Mailman Suite Documentation

Release 3.3

Free Software Foundation

Feb 10, 2020

Contents

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

Mailman Core 3.3.0 was released on September 5, 2019. The Mailman Suite consists of 5 individual projects. Below are links to documentation for each of the projects.

- Mailman Core - the mailing list manager core (required)
- Postorius - the adminstrative web user interface
- MailmanClient - the official REST API Python bindings
- HyperKitty - the web archiver
- HyperKitty Mailman plugin - archiver plugin for Core

CHAPTER 1

The Pre-Installation Guide

What do I need to know before trying to install Mailman3?

1.1 Pre-Installation Guide

1.1.1 What is the current state of Mailman 3?

Mailman 3 is a complete re-write of Mailman 2.1 and has been split into several modular components which collectively we call Mailman Suite. We also often just call it Mailman 3. Let us look very briefly into these components:

- **Mailman Core** or just Core is the main engine that is responsible for handling emails. It does all the task of managing users, subscriptions, mailing lists, email addresses, talking to the MTA etc. However, the things that it doesn't manage include user authorization & user authentication.

  However, users can manage their settings using email based commands which includes subscribing, unsubscribing, changing a few basic settings for their accounts etc.

  Most functionalities in Core can be accessed using an administrative REST API which provides full control over the Core. Core expects the clients that consume this API would take up the responsibility of authenticating and authorizing users and provide interfaces for them to manage their settings and subscriptions.

- **Mailman Client** is just sometimes called as Client and is the official Python bindings to the Core's REST API.

- **Postorius** is the official web front-end over the Core's REST API and is built on Django web framework. In Django's terminology, Postorius is an "app" which can be be plugged into any Django installation or "project". Postorius has been built to be deployed alongside other Django "apps" which makes it a little bit difficult to deploy since you need to have a "project" of your own if you just need it to work!

  See also:

  What is Django?

  If you don't have any familiarity with Django, we do provide an example Django "project" with pre-populated settings which should run Postorius out of the box. However, because of the several different ways in which you

can configure Django, it might not suite everybody's needs, in which case you might have to dive into Django's (and other helper library's) settings related documentation to figure out the best settings for you.

- **Hyperkitty** is the official archiver and, similar to Postorius, is a Django "app". It is also built along the same principles of re-usability and can be deployed alongside Postorius with the provided Django "project".

1.1.2 How Can I upgrade from Mailman 2.1.x?

Mailman 2.1 series is the current most popular series of Mailman. The short version is that as of now, upgrading from Mailman 2.1 to Mailman 3.1 is buggy.

Now the long version. Because of the changes in Database Schema, migrating from Mailman 2.1 to Mailman 3.1 is not very easy, though it can be done with some scripting. We are working on it and it should be working soon, we don't have an exact timeline on it though.

Archives however can be imported into Hyperkitty easily, but URLs to attachments are going to break because the URL paths are different in Hyperkitty. Although, You might be able to retain your HTML archives from Mailman 2.1 and continue archiving newer emails in Hyperkitty.

1.1.3 What do I need to know before deploying Mailman 3?

The installation guide presumes some knowledge of general Python based web applications and their ecosystem. This requirement however is only required if you follow the installation guide provided here. When the distro packages for Mailman 3 are out, you probably won't need any of this information. If you are new to Python and have no idea about what `pip`, `django` and `wsgi` means, this section is for you!

Here are the three most basic terminologies that you will often encounter when trying to deploy Mailman 3:

- `pip` is the Python's official package manager and can be used to download and install libraries and packages from PyPI a.k.a Python CheeseShop. On most Linux based distributions, you can install it from the distro's package manager. Here are the instructions for `apt` & `yum` (or `dnf`) based systems:

  ```
  $ sudo apt install python3-pip
  ```

  and:

  ```
  $ sudo yum install python3-pip
  ```

  If you have any other Linux distro, please check its documentation on how to install. After that, you can use it to install python packages using the command like below:

  ```
  $ sudo pip3 install <packagename>
  ```

  The above command is equivalent to:

  ```
  $ sudo python3 -m pip install <packagename>
  ```

- Django is the Python web framework that Postorius & Hyperkitty are based on. It allows you to run several different web "apps" under a single "project". People often write their web applications as reusable Django "apps" which can then be plugged into any running Django "project". A Django "project" is also sometimes called a Django "installation".

  A typical Django project has a structure of something like this:

```
toplevel_project:
\
|---- urls.py
|---- settings.py
|---- manage.py
|---- wsgi.py
```

It is important to understand the above structure even when *using* Django, because the configuration of a Django project requires you to edit these files.

Here is a brief overview of these files:

- `urls.py`: This is how Django routes requests. If you want to install a new Django "app", you have to add its `base_url` to this file.
- `settings.py` : This is Django's configuration file. It is the home for all the different configuration options that are required.
- `manage.py` : This is a generic helper script used to perform administrative tasks from the command line on a Django project. You should never edit this file after creating a project.
- `wsgi.py` : This is the WSGI or Web Server Gateway Interface application for the Django project. You will need this file later to interface Django with an actual webserver like Nginx or Apache. Usually, you don't need to change anything in this file.

- **wsgi** or the Web Server Gateway Interface is the protocol by which Python web applications talk to web servers. Django comes with a built-in web server, which you can invoke by using the following command:

```
$ python manage.py runserver
```

This will start a development server on http://localhost:8000/. You can also specify a different host:port to bind to. See `python manage.py runserver --help` for more instructions.

To deploy any Python based web application, you need an intermediate WSGI server which mediates the interaction between Python and a web server. There are several of them out there but we recommend using uwsgi. Uwsgi has several advantages over others including the ability to configure it entirely using environment variables, which helps in container based deployments. Also, Nginx and Apache web servers have plugins built-in for Uwsgi which makes the it an even more compelling candidate.

However, you are not tied to using Uwsgi and are free to choose any other WSGI server.

- When running Django using the built-in development web server, Django serves its static files which makes it easy for the developers. However, in production environment, it is advised to serve the static files separately.

To collect all the static files for all the Django projects in one single place:

```
$ python manage.py collectstatic
```

This will collect all the static files in the location mentioned in Django's settings as `STATIC_ROOT`, which is *usually* under `static` directory in Django's project's root path.

You need to serve this directly with your web server using a proxy or alias rule, depending on your webserver.

Here is the relevant portion of the configuration for Ningx:

```
server {
    ...

    location /static/ {
         alias /path/to/django/STATIC_ROOT;
```

(continues on next page)

(continued from previous page)

```
    }
}
```

You can do this in Apache using a configuration that looks something like this:

```
<VirtualHost *:443>
   ...

   Alias /static /opt/mailman/web/static
   Alias /favicon.ico /opt/mailman/web/static/hyperkitty/img/favicon.ico
   ProxyPassMatch ^/static/ !
   ...
</VirtualHost>
```

CHAPTER 2

Upgrading to Mailman Suite 3.2

Upgrading from Mailman Suite 3.1 to 3.2

2.1 Upgrading to Mailman 3

This includes the requirements to upgrade to Mailman 3 suite, which includes:

- Mailman Core 3.x
- Postorius 1.x
- MailmanClient 3.x
- Django-mailman3 1.x
- Hyperkitty 1.x

If you are interested in the full change log for each component, please refer to documentation of each project.

2.1.1 Python 3

Postorius, Django-mailman3 and Hyperkitty have been ported to Python 3 now. To upgrade from a previous version, you would need to setup your Django project to use Python 3.5+. Django 1.11+ are supported.

If you are using virtual environments, you can now run the whole stack from a single virtual environment, unlike before, when you needed separate ones for Django and Mailman Core.

2.1.2 Migration Steps

The migration for your Mailman installation depends on how you got everything running.

- If you are using system packages, they should handle the upgrade process themselves. You *may* need to perform some changes after the upgrade is done, see the sections below for those steps.

- If you are using the packages directory from PyPI or source, you can do a `pip update -U mailman` to install Mailman Core. For Django packages, you would need to create a new virtual-environment (if you are using it), or install it using Python 3 pip.
- For Django apps, the usual steps for migrations include:

```
$ python manage.py migrate
$ python manage.py collectstatic
$ python manage.py compress
```

- Mailman Core should handle database migration on its own when the new version is started.

