

1

Installation of PHP, MariaDB, and Apache

A few years ago, one would have had to walk over to the closest pizza store to order a pizza, go over to the bank to transfer money from one account to another account, and go to the book store and spend hours looking for the right book. Today, we can order a pizza, transfer money across accounts and countries, and circle down on the right book in a matter of minutes by clicking on a button. The Internet has revolutionized a lot of business processes that are known to us. Today, it is almost impossible to come across people who haven't heard or used the Internet. Now that we are talking about the Internet, how does a person, a business, a company, or a process get onto the Internet? We use web pages and websites to build our footprint on the Internet, and this chapter is a step-by-step journey of installing the components such as a web server, the libraries for a server-side scripting language, and a relational database server that would be required to build a website.


The most popular architecture in web development is the client-server architecture; a client is considered to be a PC from which a user requests for the content on the Web or it could be another web page looking for a data. A server is software that receives requests from users on the Internet and delivers the requested web content. Based on the type of request, the server may need the support of the server-side scripting programs to perform complex operations and execute site-specific process. A server-side scripting language is used to write scripts that are executed by the web server to parse the requests from the client and generate the necessary response that has to be delivered back to the client. If the data that is used for the website has to be persistent, then the data has to be stored in a file or a database. In this book, we will be working with the Apache web server, the PHP server-side scripting language, and the MariaDB database server for building our websites. Apache, MariaDB, and PHP (AMP) are **open source software (OSS)** available for free and for all the popular operating systems.

Apache HTTP Server has been the most popular web server since 1996. According to a survey done by Netcraft in July 2013, Apache HTTP Server is the preferred web server for over 50 percent of all active websites on the Internet. Apache HTTP Server is also popularly used as a load balancer. Load balancing, as the name suggests, is a method in which requests from clients are distributed across multiple computers to handle the work load. By employing a load balancer with a cluster of computers running our application, we can achieve high availability for our website. We will be taking a deeper dive into load balancing and high availability concepts in later chapters.

PHP is a popular server-side scripting language that is commonly used for web development, and is currently used by more than 244 million websites. The stable version of PHP recommended at the time of this writing is 5.5; having said this, many of the existing websites and hosts use PHP 5.4, PHP 5.3, or versions less than PHP 5.3. We will be using PHP 5.5 in this book. PHP is an interpreted programming language; the PHP code is executed line-by-line and is converted into **operation code (opcode)**. Opcode consists of machine language instructions that specify the operation that is to be performed. As the PHP code has to be interpreted for every request, the time taken to complete a request can be slower when compared to precompiled languages, the reason being that the compiled languages compile the code only once and execute the compiled code whenever required. However, PHP 5.5 comes with an inbuilt opcode cache that stores the output of the PHP bytecode compiler in the memory. The opcode cache helps in reducing the time taken for interpretation of the PHP code for future requests and for disk input-output operations. The opcode cache is not enabled by default and we will enable it in *Chapter 7, Caching*.

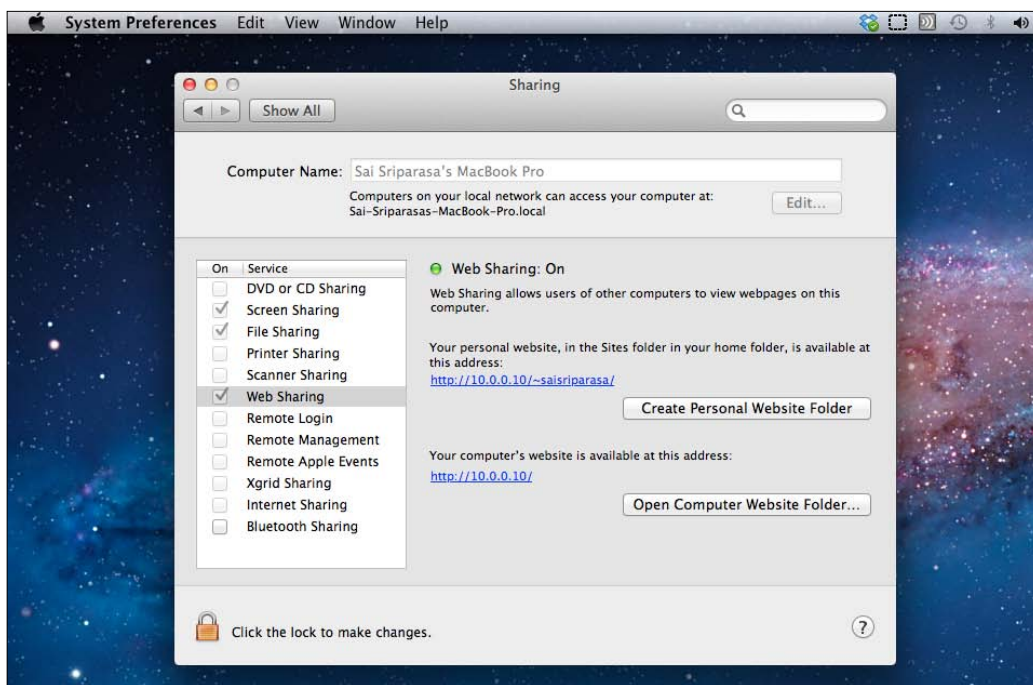
 PHP 5.3 is the most installed version.

MariaDB is a **relational database management system (RDBMS)** that is forked from the popular MySQL database management system. It is claimed to be a drop-in replacement database for MySQL. MariaDB was initially released by Michael "Monty" Widenius and a team of core MySQL developers in January 2009, after concerns were raised by the original developers of MySQL about the direction in which MySQL was headed after it was acquired by Oracle. As MariaDB is considered to be a drop-in replacement, the developers working on MariaDB take a lot of care to make sure that the code that they add is compatible with the existing MySQL APIs and commands.

 Subqueries, replication of data, and indexing are faster in MariaDB when compared to MySQL.

Installing AMP on Mac OS X

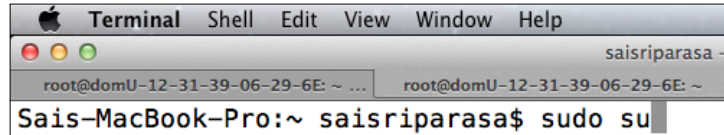
On a MAC OS X operating system, for web development using the AMP stack, the first step will be to turn on the Web Sharing option. Web Sharing opens up the required ports for HTTP, thereby allowing other users to view web pages that are available on this server. To turn on Web Sharing, click on the System Preferences icon available in the dock, click on the Sharing icon, and make sure that the **Web Sharing** checkbox is checked. If not, click on the checkbox next to **Web Sharing** to save the settings, then click on the lock icon at the bottom-left corner of the **System Preferences** window as shown in the following screenshot:



Now we are all set to start working on the AMP stack. We will begin by opening up a terminal window that provides us a shell to communicate with the operating system and allows us to execute the commands. To open up a terminal window, click on **Finder** in the dock, then click on **Applications**, then click on **Utilities**, and finally double-click on **Terminal**.

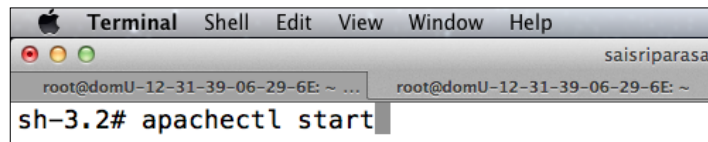
[ The path can be explained as **Finder** | **Applications** | **Utilities** | **Terminal**.]

