



Data Modelling Tools

AUTM08016

Topic 4 Build a LAMP Server



Server

Linux, Apache, M(ariaDB, ySQL), P(PHP, ython, erl)

Dr Diarmuid Ó Briain
Version 1.0 [01 January 2024]



TUS

Ollscoil Teicneolaíochta na Sionainne:
Lár Tíre, An tIarthar Láir
Technological University of the Shannon:
Midlands Midwest

Copyright © 2024 C²S Consulting

Licensed under the EUPL, Version 1.2 or – as soon they will be approved by the European Commission - subsequent versions of the EUPL (the "Licence");

You may not use this work except in compliance with the Licence.

You may obtain a copy of the Licence at:

https://joinup.ec.europa.eu/sites/default/files/custom-page/attachment/eupl_v1.2_en.pdf

Unless required by applicable law or agreed to in writing, software distributed under the Licence is distributed on an "AS IS" basis, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the Licence for the specific language governing permissions and limitations under the Licence.

Dr Diarmuid Ó Briain



Linux Version

```
~$ lsb_release -a | grep Description
Description:    Ubuntu 22.04.3 LTS
```

Apache2 Version

```
~$ apache2 -v
Server version: Apache/2.4.52 (Ubuntu)
Server built:   2023-10-26T13:44:44
```

MariaDB Version

```
~$ mariadb --version
mariadb Ver 15.1 Distrib 10.6.12-MariaDB, for debian-linux-gnu
(x86_64) using EditLine wrapper
```

php Version

```
~$ php --version | head -1
PHP 8.1.2-1ubuntu2.14 (cli) (built: Aug 18 2023 11:41:11) (NTS)
```

perl version

```
~$ perl --version | grep subversion
This is perl 5, version 34, subversion 0 (v5.34.0) built for
x86_64-linux-gnu-thread-multi
```

python version

```
~$ python3 --version
Python 3.10.12
```

Table of Contents

1. Introduction.....	5
1.1 Objectives.....	5
2. Webserver.....	6
2.1 LAMP Server.....	8
3. The LAMP Solution Stack.....	9
4. Apache HTTP Server.....	11
4.1 Install Apache2 and test.....	11
5. Hypertext PreProcessor (PHP).....	13
5.1 Install PHP.....	13
5.2 PHP Simple Calculator Example.....	14
6. Maria Database (MariaDB).....	16
6.1 Install MariaDB.....	16
7. phpMyAdmin.....	18
7.1 Introduction.....	18
7.2 Install phpMyAdmin.....	19
7.3 Connect to the database via phpMyAdmin.....	20
8. Programming Languages: Perl and Python.....	21
8.1 Install Perl.....	21
8.2 Install Python3.....	22
8.3 Simple PyCalculator Example.....	23
9. Exercise Laboratory #1.....	26
9.1 Build a LAMP Server.....	26

Table of Figures

Figure 1: Webserver.....	6
Figure 2: HTTP Communication.....	7
Figure 3: HTML from HTTP 200 OK message.....	7
Figure 4: LAMP Server extends Webserver.....	8
Figure 5: LAMP Server Solutions stack.....	9
Figure 6: Browse to Apache2 Server.....	11
Figure 7: Hello World HTML page on Apache2 Server.....	12
Figure 8: Hello World PHP page on Apache2 Server.....	14
Figure 9: MyCalculator a PHP example.....	15
Figure 10: My Simple Calculator.....	15
Figure 11: phpMyAdmin login window.....	20
Figure 12: Graphical view of the databases.....	20
Figure 13: init.py file.....	23
Figure 14: HTML Jinja2 templates.....	24
Figure 15: Running the Python3 Development Server.....	24
Figure 16: Simple PyCalculator.....	25

1. Introduction

LAMP is an acronym for software stack that includes GNU/Linux, the Apache web server, MariaDB database (MySQL), Hypertext Preprocessor (PHP), Python and Perl scripting languages. A LAMP stack facilitates the hosting of dynamic websites.

1.1 Objectives

At the end of this topic the learner will be able to build a LAMP Server:

- Install and test Apache2 Webserver
- Install and test PHP
- Install and test MariaDB
- Install and test Perl
- Install and test Python

2. Webserver

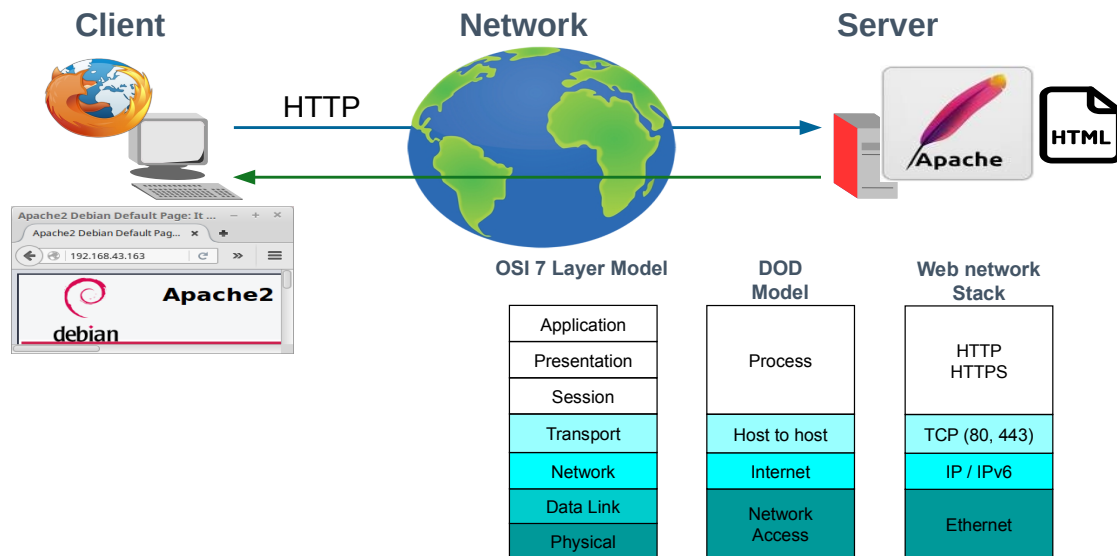


Figure 1: Webserver

At a high level, a webserver offers a Hypertext Transfer Protocol (HTTP) daemon or service at the Department of Defence (DoD) Model Process layer via the Transmission Control Protocol (TCP) at the Transport/Host to Host layer, typically on port 80. This serves up HyperText Markup Language (HTML) pages to a HTTP client, known as a web browser. Web browsers receive HTML documents, typically from the webserver, and render the documents into multimedia web pages. HTML describes the structure of a web page semantically.

