Google IT Support Certificate: Course 4: Systems Admin. and IT Infrastructure Services

Week 3: Software and Platform Services

Quiz: Manage Websites With Apache2

Learning Objectives

- Getting familiar with Apache2
- Configuring Apache2 on Linux
- Enabling and Disabling websites in Apache2

Introduction

Managing web servers is one of the most common tasks done by systems administrators. The web server could be hosting an informational website, a content management system, or even a full blown e-commerce site. Whatever the content is, it's important to understand how to manage the web server that is being used to serve those pages.

What you'll do

In this lab, you will install Apache2, a widely used web server software. You'll enable a site that is different from the default and enable additional features on the server.

Linux commands reminder

In this lab, we'll use a number of Linux commands that were already explained during Course 3. Here is a reminder of these commands and their actions:

- `sudo <command>`: executes a command with administrator rights
- `apt update`: updates the list of available packages to be installed
- `apt install package`: installs the given package in the system

- `ls <directory>`: lists the files in a directory
- `cp <old> <new>`: creates a copy of the old file with the new name
- `mv <old> <new>`: moves or renames the old file to the new name
- `nano <file>`: opens a text editor to edit the file
- `cat <file>`: outputs the whole contents of a file

We will also be using the `service` command shown in a previous lab, and we will present a number of new commands like `a2ensite` or `a2dismod`. Remember that you can always read the manual page using `man <command_name>` to learn more about a command.

While you can copy and paste the commands that are presented in this lab, we recommend typing them out manually, to help with understanding and remember them.

The scenario

A web designer in your company has developed an institutional site that will let customers learn more about the company. It's now your job to deploy this site and make it available to the world.

The virtual machine that you are going to use for this lab will represent the test instance where you will test the provided website, verify that it works correctly, figure out which steps to follow and determine what the configuration should look like. In a real-life scenario, after completing the steps in this lab you would apply the same configuration to the production machine.

The developed website is currently inside the `/opt/devel/ourcompany` directory. Let's look at the contents of this directory:

```
ls -l /opt/devel/ourcompany
```

Copied!

content_copy

```
total 20
-rwxr-xr-x 1 student student 561 Aug 11 12:07 aboutus.html
-rwxr-xr-x 1 student student 551 Aug 11 12:07 contact.html
-rwxr-xr-x 1 student student  95 Aug 11 12:07 footer.html
-rwxr-xr-x 1 student student 596 Aug 11 12:07 index.html
-rw-r--r-- 1 student student 777 Aug 11 12:07 style.css
```

So, we have a bunch of HTML pages and a CSS stylesheet that form the website that we want to serve. Let's use these and get on with it.

Installing Apache2

The machine that you are connected to does not yet have any web server running on it. Let's fix that by first updating the list of packages and then installing the `Apache2` package:

```
sudo apt update
sudo apt install apache2
```

Copied!

content_copy

This will show you a prompt with a list of packages that will be installed together with Apache2. These packages are known as the dependencies of the package. Press "y" at the prompt to continue with the installation.

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-
dbd-sqlite3 libaprutil1-ldap libbrotli1
  libjansson4 liblua5.2-0 ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
libaprutil1-dbd-sqlite3 libaprutil1-ldap libbrotli1
  libjansson4 liblua5.2-0 ssl-cert
0 upgraded, 12 newly installed, 0 to remove and 23 not upgraded.
```

```
Need to get 2628 kB of archives.
After this operation, 8946 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

As soon as it's installed. You will find the Apache2 web service isn't running.

```
sudo service apache2 status
```

Copied!

content_copy

```
apache2 is not running ... failed!
```

To start the Apache web service use the command below.

