GNU Mailman bundler Documentation

Release 3.0.0

Mailman Developers

May 09, 2016

Contents

This package uses Buildout to install GNU Mailman, its admin interface Postorius, and its archiver HyperKitty.

The Mailman home page is http://www.list.org, and there is a community driven wiki at http://wiki.list.org.

CHAPTER 1

Installation

Mailman is written in Python which is available for all platforms that Mailman is supported on, including GNU/Linux and most other Unix-like operating systems (e.g. Solaris, *BSD, MacOSX, etc.). Mailman is not supported on Windows, although web and mail clients on any platform should be able to interact with Mailman just fine.

Mailman requires at least Python 3.4, which may or may not be shipped by your operating system. If only earlier versions of Python are available, you must install Python 3.4 (or later) before following this tutorial. To do that, follow the procedure in the Python documentation.

You must also have Pip for Python 3.4 installed. If you're using your distribution's package manager, it probably comes in a `python3-pip` package (or `python3.4-pip`, or something similar). Otherwise you'll have to download it from PyPI.

Even if Mailman 3 runs on Python 3, the web interfaces and the commands in this procedure run on Python 2.7, so make sure your system Python version is 2.7.

To use Mailman, you must install and configure a *Mail Transfer Agent* (MTA) supported by Mailman. To this day, Postfix and Exim are supported. Installing and configuring an MTA is beyond the scope of this documentation; for assistance please refer to your specific operating system and preferred MTA. This package only provides Mailman configuration examples for Postfix.

If you haven't already done so, download the sources of the Mailman bundler with the command:

```
bzr branch lp:mailman-bundler
```

Install Virtualenv (the Python 2.7 version), preferably using your distribution's package manager. Now switch to a dedicated Mailman user. Setup a virtualenv for the Mailman suite and activate it with the following commands:

```
virtualenv venv
source venv/bin/activate
```

In the bundler directory, open the `mailman_web/development.py` file, look for the `SECRET_KEY` parameter and set something random.

You will need to have GCC installed on your machine to install the dependencies. If you have installed Python using your operating system's packages, make sure you have also installed the development headers. They are usually shipped in a package with the `-devel` or `-dev` suffix, for example `python-devel` and `python3-devel`. You need the headers for both Python 2 and Python 3.

In the `mailman-bundler` directory, install and run buildout:

```
pip install zc.buildout
buildout
```

You will also need to install the LESS CSS compiler. It is usually packaged by your distribution, on Fedora the package is named `nodejs-less`, so you can install it with:

```
sudo yum install nodejs-less
```

On Debian and Ubuntu, this is available in the `node-less` package, which you can install with:

```
sudo apt-get install node-less
```

Now initialize Django's database:

```
./bin/mailman-post-update
```

Now create an initial superuser to login as:

```
./bin/mailman-web-django-admin createsuperuser
```

This is the user you'll use to login to Mailman's web interface (Postorius).

CHAPTER 2

Run the services

Start mailman:

```
./bin/mailman start
```

Run Django (the web interfaces server):

```
./bin/mailman-web-django-admin runserver &
```

By default, the web interface are available at:

- http://127.0.0.1:8000/mailman3 for Postorius
- http://127.0.0.1:8000/archives for HyperKitty

CHAPTER 3

Configure Postfix

The `deployment/postfix-main.cf` will contain a few lines that you must add to your main Postfix configuration file (usually `/etc/postfix/main.cf`).

Add full qualified domain name of list(s) to `mydestination`:

```
mydestination = lists.mydomain.com mydomain.com
```

Make sure postfix has read access to the `*.db` files in `var/data` (it won't have access by default). There are two ways to do that: you can change the permissions of those files to world-readable or add postfix to your user's group.

If you have SELinux enabled, make sure it is not blocking access to those files. Their type should be `etc_aliases_t`. You can set this type permanently using the `semanage fcontext` command.

Finally, restart postfix:

```
service postfix restart
```

CHAPTER 4

Use Mailman

4.1 Create a domain

- Open your browser and navigate to Postorius (http://127.0.0.1:8000/mailman3)
- Go to Settings -> "New domain"
- Enter a mail host: `lists.example.com`
- Enter a web host: `http://lists.example.com`

4.2 Create a mailing-list

- In Postorius, go to "Lists" and click "New list"
- Enter list name, e.g. `test`
- Choose the mailhost
- Define the list owner (defaults to domain contact)
- Choose whether to advertise list
- Enter description
- Save list.

On the lists' front page, add a moderator in the corresponding category.

4.3 Subscribe to your mailing-list

To subscribe, send an email to `test-request@lists.example.com` with subject `subscribe`.

To unsubscribe, send an email to `test-request@lists.example.com` with subject `unsubscribe`.

CHAPTER 5

Setting up for production

For a production setup, you first need to change the `deployment` parameter to `production` in `buildout.cfg` and run `buildout` again. It will regenerate the scripts in `bin` and the contents of the `deployment` directory.

If you want to use a webserver module to serve Postorius and HyperKitty (like Apache's mod_wsgi), make sure your current directory is accessible by its user. On some distributions, user directories in `/home` are not accessible by the Apache user.

It is also **very** strongly recommended to use an full-blown database server for Mailman, Postorius and HyperKitty. If you choose PostgreSQL, you'll need the corresponding drivers, which can be installed with:

```
pip install psycopg2
```

This will require the PostgreSQL headers to compile, just install them from your distribution's package manager. You also obviously have to start your database server, and create the databases and database users you are going to use.

The connection parameters to the database can be configured in `mailman.cfg` for Mailman, and `production.py` for Postorius and HyperKitty. See the `DATABASES` setting value.

Also note that HyperKitty uses a database and a full-text search index. The backend used for this search engine is configured in the `HAYSTACK_CONNECTIONS` variable of the settings file. Refer to the Haystack documentation for the detailed settings available to configure the full-text search engine.

If you want to make modifications to this `production.py` file, it is recommended to create a `settings_local.py` file in the same directory and put your variables there. This file will be sourced by the `production.py` file and variables will be overrrided. This way, you can easily merge changes in the `production.py` file when you upgrade the package. Only put in `settings_local.py` the variable that you want to override.

The `production.py` file also configures two variables pointing to directories that must be created (probably as `root` if you keep the default values) and given the correct permissions.

- the `STATIC_ROOT` variable, pointing to a directory where static files will be collected to be served directly by your frontend webserver, must be owned by your current user
- the `HAYSTACK_CONNECTIONS.default.PATH` variable, pointing to a directory where the fulltext search engine files will be stored, must be readable and writable by your webserver user.

Finally, there are some other variables that need to be set in `production.py` for production serving. You must set the hostname your website will respond to in `ALLOWED_HOSTS` and `BROWSERID_AUDIENCES`. Check out the links in the comments above those variables in `production.py` for information on their expected value.

Also, if you're using a proxy to serve your website, there may be additional variables that you need to set in `production.py`, such as `USE_X_FORWARDED_HOST` and `SECURE_PROXY_SSL_HEADER`.

After settings those variables, re-run the `bin/mailman-post-update` script, it will initialize the databases and collect the static files for direct serving.

In the `deployment` directory, you will find configuration files that can be useful for a production setup.

5.1 The Mailman service

The `mailman3.service` file is an example Systemd service file to run Mailman as a system daemon. You need to edit this file to set the `User` and `Group` values to your current user (the one you ran buildout as), or you will have to change the permissions on the *var* subdirectory.

If you run `buildout` again, this file will be overwritten, and you will have to set the `User` and `Group` values again.

We currently have no SysVInit service file, but it should be easy to write if needed (then please send it to us for inclusion!).

The `postfix-main.cf` file contains lines that must be added or adjusted in Postfix' `main.cf` configuration file. This bundle does not offer an example Exim file, please send us your configuration if you use Exim.

You will also find a `mailman3.logrotate.conf` file to place in your `/etc/logrotate.d` directory to ensure rotation of the Mailman logs.

5.2 The web interface

The web interface can be made accessible in two ways:

- running as a WSGI application via a dedicated module in your webserver
- running standalone on a different port, and pointed at by a proxy or a proxy module in your webserver.

This bundle offers a few example configuration files, but it does not cover all cases. Please send us configuration files for the setups that are not covered yet.

5.2.1 Log files

In any case, the `production.py` configuration file defines a directory where Postorius and HyperKitty logs will be written to. By default, this directory is `/var/log/mailman-web`. You will have to create this directory as `root`, and then give ownership back to your webserver user.

In the `deployment` directory you will find `mailman-web.logrotate.conf`, an example logrotate file to put in the `/etc/logrotate.d` directory, to ensure log rotation both for Postorius and HyperKitty.

5.2.2 Running on Apache HTTPd with `mod_wsgi`

The `apache.conf` file is an example configuration file for Apache HTTPd's `mod_wsgi` module that will run Postorius and HyperKitty in the same Django instance. You will have to adjust the following permissions for your Apache user on the `var/mailman-web` directory:

- write access to the SQLite database
- write access to the `fulltext_index` directory
- read access to the static files.

5.2.3 Running on Gunicorn

To add Gunicorn support, just run the following command:

```
buildout install gunicorn
```

(it is not enabled by default to minimize dependencies) You can then serve the Mailman web interfaces with Gunicorn by running the following command:

```
./bin/gunicorn mailman_web.wsgi:application
```

The webserver will listen on port `8000` by default, check out the Gunicorn documentation to change that.

A service file and a socket activation file are provided in the `deployment` directory, their basename is `mailman-web-gunicorn`. You can enable them the usual Systemd way (with a symlink), but remember to change the `User` parameter to a specific user for security.

Please also read the Gunicorn deployment documentation.

5.2.4 Other setups

Those are the only setups this bundle has example configuration for, but certainly not the only ones supported! We are looking for configuration examples for the following setups:

- running in uWSGI
- proxy configuration for Apache's mod_proxy
- proxy configuration for Nginx
- anything else you may be using these days ;-)