The Apache web server, which already comes preinstalled with Mac OS X, is system software and regular user accounts do not have enough permission to work with Apache. In order to work with Apache, we would need root access, and we would use the **Terminal** window to request root access. For requesting root access, type `sudo su` in the terminal app. On hitting the return key, the operating system prompts the user for the root user's password. The following screenshot displays the terminal app:




```
Terminal Shell Edit View Window Help
saisriparasa
root@domU-12-31-39-06-29-6E: ~ ... root@domU-12-31-39-06-29-6E: ~
Sais-MacBook-Pro:~ saisriparasa$ sudo su
```

Once we have root access, the next step is to start the preinstalled Apache web server. We will be using the `start` command to fire up our Apache web server as shown in the following screenshot:

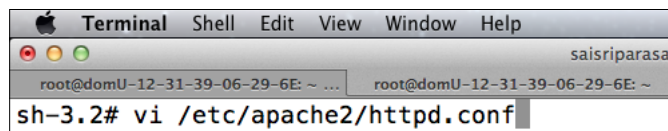


```
Terminal Shell Edit View Window Help
saisriparasa
root@domU-12-31-39-06-29-6E: ~ ... root@domU-12-31-39-06-29-6E: ~
sh-3.2# apachectl start
```

Now that we have our web server up-and-running, it is time for us to look at the existing settings of our Apache web server. These settings are stored in the `httpd.conf` file. This configuration file, along with other configurations files for Apache, is stored in the `/etc/apache2` folder. I will be using the `vi` text editor for accessing and editing the file, but any text editor of your choice can be used to modify this configuration file.

[ A good way to familiarize yourself with the `vi` editor is to use Vimtutor, a tutor designed to help working with the `vi` editor.]

Keep in mind that we would need root access to modify this file, and we have already requested for root permissions in the terminal window. The following screenshot displays the usage of the `vi` editor for accessing and editing the `httpd.conf` file stored in the `/etc/apache2` folder:



```
Terminal Shell Edit View Window Help
saisriparasa
root@domU-12-31-39-06-29-6E: ~ ... root@domU-12-31-39-06-29-6E: ~
sh-3.2# vi /etc/apache2/httpd.conf
```

The first change that we will be making to the `httpd.conf` file is to modify `DocumentRoot`. The `DocumentRoot` folder contains all files that are accessible to the web server. Any files outside `DocumentRoot` are not accessible to the web server, and therefore are not available to the outside world. Comment out the existing `DocumentRoot` by adding `#` at the beginning of the current `DocumentRoot`. Add a new line beneath and set it to a preferred location. We will be using `/var/www` as our `DocumentRoot` shown as follows:

```
<IfDefine WEBSHARING_ON>
#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
#DocumentRoot "/Library/WebServer/Documents"
DocumentRoot "/var/www"
```

The second change that we will be making is to replace the value of the existing directory that has been set to the new the directory location for `DocumentRoot`, as shown in the following screenshot:

```
#
# This should be changed to whatever you set DocumentRoot to.
#
#<Directory "/Library/WebServer/Documents">
<Directory "/var/www">
  #
  # AllowOverride controls what directives may be placed in .htaccess files.
  # It can be "All", "None", or any combination of the keywords:
  #   Options FileInfo AuthConfig Limit
  #
  AllowOverride All

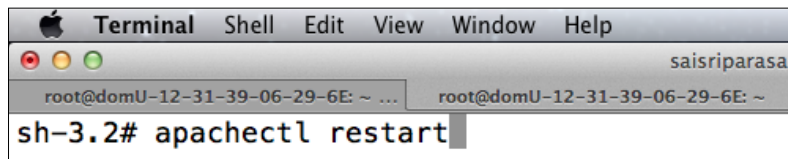
  #
  # Controls who can get stuff from this server.
  #
  Order allow,deny
  Allow from all
</Directory>
```

Now save the changes, and exit from the configuration file. Apache web server reads this configuration file upon start and in order to make these changes active, we will have to restart the web server.



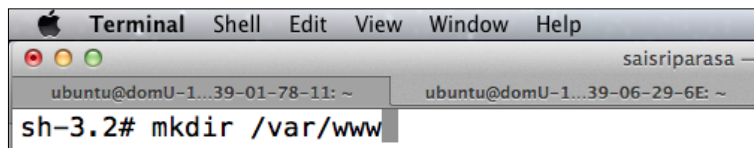
It is recommended to use the `configtest` option provided by `apachectl` to test the configuration before restarting Apache.

We will be using the `restart` command to restart the Apache web server as shown in the following screenshot:



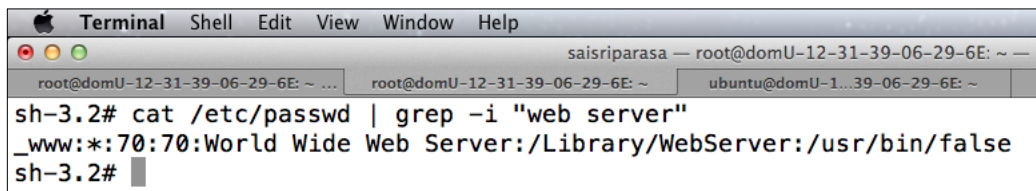
```
Terminal Shell Edit View Window Help
saisriparasa
root@domU-12-31-39-06-29-6E: ~ ... root@domU-12-31-39-06-29-6E: ~
sh-3.2# apachectl restart
```

We have now completed setting the document root, and have restarted the Apache web server so that it can start accessing the document root. Now we will have to create a folder that will be used as the document root. To create a folder for the document root, we will be using the `mkdir` command as shown in the following screenshot:



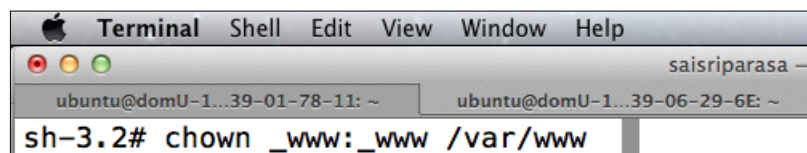
```
Terminal Shell Edit View Window Help
saisriparasa
ubuntu@domU-1...39-01-78-11: ~ ubuntu@domU-1...39-06-29-6E: ~
sh-3.2# mkdir /var/www
```

Now that the document root is created, we will have to provide Apache web server with the required access to this folder. This access would allow Apache to read, write, and/or execute the files in the document root as needed. The `/etc/passwd` file stores the list of users that are available on this machine and gives a description of their role. We will use this file to get the user identity for the Apache web server shown as follows:



```
Terminal Shell Edit View Window Help
saisriparasa
root@domU-12-31-39-06-29-6E: ~ ... root@domU-12-31-39-06-29-6E: ~ ubuntu@domU-1...39-06-29-6E: ~
sh-3.2# cat /etc/passwd | grep -i "web server"
_www:*:70:70:World Wide Web Server:/Library/WebServer:/usr/bin/false
sh-3.2#
```

Apache is registered as a `_www` user and the description of the user is **World Wide Web Server**. The next step is to provide ownership access for the `_www` user to our document root folder `/var/www`. We will be using the `chown` command to change the owner and group access for our document root shown as follows:



```
Terminal Shell Edit View Window Help
saisriparasa
ubuntu@domU-1...39-01-78-11: ~ ubuntu@domU-1...39-06-29-6E: ~
sh-3.2# chown _www:_www /var/www
```