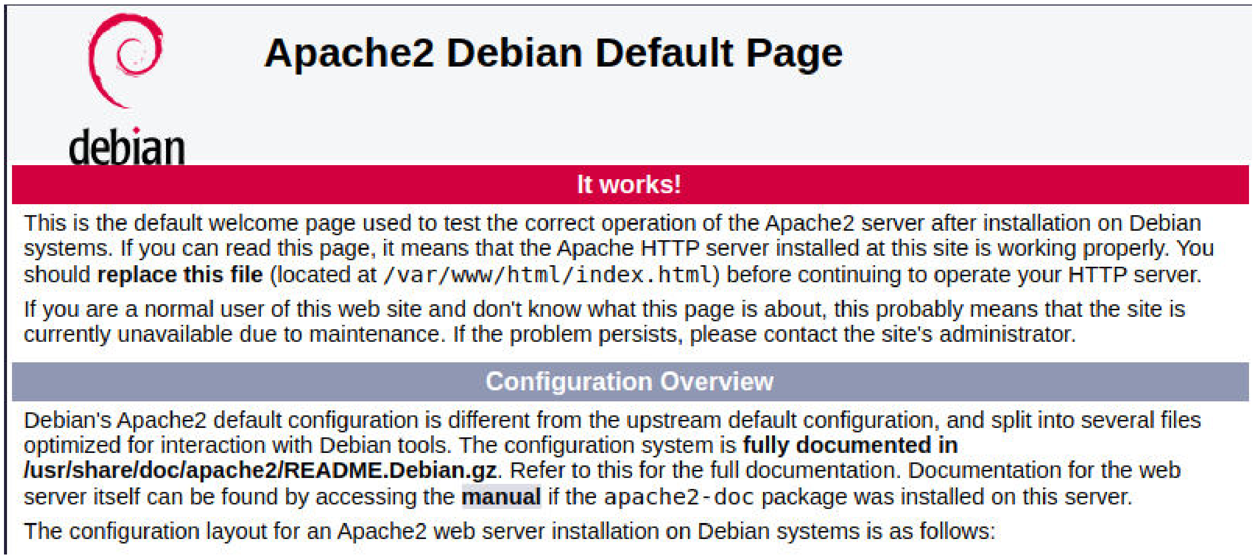
```
sudo service apache2 start
```

Copied!

content_copy

Apache2 will start serving its default webpage. To see this default web page, enter the `External IP address` that's shown in the connection panel, in a new separate browser tab.

You should see the default website there:


Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

That's great, we already have a web server working and serving its default page. Additionally, that default page is giving us a lot of useful information regarding how to configure Apache2.

Click *Check my progress* to verify the objective.

Install Apache2

Check my progress

Configuring sites

The default site lets us know that our web server is working. We now need to make sure the web server is serving *our* site, not the default one. Let's dive into how websites are managed by Apache2.

On any web server, you can have several sites running at the same time. Which site the user sees is determined by the port on which the website is served and the hostname used to reach the machine. Remember that many names can be used when referring to the same IP address.

For example, you could have an institutional site running on `http://www.example.com` and an e-commerce site running on `http://shop.example.com`, with both hosted on the same machine. This is known as **Virtual Hosts**.

The list of sites that are available is located in `/etc/apache2/sites-available`. Let's look at what the contents of that directory looks like.

```
ls -l /etc/apache2/sites-available
```

Copied!

content_copy

```
total 12
-rw-r--r-- 1 root root 1332 Aug  8  2020 000-default.conf
-rw-r--r-- 1 root root 6338 Aug  8  2020 default-ssl.conf
```

We see that there are two websites available. The default website configured by `000-default.conf` is the one we've visited, and the default encrypted website is managed by `default-ssl.conf`.

Let's look at the contents of the `000-default.conf` file to learn more about how the default website is configured.

```
cat /etc/apache2/sites-available/000-default.conf
```

Copied!

content_copy

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and
port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the
ServerName
        # specifies what hostname must appear in the request's Host:
header to
        # match this virtual host. For the default virtual host (this
file) this
        # value is not decisive as it is used as a last resort host
regardless.
        # However, you must set it for any further virtual host
explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice,
warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example
the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Lines that begin with `#` are comments. This gives us information on what the different parameters and settings mean. The actual configuration is very simple, and it's this section of the file:

Do not copy the below section of file

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

This indicates that the service will be listening on port 80 for all IPs. It then states the email address for the administrator of the service, the main path for the website, and the paths for the error and access logfiles.

