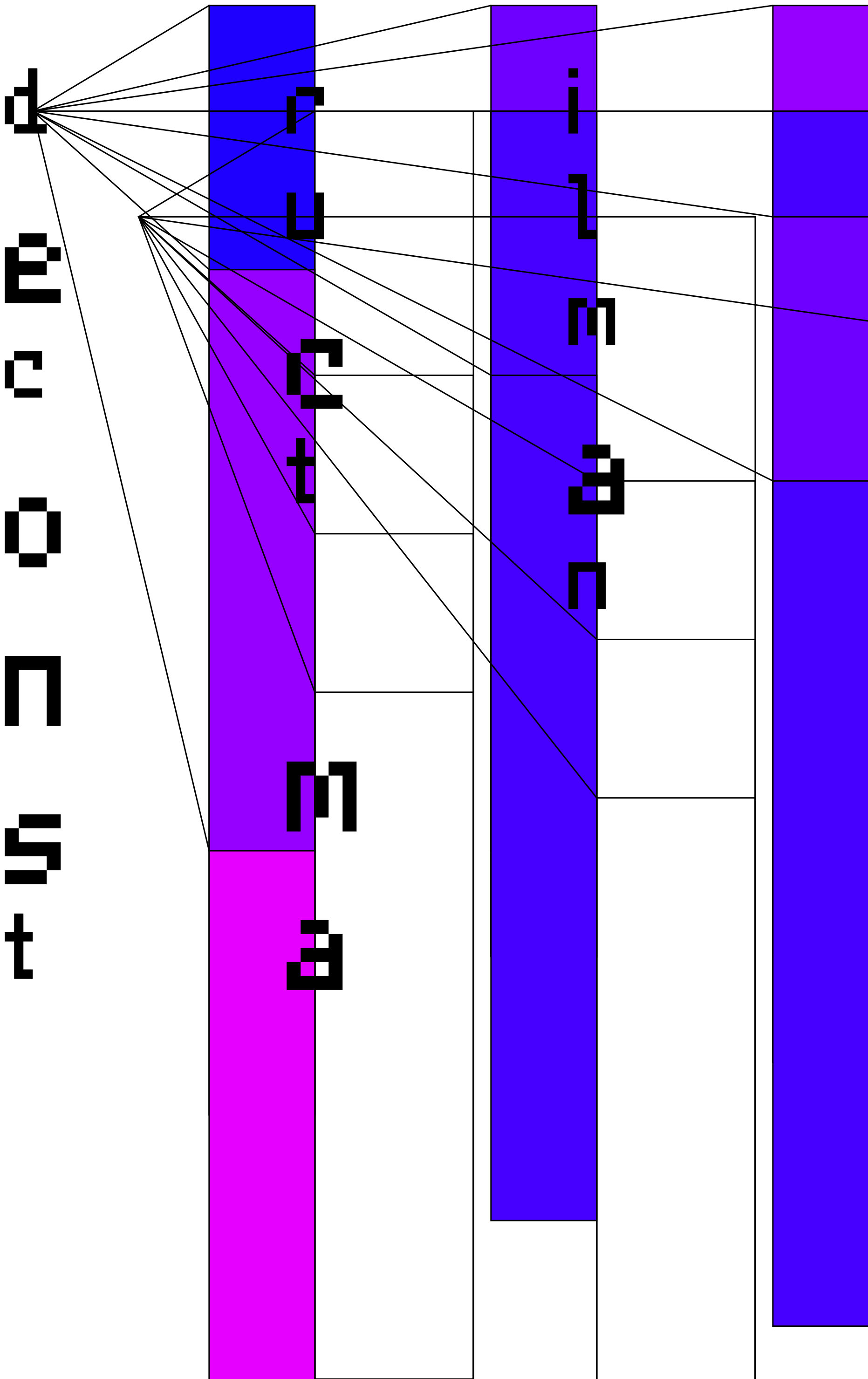
deconst
ructma
ilman

Introduction

This tutorial is all about mailman3 software for creating and managing mailing lists and archives. It goes in-depth on how to install mailman3 in a Debian, but applicable for Ubuntu OS too, and migrate existing lists from older mailman versions. Also it goes over the installation of mailman website with apache2 and gunicorn, and offers some tips for uWSGI too. It assumes that postfix (a Mail Transport Agent, aka MTA) and mailutils are already installed in the system and configured, and the system can send emails, e.g the root user is sending admin related emails. It also assumes that python3, postgresql and apache2 are installed in the system too. Postfix is one of the possible MTA to be configured with mailman3. Detailed steps for configuring a fresh postfix install and few other MTA options, are included in the CHEAT SHEET note 1.
The tutorial is intended specifically for sysadmins and users who are curious of how machines and networks work in general.
Enjoy the geeky reading!

Index

Install Dependency Libraries p.3

System Configurations p.7

Mailman Configurations p.9

Postfix Configurations p.12

Run mailman-web locally p.14

Automate mailman and schedule jobs p.15

Migrate lists p.18

Troubleshooting p.20

CHEAT SHEET p.21-22-23

Install Dependency Libraries

We enter the remote server and become root user to do system updates:

```
$ ssh user@server -i <ssh_key>
```

Once we are on the remote server we do the following (as well all the rest of the commands are meant to be ran on the remote server):

```
$ sudo su
```

We give our password and we do the updates:

```
# apt update && apt upgrade
```

Then we can install some system-wide dependencies:

```
# apt-get install build-essential libssl-dev libffi-dev
```

break down of the above libraries:
build-essential: GNU debugger, g++/GNU compiler and other tools for compiling software.
libssl-dev: portion of OpenSSL which supports TLS protocol and depends on libcrypto, a C API.