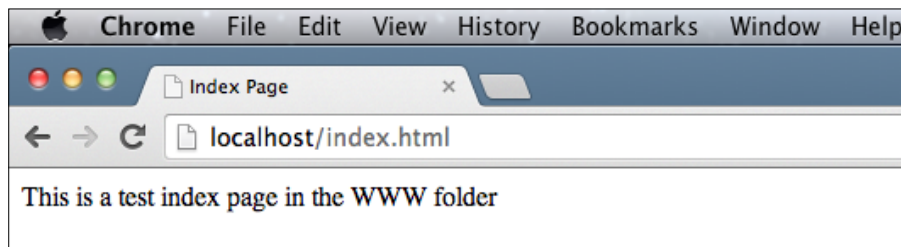
Use the `cd` command to enter into the `/var/www` folder and use your favorite editor to create a test HTML web page. The aim of this web page would be to determine if Apache is working as expected and if it serves the content as requested. The following screenshot displays the HTML web page:

```


Terminal Shell Edit View Window Help
saisriparasa — root@domU-12-31-39-06-29-6E: ~ — vim — 115
root@domU-12-31-39-06-29-6E: ~ ... root@domU-12-31-39-06-29-6E: ~ ubuntu@domU-1...39-06-29-6E: ~
<html>
  <head>
    <title>Index Page</title>
  </head>
  <body>
    <p>This is a test index page in the WWW folder</p>
  </body>
</html>

```

Now save the HTML file and load the web page in a web browser. To access this web page, we will build the URL by appending the file name to `localhost/`. Now add this URL to the address bar in your favorite web browser and hit *Return*. We should receive the success message that has been added in between the paragraph tags shown as follows:



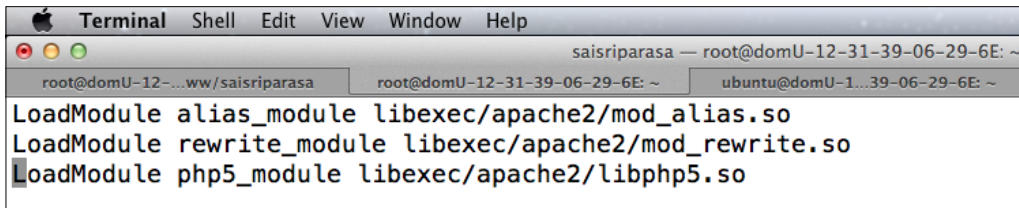
The `localhost` hostname in computer networking is considered to be the current computer, and the users can use this as a hostname to access the computer's internal network via the loopback interface. The IP address that this hostname is mapped to is `127.0.0.1`, and this IP address can be interchanged with `localhost` to load the files.


[

 127.0.0.1 is the IP address for the IPv4 loopback addresses. For the IPv6 loopback address, use `::1`. That is a 128-bit number with the first 127 bits being 0 and the 128th bit being 1.

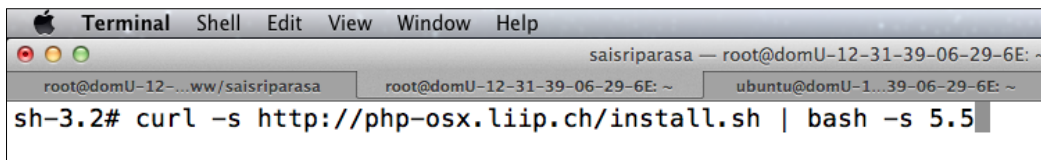
]

We have successfully tested that Apache is working. Now let us turn our attention to PHP. PHP comes preinstalled out of the box; the PHP version on Mac Mavericks is 5.4, while the PHP version on Mac Lion is 5.3. PHP is by default turned off for web development on Apache and needs to be turned on by uncommenting the line that has the instructions to load the `php5_module` when Apache is started. Open up the Apache configuration file (`/etc/apache2/httpd.conf`) that we worked on earlier in this chapter to set the document root. Search for the string `php5_module` and uncomment that line by removing the `#` tag shown as follows:



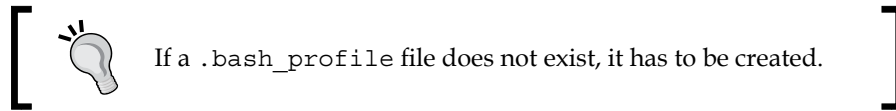
```
Terminal Shell Edit View Window Help
saisriparasa — root@domU-12-31-39-06-29-6E: ~
root@domU-12-...ww/saisriparasa root@domU-12-31-39-06-29-6E: ~ ubuntu@domU-1...39-06-29-6E: ~
LoadModule alias_module libexec/apache2/mod_alias.so
LoadModule rewrite_module libexec/apache2/mod_rewrite.so
LoadModule php5_module libexec/apache2/libphp5.so
```

We will have to restart the web server for Apache to recognize the changes that we have made in the configurations. Let us use the `restart` command that we have used earlier to restart our Apache HTTP Server. The current stable version of PHP is 5.5 and we will be installing PHP 5.5 in the next few steps. We will leave the preinstalled PHP package as it is, and this preinstalled version can be used for testing features across versions. For the installation of PHP 5.5, we will be using the shell script that is available on <http://php-osx.liip.ch/>; they provide a shell script that installs and builds PHP. We will be using the `curl` command to make a command-line request to the shell script and mention 5.5 as the version of PHP that we would want to install, as shown in the following screenshot:

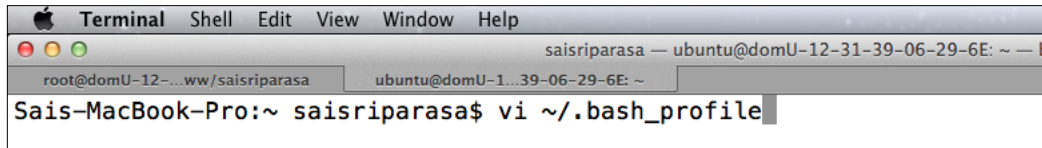


```
Terminal Shell Edit View Window Help
saisriparasa — root@domU-12-31-39-06-29-6E: ~
root@domU-12-...ww/saisriparasa root@domU-12-31-39-06-29-6E: ~ ubuntu@domU-1...39-06-29-6E: ~
sh-3.2# curl -s http://php-osx.liip.ch/install.sh | bash -s 5.5
```

Upon the successful execution of the earlier `curl` command, PHP 5.5 is now installed in `/usr/local/php5`. We will have to add the location of the binaries of PHP 5.5 to the `PATH` variable, in order to start using it. We will be using the `.bash_profile` file for appending the location of the PHP 5.5 binaries to the `PATH` variable. The `.bash_profile` file is commonly used for storing configurations that have to be loaded when a user's login is successful, and since Mac OS X's terminal app runs a login in the background every time a terminal shell is opened, the configurations that we add to the `.bash_profile` file are loaded by default.