Consider Figure 1 where the client on the left of the diagram makes a HTTP GET request of the server's TCP port 80. That port is serviced by a HTTP daemon or service, otherwise known as a webserver. In this example an Apache2 service is provided, though there are other options such as NginX.

```
Internet Protocol Version 4, Src: 10.0.0.20, Dst: 10.0.2.10
Transmission Control Protocol,
  Src Port: 54117, Dst Port: 80, Seq: 1, Ack: 1, Len: 253
Hypertext Transfer Protocol
  GET / HTTP/1.0\r\n
  Host: 10.0.2.10\r\n
  Accept: text/html, text/plain, text/sgml, text/css, app/xhtml+xml, */*;q=0.01\r\n
  Accept-Encoding: gzip, compress, bzip2\r\n
  Accept-Language: en\r\n
  User-Agent: Lynx/2.8.9dev.1 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/3.3.8\r\n  \r\n
```

The Apache2 Server responds to the HTTP GET request with a HTTP; 200 OK message. This response includes the text/html markup. The HTTP Client, the browser, receives the HTML as shown in Figure 2.

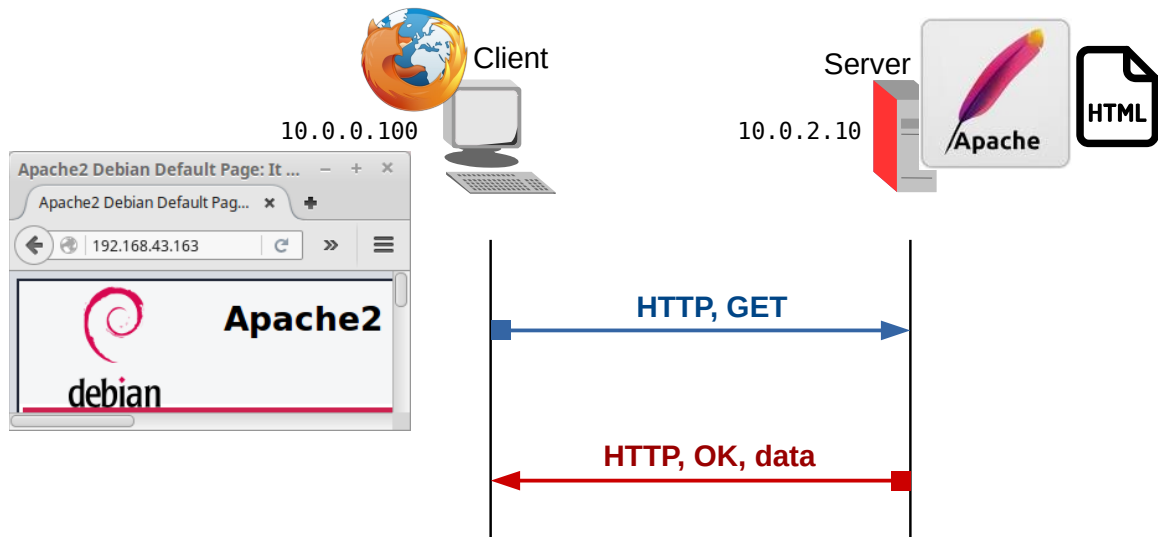


Figure 2: HTTP Communication

Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.0.20
Transmission Control Protocol, Src Port: 80, Dst Port: 54117, Seq: 1, Ack: 254, Len: 500

Hypertext Transfer Protocol

```
HTTP/1.1 200 OK\r\n
Date: Fri, 26 Feb 2016 18:33:12 GMT\r\n
Server: Apache/2.4.10 (Debian)\r\n
Last-Modified: Fri, 26 Feb 2016 18:32:10 GMT\r\n
ETag: "115-52cb08464d460"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 277\r\n
Connection: close\r\n
\r\n
Data (277 bytes)
```

<pre>0000 3c 68 74 6d 6c 3e 3c 62 6f 64 79 3e 3c 21 2d 2d 0010 20 67 65 6e 65 72 61 74 65 64 20 62 79 20 75 74 0020 69 6c 69 74 79 2e 70 79 3a 48 74 74 70 53 65 72 0030 76 69 63 65 20 2d 2d 3e 0a 3c 68 31 3e 73 76 72 0040 31 20 77 65 62 20 73 65 72 76 65 72 3c 2f 68 31 0050 3e 0a 3c 70 3e 54 68 69 73 20 69 73 20 74 68 65 0060 20 64 65 66 61 75 6c 74 20 77 65 62 20 70 61 67 0070 65 20 66 6f 72 20 74 68 69 73 20 73 65 72 76 65 0080 72 2e 3c 2f 70 3e 0a 3c 70 3e 54 68 65 20 77 65 0090 62 20 73 65 72 76 65 72 20 73 6f 66 74 77 61 72 00a0 65 20 69 73 20 72 75 6e 6e 69 6e 67 20 62 75 74 00b0 20 6e 6f 20 63 6f 6e 74 65 6e 74 20 68 61 73 20 00c0 62 65 65 6e 20 61 64 64 65 64 2c 20 79 65 74 2e 00d0 3c 2f 70 3e 0a 3c 6c 69 3e 65 74 68 30 20 2d 20 00e0 5b 27 31 30 2e 30 2e 32 2e 31 30 2f 32 34 27 2c 00f0 20 27 32 30 30 31 3a 32 3a 3a 31 30 2f 36 34 27 0100 5d 3c 2f 6c 69 3e 0a 3c 2f 62 6f 64 79 3e 3c 2f 0110 68 74 6d 6c 3e</pre>	<pre><html><body><!-- generated by ut ility.py:HttpSer vice -->.<h1>svr l web server</h1 >.<p>This is the default web pag e for this serve r.</p>.<p>The we b server softwar e is running but no content has been added, yet. </p>.eth0 - ['10.0.2.10/24', '2001:2::10/64'].</body></ html></pre>
---	---

```
</html>
<body><!-- generated by utility.py:HttpService -->.<
  <h1>svrl web server</h1>.<
  <p>This is the default web page for this server.</p>.<
  <p>The web server software is running but no content has been added, yet.</p>.<
  <li>eth0 - ['10.0.2.10/24', '2001:2::10/64']</li>.<
</body>
</html>
```

