CL-2006 Operating Systems

LAB - 05

Linux multifunction Server Management (LAMP stack), Postfix and Mozilla Thunderbird

Introduction

The LAMP stack is a set of open-source tools used for web application development. For a web application to work, it has to include a server operating system, a web server, a database, and a programming language. Each layer of software is necessary for creating a database-driven and dynamic website.

This step-by-step tutorial shows you how to install LAMP in Ubuntu.

Prerequisites

- Ubuntu 18.04 or later
- User with sudo privileges
- Access to a terminal/command line

How to Install LAMP in Ubuntu

LAMP is a collection of four components that make up a fully functional web development environment. The LAMP acronym contains the initials of the components' names:

- **Linux** Operating System
- **Apache** HTTP Server
- **MySQL** database management system
- **PHP** programming language (Pearl and Python are also sometimes used in the stack)

Follow the steps below to install each tool on your system.

Step 1: Install Apache

Apache HTTP Server is the web server running on top of Linux in the LAMP stack. The web server uses HTTP to process requests and transmit information through the internet.

Follow the procedure below to install Apache.

1. Before installing the first LAMP component, ensure the package list on the system is up to date. In the terminal, type:

```
sudo apt update
```

2. To install the Apache package, run the following command:

```
sudo apt install apache2 -y
```

Note: The `-y` flag allows skipping the installation confirmation prompt.

```
marko@test-main:~$ sudo apt install apache2 -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following NEW packages will be installed:
  apache2
0 upgraded, 1 newly installed, 0 to remove and 11 not upgraded.
Need to get 95.6 kB of archives.
After this operation, 543 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 apache2 amd64 2.4.41
-4ubuntu3.12 [95.6 kB]
Fetched 95.6 kB in 1s (119 kB/s)
Selecting previously unselected package apache2.
(Reading database ... 174557 files and directories currently installed.)
Preparing to unpack .../apache2_2.4.41-4ubuntu3.12_amd64.deb ...
Unpacking apache2 (2.4.41-4ubuntu3.12) ...
Setting up apache2 (2.4.41-4ubuntu3.12) ...
Processing triggers for systemd (245.4-4ubuntu3.18) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for ufw (0.36-6ubuntu1) ...
marko@test-main:~$
```

3. Check if Apache installed correctly by checking the Apache service status:

```
sudo service apache2 status
```

The service shows as `running` in the output:

```
marko@test-main:~$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enab>
     Active: active (running) since Thu 2022-10-13 05:26:21 EDT; 4min 53s ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 15228 (apache2)
      Tasks: 6 (limit: 11573)
     Memory: 9.4M
     CGroup: /system.slice/apache2.service
             ├─15228 /usr/sbin/apache2 -k start
             ├─15230 /usr/sbin/apache2 -k start
             ├─15231 /usr/sbin/apache2 -k start
             ├─15233 /usr/sbin/apache2 -k start
             ├─15234 /usr/sbin/apache2 -k start
             └─15235 /usr/sbin/apache2 -k start

Oct 13 05:26:21 test-main systemd[1]: Starting The Apache HTTP Server...
Oct 13 05:26:21 test-main apachectl[15227]: AH00558: apache2: Could not reliably deter>
Oct 13 05:26:21 test-main systemd[1]: Started The Apache HTTP Server.
lines 1-18/18 (END)
```

Exit the status screen by pressing **Ctrl** + **C** on the keyboard.

4. Next, make sure that the UFW firewall contains the Apache profiles by typing in the following command:

```
sudo ufw app list
```

```
marko@test-main:~$ sudo ufw app list
Available applications:
  Apache
  Apache Full
  Apache Secure
  CUPS
marko@test-main:~$
```

5. Ensure the **Apache Full** profile allows the traffic on ports **80** and **443** by running the command:

```
sudo ufw app info "Apache Full"
```