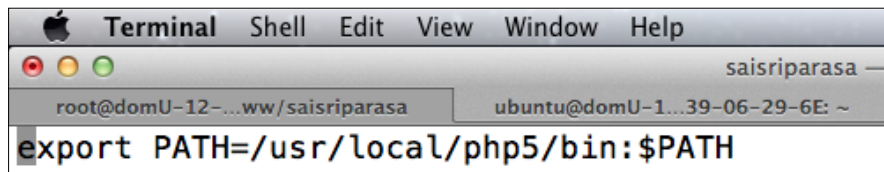


Use a text editor of your choice to open up or create the `.bash_profile` file as shown:



```
Terminal Shell Edit View Window Help
saisriparasa -- ubuntu@domU-12-31-39-06-29-6E: ~ -- b
root@domU-12-...ww/saisriparasa  ubuntu@domU-1...39-06-29-6E: ~
Sais-MacBook-Pro:~ saisriparasa$ vi ~/.bash_profile
```

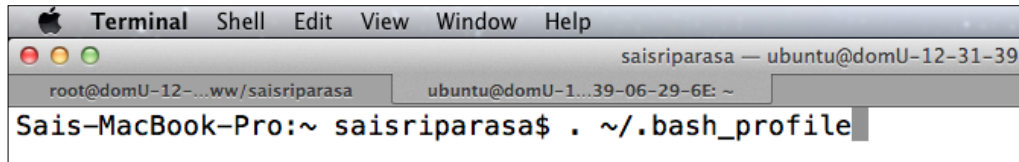
In the `.bash_profile` file, let's append the location of PHP 5.5 binaries to the existing `PATH` variable. The `PATH` variable is an environmental variable that keeps a track of directories for the shell to search for executable files in response to the commands issued by users. This is a quick way for the operating system to store the metadata of the locations of the executable files. The locations of the directories that contain the executable files are stored in a colon-separated string. When a new directory location is added to the `PATH` variable, it is only available within the scope of the script. We will use the `export` keyword to make it available outside the scope of that script as shown in the following screenshot:



```
Terminal Shell Edit View Window Help
saisriparasa --
root@domU-12-...ww/saisriparasa  ubuntu@domU-1...39-06-29-6E: ~
export PATH=/usr/local/php5/bin:$PATH
```

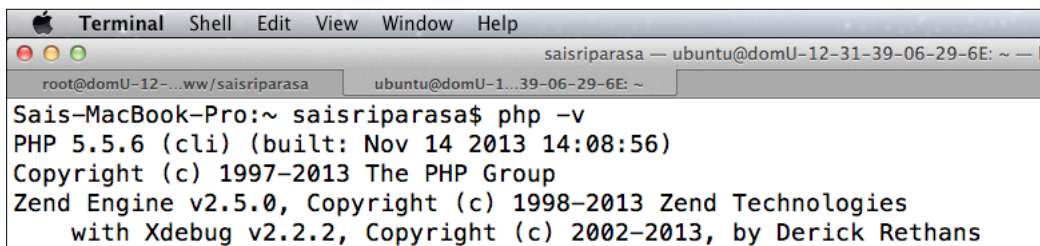
Add `export PATH=/usr/local/php5/bin:$PATH` to your `.bash_profile` file as shown in the previous screenshot, and save it. Now that the `PATH` variable has been modified, we can quickly check if our change was successful by echoing the `PATH` variable in the shell. The result would be negative, the reason being that our configurations in the `.bash_profile` file are loaded only when a new shell is loaded. Unless we reload the configurations, the changes made to the `PATH` variable will not be available in the context of the current shell.

There are two ways of reloading the `.bash_profile` file: the first is to use the `source` command followed by `.bash_profile` and the other method is just to use the `.` character to reload the configurations as shown in the following screenshot:




```
Terminal Shell Edit View Window Help
saisriparasa — ubuntu@domU-12-31-39
root@domU-12-...ww/saisriparasa  ubuntu@domU-1...39-06-29-6E: ~
Sais-MacBook-Pro:~ saisriparasa$ . ~/.bash_profile
```

Once the `.bash_profile` file is reloaded, the new `PATH` variable will be available and then we will be able to verify the version of PHP that we have installed earlier in this chapter. We will be using the `-v` option that is provided by the PHP executable to print the current version of PHP.

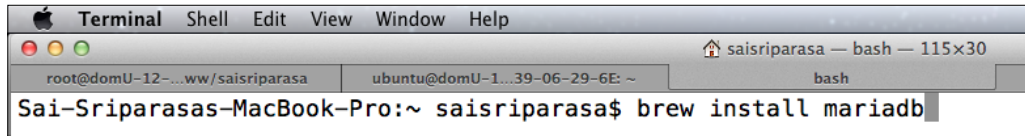


```
Terminal Shell Edit View Window Help
saisriparasa — ubuntu@domU-12-31-39-06-29-6E: ~ -- b
root@domU-12-...ww/saisriparasa  ubuntu@domU-1...39-06-29-6E: ~
Sais-MacBook-Pro:~ saisriparasa$ php -v
PHP 5.5.6 (cli) (built: Nov 14 2013 14:08:56)
Copyright (c) 1997-2013 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2013 Zend Technologies
with Xdebug v2.2.2, Copyright (c) 2002-2013, by Derick Rethans
```

The previous screenshot reflects a successful installation of PHP 5.5. Now the next step is to install MariaDB on Mac OS X. For installation of MariaDB on Mac OS X, we will be using Homebrew, which is popularly described as the missing package manager for Mac OS X.

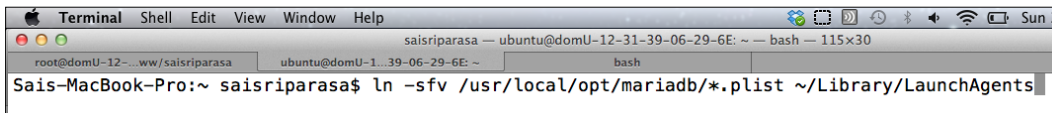
[ If Homebrew is not installed, run the following installation command in the terminal app: `ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/homebrew/go/install)"`]

Once Homebrew is installed, run `brew update` to get all the required updates. Once the update is successful, we are ready to install MariaDB. Installation of MariaDB is as simple as running a single line of instruction; we will be using the `install` command for Homebrew as shown in the following screenshot:



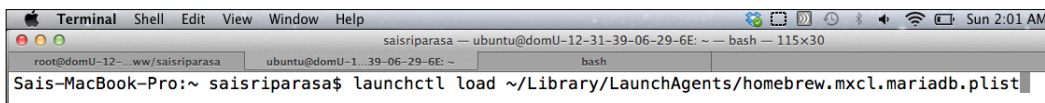
```
Terminal Shell Edit View Window Help
saisriparasa — bash — 115x30
root@domU-12-...ww/saisriparasa  ubuntu@domU-1...39-06-29-6E: ~  bash
Sai-Sriparasas-MacBook-Pro:~ saisriparasa$ brew install mariadb
```

On successful installation of MariaDB, we will have to make the operating system aware of the daemon that is required to run MariaDB. Mac OS X uses **property list (plist)** files to keep a track of the required properties for running an application. MariaDB being `launchd` compliant comes with the required plist files. The `launchd` command manages the processes for the Mac operating system and for individual users. The plist files that will be used for launching or loading the MariaDB daemon are located in the `/usr/local/opt/mariadb` folder. A reference of these files has to be created in the `~/Library/LaunchAgents` folder for MariaDB to be launched as a daemon as shown in the following screenshot:



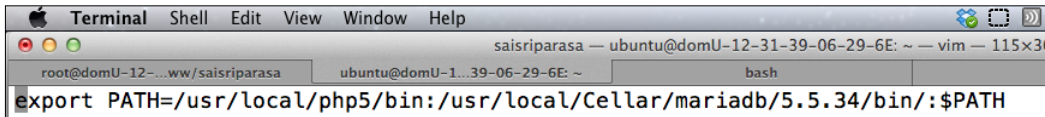
```
Terminal Shell Edit View Window Help
saisriparasa — ubuntu@domU-12-31-39-06-29-6E: ~ — bash — 115x30
root@domU-12-...ww/saisriparasa  ubuntu@domU-1...39-06-29-6E: ~  bash
Sais-MacBook-Pro:~ saisriparasa$ ln -sfv /usr/local/opt/mariadb/*.plist ~/Library/LaunchAgents
```

Now that these files are available, the MariaDB daemon can be invoked using the `launchctl` command. The `launchctl` interfaces with `launchd` to manage the required processes. We will be using the `load` command for `launchctl` and this will load the specified configurations that are needed to start the MariaDB daemon shown as follows:



```
Terminal Shell Edit View Window Help
saisriparasa — ubuntu@domU-12-31-39-06-29-6E: ~ — bash — 115x30
root@domU-12-...ww/saisriparasa  ubuntu@domU-1...39-06-29-6E: ~  bash
Sais-MacBook-Pro:~ saisriparasa$ launchctl load ~/Library/LaunchAgents/homebrew.mxcl.mariadb.plist
```

Once the configurations are successfully loaded, we will have to modify the `PATH` variable to recognize the `mysql` command. Though we will use the `mysql` command, we will be setting the `PATH` variable to look for MariaDB's executable files. Open the `.bash_profile` file in your favorite text editor and add the location of MariaDB's executable files shown as follows:



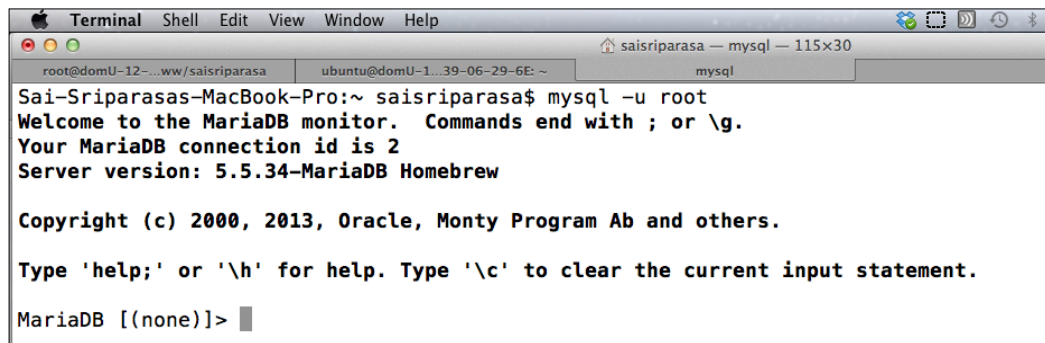
```
Terminal Shell Edit View Window Help
saisriparasa — ubuntu@domU-12-31-39-06-29-6E: ~ — vim — 115x30
root@domU-12-...ww/saisriparasa  ubuntu@domU-1...39-06-29-6E: ~  bash
export PATH=/usr/local/php5/bin:/usr/local/Cellar/mariadb/5.5.34/bin/:$PATH
```

Save the `.bash_profile` file and reload that file by either using the `source` command or by using the `.` operator. Now we have all the required configurations for running the MariaDB daemon. We will use the `start` command to start the MariaDB daemon shown as follows:



```
Terminal Shell Edit View Window Help
saisriparasa — bash — 115x30
root@domU-12-...ww/saisriparasa  ubuntu@domU-1...39-06-29-6E: ~  bash
Sai-Sriparasas-MacBook-Pro:~ saisriparasa$ mysql.server start
Starting MySQL
SUCCESS!
```

Our MariaDB installation and configuration are a success, and the MariaDB database server is now active and running. Let us open a client connection for our MariaDB database server. We will be using the `-u` option for the username and password in `root` as the username for the database server. This is shown in the following screenshot:



```
Terminal Shell Edit View Window Help
saisriparasa — mysql — 115x30
root@domU-12-...ww/saisriparasa  ubuntu@domU-1...39-06-29-6E: ~  mysql
Sai-Sriparasas-MacBook-Pro:~ saisriparasa$ mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.5.34-MariaDB Homebrew

Copyright (c) 2000, 2013, Oracle, Monty Program Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> █
```

The default `root` login for MariaDB doesn't need a password, but it is always a good practice to add the password. Let us secure the password for the `root` user, MariaDB stores the metadata information in the MySQL database as shown:

```
MariaDB [(none)]> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mysql]> █
```

Once we are in the MySQL database, we will use the `UPDATE SQL` statement, discussed in the next chapter, to change the password for the `root` user. The `UPDATE SQL` statement, as the name suggests, is used to modify data in a table, as shown in the following screenshot. We will be going over various `SQL` statements in the next chapter.

```
Database changed
MariaDB [mysql]> update user set password=
-> PASSWORD("newpassword")
-> where User = "root";
Query OK, 4 rows affected (0.10 sec)
Rows matched: 4 Changed: 4 Warnings: 0
```



Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.packtpub.com>. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

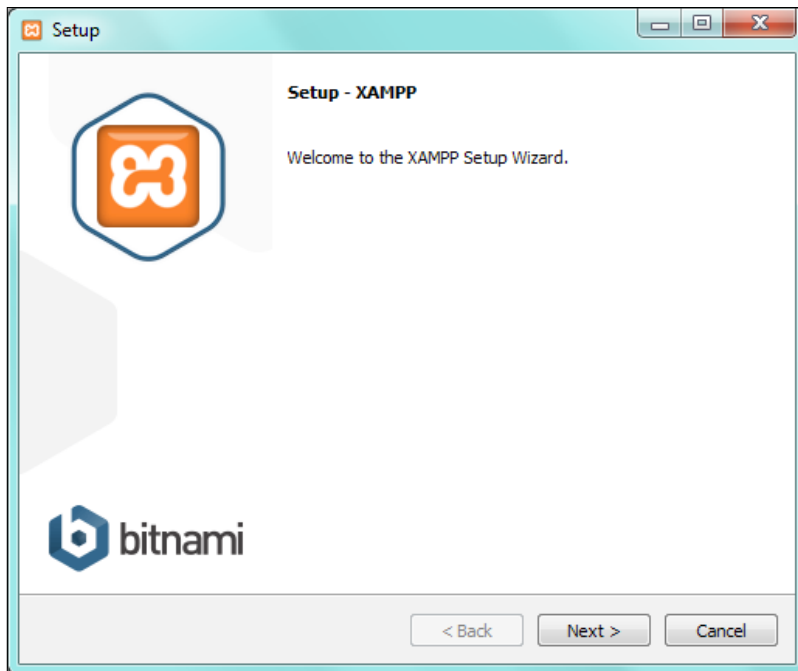
On successful execution of the query, as shown in the previous screenshot, we will need to reload the privileges and purge any of the privileges' metadata that was cached.

```
MariaDB [mysql]> flush privileges;
Query OK, 0 rows affected (0.07 sec)
```