Figure 3: HTML from HTTP 200 OK message

2.1 LAMP Server

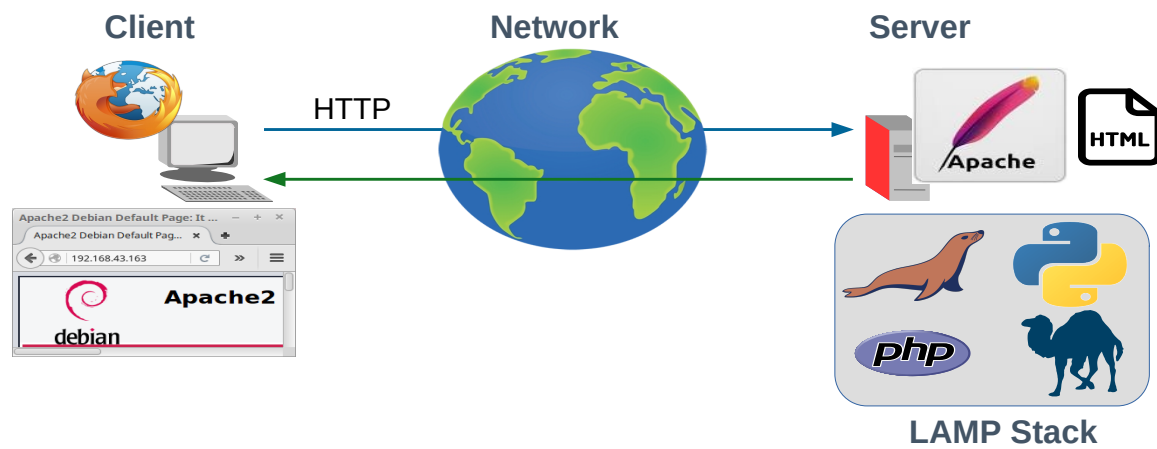


Figure 4: LAMP Server extends Webserver

The LAMP Server extends the service that the webserver offers. LAMP is a bundle of free and open-source software as a feasible alternative to expensive commercial packages. The original elements included:

- GNU/Linux Operating System
- Apache Webserver
- MySQL database
- PHP server-side scripting

3. The LAMP Solution Stack

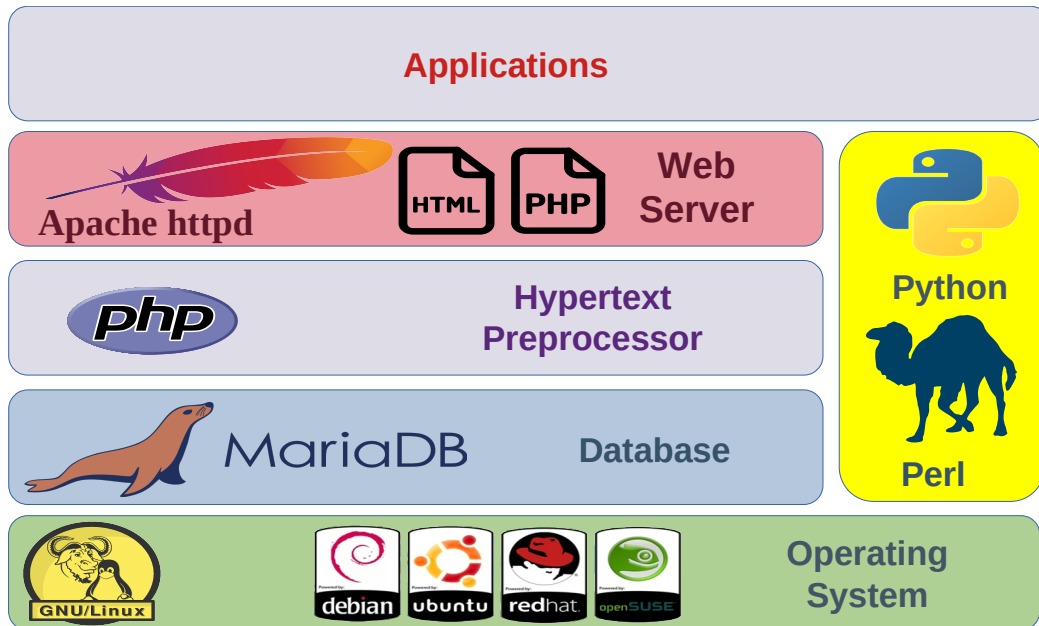


Figure 5: LAMP Server Solutions stack

Each element of the LAMP server combines to provide the overall stack:

- GNU/Linux** provides the operating system upon which the collections of software are based. It provides package management system to manage the packages and (W3Techs: as of 30 August 2021) accounts for 77% of all websites. For webserver, it is best to employ a primary GNU/Linux distribution. For example Fedora is based on RedHat but is not as secure and therefore does not make a good option whereas RedHat Enterprise Linux (RHEL) does. One exception to this is Ubuntu Server which is based on Debian GNU/Linux, it would still be best to employ Debian; however, Ubuntu is a common option for webserver too. In fact the majority of GNU/Linux webserver employ either Debian or Ubuntu. For Raspberry Pi devices the default operating system, Raspberry Pi OS (once called Raspbian), a trimmed down version of Debian, optimised for the Advanced Reduced Instruction Set Computer (RISC) Machines (ARM) hardware of the Raspberry Pi, is ideal.
- Apache2** Webserver provides the HTTP daemon service. Some solutions replace Apache2 with NginX and between Apache2 and NginX webserver they account for 66% of all known websites (W3Techs: as of 30 August 2021). Apache2 will be employed in this topic.

