your data! Watch out, you have been warned!

Make also sure the above line is in your .profile, .bash_profile or whatever in your home directory. For root, I would open the file /root/.profile and add the line:

```
export PATH="/usr/local/openldap/bin:/usr/local/openldap/sbin:$PATH"
```

right at the end of your file. Otherwise, you will have to export the path every time you log in or every time you leave the current shell.

4. Notes about linux distributions

The configuration described in this document will assume you compiled and installed OpenLDAP 2.2.30 manually like shown above.

If you specified a different `--prefix` option, or if you used the packages of your distribution, in all the following examples always remember that:

`/usr/local/openldap/etc/openldap`

is the directory where all configurations are kept. In this directory, you can find the configuration files (`slapd.conf`, `ldap.conf`, ...), the schema files (`directory.schema`) and a couple other interesting files. On most linux distributions, if you used the provided packages, it corresponds to `/etc/ldap/`.

`/usr/local/openldap/libexec/`

is the directory where the `slapd` and `slurpd` daemons have been installed. Usually, it corresponds to `/usr/sbin`.

`/usr/local/openldap/bin/`

is the directory where all command line tools to access a LDAP directory have been installed (`ldapadd`, `ldapsearch`, ...). Usually, it corresponds to `/usr/bin`.

`/usr/local/openldap/sbin/`

is the directory where all the administrative tools have been installed (`slapcat`, `slapadd`, ...). Usually, it corresponds to `/usr/sbin`.

`/usr/local/openldap/var/openldap-data/`

is the directory where your database files are kept. On most linux distributions, it corresponds to `/var/lib/ldap/`, but you can guess it by looking to your `slapd.conf` configuration file and to the "directory" parameter.

So, make sure to change the examples as required by the parameters you provided to the configure scripts or to the default setup of you linux distribution.

5. Configuring OpenLDAP

To configure OpenLDAP, open the `slapd.conf` configuration file. Either `/etc/ldap/slapd.conf` or `/usr/local/openldap/etc/openldap/slapd.conf` or `/usr/local/openldap/etc/ldap-2.3.x/slapd.conf`, depending

on how you opened it. You now need to:

1. configure the hdb backend. OpenLDAP can store data on the hard drive in different "formats". The default "format" is bdb. PigeonDeliver can use both bdb or hdb. If you plan to use PigeonAdmin, however, we strongly suggest you to use HDB.
2. add PigeonDeliver and PigeonAdmin schema files. The LDAP protocol uses "schema" files to understand the "structure" of a database. In order to use PigeonDeliver and PigeonAdmin, you must let OpenLDAP know the structure of the PigeonDeliver database.
3. initialize the OpenLDAP database, in order to insert the basic data needed for PigeonDeliver to start.
4. finally, start slapd.

Before going on with the following steps, *make sure* slapd, the OpenLDAP daemon, is not active and not answering queries, especially if you are using linux distributions that automatically start slapd after installation.

To stop it, try something like:

```
# /etc/init.d/slapd stop
```

or, in alternative, something like:

```
# killall slapd
```

(watch out that the above command will terminate all the slapd instances running on your machine).

5.1. Enabling the HDB Backend

You now need to configure the OpenLDAP backend and database. On my system, I had to add:

```
backend hdb
checkpoint 1024 10

database "hdb"
suffix "dc=foobar,dc=org"
rootdn "cn=admin,dc=foobar,dc=org"
```

to slapd.conf. Instead of "dc=foobar,dc=org", write your own domain name. If your domain name is foo.bar.org, you need to write "dc=foo,dc=bar,dc=org", which is your *basedn* (that will be necessary later on in this document). If you don't have a domain name, you can use something like "o=Your organization name". However, it is much better if you have a domain name. This name is just a "label" used to identify your system. You can write almost anything you like, it has no effect on your emails or PigeonDeliver usage. Make also sure:

- there are no other "database" sections. In case there are, just remove all the corresponding lines.
- if you have compiled openldap with support for dso modules, make sure there two lines like:

```
modulepath    /usr/local/openldap/libexec/openldap
moduleload    back_hdb.la
```

where the directory /usr/local/openldap/libexec/openldap is the directory where the file back_hdb.la can be found.

