# ISPmail tutorial for Debian Lenny

- Add new comment
- 223533 reads

This tutorial is for the former stable version "Debian Lenny". If you are using "Debian Squeeze" then please follow the new tutorial.

*A spanish translation of this tutorial is also available - courtesy of José Ramón Magán Iglesias.*

## What this tutorial is about

You surely know the internet service providers that allow you to rent a domain and use it to receive emails. If you have a computer running Debian which is connected to the internet permanently you can do that yourself. You do not even need to have a fixed IP address thanks to dynamic DNS services like dyndns.org. All you need is this document, a cup of tea and a little time. When you are done your server will be able to...

- receive and store emails for your users from other mail servers
- let your users retrieve the email through IMAP and POP3 - even with SSL to encrypt to connection
- receive and forward ("relay") email for your users if they are authenticated
- offer a webmail interface to read emails in a web browser
- detect most spam emails and filter them out or tag them

## License/Copyright



## Changelog

- 17.6.09: Lenny tutorial gets published.
- 19.6.09: The page on SPF checks is temporarily offline. A new page on spam/virus/phishing fighting is currently prepared.

## Things you will need

The server setup described here is totally standard work for a professional system administrator. Depending on your level of knowledge and experience you may walk through this document easily or curse the author and fail miserably. You will need to know or learn about different topics regarding basic system administration (e.g. how do I edit a text file, where are my log files), DNS, SMTP, MySQL and POP3/IMAP. You also need root access to an existing Debian Lenny server or be able to install one. Also you will likely have to change DNS records for your domain. If your server is protected by the firewall make sure you can change its rules. Going through this setup takes you between 2 hours (for a professional) and a week (if you are a beginner who will inevitably make mistakes). So make sure you are not in a hurry.

## About this document

Many years ago I wanted to turn my Debian server into a mail server with virus scanning, spam detection, email forwarding ("aliasing"), POP3 and IMAP access and a webmail service. All the components were there but it took a while until they worked properly together. So I summed up my desk full of scrawly notes into a tutorial that has become pretty famous. According to my web server statistics it's the main reason why people visit workaround.org.

This document is not a simple copy-and-paste tutorial where you just copy the commands from the web site and run it on your server. Instead it will make you understand the different components that you are setting up. In the end you will be skilled enough to debug problems yourself. If you feel you need help with your setup then try the hints in the Troubleshooting section or ask on the mailing list. The setup in this tutorial has been tested very thoroughly by many readers. Unlike many other Postfix tutorials on the internet this is already the fifth edition. Writing this tutorial took a lot of work so these are not just quick draft notes thrown together but a consistent document guiding you.

The whole tutorial is split into several chapters. Please use the links on the right side or below to navigate through the tutorial. If you prefer all content on a single page (e.g. for printing the tutorial) then use the "*printer-friendly*" link below. You are also invited to comment on the pages - just click on the "*Add new comment*" link at the bottom.

If you like the tutorial then a tiny donation is appreciated to run the test server website hosts and to pay for the internet connection.