Moving the website to the right location

We see that the directory where the default website is located is `/var/www/html`. It's standard practice to have all websites inside `/var/www`, so we should put ours there as well. Let's move it from its current location into `/var/www/ourcompany`.

```
sudo mv /opt/devel/ourcompany /var/www/ourcompany
```

Copied!

content_copy

The `mv` command (as many others on Linux) doesn't print any output when it succeeds. In order to see if it worked, let's look at the contents of `/var/www`

```
ls -l /var/www
```

Copied!

content_copy

```
total 8
drwxr-xr-x 2 root root 4096 Aug 11 12:09 html
drwxr-xr-x 2 root root 4096 Aug 11 12:07 ourcompany
```

Alright, we now have our website in the right place. Let's go back to configuring our site in Apache2.

Creating a new available site

We want to create our own site, so let's make a copy of the default site and then edit the new file.

```
cd /etc/apache2/sites-available
sudo cp 000-default.conf 001-ourcompany.conf
sudo nano 001-ourcompany.conf
```

Copied!

content_copy

The last command will open the nano text editor. We will be able to edit the file and change it as needed. In this case, we are going to change the directory where the files are stored. Instead of **/var/www/html** we will put our site in **/var/www/ourcompany**, so let's change that in the configuration file.

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and
port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the
ServerName
        # specifies what hostname must appear in the request's Host:
header to
        # match this virtual host. For the default virtual host (this
file) this
        # value is not decisive as it is used as a last resort host
regardless.
        # However, you must set it for any further virtual host
explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/ourcompany
        # Available loglevels: trace8, ..., trace1, debug, info, notice,
warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example
the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Once you've done this, press "**Ctrl-X**" to exit the editor. It will ask you if you want to save your changes. Press "**Y**" for yes and "**Enter**" at the filename prompt.

Enabling and disabling sites

We have now added a site that points to the right location, but this site is not yet enabled. The default site is still currently enabled. Apache2 allows us to have sites that are available and not necessarily enabled, to avoid disruptive changes. The enabled sites are managed in `/etc/apache2/sites-enabled`. Let's look at the contents of that directory:

```
ls -l /etc/apache2/sites-enabled
```

Copied!

content_copy

The arrows that we see show that this file is a symbolic link to the file in the `sites-available` directory. In other words, enabling or disabling a site in Apache2 is simply creating or removing a symbolic link between the `sites-available` and `sites-enabled` directories.

To simplify matters, there are a couple of commands `a2ensite` and `a2dissite`, that manage these symlinks for us (the names come from Apache2 enable/disable site). Let's use these commands to enable our new site and disable the default site.

```
sudo a2ensite 001-ourcompany.conf
```

Copied!

content_copy

```
Enabling site 001-ourcompany.
To activate the new configuration, you need to run:
  service apache2 reload
```

```
sudo a2dissite 000-default.conf
```

Copied!

content_copy

```
Site 000-default disabled.
To activate the new configuration, you need to run:
  service apache2 reload
```

And now let's look at the contents of the directory again:

```
ls -l /etc/apache2/sites-enabled
```

Copied!

content_copy

```
total 0
lrwxrwxrwx 1 root root 38 Aug 11 12:19 001-ourcompany.conf -> ../sites-
available/001-ourcompany.conf
```

Our site is enabled!

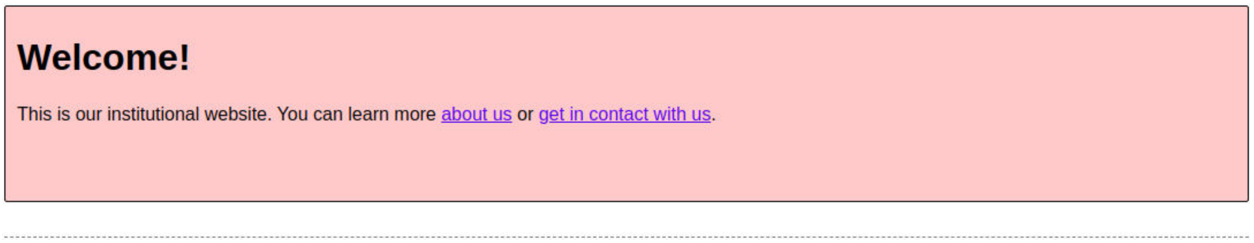
But, as the `a2` commands were telling us, if you reload the page pointing to your machine, you'll see that the default website is still the one being served. One more step is needed to make Apache2 notice that the configuration was changed: we need to tell the service to reload.

```
sudo service apache2 reload
```

Copied!

content_copy

This command doesn't give any output if it succeeds; to know if it worked, we need to visit the webpage. Do you see the institutional website?


