C4 M2 L4 Qwiklab: Managing Websites with Apache2

1 hour 1 Credit

Rate Lab

Introduction

Managing web servers is one of the most common tasks systems administrators do. 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.

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.

Head's up: Make sure to click the **"Start Lab"** button at the top of the screen. It may take a while for the lab to load. Please wait until the lab is running. To mark this lab as completed, make sure to click **"End Lab"** when you're done!

You'll have 60 minutes to complete this lab.

Start the lab

You'll need to start the lab before you can access the materials in the virtual machine OS. To do this, click the green “Start Lab” button at the top of the screen.

Note: For this lab you are going to access the **Linux VM** through your **local SSH Client**, and not use the **Google Console** (**Open GCP Console** button is not available for this lab).

After you click the “Start Lab” button, you will see all the SSH connection details on the left-hand side of your screen. You should have a screen that looks like this:

Accessing the virtual machine

Please find one of the three relevant options below based on your device's operating system.

Note: Working with Qwiklabs may be similar to the work you'd perform as an **IT Support Specialist**; you'll be interfacing with a cutting-edge technology that requires multiple steps to access, and perhaps healthy doses of patience and persistence(!). You'll also be using **SSH** to enter the labs -- a critical skill in IT Support that you'll be able to practice through the labs.

Option 1: Windows Users: Connecting to your VM

In this section, you will use the PuTTY Secure Shell (SSH) client and your VM's External IP address to connect.

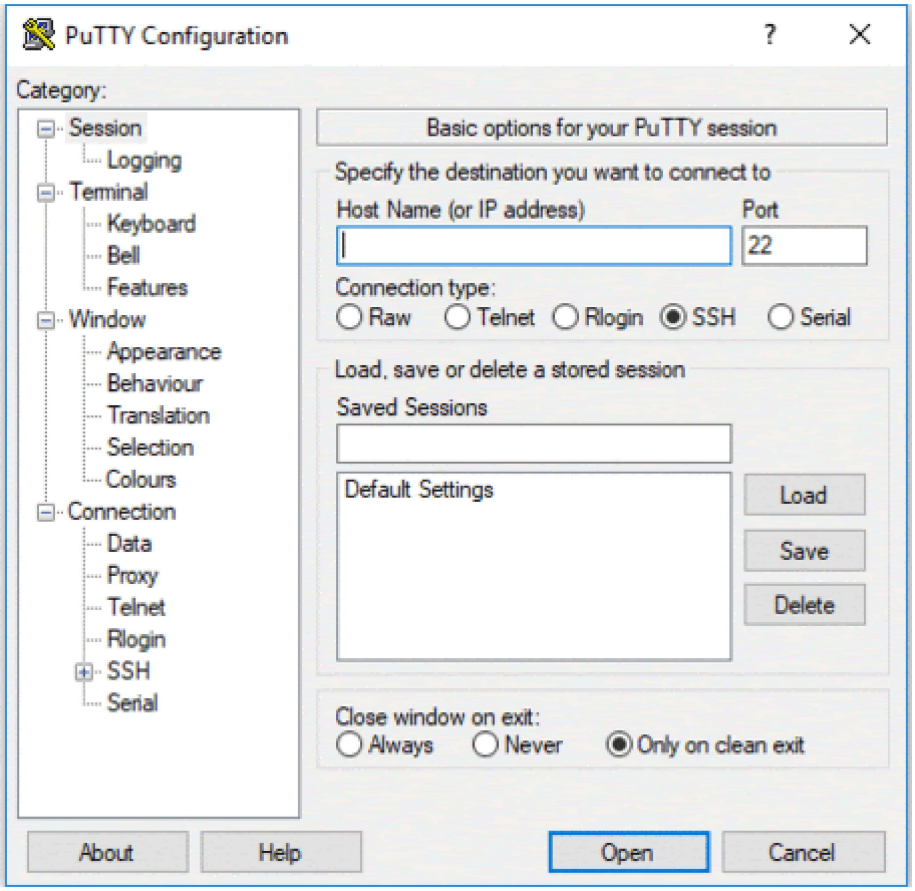
Download your PPK key file

You can download the VM's private key file in the PuTTY-compatible **PPK** format from the Qwiklabs Start Lab page. Click on **Download PPK**.

Connect to your VM using SSH and PuTTY

1. You can download Putty from here
2. In the **Host Name (or IP address)** box, enter username@external_ip_address.

Note: Replace **username** and **external_ip_address** with values provided in the lab.