- **MySQL** was the original database that formed part of the LAMP stack; however, when MySQL, then owned by Sun Microsystems, was acquired by Oracle in 2010, there was a fear that it was not in Oracle's interest to maintain MySQL considering they were a database company with commercial offerings of their own. **MariaDB** was developed as a fork of MySQL to consolidate its future as an open-source database. There are other databases that could also be considered in a webserver stack such as PostgreSQL, MongoDB or even SQLite for small sites. **MariaDB** will be employed in this topic.
- Hypertext Preprocessing or often termed server-side scripting is carried out by **PHP** on 79% of all known websites (W3Techs: as of 30 August 2021). In the LAMP Server a Common Gateway Interface (CGI) is an Application Programming Interface (API) that supports writing programs that produces data dynamically rather than supplying the webserver with static HTML pages. Data is processed by the scripting language directly to HTTP output instead of to a markup file. In effect the CGI program extends the functionality of the server. PHP is currently the most popular language for this, though it is not the only one. In the past **Perl** was the preferred language and today there are many Perl programs still in use and **Python** as a popular language for many applications can be used as a CGI language in certain cases. All three of these languages will be installed in this workshop with simple examples demonstrated for both PHP and Python.

4. Apache HTTP Server



The Apache HTTP Server is a free and open-source cross-platform web server software.

The vast majority of Apache HTTP Server instances run on a GNU/Linux distribution and Apache is the world's most popular webserver as well as the first webserver to serve more than 100 million websites.

4.1 Install Apache2 and test

Install the Apache2 Server. Once complete use a web browser on another device and input the Internet Protocol (IP) address retrieved from the `hostname -I` command in the last step. This will display the default `index.html` page located at `/var/www/html/` on the Raspberry Pi, computer or Virtual Machine (VM).

```
~$ sudo apt install -y apache2
```

```
~$ cd /var/www/html
```

```
/var/www/html$ ls  
index.html
```

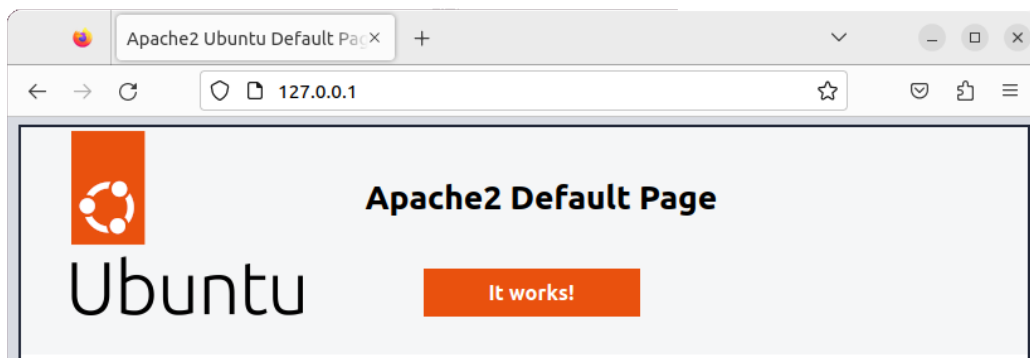


Figure 6: Browse to Apache2 Server

Create a new webpage to replace the default `index.html` page and test.

```
/var/www/html$ sudo mv index.html index.html.old
```

```
/var/www/html$ sudo vi index.html
```

```
<HTML>
  <HEAD>
    <TITLE>Hello World</TITLE>
  </HEAD>
  <BODY>
    Hello World, Apache2 HTML here
  </BODY>
</HTML>
:wq!
```

Change the ownership and mode.

```
/var/www/html$ sudo chown ada:www-data index.html
```

```
/var/www/html$ sudo chmod 770 index.html
```

```
/var/www/html$ ls -la
```

```
drwxr-xr-x 2 root    root    4096 Dec 22 14:12 .
drwxr-xr-x 2 root    root    4096 Dec 22 00:04 ..
-rwxr-xr-x 2 ada     www-data  90 Dec 22 13:53 index.html
-rw-r--r-- 1 root    root    10671 Dec 22 13:44 index.html.old
```

Test with the remote browser.

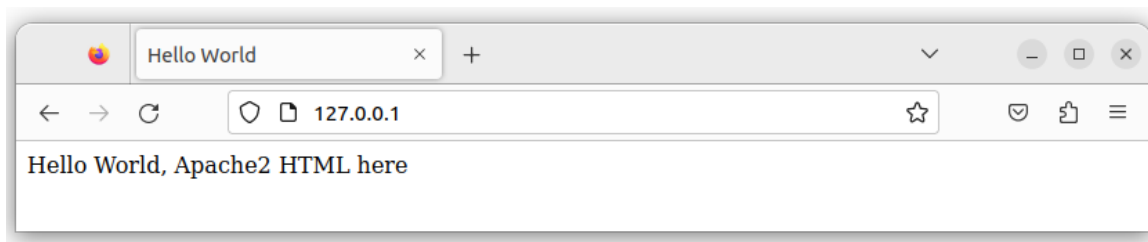


Figure 7: Hello World HTML page on Apache2 Server

5. Hypertext PreProcessor (PHP)

PHP is a general-purpose scripting language geared towards web development. PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a CGI executable.

On a web server, the result of the interpreted and executed PHP code, which may be any type of data such as generated HTML or binary image data, forms the whole or part of a HTTP response. Various web template systems, web content management systems, and web frameworks exist which can be employed to orchestrate or facilitate the generation of that response.

5.1 Install PHP

Install PHP, replace the test `index.html` file in `/var/www/html` with an `index.php` file. Restart the Apache2 webserver.

```
~$ sudo apt -y install php php-cgi libapache2-mod-php php-common php-pear php-mbstring
```

Disable the Apache2 module `mpm-event`. This module was designed to allow more requests to be served simultaneously; however, it conflicts with the Apache2 `php` module. Enable the `php` module.

```
~$ sudo a2dismod mpm_event
~$ sudo a2enmod php8.1
```

Create a PHP page.