Welcome!

This is our institutional website. You can learn more about us or get in contact with us.

Success!

Click *Check my progress* to verify the objective.

Enable Company Site

Check my progress

Additional configuration

If you look at the bottom of the website, you'll see that there's a horizontal line dividing the content from where a footer would be, but the footer isn't actually showing anything.

Let's look at the contents of the `index.html` page:

```
cat /var/www/ourcompany/index.html
```

Copied!

content_copy

```
<html>
        <head>
                <link rel="stylesheet" type="text/css" href="style.css">
                <title> Our Institutional website </title>
        </head>
        <body>
                <div class="content">
                <div class="main">
                        <h1>Welcome!</h1>
                        <p>This is our institutional website. You can
learn more <a href="aboutus.html">about us</a> or<a
href="contact.html">get in contact with us</a>.</p>
                </div>
                </div>
                <!--#include file="footer.html"-->
        </body>
</html>
```

The footer uses a feature provided by Apache2, called **Server Side Includes,** which currently is not enabled and therefore we don't see the footer.

Let's enable this feature so that the site is working properly.

Enabling modules

In the same way that we can have a list of available sites and enable them as needed, there is also a list of modules that provide additional functionality. Many of them are available and only a few that are enabled.

Let's look at the list of available modules:

```
ls /etc/apache2/mods-available
```

Copied!

content_copy

```
access_compat.load    buffer.load          headers.load
mpm_prefork.load     reqtimeout.load
actions.conf          cache.load           heartbeat.load
mpm_worker.conf      request.load
actions.load          cache_disk.conf      heartmonitor.load
mpm_worker.load      rewrite.load
alias.conf            cache_disk.load      http2.conf
negotiation.conf     sed.load
alias.load            cache_socache.load  http2.load
negotiation.load     session.load
allowmethods.load     cern_meta.load       ident.load
proxy.conf           session_cookie.load
asis.load             cgi.load             imagemap.load
proxy.load           session_crypto.load
auth_basic.load       cgid.conf            include.load
proxy_ajp.load       session_dbd.load
auth_digest.load      cgid.load            info.conf
proxy_balancer.conf  setenvif.conf
auth_form.load        charset_lite.load   info.load
proxy_balancer.load  setenvif.load
authn_anon.load       data.load            lbmethod_bybusyness.load
proxy_connect.load   slotmem_plain.load
authn_core.load       dav.load             lbmethod_byrequests.load
proxy_express.load   slotmem_shm.load
authn_dbd.load        dav_fs.conf          lbmethod_bytraffic.load
proxy_fcgi.load      socache_dbm.load
```

```
authn_dbm.load        dav_fs.load         lbmethod_heartbeat.load
proxy_fdpass.load    socache_memcache.load
authn_file.load       dav_lock.load       ldap.conf
proxy_ftp.conf       socache_shmcb.load
authn_socache.load    dbd.load            ldap.load
proxy_ftp.load       speling.load
authnz_fcgi.load      deflate.conf        log_debug.load
proxy_hcheck.load    ssl.conf
authnz_ldap.load      deflate.load        log_forensic.load
proxy_html.conf      ssl.load
authz_core.load       dialup.load         lua.load
proxy_html.load      status.conf
authz_dbd.load        dir.conf            macro.load
proxy_http.load      status.load
authz_dbm.load        dir.load            md.load
proxy_http2.load     substitute.load
authz_groupfile.load  dump_io.load        mime.conf
proxy_scgi.load      suexec.load
authz_host.load       echo.load           mime.load
proxy_uwsgi.load     unique_id.load
authz_owner.load      env.load            mime_magic.conf
proxy_wstunnel.load  userdir.conf
authz_user.load       expires.load        mime_magic.load
ratelimit.load       userdir.load
autoindex.conf        ext_filter.load     mpm_event.conf
reflector.load       usertrack.load
autoindex.load        file_cache.load     mpm_event.load
remoteip.load        vhost_alias.load
brotli.load           filter.load         mpm_prefork.conf
reqtimeout.conf      xml2enc.load
```

That's a long list. In order to know which one to enable, you would normally refer to Apache2 documentation for guidance.

