GmailPostfixFetchmail

Contents

Introduction

How to use Postfix and Fetchmail to access a single Gmail account using an old-fashioned client such as mutt or Emacs GNUS.

If you use Evolution or a similar modern e-mail client, you do not need to use this. Your client has the ability to connect directly to the Gmail POP3 and SMTP services.

- Help with Evolution: UsingGmailWithEvolution.
- Help with Thunderbird: http://mail.google.com/support/bin/answer.py?answer=38343

This setup is intended to be as simple and as close to a standard Ubuntu configuration as possible. This setup does *not* verify the Gmail SMTP server certificate.

Not For Beginners

You should be familiar with:

- How to install packages
- How to edit text configuration files.
- Terms like POP3, SMTP and SSL.

References

- http://prantran.blogspot.com/2007/01/getting-postfix-to-work-on-ubuntu-with.html
- http://souptonuts.sourceforge.net/postfix_tutorial.html
- http://www.postfix.com/TLS_README.html
- http://www.postfix.com/SASL_README.html
- http://www.postfix.com/ADDRESS_REWRITING_README.html
- http://postfix.state-of-mind.de/patrick.koetter/smtpauth/smtp_auth_mailservers.html
- http://postfix.state-of-mind.de/patrick.koetter/smtpauth/postfix_tls_support.html

Packages needed

You will need the postfix and fetchmail packages. See InstallingSoftware for more on installing packages. Postfix will work on Ubuntu as is from apt-get without any compilation necessary

Setting up your Gmail account

You will need to enable POP access for your Gmail account. This is done through the google website. See UsingGmailWithEvolution for more. Gnus queues mail to postfix, postfix forwards to Google. An openssl certificate is made by a CA signing authority signing a request to generate a server-side certificate. Although to encrypt the connection between Gnus and postfix is possible, this is not necessary if mail is being sent from gnus to postix at the local machine. The connection between postfix and google will be encrypted. This is like https but by the smtp protocol which is by emails. So it is not always necessary to use postfix to do this because some A Mail User Agents can do this itselves : a MUA is a client like gnus or evolution. Postfix will be a client when is connects to google, and the variables pertaining by this are beginning as smtp-the-something. Where postfix is the daemon it receives mails, into our case from the localhost, and the variables pertaining by this mode commence as smtpd-something. Don't forget!

A "mail delivery agent" is the back end used to store mails, which can be postfix. A "mail transfer agent" is a server talking SMTP : it receives mail via SMTP, and it can pass it on via SMTP. Postfix is a combination of MTA and MDA.

to send every mail through Google you also need to set option as

relayhost[smtp.gmail.com]:587

Another option is to use a transport map.

transport_maps = hash:/etc/postfix/transport

The easiest is to just use a mail client, and nothing inter-locuting, but we are not doing this by using postfix at all. Postfix may and can be used as a storage mail retrieval of fetchmail exclusively, and let the mail client perform the smtp encryption to google directly. So this is available as an alternative plan, when this one does work. It is very time-consuming, awkward, frustrating, and annoying.

Example username

In all the examples below, I've assumed that the username on the Ubuntu system is `jane`, and that the Gmail username is `jane.doe@gmail.com`, with password `doeadeer`. You obviously need to replace these with your local username, your Gmail username and Gmail password as appropriate.

Configuring Postfix

To setup Postfix, you will need to create 5 files:

- /etc/postfix/main.cf
- /etc/postfix/generic
- /etc/postfix/generic.db
- /etc/postfix/passwd
- /etc/postfix/passwd.db

You will need root access to create and edit these files; see RootSudo for more on gaining root access.

Stop Postfix

It's not necessary to do so, but if you wish to stop Postfix while configuring, run (as root)

```
  /etc/init.d/postfix stop
```

/etc/postfix/main.cf

When you install Postfix you will be prompted to make configurative choices. You can choose "No configuration"; in this case no configuration file will be created, and you can use the contents below. The configuration choices used to create it are listed in the Appendix.

