Email setup in Emacs with Mu4e on macOS

2021-06-16 | emacs | #email , #emacs , #mu4e , #macos

UPDATE - 16. Jan. 2023:

Recently I came across a behaviour in Mu4e that did not do what was expected. It turns out Mu4e was not selecting the proper from-address when composing emails, which I did not notice for a long time.

The short of it is that the function used to send emails did not take into account the `message-send-mail-hook`, which is responsible to select the proper from-address.

To fix this adjust the following lines in the **mu4e-sending** section:

```
copy
;; send function:
-(setq send-mail-function 'sendmail-send-it
-      message-send-mail-function 'sendmail-send-it)
+(setq send-mail-function 'message-send-mail-with-sendmail
+      message-send-mail-function 'message-send-mail-with-sendmail)
```

Motivation

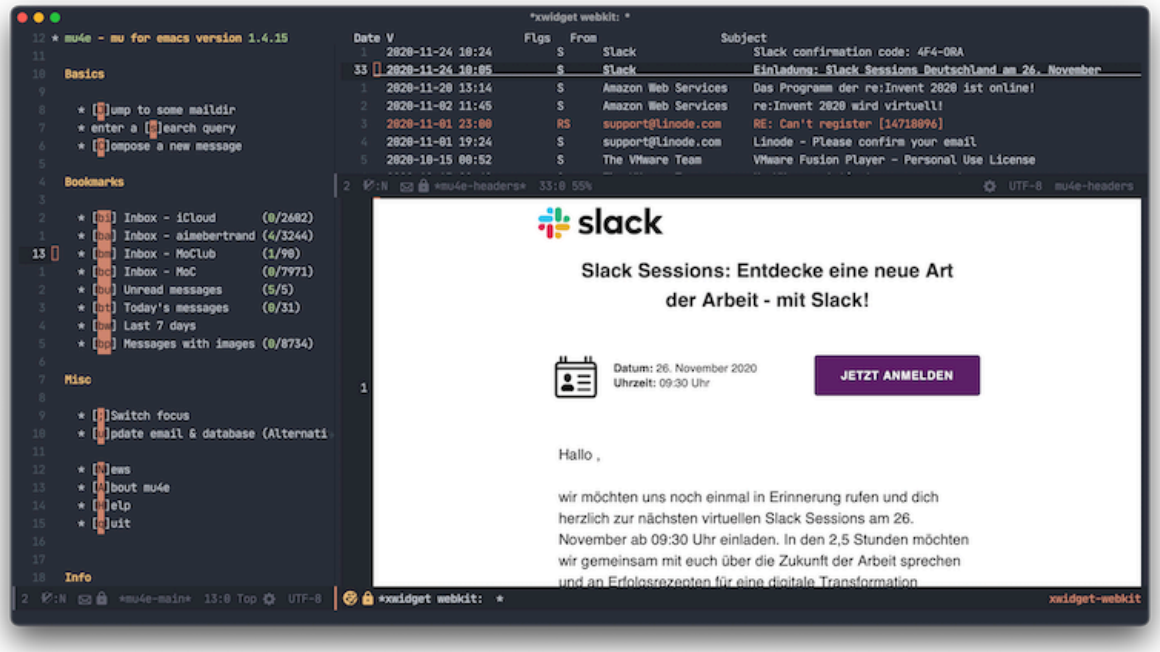
There are a bunch of instructions on setting up Mu4e on macOS online. However I found that most of them kind of require too deep a knowledge of Emacs and command-line tools for a beginner to master or troubleshoot without a proper headache. I know, because that was the case for me in the beginning.

My goal here is to provide a step by step instruction without looking too far

behind the curtain. With the hope that this way the discovery and deep dive into emailing with Mu4e in Emacs starts **after** the joy of a working system.

Now this seems long. And to be completely honest it is. But we are trying to be as thorough as possible here, right.

The last reason for typing this down is purely selfish. I want to have a record/documentation to go back to myself. As a matter of fact this is actually true for all my posts.



NOTE: This will only cover the setup. It is not a how to use Mu4e. The usage should be reasonably easy.

Prerequisites

A working Emacs configuration:

I suppose no one will setup Mu4e without that.

