mailadm Documentation

Release 1.0.0

holger krekel

Mar 31, 2023

Contents

mailadm is automated e-mail account management tooling for use by Delta Chat.

The `mailadm` command line tool allows to add or remove tokens which are typically presented to users as QR tokens. This QR code can then be scanned in the Setup screen from all Delta Chat apps. After scanning the user is asked if they want to create a temporary account.

The account creation happens via the mailadm web interface and creates a random user id (the local part of an e-mail).

Mailadm keeps all configuration, token and user state in a single sqlite database. It comes with an example install script that can be modified for distributions.

CHAPTER 1

Quick Start

Note: To use mailadm, you need admin access to a Mailcow instance. You can run mailadm as a docker container, either on the same machine as mailcow or somewhere else.

First get a git copy of the mailadm repository and change into it.

```
$ git clone https://github.com/deltachat/mailadm
$ cd mailadm
$ mkdir docker-data
```

Now you need to configure some environment variables in a file called `.env`:

- `MAIL_DOMAIN`: the domain part of the email addresses your users will have.
- `WEB_ENDPOINT`: the web endpoint of mailadm; make sure mailadm receives POST requests at this address.
- `MAILCOW_ENDPOINT`: the API endpoint of your mailcow instance.
- `MAILCOW_TOKEN`: the access token for the mailcow API; you can generate it in the mailcow admin interface.

In the end, your `.env` file should look similar to this:

```
MAIL_DOMAIN=example.org
WEB_ENDPOINT=http://mailadm.example.org/new_email
MAILCOW_ENDPOINT=https://mailcow-web.example.org/api/v1/
MAILCOW_TOKEN=932848-324B2E-787E98-FCA29D-89789A
```

Now you can build the docker container:

```
$ sudo docker build . -t mailadm-mailcow
```

Initialize the database with the configuration from `.env`:

```
$ scripts/mailadm.sh init
```

And setup the bot mailadm will use to receive commands and support requests from your users:

```
$ scripts/mailadm.sh setup-bot
```

Then you are asked to scan a QR code to join the Admin Group, a verified Delta Chat group. Anyone in the group issue commands to mailadm via Delta Chat. You can send "/help" to the group to learn how to use it.

Now, as everything is configured, we can start the mailadm container for good:

```
$ sudo docker run -d -p 3691:3691 --restart=unless-stopped --mount type=bind,source=
↪$PWD/docker-data,target=/mailadm/docker-data --name mailadm mailadm-mailcow␣
↪gunicorn --timeout 60 -b :3691 -w 4 mailadm.app:app
```

Note: As the web endpoint also transmits passwords, it is highly recommended to protect the WEB_ENDPOINT with HTTPS, for example through an nginx reverse proxy. In this case, WEB_ENDPOINT needs to be the outward facing address, in this example maybe something like *https://mailadm.example.org/new_email/*.

CHAPTER 2

First Steps

`mailadm` CLI commands are run inside the docker container - that means that we need to type `scripts/mailadm.sh` in front of every `mailadm` command. This can be abbreviated by running `alias mailadm="$PWD/scripts/mailadm.sh"` once, and adding the line to your `~/.bashrc`:

```
$ echo "alias mailadm=$PWD/scripts/mailadm.sh" >> ~/.bashrc
```

These docs assume that you have this alias configured.

2.1 Adding a First Token and User

You can now add a first token:

```
$ mailadm add-token oneday --expiry 1d --prefix="tmp."
added token 'oneday'
token:oneday
  prefix = tmp.
  expiry = 1d
  maxuse = 50
  usecount = 0
  token  = 1d_r84EW3N8hEKk
  http://localhost:3691/new_email?t=1d_r84EW3N8hEKk&n=oneday
  DCACCOUNT:http://localhost:3691/new_email?t=1d_r84EW3N8hEKk&n=oneday
```

Then we can add a user:

```
$ mailadm add-user --token oneday tmp.12345@example.org
added addr 'tmp.12345@example.org' with token 'oneday'
```

2.2 Testing the Web App

Let's find out the URL again for creating new users:

```
$ mailadm list-tokens
token:oneday
  prefix = tmp.
  expiry = 1d
  maxuse = 50
  usecount = 1
  token  = 1d_r84EW3N8hEKk
  http://localhost:3691/?t=1d_r84EW3N8hEKk&n=oneday
  DCACCOUNT:http://localhost:3691/new_email?t=1d_r84EW3N8hEKk&n=oneday
```

The second last line is the one we can use with curl:

```
$ curl -X POST 'http://localhost:3691/?t=1d_r84EW3N8hEKk&n=oneday'
{"email":"tmp.km5y5@example.org","expiry":"1d","password":"cg8VL5f0jH2U","ttl":86400}
```

We got an e-mail account through the web API, nice.

Note that we are using a localhost-url whereas in reality your `WEB_ENDPOINT` will be a full https-URL. All in all the architecture looks pretty much like this:

```
Delta Chat
    |
    | scans QR code; sends POST request
    V
NGINX Reverse Proxy (Let's Encrypt)
    |
    | proxy_pass
    V
gunicorn Python HTTP Server (e.g. in Docker)
    |
    | executes
    V
mailadm web API ------> creates user in mailadm.db
    |
    | HTTP POST request /api/v1/add/mailbox
    V
mailcow API
    |
    | creates account
    V
mailcow user management
```

CHAPTER 3

The Bot Interface

You don't have to login with SSH every time you want to create tokens. You can also use the bot interface to give commands to mailadm in a verified Delta group, the "admin group chat".