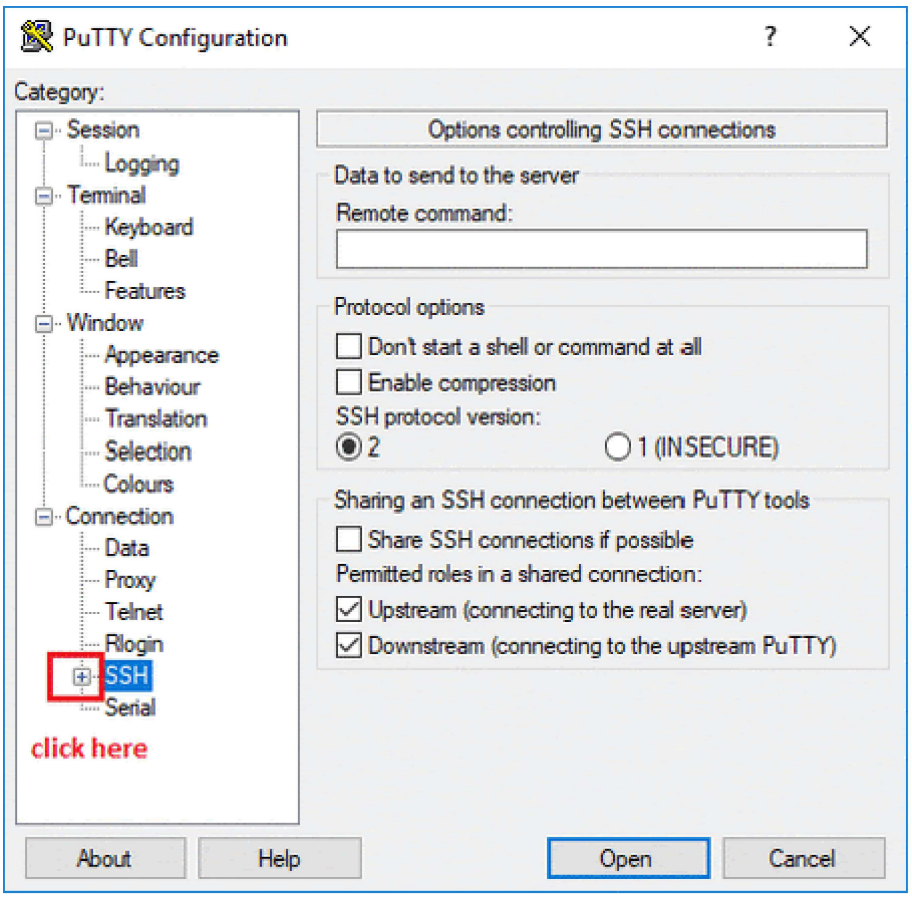
3. In the **Category** list, expand **SSH**.

4. Click **Auth** (don't expand it).

5. In the **Private key file for authentication** box, browse to the PPK file that you downloaded and double-click it.

6. Click on the **Open** button.

Note: PPK file is to be imported into PuTTY tool using the Browse option available in it. It should not be opened directly but only to be used in PuTTY.



7. Click **Yes** when prompted to allow a first connection to this remote SSH server. Because you are using a key pair for authentication, you will not be prompted for a password.

Common issues

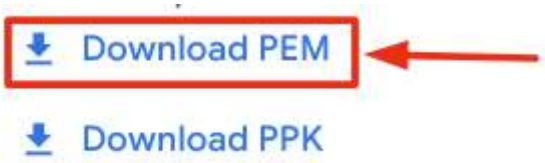
If PuTTY fails to connect to your Linux VM, verify that:

- You entered **<username>@<external ip address>** in PuTTY.
- You downloaded the fresh new PPK file for this lab from Qwiklabs.
- You are using the downloaded PPK file in PuTTY.

Option 2: OSX and Linux users: Connecting to your VM via SSH

Download your VM's private key file.

You can download the private key file in PEM format from the Qwiklabs Start Lab page. Click on **Download PEM**.



Connect to the VM using the local Terminal application

A **terminal** is a program which provides a **text-based interface for typing commands**. Here you will use your terminal as an SSH client to connect with lab provided Linux VM.

1. Open the Terminal application.
    - To open the terminal in Linux use the shortcut key **Ctrl+Alt+t**.
    - To open terminal in **Mac** (OSX) enter **cmd + space** and search for **terminal**.
2. Enter the following commands.

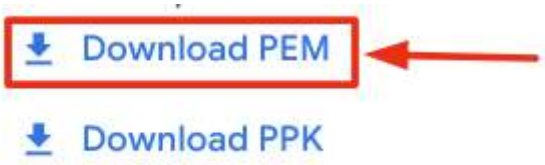
Note: Substitute the **path/filename for the PEM** file you downloaded, **username** and **External IP Address**.
You will most likely find the PEM file in **Downloads**. If you have not changed the download settings of your system, then the path of the PEM key will be **~/Downloads/qwikLABS-XXXXX.pem**

```
chmod 600 ~/Downloads/qwikLABS-XXXXX.pem
ssh -i ~/Downloads/qwikLABS-XXXXX.pem username@External Ip Address
```

Option 3: Chrome OS users: Connecting to your VM via SSH

Note: Make sure you are not in **Incognito/Private mode** while launching the application.
Download your VM's private key file.