A working Installation of `Homebrew`:

Fortunately the installation is easy. In my mind, this is the best or

maybe just the easiest way to get (open-source) software onto your Mac.

The two above things will not be covered the instructions on this page.

How this will work

Mu4e is part of the command-line indexer `mu` for emails in maildir (mail directory) format. It does need other 3rd party dependencies for retrieving and for sending emails.

This post is limited to fetching emails over `imap` and for sending over `smtp`. Other Protocols and methods - I am looking at you Exchange - are out of scope.

However I am convinced that this should cover a vast majority of the users. I will be using a dummy example account `dummy@example.com` for a generic email account and one for **Gmail**, `dummy@gmail.com`. Since we are on a Mac we will be looking at a dummy **iCloud** account `dummy@icloud.com` as well.

iCloud and **Gmail** require more password setup for access through imap. There are good instructions for "app-specific passwords" with iCloud & "app passwords" with Gmail respectively.

Preparation

making the directories

This uses the maildir system. Meaning that the messages will be stored in a directory hierarchy as separate plain text files. We will need the following directories in the user's home directory:

copy

```
~/.maildir/icloud # name used for setting up mbsync an msmtp
~/.maildir/gmail # name used for setting up mbsync an msmtp
~/.maildir/example # name used for setting up mbsync an msmtp
~/.maildir/certificates # we will store System root certificates here
```

##storing passwords in the keychain

The easiest way to store ones password in the macOS keychain is with the `security` command in your terminal. For our accounts we will setup the three items `mu4e-icloud`, `mu4e-gmail` and `mu4e-example`.

icloud:
`security add-generic-password -s mu4e-icloud -a dummy -w`
gmail:
`security add-generic-password -s mu4e-gmail -a dummy@gmail.com -w`
example:
`security add-generic-password -s mu4e-example -a dummy@example.com -w`

For each account you will be prompted for the password to be stored.

NOTE: The names “mu4e-icloud”, “mu4e-gmail” and “mu4e-example” are entirely arbitrary.

##storing trusted root certificates

The applications `mbsync` and `msmtp` need a way to trust email servers’s tls/ssl certificates. We will use the list of certificate Authority trusted by the macOS system.

1. Open the Application `Keychain Access.app`
2. Select `System Roots` in the sidebar
3. Select all items listen here - `⌘ + a`
4. Export the items with `⇧ + ⌘ + e` to the file

`~/.maildir/certificates/root-certificates.pem`

Installing dependencies

- **mu**:
  - To index your emails in your mail directory.
  - Recommended Installation: `brew install mu`
- **mbsync**:
  - To Synchronize a maildir with an imap server we will need the command `mbsync` which is part of the isync application.
  - Recommended Installation: `brew install isync`
- **msmtp**:
  - Like the name suggests, msmtp is a smtp client. According to the website, it transmits a mail to an smtp server which takes care of further delivery.
  - Recommended Installation: `brew install msmtp`

Setup imap sync

The settings used by `mbsync` are stored in the file `~/.mbsyncrc` by default.

NOTE: Which imap servers, ports and credentials to use will be most definitely documented by your email provider.

configuration file

```
copy
IMAPAccount icloud
Host imap.mail.me.com
User dummy
PassCmd "security find-generic-password -s mu4e-icloud -a dummy -w"
Port 993
```

```
SSLType IMAPS
SSLVersions TLSv1.2
AuthMechs PLAIN
SystemCertificates no
CertificateFile ~/.maildir/certificates/root-certificates.pem

IMAPStore icloud-remote
Account icloud

MaildirStore icloud-local
SubFolders Verbatim
Path ~/.maildir/icloud/
Inbox ~/.maildir/icloud/INBOX

Channel icloud
Far :icloud-remote:
Near :icloud-local:
Patterns *
Create Near
Sync All
Expunge Both
SyncState *

# ============================================================================

IMAPAccount gmail
Host imap.gmail.com
User dummy@gmail.com
PassCmd "security find-generic-password -s mu4e-gmail -a dummy -w"
Port 993
SSLType IMAPS
SSLVersions TLSv1.2
AuthMechs PLAIN
SystemCertificates no
CertificateFile ~/.maildir/certificates/root-certificates.pem

IMAPStore gmail-remote
Account gmail
```

```
MaildirStore gmail-local
SubFolders Verbatim
Path ~/.maildir/gmail/
Inbox ~/.maildir/gmail/INBOX

Channel gmail
Far :gmail-remote:
Near :gmail-local:
Patterns *
Create Near
Sync All
Expunge Both
SyncState *

# ============================================================================

IMAPAccount example
Host imap.example.com
User dummy@example.com
PassCmd "security find-generic-password -s mu4e-example -a dummy@example.com -w"
Port 993
SSLType IMAPS
AuthMechs Login
CertificateFile ~/.maildir/certificates/root-certificates.pem

IMAPStore example-remote
Account example

MaildirStore example-local
SubFolders Verbatim
Path ~/.maildir/example/
Inbox ~/.maildir/example/INBOX

Channel example
Far :example-remote:
Near :example-local:
Patterns *
```

```
Create Near
Sync All
Expunge Both
SyncState *
```