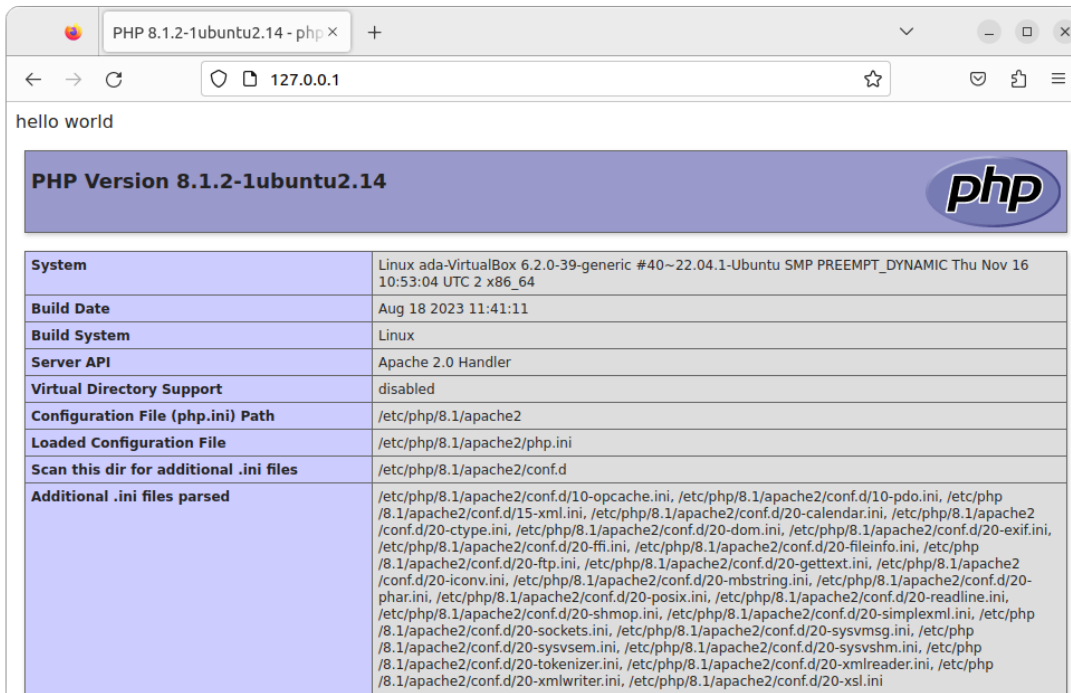
```
~$ cd /var/www/html
```

```
/var/www/html$ sudo rm index.html
/var/www/html$ sudo vi index.php
<?php
echo "hello world";
phpinfo( );
?>
:wq!
```

```
/var/www/html$ sudo systemctl restart apache2
```



Browse to the LAMP Server IP address and note the changes that are displayed.



hello world

PHP Version 8.1.2-1ubuntu2.14

System	Linux ada-VirtualBox 6.2.0-39-generic #40~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Nov 16 10:53:04 UTC 2 x86_64
Build Date	Aug 18 2023 11:41:11
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/apache2
Loaded Configuration File	/etc/php/8.1/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/apache2/conf.d
Additional .ini files parsed	/etc/php/8.1/apache2/conf.d/10-opcache.ini, /etc/php/8.1/apache2/conf.d/10-pdo.ini, /etc/php/8.1/apache2/conf.d/15-xml.ini, /etc/php/8.1/apache2/conf.d/20-calendar.ini, /etc/php/8.1/apache2/conf.d/20-ctype.ini, /etc/php/8.1/apache2/conf.d/20-dom.ini, /etc/php/8.1/apache2/conf.d/20-exif.ini, /etc/php/8.1/apache2/conf.d/20-ffi.ini, /etc/php/8.1/apache2/conf.d/20-fileinfo.ini, /etc/php/8.1/apache2/conf.d/20-ftp.ini, /etc/php/8.1/apache2/conf.d/20-gettext.ini, /etc/php/8.1/apache2/conf.d/20-iconv.ini, /etc/php/8.1/apache2/conf.d/20-mbstring.ini, /etc/php/8.1/apache2/conf.d/20-phar.ini, /etc/php/8.1/apache2/conf.d/20-posix.ini, /etc/php/8.1/apache2/conf.d/20-readline.ini, /etc/php/8.1/apache2/conf.d/20-shmop.ini, /etc/php/8.1/apache2/conf.d/20-simplexml.ini, /etc/php/8.1/apache2/conf.d/20-sockets.ini, /etc/php/8.1/apache2/conf.d/20-sysvmsg.ini, /etc/php/8.1/apache2/conf.d/20-sysvsem.ini, /etc/php/8.1/apache2/conf.d/20-sysvshm.ini, /etc/php/8.1/apache2/conf.d/20-tokenizer.ini, /etc/php/8.1/apache2/conf.d/20-xmlreader.ini, /etc/php/8.1/apache2/conf.d/20-xmlwriter.ini, /etc/php/8.1/apache2/conf.d/20-xsl.ini

Figure 8: Hello World PHP page on Apache2 Server

5.2 PHP Simple Calculator Example

Consider the simple PHP program shown in Figure 9. The first part of the program is a HTML page and form. When the form is filled and the **ADD** submit button is pressed, instead of calling another page or program, the PHP code between the `<?php ?>` tags is executed and the values from the form are passed as an array `$_POST` to the PHP code. The values for `$_POST['num1']` and `$_POST['num2']` are added and the result is printed as HTML. Past the PHP tags the remainder of the HTML page is returned.

```

/var/www/html$ cat calculator.php

<TITLE>My Simple Calculator</TITLE>
<HEAD></HEAD>

<BODY style="background-color:white;font-family:arial;color:#000000;">
<IMG SRC="images/tus_logo.png" alt="TUS logo">
<BR>
<H2>My Simple Calculator</H2>

<FORM action="" method="post">
<LABEL>Enter Num1:</LABEL>
<INPUT type="text" name="num1" /><BR>
<LABEL>Enter Num2:</LABEL>
<INPUT type="text" name="num2" /><BR><BR>
<INPUT type="submit" name="btn_submit" value="ADD">
</FORM>

<?php
    if(isset($_POST['btn_submit'])){ // is variable set & not NULL //
        $num1 = $_POST['num1']; // assign box 1 value to $num1 //
        $num2 = $_POST['num2']; // assign box 2 value to $num2 //
        $total = $num1 + $num2; // add and assign to $total //
        print ("<P><B>The total value is: " . $total . "</B></P>");
    }
?>

</BODY>
</HTML>