After the above steps, my slapd.conf file looked something like (note that lines starting with # are just comments):

```
include        /usr/local/openldap/etc/openldap/schema/core.schema
```

```
pidfile        /usr/local/openldap/var/run/slapd.pid
argsfile       /usr/local/openldap/var/run/slapd.args
```

```
# here, before my changes, I add something like:
# database bdb
# suffix "dc=my-domain,dc=com"
# rootdn "cn=Manager,dc=my-domain,dc=com"
# Instead of the above lines, I put the following:
backend hdb
checkpoint 1024 10
```

```
database       hdb
suffix         "dc=foobar,dc=org"
rootdn        "cn=admin,dc=foobar,dc=org"
```

```
# Those lines were already in the configuration file
rootpw        secret
directory    /usr/local/openldap/var/openldap-data
index objectClass eq
```

%%TODO%% if you are going to run a big email service or you are going to have a lot of users, you may want to optimize the database parameters. Please consider adding a "cachesize" parameter and a couple more "indexes". Also remember that, if you have valuable data in your database, every time you change the indexing parameters you should run the "slapindex" command.

5.2. Adding PigeonDeliver schema files

To add the PigeonDeliver schema files:

1. create a directory named "pigeonair", in /usr/local/openldap/etc/openldap/, the directory containing your configuration files.

2. put the pigeondeliver schema in the right place: copy the files in `/usr/src/pigeondeliver-x.y.z/examples/schema/` in `/usr/local/openldap/etc/openldap/pigeonair/`, with something like:

```
# cp -a /usr/src/pigeondeliver-x.y.z/examples/schema \
/usr/local/openldap/etc/openldap/pigeonair
```

3. tell slapd about the new schema files, by adding the following lines in your `slapd.conf` configuration, right after the line "include ../core.schema", file:

```
## pigeonair required schemes
include      /usr/local/openldap/etc/openldap/pigeonair/generic.schema
include      /usr/local/openldap/etc/openldap/pigeonair/domains.schema

## modules schema
include      /usr/local/openldap/etc/openldap/pigeonair/mailStore.schema
include      /usr/local/openldap/etc/openldap/pigeonair/mailForward.schema
include      /usr/local/openldap/etc/openldap/pigeonair/mailHidden.schema
include      /usr/local/openldap/etc/openldap/pigeonair/mailAlias.schema
include      /usr/local/openldap/etc/openldap/pigeonair/mailAntivirus.schema
include      /usr/local/openldap/etc/openldap/pigeonair/mailNewsletter.schema
include      /usr/local/openldap/etc/openldap/pigeonair/mailSanitizer.schema
include      /usr/local/openldap/etc/openldap/pigeonair/mailVacation.schema
include      /usr/local/openldap/etc/openldap/pigeonair/mailAntispam.schema
```

Ok, after adding the schema files, you are now ready to initialize the OpenLDAP database.

5.3. Initializing OpenLDAP database

To initialize the OpenLDAP database, there are two steps to be performed:

1. remove any file that may already be there. If you installed OpenLDAP from the `.tar.gz`, there will be an "example" database, almost useless. If you installed from your distribution packages, you may already have a "good" database. However, most linux distributions use `bdb` as the default backend, so you must recreate it anyway to switch to the new format.
2. create a `DB_CONFIG` file.
3. create the standard OpenLDAP root of the database.
4. create the PigeonDeliver entires, as needed.

5.3.1. Removing the current database

To remove the current database:

- make sure your slapd daemon is stopped, and no other processes are using it. You can either:
`/etc/init.d/slapd stop`
or

```
killall slapd
```

- make a backup of the current database, "just in case...". To backup your DB, run something like:

```
# slapcat > file.backup.ldiff
```

If you get an error like "Command not found", you didn't read the section %%TODO%% carefully. Please read it one more time, and make sure to run the command:

```
export PATH="/usr/local/openldap/bin:/usr/local/openldap/sbin:$PATH"
```

- remove all the files in /var/lib/ldap/, with something like:

```
# cd /var/lib/ldap
# rm -f *
```

5.3.2. Creating a DB_CONFIG file

The DB_CONFIG file is extremely important to OpenLDAP. It is not just a matter of performance: without a properly configured DB_CONFIG file, some versions of slapd and BDB would just hang, crash, or not work properly.