##let me explain

Bare with me here. This part might seem a tad to much, but I would have been happy to have it, back then when. You can skip it of course if you want.

- ***IMAPAccount:*** You choose the name. But be consistent. See the sections above.
- ***Host:*** This is the imap server according to your provider.
- ***User:*** This is the login/user name according to your provider
- ***PassCmd:*** Remember we added the password in the section storing passwords in the keychain. We retrieve it withe command `security find-generic-password`.
- ***Port:*** The imap port according to your provider.
- ***SSLType:*** Leave this as is in the example file.
- ***AuthMechs:*** This is the way the account authenticates with the imap server. Leave the default here.
- ***CertificateFile:*** this is the file containing the trusted root certificates of the trusted Certificate authorities. See the section storing trusted root certificates.
- ***IMAPStore:*** Value for `IMAPAccount` plus `-remote`.
- ***Account:*** Same as `IMAPAccount`.
- ***MaildirStore:*** Value for `IMAPAccount` plus `-local`.
- ***SubFolders:*** Leave this as is in the example file.
- ***Path:*** We created this in the section making the directories.
- ***Inbox:*** This is `PATH` plus `/INBOX`.
- ***Channel:*** Same as `IMAPAccount`.
- ***Far:*** The syntax with the 2 `:` is important. Value for `IMAPAccount` plus `-remote`.
- ***Near:*** The syntax with the 2 `:` is important. Value for `IMAPAccount` plus `-local`.

- ***Patterns:*** Leave this as is in the example file.
- ***Create:*** Leave this as is in the example file.
- ***Sync:*** Leave this as is in the example file.
- ***Expunge:*** Leave this as is in the example file.
- ***SyncState:*** Leave this as is in the example file.

Initial sync

Now that all the above is setup, it is time to sync our mail for the first time. This will download all our mails from the imap servers.

first step - mbsync

This is as simple as:

```
copy
mbsync -aV
# -a for 'all' and
# -V for 'vorbose' - might be helpfull to see whats happening
```

second step - mu

Now we need to initialize our email settings to get our emails indexed. This is for mu and Mu4e to be able to search emails and more.

```
copy
# initialize the email settings:
mu init -m ~/.maildir \
   --my-address dummy@icloud.com \
   --my-address dummy@gmail.com \
   --my-address dummy@example.com

# index the emails in your .maildir:
```

```
mu index
```

Setup msmtp for sending

`msmtp` by default uses the file `~/.msmtprc` to store the settings.

NOTE: Which smtp servers, ports and credentials to use will be most definitely documented by your email provider.

configuration file

```
copy
# Set default values for all the accounts.
defaults
logfile ~/.maildir/msmtp.log
tls_trust_file ~/.maildir/certificates/root-certificates.pem

# ============================================================================

account icloud
auth on
host smtp.mail.me.com
port 465
protocol smtp
from dummy@icloud.com
user dummy
passwordeval security find-generic-password -s mu4e-icloud -a dummy -w
tls on
tls_starttls off

# ============================================================================

account gmail
auth on
host smtp.gmail.com
```

```
host smtp.gmail.com
port 465
protocol smtp
from dummy@gmail.com
user dummy
passwordeval security find-generic-password -s mu4e-gmail -a dummy -w
tls on
tls_starttls off

# ====================================================================

account example
auth on
host smtp.example.com
port 465
protocol smtp
from dummy@example.com
user dummy@example.com
passwordeval security find-internet-password -s mu4e-example -a dummy@example.co
tls on
tls_starttls off

# ====================================================================

account default : gmail
```

##let me explain

Same as before in the mbsync section. Bit much, but useful to many.