This is the Postfix configuration file /etc/postfix/main.cf:

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete version


# Debian specific:  Specifying a file name will cause the first
# line of that file to be used as the name.  The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_use_tls=yes
smtpd_tls_session_cache_database = btree:${queue_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${queue_directory}/smtp_scache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.

myhostname = localhost
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
mydestination = localhost, localhost.localdomain
mynetworks = 127.0.0.0/8
mailbox_size_limit = 0
recipient_delimiter = +
```

```
inet_interfaces = loopback-only
inet_protocols = all

########################################
##### non debconf entries start here #####

##### client TLS parameters #####
smtp_tls_loglevel=1
smtp_tls_security_level=encrypt
smtp_sasl_auth_enable=yes
smtp_sasl_password_maps=hash:/etc/postfix/sasl/passwd
smtp_sasl_security_options = noanonymous

##### map jane@localhost to jane.doe@gmail.com #####
smtp_generic_maps=hash:/etc/postfix/generic

relayhost=[smtp.gmail.com]:587
```

An explanation of each non-standard line (following the comment "non debconf entries start here") is given in the Appendix.

/etc/postfix/generic and /etc/postfix/generic.db

The generic file tells Postfix how to map local e-mail addresses to Internet addresses when mail is sent via SMTP. Postfix rewrites "From:" headers to make e-mail appear to come from `jane.doe@gmail.com` instead of `jane@localhost`.

The /etc/postfix/generic is a plain text file, and should look as follows:

```
jane@localhost   jane.doe@gmail.com
```

/etc/postfix/generic.db is generated from this using the postmap command:

```
cd /etc/postfix
postmap generic
```

/etc/postfix/sasl/passwd and /etc/postfix/sasl/passwd.db

The passwd file contains your Gmail password. Like /etc/postfix/generic file discussed above, it is a plain text file; it should look as follows:

```
[smtp.gmail.com]:587    jane.doe@gmail.com:doeadeer
```

To create passwd.db, and set ownership and permissions appropriately, run the following commands:

```
cd /etc/postfix/sasl
postmap passwd
chown root.root passwd passwd.db
chmod 600 passwd passwd.db
```

Start or reload Postfix

If you previously stopped Postfix, restart it with

```
  /etc/init.d/postfix start
```

If you didn't stop Postfix, force it to reload its configuration with

```
  postfix reload
```

Testing

Postfix provides a means of testing its address rewriting rules using the sendmail command with the '-bv' option. If the mail would be sent externally (i.e., via smtp.gmail.com), the command will cause sendmail to connect and authenticate to smtp.gmail.com, which makes it a convenient way to test the Postfix setup.

One possibly inconvenient feature of sendmail -bv is that the result is mailed to the user who ran the command; thus, if mail is utterly misconfigured, you will never receive the result. If you suspect this is the case, you can check /var/log/mail.log to see what went wrong, or you can type mail within the same account as the sender.

Alternatively do `echo 'test mail' | mail -s 'testing this' foo@bar.com`

To check that basic delivery works, run the following command as a normal user (replacing "jane", as elsewhere, with your username):

```
sendmail -bv jane
```

You should receive a mail starting with:

```
This is the mail system at host localhost.

Enclosed is the mail delivery report that you requested.

                    The mail system

<jane@localhost> (expanded from <jane>): delivery via local: delivers to mailbox
```

If this didn't work, make sure that Postfix is running.

To check that Postfix can successfully connect to gmail, run

```
sendmail -bv jane.doe@gmail.com
```

You should receive a mail starting with:

```
This is the mail system at host localhost.

Enclosed is the mail delivery report that you requested.

                    The mail system

<jane.doe@gmail.com>: delivery via smtp.gmail.com[66.249.91.109]:587: 250 2.1.5 OK
```

Potential problems with this are discussed in the following section.

Potential Postfix problems

Cannot find password

If you get an error message like this:

```
<jane.doe@gmail.com>: delivery via smtp.gmail.com[66.249.91.109]:587:
    host smtp.gmail.com[66.249.91.109] said: 530 5.5.1 Authentication Required
    c24sm1773006ika (in reply to MAIL FROM command)
```

then Postfix cannot figure out what password to send gmail; make sure that the smtp_sasl_password_maps entry in /etc/postfix/main.cf is correct, that /etc/postfix/sasl/passwd is correct, and that you've created /etc/postfix/sasl/passwd.db.

No mechanism available

If you get an error message like this:

```
   SASL authentication failed; cannot authenticate to server
   smtp.gmail.com[66.249.91.109]: no mechanism available
```

you have probably forgotten the smtp_sasl_security_options line in /etc/postfix/main.cf.

Configuring Fetchmail

The setup presented here configures the system-wide fetchmail service, which is by default always running; for this use /etc/fetchmailrc is the configuration file. If you want to run fetchmail as your normal user you should use ~/.fetchmailrc; that case is not further discussed here.

Unlike the Postfix setup above, the fetchmail configuration presented here *will* verify the Gmail POP3 server's certificate.

Stop the fetchmail service

To stop fetchmail while configuring it, run

```
/etc/init.d/fetchmail stop
```

/etc/fetchmail.rc

The file /etc/fetchmailrc should look as follows:

```
set syslog

set daemon 240

poll pop.gmail.com
  with nodns,
  with protocol POP3
  user "jane.doe@gmail.com" there is jane here,
  with password doeadeer,
  with ssl, sslcertck;
```

A detailed explanation is given in the appendix, though fetchmail's configuration language hopefully makes it clear.

Since this file contains your Gmail password, you may wish to give it restrictive read permission:

```
chmod 600 /etc/fetchmailrc
```

Testing

To test your configuration, run fetchmail as below; this should be run as root, since it reads /etc/fetchmailrc.

```
fetchmail -v -d0 -f /etc/fetchmailrc
```

Take a look at /var/log/mail.log (e.g., using `less /var/log/mail.log`) to see that the connection was successful.

Restart fetchmail

Once your configuration is working, you can restart fetchmail with

```
/etc/init.d/fetchmail start
```

Appendix

Debconf choices for main.cf above

For the record, the main.cf above was created with

> dpkg-reconfigure postfix

with the following selections:

```
General type of configuration: Satellite system
Mail for root: <blank> (default)
Mail name: localhost (default)
SMTP relay host: <blank> (default is smtp.localdomain)
Other destinations to accept mail for: localhost, localhost.localdomain, localhost (default)
Synchronous updates: no (default)
Local networks: 127.0.0.0/8 (default)
Mailbox size limit: 0 (default)
Local address extension character: + (default)
Internet protocols to use: all (default)
```

Explanation of /etc/postfix/main.cf

Only the non-debconf lines are explained. For much more, run `man 5 postconf` or visit http://www.postfix.com/documentation.html.

```
smtp_tls_loglevel=1
```

Basic logging of connections to smtp.gmail.com.

```
smtp_tls_security_level=encrypt
```

Require an encrypted TLS connection to smtp.gmail.com. It would be preferable to use the verify level.

```
smtp_sasl_auth_enable=yes
```

Enable SMTP authentication.

```
smtp_sasl_password_maps=hash:/etc/postfix/sasl/passwd
```

Where the SMTP authentication data is to be found.

```
smtp_sasl_security_options = noanonymous
```

This one is a bit obscure: by specifying noanonymous, one allows plaintext passwords to be sent (I think noplaintext is the next level "up" from noanonymous). Gmail's SMTP server apparently accepts plaintext authentication only.

```
smtp_generic_maps=hash:/etc/postfix/generic
```

Where the generic mapping data is to be found.

```
relayhost=[smtp.gmail.com]:587
```

Address and port number for SMTP connections. Putting the hostname in square brackets means it is interpreted as a hostname, rather than a mail name (as I understand it, Postfix uses "normal" DNS records rather than MX records when square brackets are used).

Explanation of /etc/fetchmailrc

Run `man fetchmail` for details. Fetchmail's configuration language has the interesting property of ignoring some words (like "with") and punctuation (like the comma and semicolon).

```
set syslog
```

Log messages to syslog; fetchmail messages will appear in /var/log/mail.log.

```
set daemon 240
```

Check for mail every 240 seconds.

```
poll pop.gmail.com
```

Each account entry starts with keyword "poll", followed by the server hostname.

```
  with nodns,
```

This is probably unnecessary.

```
  with protocol POP3
```

Connect to pop.gmail.com mail using the POP3 protocol.

```
  user "jane.doe@gmail.com" there is jane here,
```

Login to the POP3 server with username "jane.doe@gmail.com"; deliver mail to local user "jane".

```
  with password doeadeer,
```

The POP3 password is "doeadeer".