2.1.3 Configuration

Postorius added a new *required* configuration flag *POSTORIUS_TEMPLATE_BASE_URL* which should be set to URL that Postorius is expected to be listening at. You should set it to whatever URL your WSGI server is listening at.

2.1.4 Extra Migration Steps

- Full text index for Hyperkitty needs to be re-built since our indexing engine doesn't maintain compatibility between index created in Python 2 and Python 3.

  Simplest way to do this to run `python manage.py rebuild_index` in your Django project. Note that if your project has a huge number of lists, this will take a lot of time.

  A new command `python manage.py update_index_one_list <listname@example.com>` was added to Hyperkitty so that you can rebuild your index one-by-one, if you prefer that.

CHAPTER 3

The Installation Guide

3.1 Installation Instructions

3.1.1 Mailman 3 in Docker Containers

Abhilash Raj maintains container images for Mailman 3 which you can use directly without having to go through all the steps to download dependencies and configuring Mailman. You can find the detailed documentation on how to use them. If you have any issues using the container images, please report them directly on the Github repo for docker-mailman.

3.1.2 Installing Mailman from Linux distro packages

Other distribution specific packages for Mailman 3 are not yet available. If you would like to help package Mailman 3 for your favorite Linux distro, please get in touch at mailman-developers@python.org.

Installing on Debian

Debian provides Mailman 3 packages. The meta-package `mailman3-full` depends on all components of a complete Mailman 3 suite. If you want to split the installation of the Mailman 3 core mailinglist delivery daemon and the Mailman 3 Django web suite with Postorius and Hyperkitty, take a look at the packages `mailman3` and `mailman3-web`.

The Debian Mailman 3 packages can be found here:

> https://packages.debian.org/search?keywords=mailman3

3.1.3 Virtualenv Installation

This is a step-by-step installation guide for Mailman Suite, also sometimes referred as Mailman 3 Suite or just Mailman 3. There are more than one ways to have Mailman 3 running on your machine.

Dependencies

Python3.5+. Mailman requires any version of Python > 3.5.

MTA Setup Mailman 3 in theory would support any MTA or mail server which can send emails to Mailman over LMTP. Officially, there are configurations for Postfix, Exim4, qmail and sendmail. Mailman Core has a fairly elaborate documentation on setting up your MTA. Look below at **'setting up mailman core'_** to find out the location of configuration file `mailman.cfg` which is mentioned in the documentation above.

The Web Front-end is based on a Python web framework called Django. For email verification and sending out error logs, Django also must be configured to send out emails.

Sass compiler A sass compiler. Syntactically Awesome Stylesheets or Sass is an extension of CSS, which is used to style webpages, that adds more power to the old CSS syntax. It allows using variables, nested rules, inline imports etc. which CSS originally doesn't support. Hyperkitty uses this to generate CSS styles.

You can either use the Ruby implementation or C/C++ implementation. Please look at the installation guide for sass to see how you can get one.

For apt based systems:

```
$ sudo apt install ruby-sass
```

For yum based systems try:

```
$ sudo yum install rubygem-sass
```

or:

```
$ sudo dnf install rubygem-sass
```

After installing this, you'd have to configure Django to use these compilers. A basic configuration would look like:

```
# To be added to Django's settings.py

COMPRESS_PRECOMPILERS = (
  ('text/x-scss', 'sass -t compressed {infile} {outfile}'),
  ('text/x-sass', 'sass -t compressed {infile} {outfile}'),
)
```

You can replace `sass` above with whatever is the name of the binary is. For the Ruby implementation it is generally `sass`, for C/C++ it is `sassc`.

Python development packages python3 dev package. This is required for building postorius

For Fedora the package is `python3-devel`:

```
$ sudo dnf install python3-devel
```

For Ubuntu/Debian the package is `python3-dev`:

```
$ sudo apt install python3-dev
```

Fulltext search A full text search engine like Whoosh or Xapian. Whoosh is the easiest to setup and can be installed in the virutualenv.

See also:

Virtualenv setup on how to setup the virtualenv.

Lynx An HTML to plaintext converter like `lynx` is required by Mailman Core if you have configured it to convert emails to plaintext.

Virtualenv setup

Virtualenv is Python's mechanism to create isoated runtime environments.

Hint: If you are not familiar with virtualenv, checkout the user guide for virtualenv.

1. Setup working directory:

```
# Setup the base installation directory.
$ sudo mkdir -p /opt/mailman
```

2. Create the virtualenv for Mailman:

```
$ cd /opt/mailman
$ python3 -m venv venv
```

3. Activate the virtualenv:

```
$ source venv/bin/activate
```

Note: The rest of this documentation assumes that virtualenv is activated, which can be done by step 3 mentioned above. Whether or not virtualenv is activated can be seen by a `(venv)` before the shell prompt.

Installing Mailman Core

Mailman Core is responsible for sending and receiving emails. It exposes a REST API that different clients can use to interact with over an HTTP protocol. The API itself is an administrative API and it is recommended that you don't expose it to outside of your host or trusted network. To install Core run:

```
(venv)$ pip install mailman
```

This _should_ install the latest release of Mailman Core, which is 3.1.0 as of writing of this document. After this, create a configuration file at `/etc/mailman.cfg` for Mailman Core.

See also:

The further configuration setup for Mailman Core *Configuring Mailman Core*.

After this, running `mailman info` will give you an output which looks something like below:

```
(venv)$ mailman info
GNU Mailman 3.1.0 (Between The Wheels)
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170118]
config file: /etc/mailman.cfg
db url: sqlite:////var/lib/mailman/data/mailman.db
devmode: DISABLED
REST root url: http://localhost:8001/3.1/
REST credentials: restadmin:restpass
```

Installing Web UI

Postorius and Hyperkitty are Mailman's official Web UI and Archiver. Both of them are Django based apps and can be integrated into an existing Django installation/website.

Installing Postorius & Hyperkitty

Postorius and Hyperkitty are Django "apps" and can be directly installed using the commands below:

```
(venv)$ pip install postorius
(venv)$ pip install hyperkitty
```

mailman-hyperkitty plugin enables interaction between Core and Hyperkitty. You can install it using:

```
(venv)$ pip install mailman-hyperkitty
```

Note that mailman-hyperkitty is a plugin for Mailman Core and not Django.

Setting up Django Project

See also:

What is Django?

If you have absolutely no idea about Django, just clone/download this mailman-suite repo . It should have a documented `settings.py` and a pre-configured `urls.py` which you can use to run new Django installation. If you find any setting that is not documented, please look at Django's settings reference for all available settings.

Exact commands would look something like this:

```
# Download and install the latest release of django.
(venv)$ pip install Django>=1.11

# Clone the repo locally.
(venv)$ git clone https://gitlab.com/mailman/mailman-suite.git
(venv)$ cd mailman-suite/mailman-suite_project/

# Create the tables in the database and load fixtures.
(venv)$ python3 manage.py migrate

# Copy all the static files to one single location.
(venv)$ python3 manage.py collectstatic

# Run the Django's "development" server at localhost:8000
(venv)$ python3 manage.py runserver
```

Setting up a WSGI server

See also:

What is WSGI?

These instructions are to setup your Django website behind a webserver. We are using a uwsgi as the wsgi server to communicate between the webserver and Django. To install uwsgi, run:

```
(venv)$ pip install uwsgi
```

Note: The configuration below doesn't serve static files, so if you are just "trying-it-out" and want static files to be served, you need to add some additional configuration and steps. See serving static files with uwsgi.

See also:

Why does my django site look ugly?

See also:

django uwsgi docs

Then you can configure it using the following configuration file:

```
# uwsgi.ini
#
[uwsgi]
# Port on which uwsgi will be listening.
http-socket = 0.0.0.0:8000

# Move to the directory wher the django files are.
chdir = /path-to-django-project-directory/

# Use the wsgi file provided with the django project.
wsgi-file = wsgi.py

# Setup default number of processes and threads per process.
master = true
process = 2
threads = 2

# Drop privielges and don't run as root.
uid = 1000
gid = 1000

# Setup the django_q related worker processes.
attach-daemon = ./manage.py qcluster

# Setup the request log.
req-logger = file:/path-to-logs/logs/uwsgi.log

# Log cron seperately.
logger = cron file:/path-to-logs/logs/uwsgi-cron.log
log-route = cron uwsgi-cron

# Log qcluster commands seperately.
logger = qcluster file:/path-to-logs/logs/uwsgi-qcluster.log
log-route = qcluster uwsgi-daemons

# Last log and it logs the rest of the stuff.
logger = file:/path-to-logs/logs/uwsgi-error.log
```

You can run uwsgi using the following command:

```
(venv)$ uwsgi --ini /path/to/uwsgi.ini
```

Note that in the above configuration, there is a command called `python manage.py qcluster` which run the `django-q` processes. You can remove this from here if you want to manage this yourself via some other init process.