- ***defaults:*** Defaults for all accounts.
- ***logfile:*** Leave this as is in the example file.
- ***tls_trust_file:*** this is the file containing the trusted root certificates of the trusted Certificate authorities. See the section storing trusted root certificates.
- ***account:*** Same as the directory you created in making the directories.

- ***auth:*** Leave this as is in the example file.
- ***host:*** smtp server according to your provider.
- ***port:*** smtp port according to your provider.
- ***protocol:*** Leave this as is in the example file.
- ***from:*** login/user name according to your provider.
- ***user:*** login/user name according to your provider.
- ***passwordeval:*** Remember we added the password in the section storing passwords in the keychain. We retrieve it withe command `security find-generic-password`.
- ***tls:*** this should be `on`.
- ***tls_starttls:*** Usually this should be `off`. Check with the providers documentation if sending doesn't work.
- ***account default***: If you are using gmail, this should be the default. Otherwise you can choose. Without going to to much details, gmail is a bit touchy.

Setup Mu4e in Emacs

Put the following settings/configuration in your init file. Should be `~/.emacs` or `~/.emacs.d/init.el`. I will divide them into sections for better understanding. `;;` is used for comments.

require packages upfront

```
copy
;; load mu4e from the installation path.
;; yours might differ check with the Emacs installation
(use-package mu4e
  :load-path  "/usr/local/share/emacs/site-lisp/mu/mu4e/")

;; for sending mails
(require 'smtpmail)
```

mu4e-general-settings

copy

```
;; we installed this with homebrew
(setq mu4e-mu-binary (executable-find "mu"))

;; this is the directory we created before:
(setq mu4e-maildir "~/.maildir")

;; this command is called to sync imap servers:
(setq mu4e-get-mail-command (concat (executable-find "mbsync") " -a"))
;; how often to call it in seconds:
(setq mu4e-update-interval 300)

;; save attachment to desktop by default
;; or another choice of yours:
(setq mu4e-attachment-dir "~/Desktop")

;; rename files when moving - needed for mbsync:
(setq mu4e-change-filenames-when-moving t)

;; list of your email adresses:
(setq mu4e-user-mail-address-list '("dummy@icloud.com"
                                    "dummy@gmail.com"
                                    "dummy@example.com"))
```