For a good DB_CONFIG file, I suggest you to start with the file /usr/src/pigeondeliver-x.y.z/examples/slapd/DB_CONFIG. Just copy it with something like:

```
# cp /usr/src/pigeondeliver-x.y.z/examples/slapd/DB_CONFIG \
    /usr/local/openldap/var/openldap-data/
```

The only change I made to the provided DB_CONFIG file was to add the DB_LOG_AUTOREMOVE option, with something like:

```
set_flags DB_LOG_AUTOREMOVE
```

You can write the option wherever you like in the file. I just uncommented the corresponding option.

This option tells BDB and OpenLDAP to automatically remove log files as soon as they are no longer needed. By default, BDB leaves those files so you can back them up from a cron job, and in case your hard drive explodes, you can replay them saving all your data. However, there are many different ways to back your OpenLDAP data up, so, you probably do not need to use BDB infrastructures for catastrophic recovery.

In case you do not enable this flag, make sure you sometime either remove or backup all log.* files as indicated by the db4.2_archive or db_archive utility, which you must run with something like:

```
# cd /usr/local/openldap/var/openldap-data/
# db_archive
log.0000000
```

```
log.0000001  
[...]
```

as shown above, this utility will return the list of log files no longer needed, that can safely be removed.

5.3.3. Creating the LDAP Root

Now that you have your OpenLDAP ready to have a database and to answer queries, you need to start to populate the database.

You have two choices here:

- use the script provided with PigeonDeliver
- perform the steps manually

In this document, we will describe both procedures.

With the script provided with PigeonDeliver, you just need to run `pd-ldap-init-root`, which can be found in `pigeondeliver-x.y.z/examples/slapd`, with something like:

```
# cd /usr/src/pigeondeliver-x.y.z/examples/slapd  
# ./pd-ldap-init-root dc=foobar,dc=org
```

The script will then prompt you for a password to be inserted into the database. Insert the password twice, and wait for it to complete. Note that while you type the password, nothing will be displayed on the screen. Note also that `dc=foobar,dc=org` is the basedn, as we wrote it in the `slapd.conf` configuration file (look to `%%TODO%%`).

If the script is succesful, you will see something like:

```
----- inserted root of OpenLDAP database.  
created - basedn: dc=pippo,dc=it, rootdn:  
  cn=admin,dc=pippo,dc=it, password: [youknowit]
```

on the screen (line has been split into multiple lines for readability purposes).

If you don't trust the script to add your data, or if the script fails for some reason, you can run the command:

```
# cd /usr/src/pigeondeliver-x.y.z/examples/slapd  
# ./pd-ldap-init-root -d dc=foobar,dc=org > /tmp/template.ldif
```

this command will save, in the file `template.ldif`, the data that would otherwise be added to your database (the `-d` parameter makes the difference).

If you then want to add that data by yourself, run the command:

```
# slapadd < /tmp/template.ldif
```

The most common error to encounter is due to the script not being able to find the needed commands. If you get an error like:

```
couldn't find an usable 'xxxxxxx'
```

first export and setup the PATH correctly, as indicated in the previous sections:

```
export PATH="/usr/local/openldap/bin:/usr/local/openldap/sbin:$PATH"
```

If you have troubles or you really want to do everything by hand, you can open the file `pippo.it-base.ldif` in the directory `/usr/src/pigeondeliver-x.y.z/examples/ldif`. In this file, you need to change all occurrences of `dc=pippo,dc=it` with your own base dn, watching out to change also `'dc: pippo'` with `'dc: foobar'`.

The final result should look something like:

```
dn: dc=foobar,dc=org
objectClass: top
objectClass: dcObject
objectClass: organization
o: nodomain
dc: foobar
structuralObjectClass: organization
entryUUID: 49e67d38-c9c2-1029-908c-82c0a376e204
creatorsName: cn=anonymous
modifiersName: cn=anonymous
createTimestamp: 20051005080342Z
modifyTimestamp: 20051005080342Z
entryCSN: 20051005080342Z#000001#00#000000

dn: cn=domains,dc=foobar,dc=org
objectClass: organizationalRole
cn: domains
structuralObjectClass: organizationalRole
entryUUID: a4bffc2-c9d0-1029-9a63-cb91ce934d26
creatorsName: cn=admin,dc=foobar,dc=org
createTimestamp: 20051005094627Z
entryCSN: 20051005094627Z#000001#00#000000
modifiersName: cn=admin,dc=foobar,dc=org
```

Once you changed that file, you can load the data into the database, with something like:

```
# slapadd < pippo.it-base.ldiff
```

After adding this file, you also need to create an administrator, by also adding the file: `cn=admin,dc=pippo,dc=it-base.ldiff` in this same directory. Before adding this file, however you need to open it with an editor and make some changes.

