Installing the Apache Web Server on Linux (CentOS 7)

Introduction

The Apache HTTP server is the most widely-used web server in the world. It provides many powerful features including dynamically loadable modules, robust media support, and extensive integration with other popular software.

In this guide, you will install an Apache web server with virtual hosts on your CentOS 7 server.

Prerequisites

You will need the following to complete this guide:

- A non-root user with sudo privileges configured on your server, set up by following the initial server setup guide for CentOS 7.
- A basic firewall configured by following the Additional Recommended Steps for New CentOS 7 Servers guide.

Step 1 — Installing Apache

Apache is available within CentOS's default software repositories, which means you can install it with the `yum` package manager.

As the non-root sudo user configured in the prerequisites, update the local Apache `httpd` package index to reflect the latest upstream changes:

```
sudo yum update httpd
```

Once the packages are updated, install the Apache package:

```
sudo yum install httpd
```

After confirming the installation, `yum` will install Apache and all required dependencies. Once the installation completes, you are ready to start the service.

Md. Ariful Islam
Manager (Data & Transmission Network), BdREN
E-mail: arif@bdren.net.bd

Step 2 — Checking your Web Server

Apache does not automatically start on CentOS once the installation completes. You will need to start the Apache process manually:

```
sudo systemctl start httpd
```

Verify that the service is running with the following command:

```
sudo systemctl status httpd
```

You will see an `active` status when the service is running:

```
Output
Redirecting to /bin/systemctl status httpd.service
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled;
vendor preset: disabled)
   Active: active (running) since Wed 2019-02-20 01:29:08 UTC; 5s ago
     Docs: man:httpd(8)
           man:apachectl(8)
 Main PID: 1290 (httpd)
   Status: "Processing requests..."
   CGroup: /system.slice/httpd.service
           ├─1290 /usr/sbin/httpd -DFOREGROUND
           ├─1291 /usr/sbin/httpd -DFOREGROUND
           ├─1292 /usr/sbin/httpd -DFOREGROUND
           ├─1293 /usr/sbin/httpd -DFOREGROUND
           ├─1294 /usr/sbin/httpd -DFOREGROUND
           └─1295 /usr/sbin/httpd -DFOREGROUND
...
```

As you can see from this output, the service appears to have started successfully. However, the best way to test this is to request a page from Apache.

You can access the default Apache landing page to confirm that the software is running properly through your IP address. If you do not know your server's IP address, you can get it a few different ways from the command line.

Md. Ariful Islam
Manager (Data & Transmission Network), BdREN
E-mail: arif@bdren.net.bd

Type this at your server's command prompt:

```
hostname -I
```

This command will display all of the host's network addresses, so you will get back a few IP addresses separated by spaces. You can try each in your web browser to see if they work.

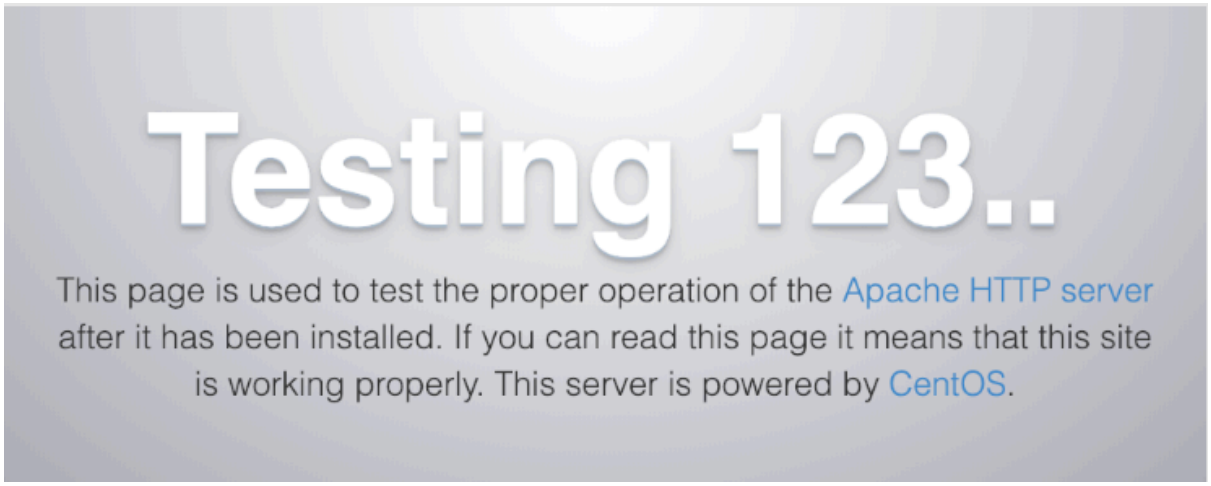
Alternatively, you can use `curl` to request your IP from `icanhazip.com`, which will give you your public IPv4 address as seen from another location on the internet:

```
curl -4 icanhazip.com
```

When you have your server's IP address, enter it into your browser's address bar:

```
http://your_server_ip
```

You'll see the default CentOS 7 Apache web page:



This page indicates that Apache is working correctly. It also includes some basic information about important Apache files and directory locations. Now that the service is installed and running, you can now use different `systemctl` commands to manage the service.