mu4e-favorites

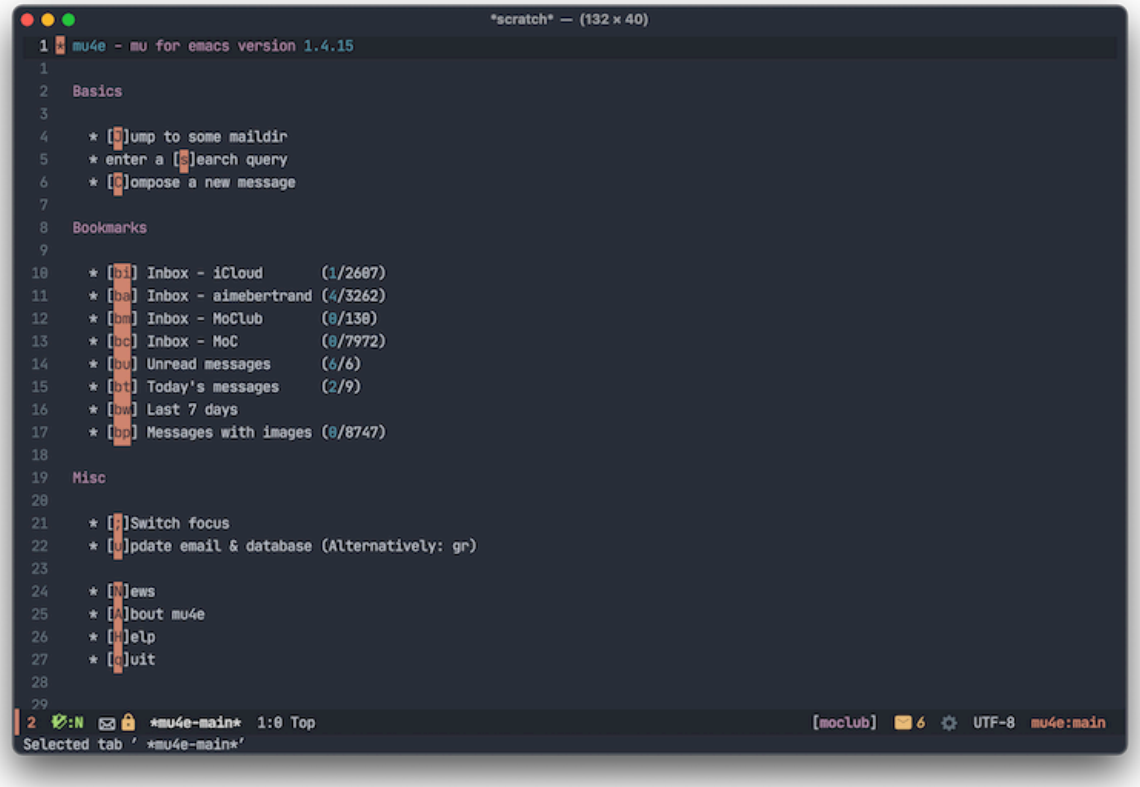
copy

```
;; check your ~/.maildir to see how the subdirectories are called
;; for the generic imap account:
;; e.g `ls ~/.maildir/example'
(setq   mu4e-maildir-shortcuts
        '(("/icloud/INBOX" . ?i)
          ("/icloud/Sent Messages" . ?I)
          ("/gmail/INBOX" . ?g)
          ("/gmail/[Gmail]/Sent Mail" . ?G)
          ("/example/INBOX" . ?e)
```

```
            ("/example/Sent" . ?E)))
```

NOTE: The letters assigned here - "?x" - are arbitrary.

mu4e-bookmarks



copy

```
;; the following is to show shortcuts in the main view.
(add-to-list 'mu4e-bookmarks
             (make-mu4e-bookmark
              :name "Inbox - iCloud"
              :query "maildir:/icloud/INBOX"
              :key ?i))
(add-to-list 'mu4e-bookmarks
             (make-mu4e-bookmark
              :name "Inbox - Gmail"
```

```
                :query "maildir:/gmail/INBOX"
                :key ?g))
  (add-to-list 'mu4e-bookmarks
               (make-mu4e-bookmark
                :name "Inbox - example"
                :query "maildir:/example/INBOX"
                :key ?e))
```

NOTE: The letters assigned here - "?x" - are arbitrary.

mu4e-context

This controls the account context one is in. Helpful for instance, when composing an email. You can then select the context, which sets at the same time the sender.

