|  |  |  |
| --- | --- | --- |
| **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**](https://phoenixnap.com/glossary/what-is-linux)Operating System
* [**Apache**](https://phoenixnap.com/kb/nginx-vs-apache) HTTP Server
* [**MySQL**](https://phoenixnap.com/kb/install-mysql-ubuntu-20-04) database management system
* [**PHP**](https://phoenixnap.com/kb/check-php-version) programming language (Pearl and [Python](https://phoenixnap.com/kb/how-to-install-python-3-ubuntu) 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](https://phoenixnap.com/blog/web-server-vs-application-server) uses HTTP to process requests and transmit information through the internet.

Follow the procedure below to install Apache.

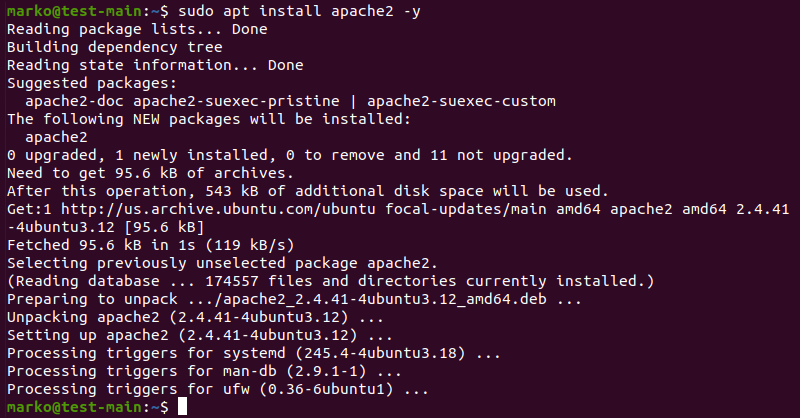
1. Before installing the first LAMP component, ensure the package list on the system is up to date. In the terminal, type:

**sudo apt update**

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

**sudo apt install apache2 -y**

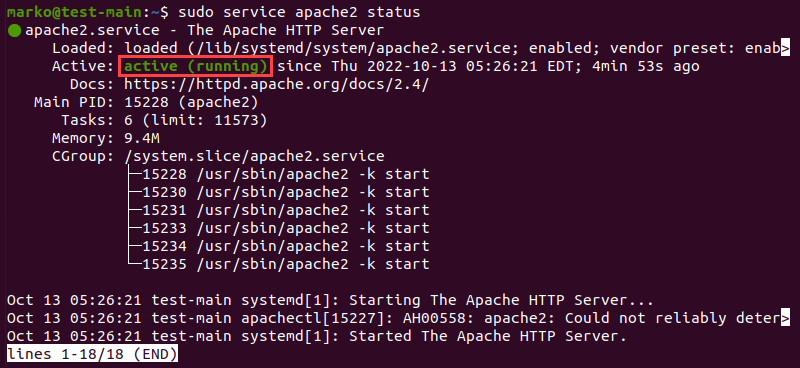
**Note:** The **-y** flag allows skipping the installation confirmation prompt.



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

**sudo service apache2 status**

The service shows as **running** in the output:



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

4. Next, make sure that the [UFW firewall](https://phoenixnap.com/kb/configure-firewall-with-ufw-on-ubuntu) contains the Apache profiles by typing in the following command:

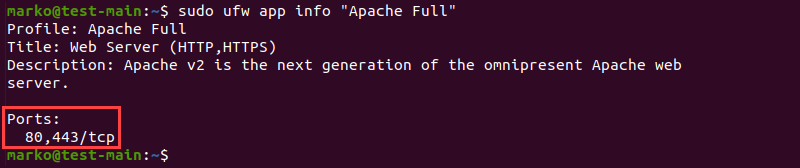
**sudo ufw app list**



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

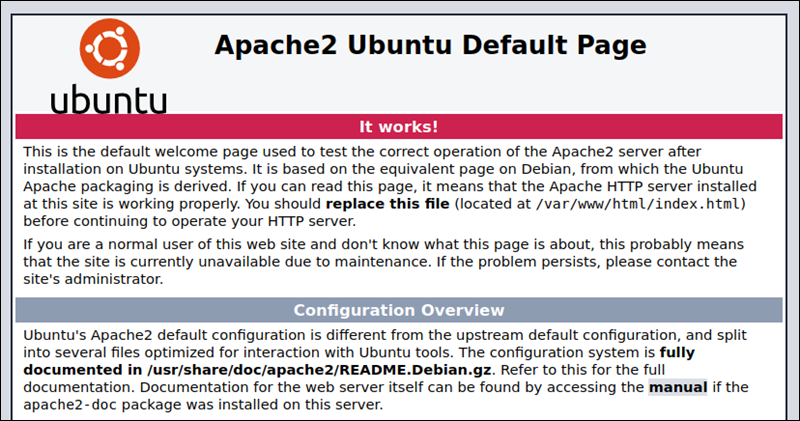
**sudo ufw app info "Apache Full"**

The output should look similar to the following example:



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.

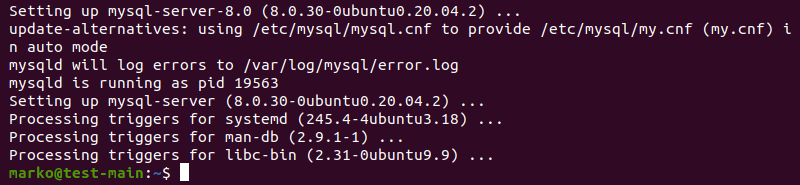
### Step 2: Install MySQL and Create a Database

MySQL is a relational [database management system](https://phoenixnap.com/kb/best-database-software) 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](https://phoenixnap.com/kb/what-is-a-relational-database).

Install MySQL by typing the following command:

**sudo apt install mysql-server -y**



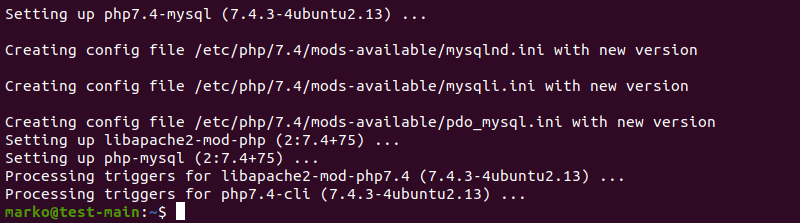
### Step 3: Install PHP

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](https://phoenixnap.com/kb/connect-mysql-with-php). 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:

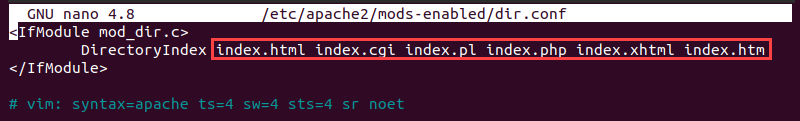
**sudo apt install php libapache2-mod-php php-mysql -y**



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

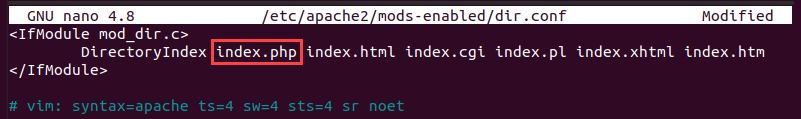
**sudo nano /etc/apache2/mods-enabled/dir.conf**

The configuration file looks like in the example below:



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.

