By Falko Timme
Published: 2007-06-14 16:05

Retrieving Emails From Remote Servers With getmail (Debian Etch)

Version 1.0
Author: Falko Timme <ft [at] falkotimme [dot] com>
Last edited 06/04/2007

Getmail is a program for retrieving emails from remote servers; it is very similar to fetchmail, but more flexible. For example, it can be configured to deliver mails directly to a Maildir or mbox mailbox without the need for an MTA such as Postfix, but of course it can also pipe the mails through an MTA if you want. Getmail can use so called filters such as SpamAssassin and ClamAV to scan the mails, and you can even tell getmail to delete mails on the original server only after a certain number of days.

I do not issue any guarantee that this will work for you!

1 Preliminary Note

I have tested getmail on a Debian Etch system with a local user called `falko` who has the local email address `falko@example.com`.

2 Install getmail

In order to install getmail, all we have to do is run

```
apt-get install getmail4
```

as root.

3 Configure getmail

Getmail can be configured through per-user configuration files, and getmail can then be run by that very user. To make getmail run automatically, we can create a cron job for that user.

In this chapter I'm assuming that you're logged in as the user falko, not root!

Getmail is looking for the configuration file *~/.getmail/getmailrc*, so first we have to create the directory *~/.getmail* with permissions of *0700*:

```
mkdir -m 0700 ~/.getmail
```

Then we create the *~/.getmail/getmailrc* configuration file. A *getmailrc* file must at least have one *[retriever]* section and one *[destination]* section.

```
vi ~/.getmail/getmailrc
```

```
[retriever]
type = SimplePOP3Retriever
server = pop.someprovider.tld
username = falko@someprovider.tld
password = secret

[destination]
type = Maildir
path = ~/Maildir/
```

The above example assumes that *falko* has a mail account with the username *falko@someprovider.tld* and the password *secret* at the server *pop.someprovider.tld*, and that he has a Maildir account on this server, with *~/Maildir/* being his Maildir directory.

That's already enough to configure getmail. *falko* could now retrieve emails from the remote server simply by running

```
getmail
```

Of course, *falko* doesn't want to start the retrieval manually every few minutes, so we create a cron job for him. Still as the user *falko*, we run

```
crontab -e
```

and create a cron job like this one (which would start getmail every five minutes):

```
*/5 * * * * /usr/bin/getmail &> /dev/null
```

Now let's assume *falko* doesn't have a Maildir account, but an mbox account on this server (*/var/mail/falko*). All we have to do is modify the *[destination]* section in *~/.getmail/getmailrc*, e.g. like this:

```
vi ~/.getmail/getmailrc
```

```
[retriever]
type = SimplePOP3Retriever
server = pop.someprovider.tld
username = falko@someprovider.tld
password = secret

[destination]
type = Mboxrd
path = /var/mail/falko
```

In the next example we want to pipe the emails that we retrieve from the remote server through an MTA such as Postfix (I assume that Postfix is already

installed and working). Postfix can then take care whether it has to deliver the mails to Maildir or mbox, and it can also **invoke spam and virus scanners, e.g. through amavisd-new**.

```
vi ~/.getmail/getmailrc
```

```
[retriever]
type = SimplePOP3Retriever
server = pop.someprovider.tld
username = falko@someprovider.tld
password = secret

[destination]
type = MDA_external
path = /usr/sbin/sendmail
arguments = ("-bm", "falko@example.com")
unixfrom = true
```

(As you see, we tell getmail that *falko*'s local email address is *falko@example.com*.)

3.1 Options

We can also add an *[options]* section to *~/.getmail/getmailrc* where we can change getmail's default behaviour, e.g. like this:

```
vi ~/.getmail/getmailrc
```

```
[...]

[options]
```

```
verbose = 1
read_all = false
delete = true
message_log_syslog = true
```

The above options tell getmail to print messages about retrieved messages, to retrieve only new messages, to delete messages from the remote server after retrieval, and to log to the syslog.