As before, you need to change every occurrence of `dc=pippo,dc=it` with `dc=foobar,dc=org`. After changing the dn, you also need to change the password of the administrator, unless you want to leave the default of `'pippo'`. To change the password, run the command:

```
# slappasswd
```

and type your password twice. `slappasswd` will encrypt your password and print it on the screen:

```
{SSHA}AfBntJxBBqCvH2rbYQ2UDDuE+JiS6lzI
```

copy this string and put it in the file, in the `userPassword` field. The final result should look something like:

```
dn: cn=admin,dc=foobar,dc=org
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword: {SSHA}AfBntJxBBqCvH2rbYQ2UDDuE+JiS6lzI
structuralObjectClass: organizationalRole
entryUUID: 49e77bb6-c9c2-1029-908d-82c0a376e204
creatorsName: cn=anonymous
modifiersName: cn=anonymous
createTimestamp: 20051005080342Z
modifyTimestamp: 20051005080342Z
entryCSN: 20051005080342Z#000002#00#000000
```

note that the double colon (`::`) near `userPassword` has been also replaced by a single colon.

To add this file, simply run the command:

```
# slapadd < cn=admin,dc=pippo,dc=it-base.ldiff
```

5.3.4. Starting slapd

Well, if you got up here, now it should be quite easy. Just run something like:

```
# cd /usr/local/openldap/libexec
```

```
# ./slapd  
#
```

If we did everything right, we shouldn't have any message at all, anything else but our prompt back. In the log file, `/var/log/daemon.log` or `/var/log/syslog`, there should be something like:

```
Jan 14 12:45:00 localhost slapd[4639]: @(#) $OpenLDAP: slapd 2.3.18  
(Jan 13 2006 22:21:14) $ ^Iroot@joshua:/usr/src/openldap/servers/slapd
```

Depending on your linux distribution, you may want to start slapd automatically at boot, adding something like:

```
/usr/local/openldap/libexec/slapd
```

to the script executed at boot, and something like:

```
kill `cat /usr/local/openldap/run/slapd.pid`
```

to your shutdown scripts, in order to properly stop slapd before turning off your computer. This is especially important since *every time slapd is not properly stopped*, you need to recover your database.

To verify that the database is properly working, you can run a command like:

```
$ ldapsearch -b "dc=foobar,dc=org" -x \  
-W -D cn=admin,dc=foobar,dc=org
```

which means: "please, search all records (ldapsearch) inserted in dc=foobar,dc=org (-b), since I am the administrator whose record in the database is cn=admin,dc=foobar,dc=org (-D) who will authenticate using simple authentication (-x) and providing a password (-W) as soon as I will be asked to do so". If everything is working, you should now see all the records we just inserted.

6. Installing PigeonDeliver with Postfix

Some Linux distributions, usually Debian based, compile postfix package with a smart patch from La Mont Jones that separates postfix from its own libraries.

If this patch has been used on your system, you can use a simplified procedure to install PigeonDeliver with Postfix. Otherwise, you will need to recompile postfix and compile PigeonDeliver to use the Postfix libraries you just recompiled.

Ok, so, if you find the files `libpostfix-master.so`, `libpostfix-global.so` and `libpostfix-users.so` in `/usr/lib` or in `/lib`, you can follow the procedure indicated to install PigeonDeliver on "Debian systems"

%%TODO%%. Otherwise, or if you are unsure, follow the instructions provided to compile PigeonDeliver on non-Debian systems.

6.1. Non Debian systems

Well, now that the OpenLDAP database is working, we can move on and configure PigeonDeliver with Postfix. Note that at time of writing there is no way to compile PigeonDeliver with postfix without having Postfix sources at hand. First of all, download the latest sources of Postfix from <http://www.postfix.org/>, and unpack them, with:

```
$ cd /usr/src
$ zcat postfix-x.y.z.tar.gz |tar xvf -
```

Now, follow the installation instructions provided with Postfix and PigeonDeliver to prepare the postfix compilation. Something like:

```
$ cd postfix-x.y.z
$ make -f Makefile.init makefiles CCARGS="-fpic -DPIC"
$ make update DIRS='src/util src/global src/master src/postconf'
```

worked for me. Don't forget to add all the other needed options, as indicated by the various README files. As an example, if you want PCRE support, don't forget to compile the Postfix sources with something like:

```
$ cd postfix-x.y.z
$ make makefiles \
  CCARGS="-fpic -DPIC -DHAS_PCRE -I/usr/local/include"
  AUXLIBS="-L/usr/local/lib -lpcre"
$ make update DIRS='src/util src/global src/master src/postconf'
```

Now that postfix is almost completely compiled, you can go on configuring, compiling and installing PigeonDeliver. I run something like:

```
$ cd /usr/src/pigeondeliver-x.y.z
$ mkdir build
$ cd build
$ sources="/usr/src/postfix-x.y.z" \
  sh ../examples/configure/configure.linux.postfix.src
$ make
$ su
# make install
```

Note that if you want to change some configuration parameters you can edit and modify the file `pigeondeliver-x.y.z/examples/configure/configure.linux.postfix.src`. To complete the Postfix installation, with support for PigeonDeliver, run:

```
$ cd /usr/src/postfix-x.y.z
$ make
$ su
# make install
```

as usual. You should now have a working installation of Postfix and PigeonDeliver.

6.2. Debian based systems

On debian systems, you can follow a simpler procedure, and you don't need to compile and install Postfix by yourself.

In order to configure and install PigeonDeliver with postfix, you just need to:

```
# apt-get install postfix postfix-dev
$ cd /usr/src/pigeondeliver-x.y.z
$ mkdir build
$ cd build
$ sh ../examples/configure/configure.linux.postfix.lib
$ make
$ su
# make install
```

without worries. The above may have been used on other distributions as well (like ubuntu, or even non debian based ones). To know if your distribution has separated postfix from its own libraries and if you can use this simplified procedure, you just need to search for `libpostfix-utils.so`, `libpostfix-global.so` and `libpostfix-master.so`. If you have those files on your system, you can probably use this procedure.

7. Configuring Postfix to use PigeonDeliver

Now that Postfix is well installed on your system, you just need to configure it to use PigeonDeliver.

Usually, it is convenient to first configure Postfix to work correctly for local users only, which means, to handle emails for one single domain and to deliver mails to the users in the `passwd` file, "local users" of the machine. Once it works for local users, we can add the PigeonDeliver parameters.

7.1. Configuring Postfix for local users

You can start configuring Postfix for local users only from the file in `/usr/src/pigeondeliver-x.y.z/examples/postfix/`, `main.cf.local-users-only`.

Copy this file in place of the `main.cf` installed by postfix, and open it with your editor of choice.

To our purpose, the most important parameters to change are:

myhostname

if you are a provider, you probably have a domain name. Usually, your server will have both an IP address and an hostname. You must specify the hostname of your server here. In our example, it could be something like "mail.foobar.org".

mydomain

well, if the hostname is mail.foobar.org, the doamin name is "foobar.org". This parameter is mainly used as a default value for other parameters.

mydestination

the `mydestination` parameter specifies for which domains this server should accept mails for. In our case, we want to configure one single domain, and deliver mails to local users. `mydestination` should then set to "foobar.org".

local_recipient_maps

we should set this parameter to "unix:passwd.byname", to tell Postfix to lookup the final users, final recipient of the mails, into the `passwd` file.

The final result could look something like:

```
myhostname = mail.foobar.org
mydomain = foobar.org
myorigin = $mydomain
inet_interfaces = all
mydestination = foobar.org
local_recipient_maps = unix:passwd.byname
unknown_local_recipient_reject_code = 550
mynetworks_style = host
mynetworks = 127.0.0.0/8
relay_domains =

smtpd_banner = $myhostname ESMTP $mail_name
```

I also suggest to configure relay access restrictions and parameters like `smtp_*restrictions` right now, in order to simplify configuration.