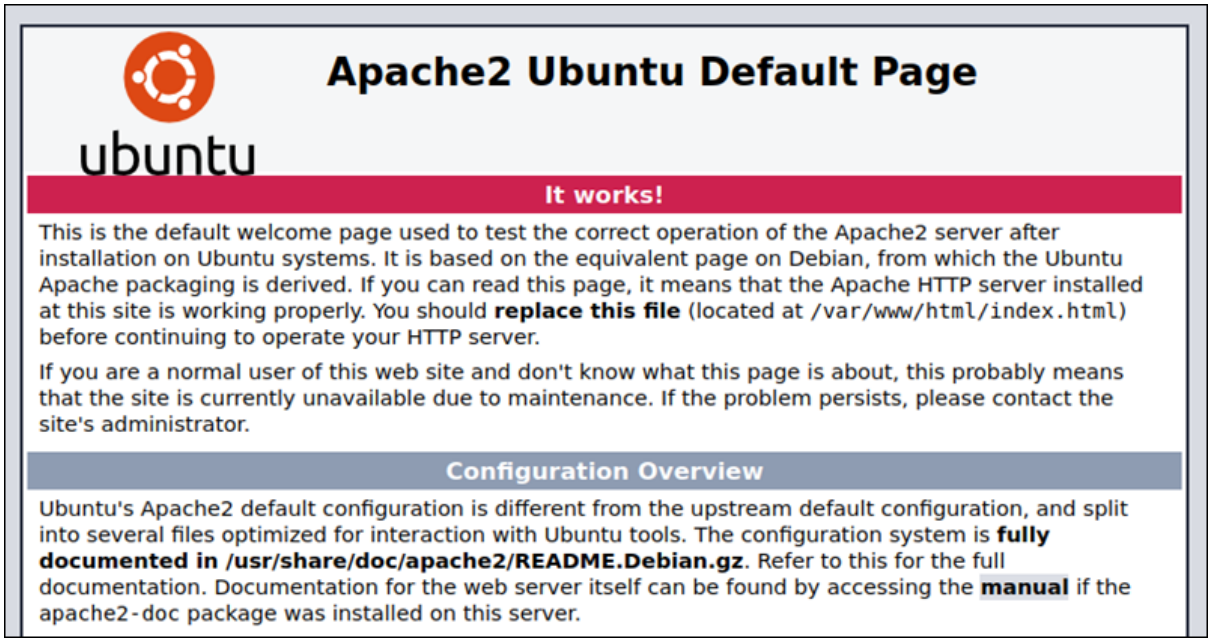
The output should look similar to the following example:

```
marko@test-main:~$ sudo ufw app info "Apache Full"
Profile: Apache Full
Title: Web Server (HTTP,HTTPS)
Description: Apache v2 is the next generation of the omnipresent Apache web
server.

Ports:
  80,443/tcp
marko@test-main:~$
```

6. To confirm that Apache is running, enter the IP address of your server in the address bar of an internet browser and press **ENTER**.

The test Apache web server page should display as below.



Note: You can also access the Apache test page by typing `localhost` in the address bar.

Step 2: Install MySQL and Create a Database

MySQL is a relational database management system for creating and maintaining dynamic enterprise-level databases. It is compatible with all major OS platforms, which makes it a good fit for web application development.

Note: Refer to our article and find out what is a relational database.

Install MySQL by typing the following command:

```
sudo apt install mysql-server -y
```

```
Setting up mysql-server-8.0 (8.0.30-0ubuntu0.20.04.2) ...
update-alternatives: using /etc/mysql/mysql.cnf to provide /etc/mysql/my.cnf (my.cnf) i
n auto mode
mysqld will log errors to /var/log/mysql/error.log
mysqld is running as pid 19563
Setting up mysql-server (8.0.30-0ubuntu0.20.04.2) ...
Processing triggers for systemd (245.4-4ubuntu3.18) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.9) ...
marko@test-main:~$
```

Step 3: Install PHP

Although other programming languages, such as Python and Pearl, also work well within LAMP, PHP is usually the final layer of the stack because it integrates well with MySQL. As a dynamically typed language, PHP embeds into HTML, improving the speed and reducing the complexity of web applications.

Install PHP by following the steps below.