Text

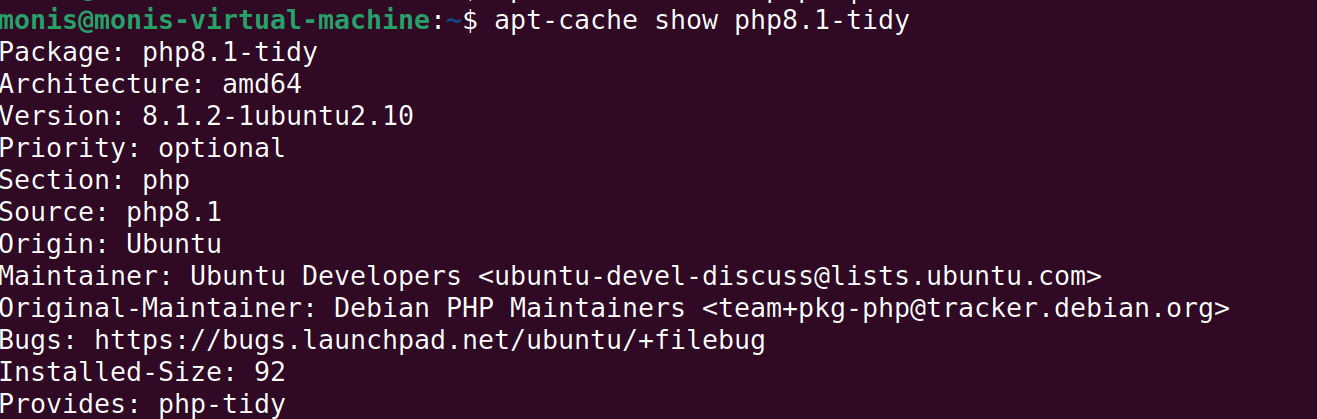
Description automatically generated

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.



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:

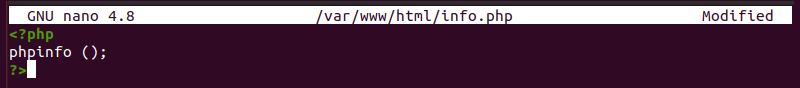
**sudo nano /var/www/html/info.php**

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.

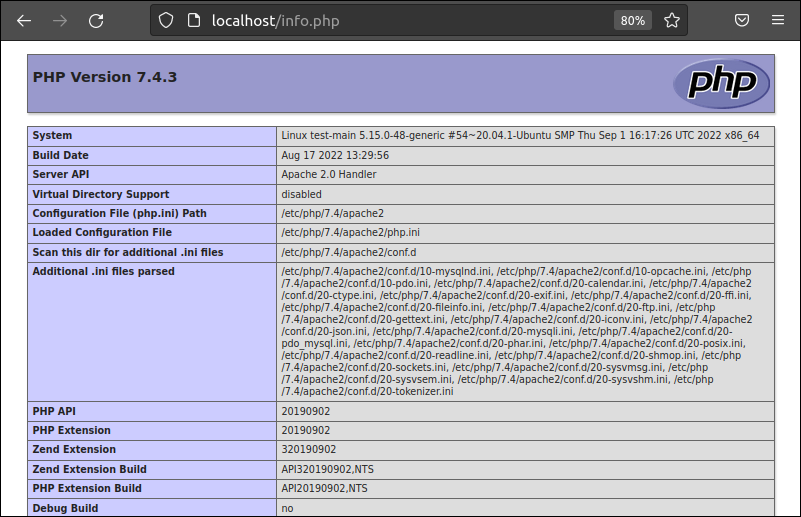
4. Open an internet browser and type the following address:

**[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:



***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](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/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

admin@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](https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-22-04), 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

set record=+sent

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:

/home/sammy/Maildir/new:

1650294586.Vfc01I7e11dM993645.mail.example.com

/home/sammy/Maildir/tmp:

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

>N 1 sammy@example.com 2022-04-18 15:09 14/452 init

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

Message-Id: <20220418150946.EE6897E11A@@mail.example.com>

From: sammy@example.com

init

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

h

Output

>R 1 sammy@example.com 2022-04-18 15:09 14/452 init

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,

This is a test. Please confirm receipt!

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 subject line

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](mailto: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>