Have a look at **'uwsgi'_** documentation to learn more about different configuration options. One minor optimization that can be done is to replace:

```
http-socket = 0.0.0.0:8000
```

with a more performant option:

```
uwsgi-socket = 0.0.0.0:8000
```

However, this requires support for uwsgi protocol in the webserver. Nginx and Apache, the two most popular webservers have the support and the Nginx configuration below uses `uwsgi-socket`.

Nginx Configuration

You can reverse proxy the requests to uwsgi server using Nginx. Uwsgi has a special protocol called uwsgi protocol that is available in Nginx as a plugin. Add the following configuration to your sites-availbale in Nginx

---

Note: This configuration relies on uwsgi listening on a `uwsgi-socket` instead of a `http-socket`. Please make sure before using this configuration.

Also, if you must use the `http-socket` option for some reason, replace the line below with `uwsgi_pass 0.0.0.0:8000;` with `proxy_pass http://0.0.0.0:8000;`.

```
server {

   listen 443 ssl default_server;
   listen [::]:443 ssl default_server;

   server_name MY_SERVER_NAME;
   location /static/ {
        alias /path-to-djang-staticdir/static/;
   }
   ssl_certificate /path-to-ssl-certs/cert.pem;
   ssl_certificate_key /path-to-ssl-certs/privkey.pem;

   location / {
           include uwsgi_params;
           uwsgi_pass 0.0.0.0:8000;

   }

}
```

Fill in the appropriate paths above in the configuration before using it.

Depending on your setup and experience administering a machine, you can choose one of the following installations methods:

- *Mailman 3 in Docker Containers*
- Installing Mailman from distro packages:
  - *Installing on Debian*
- *Virtualenv Installation*, recommended for Production

CHAPTER 4

Migrating to Mailman 3 from Mailman 2.1

Migrating from Mailman 2.1 to Mailman 3.x

4.1 Migrating from Mailman 2.1 to Mailman 3

This guide covers the migration steps required for porting from Mailman 2.1 to Mailman 3.

4.1.1 Why Upgrade?

Mailman 3 is the new version and actively developed, as compared to 2.1, which is now in maintenance mode and won't receive any feature updates.

Some of the reasons that might convince you to migrate to Mailman 3 include

- a well tested and rewritten code base,
- support for REST API (if you need to integrate with other services)
- support for multiple domains in same installation (without listname collisions)
- A real database backend for settings and configuration
- Modern Web UI for list administration and subscription management
- Support for social logins (and possibly LDAP/openid integration)
- Much improved and interactive archiver/forum with ability to reply from within the Archiver

4.1.2 Other considerations

Before you upgrade, you should consider a few things like:

- URLs to archived messages will break, unless you take extra steps to keep them around. Upgrade mechanism makes sure to import all your archived messages in the new system, but, all the URLs to the new messages are going to be different.

  If you need your URLs for Mailman 2 archives to work, you can keep the HTML files generated for the archives around and your web server configuration for the archives intact (possibly with a notice to viewers that it is now a read-only archive, see this list for example).

- The above mechanism won't work for private archives since the archives are gated with password and without a Mailman 2 list, there is no password. You can however import them to Mailman 3.
- Some configuration and settings aren't available in Mailman 3's UI yet, so even though those settings will be migrated to Mailman 3, you may not be able to change them from the Web UI today. All of those settings should be exposed in the UI very soon.
- Mailman 3 doesn't have support for bounce processing yet, but it is on the roadmap.

4.1.3 Before you upgrade

- Make sure that you have no pending subscription requests as those will not be ported over to Mailman3.
- Make sure that you don't have any pending emails in the digest mbox, if there are, you can force send the digests before moving to Mailman 3, as those won't be upgraded to Mailman3.

4.1.4 Upgrade strategy

As of now, there isn't a turn key solution to migrate all your lists from Mailman 2 to Mailman 3, although, the process is fairly automated. You need shell access to your installation in order to perform the upgrade.

Mailman 3 is split in two main parts, the Core engine which includes all the lists and their configuration and the Web UI, which includes the archiver and UI for information in Core.

Before you start with migration, you need a working Mailman 3 instance, you can see here for recommendations on installing Mailman.

Some key information that you should know about your Mailman 2 and Mailman 3 installations before you do the migration:

- Location of list configuration in Mailman 2: This is typically at `$var_prefix/lists/LISTNAME/config.pck` where `$var_prefix` is an installation dependent directory which is typically `/usr/local/mailman` or `/var/lib/mailman`.
- Location of list archives in Mailman 2: This is typically at Mailman 2: This is at `$var_prefix/archives/private/LISTNAME.mbox/LISTNAME.mbox` where `$var_prefix` is as above.
- Location of the `bin/` commands in Mailman 2: This is at `$prefix/bin` where `$prefix` is an installation dependent directory which is typically `/usr/local/mailman` or `/usr/lib/mailman`

Steps for migration:

- Create the list you are trying to migrate in Mailman 3, for the purposes of this guide, we will call it `foo-list@example.com`
- Migrate the list configuration from Mailman 2 to Mailman 3 by running the following command[1]

[1] If the Mailman 2 list does not predate Mailman 2.1, its LISTNAME.mbox file is probably in good shape, but all mailboxes should be checked for defects before importing. Certain defects such as missing Message-ID: headers or missing or unparseable Date: headers will be corrected or ignored by the import process. The one defect that will definitely cause problems is lines beginning with From in message bodies. These will be seen as the start of a new message. There is a Mailman 2 script at $prefix/bin/cleanarch. That can identify and fix most such lines, but it is not perfect. Cases have been observed where a post includes in its body a copy of some other message including the From separator. This will normally occur only on an old list which includes spam messages or other email problems in its subject matter, but is something to be aware of.

```
$ mailman import21 foo-list@example.com /path/to/mailman2/foo-list/config.pck
```

- Migrate the list archives from Mailman 2 to Mailman 3 by running the following command:

```
$ python manage.py hyperkitty_import -l foo-list@example.com $var_prefix/archives/
↪private/foo-list.mbox/foo-list.mbox
```

  After this, you will need to rebuild the index for this list:

```
$ python manage.py update_index_one_list foo-list@example.com
```

  After this, you should be able to search your messages in Hyperkitty.

- Delete Mailman 2 list:

```
$ $prefix/bin/rmlist foo-list
```

After this, you may need some additional steps based on if you want to keep your old archives around or not. To add a notice to your list archives, you can edit `index.html` available at the root of your mailing list archives.

You may also want to add a redirect at old list page to automatically redirect your users to Mailman3 list-info page. However, this process is fairly manual depending on type of webserver you are using.

CHAPTER 5

Configuring Mailman 3

Mailman 3 can be configured in a wide variety of ways. After you have installed Mailman 3, you can now proceed to configure it for production use.

5.1 Configuring Mailman Core

If you are here, it means that you have successfully installed complete Mailman 3 or some of its components. To verify that you have Mailman Core installed try:

```
$ mailman info
GNU Mailman 3.1.0 (Between The Wheels)
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170118]
config file: /etc/mailman.cfg
db url: sqlite:////var/lib/mailman/data/mailman.db
devmode: DISABLED
REST root url: http://localhost:8001/3.1/
REST credentials: restadmin:restpass
```

There are a few important parameters in the above command that can help you debug problems if they occur. Most important of which is the configuration file that is being used, pointed to by `config_file` that you see above.

Mailman 3 uses a total of two main configuration files, the first one is for Mailman Core and the other is Django's `settings.py` used to deploy the web front end.

`mailman.cfg` is the main configuration file for Mailman Core. Core will look for this file at several different places in order like:

- Filesystem Path pointed to by `MAILMAN_CONFIG_FILE` environment variable.
- `mailman.cfg` in the current working directory.
- `var/etc/mailman.cfg` relative to the current working directory.
- `$HOME/.mailman.cfg`

- `/etc/mailman.cfg`
- `../../etc/mailman.cfg` relative to the working directory of `argv[0]` i.e. the working directory of the `mailman` command script.