```

Figure 9: MyCalculator a PHP example

This PHP page returns the following HTML once the pre-processing is complete.

```

<TITLE>My Simple Calculator</TITLE>
<HEAD></HEAD>
<BODY style="background-color:white;font-family:arial;color:#000000;">
    <IMG SRC="images/tus_logo.png" alt="TUS logo">
    <BR>
    <H2>My Simple Calculator</H2>
    <FORM action="" method="post">
        <LABEL>Enter Num1:</LABEL>
        <INPUT type="text" name="num1" /><BR>
        <LABEL>Enter Num2:</LABEL>
        <INPUT type="text" name="num2" /><BR><BR>
        <INPUT type="submit" name="btn_submit" value="ADD">
    </FORM>
    <P><B>The total value is: 9</B></P>
</BODY>
</HTML>

```

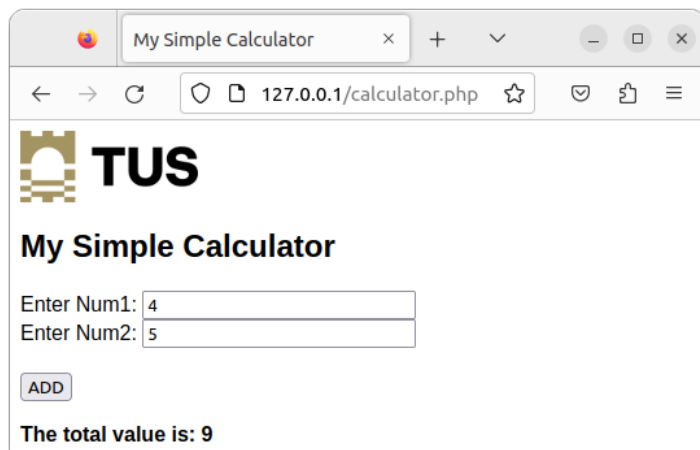


Figure 10: My Simple Calculator

6. Maria Database (MariaDB)



MariaDB is a free RDBMS that implements the SQL standard. MariaDB is a fork of MySQL and essentially structures MariaDB uses are the same as MySQL as it was designed as a drop-in replacement of MySQL.

MariaDB has added a significant number of new features, which makes MariaDB better in terms of performance and user-orientation than MySQL.

While both databases are free and open-source; however, when Oracle acquired Sun Microsystems, in 2010, there was a potential conflict of interest between MySQL and Oracle's commercial database - Oracle Database Server. As a result, some engineers from the MySQL programme created a MariaDB fork of the MySQL code base.

Both databases can be found on most distributions of GNU/Linux and exists for many other operating systems including FreeBSD, Sun Solaris, SCO UNIX, BSD UNIX and Microsoft Windows. MariaDB and MySQL software and documentation can be downloaded from:

- MariaDB - <https://mariadb.org>
- MySQL Community Server - <https://dev.mysql.com>

Many Graphical tools are available for MariaDB and MySQL. The **phpMyAdmin** project is an excellent example that can be acquired at:

<http://www.phpmyadmin.net>

6.1 Install MariaDB

Insure that the following packages installed on your system.

- **mariadb-server**
- **mariadb-client**

Here is an example of an installation on a Debian GNU/Linux Server.

```
~$ sudo apt install -y mariadb-server mariadb-client php-mysql
```

Restart the Apache2 webserver.

```
~$ sudo systemctl restart apache2
```


Run the `mysql_secure_installation` shell script which improves the security of the MariaDB installation in the following ways:

- Sets a password for root accounts.
- Remove root accounts that are accessible from outside the local host.
- Remove anonymous-user accounts.
- Remove the test database, which by default can be accessed by anonymous users.

```
~$ sudo mysql_secure_installation
```

```
Enter current password for root (Enter for none): <Enter>
Set root password? [Y/n] Y
New Password: rootpass
Re-enter new Password: rootpass
Remove anonymous users? [Y/n] Y
Disallow root login remotely? [Y/n] Y
Remove test database and access to it? [Y/n] Y
Reload privilege tables now? [Y/n] Y
```

Thanks for using MariaDB!

Login as the `root` user and create an alternative administration user. Give this new user all privileges on the database. Flush privileges to reload the grant tables in the MariaDB database enabling the changes to take effect without reloading or restarting MariaDB service.

```
~$ sudo mysql --user=root --password
```

```
Enter password: rootpass
```

```
MariaDB [(none)]>
```

```
> CREATE USER admin@localhost IDENTIFIED BY 'admpass';
```

```
Query OK, 0 rows affected (0.002 sec)
```

```
> GRANT ALL PRIVILEGES ON *.* TO admin@localhost;
```

```
Query OK, 0 rows affected (0.002 sec)
```

```
> FLUSH PRIVILEGES;
```

```
Query OK, 0 rows affected (0.002 sec)
```

```
> exit;
```

7. phpMyAdmin

7.1 Introduction

phpMyAdmin is a free and open source administration tool for MariaDB and MySQL. It is a portable web application written primarily in PHP and it is one of the most popular MariaDB/MySQL administration tools, especially for web hosting services.

phpMyAdmin can manage a complete MariaDB/MySQL server (needs a super-user) or as little as a single database. To accomplish the latter, a properly established MariaDB user with read/write privileges to the desired database is required. **phpMyAdmin** can accomplish the following tasks:

- create and drop databases
- create, copy, drop, rename and alter tables
- do table maintenance
- delete, edit and add fields
- execute any SQL-statement, even batch-queries
- manage keys on fields
- load text files into tables
- create (*) and read dumps of tables
- export (*) data to Comma-Separated Values (CSV), eXtensible Markup Language (XML) and Latex formats
- administer multiple servers
- manage MariaDB users and privileges
- check referential integrity
- using Query-By-Example (QBE), create complex queries automatically connecting required tables
- create Portable Document Format (PDF) graphics of your Database layout
- search globally in a database or a subset of it
- transform stored data into any format using a set of predefined functions.