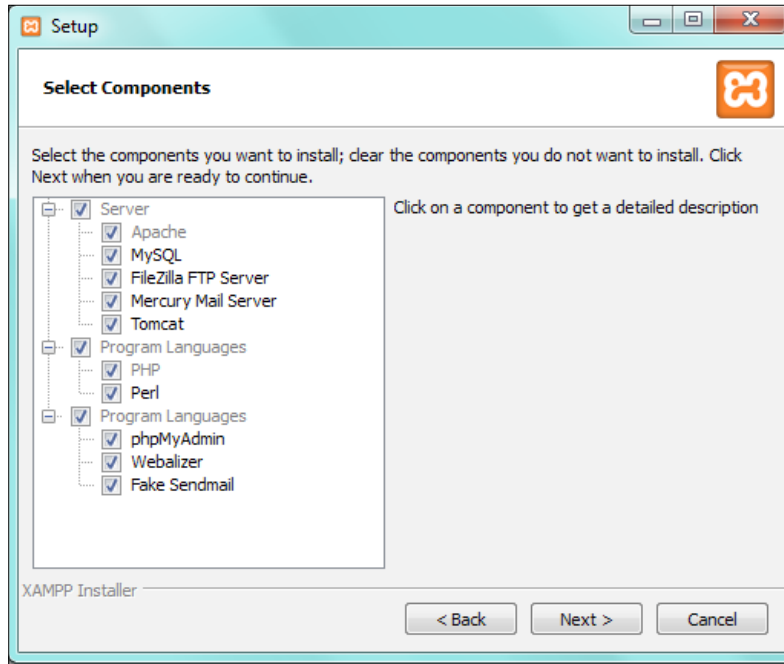
After the privileges have been flushed, we can use the `quit` or `exit` command to exit out of the MariaDB shell and again log in to the MariaDB server by using the `mysql` command with the `-u` option that mentions the user as `root` and use the `-p` option. Upon hitting the *Return* key, you are prompted for the password. Enter the new password that has been set to log in to the MariaDB server.

Installing AMP on Windows

Installation of AMP on Windows will be relatively simple when compared to the AMP stack's installation on Mac OS X. There are a few different web server solution stacks that are available from which WAMP and XAMPP are popular. For our installation process, we will be downloading two packages: the first package will be XAMPP and the second one will be the MariaDB database suite. XAMPP is a free and open source stack package that delivers using the Apache web server, MySQL database server, and interpreter libraries for PHP and Perl scripts. We will be using the Apache web server and the interpreter libraries for PHP and as we are using MariaDB as our database server, we will ignore the MySQL database server provided by the XAMPP stack. To download XAMPP for windows, visit <http://www.apachefriends.org/en/xampp-windows.html> and choose the appropriate package that contains PHP 5.5. At the time of writing this book, the version of XAMPP stack with PHP 5.5 is 1.8.3. On downloading, run the XAMPP executable as an administrator.

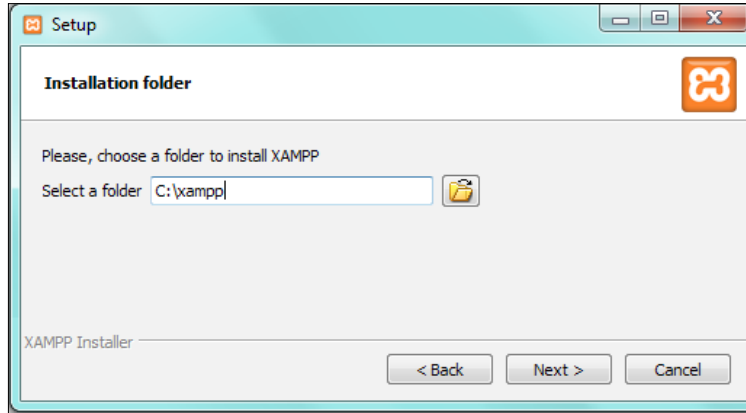


The first step of the installation for the XAMPP stack is as shown in the preceding screenshot. Click on **Next** to continue with the installation.

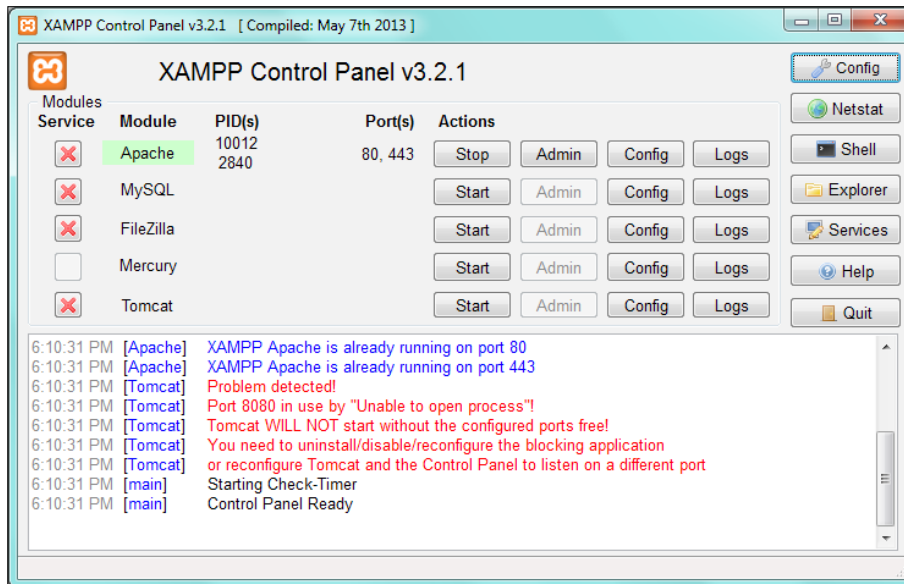


As we will be going with a default installation of the XAMPP stack, we will not be changing any default choices as shown in the preceding screenshot. In the future chapters where we discuss the optimization of our server stack to support production environment needs, we will be revisiting the installation to modify the components that are needed for our production environment.

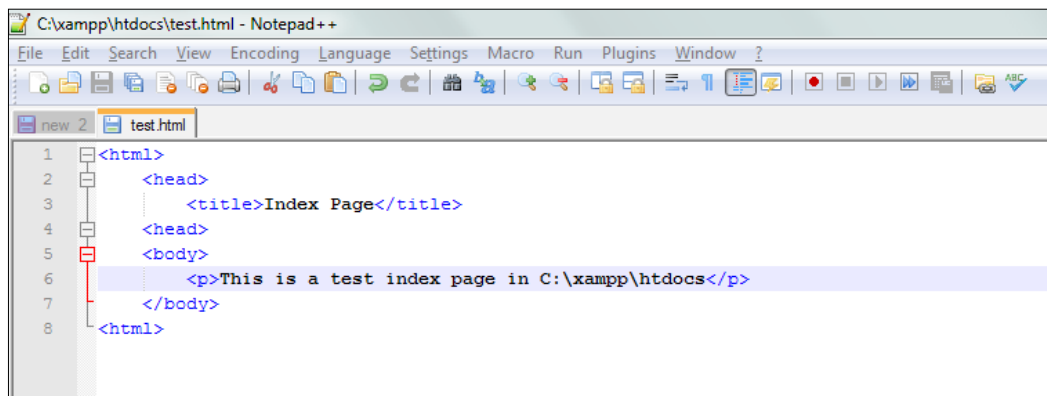
Click on **Next** to proceed with the installation.



In this step, we will be assigning a default location for the XAMPP package to be installed. As this is a default installation, let us continue with the installation by clicking on **Next** as shown in the preceding screenshot. This is the last step of installation and the XAMPP server stack is now installed.