If you'd like to delete only mails older than ten days, you could change the *[options]* section as follows:

```
vi ~/.getmail/getmailrc
```

```
[...]

[options]
verbose = 1
read_all = false
delete_after = 10
message_log_syslog = true
```

You can learn more about all available options on **http://pyropus.ca/software/getmail/configuration.html#conf-options**.

4 Integrating SpamAssassin Into getmail

If you'd like getmail to invoke SpamAssassin, you can do it as follows:

First we must install SpamAssassin. As root, we run

```
apt-get install spamassassin spamc
```

Then we must configure SpamAssassin. This can be done in the file */etc/mail/spamassassin/local.cf*. A valid file could look like this:

```
vi /etc/mail/spamassassin/local.cf
```

```
rewrite_header Subject *****SPAM*****

required_score 5.0

use_bayes 1
bayes_auto_learn 1
bayes_ignore_header X-Bogosity
bayes_ignore_header X-Spam-Flag
bayes_ignore_header X-Spam-Status
bayes_ignore_header X-getmail-filter-classifier
```

(The comments in */etc/mail/spamassassin/local.cf* will tell you more about the above settings.)

If you enable Bayes (*use_bayes 1*), it is important that you put the line

bayes_ignore_header X-getmail-filter-classifier

into */etc/mail/spamassassin/local.cf* so that Bayes ignores headers added by getmail.

Finally we must enable the SpamAssassin daemon by setting *ENABLED* to *1* in */etc/default/spamassassin*:

```
vi /etc/default/spamassassin
```

```
[...]
```



```
ENABLED=1
[...]
```

Then we start the SpamAssassin daemon:

```
/etc/init.d/spamassassin start
```

Now log in as *falko* again on the command line and open *~/.getmail/getmailrc* and add the following *[filter]* section:

```
vi ~/.getmail/getmailrc
```

```
[...]

[filter]
type = Filter_external
path = /usr/bin/spamc
arguments = ("-s 250000", )
```

(If this is not your first *[filter]* section, you must rename the first *[filter]* section to *[filter-1]*, the second one to *[filter-2]*, and so on.)

That's it. SpamAssassin is now integrated into getmail.

5 Integrating ClamAV

ClamAV is a project that publishes new releases very often, and once a new release is published, old versions will print out warnings. Therefore we must make sure that we install the latest ClamAV release by adding the following line to */etc/apt/sources.list* (as root):

```
vi /etc/apt/sources.list
```

```
[...]
deb http://volatile.debian.org/debian-volatile etch/volatile main contrib non-free
[...]
```

Afterwards we update our packages database:

```
apt-get update
```

(- if you get GPG warnings, you can ignore them -)

and install ClamAV:

```
apt-get install clamav clamav-base clamav-daemon clamav-freshclam
```

Then log in as the user *falko* again and edit *~/.getmail/getmailrc*. If you want getmail to delete virus emails, add the following filter:

```
vi ~/.getmail/getmailrc
```

```
[...]

# Drop infected messages
[filter]
type = Filter_classifier
path = /usr/bin/clamdscan
arguments = ("--stdout", "--no-summary", "-")
```

```
exitcodes_drop = (1, )
```

If you want getmail to deliver infected messages, add the following filter instead:

```
vi ~/.getmail/getmailrc
```

```
[...]

# Keep infected messages
[filter]
type = Filter_classifier
path = /usr/bin/clamddscan
arguments = ("--stdout", "--no-summary", "-")
exitcodes_keep = (0,1)
```

Again, keep in mind what I said about multiple *[filter]* sections in the previous chapter:

If this is not your first *[filter]* section, you must rename the first *[filter]* section to *[filter-1]*, the second one to *[filter-2]*, and so on.

That's it for the ClamAV integration.