7.2 Install phpMyAdmin

Install **phpMyAdmin**, during the install an option of webserver to be used will be given. Select **apache2** as the webserver.

```
~$ sudo apt install -y phpmyadmin
```

```
[*] Apache2  
[ ] lighttpd    <ok>
```

A wizard will then be presented, give an administrator password and provide the database root password.

```
Configure database for phpmyadmin with dbconfig-common?  
  
    <Yes>          <No>  
  
MySQL application password for phpmyadmin: myadmpass  
Password confirmation: myadmpass
```

Provide a **phpMyAdmin** access password for the user **phpmyadmin**: **myadmpass**.

7.2.1 Enable the MySQL Improved Extension (MySQLi)

Enable the PHP MySQL Improved Extension (MySQLi) extension. This extension gives access to the functionality provided by MySQL 4.1 and above.

```
~$ sudo phpenmod mysqli
```

7.2.2 Complete the installation

Add a soft link from **phpmyadmin** to the Apache2 webserver root directory.

```
~$ sudo ln -s /usr/share/phpmyadmin /var/www/html
```

```
~$ sudo chown -R ada:www-data /var/www/html
```

```
~$ sudo chmod -R 770 /var/www/html
```

Finally restart Apache2 to effect the changes.

```
~$ sudo systemctl restart apache2
```

7.3 Connect to the database via phpMyAdmin

Browse to the **phpMyAdmin** webserver and login using the **admin** credentials.



Figure 11: phpMyAdmin login window

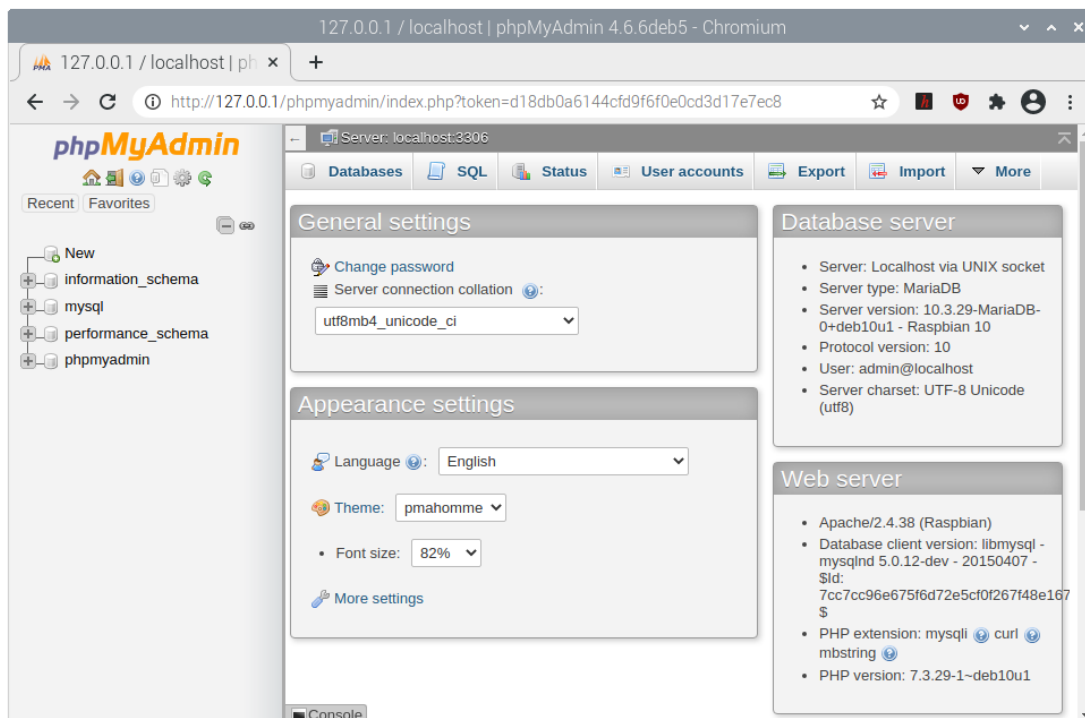


Figure 12: Graphical view of the databases

8. Programming Languages: Perl and Python

The final step in the creation of the LAMP Server is the installation of Perl and Python programming languages. Practical Extraction and Reporting Language (Perl) and Python are both high-level, general-purpose, interpreted, dynamic programming languages.

Perl was developed by Larry Wall in 1987 as a text processing language, it maintains a similarity to C syntax though it supports both Object-Oriented (OOP) and procedural programming.

Guido van Rossum created Python in 1991 with an emphasis on code readability.

While Perl and Python originate from similar backgrounds and have many similar features, they were designed from different philosophies. Perl places emphasis on the support of common application-oriented tasks. For example built-in regular expressions, file scanning and report generating features whereas Python places its emphasis on common programming methodologies such as data structure design and object-oriented programming. In Perl there are multiple ways to do everything, whereas Python encourages programmers to write readable and easily maintainable code by providing an elegant but not overly cryptic notation.

8.1 Install Perl

```
~$ sudo apt install perl
```

```
~$ perl --version
```

```
This is perl 5, version 34, subversion 0 (v5.34.0) built for x86_64-  
linux-gnu-thread-multi  
(with 60 registered patches, see perl -V for more detail)
```

```
Copyright 1987-2021, Larry Wall
```

```
Perl may be copied only under the terms of either the Artistic  
License or the GNU General Public License, which may be found in the  
Perl 5 source kit.
```

```
Complete documentation for Perl, including FAQ lists, should be found  
on this system using "man perl" or "perldoc perl". If you have  
access to the Internet, point your browser at http://www.perl.org/,  
the Perl Home Page..
```



8.2 Install Python3

Install Python3 and its package installer, **pip**.

```
~$ sudo apt install python3 python3-pip idle3 python3.10-venv
```

```
~$ python3 --version  
Python 3.7.3
```

```
~$ python3 -m pip --version  
pip 18.1 from /usr/lib/python3/dist-packages/pip (python 3.7)
```

Create a Python virtual environment.

```
~$ python3 -m pip install --upgrade pip  
Looking in indexes: https://pypi.org/simple,  
https://www.piwheels.org/simple  
Collecting pip  
  Downloading  
https://files.pythonhosted.org/packages/ca/31/b88ef447d595963c0106099  
8cb329251648acf4a067721b0452c45527eb8/pip-21.2.4-py3-none-any.whl  
(1.6MB)  
100% |████████████████████████████████████████| 1.6MB 112kB/s  
Installing collected packages: pip  
Successfully installed pip-21.2.4
```