7.2. Configuring Postfix to deliver with PigeonDeliver

In order to configure Postfix to deliver mails with PigeonDeliver and to accept mails for all PigeonDeliver users, you need to modify the `master.cf` configuration file, add a couple parameters in the `main.cf` file, and add a new line in the `dynamicmaps.cf` file, only on Debian based systems.

7.2.1. Configuring the `master.cf` file

Add the following lines in your `master.cf` file:

```
mailForward unix - - - - - pd-postfix
mailVacation unix - - - - - pd-postfix
mailAntivirus unix - - - - - pd-postfix
mailMaster unix - - - - - pd-postfix
mailStore unix - n n - - pd-postfix
```

Remember that there must be a line for each and every PigeonDeliver service you will want to use.

7.2.2. Configuring the `main.cf` file

Add the following lines in the `main.cf` file:

```
dscm_datatree_ldap_method = simple
dscm_datatree_ldap_binddn = cn=admin,dc=foobar,dc=org
dscm_datatree_ldap_password = pippo
dscm_datatree_ldap_dnbase = cn=domains,dc=foobar,dc=org
dscm_datatree_ldap_maxsize = 200
dscm_datatree_ldap_server = ldap://127.0.0.1/

virtual_mailbox_domains = dscm:domain
virtual_mailbox_maps = dscm:user
virtual_transport = mailMaster
```

Make sure to specify the password you inserted when creating the database.

7.2.3. Configuring the `dynamicmaps.cf` file

If your version of postfix uses a `dynamicmaps.cf` file, just add the following line:

```
dscm /usr/lib/postfix/dict_dscm.so dict_dscm_open
```

7.2.4. The final result

The final result should look something like below.

main.cf:

```
myhostname = mail.foobar.org
mydomain = foobar.org

dscm_datatree_ldap_method = simple
dscm_datatree_ldap_binddn = cn=admin,dc=foobar,dc=org
dscm_datatree_ldap_password = pippo
dscm_datatree_ldap_dnbase = cn=domains,dc=foobar,dc=org
dscm_datatree_ldap_maxsize = 200
dscm_datatree_ldap_server = ldap://127.0.0.1/

mailAntivirus_policy = drop
myorigin = $mydomain
inet_interfaces = all
mydestination = foobar.org

virtual_mailbox_domains = dscm:domain
virtual_mailbox_maps = dscm:user
virtual_transport = mailMaster

local_recipient_maps = unix:passwd.byname
unknown_local_recipient_reject_code = 550
mynetworks_style = host
mynetworks = 127.0.0.0/8
relay_domains =
smtpd_banner = $myhostname ESMTPE $mail_name
```

master.cf:

```
smtp      inet  n       -       -       -       -       smtpd
pickup   fifo  n       -       -       60      1       pickup
cleanup  unix  n       -       -       -       0       cleanup
qmgr     fifo  n       -       -       300     1       qmgr
tlsmgr   unix  -       -       -       1000?   1       tlsmgr
rewrite  unix  -       -       -       -       -       trivial-rewrite
bounce   unix  -       -       -       -       0       bounce
defer    unix  -       -       -       -       0       bounce
trace    unix  -       -       -       -       0       bounce
verify   unix  -       -       -       -       1       verify
flush    unix  n       -       -       1000?   0       flush
proxymap unix  -       -       n       -       -       proxymap
relay    unix  -       -       -       -       -       smtp
-o fallback_relay=
showq    unix  n       -       -       -       -       showq
```

```
error      unix  -      -      -      -      -      error
discard   unix  -      -      -      -      -      discard
local     unix  -      n      n      -      -      local
virtual   unix  -      n      n      -      -      virtual
lmtpl     unix  -      -      -      -      -      lmtpl
anvil     unix  -      -      -      -      1      anvil
scache    unix  - - - - 1 scache

mailForward unix  - - - - - pd-postfix
mailVacation unix  - - - - - pd-postfix
mailAntivirus unix  - - - - - pd-postfix
mailMaster unix  - - - - - pd-postfix
mailStore unix  - n n - - pd-postfix
```

dynamicmaps.cf:

```
tcp /usr/lib/postfix/dict_tcp.so dict_tcp_open
dscm /usr/lib/postfix/dict_dscm.so dict_dscm_open
```

8. Configuring PigeonDeliver and PigeonDeliver modules

Depending on the modules that will be used by your users, you may need to do some tweaking on PigeonDeliver configuration variables.