6 A Sample Configuration File

This is how my *~/.getmail/getmailrc* file looks in the end:

```
vi ~/.getmail/getmailrc
```

```
[retriever]
type = SimplePOP3Retriever
server = pop.someprovider.tld
username = falko@someprovider.tld
password = secret

[destination]
type = Maildir
path = ~/Maildir/



[options]
verbose = 1
read_all = false
delete = true
message_log_syslog = true

[filter-1]
type = Filter_classifier
path = /usr/bin/clamdscan
arguments = ("--stdout", "--no-summary", "-")
exitcodes_keep = (1, )

[filter-2]
type = Filter_external
path = /usr/bin/spamc
arguments = ("-s 250000", )
```

7 Further Configuration Examples

You can find further configuration examples on **http://pyropus.ca/software/getmail/configuration.html** and in the *getmailrc-examples* file that comes with the getmail *.tar.gz* file that you can download from **http://pyropus.ca/software/getmail**. Here's the contents of that file:

```
#
  # This file contains various examples of configuration sections to use
  # in your getmail rc file. You need one file for each mail account you
  # want to retrieve mail from. These files should be placed in your
  # getmail configuration/data directory (default: $HOME/.getmail/).
  # If you only need one rc file, name it getmailrc in that directory,
  # and you won't need to supply any commandline options to run getmail.
#

#
  # Example 1: simplest case of retrieving mail from one POP3 server and
  # storing all messages in a maildir.
  #

[retriever]
  type = SimplePOP3Retriever
  server = pop.example.net
  username = jeff.plotzky
  password = mailpassword

[destination]
  type = Maildir
  path = ~jeffp/Maildir/

#
  # Example 2: same as (1), but operate quietly, delete messages from
  # the server after retrieving them, and log getmail's actions to a file.
  #

[options]
  verbose = 0
  delete = true
  message_log = ~/.getmail/log
```

```
[retriever]
  type = SimplePOP3Retriever
  server = pop.example.net
  username = jeff.plotzky
  password = mailpassword

[destination]
  type = Maildir
  path = ~jeffp/Maildir/

#
  # Example 3: same as (1), but the mail account is accessed via IMAP4 instead
  # of POP3.
  #

[retriever]
  type = SimpleIMAPRetriever
  server = mail.example.net
  username = jeff.plotzky
  password = mailpassword

[destination]
  type = Maildir
  path = ~jeffp/Maildir/

#
  # Example 4: same as (3), but retrieve mail from the INBOX, INBOX.spam, and
  # mailing-lists.getmail-users mail folders.
  #

[retriever]
  type = SimpleIMAPRetriever
  server = mail.example.net
```

```
  username = jeff.plotzky
  password = mailpassword
  mailboxes = ("INBOX", "INBOX.spam", "mailing-lists.getmail-users")

[destination]
  type = Maildir
  path = ~jeffp/Maildir/

#
  # Example 5: same as (3), but move messages to the mail folder "sent-mail"
  # after retrieving them. Note that you do this by setting delete and
  # move_on_delete options.
  #

[options]
  delete = true

[retriever]
  type = SimpleIMAPRetriever
  server = mail.example.net
  username = jeff.plotzky
  password = mailpassword
  move_on_delete = sent-mail

[destination]
  type = Maildir
  path = ~jeffp/Maildir/

#
  # Example 6: same as (1), but deliver the messages to an mboxrd-format mbox
  # file as user "jeffp".
  #
```

```
[retriever]
  type = SimplePOP3Retriever
  server = pop.example.net
  username = jeff.plotzky
  password = mailpassword

[destination]
  type = Mboxrd
  path = ~jeffp/Mail/inbox
  user = jeffp

#
  # Example 7: same as (1), but deliver the messages through an external MDA
  # which takes several arguments.
  #

[retriever]
  type = SimplePOP3Retriever
  server = pop.example.net
  username = jeff.plotzky
  password = mailpassword

[destination]
  type = MDA_external
  path = /usr/local/bin/my-mda
  arguments = ("--message-from-stdin", "--scan-message", "--to-maildir",
"~jeffp/Maildir/")