1. Obtain the necessary PHP packages by typing:

```
sudo apt install php libapache2-mod-php php-mysql -y
```

```
Setting up php7.4-mysql (7.4.3-4ubuntu2.13) ...

Creating config file /etc/php/7.4/mods-available/mysqlnd.ini with new version

Creating config file /etc/php/7.4/mods-available/mysqli.ini with new version

Creating config file /etc/php/7.4/mods-available/pdo_mysql.ini with new version
Setting up libapache2-mod-php (2:7.4+75) ...
Setting up php-mysql (2:7.4+75) ...
Processing triggers for libapache2-mod-php7.4 (7.4.3-4ubuntu2.13) ...
Processing triggers for php7.4-cli (7.4.3-4ubuntu2.13) ...
marko@test-main:~$
```

2. Modify the way Apache serves files by opening the ***dir.conf*** file in a text editor with root privileges:

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

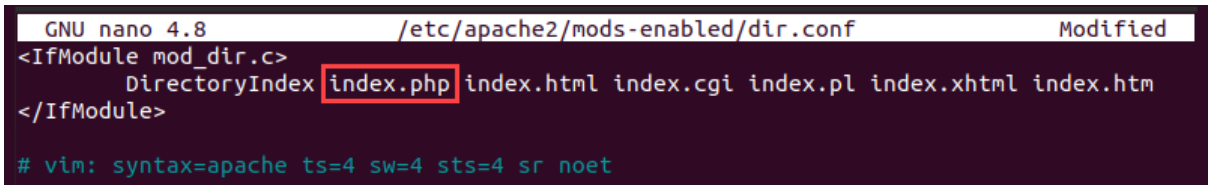
The configuration file looks like in the example below:

```
  GNU nano 4.8                 /etc/apache2/mods-enabled/dir.conf
<IfModule mod_dir.c>
        DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

By default, Apache first looks for an ***index.html*** file card.

3. Edit the list so that the ***index.php*** file is in the first position:



4. Press **CTRL + X** to save and close the file. Press **y** and **ENTER** to confirm.

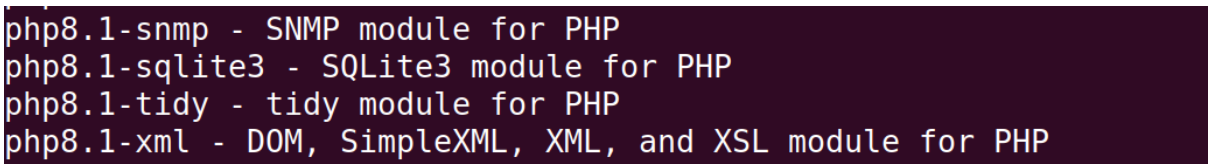
Install PHP Modules (Optional)

If necessary, add more modules to improve the functionality of PHP. Search, view, and install various libraries and modules by following the procedure below.

1. Get a list of available PHP modules with:

```
apt-cache search php- | less
```

The command pipes the results of the `apt-cache` search into `less` to simplify viewing the output.



2. Scroll up and down by using the arrow keys to see all the options, including a short description for each module.

3. For example, to find out what the module **php8.1-tidy** does, type:

```
apt-cache show php8.1-tidy
```

The output displays the module description.

```
monis@monis-virtual-machine:~$ apt-cache show php8.1-tidy
Package: php8.1-tidy
Architecture: amd64
Version: 8.1.2-1ubuntu2.10
Priority: optional
Section: php
Source: php8.1
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Debian PHP Maintainers <team+pkg-php@tracker.debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 92
Provides: php-tidy
```

4. To install the `php8.1-tidy` package after viewing its description, use the following command:

```
sudo apt install php8.1-tidy
```

5. When you finish, press **q** to quit.

Step 4: Restart Apache

For the changes to take effect, restart the Apache service by typing:

```
sudo systemctl restart apache2
```

If the command executes correctly, it returns no output.

Step 5: Test PHP Processing on Web Server

To test the new LAMP installation, create a basic PHP script and place it in the web root directory located at **/var/www/html/**, then check if the script is accessible via an internet browser. The steps below explain the procedure for performing this test.

1. Create a file in the web root directory by typing the following command:

```
sudo nano /var/www/html/info.php
```

2. Inside the file, type the PHP code:

```
<?php
phpinfo ();
?>
```

3. Press **CTRL + X** to save and close the file. Press **y** and **ENTER** to confirm.

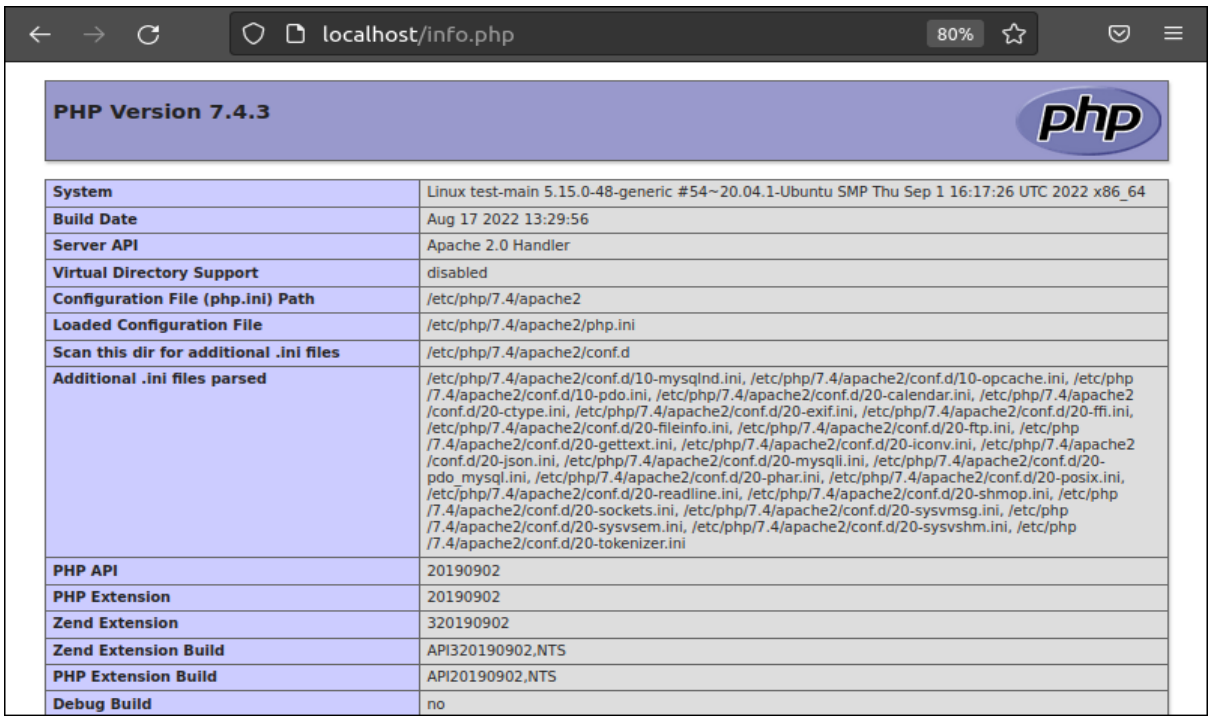
4. Open an internet browser and type the following address:

```
[server-ip-address]/info.php
```

Alternatively, type:

```
localhost/info.php
```

The output should display the details of the LAMP stack, as seen in the image below:



System	Linux test-main 5.15.0-48-generic #54~20.04.1-Ubuntu SMP Thu Sep 1 16:17:26 UTC 2022 x86_64
Build Date	Aug 17 2022 13:29:56
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d
Additional .ini files parsed	/etc/php/7.4/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20-exif.ini, /etc/php/7.4/apache2/conf.d/20-ffi.ini, /etc/php/7.4/apache2/conf.d/20-fileinfo.ini, /etc/php/7.4/apache2/conf.d/20-ftp.ini, /etc/php/7.4/apache2/conf.d/20-gettext.ini, /etc/php/7.4/apache2/conf.d/20-iconv.ini, /etc/php/7.4/apache2/conf.d/20-json.ini, /etc/php/7.4/apache2/conf.d/20-mysqli.ini, /etc/php/7.4/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.4/apache2/conf.d/20-phar.ini, /etc/php/7.4/apache2/conf.d/20-posix.ini, /etc/php/7.4/apache2/conf.d/20-readline.ini, /etc/php/7.4/apache2/conf.d/20-shmop.ini, /etc/php/7.4/apache2/conf.d/20-sockets.ini, /etc/php/7.4/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.4/apache2/conf.d/20-sysvsem.ini, /etc/php/7.4/apache2/conf.d/20-sysvshm.ini, /etc/php/7.4/apache2/conf.d/20-tokenizer.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,NTS
PHP Extension Build	API20190902,NTS
Debug Build	no

POSTFIX:

Postfix is a popular open-source Mail Transfer Agent (MTA) that can be used to route and deliver email on a Linux system. It is estimated that around 25% of public mail servers on the internet run Postfix.

Step 1 — Installing Postfix

Postfix is included in Ubuntu's default repositories, so you can install it with APT.

To begin, update your local `apt` package cache:

```
sudo apt update
```

Then install the `postfix` package with the following command. Note that here we pass the `DEBIAN_PRIORITY=low` environmental variable into this installation command. This will cause the installation process to prompt you to configure some additional options:

```
sudo DEBIAN_PRIORITY=low apt install postfix
```

This installation process will open a series of interactive prompts. For the purposes of this tutorial, use the following information to fill in your prompts:

- **General type of mail configuration?** For this, choose **Internet Site** since this matches our infrastructure needs.
- **System mail name**: This is the base domain used to construct a valid email address when only the account portion of the address is given. For instance, let's say the hostname of your server is `mail.example.com`. You will likely want to set the system mail name to `example.com` so that, given the username `user1`, Postfix will use the address `user1@example.com`.
- **Root and postmaster mail recipient**: This is the Linux account that will be forwarded mail addressed to `root@` and `postmaster@`. Use your primary account for this. In this example case, **Sammy**.
- **Other destinations to accept mail for**: This defines the mail destinations that this Postfix instance will accept. If you need to add any other domains that this server will be responsible for receiving, add those here. Otherwise, the default will be sufficient.
- **Force synchronous updates on mail queue?** Since you are likely using a journaled filesystem, accept **No** here.
- **Local networks**: This is a list of the networks for which your mail server is configured to relay messages. The default will work for most scenarios. If you choose to modify it, though, make sure to be very restrictive in regards to the network range.
- **Mailbox size limit**: This can be used to limit the size of messages. Setting it to `0` disables any size restriction.
- **Local address extension character**: This is the character that can be used to separate the regular portion of the address from an extension (used to create dynamic aliases). The default, `+` will work for this tutorial.

- **Internet protocols to use**: Choose whether to restrict the IP version that Postfix supports. For the purposes of this tutorial, pick **all**.

To be explicit, these are the settings used in this guide:

- **General type of mail configuration?**: **Internet Site**
- **System mail name**: `example.com` (not `mail.example.com`)
- **Root and postmaster mail recipient**: The username of your primary Linux account (**sammy** in our examples)
- **Other destinations to accept mail for**: `$myhostname, example.com, mail.example.com, localhost.example.com, localhost`
- **Force synchronous updates on mail queue?**: **No**
- **Local networks**: `127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128`
- **Mailbox size limit**: `0`
- **Local address extension character**: `+`
- **Internet protocols to use**: **all**

Note: If you need to ever return to change these settings, you can do so by typing:

```
sudo dpkg-reconfigure postfix
```

When you are prompted to restart services, accept the defaults and choose `OK`.

When the installation process finishes, you're ready to make a few updates to your Postfix configuration.

Step 2 — Changing the Postfix Configuration

Now you can adjust some settings that the package installation process didn't prompt you for. Many of Postfix's configuration settings are defined in the `/etc/postfix/main.cf` file. Rather than editing this file directly, you can use Postfix's `postconf` command to query or set configuration settings.

To begin, set the location for your non-root Ubuntu user's mailbox. In this guide, we'll use the *Maildir* format, which separates messages into individual files that are then moved between directories based on user action. The alternative option that isn't covered in this guide is the *mbox* format, which stores all messages within a single file.

Set the `home_mailbox` variable to `Maildir/`. Later, you will create a directory structure under that name within your user's home directory.
Configure `home_mailbox` by typing:

```
sudo postconf -e 'home_mailbox= Maildir/'
```

Next, set the location of the `virtual_alias_maps` table, which maps arbitrary email accounts to Linux system accounts. Run the following command, which maps the table location to a hash database file named `/etc/postfix/virtual`:

```
sudo postconf -e 'virtual_alias_maps= hash:/etc/postfix/virtual'
```

Now that you've defined the location of the virtual maps file in your `main.cf` file, you can create the file itself and begin mapping email accounts to user accounts on your Linux system. Create the file with your preferred text editor; in this example, we'll use `nano`:

```
sudo nano /etc/postfix/virtual
```

List any addresses that you wish to accept email for, followed by a whitespace and the Linux user you'd like that mail delivered to.

For example, if you would like to accept email at `contact@example.com` and `admin@example.com` and would like to have those emails delivered to the **sammy** Linux user, you could set up your file like this:

/etc/postfix/virtual

```
contact@example.com sammy
```

After you've mapped all of the addresses to the appropriate server accounts, save and close the file. If you used `nano`, do this by pressing `CTRL + X`, `Y`, then `ENTER`.

Apply the mapping by typing:

```
sudo postmap /etc/postfix/virtual
```

Restart the Postfix process to be sure that all of your changes have been applied:

```
sudo systemctl restart postfix
```

Assuming you followed the prerequisite Initial Server Setup guide, you will have configured a firewall with UFW. This firewall will block external connections to services on your server by default unless those connections are explicitly allowed, so you'll have to add a firewall rule to allow an exception for Postfix.

You can allow connections to the service by typing:

```
sudo ufw allow Postfix
```

With that, Postfix is configured and ready to accept external connections. However, you aren't yet ready to test it out with a mail client. Before you can install a client and use it to interact with the mail being delivered to your server, you'll need to make a few changes to your Ubuntu server's setup.

Step 3 — Installing the Mail Client and Initializing the Maildir Structure

In order to interact with the mail being delivered, this step will walk you through the process of installing the `s-nail` package. This is a feature-rich variant of the BSD `xmail` client which can handle the Maildir format correctly.

Before installing the client, though, it would be prudent to make sure your `MAIL` environment variable is set correctly. `s-nail` will look for this variable to figure out where to find mail for your user.

To ensure that the `MAIL` variable is set regardless of how you access your account — whether through `ssh`, `su`, `su -`, or `sudo`, for example — you'll need to set the variable in the `/etc/bash.bashrc` file and add it to a file within `/etc/profile.d` to make sure it is set for all users by default.

To add the variable to these files, type:

```
echo 'export MAIL=~/Maildir' | sudo tee -a /etc/bash.bashrc | sudo tee -a /etc/profile.d/mail.sh
```

To read the variable into your current session, source the `/etc/profile.d/mail.sh` file:

```
source /etc/profile.d/mail.sh
```

With that complete, install the `s-nail` email client with APT:

```
sudo apt install s-nail
```

Before running the client, there are a few settings you need to adjust. Open the `/etc/s-nail.rc` file in your editor:

```
sudo nano /etc/s-nail.rc
```

At the bottom of the file, add the following options:

/etc/s-nail.rc

```
. . .

set emptystart

set folder=Maildir
```

Here's what these lines do:

- `set emptystart`: allows the client to open even with an empty inbox
- `set folder=Maildir`: sets the `Maildir` directory to the internal `folder` variable
- `set record=+sent` creates a `sent` mbox file for storing sent mail within whichever directory is set as the `folder` variable, in this case `Maildir`

Save and close the file when you are finished. You're now ready to initialize your system's Maildir structure.

A quick way to create the Maildir structure within your home directory is to send yourself an email with the `s-nail` command. Because the `sent` file will only be available once the Maildir is created, you should disable writing to it for this initial email. Do this by passing the `-Snorecord` option.

Send the email by piping a string to the `s-nail` command. Adjust the command to mark your Linux user as the recipient:

```
echo 'init' | s-nail -s 'init' -Snorecord sammy
```

NOTE: The S- in snorecord is capital

You can can check to make sure the directory was created by looking for your `~/Maildir` directory:

```
ls -R ~/Maildir
```

You will see the directory structure has been created and that a new message file is in the `~/Maildir/new` directory:

```
Output

/home/sammy/Maildir/:

cur  new  tmp

/home/sammy/Maildir/cur:
```

Now that the directory structure has been created, you're ready to test out the `s-nail` client by viewing the `init` message you sent and sending a message to an external email address.

Step 4 — Testing the Client

To open the client, run the `s-nail` command:

```
s-nail
```

In your console, you'll see a rudimentary inbox with the `init` message waiting:

```
Output

s-nail version v14.9.15.  Type `?' for help

"/home/sammy/Maildir": 1 message 1 new
```

Press `ENTER` to display the message:

```
Output

[-- Message  1 -- 14 lines, 452 bytes --]:

Date: Mon, 18 Apr 2022 15:09:46 +0000

To: sammy@example.com

Subject: init
```

You can get back to the message list by typing `h`, and then `ENTER`:

```
h

Output
```

Notice that the message now has a state of `R`, indicating that it's been read.

Since this message isn't very useful, you can delete it by pressing `d`, and then `ENTER`:

```
d
```

To get back to the terminal, type `q` and then `ENTER`:

```
q
```

As a final test, check whether `s-nail` is able to correctly send email messages. To do this, you can pipe the contents of a text file into the `s-nail` process, like you did with the `init` message you sent in the previous step.

Begin by writing a test message in a text editor:

```
nano ~/test_message
```

Inside, enter some text you'd like to send:

~/test_message

```
Hello,
```

Save and close the file after writing your message.

Then, use the `cat` command to pipe the message to the `s-nail` process. You can do so with the following example, which uses these options:

- `-s`: This defines the subject line of the email message
- `-r`: An optional change to the "From:" field of the email. By default, the Linux user you are logged in as will be used to populate this field. The `-r` option allows you to override this with a valid address, such as one of those you defined in the `/etc/postfix/virtual` file. To illustrate, the following command uses `contact@example.com`

Also, be sure to change `user@email.com` to a valid email address which you have access to:

```
cat ~/test_message | s-nail -s 'Test email subject line' -r
contact@example.com user@email.com
```

NOTE: The above two commands are in a single line

Then, navigate to the inbox for the email address to which you sent the message. You will see your message waiting there almost immediately.

Note: If the message isn't in your inbox, it may have been delivered to your Spam folder.

You can view your sent messages within your `s-nail` client. Start the interactive client again:

```
s-nail
```

From the email client, view your sent messages by typing:

```
file +sent
```

You'll see output like this:

```
Output

+[/home/sammy/Maildir/]sent: 1 message 1 new

>N  1 To contact@example.com  2022-04-18 15:12  10/211   Test email
```

You can manage sent mail using the same commands you use for incoming mail.

Exercises:

1. Recreate the above illustrated example on your virtual machine and attach screenshots of your work for LAMP Stack and Postfix
2. Create an email via nano and sent to your nu id with the subject K21-XXXX (Attach screenshots of the entire process)
3. Sent an email to admin@example.com , read the email via terminal and delete the email and send it back to the recipient.
4. Explore Mozilla's Thunderbird mail server and client and perform the following tasks.

    1. Set up an email account.
    2. Sent and receive and email from the originally created client (Locally) (i.e., email to username@example.com to admin@example.com).
    3. Perform the following above mentioned tasks and attach screenshots of the operation either in the terminal or gui

Resources for Mozilla's Thunderbird
Set up Email
https://www.hostinger.com/tutorials/thunderbird-email-setup#:~:text=Setting%20up%20Email%20in%20Mozilla%20Thunderbird%20Automatically&text=Start%20by%20opening%20Mozilla%20Thunderbird,Then%2C%20press%20Continue.

How to configure an email account in Thunderbird

https://www.youtube.com/watch?v=fUKJhx1vM04