8.1. Configuring the mailStore module

By default, the mailStore module will save your users mails in /dscm/data. Just create that folder, with something like:

```
# mkdir /dscm
# mkdir /dscm/data
```

and you should be ok. If you want to change the location of this directory, you can add:

```
mailStore_base = /home/dscm
```

to your main.cf file to set the home directory to /home/dscm.

By default, the mailStore module uses the dscmtree delivery agent with maildir support. This means that under /dscm/data/hostname/ will be created 256 folders, from 00 to FF, where each folder will hold some of you users maildirectories, and each user will be assigned a random uid in the range 1000 - 500000.

You must make sure in the passwd file there are no users using any of those uid, otherwise they will be granted the same privileges.

To change this range, you must use the:

```
mailStore_minuid = 10000
mailStore_maxuid = 20000
```

parameters in your main.cf. With the example above, we have the ability to handle up to 10000 users. However, keep in mind that since hashing techniques as used by dscmtree to assign uids, when the range is almost full the efficiency in adding new users will be much lower, and in case the range completely fills, you won't be able to add new users at all.

So, your range should be about twice the number of users you expect to have, and no less than 1000.

By default, user emails will be stored using the maildir format.

8.2. Configuring the mailAntivirus module

The mailAntivirus module requires clamav or some other antivirus to be installed. At time of writing, the only free antivirus supported is "clamav".

By default, the mailAntivirus module will thus look for a clamd to be available. To allow the mailAntivirus module to find it, you must make the clamd socket available to this module. Just open your clamd.conf file, usually in /etc/clamav/, and change the LocalSocket line into something like:

```
LocalSocket /var/spool/postfix/clamdctl
```

and make sure to have the parameters:

```
ScanMail
ScanArchive
```

in your configuration file. You can find a complete configuration file examples in /usr/src/pigeondeliver-x.y.z/examples/clamav/.

Obviously, you can also configure the mailAntivirus module not to look for the clamd socket in /var/spool/postfix/clamdctl. However, if you run postfix chrooted, this is quite important, since postfix daemons will not be able to look for files outside that directory.

The most interesting parameters of the mailAntivirus module you could be changing in the main.cf file are:

mailAntivirus_socket

to use a different path than `/var/run/clamav/clamd.ctl` or `/clamd.ctl`. With this parameter, you can specify a list of paths that will be tried, until one succeeds, in order to connect clamd. Something like:

```
mailAntivirus_socket = /var/spool/postfix/clamd.ctl  
                    /clamd.ctl /var/run/clamav/clamd.ctl
```

note that `"/clamd.ctl"` is extremely important if you run postfix chrooted.

mailAntivirus_session, mailAntivirus_workaround

if `mailAntivirus_session` is enabled, the `mailAntivirus` module will strongly reduce the number of connections opened with clamd. However, every connection opened will cost about 1 second, unless `mailAntivirus_workaround` is disabled.

So, the question is, is it better to avoid the overload of opening and closing connections, and having to wait 1 second every time we need to do it, or is it better to open lot of connections without having to pay the 1 second penalty? I still don't know...

You can disable the 1 second penalty by disabling `mailAntivirus_workaround`. However, even recent versions of clamd, have a wierd bug that will make the connection hang forever under some conditions if this option is not enabled.

By default, `mailAntivirus_session` is enabled as much as `mailAntivirus_workaround`. However, to avoid the 1 second penalty, you can increase the `IdleTimeout` parameter in your `clamd.conf` file. That way, clamd will not close connections forcing the `mailAntivirus` module to open new ones. To do so, set something like:

```
IdleTimeout 180  
in your clamd configuration file.
```

9. Conclusions

You should now be ready to run and test postfix and PigeonDeliver. Just start postfix as usual, and see what happens... take a look to your `mail.log` files, and make sure there are no dangerous messages in there.

Right after configuring PigeonDeliver, you shouldn't see any visible difference in the deliver process for local users. However, as soon as you add a domain with PigeonAdmin or to the `ldaptree`, you will see that postfix will accept their mails without troubles, and pass them over to the PigeonDeliver delivery agent. At first, you will see PigeonDeliver complaining for some problems. Follow the instructions provided and read the manual to know how to solve the various problems.