libffi-dev: the glue between the interpreter program (python in this case) and the compiled code, for values of arguments to be converted and passed in run-time between the two programs. [Note 3]

Install party goes on:

```
# apt install python3-dev python3-venv lynx
```

break down of the above libraries:

python3-dev: tools for extending the python interpreter and building python modules.

python3-venv: a tool to create virtual environment to isolate a python project's dependencies from the OS main python libraries.

lynx: An HTML to plaintext converter like lynx is required by Mailman Core for converting emails to plaintext.

Then install rust from source, which is needed for python Cryptography library later on.

```
# curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

Ensure that rust is installed:

```
# rust --version
```

The above script will also install cargo, which is a package manager, so it fetches any extra dependencies needed for rust programming, and a builder for rust programs to be compiled.

[Note 2]

We need also sassc: Syntactically Awesome Stylesheets or

Sass is an extension of CSS, which allows using variables, nested rules etc. Here we install a C/C++ flavor needed for Hyperkitty archiver that uses sass to generate its CSS styles. https://sass-lang.com/. For the sass installation for debian, download from source and make a symbolic link to /usr/local/bin:

```
# cd /usr/local/lib [Note 4]
# wget
https://github.com/sass/dart-sass/releases/download/1.32.5/dart-sass-1
.32.5-linux-x64.tar.gz
# tar -xf dart-sass-1.32.5-linux-x64.tar.gz
# chmod -R 755 dart-sass
# ln -s /usr/local/lib/dart-sass/sass /usr/local/bin/sass
# rm -f dart-sass-1.32.5-linux-x64.tar.gz
```

Ref: Note 5

GNU mailman wiki suggests to install also:
Fail2ban for blocking IP addresses that have too many connection failures.

```
# apt install fail2ban
```

Memcached for Django caches in memory, in order to render mailman's front-end UI faster.