```
  with ssl, sslcertck;
```

Use SSL in communicating to the POP3 server, and verify that the certificate is valid. fetchmail uses the certificates provided by the ca-certificates packages for this.

Verifying the Gmail SMTP server certificate

The configuration above does *not* verify the certificate of the Gmail SMTP server. This would be very easy to do but for Bug 118963

If you need this verification, you can either read reference 2 above, which shows you how to download and install the certificate yourself, or you can do something like this:

```
mkdir /var/spool/postfix/certs
cp -R /etc/ssl/certs/* /var/spool/postfix/certs
mkdir -p /var/spool/postfix/usr/share/ca-certificates
cp -R /usr/share/ca-certificates /var/spool/postfix/usr/share/ca-certificates
```

Then, in main.cf, change the smtp_tls_security_level line and add an smtp_tls_CApath line as follows:

```
smtp_tls_security_level=verify
smtp_tls_CApath=/certs
```

This might need to be redone if you upgrade postfix (e.g., when upgrading Ubuntu).

If Nothing Is Working

If possible, check that you can access the Gmail SMTP and POP3 services with a client like Thunderbird; Google provide complete instructions for setting up Thunderbird here.

You can try port 465 instead of 587 for SMTP.

You can do a check that SMTP connections can be made using stunnel, as follows:

```
stunnel -v 2 -c -n smtp -f -r smtp.gmail.com:587
```

You should see something like this:

```
2007.10.15 22:10:13 LOG5[9230:3083238176]: Using 'smtp.gmail.com.587' as tcpwrap
per service name
2007.10.15 22:10:13 LOG5[9230:3083238176]: stunnel 3.26 on i486-pc-linux-gnu PTH
READ+LIBWRAP with OpenSSL 0.9.8c 05 Sep 2006
220 mx.google.com ESMTP b30sm3913237ika
2007.10.15 22:10:15 LOG5[9230:3083238176]: VERIFY OK: depth=1, /C=ZA/ST=Western
Cape/L=Cape Town/O=Thawte Consulting cc/OU=Certification Services Division/CN=Th
awte Premium Server CA/emailAddress=premium-server@thawte.com
2007.10.15 22:10:15 LOG5[9230:3083238176]: VERIFY OK: depth=0, /C=US/ST=Californ
ia/L=Mountain View/O=Google Inc/CN=smtp.gmail.com
```

Terminate this connection with Ctrl-C.

You can also try testing the POP3 connection, though I had no success with this:

```
stunnel -v 2 -c -n pop3 -f -r pop.gmail.com:995
```

Note that you *cannot* use openssl's s_client to test the SMTP connection; Gmail's SMTP server requires the client to begin communications with HELO (or EHLO), while s_client jumps straight to STARTTLS.

Why is everything still broken?

```
smtpd_recipient_restrictions =
                            permit_mynetworks
                            permit_sasl_authenticated
                            reject_unauth_destination
smtpd_sasl_authenticated_header = yes
```

Now you MUST MUST MUST alter the master.cf file as demonstrated in reference 2. If you do not you might experience looking at a server certificate error in your mailq.

Another, and THE MOST IMPORTANT, thing is that probably your $mydomain, $myhostname, and a load of other things are not concurring with register user accounts on your local computer. The solution is within /usr/share/postfix/main.cf.dist which is a commented, more complete version of a main.cf The clue lies in the fall_transport variable. This permits postfix to work far less rigidly on machines which have spoof domains established within /etc/hosts by the benefit til the operation of nntp servers like leafnode.

```
inet_interfaces = all
mynetworks_style = host
local_recipient_maps =
fallback_transport =
```

If you are still having problems with authorisation and certificate recognition, forget the above about the snakeoil certificate, and follow the most excellent instructions to make your own certificate at reference 7 This page is more useful than this whole venture by postfix itself. If it works it feels rewarding, but it can take one week of effort and worthwhile learning.

CategoryEmail CategoryInternet

GmailPostfixFetchmail (last edited 2011-04-09 00:37:20 by https://login.launchpad.net/+id/FXXJPym @ D9784B24.cm-3-1b.dynamic.ziggo.nl[217.120.75.36]:k.dejong)

- Page History

- Show editing options

- Edit
- Page History
- Subscribe
- Attachments
- More Actions:
- Recent Changes
- Wiki Guide
- rudgergravestein @ LaunchpadHome:rudgergravestein
- Preferences
- Logout