Step 3 — Managing the Apache Process

Now that you have your web server up and running, let's go over some basic management commands.

Md. Ariful Islam
Manager (Data & Transmission Network), BdREN
E-mail: arif@bdren.net.bd

To stop your web server, type:

```
sudo systemctl stop httpd
```

To start the web server when it is stopped, type:

```
sudo systemctl start httpd
```

To stop and then start the service again, type:

```
sudo systemctl restart httpd
```

If you are simply making configuration changes, Apache can often reload without dropping connections. To do this, use this command:

```
sudo systemctl reload httpd
```

By default, Apache is configured to start automatically when the server boots. If this is not what you want, disable this behavior by typing:

```
sudo systemctl disable httpd
```

To re-enable the service to start up at boot, type:

```
sudo systemctl enable httpd
```

Apache will now start automatically when the server boots again.

The default configuration for Apache will allow your server to host a single website. If you plan on hosting multiple domains on your server, you will need to configure virtual hosts on your Apache web server.

Step 4 — Setting Up Virtual Hosts (Recommended)

When using the Apache web server, you can use *virtual hosts* to encapsulate configuration details and host more than one domain from a single server. In this step, you will set up a domain called `example.com`, but you should replace this with your own domain name.

Md. Ariful Islam
Manager (Data & Transmission Network), BdREN
E-mail: arif@bdren.net.bd

Apache on CentOS 7 has one server block enabled by default that is configured to serve documents from the `/var/www/html` directory. While this works well for a single site, it can become unwieldy if you are hosting multiple sites. Instead of modifying `/var/www/html`, you will create a directory structure

within `/var/www` for the `example.com` site, leaving `/var/www/html` in place as the default directory to be served if a client request doesn't match any other sites.

Create the `html` directory for `example.com` as follows, using the `-p` flag to create any necessary parent directories:

```
sudo mkdir -p /var/www/example.com/html
```

Create an additional directory to store log files for the site:

```
sudo mkdir -p /var/www/example.com/log
```

Next, assign ownership of the `html` directory with the `$USER` environmental variable:

```
sudo chown -R $USER:$USER /var/www/example.com/html
```

Make sure that your web root has the default permissions set:

```
sudo chmod -R 755 /var/www
```

Next, create a sample `index.html` page using `vi` or your favorite editor:

```
sudo vi /var/www/example.com/html/index.html
```

Press `i` to switch to `INSERT` mode and add the following sample HTML to the file:

/var/www/example.com/html/index.html

```
<html>
  <head>
    <title>Welcome to Example.com!</title>
  </head>
  <body>
    <h1>Success! The example.com virtual host is working!</h1>
  </body>
</html>
```

Save and close the file by pressing `ESC`, typing `:wq`, and pressing `ENTER`.

Md. Ariful Islam
Manager (Data & Transmission Network), BdREN
E-mail: arif@bdren.net.bd

With your site directory and sample index file in place, you are almost ready to create the virtual host files. Virtual host files specify the configuration of your separate sites and tell the Apache web server how to respond to various domain requests.

Before you create your virtual hosts, you will need to create a `sites-available` directory to store them in. You will also create the `sites-enabled` directory that tells Apache that a virtual host is ready to serve to visitors. The `sites-enabled` directory will hold symbolic links to virtual hosts that we want to publish. Create both directories with the following command:

```
sudo mkdir /etc/httpd/sites-available /etc/httpd/sites-enabled
```

Next, you will tell Apache to look for virtual hosts in the `sites-enabled` directory. To accomplish this, edit Apache's main configuration file and add a line declaring an optional directory for additional configuration files:

```
sudo vi /etc/httpd/conf/httpd.conf
```

Add this line to the end of the file:

```
IncludeOptional sites-enabled/*.conf
```

Save and close the file when you are done adding that line. Now that you have your virtual host directories in place, you will create your virtual host file.

Start by creating a new file in the `sites-available` directory:

```
sudo vi /etc/httpd/sites-available/example.com.conf
```

Add in the following configuration block, and change the `example.com` domain to your domain name:

/etc/httpd/sites-available/example.com.conf

```
<VirtualHost *:80>
    ServerName www.example.com
    ServerAlias example.com
    DocumentRoot /var/www/example.com/html
    ErrorLog /var/www/example.com/log/error.log
    CustomLog /var/www/example.com/log/requests.log combined
</VirtualHost>
```

Md. Ariful Islam
Manager (Data & Transmission Network), BdREN
E-mail: arif@bdren.net.bd

This will tell Apache where to find the root directly that holds the publicly accessible web documents. It also tells Apache where to store error and request logs for this particular site.