```
# apt install memcached
```

After installation check that is running at port 11211 with

```
# service memcached status
```

Output of memcached should show an active status, e.g:

```
   Loaded: loaded (/lib/systemd/system/memcached.service;
enabled; vendor preset: enabled)
    Active: active (running) since Sun 2021-08-29 15:08:13
EEST; 3h 29min ago
```

Later we see how to add the CACHES configuration in the mailman settings.py.

Gettext for supporting internationalization and localization, needed for multilingual environments.

```
# apt install gettext
```

See configuration settings in CHEAT SHEET [Notes 6, 7, 8]

System Configurations

Create a postgresql or mysql database

Here is an example with postgresql. We replace the names according to our likes and available system paths. Enter the postgresql user and initiate the psql shell (psql command requests the postgres user password which was configured during the postgresql setup). When creating the database, we can opt for setting up a tablespace where the database objects are stored. This is handy when we want to migrate the database because we ran out of disk space. Or when we want to optimize performance. E.g, make use of a fast solid state device (SSD) available as a mounted volume.

```
# sudo su postgresql
$ psql
> CREATE TABLESPACE mailman_vol LOCATION
'/ssd1/postgresql/data';
> CREATE DATABASE mailman_db OWNER mailman TABLESPACE
mailman_vol;
> exit;
```

Setup mailman user and virtualenv

```
# useradd -m -d /opt/mailman -s /usr/bin/bash mailman
# sudo su mailman
```

Enter mailman directory, create a virtualenv and activate it:

```
$ cd ; python3 -m venv venv
$ source /opt/mailman/venv/bin/activate
```

Note: we can add the above command in .bashrc under mailman's home, so every time we enter this user, the virtualenv gets activated automagically.

Install Mailman and other python libraries

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

if project connects to a postgresql database then we need also:

```
(venv)$ pip install psycopg2-binary
```

Install front-end UI and archiver

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

mailman-web provides hyperkitty and postorius which are built atop Django, a Python based web framework. It also provides shortcuts to django admin commands. Later we will create a superuser that has all permissions for administering the lists and can enter the admin area via the browser.

Install the following for mailman-web application to be able to talk with apache2 server (here we opt for gunicorn, other option is uWSGI) and a python client for Django to connect to memcached:

```
(venv)$ pip install gunicorn pylibmc
```

Mailman Configurations

Mailman

Exit mailman user and as root we create a new dir:

```
# mkdir -p /etc/mailman3/
```

We make owner of this directory the mailman user:

```
# chown -R mailman:mailman /etc/mailman3
```

and under it, we create the files mailman.cfg and settings.py. Under /opt/mailman/mm we create the file mailman-hyperkitty.cfg ('mm' or other name of our choice, it is a new directory to park hyperkitty configuration and logs).

In the mailman.cfg edit the archiver directive as following:

```
<conf>
[archiver.hyperkitty]
class: mailman_hyperkitty.Archiver
enable: yes
configuration: /opt/mailman/mm/mailman-hyperkitty.cfg
</conf>
```

In mailman-hyperkitty.cfg add the base url of the archives as localhost, and the shared API key, which must be identical to the value in the /etc/mailman3/settings.py

```
<conf>
base_url: http://localhost/archives
api_key: SecretArchiverAPIKey
</conf>
```

For a settings.py sample see in the CHEAT SHEET. The important setting to add, which allows automated correspondence from the site manager:

```
<conf>
DEFAULT_FROM_EMAIL = 'lists@sdomain-name.org' or
'user@localhost'
</conf>
```

And for activating the memcached we need to add:

```
<conf>
'default': {
    'BACKEND':
'django.core.cache.backends.memcached.MemcachedCache',
     'LOCATION': os.environ.get('CACHE_LOCATION','127.0.0.1:11211'),
    }
}
</conf>
```

Note 1: django's global settings is located in /opt/mailman/venv/lib/python3.7/site-packages/django/conf/global_settings.py. These are imported in the /etc/mailman3/settings.py and they get overwritten if declared again in the latter file.

Note 2: If we want to use an external mail service than the localhost, we need also to set:

```
<conf>
EMAIL_HOST = <third-party provider>
EMAIL_PORT = 25
EMAIL_HOST_USER = <username>
EMAIL_HOST_PASSWORD = <password>
Extra options to set in case they are needed:
```

```
EMAIL_TIMEOUT = <time in secs>
EMAIL_USE_TLS = True
OR
EMAIL_USE_SSL = True
</conf>
```

See Note 9, and an example with gmail see note 10.
If we use local postfix email configuration, then the default values in global_settings for localhost are fine.

Postfix configuration

Check open ports in the system. Look if the smtpd port 25 is open. Postfix is the MTA which will relay incoming and outgoing mails to mailman. [Note 11]

```
$ sudo ss -tulpn | grep smtpd
```

if postfix is already installed, edit the /etc/postfix/main.cf:

```
<conf>
inet_interfaces = all
myhostname = server_hostname
mydestination = $myhostname, localhost.$myhostname, localhost
inet_protocols = all
unknown_local_recipient_reject_code = 550
owner_request_special = no
always_add_missing_headers = yes
transport_maps =
   hash:/opt/mailman/mm/var/data/postfix_lmtp
local_recipient_maps =
   hash:/opt/mailman/mm/var/data/postfix_lmtp
relay_domains =
   hash:/opt/mailman/mm/var/data/postfix_domains
default_destination_recipient_limit = 30
default_destination_concurrency_limit = 15
header_checks = regexp:/etc/postfix/header_checks
</conf>
```

Save and close the file.

If we need to install and configure postfix, see note 12.

Now that most of configuration is done, we populate the mailman_db table with the postorius and hyperkitty fields. To do so in django, we run the infamous migrations. Enter mailman user again:

```
# sudo su mailman
(venv) $ cd
(venv) $ mailman-web generate_secret_key
```

Add the value from the above in the /etc/mailman3/settings.py SECRET_KEY and run:

```
(venv)$ mailman-web migrate
```

collect static files for the mailman-web

```
(venv)$ mailman-web collectstatic
```

and create a django admin superuser

```
(venv)$ mailman-web createsuperuser
```

Run mailman-web locally

```
(venv) $ pip install Werkzeug
(venv) $ mailman-web runserver_plus
```

If django runs locally with the above command, we now try and run it with gunicorn

```
(venv) $ gunicorn -c /opt/mailman/gunicorn.py
mailman_web.wsgi:application
```

This runs the django application and listens to port 8000

An example of gunicorn.py:

```
<conf>
import os
import sys
sys.path[0:0] = [
  '/opt/mailman/',
  '/etc/mailman3/'
  ]
os.environ['DJANGO_SETTINGS_MODULE'] = 'settings'
import gunicorn.app.wsgiapp
if __name__ == '__main__':
    sys.exit(gunicorn.app.wsgiapp.run())
</conf>
```

[Notes 13, 14, 15]

If we opt for uWSGI instead of gunicorn see note 16.

Automate mailman and schedule jobs

System services

Run as daemon the above gunicorn (or uwsgi) command by adding it as service to persist reboots. As root do:

```
# vi /lib/systemd/system/gunicorn.service
```

Inside the file add the following:

```
<conf>
[Unit]
Description=GNU Mailman web interfaces
After=network-online.target firewalld.service
Wants=network-online.target

[Service]
PIDFile=/opt/mailman/mm/var/gunicorn.pid
WorkingDirectory=/opt/mailman/
ExecStart=/opt/mailman/venv/bin/gunicorn -c /opt/mailman/gunicorn.py
mailman_web.wsgi:application
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s QUIT $MAINPID
PrivateTmp=true
User=mailman
Group=mailman
Restart=always

[Install]
```

```
WantedBy=multi-user.target
</conf>
```

For Qcluster startup service:

```
# vi /lib/systemd/system/qcluster.service
```

Inside the file you may add:

```
<conf>
[Unit]
Description=HyperKitty async tasks runner
After=network-online.target remote-fs.target

[Service]
ExecStart=/opt/mailman/venv/bin/mailman-web qcluster
User=mailman
Restart=always

[Install]
WantedBy=multi-user.target
</conf>
```

Finally for Mailman core service:

```
# vi /lib/systemd/system/mailman3.service
```

See sample at note 17

Then we reload the services and check their status

```
# systemctl daemon-reload
# systemctl status mailman3
# systemctl status gunicorn
# systemclt status qcluster
```

Cron jobs

As mailman user

(venv) $ crontab -e

```
@hourly   /opt/mailman/mm/bin/django-admin runjobs hourly
@daily    /opt/mailman/mm/bin/django-admin runjobs daily
@weekly   /opt/mailman/mm/bin/django-admin runjobs weekly
@monthly /opt/mailman/mm/bin/django-admin runjobs monthly
@yearly   /opt/mailman/mm/bin/django-admin runjobs yearly
0,15,30,45 * * * * /opt/mailman/mm/bin/django-admin runjobs
quarter_hourly
* * * * * /opt/mailman/mm/bin/django-admin runjobs minutely

# Send periodic digests.
30 3 * * * /opt/mailman/mm/bin/mailman digests --periodic

# Send request reminder for MM 3. Like the checkdbs job for 2.1
0 8 * * * /opt/mailman/mm/bin/mailman notify
```

Apache

Go to /etc/apache2/sites-available and create new configuration for the domain name of our lists. Check if proxy_http is enabled and use it for the localhost mailman-web application. See sample at note 18.

Migrate lists

Create ssh keys for old server where existing lists reside:

```
$ ssh-keygen -t ecdsa -b 521
$ chmod 644 <id_name>.pub # permissions for public key
```

On the new server, edit /etc/ssh/sshd_config and set PasswordAuthentication on. As mailman user, under /opt/mailman, create:

```
$ mkdir -p ./.ssh && touch ./ssh/authorized_keys && chmod 700 ./ssh
```

Copy the public key we created from the old server to the new one (we need mailman's user password):

```
$ ssh-copy-id mailman@remote-host
```

Open the sshd_config file and set PasswordAuthentication no after copying the public key. More info on how to copy ssh keys securely see note 19.

Copy existing list from old to new server's under a temporary directory with rsync [Note 20]:

```
# rsync -avz ssh /var/lib/mailman/lists mailman@<new-server>:~/tmp/
# rsync -avz ssh /var/lib/mailman/archives
mailman@<new-server>:~/tmp
```

Import old lists

From new server, enter mailman user again. [Note 21]

```
(venv) $ mailman create foo-list@<new-lists-domain>

(venv) $ mailman import21 foo-list@<new-lists-domain>
~/tmp/lists/foo-list/config.pck

(venv) $ python manage.py hyperkitty_import -l
foo-list@<new-lists-domain>
~/tmp/archives/private/<list>.mbox/<list>.mbox

(venv) $ mailman-web update_index_one_list
foo-list@<new-lists-domain>
```

Troubleshooting

1. Domain name shows as example.com
First we need to login at the web front-end with the superuser credentials we created before. Then we create a domain for our site from url <lists-domain>/mailman3/domains/
We copy the site_id number and edit the mailman3/settings.py accordingly.
We restart gunicorn + apache2 service. Test by creating a new list from web front-end at <lists-domain>/mailman3/lists/. [Note 22]

2. mailman web command that shows a bunch of cool actions

```
$ mailman-web help
```

3. sass bin executable symbolic link didn't work until I set the right permissions rwxr-xr-x on the dart-sass/sass binary

4. In the /etc/mailman/mm/mailman-hyperkiitty.cfg file add:
base_url: http://localhost:8000/archives/
If you chose other uri for the archives, modify respectively. It should match with the url in apache2 configuration. See below an apache2 sample.
Also add in mailman-hyperkiitty.cfg the same "api_key" as in the /etc/mailman3/settings.py

5. Outgoing mail not sent! It can drive you nuts. Reading AGAIN the mailman's project postfix configuration. In the mailman.cfg, the "lmtp_post" should be the domain name or, 127.0.0.1 if all components are found in the same server.
*** localhost has to be in numeric notation, or postfix doesn't recognize it!
Also add the domain name of the lists in "MAILMAN_ARCHIVER_FROM" in the /etc/mailman3/settings.py.
Note 23

6. Static files not being served with apache2 and proxy_http
Place "ProxyPass /static/ !" should come first in the apache2 server configuration, before the proxies to the localhost:8000 urls.
[Note 24]

CHEAT SHEET

1. Mail Transport Agent options:
https://mailman.readthedocs.io/en/latest/src/mailman/docs/mta.html

2. https://doc.rust-lang.org/cargo/guide/why-cargo-exists.html
3. About libfii https://sourceware.org/libffi/
4. sass https://sass-lang.com/install
5.
https://docs.mailman3.org/en/latest/install/virtualenv.html#installing-dependencies
6. GNU mailman3 wiki
https://wiki.list.org/DOC/Howto_Install_Mailman3_On_Debian10

7. fail2ban configure settings
https://www.howtogeek.com/675010/how-to-secure-your-linux-computer-with-fail2ban/
8. Gettext
https://www.gnu.org/software/gettext/manual/html_node/Concepts.html#Concepts
9. Email settings for django
https://docs.djangoproject.com/en/3.0/ref/settings/?ref=hackernoon.com#email-use-tls

10. Example with gmail setup for django:

https://www.geekinsta.com/send-email-from-django-using-gmail-smtp/

11. Mail server ports
https://serverfault.com/questions/149903/what-ports-to-open-for-mail-server
https://vitux.com/find-open-ports-on-debian/

12. Postfix install and configuration
https://docs.mailman3.org/en/latest/install/virtualenv.html#setup-mta

13. Mailman django migrations
https://docs.mailman3.org/en/latest/install/virtualenv.html#run-database-migrations

14. Extra options for setting up the django mailman_web
https://docs.mailman3.org/en/latest/install/virtualenv.html

15. Gunicorn installation and guide
https://docs.gunicorn.org/en/stable/

16. Setting up with uWSGI
https://docs.mailman3.org/en/latest/install/virtualenv.html#setting-up-a-wsgi-server

17. Mailman service
https://docs.mailman3.org/en/latest/install/virtualenv.html#starting-mailman-automatically

18. Apache2 + gunicorn
https://djangodeployment.readthedocs.io/en/latest/05-static-files.html?highlight=apache2#setting-up-apache
19. https://www.simplified.guide/ssh/copy-public-key
20. https://docs.rc.fas.harvard.edu/kb/rsync/
21. Mailman import commands
https://docs.mailman3.org/en/latest/migration.html#upgrade-strategy
22.
https://docs.mailman3.org/en/latest/faq.html#the-domain-name-displayed-in-hyperkitty-shows-example-com-or-something-else

23. Mail server setup:
https://docs.mailman3.org/projects/mailman/en/latest/src/mailman/docs/mta.html
24. ProxyPass troubleshooting
/https://stackoverflow.com/questions/50621464/deploy-django-static-files-with-apache-
gunicorn
25. mailman commands
https://docs.mailman3.org/projects/mailman/en/latest/src/mailman/commands/docs/commands.html

This manual is created by m4ra and is part of

the tech-zines collection of

psaroskalazines.gr

Layout design is generated in Python by the

author

Code repository at

git.systerserver.net/mara/zine_maker

Cover and colophon font is Casale Two NBP

Content font is KpProgrammerAlternatesNbp

Code blocks font is Helvetica