#
  # Example 8: retrieve mail from a corporate POP3-SSL domain mailbox,
  # sort messages for several local users and deliver to maildirs in their
  # home directories (except Sam, who likes mbox files, and Christina, who
  # uses procmail for further sorting), and deliver all other mail to
```

```
  # Joe, who serves as postmaster for the company. Sam also needs
  # to receive mail for "sam1", "sam23", etc, so we use a regular expression
  # matching "sam" plus zero or more decimal digits.
  #

[retriever]
  type = MultidropPOP3SSLRetriever
  server = pop.example.net
  username = companylogin
  password = mailpassword
  # Our domain mailbox mailhost records the envelope recipient address in a
  # new Delivered-To: header field at the top of the message.
  envelope_recipient = delivered-to:1

[destination]
  type = MultiSorter
  default = [postmaster]
  locals = (
  ("jeffk@company.example.net", "[jeff]"),
  ("martinh@company.example.net", "[martin]"),
  (r"samD*@company.example.net", "[sam]"),
  ("c.fellowes@company.example.net", "[christina-procmail]")
  )

[postmaster]
  type = Maildir
  path = ~joe/Mail/postmaster/
  user = joe

[jeff]
  type = Maildir
  path = ~jeffp/Maildir/
  user = jeffp
```

```
[martin]
  type = Maildir
  path = ~martinh/Maildir/
  user = martinh

[sam]
  type = Mboxrd
  path = ~sam/Mail/inbox
  user = sam

[christina-procmail]
  type = MDA_external
  path = /usr/local/bin/procmail
  # procmail requires either that the message starts with an mboxrd-style
  # "From " line (which getmail can generate by setting "unixfrom" to True), or
  # that the -f option is provided as below.
  arguments = ("-f", "%(sender)", "-m", "/home/christina/.procmailrc")
  user = christina

#
  # Example 9: same as (3), but use SpamAssassin to filter out spam,
  # and ClamAV to filter out MS worms.
  #

[retriever]
  type = SimpleIMAPRetriever
  server = mail.example.net
  username = jeff.plotzky
  password = mailpassword

[filter-1]
  type = Filter_external
  path = /usr/local/bin/spamc
```

```
[filter-2]
  type = Filter_classifier
  path = /usr/local/bin/clamscan
  arguments = ("--stdout", "--no-summary",
"--mbox", "--infected", "-")
  exitcodes_drop = (1,)

[destination]
  type = Maildir
  path = ~jeffp/Maildir/

#
  # Example 10: same as (3), but deliver all mail to two different local
  # mailboxes.
  #

[retriever]
  type = SimpleIMAPRetriever
  server = mail.example.net
  username = jeff.plotzky
  password = mailpassword

[destination]
  type = MultiDestination
  destinations = (
"~jeff/Maildir/",
"/var/log/mail-archive/current",
  )

#
  # Example 11: retrieve mail from a simple (non-multidrop) POP3 mailbox.
  # Then extract addresses from the message header (see documentation for which
  # fields are examined), and deliver mail containing the address
```

```
  # <list1@domain.example.net> to ~/Mail/lists/list1/, mail containing the
  # address <list2@otherdomain.example.com> to ~/Mail/lists/list2/,
  # mail containing the address <othername@example.org> to ~/Mail/other/,
  # and all other mail gets delivered through the external MDA program
  # "my-mda" with some default arguments.
  #

[retriever]
  type = SimplePOP3Retriever
  server = pop.example.net
  username = jeff.plotzky
  password = mailpassword

[destination]
  type = MultiGuesser
  default = [my-mda]
  locals = (
  ("list1@domain.example.net", "~/Mail/lists/list1/"),
  ("list2@otherdomain.example.com", "~/Mail/lists/list2/"),
  ("othername@example.org", "~/Mail/other/"),
  )

[my-mda]
  type = MDA_external
  path = /path/to/my-mda
  arguments = ("-f", "%(sender)", "${HOME}/.mymdarc")
```

Also take a look at the getmail man page:

```
man getmail
```

to learn more about available command line parameters that you can pass to getmail.

8 Links

- Getmail: **http://pyropus.ca/software/getmail**
- SpamAssassin: **http://spamassassin.apache.org**
- ClamAV: **http://www.clamav.net**
- Debian: **http://www.debian.org**