Save and close the file when you are finished.

Now that you have created the virtual host files, you will enable them so that Apache knows to serve them to visitors. To do this, create a symbolic link for each virtual host in the `sites-enabled` directory:

```
sudo ln -s /etc/httpd/sites-available/example.com.conf
/etc/httpd/sites-enabled/example.com.conf
```

Your virtual host is now configured and ready to serve content. Before restarting the Apache service, let's make sure that SELinux has the correct policies in place for your virtual hosts.

Step 5 — Adjusting SELinux Permissions for Virtual Hosts (Recommended)

SELinux is configured to work with the default Apache configuration. Since you set up a custom log directory in the virtual hosts configuration file, you will receive an error if you attempt to start the Apache service. To resolve this, you need to update the SELinux policies to allow Apache to write to the necessary files. SELinux brings heightened security to your CentOS 7 environment, therefore it is not recommended to completely disable the kernel module.

There are different ways to set policies based on your environment's needs, as SELinux allows you to customize your security level. This step will cover two methods of adjusting Apache policies: universally and on a specific directory. Adjusting policies on directories is more secure, and is therefore the recommended approach.

Adjusting Apache Policies Universally

Setting the Apache policy universally will tell SELinux to treat all Apache processes identically by using the `httpd_unified` boolean. While this approach is more convenient, it will not give you the same level of control as an approach that focuses on a file or directory policy.

Run the following command to set a universal Apache policy:

```
sudo setsebool -P httpd_unified 1
```

The `setsebool` command changes SELinux boolean values. The `-P` flag will update the boot-time value, making this change persist across reboots. `httpd_unified` is the boolean that will tell SELinux to treat all Apache processes as the same type, so you enabled it with a value of `1`.

Adjusting Apache Policies on a Directory

Individually setting SELinux permission for the /var/www/example.com/log directory will give you more control over your Apache policies, but may also require more maintenance. Since this option is not universally setting policies, you will need to manually set the context type for any new log directories specified in your virtual host configurations. First, check the context type that SELinux gave the ***/var/www/example.com/log*** directory:

```
sudo ls -dZ /var/www/example.com/log/
```

This command lists and prints the SELinux context of the directory. You will see output similar to the following:

```
Output
drwxr-xr-x. root root unconfined_u:object_r:httpd_sys_content_t:s0
/var/www/example.com/log/
```

The current context is `httpd_sys_content_t`, which tells SELinux that the Apache process can only read files created in this directory. In this tutorial, you will change the context type of the `/var/www/example.com/log` directory to `httpd_log_t`. This type will allow Apache to generate and append to web application log files:

```
sudo semanage fcontext -a -t httpd_log_t "/var/www/example.com/log(/.*)?"
```

Next, use the `restorecon` command to apply these changes and have them persist across reboots:

```
sudo restorecon -R -v /var/www/example.com/log
```

The `-R` flag runs this command recursively, meaning it will update any existing files to use the new context. The `-v` flag will print the context changes the command made. You will see the following output confirming the changes:

```
Output
```

```
restorecon reset /var/www/example.com/log context
unconfined_u:object_r:httpd_sys_content_t:s0-
>unconfined_u:object_r:httpd_log_t:s0
```

You can list the contexts once more to see the changes:

```
sudo ls -dZ /var/www/example.com/log/
```

The output reflects the updated context type:

```
Output
drwxr-xr-x. root root unconfined_u:object_r:httpd_log_t:s0
/var/www/example.com/log
```

Now that the `/var/www/example.com/log` directory is using the `httpd_log_t` type, you are ready to test your virtual host configuration.

Step 6 — Testing the Virtual Host (Recommended)

Once the SELinux context has been updated with either method, Apache will be able to write to the `/var/www/example.com/log` directory. You can now successfully restart the Apache service:

```
sudo systemctl restart httpd
```

List the contents of the `/var/www/example.com/log` directory to see if Apache created the log files:

```
ls -lZ /var/www/example.com/log
```

You'll see that Apache was able to create the `error.log` and `requests.log` files specified in the virtual host configuration:

```
Output
-rw-r--r--. 1 root root 0 Feb 26 22:54 error.log
-rw-r--r--. 1 root root 0 Feb 26 22:54 requests.log
```

Now that you have your virtual host set up and SELinux permissions updated, Apache will now serve your domain name. You can test this by navigating to `http://example.com`, where you should see something like this:

This confirms that your virtual host is successfully configured and serving content. Repeat Steps 4 and 5 to create new virtual hosts with SELinux permissions for additional domains.

Conclusion

Now that you have your web server installed, you have many options for the type of content you can serve and the technologies you can use to create a richer experience.

Md. Ariful Islam
Manager (Data & Transmission Network), BdREN
E-mail: arif@bdren.net.bd