During installation, you are asked to scan a QR code to join the Admin Group, a verified Delta Chat group. Anyone in the group issue commands to mailadm via Delta Chat. You can send "/help" to the group to learn how to use it.

3.1 Re-Initializing the Admin Group

If you ever lose access to the Admin Group, or want to change the email account the bot uses, you can just re-run `mailadm setup-bot` to invalidate the old Admin Group and create a new one.

By default your bot is called `mailadm@yourdomain.tld`, but you can use the `mailadm setup-bot --email` command if you want to use a different address. If you want to use an existing account for the mailadm bot, you can specify credentials with `--email` and `--password`. If it is an existing email account, it doesn't need to be on your mailcow server.

The bot is initialized during installation. If you want to re-setup the bot account or admin group, you need to stop mailadm first:

```
$ sudo docker stop mailadm
$ mailadm setup-bot
$ sudo docker start mailadm
```

3.2 QR Code Generation

Once you have mailadm configured and integrated with nginx and mailcow, you can generate a QR code:

```
$ mailadm gen-qr oneday
dcaccount-testrun.org-oneday.png written for token 'oneday'
```

This creates a .png file with the QR code in the `docker-data/` directory. Now you can download it to your computer with `scp` or `rsync`.

You can print or hand out this QR code file and people can scan it with their Delta Chat to get a temporary account which is valid for one day.

CHAPTER 4

Configuration Details

During setup, but also every time after you changed a config option, you need to run `mailadm init` to apply them, and restart the mailadm process/container.

`mailadm init`, saves the configuration in the database. `mailadm init` should be called from inside the docker container. Best practice is to save the environment variables in a `.env` file, and pass it to `docker run` with the `--env-file .env` argument. `mailadm.sh` script does this for you.:

```
$ mailadm init
```

mailadm has 4 config options:

4.1 MAIL_DOMAIN

This is the domain part of the email addresses your mailadm instance creates later. For addresses like `tmp.12345@example.org`, your `MAIL_DOMAIN` value in `.env` needs to look like:

```
MAIL_DOMAIN=example.org
```

4.2 WEB_ENDPOINT

The `WEB_ENDPOINT` is used for generating the URLs which are later encoded in the account creation QR codes. For mailadm to work, it must be reachable with `curl -X POST "$WEB_ENDPOINT?t=$TOKEN"` (see *testing-the-web-app*). For example:

```
WEB_ENDPOINT=http://mailadm.example.org/new_email
```

4.3 MAILCOW_ENDPOINT

mailadm needs to talk to the mailcow API to create and delete accounts. For this, add */api/v1/* to the URL of the mailcow admin interface, e.g.:

```
MAILCOW_ENDPOINT=https://mailcow-web.example.org/api/v1/
```

4.4 MAILCOW_TOKEN

To authenticate with the mailcow API, mailadm needs an API token. You can generate it in the mailcow admin interface, under "API". Note that you need to allow API access from the IP address of the server where you're running mailadm, or enable "Skip IP check for API" to allow API access from everywhere.

When you have activated the API, you can pass the token to mailadm like this:

```
MAILCOW_TOKEN=932848-324B2E-787E98-FCA29D-89789A
```

CHAPTER 5

Setup Development Environment

To setup your development environment, you need to do something like this:

```
git clone https://github.com/deltachat/mailadm
python3 -m venv venv
. venv/bin/activate
pip install pytest tox pytest-xdist pytest-timeout pyzbar
sudo apt install -y libzbar0
pip install .
```

With `tox` you can run the tests - many of them need access to a mailcow instance though. If you have access to a mailcow instance, you can pass a `MAILCOW_TOKEN`, `MAIL_DOMAIN`, and `MAILCOW_ENDPOINT` via the command line to run them.

CHAPTER 6

Mailadm HTTP API

`/`, method: `POST`: Create a temporary account with a specified token.

Attributes:

- `?t=` a valid mailadm token

Successful Response:

```
{
  "status_code": 200,
  "email": "addr@example.org",
  "password": "p4$$w0rd",
  "expiry": "1h",
  "ttl": 3600,
}
```

Example for an error:

```
{
  "status_code": 403,
  "type": "error",
  "reason": "?t (token) parameter not specified",
}
```

Possible errors:

403	?t (token) parameter not specified
403	token $t is invalid
409	user already exists in mailcow
409	user already exists in mailadm
500	internal server error, can have different reasons
504	mailcow not reachable

CHAPTER 7

Migrating from a pre-mailcow setup

mailadm used to be built on top of a standard postfix/dovecot setup; with mailcow many things are simplified. The migration can be a bit tricky though.

What you need to do:

- create all existing dovecot accounts in mailcow
- create a master password for dovecot
- do an IMAP sync to migrate the inboxes of all the dovecot accounts to mailcow (see https://mailcow.github.io/mailcow-dockerized-docs/post_installation/firststeps-sync_jobs_migration/)
- migrate the mailadm database (maybe the `mailadm migrate-db` command works for you; but better make a backup beforehand)
- re-configure mailadm with your mailcow credentials (see *configuration-details*)

If you get `NOT NULL constraint failed: users.hash_pw` errors when you try to create a user, you probably need to migrate your database. You can use `scripts/migrate-pre-mailcow-db.py` for this; it's not well tested though, so make a backup first and try it out.