In our case, we know that we want to enable `include`, which is the module used for **Server Side Includes**. Similarly to `a2ensite` and `a2dissite`, there are `a2enmod` and `a2dismod` for managing which modules get enabled. Let's enable `include` now.

```
sudo a2enmod include
```

Copied!

content_copy

```
Considering dependency mime for include:
Module mime already enabled
Enabling module include.
To activate the new configuration, you need to run:
  service apache2 restart
```

In this case, the message tells us that we need to restart the server. Merely reloading is not enough as this is not a configuration change. We need to install new functionality on the server. Let's do that.

```
sudo service apache2 restart
```

Copied!

content_copy

With that, the functionality is enabled in the server.

However, if you reload the website, the footer will still not be present. We need to also enable it in the configuration for our site.

Configuration options

In order to allow our site to use **Server Side Includes**, we need to set a few options in the configuration file for our site. So, let's open the file again and add the necessary lines.

```
sudo nano /etc/apache2/sites-available/001-ourcompany.conf
```

Copied!

content_copy

This is the snippet that you need to add after the `DocumentRoot` line:

```
<Directory /var/www/ourcompany>
        Options +Includes
        XBitHack on
</Directory>
```

Copied!

content_copy

This snippet is indicating that we want to add "`Includes`" as an option for the directory `/var/www/ourcompany`. Additionally, we are enabling a flag called `XBitHack`. This means that we indicate whether a file uses **Server Side Includes** or not by setting the file as executable (which our files are).

As before, once you've added this, press **"Ctrl-X"** to exit the editor. It will ask you if you want to save your changes; press **"Y"** for yes and then **"Enter"** at the filename prompt.

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and
port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the
ServerName
        # specifies what hostname must appear in the request's Host:
header to
        # match this virtual host. For the default virtual host (this
file) this
        # value is not decisive as it is used as a last resort host
regardless.
        # However, you must set it for any further virtual host
explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/ourcompany

        <Directory /var/www/ourcompany>
                Options +Includes
                XBitHack on
        </Directory>

        # Available loglevels: trace8, ..., trace1, debug, info, notice,
warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example
the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

```
# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

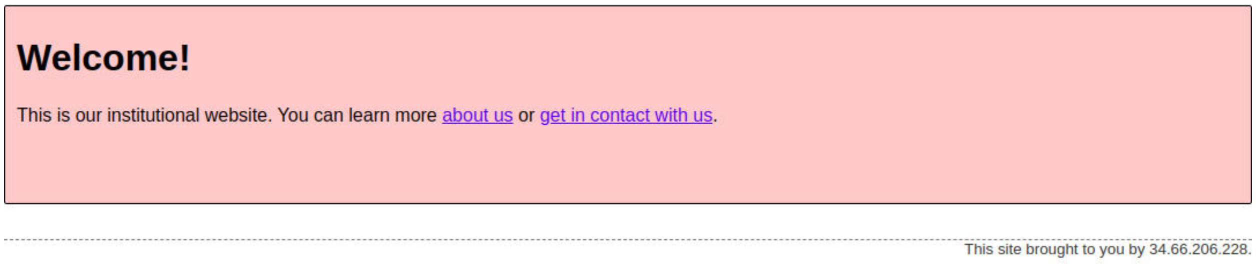
Now, let's tell Apache2 to reload its configuration again, so that it reads this last change:

```
sudo service apache2 reload
```

Copied!

content_copy

Now, if you visit the webpage, not only should you see the website, but the footer should be there as well.



Welcome!

This is our institutional website. You can learn more about us or get in contact with us.

This site brought to you by 34.66.206.228.

Click *Check my progress* to verify the objective.

Enable Server-Side Includes

Check my progress

Conclusion

You have now deployed and configured a website using Apache2. You know how to enable sites and modules as well as what Server Side Includes are. Congratulations!

Apache2 has many additional features that were not covered in this lab. You can additional documentation in the Apache2 website.