copy

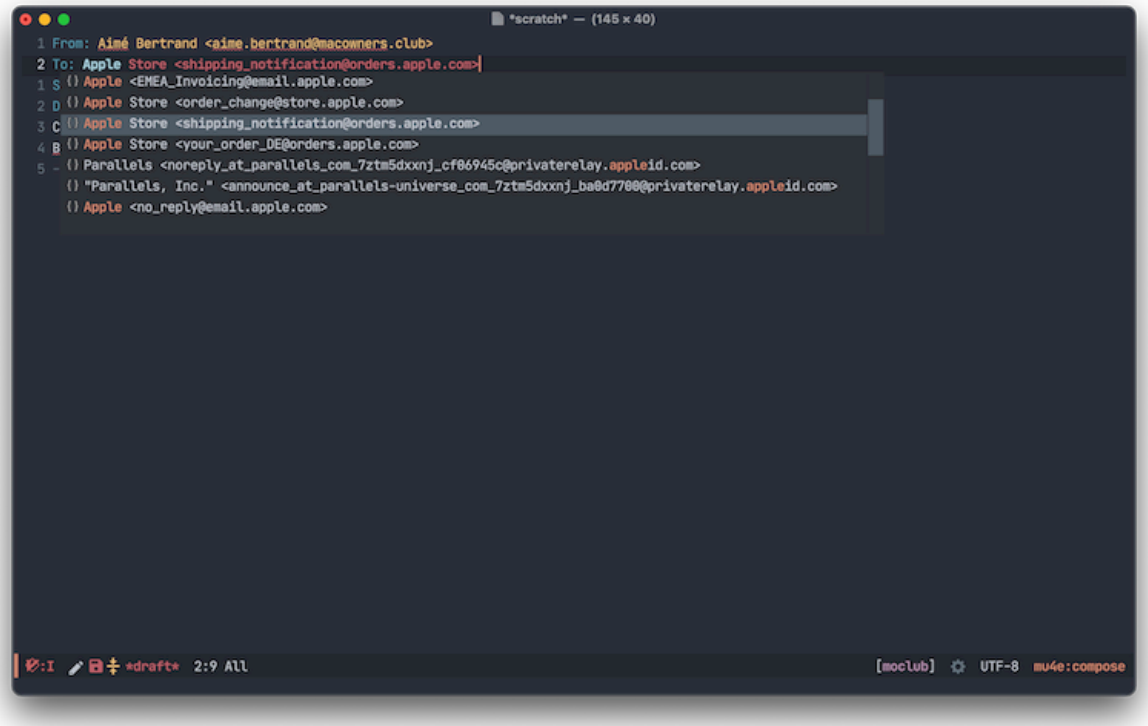
```
(setq mu4e-contexts
      `(,(make-mu4e-context
          :name "icloud"
          :enter-func
          (lambda () (mu4e-message "Enter dummy@icloud.com context"))
          :leave-func
          (lambda () (mu4e-message "Leave dummy@icloud.com context"))
          :match-func
          (lambda (msg)
            (when msg
              (mu4e-message-contact-field-matches msg
                                                  :to "dummy@icloud.com")))
          :vars '((user-mail-address . "dummy@icloud.com" )
                  (user-full-name . "Dummy McDummerson")
                  (mu4e-drafts-folder . "/icloud/Drafts")
                  (mu4e-refile-folder . "/icloud/Archive")
                  (mu4e-sent-folder . "/icloud/Sent Messages")
                  (mu4e-trash-folder . "/icloud/Deleted Messages")))
```

```
,(make-mu4e-context
  :name "gmail"
  :enter-func
  (lambda () (mu4e-message "Enter dummy@gmail.com context"))
  :leave-func
  (lambda () (mu4e-message "Leave dummy@gmail.com context"))
  :match-func
  (lambda (msg)
    (when msg
      (mu4e-message-contact-field-matches msg
                                          :to "dummy@gmail.com")))
  :vars '((user-mail-address . "dummy@gmail.com")
          (user-full-name . "Dummy McDummerson")
          (mu4e-drafts-folder . "/gmail/Drafts")
          (mu4e-refile-folder . "/gmail/Archive")
          (mu4e-sent-folder . "/gmail/Sent")
          (mu4e-trash-folder . "/gmail/Trash")))

,(make-mu4e-context
  :name "example"
  :enter-func
  (lambda () (mu4e-message "Enter dummy@example.de context"))
  :leave-func
  (lambda () (mu4e-message "Leave dummy@example.de context"))
  :match-func
  (lambda (msg)
    (when msg
      (mu4e-message-contact-field-matches msg
                                          :to "dummy@example.de")))
  :vars '((user-mail-address . "dummy@example.de")
          (user-full-name . "Dummy McDummerson")
          ;; check your ~/.maildir to see how the subdirectories are cal
          ;; e.g `ls ~/.maildir/example'
          (mu4e-drafts-folder . "/example/Drafts")
          (mu4e-refile-folder . "/example/Archive")
          (mu4e-sent-folder . "/example/Sent")
          (mu4e-trash-folder . "/example/Trash")))))
```

```
(setq mu4e-context-policy 'pick-first) ;; start with the first (default) context
(setq mu4e-compose-context-policy 'ask) ;; ask for context if no context matches
```