Create a python virtual environment.

```
~$ python3 -m venv ~/.venv
```

Activate the virtual environment.

```
~$ source ~/.venv/bin/activate
```

Install the Python Flask web development framework.

```
(.venv)~$ python3 -m pip install flask
```

Install the **Mariadb/Python** connector.

```
(.venv)~$ sudo apt install -y libmariadb3 libmariadb-dev  
(.venv)~$ python3 -m pip install mariadb
```

Install the Yet Another Markup Language (YAML) module.

```
(.venv)~$ python3 -m pip install pyyaml
```



8.3 Simple PyCalculator Example

```

PyCalculator
├── init.py
├── static
│   ├── images
│   └── tus.png
└── templates
    ├── answer.html
    ├── calculator.html
    └── layout.html

```

3 directories, 5 files

Consider the Python version of the Simple Calculator in Figure 13. This Python program imports the Flask micro web framework and a Flask app is instantiated as `app`.

```

~$ cat init.py
from flask import Flask, render_template, request

# // Flask app //
app = Flask(__name__)

@app.route("/")
def index():
    """The calculator index page"""
    return render_template("calculator.html")

@app.route("/answer", methods=["GET", "POST"])
def answer():
    """The calculator answer page"""
    post_data = dict(request.form)
    num1, num2 = int(post_data["num1"]), int(post_data["num2"])
    num3 = num1 + num2
    data = (num1, num2, num3)
    return render_template("answer.html", data=data)

if __name__ == "__main__":
    app.run(debug=True)

```

Figure 13: `init.py` file

The program has two routes, a default route and an answer route. Also associated with this program are three templates. The first `layout.html`, as in Figure 14, is simply the header and end for both webpages. The other two pages are fit in between the Jinja2 constructs `{% block content %}` and `{% endblock %}`. When the website is browsed to the `calculator.html` page is displayed. It extends `layout.html`. The user enters two numbers and clicks the **ADD** button. The `init.py` program extracts these two numbers via the Flask `request.form` method. The numbers are added and all three are send as a tuple to the `answer.html` page where the three numbers are laid out as a response. This is evident from Figure 15.

```

~$ cat layout.html
<TITLE>My Simple Calculator</TITLE>
<HEAD></HEAD>
<BODY style="background-color:white;font-family:arial;color:#black;">
  <IMG SRC="{{ url_for('static', filename='images/tus.png') }}" alt="TUS ">
  <BR>
  {% block content %}
  {% endblock %}
</BODY>
</HTML>

~$ cat calculator.html
{% extends "layout.html" %}
{% block content %}
<H1>My Simple Calculator</H1>
  <FORM action="/answer" method="POST" enctype="multipart/form-data">
    <P>Enter Num1: </td><td><input type = "text" name = "num1"></P>
    <P>Enter Num2: </td><td><input type = "text" name = "num2"></P>
    <P><INPUT type = "submit" value = "ADD" /></P>
  </FORM>
{% endblock %}

~$ cat answer.html
{% extends "layout.html" %}
{% block content %}
<H1>My Simple Calculator</H1>
<H3>{{ data[0] }} + {{ data[1] }} = {{ data[2] }}</H3>
{% endblock %}

```

Figure 14: HTML Jinja2 templates

Run the **PyCalculator** using the development server as in Figure 15. A browse to the webserver will result in the webpages shown in Figure 16.

```

~$ python3 PyCalculator/init.py
* Serving Flask app 'init'
* Debug mode: on
WARNING: This is a development server. Do not use it in
a production deployment. Use a production WSGI server
instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 764-273-319

127.0.0.1 - - [16/Oct/2022 01:58:26] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [16/Oct/2022 01:58:26] "GET
/static/images/tus_logo.png HTTP/1.1" 304 -
127.0.0.1 - - [16/Oct/2022 01:58:34] "POST /answer HTTP/1.1" 200 -
127.0.0.1 - - [16/Oct/2022 01:58:34] "GET
/static/images/tus_logo.png HTTP/1.1" 304 -

```

Figure 15: Running the Python3 Development Server

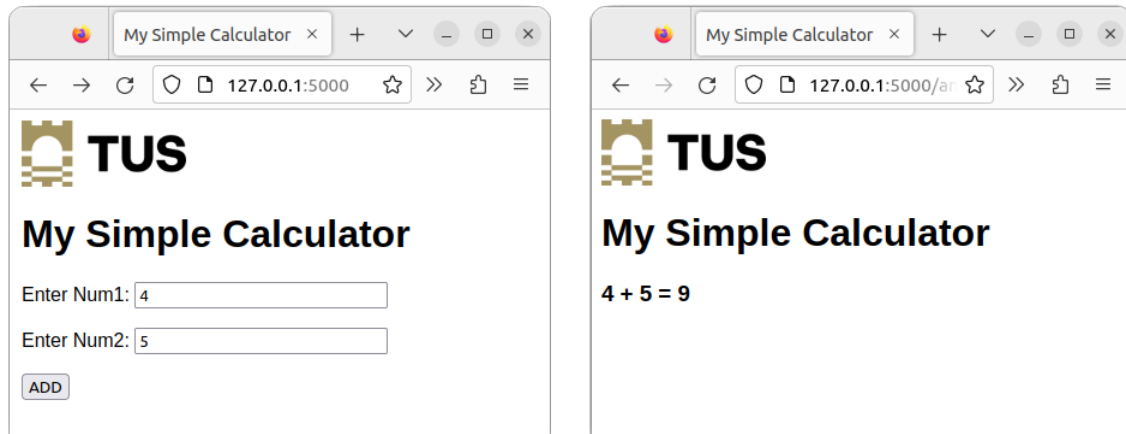


Figure 16: Simple PyCalculator

9. Exercise Laboratory #1

9.1 Build a LAMP Server

- Using either a Raspberry Pi or a GNU/Linux image from an Internet Cloud Provider build a full stack LAMP Server, include:
 - Maria DB Database
 - Apache Webserver
 - PHP
 - Perl
 - Python3.
- Install **phpMyAdmin** to administrate the database.
- Build a simple database.

Notes: