Apache Web Server Administration for Linux

with examples and
hands-on exercises

WEBUCATOR

Version: 2.1.1

The Author

Stephen Withrow

Stephen has over 30 years of experience in training, development, and consulting in a variety of technology areas including Python, Java, C, C++, XML, JavaScript, Tomcat, JBoss, Oracle, and DB2. His background includes design and implementation of business solutions on client/server, Web, and enterprise platforms. Stephen has a degree in Computer Science and Physics from Florida State University.

Class Files

Download the class files used in this manual at https://static.webucator.com/media/public/materials/classfiles/AWS201-2.1.1.zip.

Errata

Corrections to errors in the manual can be found at https://www.webucator.com/books/errata/.

Table of Contents

LESSON 1
Apache Web Server

Topics Covered

☑ The Apache Web Server.

☑ The functionality of the Web Server.

☑ Downloading Apache.

☑ Installing the Web Server on Linux.

☑ Testing the installation.

Introduction

The Apache Web Server is the one of the oldest and most popular web servers. According to various estimates, more than one-half of all websites on the World Wide Web are hosted by the Apache Web Server.

Apache = Apache Web Server
The Apache Web Server is often referred to as simply *Apache*.

The popularity of Apache is due to the web server's functionality and speed. The web server offers a broad range of capability including virtual hosting, MIME type specification, directory mapping, securing web resources, and logging configuration. The speed of Apache is due in part to the fact that it is written in C and can be assembled from modules.

1.1. The Apache Web Server

Apache was initially developed by Rob McCool at the NCSA (National Center for Supercomputing Applications) in early 1995. By the end of the year, a group of engineers led by Brian Behlendorf and Cliff Skolnick created the first "official" release of Apache. The group had assembled the web server

from “patches” and the server became known as “a patchy” web server. Eventually, the word “Apache” was used as the server’s official name.

By 1999, the **Apache Software Foundation** was formed and the Apache Web Server became generally available as the foundation’s first open-source product.

1.2. Features

Apache offers a broad range of features including virtual hosting, server-side language support, authentication and authorization, SSL support, and configurable error messages.

The functional capability is delivered through **modules** that can be either compiled into the server software or configured as dynamic objects. This modular design allows administrators to adapt the web server to the needs of an organization.

1.3. Download

This part of the lesson will guide you in downloading the Apache Web Server for Linux.

❖ 1.3.1. Linux

Distributions of Linux such as Redhat’s Fedora are shipped with Apache. In addition, most Linux distribution vendors provide a pre-built RPM for Apache.

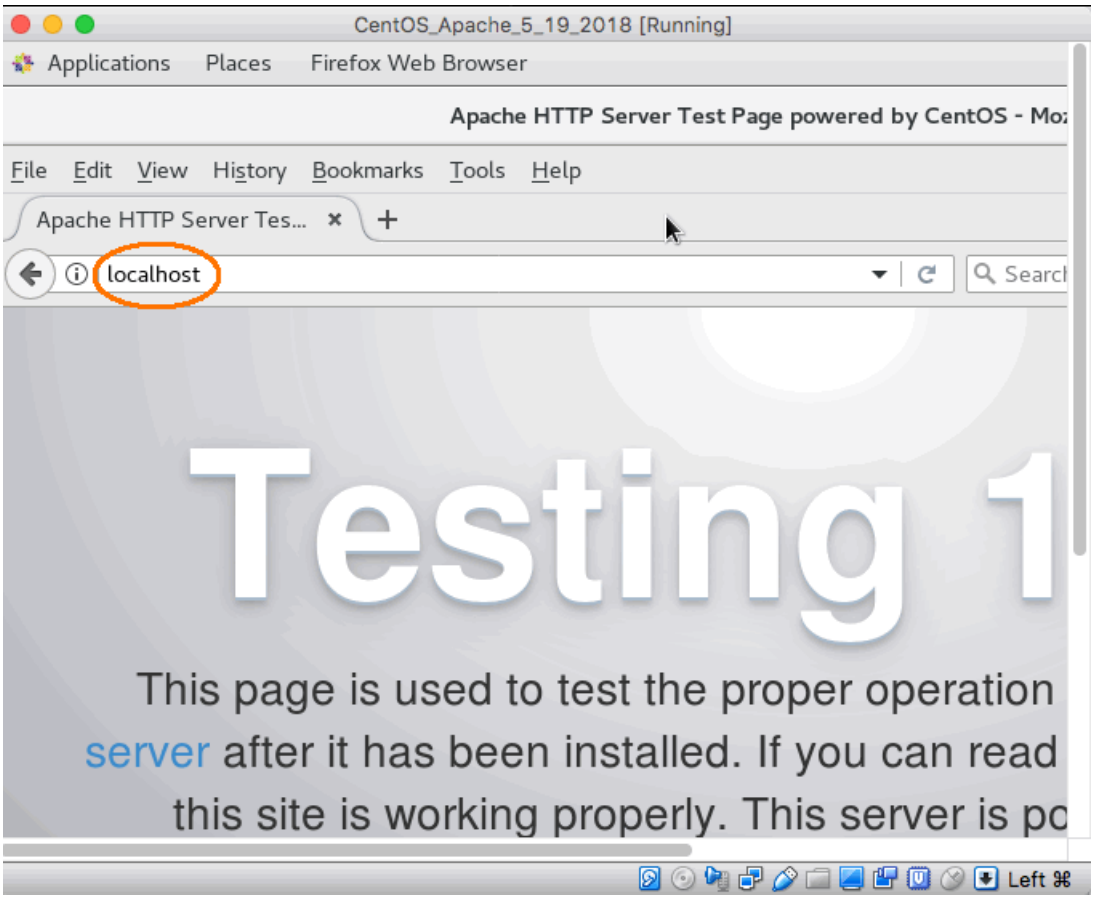
For CentOS, you can use `yum` to install Apache as shown below:

```
sudo yum install httpd
```

Once Apache is installed, you can start the server as shown below:

```
httpd -k start
```

Point your browser to `http://localhost`. You should see the welcome page:



1.4. Multi-Processing Modules

A Multi-Processing Module (MPM) enables Apache to work efficiently with the operating system. The MPM handles binding to network ports, handling requests and dispatching child processes to handle the requests.

Apache is shipped with an MPM compiled into the server. The following table indicates the default MPM for each OS:

MPM	Operating System
mpm_netware	Netware
mpmt_os2	OS/2
prefork, worker or event	UNIX/Linux
mpm_winnt	Windows

The `mpm_winnt` MPM is compiled from `mpm_winnt.c`. This module uses a single control process that creates a child process to create threads to handle requests.

The `prefork` MPM is compiled from `prefork.c`. This module uses a single control process that creates child processes that listen for requests. The requests are serviced by a child process. Several idle, or spare, processes are maintained by Apache so that a new process does not need to be forked to service a client request. The maximum number of child processes ("workers") can be set using the `MaxRequestWorkers` directive.

The `worker` MPM is compiled from `worker.c`. This module uses a multi-process multi-threaded server to create threads to serve requests. A parent process launches child processes and each child process creates a fixed number of threads to handle requests in addition to a listener thread to listen for requests. The maximum number of threads ("workers") that can be created is set using the `MaxRequestWorkers` directive. The maximum number of threads a child process can create is set using the `ThreadsPerChild` directive.

The settings for each MPM are located in `APACHE_HOME/conf/extra/httpd-mpm.conf`.

1.5. Building Apache from Source

For Linux and UNIX environments, Apache can be compiled from the C source files. This presents the opportunity to statically compile modules into the Apache binary.

The procedure for compiling Apache consists of the following steps:

1. Download the Apache server.
2. Extract the source from the Apache server tarball.
3. Configure the Apache source tree.

4. Perform the compile.
5. Install the package.
6. Customize the configuration.
7. Test by starting the Apache server.

For detail on these steps go to `http://httpd.apache.org/docs/current/install.html`.

Exercise 1: Testing Apache

15 to 25 minutes

In this exercise, you will test the Apache Web Server on Linux.

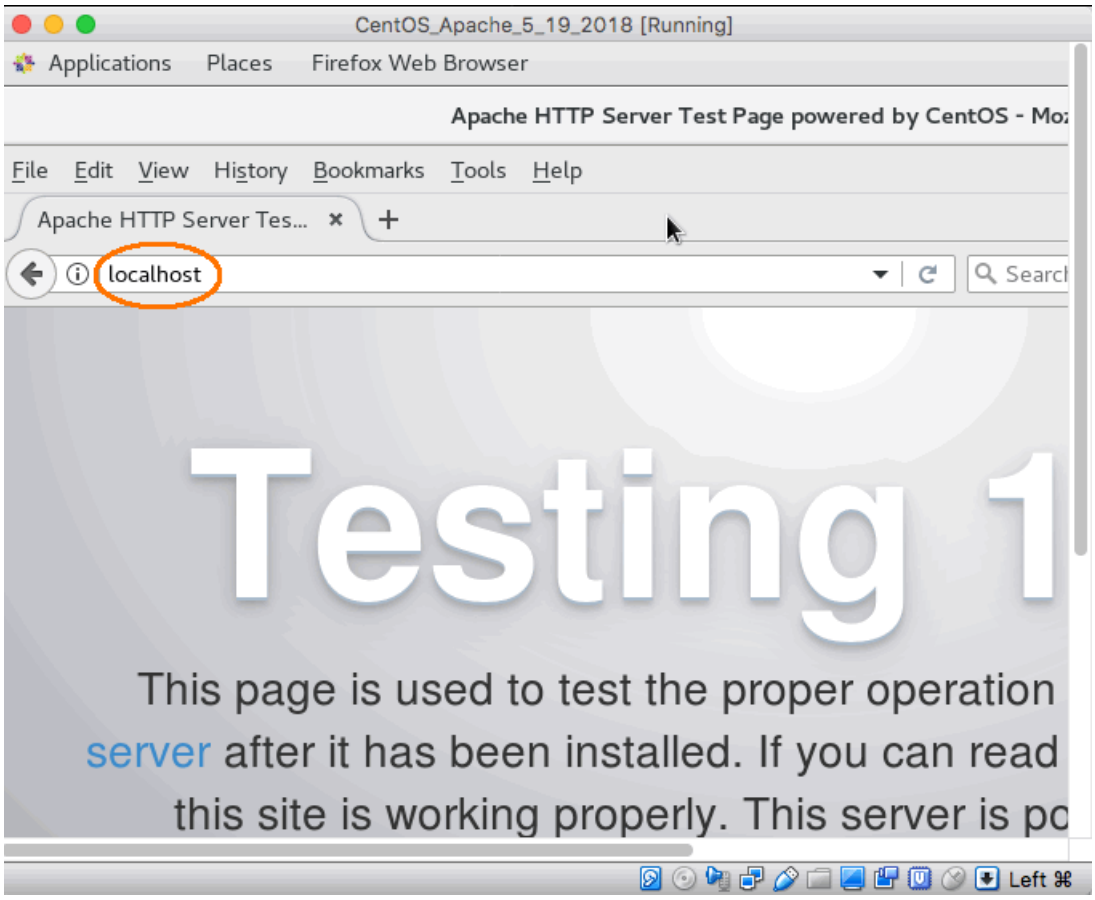
1. Open `/etc/httpd/conf/httpd.conf` for edit in gedit or similar text editor. Locate the line that starts with `ServerRoot`. After this line add the following:

   ```
   ServerName localhost
   ```


2. Save the file.
3. Start the web server with the following command in a terminal window:

   ```
   sudo httpd -k start
   ```

4. Point your browser to `http://localhost`. You should see the welcome page:



5. You can stop the web server with this command:

```
sudo httpd -k stop
```

6. In this course you will be required to create folders and edit files. These tasks can be accomplished in the File Manager (denoted by "Files" on the menu in CentOS) if you have root privileges.
7. To start the File Manager as root, open a new terminal window by clicking "Terminal" on the menu In the terminal window, enter `sudo nautilus`. This will open the File Manager in super user mode, allowing you to create, rename and edit folders and files.
8. You can open another terminal window so you will be able to manage command line tasks as indicated throughout the training.

Conclusion

In this lesson, you have learned:

- About the Apache Web Server.
- About the Functional Purpose of the Web Server.
- How to download Apache.
- How to install the Web Server on Linux.
- About testing the installation.

LESSON 2
Directory Structure

Topics Covered

☑ The directories in the Apache Web Server.

☑ The role of each directory.

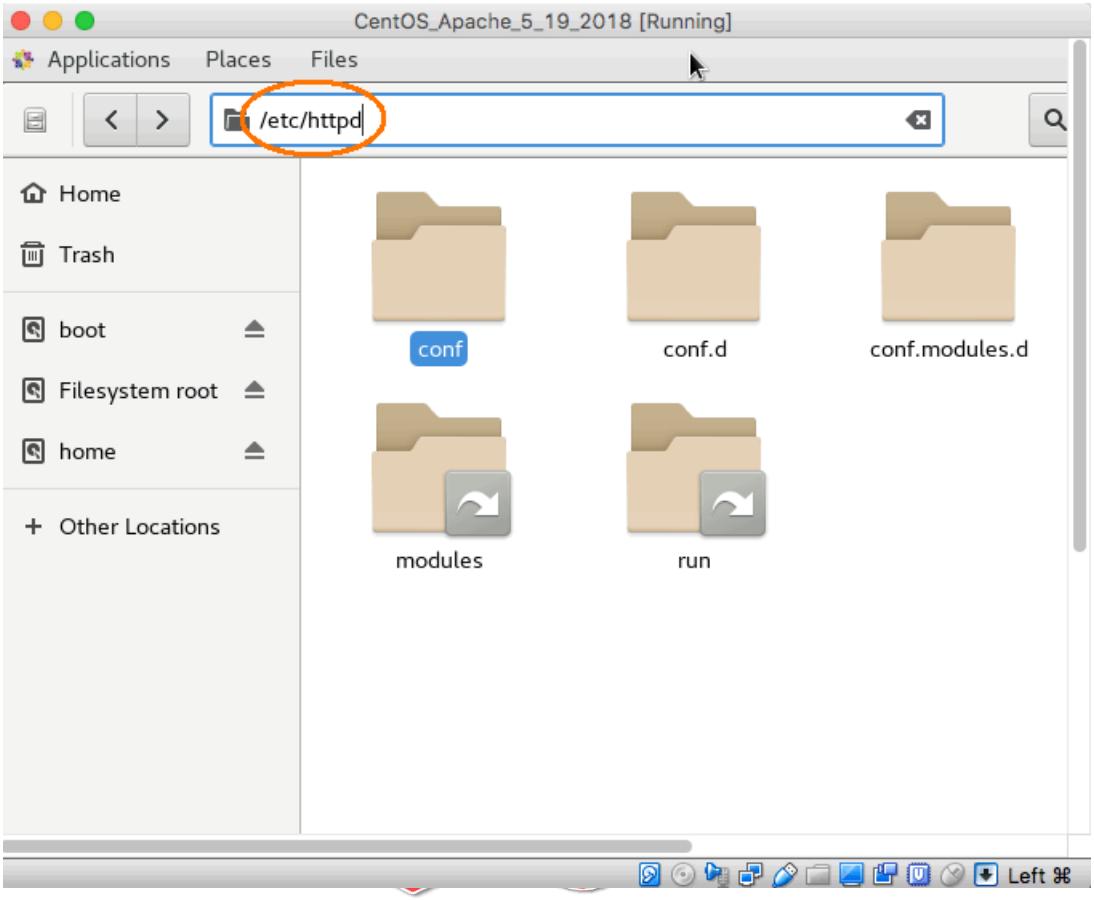
☑ `DocumentRoot`.

Introduction

The Apache Web Server as represented on the file system consists of several directories. Each directory has a distinct purpose. In this lesson, you will learn the purpose and the importance of each directory.

2.1. Apache Directories

After you install Apache, the directory structure for the configuration is show below:



Note that the directory is `/etc/httpd`. The directory structure may be different on the various Linux distributions. In this course, we focus on configuration of the Apache Web Server in the CentOS environment.

2.2. The Role of Each Directory

❖ 2.2.1. conf

The `/etc/httpd/conf` folder contains `httpd.conf`. This is the primary configuration file. We will study this file extensively.

❖ 2.2.2. conf.d

This directory contains configurations for the welcome page and PHP, among others.

❖ 2.2.3. conf.modules.d

This folder contains the base module configuration that includes the majority of modules you are likely to need to realize full functionality out of Apache.

❖ 2.2.4. logs

This is a symbolic link to the log files directory, `/var/log/httpd`, that contains the error and access logs.

❖ 2.2.5. modules

This is another symbolic link that points to the location of the Apache modules.

❖ 2.2.6. run

Additional files for the operation of the Apache server.

2.3. Document Root

The **document root** is the root folder for the web pages served by Apache. The document root is set using `DocumentRoot` directive.

The following is the document root setting in `/etc/httpd/conf/httpd.conf`:

```
DocumentRoot /var/www/html
```

Exercise 2: Changing the Directory for the Log Files

20 to 30 minutes

In this exercise, you will modify the Apache configuration to write the error log to a specific folder.

1. Stop the Apache server.
2. Delete the error log in `/etc/httpd/logs`. The file name is `error.log`.
3. Create a new folder called `/errorLog`. Change the permissions on the directory to permits all users to read and create files.
4. Open `/etc/httpd/conf/httpd.conf` for edit.
5. Locate the following line:

```
ErrorLog "logs/error.log"
```

6. Insert a "#" (pound sign) as the first character on this line:

```
#ErrorLog "logs/error.log"
```


7. Add this line:

```
ErrorLog /errorLog/error.log
```

8. Save your changes. Start Apache.
9. Display the welcome web page to ensure Apache is operational.
10. Navigate to the `/errorLog` folder and open `error.log`. You should see several log entries that were generated at server startup.
11. Check the `/etc/httpd/logs` folder. Ensure that the original error log file was not generated.

Conclusion

In this lesson, you have learned:

- About the directories in the Apache Web Server.

- About the purpose of each directory.
- How to use DocumentRoot.

LESSON 3
httpd.conf Configuration

Topics Covered

☑ The contents of the `httpd.conf` file.

☑ Comments, directives, and containers.

☑ The scope of directives.

☑ The order of evaluation for containers.

☑ Port assignments.

☑ Specifying the Document Root for web pages.

☑ Including files.

☑ Guidelines for modularization and organization of `httpd.conf`.

☑ Updating the configuration between releases.

Introduction

The `httpd.conf` file is the primary configuration file for the Apache Web Server. The file is located in the `/etc/httpd/conf` directory.

`httpd.conf` is a text file and consists of directives, containers, and comments. The containers are coded in an XML syntax but the configuration file itself is not XML.

Modifications to the file are not picked up automatically by Apache. The server must be restarted in order to load the directives and containers coded in the configuration file.

3.1. Contents of httpd.conf

This file contains three types of statements:

1. **Directives** are single-line statements that associate a variable (e.g., `ServerRoot` with a value).
2. **Containers** enclose one or more directives and are presented in an XML syntax.
3. **Comments** are sequences of characters preceded by the # (pound sign) and are ignored by Apache.

3.2. Coding Directives, Containers, and Comments

❖ 3.2.1. Directives

Directives define values for Apache server variables. For example, the following directive defines the value for the `ServerRoot` variable:

```
ServerRoot /etc/httpd
```

The following line of code assigns the port value for incoming HTTP requests:


```
Listen 80
```

Case Sensitivity

Directive names are not case sensitive but the assigned values might be case sensitive (e.g., file system references).

The `LoadModule` directive assigns a variable to a module name. The load module directives are stored in a separate file named `/etc/httpd/conf.modules.d/00-base.conf` that is included within `httpd.conf`. For example, the following directive assigns the `alias_module` variable to the module named `mod_alias.so`:

```
LoadModule alias_module modules/mod_alias.so
```

This directive activates the `mod_alias` module. This module implements the `Alias` directive. Therefore, the `Alias` directive can now be used in the configuration file.

Note

Every directive is implemented by a module.

❖ 3.2.2. Containers

Containers enclose one or more directives. For example, the following Directory container encloses two directives:

```
<Directory />
  AllowOverride none
  Require all denied
</Directory>
```

The directory that is the target of the directives is `/`, which covers the entire file system.

❖ 3.2.3. Comments

A comment begins with a # (pound sign or hash character). The following is an example of a comment:

```
# A comment can be placed anywhere in apache2.conf
```

Otherwise "live" code can be commented out and reserved for future use:

```
#LoadModule mod_jk modules/mod_jk.so
```

3.3. Scope of Directives

A directive in `httpd.conf` applies to the entire server. This is referred to as **global scope**. To narrow the scope of a directive, the directive can be placed in a container.

For example, the `DirectoryIndex` directive specifies one or more files that can be served to a client if a web page is not present in the request. If `DirectoryIndex` is placed in `httpd.conf` outside of any containers, then the directory index file specified will be applied to any directory that is accessible

to clients. If the directory index directive is placed within a Directory container, then it will override the global specification for that directory.

A directive placed within a Directory container is said to have **container context**. Directives defined in Location and File containers also have container context.

Virtual host context applies to directives defined in a Virtual Host container. **.htaccess context** applies to directives defined in a `.htaccess` file.

Directives stored in a Directory container are applicable only to that directory and its subdirectories. They override identical directives coded elsewhere in the configuration file.

Directives stored in a Location container are applicable to the URL (Uniform Resource Locator) identified by the Location directive.

Virtual Host containers specify directives that are applicable for the virtual host. Unlike the other containers virtual hosts are associated with distinct IP address, host names, and port. In addition, a separate document root can be specified for a virtual host.

3.4. Order of Evaluation of Containers

Apache evaluates containers in a particular order for each request with the exception of the Virtual Host container. This container is processed during startup.

We will first take a look at the steps that are performed by Apache when the server is started:

1. Apache reads the configuration file (`httpd.conf`).
2. Included files are merged into the configuration file.
3. Apache stores the values of main server directives such as `Listen` (HTTP listener port number) and `ServerName`.
4. Virtual host containers are processed.

All other containers are read by Apache for each incoming HTTP request. For each request:

1. Any relevant Directory container is processed first and directives from `.htaccess` (if present on the directory and if overrides are allowed) will be merged with container directives.
2. Apache will apply any relevant Files container directives (defined by the `Files` directive).

3. Apache applies any relevant Location container directives last, and these directives will override any conflicting directives located in other containers.

3.5. Making Port Assignments

The port assignments for the HTTP listener and SSL (Secure Socket Layer) listener can be set in `httpd.conf`. The directive that is used to set the port is Listen. The following line appears in the configuration file:

```
Listen 80
```

HTTP requests directed to the Apache server can be specified in the browser's location field as follows:

```
http://localhost:80
```

The port number of 80 can be omitted because major browsers (e.g., Firefox, Internet Explorer) default to 80.

The SSL port will be covered in the lesson on security.

3.6. Specifying the Document Root

The document root is the root directory for the web pages that Apache will serve. Consider the following request that is typed into the browser's location field:

```
http://localhost/myWebPage.html
```

This request will be successful if a file named `myWebPage.html` is present under `/var/www/html`.

The document root is specified using the `DocumentRoot` directive and is set to `/var/www/html` as shown in the following line from `httpd.conf`:

```
DocumentRoot /var/www/html
```

You can modify the `DocumentRoot` directive to specify a different location for the web pages.

3.7. Including Files

You can include files in `httpd.conf`. Including files is advisable to keep the size of `httpd.conf` relatively small and to ease maintenance efforts. To include a file, you use the **Include** or **IncludeOptional** directive.

Several include directives are contained in `httpd.conf` as shipped from the factory. For example:

```
Include conf.modules.d/*.conf
```

The "Include" directive will copy the file(s) under `/etc/httpd/conf.modules.d` into the current file.

3.8. Modularization and Organization

Effective use of modularization and organization of the configuration file will help ensure correct results and efficient maintenance.

❖ 3.8.1. Guidelines for Modularization

1. Use the `Include` directive.
2. The include file should be focused on a particular task and should only contain directives necessary for that task.
3. Store included files in a convenient location (e.g., in a subfolder of `/etc/apache2`).

As pointed out earlier, the `httpd.conf` file contains several Include directives. The included files may consist of a dozen lines or more. Therefore the include operation reduces the size of `httpd.conf` significantly. Future modifications to the `httpd.conf` file are more straightforward simply because the file is relatively small (as measured by the number of lines).

Modifications to directives in the included files is also less time consuming because the files are generally small. In addition, the directives in an included file are focused on a particular task (e.g., setting up directives to allow access to the Apache manual).

❖ 3.8.2. Organization

Organization of the configuration file is also important. We'll consider the presentation order in the "factory" `httpd.conf` file:

1. The opening comments offering helpful information concerning the configuration file.
2. The main server configuration parameters, such as the server root directory, the document root, the server name, and HTTP listener port.
3. The "main server configuration" containing the ServerAdmin email address, ServerName, DocumentRoot, DirectoryIndex file, and additional directives.
4. Load modules are included from external load module entries.

If you need to add directives to `httpd.conf`, then you can follow the organization that is in place. For example, LoadModule entries should be placed in the appropriate include file or placed in a separate `.conf` that will be included into `httpd.conf`. Containers should be placed before the includes.

Comments are very useful to document the additional directives you place in the configuration file.

3.9. Verifying the Syntax of httpd.conf

Apache will not start if syntax errors exist in the configuration file. Before you change the file, you should make a backup of the file.

To check for syntax errors in `httpd.conf`, enter the following command in a command or terminal window:

```
httpd -t
```

3.10. Updating httpd.conf between Releases

When upgrading to a newer release of Apache, you will need to incorporate your customizations to the new `httpd.conf` file. The syntax of the directives will not change so you needn't be concerned about rewriting your directives or containers unless you want to take advantage of a new feature.

Carefully merge your modifications to the new configuration file and insert any includes that are required to copy additional directives and containers.

Perform a syntax check of the configuration file as described earlier.

Restart the server and test your changes.

3.11. Securing httpd.conf

`httpd.conf` should be secured using OS file permission commands.

For Linux, the `chmod` command can be used to secure the `conf` folder as shown below:

```
chmod -R 0770 /etc/httpd/conf
```

where `/etc/httpd/` is the Apache home. The command grants read, write, and execute privileges to the folder's owner and the owner's group. All others are not authorized to access the folder or its contents.

The Apache home directory can be assigned an owner using the `chown` command.

Exercise 3: Changing the Configuration

20 to 30 minutes

In this exercise, you will modify the Apache configuration to change the HTTP listener port.

1. Stop the Apache Web Server.
2. Open `/etc/httpd/conf/httpd.conf` for edit.
3. Locate the line that contains `Listen 80`.
4. Change the port number to 90:

```
Listen 90
```

5. Save your changes. Keep the file open in your text editor. You will restore the original port number after testing your change.
6. Start Apache.
7. Display the welcome web page to ensure Apache is operational. Be sure to specify a port value of "90": `http://localhost:90`
8. Return to the text editor. Change the port number back to "80". Save your changes.
9. Restart Apache. Verify the port number is 80:

```
http://localhost
```


Conclusion

In this lesson, you have learned:

- About the role of httpd.conf file.
- How to code comments, directives, and containers.
- About the scope of directives.
- About the order of evaluation for containers.
- How to make port assignments.
- How to specify the Document Root for web pages.
- About including files.

- Guidelines for modularization and organization of httpd.conf.
- About updating the configuration between releases.

LESSON 4
Load Modules

Topics Covered

☑ The purpose of modules.

☑ Dynamic shared objects.

☑ The core module.

☑ Static modules.

☑ Building modules from source and install from a package.

☑ The `LoadModule` directive.

☑ The relationship between directives and modules.

☑ MIME types and the role of `mod_mime`.

Introduction

The Apache Web Server is assembled from **modules**. A module is a compiled C program. A module provides functionality in the form of directives.

A module is either "static" or "dynamic." A static module is compiled into the `httpd` binary. You can list the modules that are compiled into the server by typing the following command in a command or terminal window:

```
httpd -l
```

A dynamic module is referred to as a **Dynamic Shared Object** (DSO). A DSO is not compiled into the `httpd` binary. The module `mod_so` is required for DSO support.

Note

`mod_so` and `core.c` must be compiled into `httpd`. All other modules can be provided dynamically.

4.1. Dynamic Shared Objects

A dynamic shared object, or DSO, must be declared to Apache using a LoadModule directive. A DSO is compiled with a name (e.g., `mod_authn_core`). This name will be associated with a corresponding module using the LoadModule directive.

For example, here is the declaration of `mod_authn_core.so`:

```
LoadModule authn_core_module modules/mod_authn_core.so
```

`AuthType` and `AuthUserFile` are examples of directives implemented by this module.

4.2. The Core Module

The core module of `httpd` is `core.c`. This module implements many fundamental directives including `Directory`, `ErrorLog`, `Files`, `Include`, and `Location`.

This module must be compiled into `httpd` and cannot be implemented as a DSO.

4.3. Static Modules

Any module can be compiled into `httpd`. Therefore, any module potentially can be a static module. However, this practice is discouraged because the module is now a permanent part of the binary file. If the module's directives are no longer needed in the future, then removing the static module will not be possible without recompiling the entire server code.

A DSO can be readily removed by simply commenting out (or deleting) the `LoadModule` directive that declares the DSO.

4.4. Building Modules from Source and Installing from a Package

A DSO can be created from the C source using the **Apache Extension Tool** (apxs). The tool requires `mod_so`.

Additional information can be obtained from the Apache documentation at `http://httpd.apache.org/docs/2.2/programs/apxs.html`

The `mod_jk` module is a typical example of a module that is prepared using apxs. This module can be used to enable Apache as a front-end proxy to other servers such as **Tomcat**.

We will take a look at using Apache as a front-end proxy to Tomcat later in this course.

4.5. Relationship between Modules and Directives

A module implements one or more directives. A module must be loaded into memory before any of its directives can be used.

❖ 4.5.1. LoadModule Directive

The `LoadModule` directive is used to load a module as a DSO. This directive is implemented by `mod.so`.

Like all directives, `LoadModule` is may be coded in `httpd.conf`. For CentOS, the `LoadModule` directives are present in the file in the included directory `/etc/httpd/conf/conf.modules.d/00-base.conf`.

The general form of this directive is:

```
LoadModule module location
```

`module` is the name of the variable stored in the file. The file is referenced in `location`.

The example presented earlier is repeated below:

```
LoadModule authn_core_module modules/mod_authn_core.so
```

The value of the module variable is `authn_core_module` and the DSO file is located in the `modules` folder under the Apache home directory.

❖ 4.5.2. Using a Module's Directives

The `LoadModule` entry for a DSO must be present in the configuration file prior to any references to the module's directives.

The directives are implemented as C functions within the module's code. An arbitrary number of directives can be implemented in a module. Typically, a directive accepts one parameter but multiple parameters are also possible. For example, the `Options` directive (implemented in the core module) supports seven parameters.

4.6. MIME Types and mod_mime

MIME (Multipart Internet Mail Extensions) types are used by Apache to write the `Content-type` in the HTTP header. The `Content-type` informs the browser of what type of file has been served and is equivalent to the MIME type.

The `mod_mime` module writes the content type based on the file extension. For example, a file with an extension of `.html` is assigned a content type of `text/html`.

The MIME value consists of a "type" and a "subtype." For example, the content type of `text/html` contains the type `text` and the subtype `html`.

The MIME module is enabled by default in the Apache configuration using the "TypesConfig" directive. The directive references `/etc/mime.types`, the file that maps file extensions to MIME types.

Exercise 4: Activating a Module

20 to 30 minutes

In this exercise, you will test the `server_status` module by adding a `Location` container to your configuration. You will test to make sure the module is active. Next, you will comment out the module in the configuration, disabling the server status module. You will restore the module by removing the comment.

1. Stop the Apache Web Server.
2. Open `/etc/httpd/conf/httpd.conf` for edit. Add the following Location container:

```
<Location /status>
        SetHandler server-status
      </Location>
```

3. Save your changes.
4. Start the Apache Web Server.
5. In your browser type:


```
localhost/status
```

6. You should see the server status web page.
7. Stop the Apache server.
8. Open `/etc/httpd/conf.modules.d/00-base.conf` for edit. Locate the entry for the server status module::

```
LoadModule status_module modules/mod_status.so
```

9. Comment out the load module directive:

```
#LoadModule status_module modules/mod_status.so
```

10. Save your changes.
11. Start the Apache server.

12. In your browser type:

```
localhost/status
```

13. You should receive a “Not Found”.
14. Stop the Apache server.
15. Return to the edit session on `/etc/httpd/conf.d/modules.conf`. Remove the “#” symbol to uncomment the server-status load module directive:

```
LoadModule status_module modules/mod_status.so
```

16. Save your changes.
17. Start the Apache server.
18. In your browser type:

```
localhost/status
```

19. You should now see the server status web page.

Conclusion

In this lesson, you have learned:

- About the purpose of modules.
- About dynamic shared objects.
- About the core module.
- About static modules.
- How to build modules from source and install from a package.
- About the LoadModule directive.
- About the relationship between directives and modules.
- How MIME types are used by Apache and the role of mod_mime.

LESSON 5
Security

Topics Covered

☑ Secure socket layer.

☑ Configuring access control with `Require`.

☑ Basic authentication.

☑ Digest authentication.

☑ LDAP authentication.

☑ Apache authorization.

Introduction

Securing your Apache Web Server is obviously important so that web applications, data, and files are not accessed in a malicious manner. Broadly speaking, Apache offers two security models. One is securing data going out on the wire using secure socket layer (SSL). The second model involves verifying users with prompts for credentials.

In this lesson, you will learn the techniques to implement these security models.

5.1. Secure Socket Layer

Secure Socket Layer (SSL) is a protocol designed to secure transmission of data over the wire. The protocol involves the use of a digital certificate that has been certified by a "certificate authority" (e.g., Verisign). In addition to a certificate, the client and the server each have a "key pair."

A key pair contains a public key and a private key. For example, the client can encrypt a message with the server's public key and send that message to the server. The server decrypts the message with its private key. The private key is known only to the server.

A key pair contains a public key and a private key. For example, the client can encrypt a message with the server's public key and send that message to the server. The server decrypts the message with its private key. The private key is known only to the server.

❖ 5.1.1. Creating a Certificate

The `openssl` tool can be used to create a "self-signed" certificate. Although this certificate might be adequate for testing purposes, it should not be used in production because the browser will warn the user that the certificate cannot be verified.

The following procedure can be used to generate a private key and self-signed certificate for testing SSL:

- Open a terminal window.
- Activate the SSL support in Apache:

```
sudo yum install mod_ssl openssl
```

- Create a folder where you can store your key and certificate:


```
sudo mkdir /apacheSecure
```

- Ensure that Apache can access your directory:

```
sudo chmod 755 /apacheSecure
```

- Create the certificate signing request:

```
sudo openssl req  -new -out /apacheSecure/testcertificate.csr
```

  Respond to each prompt by entering the requested data. The first prompt will request a "pass phrase". You may enter arbitrary text (e.g., "Apache"). The pass phrase will be required in the next step.

- Generate the key file:

```
sudo openssl rsa -in privkey.pem -out /apacheSecure/testcertificate.key
```

- Generate the test certificate for Apache using the certificate signing request and the key file created previously:

```
sudo openssl x509 -in /apacheSecure/testcertificate.csr  -req -signkey /apacheSe  ↵↵
cure/testcertificate.key
    -out /apacheSecure/testcertificate.crt -days 365
```

❖ 5.1.2. Configuring Apache for SSL

Follow these steps to configure Apache for SSL:

1. Stop the Apache server.
2. Open `/etc/httpd/conf.d/ssl.conf` for edit.
3. Locate the `SSLCertificateFile` and change the path to the certificate file you created:

```
SSLCertificateFile "/apacheSecure/testcertificate.crt"
```

4. Locate the `SSLCertificateKeyFile` and change the path to the certificate file you created:

```
SSLCertificateKeyFile "/apacheSecure/testcertificate.key"
```


5. Save your changes.
6. Start the Apache service.
7. To test, go to `https://localhost`. The browser will challenge the request because the certificate is not certified by a Certification Authority (e.g., Verisign). You will have the option to accept the risk and to be allowed to view the web page.

For more information on SSL and Apache, consult the Apache documentation at `http://httpd.apache.org/docs/current/ssl/`.

5.2. Access Control

Access control secures directory access based on criteria such as IP address or user ID. The directive that controls access is `Require`. The directive has "core" options implemented by `mod_authz_core` and additional options implemented by several modules.

Note

The `Order` directive is deprecated in Apache 2.4 and will be discontinued in future releases of Apache. Therefore, `Order` should not be used.

❖ 5.2.1. Host and IP

Directories can be secured so that they are accessible only by requests originating from certain host names or IP addresses.

To understand how access control works, we will consider securing a directory named `accesscontroldir`. Create this directory under the root ("/") of the file system.

Open `httpd.conf` for edit. Locate the following section:

```
<IfModule dir_module>
  DirectoryIndex index.html
</IfModule>
```


Add the following `Alias` directive and `Directory` section to the file:

```
Alias /access /accesscontroldir
<Directory /accesscontroldir>
  Options Indexes
  IndexOptions HTMLTable
  #Require local
</Directory>
```

Notice that the `Require` directive is commented out. This will cause an attempt to reference the directory to fail.

Save your changes. Leave the editor open as we'll be returning to the edit session.

Restart the Apache server and try to access the directory by typing the following URL in your browser:

```
http://localhost/access
```

The browser responds by displaying the HTTP status "403" returned by Apache.

Return to the edit session. Uncomment the `Require` directive:

```
Alias /access /accesscontroldir
<Directory /accesscontroldir>
  Options Indexes
  IndexOptions HTMLTable
  Require local
</Directory>
```

Restart the Apache server and type the following URL in your browser:

```
http://localhost/access
```

The browser responds by displaying an index of the directory.

Individual host names and/or IP addresses can be supplied with `Require`. The following directives will permit access to the directory from requests originating from host "mysite.com" or IP "192.1.2.3":

```
Require host mysite.com
Require ip 192.1.2.3
```


You can specify that all incoming requests regardless of host name or IP are permitted:

```
Require all granted
```

You can specify that all incoming requests regardless of host name or IP are **not** permitted:

```
Require all denied
```

5.3. Authentication

Authentication is the process of identifying a user. This is accomplished by "challenging" the user with a dialog box so that the individual can enter his/her user ID and password. The credentials will then be verified by Apache before the user is allowed to access the requested resource.

❖ 5.3.1. Basic Authentication

Basic authentication is configured with directive options implemented by `mod_auth_basic`. A user ID and password must be established using the `htpasswd` utility. The resulting password file must be configured in `httpd.conf`.

We will perform a demonstration to set up basic authentication.

1. Open a terminal window. Type the following command to install Apache authentication setup :

```
sudo yum install httpd-tools
```

2. Create the user data file:

```
sudo htpasswd -c /apacheSecure/apacheUsers.data Sam
```

   This will create the password file named `apacheUsers.data` containing a user ID of "Sam." You will be prompted for a password. Enter "sampassword". You will be prompted to retype the password. Enter "sampassword" again.

3. You will now enter another user into the file created. Type the following command:


```
sudo htpasswd /apacheSecure/apacheUsers.data Mary
```

   Notice that the "-c" option has been omitted since we wish to use the existing file. You will be prompted for a password again. Enter "marypassword". You will be prompted to retype the password. Enter "marypassword" again.

4. Open `httpd.conf` for edit.
5. Locate the directory container for "accesscontroldir" that you created earlier in this lesson.
6. Add the authentication directives and options to the directory container as shown below:

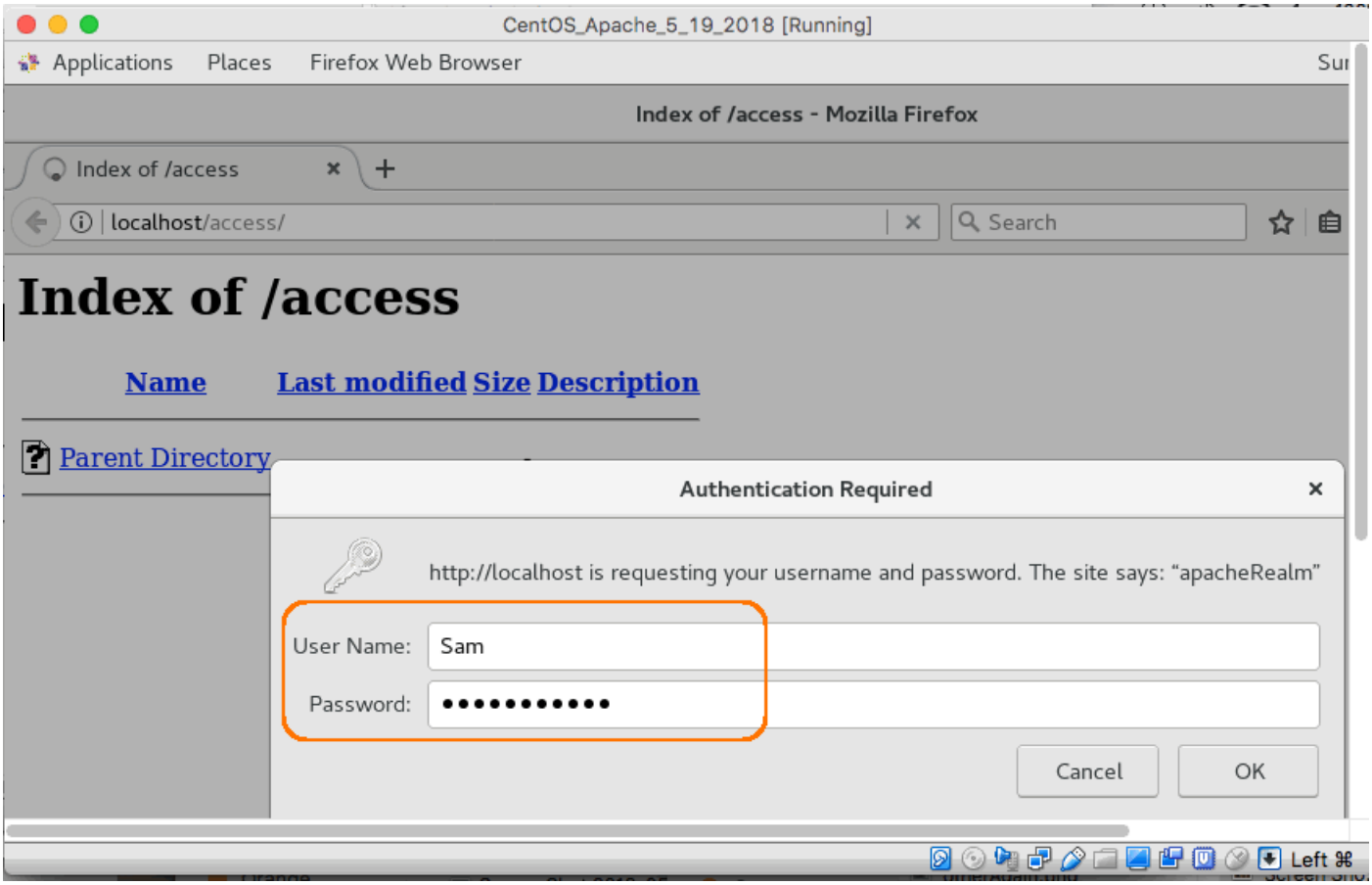
```
<Directory /accesscontroldir>
  Options Indexes
  IndexOptions HTMLTable
  AuthType Basic
  AuthName "Apache Security Team"
  AuthUserFile /apacheSecure/apacheUsers.data
  Require valid-user
</Directory>
```

Note that you have modified the `Require` directive to indicate a valid user must be identified before the directory can be accessed.

7. Save your changes. Leave the file open in the text editor as you will be making an additional modification after you test the current configuration.
8. Restart the Apache server and try to access the directory by typing the following URL in your browser:

```
http://localhost/access
```

9. You will be prompted for user name and password. Enter "Sam" for the user name and "sampassword" for the password:

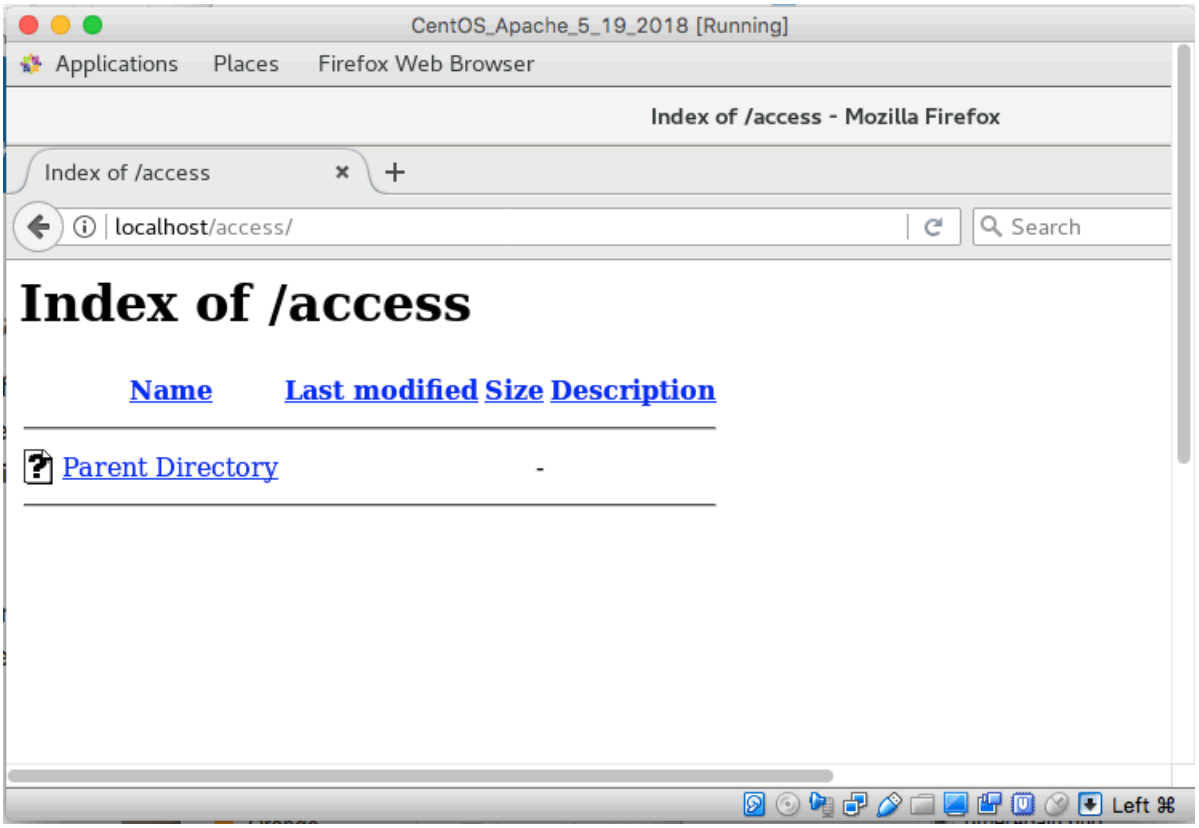


10. The directory index will be displayed.
11. Return to the editing session. Change "valid-user" to "user Mary". Save your change.
12. Restart the Apache server.

13. In your browser, delete active sign-on data. Type the following URL in your browser:

```
http://localhost/access
```

14. You will be prompted again for user name and password. Enter “Sam” for the user name and “sampassword” for the password.
15. The request will fail causing the dialog box to remain open with the fields cleared. Type “Mary” and “marypassword”.
16. The directory index will be displayed:



❖ 5.3.2. Digest Authentication

Digest Authentication is similar to basic authentication except that the password’s digest is sent to server as opposed to the clear text value of the password. The password is converted using the MD5 (Message Digest 5) algorithm. The value sent over wire is virtually impossible to translate back to the original clear text value.

The `Digest` option of the `AuthType` directive is implemented by `mod_auth_digest`.

5.4. How Apache Implements Authorization

Authorization is the process of determining if a user can access a resource.

In the demonstration from the "Basic Authentication" section, you defined two users, Sam and Mary. Initially, both users were authorized to access a target directory. Then you changed the `Require` directive so that only "Mary" was authorized to access the directory.

The `Require` directive is used to implement authorization in Apache. Authorization controls access to a resource (e.g., a directory).

5.5. LDAP Authentication and Authorization

An LDAP (Lightweight Directory Access Protocol) server can be used to authenticate users and therefore control authorization. The module required to perform the authentication and authorization is `mod_authnz_ldap`. The following directive in `apache2.conf` declares this module:

```
LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
```

`mod_authnz_ldap` can use `mod_auth_basic` as an authentication front end. The following directives set up LDAP authentication:

```
AuthType Basic
    AuthBasicProvider ldap
```

Authentication involves the following steps performed by `mod_authnz_ldap`:

1. Generate a search filter by combining the attribute and filter (specified in the `AuthLDAPURL` directive) with the user name provided by the user.
2. Search the LDAP directory seeking one match.
3. If matched, then read the DN (Distinguished Name). If not matched, then reject the request and exit.

4. Attempt to bind to the LDAP server using the DN and the password provided by the browser. If unsuccessful reject the request and exit.

The `AuthLDAPURL` directive is used during authentication. This directive specifies the URL of the LDAP server, the base DN, and the search attribute and filter.

During authorization `mod_authnz_ldap` determines if the user is permitted to access the resource. `mod_authnz_ldap` will grant access if one of the following is true:

1. `Require ldap-user` directive is present and user name in the directive matches user name provided by the user.
2. `Require ldap-dn` directive is present and the DN in the directive matches the DN retrieved from the LDAP directory.
3. `Require ldap-group` directive is present and the DN retrieved from the directory, or the user name provided by the user, occurs in the LDAP group specified by the directive or one of its subgroups.
4. `Require ldap-attribute` directive is present and the attribute retrieved from the directory matches the attribute specified by the directive.
5. `Require ldap-filter` directive is present and the search filter finds a single user object that is equal to the DN of the authenticated user.

If none of the previous cases is successful, then the access is denied.

5.6. Limit Directive

By default, access controls are in effect for all HTTP methods (e.g., GET, POST). This is generally acceptable.

The `Limit` directive restricts the access controls to certain HTTP methods. For example, to limit the access control to GET the following `Limit` directive can be coded:

```
<Limit GET>
  Require valid-user
</Limit>
```

The `Require` directive will have no effect on all other HTTP methods (e.g., POST, DELETE).

5.7. LimitExcept Directive

The `LimitExcept` directive is the opposite of `Limit`. `LimitExcept` specifies access controls that should not be applied to the HTTP methods listed in the directive.

For example, to limit the access control to all HTTP methods except GET, the following `LimitExcept` directive can be coded:

```
<LimitExcept GET>
  Require valid-user
</Limit>
```

The `Require` directive will have no effect on the GET HTTP method.

Exercise 5: Using Digest Authentication

30 to 45 minutes

In this exercise, you will implement digest authentication to secure files with a directory.

1. Stop the Apache Web Server.
2. Open `httpd.conf` for edit.
3. Locate the `Directory` section you developed for the lesson demonstration.
4. Change “AuthType Basic” to “AuthType Digest”.
5. Change “AuthName ”Apache Security Team"“ to ”AuthName apacheRealm".
6. Comment out the `AuthUserFile` directive. On the next line, insert the updated form of the directive:


```
AuthUserFile "apacheSecure/apacheUsersDigest.data"
```

7. Compare your results with the `Directory` container shown below:

```
<Directory /accesscontroldir>
  Options Indexes
  IndexOptions HTMLTable
  AuthType Digest
  AuthName apacheRealm
  #AuthUserFile "/apacheSecure/apacheUsers.data"
  AuthUserFile "/apacheSecure/apacheUsersDigest.data"
  Require valid-user
</Directory>
```

8. Add a `Files` container within the directory container as shown below:

```
<Directory /accesscontroldir>
  Options Indexes
  IndexOptions HTMLTable
  AuthType Digest
  AuthName apacheRealm
  #AuthUserFile "/apacheSecure/apacheUsers.data"
  AuthUserFile "/apacheSecure/apacheUsersDigest.data"
  Require valid-user
  <Files "secure.txt">
    Require user Linda
  </Files>
</Directory>
```

   Note that "secure.txt" is a text file in the "accesscontroldir" directory. This file is provided in the Exercises folder for this lesson. Copy this file to `/accesscontroldir`.

9. Open a terminal window. Type the following command:

```
sudo htdigest -c /apacheSecure/apacheUsersDigest.data apacheRealm Paul
```

   This will create the digest file named `apacheUsersDigest.data` with a user ID of "Paul" in a realm called apacheRealm. You will be prompted for a password. Enter "paulpassword". You will be prompted to retype the password. Enter "paulpassword" again.

10. You will now enter another user into the file you just created. Type the following command:

```
sudo htdigest /apacheSecure/apacheUsersDigest.data apacheRealm Linda
```

    Notice that the "-c" option has been omitted since we wish to use the existing file. You will be prompted for a password. Enter "lindapassword". You will be prompted to retype the password. Enter "lindapassword" again.

11. Start Apache.

12. Enter the following address in your browser:

```
http://localhost/access
```

13. You will be prompted for user name and password. Enter "Linda" for the user name and "lindapassword" for the password.

14. The `secure.txt` file will be displayed in the directory list.

15. Clear the sign-on data in your browser.

16. Enter the following address in your browser:

```
http://localhost/access
```

17. You will be prompted for user name and password. Enter “Paul” for the user name and “paulpassword” for the password.
18. The `secure.txt` file will not be present in the directory list.

Conclusion

In this lesson, you have learned:

- About secure socket layer.
- How to configure access control with Require.
- About basic authentication.
- About digest authentication.
- About LDAP authentication.
- How Apache implements authorization.

LESSON 6
Logging

Topics Covered

☑ Logging.

☑ Seting the log level.

☑ The access log.

☑ Setting up a log file with `CustomLog`.

☑ Formatting a log record with `LogFormat`.

☑ Log rotation.

Introduction

Logging is the process of writing messages to a file. The messages originate from modules and web applications. Apache supports two logs: the `access log` and the `error`.

6.1. Logging Overview

❖ 6.1.1. The Error Log

Apache writes error messages to the error log. Errors that arise from processing the configuration file are recorded in this log.

The default name for this log is `error.log` in Linux.

The log directory referenced in the Apache configuration is `/var/log/httpd`.

❖ 6.1.2. Setting the Log Level

The logging level of the error log can be established using the "LogLevel" directive. The following line is present in in the `httpd.conf` file for the log level:

```
LogLevel warn
```

This setting indicates that warning messages and more severe messages will be written to the log. Less severe messages (e.g. "info" messages) will not be written to the log.

The following table lists the log levels in increasing order of severity

Level	Meaning
trace8, trace7	Dump large amounts of data.
trace7	Dump large amounts of data.
trace6..trace1	Additional trace levels 6, 5, 4, 3, 2 and 1.
debug	Debugging messages.
info	Informational messages.
notice	Informational messages but of slighly higher significance.
warn	Warning messages.
error	Messages stating an error occurred.
crit	Messages indicating critical situations that could interfere with the successful operation of the server.
alert	Messages that indicate action is required.
emerg	Messages indicating emergency situations that must be addressed in order to stabilize the server.

❖ 6.1.3. The Access, Agent, and Referer Logs

Apache writes three other types of messages.

1. **access**: requests received by the web server.
2. **user agent**: browser that sent the request.
3. **referer**: referring URL from which the request has arrived.

These messages could be destined for separate log files; however, the configuration as shipped combines these message types into one format with the nickname of "combined." The nickname can be used in a `CustomLog` directive in order to generate a log file.

The following lines are present in the `httpd.conf` file:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
  LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

The format specifiers are defined below:

Format specifier	Meaning
%a	Client IP address and port number.
%h	Remote host name.
%H	The request protocol.
%l	Remote log name.
%u	User name.
%t	Date and time the request was received.
%r	The request (HTTP method and URL).
%m	The request method.
%q	The query string.
%>s	Last status of the request. The > sign specifies only the last status is displayed.
%b	The number of bytes sent.
%{Referer}i	Item extracted from the HTTP header for the referer.
%{User-Agent}i	Item extracted from the HTTP header for the user agent, or browser.

For a complete list go to `http://httpd.apache.org/docs/2.4/mod/mod_log_config.html#formats`.

6.2. CustomLog

The `CustomLog` directive determines the destination of log message output. `CustomLog` is implemented in the module `mod_log_config`.

The content of the log message is determined by the log format nickname specified by `CustomLog` or by a string containing format specifiers.

The following `CustomLog` statements are provided in the "factory" `httpd.conf` file:

```
CustomLog "logs/access_log" combined
```

The "common" refers to the log format that has been previously defined in the configuration file. In the lesson exercise, you will have the opportunity to use the combined format in addition to formats that you customize.

6.3. Formatting the Log Record with LogFormat

The `LogFormat` directive permits you to define the format of a log record. The directive accepts a string that contains format specifiers (presented above) and text. `LogFormat` is implemented in the module `mod_log_config`.

6.4. Log Rotation

Log rotation is the process of creating a new log file after a time interval has elapsed or when a specified size has been reached. Log rotation is compelling for production systems because of the large amount of data written to the log. If left unchecked, these files could grow quite large, impacting disk resources and becoming too unwieldy to inspect in the event of errors.

Apache is shipped with a binary named `rotatelogs` that can be used to implement log rotation. This utility can be accessed through the `CustomLog` and `ErrorLog` directives using log file piping.

We will consider an example. The access log will be rotated every hour and the log file name will contain an extension generated automatically by `rotatelogs`. Here is the `CustomLog` directive in `httpd.conf` to accomplish this:

```
CustomLog "|rotatelogs /logs/mylog 3600" common
```

The first parameter is the piping to `rotatelogs`. Next, the log file name is provided followed by a rotation time interval in seconds. The value of "3600" requests that the log be rotated every hour. When a log file is generated, the system time will be used as the extension (e.g., mylog.1346648940).

The final parameter in the example above is the log format nickname.

A file extension can be specified in the directive by using string substitutions. The following is a sample of string substitutions:

String substitution	Meaning
%A	Day of the week name
%a	3-character day of the week name
%B	Month name
%b	3-character month name
%Y	4-digit year
%y	2-digit year
%m	2-digit month
%d	2-digit day of month
%H	24 hour 2-digit hour
%h	12 hour 2-digit hour
%M	2-digit minute
%S	2-digit second

The `rotatelogs` program accepts several options. We will consider two options in the section. The first option is `-l`. This option tells `rotatelogs` to use local time as the base for rotation (GMT is the default):

```
CustomLog "|rotatelogs -l /logs/mylog.%h%M%S 3600" common
```

An extension of an hour, minute, and second has been specified. Every hour the log file will rotate and a new log file will be generated. If the log file rotates at 4:00PM local time, then the name of the new log file will be `mylog.040000`.

The second option we consider is `-t`. This option directs `rotatelogs` to truncate the log file. All records will be deleted and the log file name will not be suffixed.

We consider an example:

```
CustomLog "|rotatelogs -t /logs/mylog.log 3600" common
```

The log file records will be deleted every hour prior and then new records will be written as required.

A maximum size can be specified instead of a time interval to determine when a new log file should be created. The following example will generate a new log file when the size of the current log file reaches 2 gigabytes:


```
CustomLog "|rotatelogs -l /logs/mylog 2G" common
```

The size can be specified as “B” (bytes), “K” (kilobytes), “M” (megabytes), or “G” (gigabytes).

For more information on `rotatelogs` and a complete list of string substitutions, go to `http://httpd.apache.org/docs/current/programs/rotatelogs.html`.

Exercise 6: Generating a Custom Log

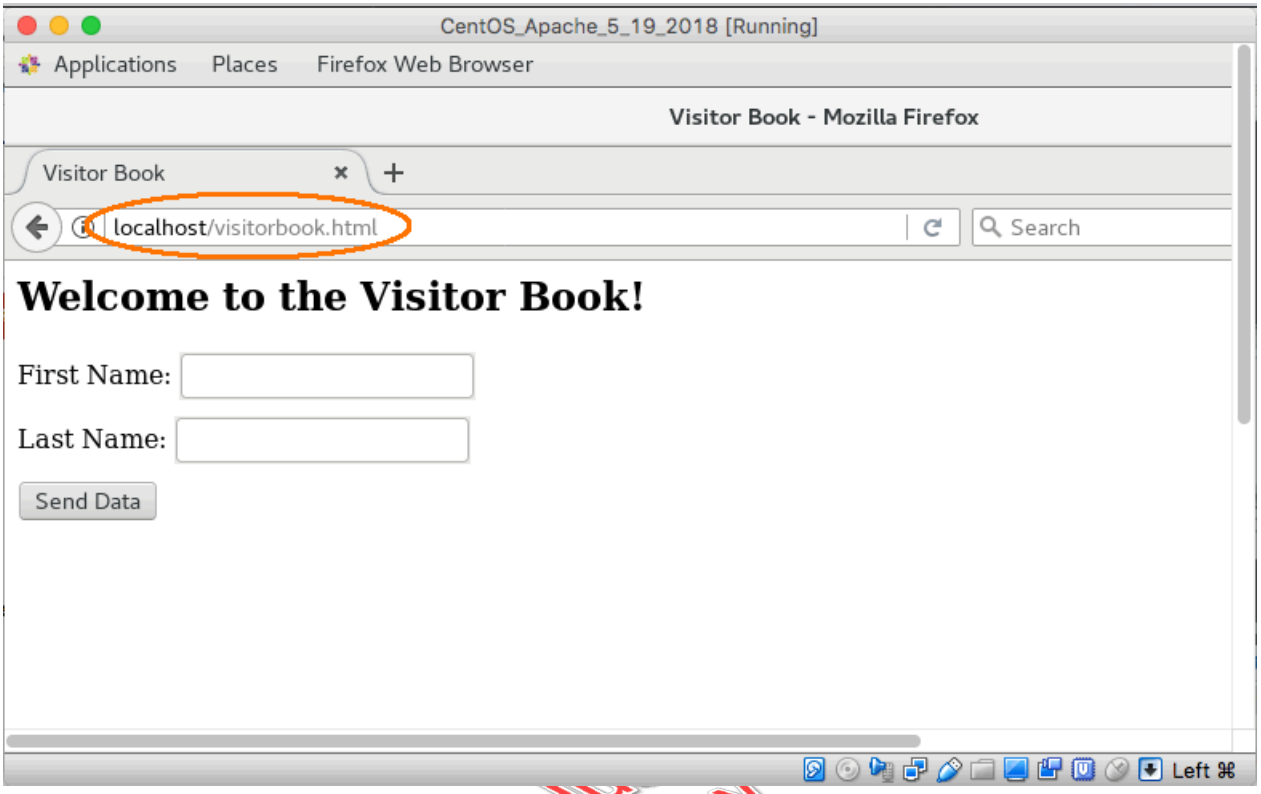
30 to 45 minutes

In this exercise, you will modify the configuration to generate a custom log. The log will be routed to a location in the file system outside of `/var/log/httpd`.

1. Stop the Apache Web Server.
2. Delete the access log from the `/var/log/httpd` folder.
3. Create a new folder under the root named `mylogs`. Be sure permissions are set so that all users can create files in this folder. You will route one of your custom log files to this folder.
4. A very simple web application is available in the `logging/exercises` folder to generate request activity to the access log. Copy `visitorbook.html` and `visitorbookresponse.html` to `/var/www/html`.
5. Open `/etc/httpd/conf/httpd.conf` for edit.
6. Locate the following line:


```
LogFormat "%h %l %u %t \"%r\" %>s %b" combined
```

7. Insert a new `LogFormat` directive with a nickname of "requestTimeStatus". The log record should consist of the request, the time (date included)and the last HTTP status.
8. Locate the `CustomLog` directive that references the "common" nickname. Copy this line and comment out the original line.
9. Change the copied line replacing "common" with "requestTimeStatus".
10. Change the copied line replacing `logs` with `/mylogs`.
11. Save your changes.
12. Start Apache.
13. In your browser, display the welcome screen at `http://localhost`.
14. Display the guestbook in the sample web application at `http://localhost/webapp/visitorbook.html`:



15. Enter a first and last name and then click the **Send Data** button.
16. In File Manager, navigate to `/mylogs` and open the generated log file in a text editor. Verify that the request, status, and time are present in the log record. The request will contain the query string consisting of the data you entered into the form.
17. Close the text editor.
18. Create another guestbook entry by providing a name that is different from the previous name you entered.
19. Open the log file again to verify that another log entry has been generated.

Conclusion

In this lesson, you have learned:

- About logging.
- How to set the log level.
- How to use the access log.

- How to set up a log file with CustomLog.
- How to format a log record with LogFormat.
- About log rotation.

LESSON 7
Configuring Directories

Topics Covered

☑ Directory containers in `httpd.conf`.

☑ Specifying directory options.

☑ Indexing with `IndexOptions`.

☑ Directory Index files.

☑ Excluding files with `IndexIgnore`.

☑ The `.htaccess` file.

☑ Handling HTTP status codes with `ErrorDocument`.

☑ Using `Location` containers.

Introduction

A Directory container specifies what type of access will be permitted to a directory and its subfolders. Directory containers are coded in `httpd.conf` or in one of its included files.

A directory index can be presented to the client if a specific file is not requested. The formatting of the index is controlled by the Directory container.

Directory options can be specified on a directory basis by using a `.htaccess` file. This is permitted only if the configuration file allows overrides for this directory.

7.1. Directory Containers in httpd.conf

The default configuration in `httpd.conf` contains a Directory container that secures the entire file system of the server. Here is the Directory section:

```
<Directory />
  AllowOverride none
  Require all denied
</Directory>
```

Notice that the container definition starts with `<Directory>` and terminates with `</Directory>`. Although this syntax might remind you of XML, the definition is not XML. This syntax is provided so that one or more directives can be applied to the directory specified on the `Directory` directive.

The directory in this example is `/` and applies to the entire file system of the server. File system meta characters such as "*" can be used in the syntax.

The `AllowOverride` directive is given a parameter of `none`. Therefore, no overrides are permitted in an `.htaccess` file. `.htaccess` files will be discussed later in this lesson.

`Require all denied` indicates that access to the file system is denied to all clients.

The `/var/www/html` directory is specified as the document root in the `DocumentRoot` directive. This directive alone is not sufficient to permit access to `/var/www/html`. A Directory container must also be present to indicate what access is permitted.

This Directory container is provided in the configuration file and is shown below:

```
<Directory /var/www/>
  Options Indexes FollowSymLinks
  AllowOverride None
  Require all granted
</Directory>
```

An `Options` directive is present. This directive will be explained in the next topic.

7.2. The Options Directive

The `Options` directive indicates what options are available for the given directory. The various parameters that can be coded on this directive are presented below:

Parameter	Meaning
All	All options except multiviews
ExecCGI	CGI scripts may be executed
FollowSymLinks	Symbolic links can followed; this is the default
Includes	Server side includes are permitted
IncludesNOEXEC	Server side includes are permitted but `#exec` and `#exec cgi` are disabled
Indexes	A formatted listing of the directory is returned if a directory index file (e.g., `index.jsp`) is not present in the directory
MultiViews	Content negotiation is permitted (e.g., the HTTP header on the request indicates a ".jpeg" or a ".gif" file will be acceptable)
SymLinksIfOwnerMatch	Symbolic link can only be followed if the link and target file or directory are owned by the same user ID

An option can be preceded with a `+` or `-` sign. An option preceded with a `+` is added to whatever options are in effect and an option preceded with a `-` is removed from whatever options are in effect.

Let's consider an example. Here is the first Directory section:

```
<Directory /var/www/mainfolder>
  Options Indexes
</Directory>
```

The `Indexes` option is now in effect for `mainfolder`. The next Directory section is presented below:

```
<Directory /var/www/mainfolder/subfolder>
  Options +Includes
</Directory>
```

The `Includes` is merged with `Indexes` for `mainfolder` so that these two options are in force for this folder.

Note

If a + or - precedes any option then if any options does not have a preceding + or -, then a syntax error results and Apache will not start.

7.3. Directory Indexing

The `Indexes` option permits Apache to return a formatted directory listing for a directory.

❖ 7.3.1. IndexOptions

The format of the index can be specified using `IndexOptions`.

The following table lists some of the formatting options available on `IndexOptions`:

Option	Meaning
FancyIndexing	Enables sorting by columns
HTMLTable	Displays the index in a table
FoldersFirst	Displays folders before displaying files
SuppressColumnSorting	Disables column sorting in fancy indexing
SuppressDescription	Removes description column from the output in fancy indexing
SuppressLastModified	Removes last modified date from the output in fancy indexing

An option may be preceded by a + or - sign. A + sign indicates that the option should be merged with any existing options. A - sign indicates the option should be removed from any existing options.

To demonstrate the `IndexOptions` directive, you will first create a folder under APACHE_HOME and then configure a directory container.

1. The folder and its contents are provided in `configuring-directories/Demos/mainfolder.zip`. Unzip this folder under `/var/www`. The folder name is `mainfolder`.
2. You will need to access this folder using a URL. To do this, you will add an `Alias` directive to `httpd.conf`.

3. Open `httpd.conf` for edit.
4. Locate the following line:

```
<Directory /var/www/>
```

   Skip to the closing statement, `</Directory>`.

5. Insert the following line:

```
Alias /mainf /var/www/mainfolder
```

   This directive associates the URL `/mainf` with the `mainfolder` directory. The `Alias` directive will be covered in a later lesson.

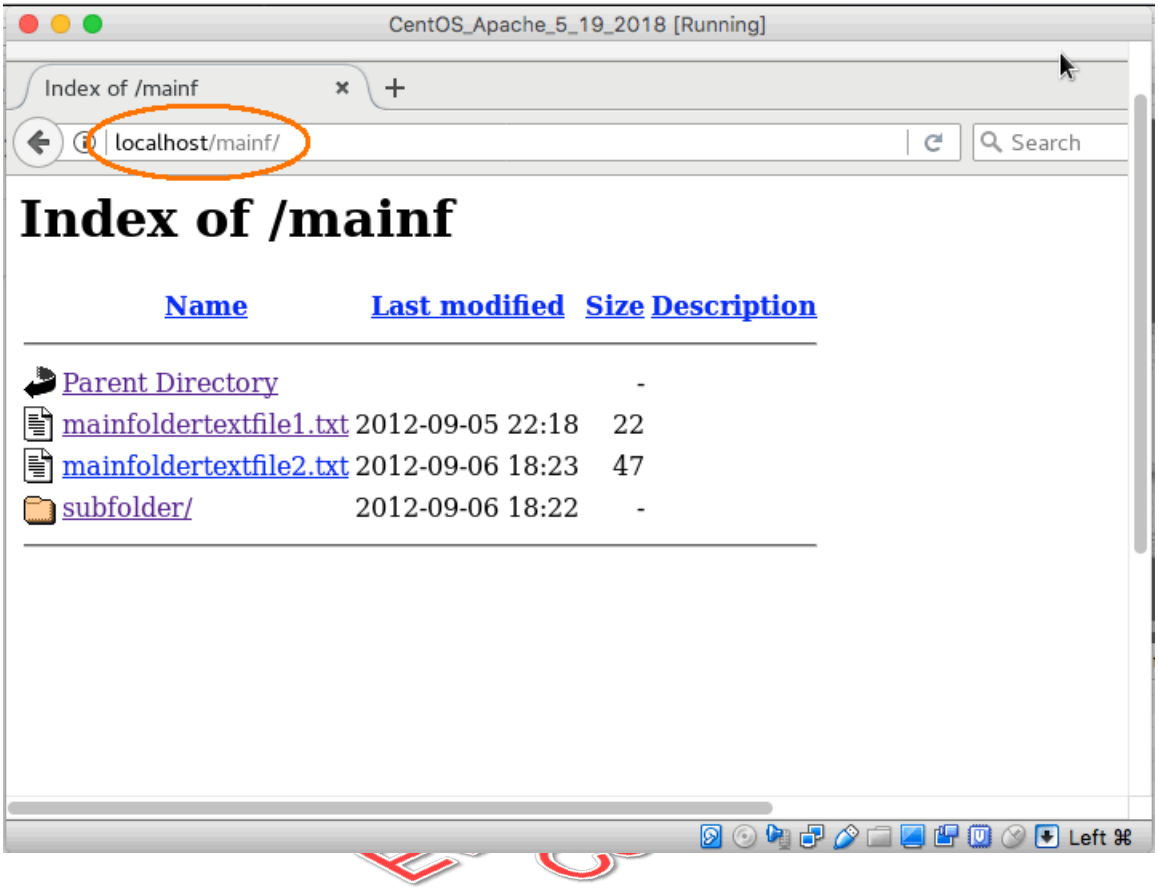
6. Insert the `Directory` section:


```
<Directory /var/www/mainfolder>
  Options Indexes
  IndexOptions FancyIndexing HTMLTable
  Require all granted
</Directory>
```

7. Save your changes and close the edit session.
8. Start Apache.
9. Type the following URL in your browser:

```
http://localhost/mainf
```

10. The directory listing will be displayed as an HTML table with sortable columns:



❖ 7.3.2. DirectoryIndex

The `DirectoryIndex` directive lists the file or files that can be returned to a client by default. For example, the following directive indicates `index.html` or `default.html` can be sent to a client if a file has not been requested:

```
DirectoryIndex index.html default.html
```

If the directory does not contain a file with one of the names listed and the `Indexes` option is in effect, then a directory listing will be sent to the client.

❖ 7.3.3. Excluding Files with IndexIgnore

By default the directory listing will contain all files. If you wish to exclude one or more files then you can use the `IndexIgnore` directive.

For example, the following excludes `Instructions.txt`, any file with an extension of `pdf` and any file that starts with `.`:

```
IndexIgnore Instructions.txt *.pdf .*
```

❖ 7.3.4. DirectoryMatch

The `DirectoryMatch` directive provides the same functionality as `Directory` except the directory can be specified as a regular expression.

For example, the following `DirectoryMatch` will match directories that begin with "training":

```
<DirectoryMatch ^training>
  Require all granted
</Directory>
```


7.4. .htaccess

The options for a directory can be stored in a file named `.htaccess`. For these options to take effect, an `Override` must be present in `httpd.conf` or one of its included files for the directory.

The `Override` directive specifies what directives may be used in `.htaccess`. For example, consider the directory section presented earlier:

```
<Directory mainfolder>
  AllowOverride Indexes
  IndexOptions FancyIndexing HTMLTable
  Require all granted
</Directory>
```

The `AllowOverride Indexes` permits use of directives such as `IndexOptions` in `.htaccess`.

Option	Meaning
AuthConfig	Permit use of authorization directives such as `AuthName`, `AuthGroup`, and `Require`
FileInfo	Permit use of directory type directives (e.g., `ErrorDocument`, `SetHandler`), document meta data (e.g., `RequestHeader`), and `Action`.
Indexes	Permit use of directives controlling directory indexing(e.g., `DirectoryIndex`, `FancyIndexing`)
Limit	Permit use of `Order`, `Allow`)
Nonfatal=	Allow syntax errors in `.htaccess` to be treated as non-fatal; can be set to `Unknown`,`Override`, or `All`
Options=	Allows use of specific directives for controlling directory features

For a complete description of the options available for `AllowOverride`, go to `http://httpd.apache.org/docs/current/mod/core.html#allowoverride`.

The primary advantage of using a `.htaccess` file is that configuration options can be stored on the directory to which they apply.

A `.htaccess` file may be located on a subfolder of a folder that contains a `.htaccess` file. If a conflict exists in the subfolder's `.htaccess` file options in reference to the those found in the file in the main folder, then the subfolder's file options take precedence.

7.5. Handling HTTP Status Codes with Error Documents

In the event of an error, Apache will send a web page to the client that contains the HTTP status code and a description of the error.

A customized response can be sent to the client using the `ErrorDocument` directive. The response can be a string or a web page. Alternatively a script can be executed. The HTTP status code that will trigger the response is specified as the first parameter.

The following directives exemplify each approach:

```
ErrorDocument 404 "The web page requested was not found on the server"
ErrorDocument 500 /internal_server_error.html
ErrorDocument 401 /cgi-bin/auth_error.pl
```

The "/" indicates the resource is located relative to the document root, e.g., `internal_server_error.html` is located under `htdocs`.

7.6. Location Containers

Location containers map an incoming request to a resource.

For example,the following Location container associates a request for "/status" with a handler:

```
<Location /status>
  SetHandler server-status
</Location>
```

The `LocationMatch` container is identical to `Location` except that the location can be specified as a regular expression.

You will learn more about Location containers and handlers in a later lesson.

Exercise 7: Configuring a Directory and a Subdirectory

30 to 45 minutes

In this exercise, you will continue configuring the main directory and subdirectory that you created in the lesson demo.

1. Stop the Apache Web Server.
2. The `mainfolder` developed in the lesson demonstration is available in `configuring-directories/Exercises/mainfolder.zip`. If you did not have an opportunity to build this folder, then extract this file under `/var/www/`. The `.htaccess` files are already provided.
3. Open `httpd.conf` for edit.
4. Locate the Directory container that you developed in the lesson demonstration for `mainfolder`:

```
<Directory /var/www/mainfolder>
  Options Indexes
  Require all granted
</Directory>
```


5. Comment out the `Options` directive.
6. Insert the following line after the comment:

```
DirectoryIndex mainfoldertextfile1.txt
```

7. Save your changes.
8. Start Apache.
9. Enter the following address in your browser: `http://localhost/mainf`
10. The web page generated from the text file will be displayed.
11. Return to the edit session and uncomment the `Options` directive. Comment out the `DirectoryIndex` directive you coded above.
12. Save your changes.
13. Restart Apache and enter the same address as above.
14. Question 1: What do you observe? Why?

15. Enter the following address: `http://localhost/mainf/subfolder`
16. Question 2: What do you observe? Why?
17. Return to edit session and insert the following lines after the `mainfolder` directory container you have been working on:

```
<Directory /var/www/mainfolder/subfolder>
  IndexOptions +SuppressColumnSorting
</Directory>
```

18. Save your changes and restart Apache.
19. Enter the following address: `http://localhost/mainf/subfolder`
20. Question 3: What happens when you attempt to sort column data by clicking a column header? Why?
21. You will now modify the current configuration for `mainfolder/subfolder` using `.htaccess` files.
22. In your text editor open the file named `.htaccess` that is located under `mainfolder/subfolder`.
23. The file is currently empty. Add the following line:


```
IndexOptions -SuppressColumnSorting
```

This option indicates that column sorting is enabled.

24. This is the only line in the file. Save the file.
25. In your browser, enter the following address: `http://localhost/mainf/subfolder`
26. Notice that column sorting is still disabled, even though column sorting has been specified in `.htaccess`. Question 4: Why is the the option not taking effect?
27. Return to the edit session on `httpd.conf`. Modify the Directory section for `mainfolder/subfolder` as follows:

```
<Directory mainfolder/subfolder>
AllowOverride Indexes
#IndexOptions +SuppressColumnSorting
</Directory>
```

Note that you have added `AllowOverride Indexes` and commented out `IndexOptions +SuppressColumnSorting`.

28. Save the file.
29. Restart Apache.
30. Return to the browser and enter: `http://localhost/mainf/subfolder (http://localhost/mainf)`
31. Question 5: Are you now able to sort column data? Why or why not?
32. In your text editor, comment out all lines in `mainfolder/subfolder/.htaccess`. Save your changes.
33. Open `mainfolder/.htaccess` for edit. The file is currently empty. Add the following line:

```
IndexOptions +SuppressColumnSorting
```

34. Save your changes.
35. Open `httpd.conf` for edit.
36. Locate the Directory container that you developed in the lesson demonstration for `mainfolder`. Add a line to allow overrides in `.htaccess`:

```
<Directory /var/www/mainfolder>
Options Indexes
IndexOptions FancyIndexing HTMLTable
AllowOverride IndexOptions
Require all granted
</Directory>
```


37. Save the file.
38. Restart Apache.
39. Return to the browser and enter: `http://localhost/mainf`
40. Column sorting will be disabled on the main folder and its subfolder because of the setting in the `.htaccess` file under the main folder. The subfolder inherits the setting.
41. Finally, modify `mainfolder/.htaccess` so that column sorting is not suppressed:

```
IndexOptions -SuppressColumnSorting
```

42. Save your changes.
43. Keep the Apache server up and running. In your browser refresh the directory listing of either folder. You will now be able to sort the columns for both the main folder and its subfolder.

Changes made to `.htaccess` files are picked up by Apache every time a directory fetch is requested, therefore a server restart is not required.

Conclusion

In this lesson, you have learned:

- About Directory sections in httpd.conf.
- How to specify directory options.
- About indexing.
- About Directory Index files.
- How to exclude files with IndexIgnore.
- About the .htaccess file.
- How to handle HTTP status codes with ErrorDocument.
- About using Location and Directory section.

LESSON 8
Virtual Hosts

Topics Covered

☑ The Virtual Host container.

☑ Setting up an IP-based virtual host.

☑ Setting up an name-based virtual host.

☑ Setting up a port-based virtual host.

Introduction

A web host is essentially a container for web pages. Additional containers can be established on one Apache Web Server as "virtual hosts." A virtual host does not reside on a dedicated server, hence the adjective "virtual."

A virtual host is defined in a Virtual Host container. The document root for the host is specified within the container as well as logging destinations and other host-specific items.

A given virtual host must be identified by a unique IP, name, or port number.

8.1. Virtual Host Container

A virtual host is defined using a Virtual Host container. The `VirtualHost` directive defines this container. The `VirtualHost` directive specifies the address of the host and can be an IP address, domain name, a wildcard character ("*") or `_default_` (alias for the wildcard character).

The `_default_` identifies the host container that Apache selects if the incoming host address does not match any other container.

8.2. Setting Up the Virtual Host

❖ 8.2.1. IP-based

An IP-based virtual host is identified by an IP address on the `VirtualHost` directive. Consider the following example:

```
<VirtualHost 190.1.2.3:80>
    ServerName www.classserver.com
    DocumentRoot /classserver/docs
    ErrorLog logs/ErrorLog
</VirtualHost>
```

The IP address is "190.1.2.3" and the port is "80". If a request contains this IP and port, then Apache will apply the directives within the container to this request.

The document root is applicable to this virtual host and overrides any `DocumentRoot` directive elsewhere in the configuration file. The same is true for the other directives specified in this container.

❖ 8.2.2. Name-based

A name-based virtual host is identified by the server name provided in the Virtual Host container. The the wildcard character (or `_default_`) is present instead of an IP address

Here is an example:

```
<VirtualHost *:80>
    ServerName www.classroomserver.com
    DocumentRoot /classroomserver/docs
    ErrorLog logs/classroomserver_Error.Log
</VirtualHost>
```

The IP address is not specified. The name in the request will be compared to the server name that is present within the Virtual Host section. If matched, then Apache will apply the directives within the container to this request.

If a match does not occur (i.e., Apache cannot match the request with a server name in a virtual host), then Apache will select the first container.

The document root is applicable to this virtual host and overrides any `DocumentRoot` directive elsewhere in the configuration file. The same is true for the other directives specified in this container.

Let's add a virtual host container for localhost to our configuration. Open `/etc/httpd/conf/httpd.conf` in your editor and add the following lines:

```
<VirtualHost *:80>
    ServerName localhost
    DocumentRoot /var/www/html
    ErrorLog logs/Error.Log
</VirtualHost>
```

Save your changes and then restart Apache. When you display the welcome page you will notice no difference from earlier tests. However, you now have a container for localhost in which you can specify host-specific settings such a document root and rewrite directives (covered in a later chapter).

❖ 8.2.3. Port-based

A port-based virtual host is identified by the port number provided prior to the Virtual Host container. Again, we consider an example:

```
Listen 80
<VirtualHost *:80>
    ServerName www.classroomserver.com
    DocumentRoot /classroomserver/docs
    ErrorLog logs/classroom_Error.Log
</VirtualHost>
Listen 90
<VirtualHost *:90>
    ServerName www.classroomserver.com
    DocumentRoot /classroomserver2/docs
    ErrorLog logs/classroom2_Error.Log
</VirtualHost>
```

The document root is applicable to this virtual host and overrides any `DocumentRoot` directive elsewhere in the configuration file. The same is true for the other directives specified in this container.

Exercise 8: Defining Name-based Virtual Hosts

30 to 45 minutes

In this exercise, you will define multiple name-based virtual hosts.

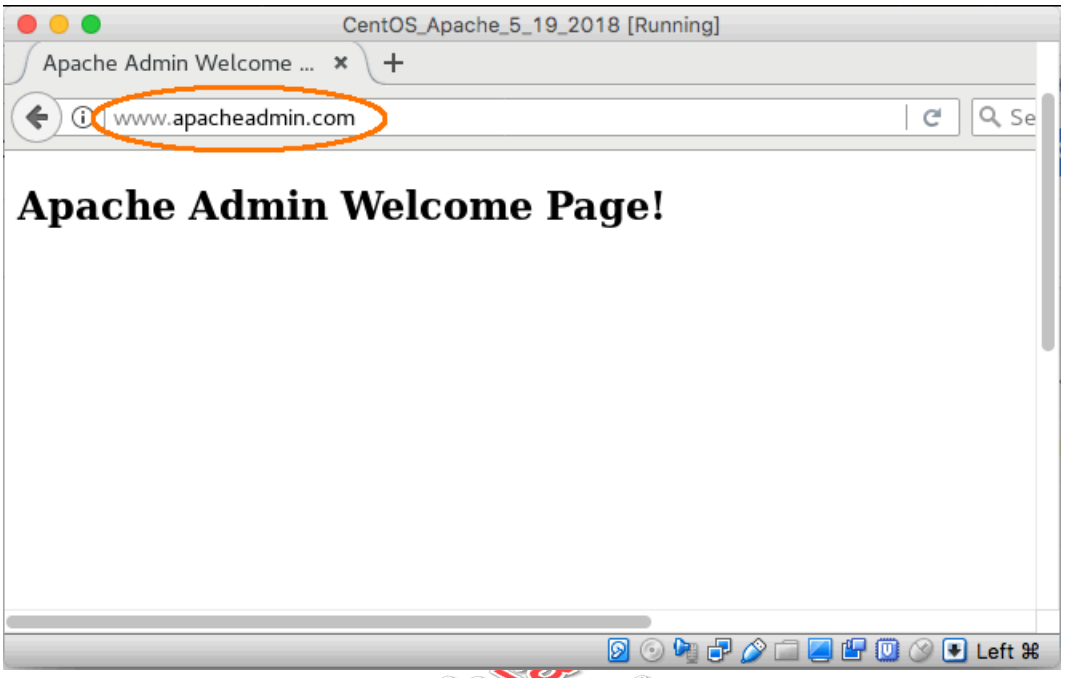
1. Stop the Apache Web Server.
2. Extract `virtual-hosts/Exercises/apacheadmin.zip` to `/var/www/apacheadmin`. The extract operation will create the `apacheadmin` folder.
3. Extract `virtual-hosts/Exercises/apacheadmin2.zip` to `/var/www/apacheadmin2`. The extract operation will create the `apacheadmin2` folder.
4. Open `/etc/httpd/conf/httpd.conf` for text editing. Add the following statements to this file after the localhost VirtualHost container you created during the classrooom demonstration:

```
<VirtualHost *:80>
  ServerName www.apacheadmin.com
  DocumentRoot /var/www/apacheadmin
</VirtualHost>
```


   Save your changes.
5. A domain name of `www.apacheadmin.com` and `www.apacheadmin2.com` must be assigned to the loopback IP (127.0.0.1) in order to test your virtual host.
6. The procedure for accomplishing this task varies depending on your operating system. For most Linux distributions, you will modify `etc/hosts`. Add the following lines at the end of the file:

```
127.0.0.1   www.apacheadmin.com
127.0.0.1   www.apacheadmin2.com
```

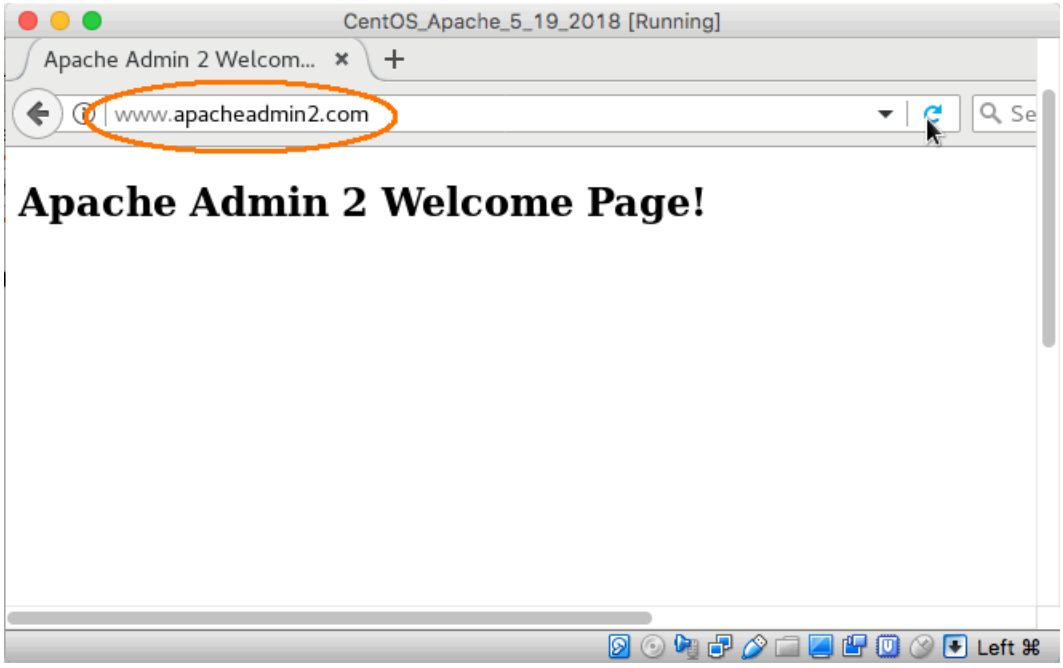
   Save your changes.
7. Start Apache.
8. In your browser, go to: `http://www.apacheadmin.com`
9. The following page will be displayed:



10. Stop the Apache server.
11. Extract `virtual-hosts/Exercises/apacheadmin2.zip` to `/var/www/apacheadmin2`. The extract operation will create the `apacheadmin2` folder.
12. Add the following statements to `/etc/httpd/conf/httpd.conf` after the VirtualHost container you coded earlier in the assignment :

```
<VirtualHost *:80>
  ServerName www.apacheadmin2.com
  DocumentRoot /var/www/apacheadmin2
</VirtualHost>
```

    Save your changes.
13. Start the Apache server.
14. In your browser, go to `http://www.apacheadmin2.com`
15. The following page will be displayed:



Conclusion

In this lesson, you have learned:

- About the Virtual Host section.
- How to set up an IP-based virtual host.
- How to set up an name-based virtual host.
- How to set up a port-based virtual host.

LESSON 9
Using Aliases and Redirecting

Topics Covered

☑ About configuring an alias for a URL.

☑ Redirecting requests.

☑ Using `mod_rewrite`.

Introduction

A common task for an administrator is associating a request with a web page located outside of the document root. This task is accomplished by using the `Alias` directive.

Another often-needed requirement is to redirect a request for an outdated or deleted web page to another web page.

In this lesson, you will learn how to accomplish these important tasks.

9.1. Configuring an Alias for a URL

The `Alias` directive is used to declare an alias of a URL (Uniform Resource Locator). This is typically done so that a request can be directed to a web page that is stored in a folder other than the document root. This directive is implemented by `mod_alias`.

We will consider an example where a web page named `TestPage.html` is stored in `/testdocs`. First, a directory section must be declared in `httpd.conf` so the directory can be accessed by Apache:

```
<Directory /testdocs>
  Require all granted
</Directory>
```

Next, we will place an alias definition in the configuration file:

```
Alias /testd /testdocs
```

A request for `/testd` will be mapped to `/testdocs`. Therefore, a web page request will be resolved by reading this directory and attempting to locate the document.

We can test the alias by pointing the browser to:

```
http://localhost/testd/TestPage.html
```

The web page will be retrieved by Apache and sent to the browser.

The directory container is required if the alias is located outside of document root directory. The Directory containers are processed after the Alias directive.

9.2. Redirect

The `Alias` is useful when mapping a URL to a directory resource on the web server's file system. If the URL must be mapped to another server, then the `Redirect` directive must be used.

Imagine a scenario where we have discontinued a "home-grown" search application. From now on, we would like to direct requests for the old application to a search engine, such as Google, that resides on a different server.

To accomplish this, we add the following line to `httpd.conf`:

```
Redirect /search http://www.google.com
```

A request for `/search` will be redirected to Google. Specifically, Apache will respond to the browser with the address of Google and an HTTP status of 302. The "302" indicates a temporary redirect to the browser. This means the original page requested is only temporarily unavailable.

Other options are available on the redirect. For instance:

```
Redirect permanent /search http://www.google.com
```

In this case, Apache will send a status of "301", which reflects a permanent redirect. A permanent redirect indicates the website has moved and all future references to the site should use the URL returned with the "301" status. For example, a search engine might update its database with the new URL upon encountering the "301" status.

The `Redirect` directive accepts a URL and maps it to a new URL. The original URL must be an absolute reference; i.e., a slash ("/") must be present as the first character. The new URL may have a hostname prepended to it or it may begin with a slash. If the new URL begins with a slash, then Apache will prepend the hostname to it prior to sending the URL to the client.

This directive is implemented by `mod_alias`.

9.3. Using mod_rewrite

The `mod_rewrite` module supports directives that permit the URL to be rewritten, or modified. Once the URL is rewritten, it can be processed by Apache just like any other request.

To use `mod_rewrite`, the "mod_rewrite.so" module must be enabled. By default, the module is enabled in most configurations. You can verify that the module is enabled by searching for "rewrite_module" in `/etc/httpd/conf.modules.d/00-base.conf` and ensuring that the module is not commented out with a "#" symbol.

Several directives are implemented by `mod_rewrite` The `RewriteEngine` directive turns the rewrite engine on or off (the default is "off"). The `RewriteRule` directive specifies the rule used to rewrite the URL. The rule is written as a regular expression.

We return to the previous example of redirecting a URL to Google. Instead of coding a `Redirect`, we will use `RewriteRule`:

```
RewriteEngine on
RewriteRule /search http://www.google.com
```

The URL to be rewritten in this case is `/search`. This URL is treated as a regular expression pattern by `mod_rewrite` and therefore may contain regular expression meta characters. The substitution URL in the example above is address of Google.

The rewrite rule will not redirect the request to the browser.

Rewrite rules can be applied to aliases as well. Consider the following rewrite rule that uses the alias presented earlier:

```
RewriteRule /classdocs(.*) /test$1 [R]
```

Note the use of `(.*)`. This is a regular expression that maps whatever text is encountered following `/classdocs`. This text is inserted after `test` by using `$1`.

The rewrite rule presented above permits another URL to be associated with the alias. In this way, multiple mappings can be achieved without repeating the Alias directive. If the filesystem reference changes, the change has to be applied only to the Alias directive.

Note the `[R]` in the example. This is a flag and forces a redirect to the browser informing the browser to use the alias.

Other flags are supported by `RewriteRule`. For a list of flags, plus more information on URL rewriting in general, go to `http://httpd.apache.org/docs/current/mod/mod_rewrite.html`.

The pattern specified on the rewrite rule is matched according to the following contexts:

- If the rewrite rule appears inside a Virtual Host container, the pattern will be matched against the URL after the host name and port and prior to a query string (if present).
- If the rewrite rule appears inside a Directory container (or in `.htaccess`), the pattern will be matched against the file system after removing the prefix that led the server to the rewrite rule.
- To match against the host name, port, or query string, the `RewriteCond` directive should be used along with the variables ${HTTP_HOST}, ${SERVER_PORT}, or ${QUERY_STRING}.

For Directory context (including `.htaccess` files), additional rules apply:

- Turn the rewrite engine on.
- Set `Options FollowSymLinks`.
- For `.htaccess` files, the per-directory prefix is automatically added for any relative substitution.
- To match against the full URL, use ${REQUEST_URI} in `RewriteCond`.
- A match occurs against a string without leading slash because the removed prefix always contains a trailing slash.

Rewrite rules can be syntactically placed in Location containers but this is not supported by Apache.

Exercise 9: Using Alias, Redirect and mod_rewrite for different Virtual Hosts

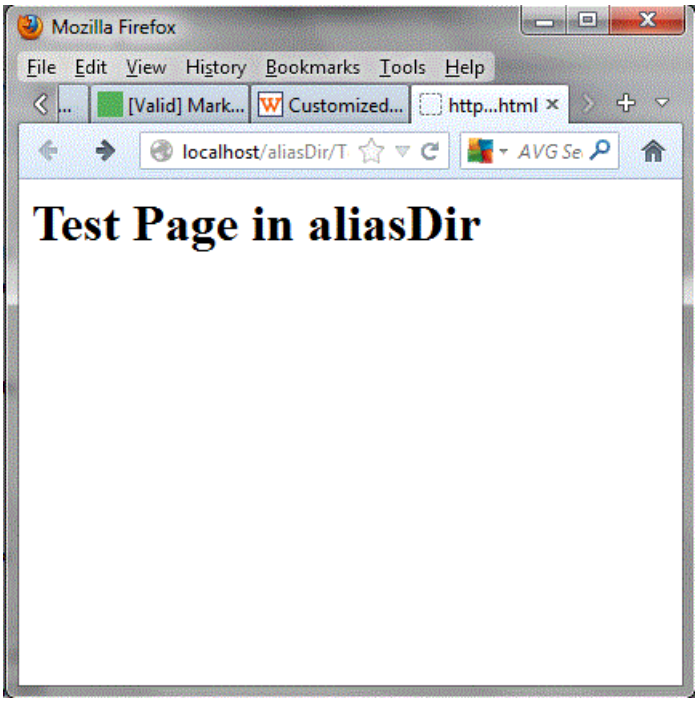
30 to 45 minutes

In this exercise, you will implement an alias that is global to the virtual hosts. You will define a redirect in the scope of one virtual host and a URL rewrite in the scope of a different virtual host.

1. Stop the Apache web server.
2. Create a folder under the the root directory and name the directory "aliasDirectory".
3. A very simple web page is provided in `using-aliases-redirecting/Exercises/TestPage.html`. Copy this file under `aliasDirectory`.
4. Open `httpd.conf` for edit.
5. Locate the Directory container for `/var/www/html`. Add the following statements after this container:

```
<Directory /aliasDirectory>
  Require all granted
</Directory>
Alias /aliasDir /aliasDirectory
```

6. Save your changes. Leave the edit session open as you will be making further modifications to the configuration file.
7. Start the Apache web server.
8. In your browser, enter the following address: `http://localhost/aliasDir/TestPage.html`
9. The test web page under `aliasDir` should be displayed:



10. The same result should be observed for the following address: `http://www.apacheadmin.com/aliasDir/TestPage.html` and this address:`http://www.apacheadmin2.com/aliasDir/TestPage.html`

11. Return to the editing session for `/etc/httpd/conf/httpd.conf`. Comment out the "Alias" directive for `/aliasDir`. Save your changes.
12. Add an Alias directive in the localhost Virtual Host container as soon below

```
<VirtualHost *:80>
  ServerName localhost
  DocumentRoot /var/www/html
  ErrorLog logs/Error.Log
  Alias /aliasDir /aliasDirectory
</VirtualHost>
```

13. Save your changes and leave the edit session open.
14. Restart the Apache server.
15. In your browser, enter the following address: `http://localhost/aliasDir/TestPage.html`
16. The test web page under `aliasDir` should be displayed as before.
17. Enter the following address: `http://www.apacheadmin.com/aliasDir/TestPage.html`

18. Question 1: What do you observe? Why?
19. Enter the following address: `http://www.apacheadmin2.com/aliasDir/TestPage.html`
20. Question 2: What do you observe? Why?
21. Return to the editing session. Add the following `Redirect` directive to the `www.apacheadmin.com` container:

```
<VirtualHost *:80>
  ServerName www.ApacheAdmin.com
  DocumentRoot /var/www/apacheadmin
  Redirect /search http://www.google.com
</VirtualHost>
```

22. Save your changes and leave the edit session open.
23. Restart the Apache service.
24. In your browser, enter the following address: `http://www.apacheadmin.com/search`
25. Question 3: What do you observe? Why?
26. In your browser enter the following address: `http://localhost/search`
27. Question 4: What do you observe? Why?
28. We wish to use the alias defined in the "localhost" virtual host from the "www.apacheadmin2.com" virtual host. To accomplish this task, return to the editing session of `/etc/httpd/conf/httpd.conf` and add the `RewriteEngine` and `RewriteRule` directives:

```
<VirtualHost *:80>
  ServerName www.ApacheAdmin2.com
  DocumentRoot /var/www/apacheadmin2
  RewriteEngine on
  RewriteRule /aliasDir(.*) http://localhost:80/aliasDir$1 [R]
</VirtualHost>
```

29. Save your changes and then restart Apache.
30. Enter the following address: `http://www.apacheadmin2.com/aliasDir/TestPage.html`
31. Question 5: What do you observe? Why?

32. We now wish to use the alias defined in the "localhost" virtual host from the "www.apacheadmin.com" virtual host. Add the following lines to the "www.apacheadmin.com" Virtual Host container:

```
RewriteEngine on
RewriteRule /aliasDir(.*) "http://localhost:80/aliasDir$1" [R]
```

33. Save your changes. Restart Apache.
34. In your browser, enter the following address: `http://www.apacheadmin.com/aliasDir/TestPage.html`
35. The test web page in `aliasDir` should be displayed.

Conclusion

In this lesson, you have learned:

- About configuring an alias for a URL.
- How to redirect requests.
- How to use mod_rewrite.

LESSON 10
Performance Considerations

Topics Covered

☑ Adjusting the `apache2.conf` file.

☑ DNS name lookup overhead.

☑ Log file I/O overhead.

☑ Checking web applications for performance issues.

☑ Network issues.

Introduction

The goal of performance tuning is to reduce response time while not sacrificing accuracy. Improved response is conducive to meeting company sales goals, achieving higher customer ratings, and using computer resources more efficiently .

In this lesson, you will learn what areas to check in Apache for potential performance warning signs.

10.1. Adjusting httpd.conf

The `httpd.conf` file can be potentially streamlined to help boost performance.

Modules that are not in use should be commented out to streamline startup processing and reduce memory consumption.

Unnecessary containers should be commented out or removed altogether. Directory and Location sections have to be scanned for every request received by Apache. Reducing the number of such containers will reduce the time Apache spends processing inbound traffic.

10.2. DNS Name Lookup

The `HostNameLookups` directive should be set to "off". This is the default value although the configuration file as shipped sets this directive to "off".

10.3. Logging I/O

Logging I/O can slow down the Apache server. This is especially problematic with multiple virtual hosts, each with its own access logging overhead. Every request for each host is duly written to a log file if the logging strategy of the factory configuration is maintained.

Of course, the log record is potentially quite useful and therefore logging must be implemented. However, the amount of data logged can be controlled.

The `SetEnvIf` directive can be employed to potentially reduce the I/O applicable to access logging. This directive can be used to set an environment variable that can potentially reduce the access log volume. You will have the opportunity to use the `SetEnvIf` directive in the lesson exercise.

The amount of data written to the error log can be controlled using the `LogLevel` directive. This directive tells Apache what logging level is to be used in error logging.

The log levels are presented below:

Level	Meaning
emerg	Emergency - Apache is unstable
alert	Alert - action is required
crit	Critical - action should be taken
error	Error - action indicated at the application level
warn	Warning - action may be indicated
notice	Normal
info	Informational
debug	Debug
trace	Trace

The levels are listed in order of decreasing severity. The level specified on `LogLevel` results in messages at that level and higher to be written to the error log. Accordingly, if you raise the level (e.g., from

"warn" to "alert"), then fewer messages in general will be written to the log and thus reduce I/O. However, in the event of a problem you will have less logging data at your disposal.

10.4. Web Applications

Web applications can contribute to degraded performance and slower response times. As an administrator, you can assist developers in reviewing web apps and focusing on these potential problem areas:

- Database operations
- Logging
- Design of the web programs

10.5. Network Issues

All of our efforts are in vain if the network is not operating efficiently. When performance issues (e.g., sluggish response) is reported to you, you should first rule out network difficulties by contacting your network control operations specialist.

Exercise 10: Tuning the Access Log

30 to 45 minutes

In this exercise, you will implement a strategy to tune the access log by decreasing the amount of request traffic that is logged.

1. Stop the Apache Web Server.
2. Open `httpd.conf` in a text editor. Find the `CustomLog` directive to determine the location of the access log.
3. Using operating system commands, delete the access log(s).
4. A web page is provided in `performance-considerations/Exercises/ImagePage.html`. Copy this file under `/var/www/html`.
5. An image file is provided in `performance-considerations/Exercises/apache_logo.jpg`. Copy this file under `/var/www/html`.
6. Start the Apache Web Server.
7. In your browser, go to the following address: `http://localhost/ImagePage.html`
8. The web page will be displayed with an image.
9. Open the access log in a text editor. You should observe two entries in the log applicable to the request you just entered: one for the web page and one for the image file. Close the log file.
10. Stop the Apache server.
11. Return to the edit session on `httpd.conf`.
12. Prior to the `CustomLog` directive, insert the following line:

    ```
    SetEnvIf Request_URI "\.(jpg|png|gif)$" Bypass
    ```

    This directive creates an environment variable if the condition is true. In this case, a variable named `Bypass` will be created if the extension of the requested file ends with one of the listed values (e.g., `png`).

13. On the `CustomLog` directive, add the `Env` parameter:

    ```
    CustomLog "access.log" combined Env=!Bypass
    ```

This will cause the `CustomLog` directive to be bypassed for this request (preventing a log record from being written to the file) if the `Bypass` variable has been defined (i.e., an image file has been requested).

14. Save your changes.
15. Using operating system commands, delete the access log.
16. Start Apache.
17. In your browser, go to the following address: `http://localhost/ImagePage.html`
18. The web page will be displayed with an image.
19. Open the access log in a text editor. You should observe only one entry in the log for the web page. The image file request was not logged.
20. Close the log file.

Conclusion

In this lesson, you have learned:

- How to adjust the httpd.conf file.
- About DNS name lookup overhead.
- About Log file I/O overhead.
- How to check web applications for performance issues.
- About network issues.

LESSON 11

Customizing Request/Response Processing

Topics Covered

☑ Handlers and requests.

☑ Built-in handlers.

☑ `AddHandler`, `SetHandler`, and `RemoveHandler` directives.

☑ Filters and filter directives.

Introduction

A handler is a software component that implements the action to be performed when a file is called. Handlers may be built into the server, added as modules or specified using an `Action` directive.

11.1. Handlers and Requests

Handlers provide custom processing for files when they are requested by a client. The core module provides a default handler for static content. In most cases, the file is sent to the client without special handling.

A CGI script, on the other hand, is not intended to be served as static content. The `cgi-script` handler, stored in the module `mod_cgi`, can be specified as a handler for files with an extension of "cgi". Presumably, the file contains a script.

11.2. Built-in Handlers

The following handlers are built into Apache:

Handler	Module	Purpose
default-handler	core	Serve static files by default
send-as-is	mod_asis	Send a file with HTTP headers as is
cgi-script	mod_cgi	Process the file as a CGI script
imap-file	mod_imagemap	Process the file as an image map rule file
server-info	mod_info	Send server configuration data
server-status	mod_status	Send server status data
type-map	mod_negotiation	Parse the file as a type map file for content negotiation

The `server-info` and `server-status` handlers are not associated with file types. To invoke one of these handlers, we could use a Location container. For example:


```
<Location /info>
  SetHandler server-info
</Location>
```

The `mod_info` module must be uncommented in order to use the handler.

To test the handler, enter the following address into your browser:

```
http://localhost/info
```

The server information web page will be displayed.

11.3. Handler Directives

The core module provides directives to set, add, and remove a handler.

❖ 11.3.1. Location Directive

The `Location` directive maps a URL to one more directives. In the previous section, the `Location` directive mapped a URL to the `server-info` handler. The `LocationMatch` directive is identical to `Location` except the location, or URL, can be specified using a regular expression.

❖ 11.3.2. Files Directive

The `Files` container is used in conjunction with the handler directives when the handler applies to a file or to files with certain extensions. The `Files` directive applies directives to one or more files specified in the container.

Regular expressions can be used when specifying the file name. The `FilesMatch` directive also permits regular expressions and therefore is functionally equivalent to `Files`.

❖ 11.3.3. SetHandler

The `SetHandler` directive associates a handler with a file type. It can also associate a location with a handler as the earlier example on `server-info` demonstrated.

The following example shows how you can associate a handler with a file type using `Files`:

```
<Files *.html>
  SetHandler type-map
</Files>
```

A request for an HTML document will be handled by `type-map`. In the lesson exercise, you will use the type map handler to serve content written in different languages (e.g., French) according to browser preference.

❖ 11.3.4. AddHandler

The `AddHandler` directive adds a handler with a file extension.

❖ 11.3.5. RemoveHandler

The `RemoveHandler` directive removes the association of a handler from a file extension. This directive can be applied to a subdirectory that might have inherited a handler from a higher level directory.

11.4. Filters

Filters are programs that can process input and output data. Whereas handlers are applied to files that are requested by a client, filters can be specified to handle request and response data.

Apache provides several filters including `mod_include` that implements server-side includes.

The following directives are supported by Apache (module name is indicated in parentheses):

Directive	Purpose
AddInputFilter	Process requests and HTTP POST data based on file extension (mod_mine)
AddOutputFilter	Process responses based on file extension (mod_mine)
RemoveInputFilter	Remove filter that processes requests and HTTP POST data based on file extension (mod_mine)
RemoveOutputFilter	Remove filter that processes responses based on file extension (mod_mine)
SetInputFilter	Process requests and HTTP POST data (core)
SetOutputFilter	Process responses (core)

Exercise 11: Using the type-map Handler

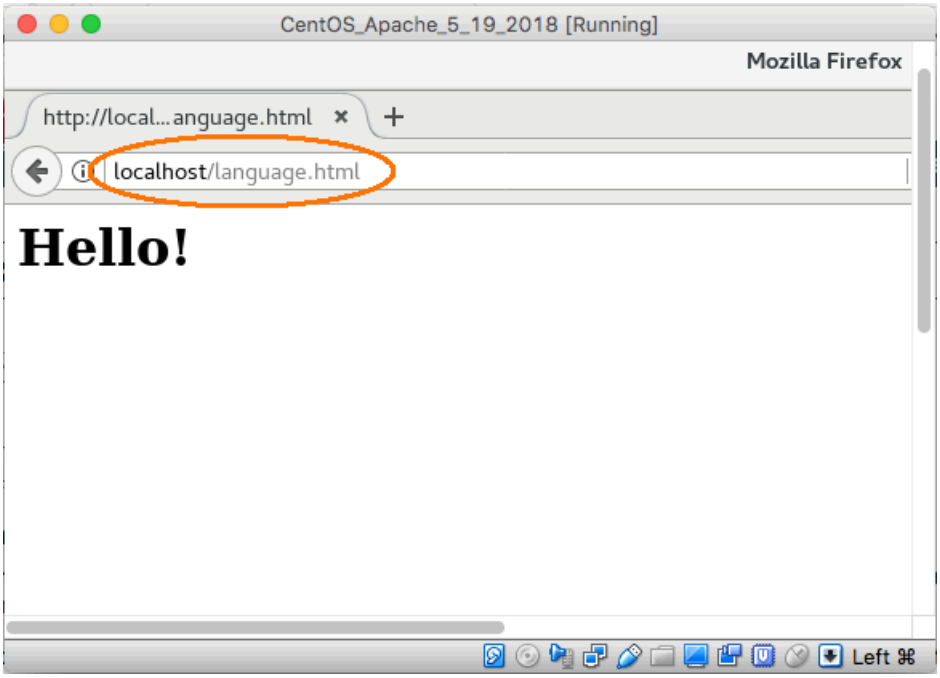
10 to 15 minutes

In this exercise, you will use the "type-map" handler to detect browser language preference.

1. Stop the Apache Web Server.
2. Copy "language.html" in the `Exercises` folder to `/var/www/html`.
3. Open `httpd.conf` in the text editor.
4. Locate the VirtualHost directive for localhost. Add a Files container as shown below: container:

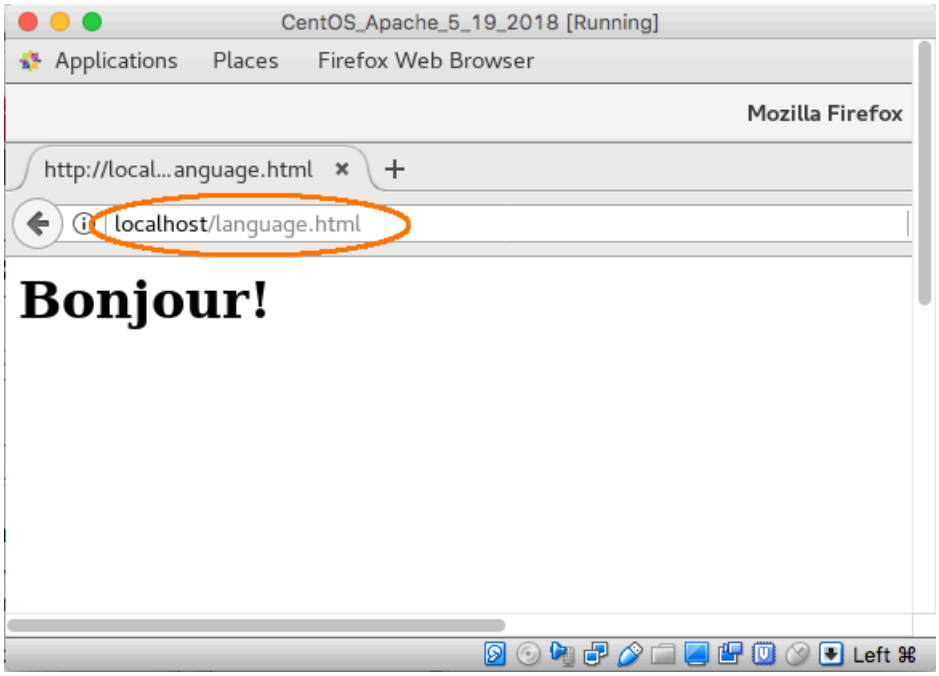
```
<VirtualHost *:80>
    ServerName localhost
    DocumentRoot /var/www/html
    ErrorLog logs/Error.Log
    <Files "language.html">
      SetHandler type-map
    </Files>
</VirtualHost>
```


5. Save your changes.
6. Start the Apache server.
7. In your browser, type the following address: `http://localhost/language.html`
8. If the browser language is "en" (English), then the following web page will be displayed:



9. Change the browser language preference to "fr" (French). For example, in FireFox select **Edit > Preferences** . Choose the **Content** tab and then click **Choose...**. Remove any existing language preference. From **Select a language to add...** scroll to **French [fr]** and select this entry. Then click **Add**.

10. Reload the web page. The following web page will be displayed:



11. Change the browser language preference to “it” (Italian). Reload the web page. The “Italian” greeting will be displayed.

Conclusion

In this lesson, you have learned:

- About handlers and requests.
- How to use built-in handlers.
- About AddHandler, SetHandler, and RemoveHandler directives.
- About filters and filter directives.

LESSON 12
PHP

Topics Covered

☑ Downloading and installing PHP.

☑ A basic PHP web page.

☑ MySQL with PHP.

☑ WordPress.

Introduction

PHP is the most popular technology for setting and running websites on Apache. In this lesson, you will learn how to download and install PHP on Apache.

PHP is often used with the MySQL relational database management system. This lesson will address the installation of MySQL and configuring MySQL for use with PHP.

12.1. PHP

PHP was developed by Rasmus Lerdorf in 1994 as CGI application written in C. Originally, the acronym stood for Personal Home Pages. Lerdorf turned the source code over to public domain the following year and in 1998 PHP (PHP: Hypertext Preprocessor) 3.0 was released.

The current version is PHP 5. PHP offers a procedural language that is coded in-line with HTML tags. The PHP "pages" are interpreted at runtime by the "Zend" engine.

12.2. Installation

For CentOS you can install PHP and MySQL using the following `yum` command:

```
sudo yum install php php-mysql
```

Restart the Apache server so that your modifications will be picked up by Apache.

❖ 12.2.1. Apache Configuration

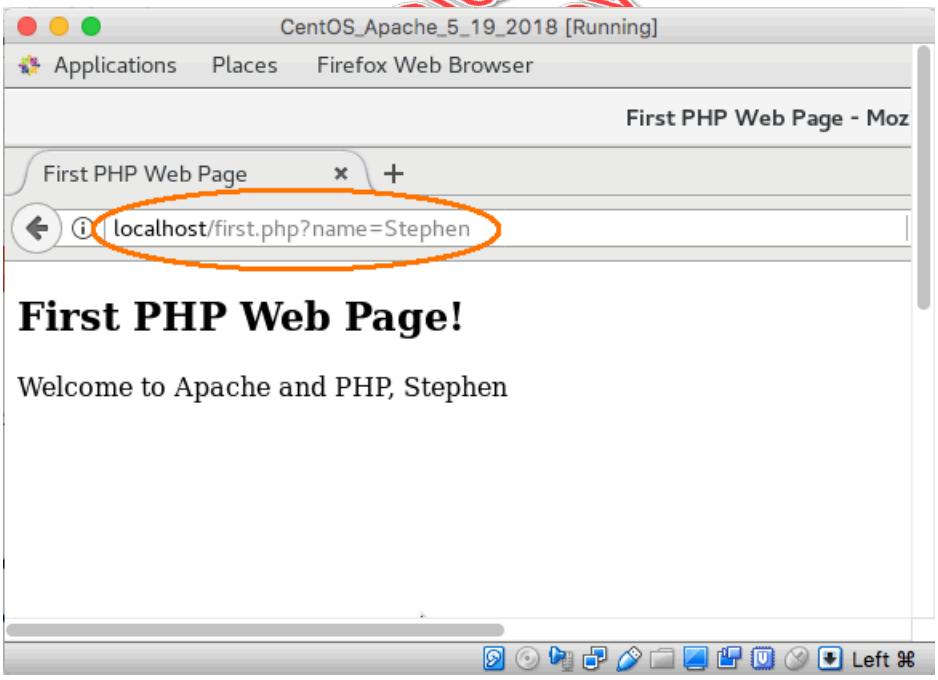
The PHP load module configuration is stored in `/etc/httpd/conf.modules.d/10-php.conf`. The mapping of "php" file extensions to the PHP interpreter is located in `/etc/httpd/conf.d/php.conf`.

12.3. Writing a Basic PHP Web Page

A simple PHP web page is provided in `php/Demos/first.php`. Copy this web page to `/var/www/html`.

In your browser, go to `http://localhost/first.php?name=your_name`.

The web page will echo back the user name:



Here is the source code for the web page:

```
<html>
<body>
<title>First PHP Web Page</title>
</body>
<h2>First PHP Web Page!</h2>
<?php
if (!isset($_GET['name'])) {
  echo "Please enter a user name";
  return;
}
$userName=$_GET['name'];
  echo "Welcome to Apache and PHP, ".$userName;
?>
</html>
```

The PHP programming statements are contained with the `<?php>` and `<?>` tags.

The `if` statement tests the return value of the `isset` function. This function returns "true" if the variable passed as an argument has been set (i.e., a value has been established for the variable). The `!` is the logical "not" operator. Therefore, if the GET parameter has not been set, the statements within the scope of the `if` statement will be executed.

The curly braces ("{", "}") are used to indicate scope on the `if` statement and on looping statements.

The parameters passed on an HTTP GET request are available in the `$_GET` array. The "$" indicates this a variable name, in this case an array. An "associative" index (i.e., the name of the parameter) can be used to retrieve the parameter value.

The variable `$username` is assigned the value of the GET parameter `name`. The `echo` statement writes content to the output HTML file. The "." is the concatenation operator. Therefore, the displayed text is a concatenation of static text and the contents of `$username`.

12.4. Using MySQL with Apache and PHP

MySQL is a popular database management system and is commonly used with Apache and PHP. In fact, "stacks" such as XAMPP (cross platform Apache MySQL PHP Perl) offer Apache, MySQL, and PHP in the same package.

MySQL is available for the major Linux distributions.

12.5. WordPress

WordPress is a content management system that permits you to design your own website or blog (web log). WordPress is free and is written in PHP. WordPress requires a MySQL database to store blog data.

For more information, go to `www.WordPress.org (https://www.wordpress.org)`.

Exercise 12: Installing a PHP/MySQL Website

30 to 45 minutes

In this exercise, you will install a simple guest book website. You will create the database table in MySQL.

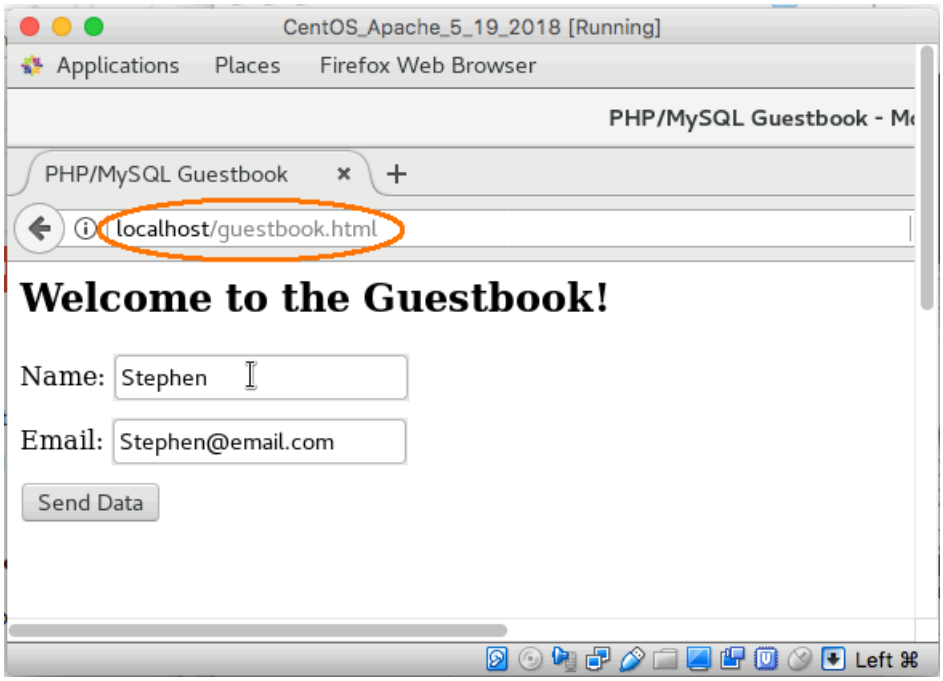
1. A MySQL script file is provided in `php/Exercises/GuestBook.sql`. Copy this file to a folder in your file system (e.g., `MySQL`).
2. Start a terminal window and type:

   ```
   mysql -u root -p test
   ```

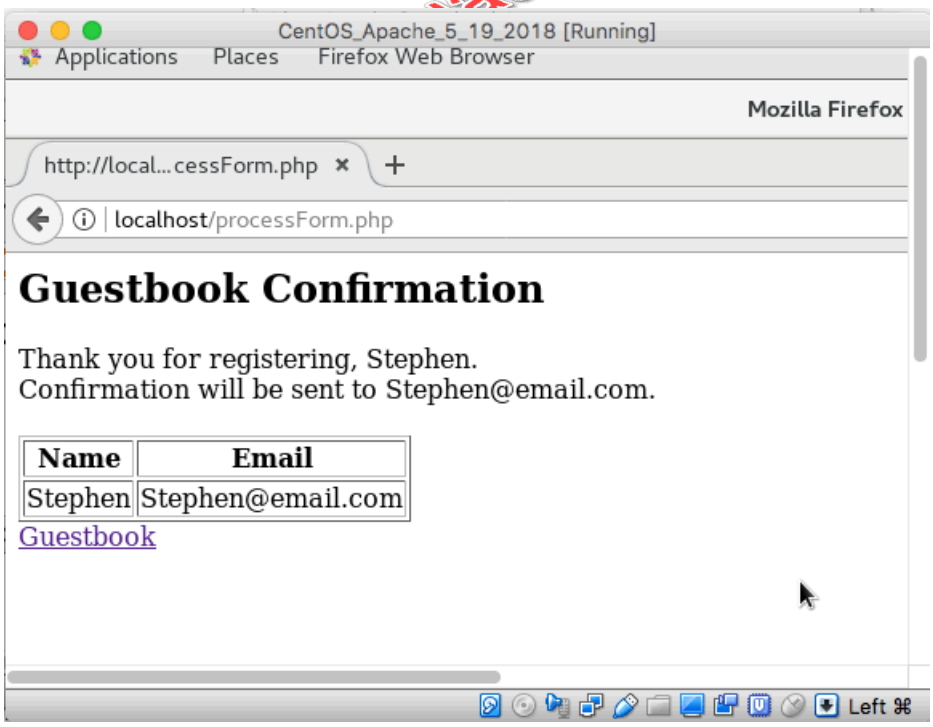
   Hit the **ENTER** key.
3. You will be prompted for the root account password. Type `password` and then hit **Enter**.
4. Type the following in the command prompt and then hit **Enter** replacing "path to script" with path to the script contained in the "Exercises" folder for this lesson:

   ```
   source path to script/GuestBook.sql
   ```

5. Two results should be displayed, both with "Query OK".
6. A web page is provided in `php/Exercises/guestbook.html`. Copy this file to `/var/www/html`.
7. A PHP web page is provided in `php/Exercises/processForm`. Copy this file to `/var/www/html`.
8. Start Apache.
9. In your browser, go to `http://localhost/guestbook.html`
10. Enter a name and an email address:



11. Click the **Send Data** button. A confirmation page is displayed:



The guest book entries are displayed to verify each guest has been written to the database.

Conclusion

In this lesson, you have learned:

- About PHP.
- How to download and install PHP.
- How to write a basic PHP program.
- How to use MySQL with PHP.
- Considerations for WordPress Deployment.

LESSON 13

Mod Proxy and Mod Proxy Balancer

Topics Covered

☑ The `mod_proxy` module.

☑ The `mod_proxy` module.

☑ The `mod_proxy_balancer` module.

☑ The `mod_proxy_balancer` module.

Introduction

The `mod_proxy` module enables you to use Apache as a proxy server. A proxy server supports requests from clients that are ultimately bound for other servers (e.g., Tomcat). The purpose of a proxy server is to provide a single point of access for purposes of convenience, security, high availability and load balancing.

Apache can function as a "forward proxy" or as a "reverse proxy" server. A forward proxy server gives the client the ability to use Apache as a local server to access remote websites whose addresses are known to the client. A reverse proxy server establishes a mapping of a path to a remote server whose address is not known to the client.

13.1. Apache as a Proxy Server

❖ 13.1.1. Installing and Enabling Required Modules

Apache is shipped with the `mod_proxy` module and other supporting modules. The load module references are stored in `/etc/httpd/conf.modules.d/00-proxy.conf`.

❖ 13.1.2. Using mod_proxy

The `mod_proxy` module supports directives that implement proxy functionality in Apache. This module is located in `mod_proxy.so`.

The `mod_proxy` module supports several protocols that enable Apache to talk to back-end servers. For example, HTTP communication is used by Apache to talk to any HTTP server (e.g., Tomcat, IIS, Websphere). AJP (Apache JServ Protocol) can be used to communicate with a Tomcat cluster. For a complete list of supported protocols see `http://httpd.apache.org/docs/2.4/mod/mod_proxy.html`.

The primary directive in this module is `ProxyPass`. This directive permits you to map a path to a remote server web resource address (URL). The `Proxy` container gives you the ability to specify directives for a URL defined in a `ProxyPass` directive such as security and load balancer requirements.

❖ 13.1.3. Forward vs. Reverse Proxy

Apache can serve as a forward proxy or as a reverse proxy.

To use Apache as a forward proxy, add the `ProxyRequests on` directive to your Apache configuration (this directive can also be coded in a virtual host container). You should also secure your server because a client can now anonymously access potentially any website. When using Apache as a forward proxy server you can secure Apache by using the `Require` directive in a `Proxy` container as illustrated below:


```
ProxyRequests on
    # www.safehost.com, www.goodguys.com are hypothetical trusted domains, domains that
 contain "phishers" are untrusted
    <Proxy "*">
        Require host www.safehost.com www.goodguys.com
        Require not host phishers
    </Proxy>
```

To use Apache as a reverse proxy, add one or more `ProxyPass` and `ProxyPassReverse` directives. A client using a reverse proxy is unaware of the back-end, or remote, server address. Apache maps a path, known to the client, to the remote site with the `ProxyPass` directive. For example, we wish to map "/tomcat" to the Tomcat home page, "http://localhost:8080":

```
    ProxyPass /tomcat http://localhost:8080
```

The path (e.g., "/tomcat") is in effect a mirror of the remote site (e.g., "http://localhost:8080").

Note that the `ProxyPass` directive does not require `ProxyRequests on`.

The `ProxyPass` can also be specified in a `Location` container. In this case, the mapped path is coded on the location directive and is omitted on `ProxyPass`:

```
<Location /tomcat>
  ProxyPass http://localhost:8080
  Require host www.OurTomcatUsers.com
</Location>
```

If the back-end server sends a redirect request to the client, then the URL should be rewritten to the "local" URL. In this way, the browser sends a request that does not bypass the reverse proxy. To accomplish this task use the `ProxyPassReverse` directive in conjunction with `ProxyPass`:

```
ProxyPass /tomcat http://localhost:8080
ProxyPassReverse /tomcat http://localhost:8080
```


❖ 13.1.4. mod_proxy_balancer

Apache can also serve as a load balancer-a piece of software that directs traffic across a network of two or more servers. The routing of traffic can be based on server busyness (number of requests a server is assigned), pending requests or traffic (overall number of bytes a server is expected to handle).

In order to use the directives for load balancing, the `mod_proxy_balancer` module must be enabled in addition to `mod_proxy`.

A **virtual worker** must be defined to direct the request to a **real worker**. The real workers are specified as a list of `ProxyPass` directives contained within a `Proxy` container.

Consider the following example:

```
ProxyPass / balancer://myBalancer/ stickysession=JSESSIONID
<Proxy balancer://myBalancer >
  BalancerMember http://localhost:8080 loadfactor=10
  BalancerMember http://localhost:8081 loadfactor=5
  ProxySet lbmethod=byrequests
</Proxy>
```

In the example above, the `ProxyPass` directive associates the path "/" with a virtual worker. Note that the worker name is prefixed with "balancer"; this label indicates to Apache that a virtual worker is now defined for load balancing. The text following "balancer" is arbitrary. The "stickysession" attribute specifies that a session should be handled by the same real worker based on the JSESSIONID cookie name. This value is used for Tomcat back-end servers. Tomcat assigns the value of the session id to this cookie as well as the JVM route that has been assigned on the `Engine` element in the Tomcat `server.xml` configuration file.

The `Proxy` container defines the real workers. Each worker is associated with a back-end server, in this case a Tomcat cluster. Apache talks to the cluster using the AJP protocol. Note that each worker must listen on a unique AJP port.

The `loadfactor` weights each server for workload. The numeric values chosen are important in a relative sense. In the example above, a load factor of "10" indicates that the load balancing goal is to assign twice as much work to this worker as compared to the worker with an assigned load factor of "5". This attribute is meaningful if the `lbmethod` is assigned the value of `byrequests` or `bytraffic`.

The default value of `lbmethod` is `byrequests`. This value instructs Apache to balance the load based on the number of requests each server is currently processing. Assigning the load balancer the method of `bytraffic` is similar, except that overall byte count of the requests is also taken into consideration. The load balance method can also be specified as `bybusyness` which balances the load based on the number of requests queued for each server.

For a detailed discussion of `mod_proxy_balancer` go to `http://httpd.apache.org/docs/2.4/mod/mod_proxy_balancer.html`.

Exercise 13: Testing mod_proxy and mod_proxy_balancer

15 to 25 minutes

In this exercise, you will test mod_proxy and mod_proxy_balancer.

1. Stop the Apache server.
2. Open `/etc/httpd/conf/httpd.conf` for edit. Add the following line of code to the file:

```
ProxyPass /google http://www.google.com
```

3. Save your changes. Leave the edit session open as you will make additional changes to the configuration file later in the assignment.
4. Start the Apache server.
5. Point your browser to `http://localhost/google`. You should see the Google home page.
6. Return to the edit session on `httpd.conf`. On the next line after the `ProxyPass` statement you coded earlier add another `ProxyPass` statement as follows:

```
ProxyPass /tomcat http://localhost:8080
```

7. Save your changes. Leave the edit session open as you will make additional changes to the configuration file later in the assignment.
8. Restart the Apache server.
9. Start the Tomcat server by changing directory to `/opt/tomcat9/bin` and entering the following command:

```
sudo ./startup.sh
```

10. Point your browser to `http://localhost/tomcat`. You should see the Tomcat home page.

11. Return to the edit session on `/etc/httpd/conf/httpd.conf`. Add another `ProxyPass` statement followed by a Proxy container. The previous `ProxyPass` statement you coded earlier is presented below followed by the `ProxyPass` and `Proxy` container you are to add:

```
ProxyPass /tomcat http://localhost:8080
ProxyPass /ReplicateDemo/ balancer://mybalancer.com/
<Proxy balancer://mybalancer.com/ >
  BalancerMember http://localhost:8080/ReplicateDemo/
  BalancerMember http://localhost:8081/ReplicateDemo/
</Proxy>
```

Note that only the first Tomcat worker is active.

12. Save your changes and then restart the Apache server.
13. In your browser, go to `http://localhost/ReplicateDemo/FirstServlet?name=Paul`. You should see a web page confirming that the name you entered has been stored as a session attribute.

Conclusion

In this lesson, you have learned

- About the mod_proxy module.
- How to use the mod_proxy module.
- About the mod_proxy_balancer module.
- How to use the mod_proxy_balancer module.