Alternatively, you can specify the absolute path of the configuration file using the `-C` flag in the command line.

5.1.1 Configuring Filesystem Paths

Mailman by default puts all the configuration, logs and lock files in the current working directory under a `var` directory. Each `var` directory belongs to either a running or previously run Mailman Core instance.

While it is often confusing to several people, the reason it is default is because it helps a lot with debugging and testing. In production you want a static location for your configuration, locks and logs.

All the different paths that Mailman creates are, by default, configured relative to `var_dir`. For example, `paths.fhs` which is based on Filesystem Hierarchy Standard . To use it add the following to your configuration:

```
[mailman]
layout: fhs
```

This is equivalent to the following configuration:

```
bin_dir: /sbin
var_dir: /var/lib/mailman
queue_dir: /var/spool/mailman
log_dir: /var/log/mailman
lock_dir: /var/lock/mailman
etc_dir: /etc
ext_dir: /etc/mailman.d
pid_file: /var/run/mailman/master.pid
```

Most of the path names are self explanatory, like all the runnable scripts are put under */sbin* and all the logs are placed under */var/log/mailman*.

If you want a custom directory layout you can define a new layout and use that instead. For example:

```
[paths.custom]
var_dir: /home/user/.mailman

[mailman]
layout: custom
```

If you are installing Mailman from source (or using Pip), it is recommended to use `paths.local`. To use it, add the following to your `mailman.cfg`:

```
[mailman]
layout: local
```

It is equivalent to the following configuration:

```
var_dir: /var/tmp/mailman
bin_dir: $argv
log_dir: $var_dir/logs
lock_dir: $var_dir/locks
data_dir: $var_dir/data
cache_dir: $var_dir/cache
etc_dir: $var_dir/etc
```

(continues on next page)

(continued from previous page)

```
messages_dir: $var_dir/messages
archive_dir: $var_dir/archives
template_dir: $var_dir/templates
pid_file: $var_dir/master.pid
lock_file: $lock_dir/master.lck
```

5.1.2 Configuring MTA

The first step to get Core to work is to enable it to talk to the Mail Transport Agent or MTA. It supports various different MTAs.

Postfix Core automatically generated transport maps(`postfix_lmtp` and `postfix_lmtp`) to be used by Postfix at `var/data/`. To configure Postfix, add the following configuration to `main.cf`:

```
# Support the default VERP delimiter.
recipient_delimiter = +
unknown_local_recipient_reject_code = 550
owner_request_special = no
transport_maps =
    hash:/path-to-mailman/var/data/postfix_lmtp
local_recipient_maps =
    hash:/path-to-mailman/var/data/postfix_lmtp
relay_domains =
    hash:/path-to-mailman/var/data/postfix_domains
```

Mailman's `var` directory can vary according your source of installation. Please refer to the documentation provided with your source or ask Mailman Developers at mailman-developers@python.org.

To configure Core to use Postfix, add the following configuration to your `mailman.cfg` configuration, be sure to replace `mail.example.com` with your email domain:

```
[mta]
incoming: mailman.mta.postfix.LMTP
outgoing: mailman.mta.deliver.deliver
lmtp_host: mail.example.com
lmtp_port: 8024
smtp_host: mail.example.com
smtp_port: 25
```

Exim4 To setup Exim4, add the following files to your configuration as file names mentioned above:

```
# /etc/exim4/conf.d/main/25_mm3_macros
# The colon-separated list of domains served by Mailman.
domainlist mm_domains=list.example.net

MM3_LMTP_PORT=8024

# MM3_HOME must be set to Mailman's var directory, wherever it is
# according to your installation.
MM3_HOME=/path-to-mailman/var/
MM3_UID=list
MM3_GID=list

##################################################################
# The configuration below is boilerplate:
```

(continues on next page)

(continued from previous page)

```
# you should not need to change it.

# The path to the list receipt (used as the required file when
# matching list addresses)
MM3_LISTCHK=MM3_HOME/lists/${local_part}.${domain}

# /etc/exim4/conf.d/router/455_mm3_router
mailman3_router:
driver = accept
domains = +mm_domains
require_files = MM3_LISTCHK
local_part_suffix_optional
local_part_suffix = \
-bounces   : -bounces+* : \
-confirm   : -confirm+* : \
-join      : -leave     : \
-owner     : -request   : \
-subscribe : -unsubscribe
transport = mailman3_transport

# /etc/exim4/conf.d/transport/55_mm3_transport
mailman3_transport:
driver = smtp
protocol = lmtp
allow_localhost
hosts = localhost
port = MM3_LMTP_PORT
rcpt_include_affixes = true
```

You should note that Exim4 configuration above doesn't support multiple domains like Postfix does. Please change the variables in the configuration above before installing it.

To configure Mailman to use Exim4 add the following to your `mailman.cfg`

```
[mta]
incoming: mailman.mta.exim4.LMTP
outgoing: mailman.mta.deliver.deliver
lmtp_host: mail.example.com
smtp_host: mail.example.com
lmtp_port: 8024
smtp_port: 25
```

Other MTAs Mailman also supports Sendmail and qmail. Please check the Mailman Core documentation for sendmail setup and qmail setup to configure them.

5.1.3 Configuring REST API

Core presents a HTTP Rest API which clients can use to interact with it. Note that this is an administrative API and **must** not be exposed to the public internet. It has a very basic HTTP Basic Authentication which can be configured using the configuration below:

```
[webservice]
hostname: localhost
port: 8001
use_https: no
```

(continues on next page)

(continued from previous page)

```
admin_user: restadmin
admin_pass: restpass
api_version: 3.1
```

If you need to bind to a different host or port, you can change the configuration above according to you needs.

There are several different other ways to configure Core and you can find more details about it in the Core's documentation.

5.1.4 Configure Templates

You can configure templates for headers, footers and automated emails generated by Mailman. By default, the only templates available are in English, but you can add them other languages too. You can add templates to the var directory of Mailman, which will be picked up by Core.

- sitewide templates can be put in $var/templates/site/LC/,
- domain specific templates in var/templates/domains/DOMAIN/LC/
- list specific templates invar/templates/lists/LIST-ID/LC/

List overrides domain overrides site overrides defaults.

5.1.5 Configuring Databases

Core supports Sqlite, PostgreSQL and MySQL and each of them are tested. By default, without any configuraiton, Core uses Sqlite database. For a production use, a more efficient database like PostgreSQL or MySQL is highly recommended. Please have a look at the setting up your database in Core's documentation for the same.

5.1.6 Configuring Cron Jobs

Depending on your configuration, Core has some periodic tasks that need to be run using job schedulers like `cron`. Right now, the two tasks required to be run routinely for Core are periodic sending of digests for mailing lists that have `digest_send_periodic` set to `true` and periodic sending of notices of pending requests to list moderators. These can be customized to send at any periodic time, the following format sends out digests at midnight:

```
@daily /path/to/mailman digests --periodic
```

This format will send moderator notices at 08:00:

```
0 8 * * * /path/to/mailman notify
```

5.2 Configure Web Frontend

Mailman 3 has a web frontend which can be used to administer the Core, manage subscriptions and view the Archives. There are two principal components, Postorius, the Mailman's web frontend, and Hyperkitty, the Mailman's official Archiver.

Both Postorius and Hyperkitty are built atop Django, a Python based web framework.

See also:

What is Django?

Django is generally configured using a python module called `settings.py` which is usually present at the root of the Django project. It doesn't have to be at the root of the project, but anywhere *importable* by python.

Assuming that you have already cloned or installed the Django project and know their location, we can now start configuring Mailman Components. If you haven't have a look at *Setting up Django Project* .

5.2.1 Setting up Admin Account

To create a superuser account in Django, run the following interactive command:

```
$ cd mailman-suite_project
$ python manage.py createsuperuser
```

Before you can login with this user, you have to configure Django to send emails, so that it can verify the email provided for the superuser account above.

In Postorius, a superuser is assumed to be the Site Owner and has access to all the domains, mailing lists and their settings. List Owners and Moderators can be added based on per-list basis and don't need to have a superuser account in Django.