mu4e-sending



copy

```
;; gpg encryptiom & decryption:
;; this can be left alone
(require 'epa-file)
(epa-file-enable)
(setq epa-pinentry-mode 'loopback)
(auth-source-forget-all-cached)

;; don't keep message compose buffers around after sending:
(setq message-kill-buffer-on-exit t)

;; send function:
```

```
(setq send-mail-function 'sendmail-send-it
      message-send-mail-function 'sendmail-send-it)

;; send program:
;; this is exeranal. remember we installed it before.
(setq sendmail-program (executable-find "msmtp"))

;; select the right sender email from the context.
(setq message-sendmail-envelope-from 'header)

;; chose from account before sending
;; this is a custom function that works for me.
;; well I stole it somewhere long ago.
;; I suggest using it to make matters easy
;; of course adjust the email adresses and account descriptions
(defun timu/set-msmtp-account ()
  (if (message-mail-p)
      (save-excursion
        (let*
            ((from (save-restriction
                     (message-narrow-to-headers)
                     (message-fetch-field "from")))
             (account
              (cond
               ((string-match "dummy@icloud.com" from) "icloud")
               ((string-match "dummy@gmail.com" from) "gmail")
               ((string-match "dummy@example.com" from) "example"))))
          (setq message-sendmail-extra-arguments (list '"-a" account))))))

(add-hook 'message-send-mail-hook 'timu/set-msmtp-account)

;; mu4e cc & bcc
;; this is custom as well
(add-hook 'mu4e-compose-mode-hook
          (defun timu/add-cc-and-bcc ()
            "My Function to automatically add Cc & Bcc: headers.
    This is in the mu4e compose mode."
            (save-excursion (message-add-header "Cc:\n"))
```

```
            (save-excursion (message-add-header "Bcc:\n"))))

  ;; mu4e address completion
  (add-hook 'mu4e-compose-mode-hook 'company-mode)
```

##optional

This are optional variables to set. At least the way I have set them. They are not needed for the system to work, but are a suggestion. My sane defaults sort of speak.

```
copy
;; store link to message if in header view, not to header query:
(setq org-mu4e-link-query-in-headers-mode nil)
;; don't have to confirm when quitting:
(setq mu4e-confirm-quit nil)
;; number of visible headers in horizontal split view:
(setq mu4e-headers-visible-lines 20)
;; don't show threading by default:
```

```
(setq mu4e-headers-show-threads nil)
;; hide annoying "mu4e Retrieving mail..." msg in mini buffer:
(setq mu4e-hide-index-messages t)
;; customize the reply-quote-string:
(setq message-citation-line-format "%N @ %Y-%m-%d %H:%M :\n")
;; M-x find-function RET message-citation-line-format for docs:
(setq message-citation-line-function 'message-insert-formatted-citation-line)
;; by default do not show related emails:
(setq mu4e-headers-include-related nil)
;; by default do not show threads:
(setq mu4e-headers-show-threads nil)
```

> **That is it! You should now be ready to go and discover.**

Conclusion

The opinions on which email client to use are many. I was a happy user of the macOS built-in Mail.app till I discovered Emacs. Even then it took me a while to get into migrating my mail workflow into Mu4e.

Now that I am quite invested and committed however, I cannot imagine myself using another client. It is true that I keep the Mail.app around as a backup, because I like to tinker with my Emacs config, but I have not needed it so far. The Mu4e system works and the advantages cannot be denied.

- Emails in plain text
  - When all fails, I can read my mails in an editor.
  - I can even keep my mails in a version control. Deleted mails you did not want to. Go back a few commits.
- Another workflow into Emacs
  - We Emacs enthusiast know, why this is a brilliant idea. You can use all the amenities with your mails.

- Much more.

Since Mu4e is inside Emacs and adheres to the principles as well, there is nothing you cannot do with it.

- You got all manner of packages - this is just a selection - to extend the capabilities.
- You can easily write or steal Emacs-lisp code for Mu4e to make it yours. This might not be obvious from the start, but be patient and you will see.

Sources

Starting I scoured the web for any info I could find.

- Mu4e user manual
- Mu Cheatsheet
- Drowning in Email; mu4e to the Rescue
- mu4e 0.9.18: E-Mailing with Emacs now even better
- Pragmatic Emacs's Mu4e posts
- Mike Zamansky's YouTube videos about mu4e
- System Crafters's YouTube series about mu4e
- All kind of posts on reddit.com/r/emacs

categories | tags