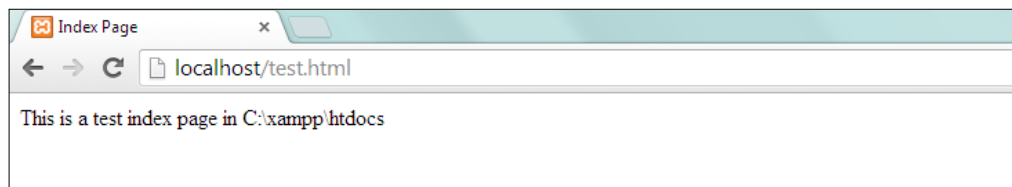


The XAMPP control panel, as shown in the following screenshot, is the control station for all the tools that XAMPP provides. As we are only utilizing the Apache web server and PHP's interpreter libraries, we will leave the rest of the tools turned off. The XAMPP server stack is installed in `C:/xampp`. To access the necessary configuration files, browse `C:\xampp\apache\conf` for Apache and `C:\xampp\php` for PHP. The default document root for XAMPP stack is located at `C:\xampp\htdocs`. Let us create an HTML test page to verify that the Apache web server is working as expected.



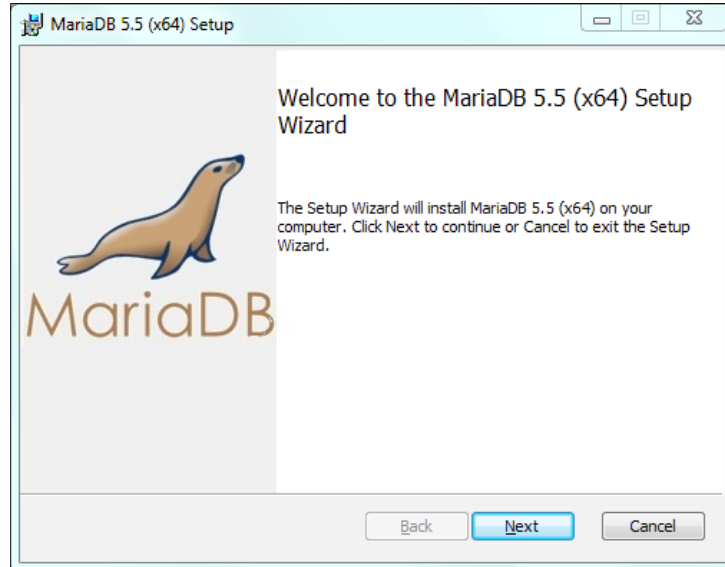
```
C:\xampp\htdocs\test.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
new 2 test.html
1 <html>
2   <head>
3     <title>Index Page</title>
4   </head>
5   <body>
6     <p>This is a test index page in C:\xampp\htdocs</p>
7   </body>
8 </html>
```

This is a simple HTML test page that we will be creating in the `C:\xampp\htdocs` folder; text of our choice can be used to create this document. I have used Notepad++ for building this page and would recommend using this editor. Now that we have the HTML page ready, let us verify if Apache is able to serve this page.

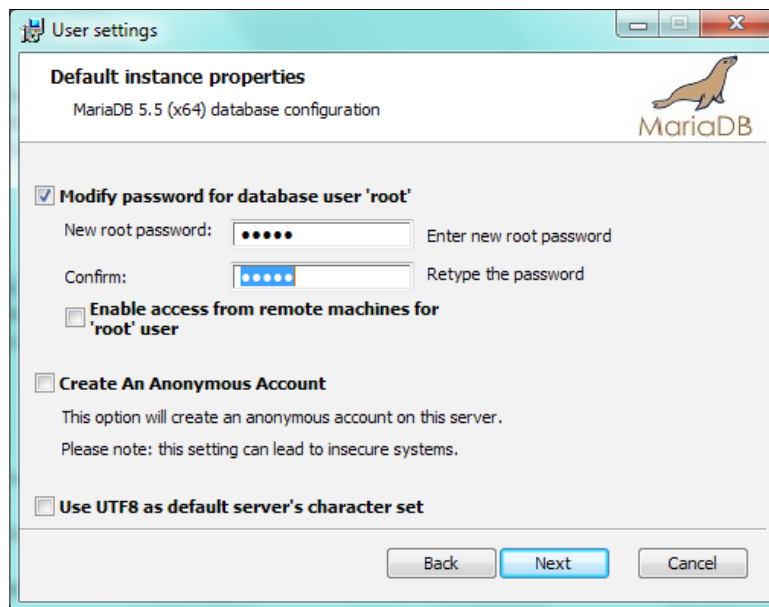


As seen from the preceding screenshot, the Apache HTTP Server provided by the XAMPP stack has served our test HTML web page as expected. Now let us move forward with our installation of the MariaDB database server. For the installation of MariaDB on Windows, we will have to download the latest stable MSI package of the MariaDB server. The latest MSI package available for installation during this book being written is 5.5.34, and it is advised to visit the **Downloads** section on the MariaDB website at <https://downloads.mariadb.org/mariadb> to download the latest version. Once the download has been completed, right-click on the installer and run it as an administrator.

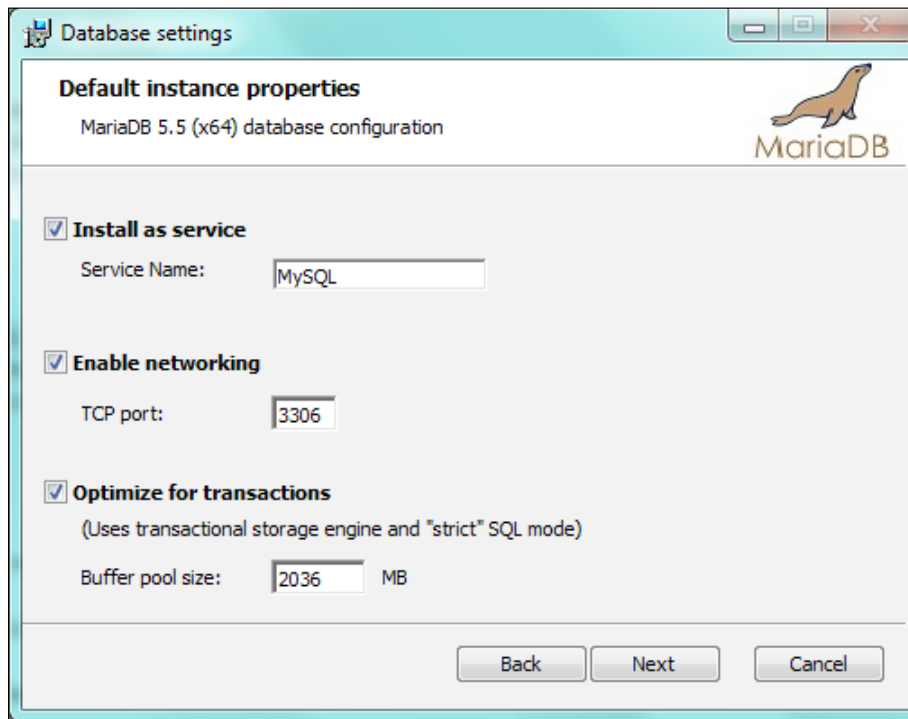
Next, refer to the following screenshot:



This is the initial screen. Click on the **Next** button and accept the terms in the license agreement after going through them. You will come to a screen as shown in the following screenshot:

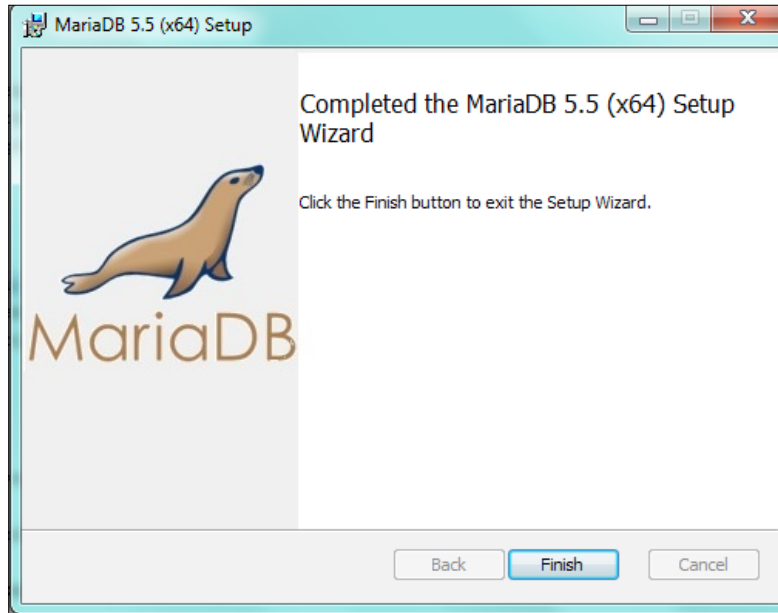


Choose a strong password for the `root` user and click on the **Next** button. We will be discussing the use of allowing users to access our MariaDB server from remote machines in the next chapters. We will thoroughly address the benefits, hazards, and precautions for allowing access from remote machines. The following screenshot shows the default properties for the database configuration:




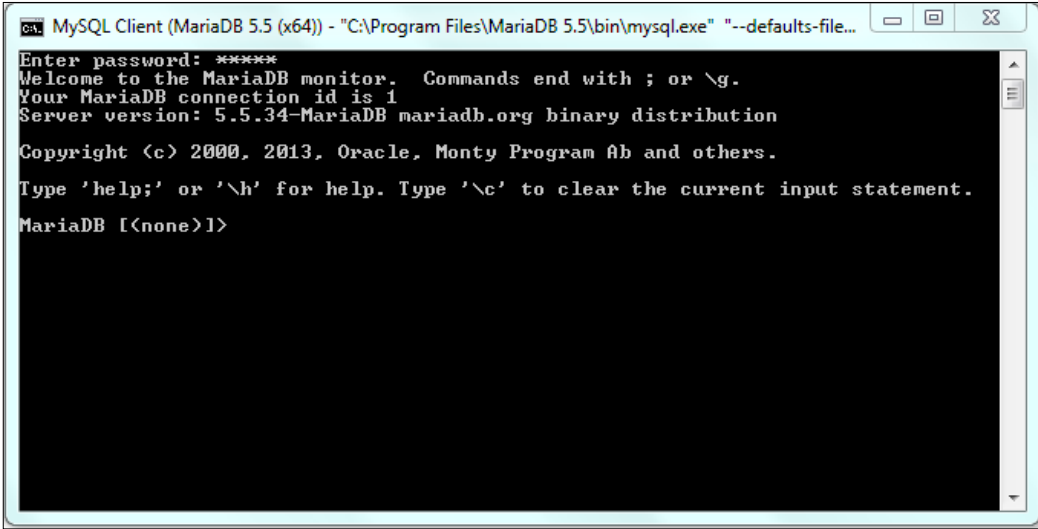
As this is a default installation, we will not be changing any of the existing values. Keep in mind that the MariaDB server will be running on port 3306, so the MySQL server on XAMPP should always be turned off. If not, both the daemons will compete to use the same port.

After our installation is completed, we will come to the screen shown in the following screenshot:



Our installation has completed and is successful. Now to connect to our MariaDB database server, click on **Start**, then click on **All Programs**, scroll down to **MariaDB (5.5)**, and click on **MySQL client (MariaDB 5.5)** to fire up a client connection to our MariaDB database server. MariaDB database server would by default assume that a `root` user has requested the client connection access, and would prompt the user to enter the `root` user's password. Upon successful login, the user will receive a success message that would let them run queries against the MariaDB database server as shown in the following screenshot:

[ MySQL Workbench is a popular GUI tool that can be used to connect to MariaDB and execute queries.]



```
MySQL Client (MariaDB 5.5 (x64)) - "C:\Program Files\MariaDB 5.5\bin\mysql.exe" "--defaults-file...
Enter password: *****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 1
Server version: 5.5.34-MariaDB mariadb.org binary distribution
Copyright (c) 2000, 2013, Oracle, Monty Program Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]>
```

Installing AMP on Linux (Ubuntu)

There are multiple distributions of Linux, of which Ubuntu is one of the most popular distributions among open source programmers. We will be using Ubuntu 13.10 (Saucy Salamander), the latest Ubuntu distribution that is available. Saucy Salamander arrives with preinstalled Apache web server. The default document root for Apache on Ubuntu is `/var/www`. Now let us create an HTML test page to verify that Apache web server is working as expected, shown as follows:



```
root@adminuser-VirtualBox: /var/www
<html>
  <head>
    <title>Index Page</title>
  </head>
  <body>
    <p>This is a test index page in /var/www</p>
  </body>
</html>
```

Now that we have verified that Apache is up-and-running, we will need to install PHP 5.5 and MariaDB. To install PHP 5.5, we will be using Ondrej Sury's repository, which is available for open source development. To execute these commands, open up a terminal window and run these commands step-by-step. We would need the `sudo` permissions for executing these commands successfully shown as follows:

```
sudo apt-get install -y python-software-properties
sudo add-apt-repository ppa:ondrej/php5
sudo apt-get update

sudo apt-get install php5
```

Upon successful installation of PHP 5.5, we will move to the installation of the MariaDB database server. For the installation of the MariaDB database server, we will be using Open Source Lab's mirror that hosts the required binaries. Use the terminal window and execute these commands step-by-step as shown in the following screenshot:

```
sudo apt-get install -y software-properties-common
sudo apt-key adv --recv-keys
--keyserver hkp://keyserver.ubuntu.com:80 0xc9cb082a1bb943db
sudo add-apt-repository
'deb http://ftp.osuosl.org/pub/mariadb/repo/5.5/ubuntu saucy main'

sudo apt-get update
sudo apt-get install -y mariadb-server
```

Upon successful installation of the MariaDB database server, use the existing terminal window to request a client connection to the MariaDB server. Use MySQL's `-u` operator to specify the user as `root` and use the `-p` operator for providing a password on prompt, and press *Enter*. Then the user will be prompted to enter the root user's password as shown:

```
root@adminuser-VirtualBox:~# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 35
Server version: 5.5.34-MariaDB-1-saucy-log mariadb.org binary distribution

Copyright (c) 2000, 2013, Oracle, Monty Program Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> █
```


Summary

This chapter deals with a basic introduction to web development and how Apache, MariaDB, and PHP (AMP) stack can be used for web development. We have thoroughly gone through the process of installing and configuring the AMP stack on the Mac OS X, Windows, and Linux (Ubuntu) operating systems.

In the next chapter, we will begin an introduction to the relational Database Management Systems and Structured Query Language; we will also understand how MariaDB can be used for data storage.