5.2.2 Setting up Email (required)

It is important that Django be able to send emails to verify the addresses that are subscribing to the Mailman. This configuration is separate from what is done is Core. Please have a look at how to setup email backend for django. A simple configuration would look something like this for a mail server listening on localhost:

```
# To be added to Django's settings.py

EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'localhost'
EMAIL_PORT = 25
EMAIL_HOST_USER = <username>
EMAIL_HOST_PASSWORD = <password>
```

Note: Mailman Suite project in Gitlab disables sending of emails when `DEBUG=True` is set in `settings.py`, and instead prints the emails to a directory `emails` under `mailman-suite_project`. If you don't see any outgoing emails, please check your `settings.py` and set `DEBUG=False`.

If you want to sent out emails along with `DEBUG=True`, you can remove the conditional setting at the bottom of `settings.py` in `mailman-suite` project.

Read more about `DEBUG`.

Here are some settings that determine how your emails from them look like:

- `DEFAULT_FROM_EMAIL` : This is the default address that used used as the `FROM` header in all the emails from your django installation.
- `SERVER_EMAIL` : This is the address from which the errors emails will be sent to you.

Note: In order to make django send mails, you need to change `EMAIL_BACKEND` from `django.core.mail.backends.console.EmailBackend` to `django.core.mail.backends.smtp.EmailBackend` in the mailman-suite project, if you are using that.

Note that both of these are general Django related settings and will affect other parts of your Django installation too.

5.2.3 Running the task queue (required)

Hyperkitty uses django_q as a task queue. It supports various different backends like Redis, Disque, IronMQ, SQS etc. Please check the documentation to better understand how to configure it. The most basic setup where it uses Django orm as the queue can be configured using the settings below:

```
Q_CLUSTER = {
    'timeout': 300,
    'save_limit': 100,
    'orm': 'default',
}
```

You will also have to run *python manage.py qcluster* command along with Hyperkitty. See Hyperkitty's docs about asynchronous tasks.

5.2.4 Enable full text search (required)

Hyperkitty uses django-haystack for its full text search. There are several full text engines that can be used. See django-haystack documentation for more details about the different backend engines that it supports. For the most basic setup, you can use *whoosh* backend. To install the library try:

```
(venv)$ pip install whoosh
```

Then add the following configuration to the Django's settings.py to enable whoosh engine for full text search.:

```
HAYSTACK_CONNECTIONS = {
      'default': {
      'ENGINE': 'haystack.backends.whoosh_backend.WhooshEngine',
      'PATH': os.path.join(BASE_DIR, "fulltext_index"),
      # You can also use the Xapian engine, it's faster and more accurate,
      # but requires another library.
      # http://django-haystack.readthedocs.io/en/v2.4.1/installing_search_engines.html
↪#xapian
      # Example configuration for Xapian:
      #'ENGINE': 'xapian_backend.XapianEngine'
      },
}
```

5.2.5 Scheduled Tasks (required)

There are some routine tasks that need to be run alongside Django, most of which are meant to do some specific routine functions in Hyperkitty. You can add the following to your crontab run them using other cron-like interfaces.:

```
# This goes in /etc/cron.d/mailman

# Replace "apache" by your webserver user ("www-data" on Debian systems) and
# set the path to the Django project directory

@hourly  apache  django-admin runjobs hourly  --pythonpath /path/to/project --
↪settings settings
@daily   apache  django-admin runjobs daily   --pythonpath /path/to/project --
↪settings settings
```

(continues on next page)

(continued from previous page)

```
@weekly  apache  django-admin runjobs weekly  --pythonpath /path/to/project --
↪settings settings
@monthly apache  django-admin runjobs monthly --pythonpath /path/to/project --
↪settings settings
@yearly  apache  django-admin runjobs yearly  --pythonpath /path/to/project --
↪settings settings
* * * * *  apache  django-admin runjobs minutely  --pythonpath /path/to/project --
↪settings settings
2,17,32,47 * * * * apache  django-admin runjobs quarter_hourly --pythonpath /path/to/
↪project --settings settings
```

To Check what jobs do exist and will run on scheduled time, you can run:

```
$ python manage.py runjobs -l
Job List: 11 jobs
 appname           - jobname                - when     - help
------------------------------------------------------------------------------
 django_extensions - cache_cleanup          - daily    - Cache (db) cleanup Job
 django_extensions - daily_cleanup          - daily    - Django Daily Cleanup Job
 hyperkitty        - empty_threads          - monthly  - Remove empty threads
 hyperkitty        - new_lists_from_mailman - hourly   - Import new lists from Mailman
 hyperkitty        - orphan_emails          - daily    - Reattach orphan emails
 hyperkitty        - recent_threads_cache   - daily    - Refresh the recent threads␣
↪cache
 hyperkitty        - sync_mailman           - daily    - Sync user and list␣
↪properties with Mailman
 hyperkitty        - thread_order_depth     - yearly   - Compute thread order and␣
↪depth for all threads
 hyperkitty        - thread_starting_email  - hourly   - Find the starting email when␣
↪it is missing
 hyperkitty        - update_and_clean_index - monthly  - Update the full-text index␣
↪and clean old entries
 hyperkitty        - update_index           - minutely - Update the full-text index
```

5.2.6 Setting up Database (required)

Django supports a wide variety of databases and you can have a look at the documentation for different options and ways to configure your database to use with Django.

5.2.7 Configure Login to Django

Postorius & Hyperkitty both use django-allauth for authentication because it supports a wire variety of social providers and also allows users to sign up with their email if they desire.

Note that if you have any problems with the account signup/signin related emails, you should look the documentation for django-allauth.

Some of the very basic settings that are required to be set for Postorius & Hyperkitty to work are mentioned below:

- `ACCOUNT_AUTHENTICATION_METHOD` = "username_email"
- `ACCOUNT_EMAIL_REQUIRED` = True
- `ACCOUNT_EMAIL_VERIFICATION` = "mandatory"
- `ACCOUNT_DEFAULT_HTTP_PROTOCOL` = "http"

- ACCOUNT_UNIQUE_EMAIL = True

5.2.8 Configure Social Login

Yes, so the way social login works in Mailman is by using a library called django-allauth. There are a few social providers already "enabled" in the Django configuration for the container images, to add more you would have change INSTALLED_APPS in your settings_local.py. There is no way to "add" any apps as the one defined in settings_local.py will override the original one (they are just python variables), so you'd have to copy the entire INSTALLED_APPS3 and then add whatever new ones you want.

To see a list of all the providers, please have a look at the documentation of django-allauth. Make sure that the one you choose provides "email" as part of user data, otherwise it won't work with Mailman. e.g. Twitter doesn't give out emails.

Now to be able to use any provider, you'd have to configure them in your site. django-allauth documentation2 does provide instructions and direct urls to configure each one. To summarize the documentation here are the steps:

Go to your Django admin interface (located at '/admin', like http://example.com/admin) and login with whatever superuser you created. Scroll down to the section "SOCIAL ACCOUNTS" and enter the one saying "Social applications".The relative url for that is '/admin/socialaccount/socialapp/' Click on "ADD SOCIAL APPLICATION" button on the top left of the page. Fill out the details from your social provider and choose the "Sites" you want to use for that particular social provider. You can have separate credentials for separate "Sites".

That should be all. Make sure when you request client_id and client_secret from the social providers, you provide the correct callback_url. The documentation for django-allauth has the correct url for each provider, but it looks basically like

http://mysite.com/accounts/<provider>/login/callback

and you replace <provider> with amazon, facebook, google or whatever provider.

5.2.9 Configure Postorius & Hyperkitty

Here are the parameters that will affect Postorius and Hyperkitty will function. These parameters are configured in your Django's `settings.py`.

MAILMAN_REST_API_URL Complete URL to the Core's REST API Server. Usually, Mailman Core listens on port 8001 for REST API. e.g. `http://localhost:8001/`

MAILMAN_REST_API_USER Username for the Core's REST API, default value in core is 'restadmin' if not set.

MAILMAN_REST_API_PASS Password for Mailman REST API User, default value in core is 'restpass' if not set.

MAILMAN_ARCHIVER_KEY The Key used to authenticate the emails for archiving in Hyperkitty. Its value should be exactly same as set in Core.

Also note that the value in `settings.py` will be within single quotes, but in `mailman.cfg` it will be without any quotes.