# The Big Picture

- Add new comment
- 114938 reads

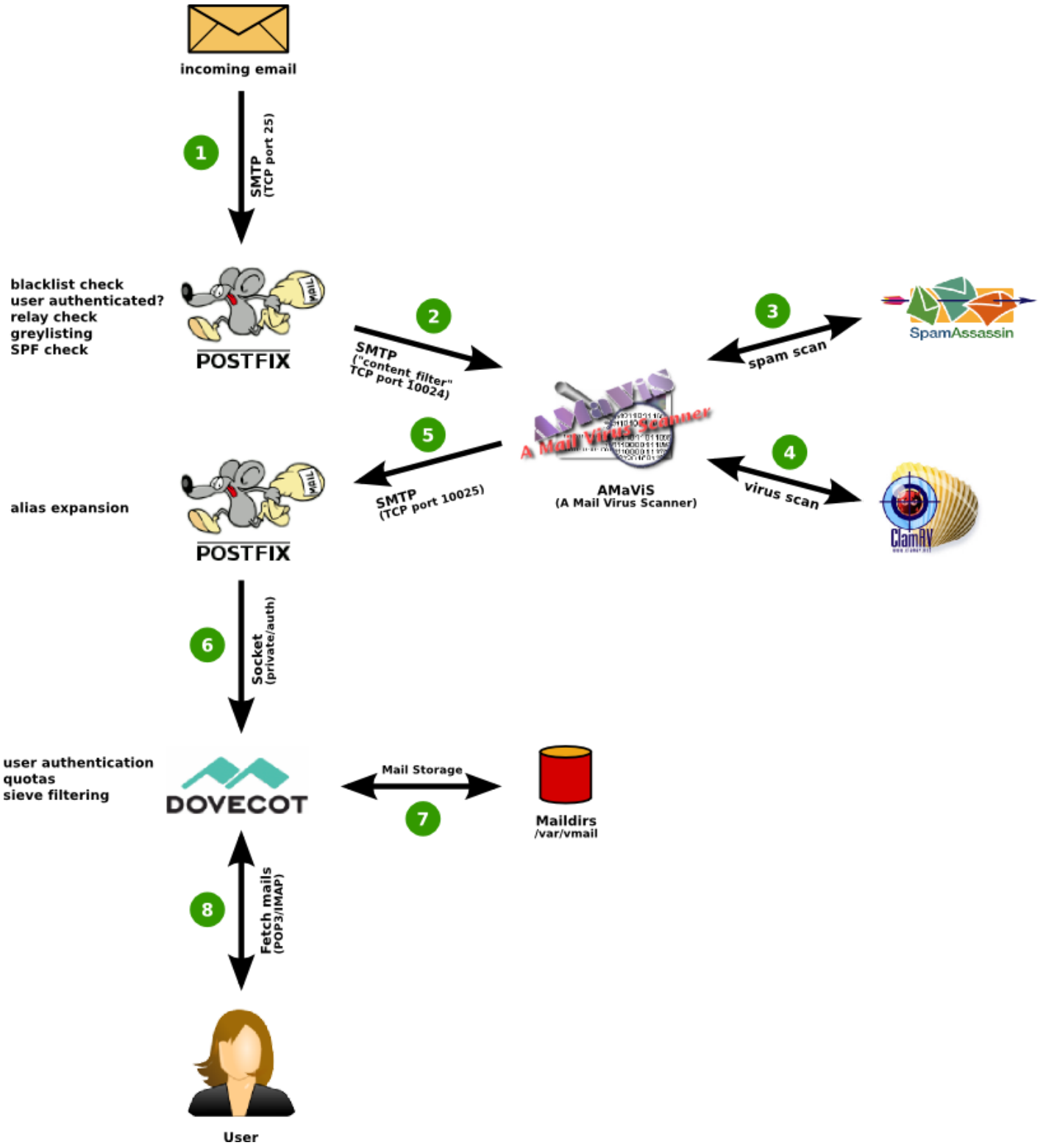
## The Software we will use

The configuration described here uses these software components (the versions are Debian Lenny's default versions):

- Postfix (2.5.5) for receiving incoming emails from the internet and doing basic checks
- Dovecot (1.0.15) to store emails on hard disk and allow users to access their emails using POP3 and IMAP
- Squirrelmail (1.4.15) as a webmail interface so users can read their emails using a web browser
- MySQL (5.0.51a) as the database backend storing information about domains, user accounts and email forwardings
- AMaViS (2.6.1) for content scanning incoming emails using ClamAV and SpamAssassin
- Clam Antivirus (0.94) for virus checking
- SpamAssassin (3.2.5) for spam checking

## The wonderous Ways of an Email

Before going into the details let's see the big picture:




1. An email is sent to your server via the SMTP protocol on TCP port 25. Postfix accepts the connection, reads the email and does some basic checks. Is the sender blacklisted on a realtime blacklist? Is the email from an authenticated user so we bypass relay checks? Or is the recipient of the email a valid user on our system? If we don't trust the remote system yet we apply greylisting. At this stage Postfix can reject the email or accept it.
2. Postfix forwards the email via the SMTP protocol on the TCP port 10024 to AMaViS for content checking. Notice that at this stage the email can't be rejected any more. So AMaVis can either accept it or throw it away. Commonly AMaViS is configured to add a certain email header so the user can see that AMaViS thinks it is spam.
3. AMaViS lets SpamAssassin check the email for spam. SpamAssassin will be taught which emails are spam to increase its detection accuracy.
4. AMaViS also runs the email through ClamAV to see if it contains any viruses.
5. After these checks AMaViS returns the email to the Postfix process but on TCP port 10025. Postfix is configured to trust emails sent to this port so further content checks are skipped.
6. Postfix forwards the email to Dovecot. Dovecot can optionally apply per-user filters so that emails can be stored in certain email folders automatically if desired.

7. Dovecot writes the email to the hard disk in maildir format.
8. The user's email client can now fetch the new emails from Dovecot using the POP3 or IMAP protocol.

# Migrating from the Etch tutorial

- Add new comment
- 39767 reads

The ISPmail tutorial is maintained since 2002. You may have followed former versions of the tutorial and now want to know how to upgrade your mail server to Lenny properly. It is hard to provide exact instructions on what steps to go through. Here are some issues:

## Database schema change

I know you'll hate me for that. But the normalized database layout used in the Etch tutorial was overshooting. So this tutorial uses a more human-readable layout which isn't that much different actually. And it's lighter on the database because the former queries used string operations on a view which might get slow on mail servers with very many users. To migrate your database please first make a backup of it (you never know). Then issue these SQL queries which should migrate your database painlessly to the new schema:

```
-- Create an additional 'email' column in the virtual_users table
ALTER TABLE virtual_users ADD email VARCHAR(100) NOT NULL;

-- Fill the 'email' column with the complete email address.
UPDATE virtual_users LEFT JOIN virtual_domains ON virtual_users.domain_id=virtual_domains.id SET
email=concat(virtual_users.user,'@',virtual_domains.name);

-- Remove the 'user' column
ALTER TABLE virtual_users DROP user;

-- Drop the 'view_users' view
DROP VIEW view_users;

-- Enlarge the email fields in the virtual_aliases view
ALTER TABLE virtual_aliases CHANGE source source VARCHAR(100);
ALTER TABLE virtual_aliases CHANGE destination destination VARCHAR(100);

-- Rewrite the source side of the virtual_aliases to the full email address
UPDATE virtual_aliases LEFT JOIN virtual_domains ON virtual_aliases.domain_id=virtual_domains.id SET
source=concat(virtual_aliases.source,'@',virtual_domains.name);

-- Drop the 'view_aliases' view
DROP VIEW view_aliases;
```

You will also need to adjust the ".cf" configuration files as described in the respective chapter.

## Mail directories now under /var/vmail

The FHS (file hierarchy standard) suggests to put mails under /var/mail. Previously the tutorial expected to put all mails under /home/vmail but the mail directories do not really belong there. So the new tutorial uses /var/vmail as a location. This issue is a bit nasty as you need to move your Maildirs. This tutorial introduces server-side filtering to allow users to apply pre-defined filters when an email arrives. Dovecot stores these filter files in the user's mail directory so the actual mails need to be moved one level deeper in your directory structure.

It's just cosmetics so you don't have to move your mails there. But if you do it then change the following files:

- /etc/dovecot/dovecot.conf:
  mail_location = maildir:/var/vmail/%d/%n/Maildir
  (Early versions of the Etch tutorial didn't use a seperate "Maildir" subdirectory. So if you have an existing directory without that structure you have to create a `Maildir` folder right there and move all mail folders (cur, new, tmp and all folders starting with a dot there. Otherwise the "sieve"-based filtering described in this tutorial won't work.)

- /etc/dovecot/dovecot.conf:
  Check your "namespace private" section to change "/home/vmail" to "/var/vmail".

## AMaViS configuration file

Previously the tutorial recommended to put your custom AMaViS configuration into the /etc/amavis/conf.d/20-debian_defaults file. This will make this file risk getting changed during an update. Please move your settings to the /etc/amavis/conf.d/50-user file instead.

## Global Dovecot/Sieve config file

In recent Dovecot versions the configuration directory for the global sieve filter file has changed. Previously it was configured as the "global_*script*_path" in the "*protocol lda*" section. Now it's the "sieve_global_path" setting in the /etc/dovecot.conf "plugin" section. See also.

## Dovecot now creates maildirs automatically

The Dovecot version used in Debian Etch threw errors if a user accessed a mailbox which had not received any email yet. The Debian Lenny version does not have this issue. So you don't have to send the user a "welcome mail" or even create the maildir manually.

## Dovecot's configuration file

In the /etc/dovecot/dovecot-sql.conf file you need to change the line

```
password_query = SELECT email as user, password FROM view_users WHERE email='%u';
```

to

```
password_query = SELECT email,password FROM virtual_users WHERE email='%u';
```

## Debian's release notes

Also you should read Debian Lenny's release notes before attempting to upgrade your system from Etch to Lenny.

# Preparing the system

- Add new comment
- 50318 reads

## If your server was upgraded from "Etch"

If your server has been upgraded from a former Debian 4.0 "Etch" installation then you may need to fix two files. First make sure that your `/etc/hostname` contains the host name *without* the domain part. It does that on Lenny by default but if you are upgrading from a previous "Etch" installation then this may be wrong. The file `/etc/mailname` is supposed to contain the fully-qualified host name *with* the domain part.

Your `/etc/hosts` may also need to be fixed. Run `hostname --fqdn` and see if you get the fully-qualified hostname. If you just get the hostname without the domain please check that your `/etc/hosts` file has the fully-qualified hostname *first* in the list.

Wrong:

```
20.30.40.50   mailserver42 mailserver42.example.com
```

Right:

```
20.30.40.50   mailserver42.example.com mailserver42
```

## Installing packages

Then begin by installing Postfix with MySQL backend support:

```
$> aptitude install postfix-mysql
```

This will also install the `postfix` package automatically. Exim (the default mail service installed with a fresh installation) will be removed. When asked for the general type of configuration choose "*Internet Site*". Answer the question for the "*System mail name*" by entering the fully qualified hostname of your system.

Run this line on the system to install the MySQL server:

```
$> aptitude install mysql-server
```

Note: You can run MySQL server on the same system as your actual mail service - but don't have to. The mail server can communicate with the MySQL server via TCP networking. During the installation you will get asked for the password of a newly created "root" user. This is not your system administrators login account but a special user to access the MySQL server. Choose any password you want and write it down. And another warning: during the installation a MySQL user account called "debian-sys-maint" will be created with a random password. The password is stored in the `/etc/mysql/debian.cnf` file. Do not touch this file or change the user's password in the database or you won't be able to stop or start the MySQL service any more.

You will want to offer POP3 and IMAP services to your users so you need to install Dovecot's services:

```
$> aptitude install dovecot-pop3d dovecot-imapd
```

Some of the packages needed to scan for virus-infected attachments are not "free enough" to be included in Debian's *main* section (e.g. "unrar" or "lha"). If you want to install them you first need to add the *non-free* section to your Debian mirrors in the `/etc/apt/sources.list` file. You usually just need to add "`non-free`" to your existing mirror lines for the Debian archive like this:

```
deb http://ftp.debian.org/debian/ lenny main non-free
```

and run

```
$> aptitude update
```

to update your package cache with the list of "non-free" packages.

It's a good service to your users to filter out spam and viruses for them. AMaViS is doing a good job here to detect unwanted emails:

```
$> aptitude install amavisd-new spamassassin clamav-daemon lha arj \
   unrar zoo nomarch cpio lzop cabextract
```

AMaViS is now installed with a number of suggested packages to scan attachments for viruses and detect spam.

If you intend to offer a webmail service I can recommend the Squirrelmail package. It will automatically install the Apache web server if you do not yet have one installed. Type:

```
$> aptitude install squirrelmail
```

As your control information for Postfix will be stored in a MySQL database you may want to install the PhpMyAdmin software that allows you to manage the database and its data in your web browser:

```
$> aptitude install phpmyadmin
```

(If you get asked which web server to configure you will probably want to choose "apache2".)

The console-based *mutt* email client lets you read mail from mailboxes directly from the hard disk. It will be helpful for testing the configuration. And it's even a very powerful IMAP email client that many people use as their main mail program. Maybe you'll start to like it, too. You should install it at least for testing:

```
$> aptitude install mutt
```

Now all basic packages are installed and it's time to prepare the database in the next chapter.

# DNS

- Add new comment
- 30737 reads

DNS is short for "domain name system" and is the magic on the internet that turns names like "www.workaround.org" into IP addresses like "212.12.58.129". If you want to receive email for a certain domain you need to be in charge of its "zone" which contains the actual "records". The zone file for "workaround.org" contains an A (address) record so that you can view this website. And among other records it contains an MX

(mail exchanger) record telling which servers should be used when someone sends an email to devnull@workaround.org. This is a simplified extract of the zone file:

```
IN  TXT  "v=spf1 ip4:212.12.58.128/27 -all"
IN  MX      10 mx1.workaround.org
IN  MX      20 mx2.workaround.org
IN  MX      30 mx3.workaround.org
IN  MX      40 mx4.workaround.org
IN  A    212.12.58.129
```

The "IN" is the *class* of the records. On the internet it's always "IN". The second column designates the record *type*:

- TXT = Text record (can be any text up to 255 characters) - used for SPF here (will be described later in the tutorial)
- MX = Mail Exchanger record (designates the email server responsible for this domain)
- A = Address record (default entry)

## How other mail servers find your mail server

Imagine that some mail server on the internet has an email for your domain in its queue. Now it needs to find out which IP address to connect and send the email via the SMTP protocol. First it does a DNS lookup for an MX record of the domain used in the email address. If it gets several addresses back it will try the servers mentioned in the response beginning with the servers having the highest priority (=smallest number) assigned. If it doesn't reach it then it will try the next mail server in order. If the mail server did not find any MX entries then it will try another DNS lookup for an A record and send the email there.

Let's try an example. You want to send an email to devnull@workaround.org. So the domain is apparently the "workaround.org" part. Is there an MX entry? You can try that yourself using the dig tool. ("host" or "nslookup" will work equally well. Install dig using "aptitude install dnsutils" if you haven't already.)

```
$> dig +short workaround.org mx
30 mx3.workaround.org.
40 mx4.workaround.org.
10 mx1.workaround.org.
20 mx2.workaround.org.
```

So there are four mail servers in question. The one with the highest priority (10) is mx1.workaround.org. Let's find out its IP address by asking for an A record of this hostname:

```
$> dig +short mx1.workaround.org
212.12.58.158
```

The mail server will now try to reach the mail server on this IP address on TCP port 25 (SMTP) and attempt to deliver the email. Incidentally (or maybe not) the mail server does not appear to be reachable. There are no more mail servers with priority 10 so it tries the next best mail server mx2.workaround.org with priority 20. Which IP address does it have?

```
$> dig +short mx2.workaround.org
212.12.58.129
```

This mail server works better and the SMTP connection is established and the email delivered.

So in order to receive email for a domain you need to set at least one MX entry in its zone to point to your mail server. Note that an MX entry points to a hostname - never an IP address. If you can't add MX entries then an A record will do, too.

# SPF to avoid spoofing

- Add new comment
- 87890 reads

If you are tired of faked emails from popular domains like eBay or Paypal then there is hope. And you can even protect your own domain from spammers sending in your name. It is about *SPF* which is short for sender policy framework. Basically the owner of a domain defines which IP addresses are allowed to send email. Let's take an example. If you have the dig tool around then run…

```
$> dig +short workaround.org txt
```

and you should see something like

```
"v=spf1 ip4:85.214.93.191 ip4:85.214.149.150 -all"
```

This entry means that if someone sends you an email with a sender email address of …@workaround.org then you should only accept it if it was sent from one of these two IP addresses. The "-all" at the end tells you that no other IP addresses should be accepted (*FAIL*). Another definition like "~all" means a *SOFTFAIL* - you should at least be suspicious and perhaps increase the spam score. (Actually "~all" is widely used but pretty useless. Either you know what you are doing and use "-all" or leave it off entirely. If you want to test if SPF workd then you could use "?all".) So if someone sent you an email from …@workaround.org from another IP address then you should drop it - it's spam.

Many organisations already have SPF records that you can use to reduce the amount of spam you receive. So your duties as a mail server administrator are:

- set up a TXT entry in your DNS zone defining which IP addresses are allowed to send email in the name of your domain
- check SPF entries of sending email servers and reject email coming from unauthorized IP addresses

# Set up an SPF entry

Obviously you need to have full control of your DNS zone to add a TXT record. If you don't then speak to your ISP and let them add the TXT entry. The SPF record needs to be properly machine readable. I suggest you go to the OpenSPF web site and use their wizard to create a proper SPF entry. Add this string as a TXT record for your domain. Just like above you should get the SPF entry back if you run:

```
$> dig +short mydomain.com txt
```

You need to be aware of one caveat though which is also SPFs biggest problem: if your users forward the email somewhere else then it might get rejected. By all means you need to be sure that your users always send email just through your mail server.

# Check other mail servers' SPF entries

Fortunately there is a Debian package for the *tumgreyspf* software which makes checking SPF entries easy. Just install it:

```
$> apt-get install tumgreyspf
```

tumgreyspf is a policy daemon written in Python that does both greylisting and SPF checking of incoming emails. Using it is detailed in the /usr/share/doc/tumgreyspf/README.Debian file. Basically it boils down to adding one line to your smtpd_sender_restrictions (or smtpd_recipient_restrictions if you put everything in there) making use of it. Example:

```
smtpd_sender_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    [ ... ]
    check_policy_service unix:private/tumgreyspf
    [ ... ]
    permit
```

And to define the program that is called when using this policy service you need to add two lines to your /etc/postfix/master.cf:

```
tumgreyspf unix  -      n       n       -       -       spawn
    user=tumgreyspf argv=/usr/bin/tumgreyspf
```

Now reload your Postfix and that's it:

```
$> postfix reload
```

### Case: SPF okay

Watch your /var/log/mail.log logfile. Every incoming email should now log an additional line from tumgreyspf. If the SPF check was positive then you will get:

```
tumgreyspf[24672]: sender SPF authorized: QUEUE_ID=""; identity=mailfrom;
   client-ip=26.21.244.31; helo=squedge2.squ.edu.om;
   envelope-from=…@squ.edu.om;
   receiver=…@workaround.org;
```

This means that the sender …@squ.edu.om was allowed in after checking the SPF entry.

### Case: SPF fail

If the SPF check fails then you will see something like:

```
tumgreyspf[24672]: SPF fail - not authorized: QUEUE_ID=""; identity=mailfrom;
   client-ip=41.234.18.141; helo=gmx.de;
   envelope-from=…gmx.de;
   receiver=…christoph-haas.de;
```

The email got rejected. One spam mail less in the world. :)

### Case: SPF softfail

The third case is when the SPF entry does not enforce a FAIL (-all) but just uses SOFTFAIL (~all). In your log file it will look like:

```
tumgreyspf[20408]: domain owner discourages use of this host: QUEUE_ID="";
   identity=mailfrom; client-ip=220.245.2.67; helo=220-245-2-67.static.tpgi.com.au;
   envelope-from=…@rollouts.com; receiver=…@workaround.org
```

Unfortunately the SOFTFAIL does not actually reject the email. But the sender still believes that this IP address should not be sending email in their name. Luckily tumgreyspf adds this information to a "*Received-SPF*" header into the email. Just like:

```
Received-SPF: Softfail (domain owner discourages use of this host) identity=mailfrom;
   client-ip=61.146.93.243; helo=mail.163gd.com;
   envelope-from=…@cantv.net; receiver=…@christoph-haas.de;
```

So you still have a chance to mark such emails in spam in your email client by filtering out emails having a "Received-SPF: Softfail" header.

(Unfortunately tumgreyspf does not have a configuration option to treat SOFTFAIL as FAIL.)

### Case: No SPF information

If the remote domain is ignorant and stupid and does not have any SPF entries yet then your log file will read:

```
Received-SPF: Neutral (access neither permitted nor denied) identity=mailfrom;
   client-ip=80.65.65.222; helo=mail.unze.ba;
   envelope-from=…@gmail.com; receiver=…@christoph-haas.de;
```

# Quotas

- Add new comment
- 46871 reads

Your mail server doesn't have infinite space. Especially if you use IMAP then users appreciate the comfort of leaving their emails on the server. And there are even IMAP users who are not aware of emails that are just *marked* for deletion but are still occupying space. So unless you have very few or just highly disciplined users you may want to limit the space a user can occupy. Dovecot can store the size of a user's mailbox and the number of mails therein along with the virtual folder.

## Enabling the quota plugin in Dovecot

There are three locations in the `/etc/dovecot/dovecot.conf` file where you need to enable the **quota** plugin (taken from the Dovecot documentation):

```
protocol imap {
  mail_plugins = quota imap_quota
}
protocol pop3 {
  mail_plugins = quota
}
protocol lda {
  mail_plugins = quota
}
```

## Setting global quota

The simplest case is a general quota limit for all users. Say you grant every user 1 GB of space with no more

than 1000 emails. This would be the configuration in your `/etc/dovecot/dovecot.conf` file:

```
plugin {
  quota = maildir:storage=1000000:messages=1000
}
```

Keep in mind that the **storage** parameter refers to KB. You can even omit the **messages** parameter if you don't intend to limit the number of messages.

## Setting per-user quota

If you have certain users who have different quotas from your global quota setting then you can store the quota settings in your *virtual_users* table. Use this SQL statement to add two columns to the virtual_users table:

mysql>
**ALTER TABLE `virtual_users` ADD `quota_kb` INT NOT NULL,**
**ADD `quota_messages` INT NOT NULL ;**

And you will have to enable the user_query in your `/etc/dovecot/dovecot-sql.conf`. Throughout this tutorial it was suggested you use the "userdb static" approach in your `/etc/dovecot/dovecot.conf`. But this assumes that all users have equal settings. So in this per-user quota situation you will need to disable "userdb static" again and enable "userdb sql" to make Dovecot make an extra query to fetch the user's data from the database:

#userdb static {
#  args = uid=5000 gid=5000 home=/var/vmail/%d/%n allow_all_users=yes
#

userdb sql {
   args = /etc/dovecot/dovecot-sql.conf
}

And in your `/etc/dovecot/dovecot-sql.conf` add this line:

user_query = SELECT
CONCAT('/var/vmail/',CONCAT(SUBSTRING_INDEX(email,'@',-1),'/',SUBSTRING_INDEX(email,'@',1))) AS home, 5000 AS uid, 5000 AS gid, CONCAT('maildir:storage=',quota_kb,':messages=',quota_messages) AS quota FROM virtual_users WHERE email='%u';

This query may look a bit weird but it does the same as the above

args = uid=5000 gid=5000 home=/var/vmail/%d/%n allow_all_users=yes

line and in addition gets the **quota_kb** and **quota_messages** information.

## What happens if a user is over quota

Quotas in Dovecot aren't especially user-friendly. For example the recipient does not get a warning at a *soft quota* limit. Just the sender gets a bounce email with the subject reading "*Automatically rejected mail*" saying "Your message to <john@example.com> was automatically rejected: Quota exceeded"

# Troubleshooting

- Add new comment
- 31353 reads

If you are having trouble receiving or sending email you may try these general steps:
Check your `/var/log/mail.log`. 93.7% of all problems cause a more or less clear error message there. :)

- Run postfix check. No output means everything is well.
- Ask on the #postfix IRC channel on irc.freenode.net. I usually attend there as Signum but please just ask and wait for people to help you. Especially read the hints in the channel's topic and my hints on how to get useful help on IRC.
- Ask on the forums.
- Check the (deprecated) workaround chitchat mailing list archives
- This tutorial is meant for Debian 5.0 "Lenny". If you are trying this setup on any other distribution you are completely on your own.

- List contents of the mail queue (add -v to be more verbose, add multiple -v's for increased verbosity):
  **mailq** or **postqueue -p**

  Schedule immediate delivery of all mail that is queued:
  **mailq -q**

  Amavis:
  **/etc/init.d/amavisd-new stop**
  **/etc/init.d/amavisd-new debug**

Sending an email directly to the author asking for help is the worst choice. Sending an email directly to the author giving suggestions or feedback is a good choice. :)