FILTER_VHOST Filter the list of available lists in Postorius and Hyperkitty depending on the domain they are being currently served from. Mailman 3 supports multiple domains in a single installation.

LOGIN_URL 'account_login'

LOGIN_REDIRECT_URL 'list_index'

LOGOUT_URL 'account_logout'

STATICFILE_FINDERS Add `compressor.finders.CompressorFinder` to your `STATICFILES_FINDERS`.

See also STATICFILE_FINDERS.

COMPRESS_PRECOMPILERS This setting is for django-compressor which is used here to compile and compress static files:

```
COMPRESS_PRECOMPILERS = (
  ('text/x-scss', 'sassc -t compressed {infile} {outfile}'),
  ('text/x-sass', 'sassc -t compressed {infile} {outfile}'),
)
```

POSTORIUS_TEMPLATE_BASE_URL should be set to URL that Postorius is expected to be listening at. You should set it to whatever URL your WSGI server is listening at.

CHAPTER 6

The User Guide

6.1 GNU Mailman 3.1 - List Member Manual

Mailman is free software for managing electronic mail discussion and e-newsletter lists. Mailman is integrated with the web, making it easy for users to manage their accounts and for list owners to administer their lists. Mailman supports built-in archiving, automatic bounce processing, content filtering, digest delivery, spam filters, and more.

This guide contains instructions for members of Mailman mailing lists so they can learn to use the features available to them. This focuses on the web interface and includes sections on joining and leaving lists, editing options and other subscriber-level tasks.

This guide is written for Mailman 3.1. If you are using Mailman 2.1 (our popular previous stable release), you should see the Mailman 2.1 Members Manual.

6.1.1 Introduction to Mailman Suite

It may be easier to think of Mailman 3 as a single piece of software that does mailing list management, but it's actually a set of interconnected pieces of software under the hood.

The big pieces you care about as a user are as follows:

- **Mailman Core** - This is the "core" of Mailman that handles getting and sending email and stores all your email-related preferences.
- **Postorius** - This is the web interface to Mailman, which allows users to subscribe and unsubscribe from mailing lists and set preferences from the web.
- **Hyperkitty** - This is the archiver for Mailman, which allows users to view and interact with list archives from the web.

These were divided up so that you could replace pieces of Mailman as needed. For example, if a site already had a user settings page, you might want to run a modified version of Postorius so that users can set all their preferences in one place, or you might want to replace the web interface entirely.

This document is going to assume you're using these pieces together in their default states. If your Mailman installation is customized or has replaced any of these pieces, you'll need to adjust the instructions accordingly.

Mailing List Terminology

- A "**post**" typically denotes a message sent to a mailing list.
- People who are part of an electronic mailing list are usually called the list's "**members**" or "**subscribers**."
- "**List administrators**" are the people in charge of maintaining that one list. Lists may have one or more administrators.
- A list may also have people in charge of reading posts and deciding if they should be sent on to all subscribers. These people are called "**list moderators**."
- Often more than one electronic mailing list will be run using the same piece of software. The person who maintains the software which runs the lists is called the "**site administrator**." Often the same person who acts as site administrator also administrates individual lists.

Translating from our examples to real lists

Often, it's easier to simply give an example than explain exactly how to find the address for your specific list. As such, we'll frequently give examples for a fictional list called LISTNAME@DOMAIN whose list information page can be found at http://WEBSERVER/mailman3/lists/LISTNAME.DOMAIN.

Neither of these are real addresses, but they show the form of a typical list address. The capital letters used for the list-specific parts of each address should make it easier to see what should be changed for each list. Although specific list configurations may be different, you will probably be able to just replace the words given in capital letters with the appropriate values for a real list:

LISTNAME The name of your list.

DOMAIN The name of the mail server which handles that list.

WEBSERVER The name of the web server which handles the list web interface. This may be the same as DOMAIN, and often refers to the same machine, but does not have to be identical.

As a real-life example, if you are interested in the mailman-users list that runs on Mailman3.org, you'd make the following substitutions: LISTNAME=mailman-users, DOMAIN=mailman3.org, WEBSERVER=lists.mailman3.org. As such, for the mailman-users mailing list on mailman3.org, the list information page can be found at the URL http://lists.mailman3.org/mailman3/lists/mailman-users.mailman3.org/. (These, unlike most of the examples given in this document, are real addresses.)

Most lists will have this information stored in the List-* headers. Many mail programs will hide these by default, so you may have to choose to view full headers before you can see these informational headers.

6.1.2 I need to talk to a human!

If you have any trouble with any of these commands, you can always reach the person or people in charge of a list by using the list administrator email address. The list administrators can help you figure out how to do something, subscribe/unsubscribe you, or change your settings if you are unable to change them yourself for some reason. Please remember that many mailing list administrators are volunteers who are donating their spare time to run the list, and they may be very busy people.

This list administrator email address is in the form LISTNAME-owner@DOMAIN, where LISTNAME is the name of the list and DOMAIN is the name of the server. So for an example list called <wolfhounds@example.com> the administrators could be reached using <wolfhounds-owner@example.com>

This email address can also be found on the list information pages.

6.1.3 Making a Mailman account

In order to manage your options and easily subscribe to or unsubscribe from Mailman lists, you typically want to make an account. There is a "sign up" link on most list pages (usually displayed in the upper right of the page), or you can go directly to the sign up interface at a URL that will be something like http://WEBSERVER/accounts/signup/

If you've been subscribed to a list without making an account (because you did this yourself or because your lists were migrated from a Mailman 2.1 setup) you can make an account using the same email address and once you've confirmed that you have access to that email, you'll be able to edit all the associated options.

Note that in Mailman 3, you can actually have multiple email addresses associated to the same user account, so you don't need to make many separate accounts to handle your permissions.

6.1.4 Subscribing and Unsubscribing

How do I join a list?

1. Go to the list information page for the list you want to join. This will be something like http://WEBSERVER/mailman3/lists/LISTNAME.DOMAIN
2. Usually, it is a good idea to make an account first using the "sign up" option (on the upper right). This account will allow you to change your settings later and make it easier for you to unsubscribe.
3. Once you are signed in, go back to the list information page and there will be a large section labelled "Subscribe to this list" where you can choose the email address you want to use and optionally choose a display name. Fill out this form and click the "subscribe" button.
4. Alternatively, you can also join a list without signing in by using the subscribe boxes at the bottom of the list information page. If you need to edit your settings later, you will need to create an account associated with the same email address.

How do I leave a list?

1. Go to the list information page for the list you want to leave. This will be something like http://WEBSERVER/mailman3/lists/LISTNAME.DOMAIN
2. Log in to confirm that you are the owner of the address that you wish to unsubscribe. If you don't already have an account associated with that address, you may need to make one to prove that you are the correct owner of that email address.
3. Once you are logged in, there will be a section marked "Subscription / Unsubscription" that shows you the address which is subscribed to that list and a large "unsubscribe" button you can click to leave the list.

6.1.5 Changing your list settings

Mailman has a number of different settings for list subscribers as follows:

Delivery status Set this option to Enabled to receive messages posted to this mailing list. Set it to Disabled if you want to stay subscribed, but don't want mail delivered to you for a while (e.g. you're going on vacation). If you disable mail delivery, don't forget to re-enable it when you come back; it will not be automatically re-enabled.

Delivery mode If you select summary digests , you'll get posts bundled together (usually one per day but possibly more on busy lists), instead of singly when they're sent. Your mail reader may or may not support MIME digests. In general MIME digests are preferred, but if you have a problem reading them, select plain text digests.

Receive own postings Ordinarily, you will get a copy of every message you post to the list. If you don't want to receive this copy, set this option to No. Note that some mail services (most prominently Gmail) will suppress this copy no matter what you do. If you need to know when your email went through and your mail provider is blocking or removing the copy, you can also use the "Acknowledge posts" option to get a separate email acknowledging your post.

Acknowledge posts Receive acknowledgement mail when you send mail to the list? The options are yes or no. This is useful if your mail provider is making it difficult for you to know if your post has gone through.

Hide address When someone views the list membership, your email address is normally shown (in an obscured fashion to thwart spam harvesters). If you do not want your email address to show up on this membership roster at all, select Yes for this option.

Avoid Duplicates When you are listed explicitly in the To: or Cc: headers of a list message, you can opt to not receive another copy from the mailing list. Select Yes to avoid receiving copies from the mailing list; select No to receive copies.

Each of these settings can be set globally, per address, or per list. Your per-list settings over-ride the per-address settings which over-ride the global settings.

How do I view my list settings?

To change any settings, you can go to your settings page.

1. Log in to Mailman.
2. Click on the dropdown menu by your username (in the upper right) and select "Mailman settings"
3. Alternatively, the URL for this page will be something like http://WEBSERVER/mailman3/accounts/subscriptions/LISTNAME.DOMAIN

How do I disable/enable my mail delivery?

You may wish to temporarily stop getting messages from the list without having to unsubscribe. If you disable mail delivery, you will no longer receive messages, but will still be a subscriber and will retain your other settings.

To disable or enable mail delivery from the web interface:

1. Log in and go to your list settings page
2. There is an option labelled "Delivery status" which you can enable or disable on your preferences tabs, either globally, per address, or per list subscription.

This can be handy in a many different cases. For example, you could be going on vacation or need a break from the list because you're too busy to read any extra mail. Many mailing lists also allow only subscribers to post to the list, so if you commonly send mail from more than one address (e.g., one address for at home and another for when you're travelling), you may want to have more than one subscribed account, but have only one of them actually receive mail. You can also use this as a way to read private archives even on a list which may be too busy for you to have sent directly to your mailbox. All you need to do is subscribe, disable mail delivery, and use your password and email to log in to the archives.

How can I start or stop getting the list posts grouped into one big email?

Groups of posts are called "digests" in Mailman. Rather than get messages one at a time, you can get messages grouped together. On a moderately busy list, this typically means you get one email per day, although it may be more or less frequent depending upon the list.

To change your digest settings:

1. Log in and go to your list settings page
2. There is an option labelled "Delivery Mode" which you can set on your preferences tabs, either globally, per address, or per list subscription.

There are a number of different options:

Regular You get an email every time the list sends one out.

Mime Digests MIME is short for Multipurpose Internet Mail Extensions. It is used to send things by email which are not necessarily simple plain text. (For example, MIME would be used if you were sending a picture of your dog to a friend.) A MIME digest has each message as an attachment inside the message, along with a summary table of contents.

Plain Text Digests A plain text digest is a simpler form of digest, which should be readable even in mail readers which don't support MIME. The messages are simply put one after the other into one large text message.

Most modern mail programs do support MIME, so you only need to choose plain text digests if you are having trouble reading the MIME ones.

How do I stop or start getting copies of my own posts?

By default in Mailman, you get a copy of every post you send to the list. Some people like this since it lets them know when the post has gone through and means they have a copy of their own words with the rest of a discussion, but others don't want to bother downloading copies of their own posts.

To receive or stop receiving your own posts:

1. Log in and go to your list settings page
2. There is an option labelled "Receive own postings" which you can set to yes or no on your preferences tabs, either globally, per address, or per list subscription.

Note: This option has no effect if you are receiving digests.

Despite the availability of this option, some mail hosts (such as Gmail) will hide these posts from you. You may wish to see "How can I get Mailman to tell me when my post has been received by the list?" as an alternative solution if your posts are being eaten by your mail host.

How can I get Mailman to tell me when my post has been received by the list?

On most lists, you will simply receive a copy of your mail when it has gone through the list software, but if this is disabled, your list mail delivery is disabled, you use a mail host such as Gmail which blocks copies of your post from being received, or you simply want an extra acknowledgement from the system, this option may be useful to you.

To receive or stop receiving your own posts:

1. Log in and go to your list settings page
2. There is an option labelled "Acknowledge Posts" which you can set to yes or no on your preferences tabs, either globally, per address, or per list subscription.

How can I hide my email address on the subscriber list?

When someone views the list membership, your email address is normally shown (in an obscured fashion to thwart spam harvesters), but you can hide the address if you want:

1. Log in and go to your list settings page
2. There is an option labelled "Hide adresss" which you can set to yes or no on your preferences tabs, either globally, per address, or per list subscription.

Note that this does NOT hide your address in the list archives (if the list has archives) or when it's sent out in emails, so a dedicated spammer could probably get your address in other ways.

How can I avoid getting duplicate messages?

Mailman can't completely stop you from getting duplicate messages, but it can help. One common reason people get multiple copies of a mail is that the sender has used a "group reply" function to send mail to both the list and some number of individuals. If you want to avoid getting these messages, Mailman can be set to check and see if you are in the To: or CC: lines of the message. If your address appears there, then Mailman can be told not to deliver another copy to you.

To avoid duplicates:

1. Log in and go to your list settings page
2. There is an option labelled "Avoid Duplicates" which you can set to yes or no on your preferences tabs, either globally, per address, or per list subscription.

Documentation for Mailman 3 List Owners and Site Administrators is not yet complete.

CHAPTER 7

The Contributor Guide

7.1 Internationalization(i18n)

GNU Mailman project uses Weblate for translations. If you are interested to port Mailman to various languages, please head over to Weblate and create an account to get started with translations.

Weblate has very good documentation on how to use it:

> https://docs.weblate.org/en/latest/user/translating.html

Please do not create Merge Requests for translations since it can create merge conflicts when pulling changes from Weblate and break automation which pulls and pushes changes between Gitlab and Weblate.

If you have existing translated `.po` files, you can reach out to mailman-developers@python.org.

7.1.1 Integration with Weblate

Integration with source control in Gitlab and translation project in Weblate works using webhooks and some scripts.

Weblate supports webhooks for notifications when there are changes to the source control. This allows pulling changes to source strings from Gitlab by adding a webhook notification in Gitlab.

Although Weblate also supports pushing the translations back to the source control routinely, this hasn't been enabled yet as it would require giving out push access to the main repository to a 3rd party service.

Mailman maintainers run a routine script that runs every day to pull changes from Weblate and create Merge Request on the respective projects.

7.2 The Contributor Guide

Mailman 3 consists of a collection of separate-but-linked projects, each of which has its own development setup guide. This makes sense when you want to focus on a single piece of Mailman, but can make setting up the entire Mailman

Suite in one go somewhat confusing. This guide attempts to move all the information currently in the wiki and various package documentation into a single "definitive" guide.

Main package documentation on Readthedocs.io:

- Mailman core start guide
- Mailman core "web ui in 5" guide
- Mailman core "archive in 5"
- Postorius dev guide
- Hyperkitty dev guide

7.2.1 Getting prerequisites

For the most part, setup for each project will download any needed packages. However, you will need a few system packages to be sure you've got the necessary version of Python and its tools, git (to get the source code), postfix (a mail server), and a few other tools that are used during setup.

On Fedora, you probably want to run:

```
$ sudo yum install python3-setuptools python3-virtualenv python3-devel git gcc nodejs-
↪less postfix python3-tox
```

On Debian and Ubuntu, this may be something like:

```
$ sudo apt install python3-setuptools python3-virtualenv python3-dev git gcc node-
↪less nodejs postfix tox
```

On macOS, if you have Homebrew installed, you may run:

```
$ brew install tox python node gcc git
$ npm install --global less
$ # Note: postfix is pre-installed on macOS.
```

If you prefer, you can substitute Exim4 for Postfix. Postfix is the MTA used by most Mailman developers, but we do support Exim 4. (Sendmail support is very much desired, but the Mailman core developers need contributors with Sendmail expertise to help.) For development purposes it doesn't matter, since we will mock all interactions to external MTA.

You will need tox to run tests.

HyperKitty also needs `sassc`. You can install `sassc` using your OS package manager

For Fedora/CentOS:

```
$ sudo dnf install sassc
```

`sassc` is available in the newer versions of Debian(9)/Ubuntu(18.04):

```
$ sudo apt install sassc
```

On macOS with Homebrew, you may run:

```
$ brew install sassc
```

You can also install sassc from source as per their build documentation

7.2.2 Gitlab Setup

We use Gitlab for source code hosting and our CI. You can fork any of the projects you want and start working on it. If you don't already have an account on Gitlab, please create one, you will need that for contributing code or participating in any other way.

We also use Gitlab for code reviews. Our workflow looks very similar to the official Gitlab Workflow. Please remember to enable shared runners on your fork, it will be used to build your code and run unittests on pull requests that you will make. It is mandatory that you have runners enabled before you send any pull requests.

7.2.3 Set up a directory

Setting up the whole Mailman suite means you have to pull code from a bunch of different related repositories. You can put all the code anywhere you want, but you might want to set up a directory to keep all the pieces of mailman together. For example:

```
$ mkdir ~/mailman
# cd ~/mailman
```

For the rest of this development guide, we are going to assume you're using `~/mailman` as your directory, but you can use whatever you want.

7.2.4 Set up virtual environments

All parts of Mailman support only Python 3.5+. For your development, it is advised that you create a virtualenv so that the packages you install don't break any of the system packages using Python.

To create the virtualenv run the following command:

```
$ python3 -m venv venv3
```

To activate a virtualenv, you need to run the appropriate activate script:

```
$ source venv3/bin/activate
```

You *must* use `source` (or `.` if your shell is a pure POSIX shell) everytime you want to activate your development environment. To make your life easier when managing virtualenvs, see virtualenvwrapper .

7.2.5 Set up and run Mailman Core

First, get the code:

```
$ cd ~/mailman
$ git clone https://gitlab.com/mailman/mailman.git
```

To set up Mailman Core, you'll need to switch to your Python 3 virtualenv:

```
$ source venv3/bin/activate
```

Then, go into the mailman directory, run setup, and then run `mailman info` to be sure everything is set up correctly, and that the right settings are in place:

```
$ cd mailman
$ python setup.py develop
$ mailman info
```

You can edit your `mailman.cfg` file to make any necessary changes. By default, during development, it is located at `var/etc/mailman.cfg`. Then start things up:

```
$ mailman start
$ cd ..
```

Note that mailman just makes a `var/` directory wherever you start it and uses that to store your data. This is great for the purposes of testing so you can easily make fresh installs, but might be confusing if you restart your instance later from a different location and don't have your original mailman.db file, or if you start looking around and finding var/ directories everywhere.

Later on, if you need to restart Mailman (i.e. if you get the error "Mailman REST API not available. Please start Mailman core.") then you can also do that by calling the `mailman` executable from the venv as follows:

```
$ ~/mailman/venv3/bin/mailman start
```

Note that the `mailman` executable has several sub-commands. One that is particularly useful for debugging is `mailman shell`.

Note: If you like IPython shell (like I do!), you add the following to your `mailman.cfg`:

```
[shell]
use_ipython: yes

    Also, remember to install ipython using pip::

$ pip install ipython
```

You can run tests for Mailman Core (or any Mailman project) using *tox <https://tox.readthedocs.io/en/latest/>*

```
$ tox -e py37-nocov
```

This requires that you have Python3.7 installed. You change it to `py36-nocov` and `py35-nocov` to run tests with Python 3.6 and 3.5 respectively.

7.2.6 Set up Mailman Client

Get the code:

```
$ cd ~/mailman
$ git clone https://gitlab.com/mailman/mailmanclient.git
```

Then set up mailmanclient:

```
$ cd mailmanclient
$ python setup.py develop
$ cd ..
```

To run the tests:

```
$ tox -e py37
```

7.2.7 Set up Django-mailman3

This package holds the Django libraries and templates used by Postorius and HyperKitty.

Get the code and set it up:

```
$ cd ~/mailman
$ git clone https://gitlab.com/mailman/django-mailman3.git
$ cd django-mailman3
$ python setup.py develop
$ cd ..
```

To run the tests:

```
$ tox -e py37-django21
```

7.2.8 Set up and run Postorius

The Postorius documentation, including a more extensive setup guide, can be found here: http://postorius.readthedocs.org/

Make sure to install mailmanclient and django-mailman3 before setting up Postorius. (If you're following this guide in order, you've just done that.)

Get the code and run setup. Make sure you're in venv which has Python 3.5+ for Postorius:

```
$ cd ~/mailman
$ git clone https://gitlab.com/mailman/postorius.git
$ cd postorius
$ python setup.py develop
$ cd ..
```

Postorius and HyperKitty both come with `example_project` directories with basic configuration so you can try them out. For this tutorial, however, we'll be using a project that combines both instead.

You can run tests using:

```
$ tox -e py37-django21
```

7.2.9 Set up a Fake mail server

To be able to actually receive emails, you need to setup a mail server. Mailman core receives emails over LMTP Protocol, which most of the modern MTAs support. However, setup instructions are provided only for Postfix, Exim4 and qmail. Please refer to the MTA documentation at Mailman Core for the details.

You will also have to add some settings to your django configuration. The setup instructions are provided in django's email documentation.

For development setup, you don't _have_ to install a working MTA. You can add the following to your `mailman.cfg` to make sure that it doesn't try to send emails out:

```
[devmode]
enabled: yes
recipient: you@yourdomain.com

[mta]
smtp_port: 9025
lmtp_port: 9024
incoming: mailman.testing.mta.FakeMTA
```

Also, in Django you can add the following configuration to your `settings.py`:

```
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
```

This writes everything to `stdout`. There are other email backends available to use for testing like `django.core.mail.backends.filebased.EmailBackend` that one can use to write outgoing emails to a file on disk. Please see the docs for other options.

7.2.10 Set up and run HyperKitty

Complete guide here: https://hyperkitty.readthedocs.org/en/latest/development.html

Make sure to install mailmanclient and django-mailman3 before setting up Hyperkitty. (If you're following this guide in order, you've just done that.)

HyperKitty's default configuration uses the Whoosh search engine in the backend. Install `Whoosh` using:

```
$ pip install whoosh
```

Get the code and run setup:

```
$ cd ~/mailman
$ git clone https://gitlab.com/mailman/hyperkitty.git
$ cd hyperkitty
$ python setup.py develop
$ cd ..
```

Postorius and HyperKitty both come with `example_project` directories with basic configuration so you can try them out. By default, they both use port 8000, so if you do want to run both example projects at the same time, do remember that you'll need to specify a different port on the command line for one of them.

You can run tests using:

```
$ tox -e py37-django21
```

However, we're going to run them both in a single Django instance at the end of this guide, so don't worry about ports right now.

7.2.11 Set up mailman-hyperkitty

`mailman-hyperkitty` is the package that actually sends the incoming emails to HyperKitty for archiving. Note that this is one of the components that uses Python 3.

Setting it up:

```
$ cd ~/mailman
$ git clone https://gitlab.com/mailman/mailman-hyperkitty.git
$ cd mailman-hyperkitty
$ python setup.py develop
$ cd ..
```

You'll need to fix the default `mailman-hyperkitty.cfg` file to use the correct url for HyperKitty. If you're running it on http://localhost:8002 then you need to change `base_url` to match that.

You can run tests using:

```
$ tox -e py37-coverage
```

7.2.12 Link Mailman to HyperKitty

Now you have to enable HyperKitty in Mailman. To do that, edit the `mailman.cfg` (in `~/mailman/mailman/var/etc`, or wherever the output of `mailman info` says it is) and add the following config. Note that you need to fill in the absolute path to your `mailman-hyperkitty.cfg` in the configuration below:

```
# mailman.cfg
[archiver.hyperkitty]
class: mailman_hyperkitty.Archiver
enable: yes
configuration: <absolute path to mailman-hyperkitty.cfg>
```

7.2.13 Run the Mailman Suite (combined hyperkitty+postorius)

You can run HyperKitty and Postorius as separate applications, but many developers are going to want to run them on a single server. The configuration files for this are in a repository called mailman-suite.

The first time you run the suite, you will want to set up a superuser account. This is the account you will use in the web interface to set up your first domains. Please enter an email address otherwise the database won't be setup correctly and you will run into errors later:

```
$ cd ~/mailman
$ git clone https://gitlab.com/mailman/mailman-suite.git
$ cd mailman-suite/mailman-suite_project
$ python manage.py migrate
$ python manage.py createsuperuser
```

You'll want to run the following commands in a window where you can leave them running, since it dumps all the django logs to the console:

```
$ python manage.py runserver
```

At this point, you should be able to see Mailman Suite running! In the default setup, you can go to http://127.0.0.1:8000 and start poking around. You should be able to use the superuser account you created to log in and create a domain and then some lists.

The default config file uses a dummy email backend created by this line in settings.py:

```
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
```

Using this backend, all emails will be printed to the Postorius console (rather than sent as email) so you can get the url to verify your email from the console. You can also use FileBackend to write emails to a file on disk.

Don't leave the console email backend configured and running once you get to the point where you want to send real emails, though!

Index