You can download the private key file in PEM format from the Qwiklabs Start Lab page. Click on **Download PEM**.



Connect to your VM

1. Add Secure Shell from here to your Chrome browser.
2. Open the Secure Shell app and click on **[New Connection]**.

3. In the **username** section, enter the username given in the Connection Details Panel of the lab. And for the **hostname** section, enter the external IP of your VM instance that is mentioned in the Connection Details Panel of the lab.

4. In the **Identity** section, import the downloaded PEM key by clicking on the **Import…** button beside the field. Choose your PEM key and click on the **OPEN** button.

Note: If the key is still not available after importing it, refresh the application, and select it from the **Identity** drop-down menu.

5. Once your key is uploaded, click on the **[ENTER] Connect** button below.

6. For any prompts, type **yes** to continue.

7. You have now successfully connected to your Linux VM.

You're now ready to continue with the lab!

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.

Go ahead and connect to `linux-instance` by following the instructions given in the section `Accessing the virtual machine`. Click on `Accessing the virtual machine` from the navigation pane at the right side.

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
```

```
gcpstaging21716_student@linux-instance:~$ ls -l /opt/devel/ourcompany
total 20
-rwxr-xr-x 1 root root 576 Aug 24 17:59 aboutus.html
-rwxr-xr-x 1 root root 566 Aug 24 17:59 contact.html
-rwxr-xr-x 1 root root  97 Aug 24 17:59 footer.html
-rwxr-xr-x 1 root root 610 Aug 24 17:59 index.html
-rw-r--r-- 1 root root 819 Aug 24 17:59 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:

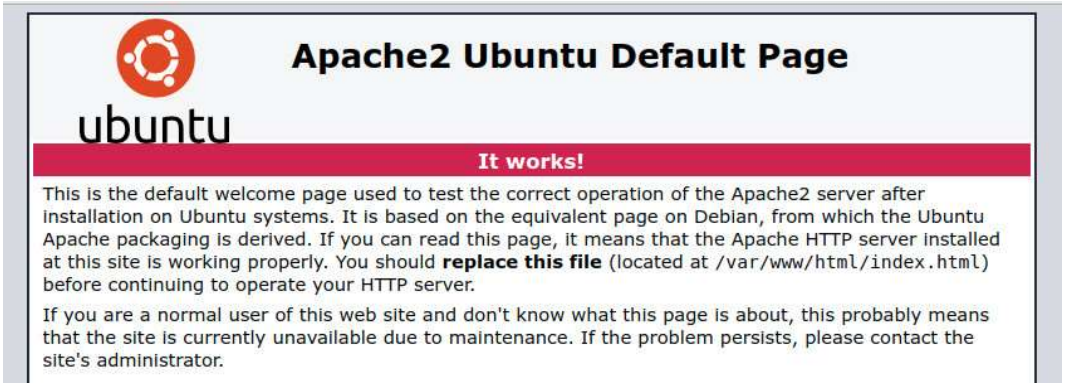
```
sudo apt update
sudo apt install apache2
```

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.

```
eduit531483_student@linux-instance:~$ sudo apt install apache2
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 liblua5.1-0
  ssl-cert
Suggested packages:
  www-browser apache2-doc apache2-suexec-pristine | apache2-suexec-custom openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
  liblua5.1-0 ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 1,557 kB of archives.
After this operation, 6,436 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

As soon as it's installed, Apache2 will start serving its default webpage. To see this default web page, enter the `linux-instance External IP address` in a new separate browser tab.

You should see the default website there:



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

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
```

```
gcpstaging21730_student@linux-instance:~$ ls -l /etc/apache2/sites-available
total 12
-rw-r--r-- 1 root root 1332 Sep 19  2017 000-default.conf
-rw-r--r-- 1 root root 6338 Apr  5 18:32 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
```

```
gcpstaging21730_student@linux-instance:~$ cat /etc/apache2/sites-available/000-default.conf
<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
```

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
```

```
gcpstaging21730_student@linux-instance:~$ ls -l /var/www
total 8
drwxr-xr-x 2 root root 4096 Aug 24 22:48 html
drwxr-xr-x 2 root root 4096 Aug 24 22:45 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
```

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

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     ^Y Prev Page
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line  ^V Next Page
```

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

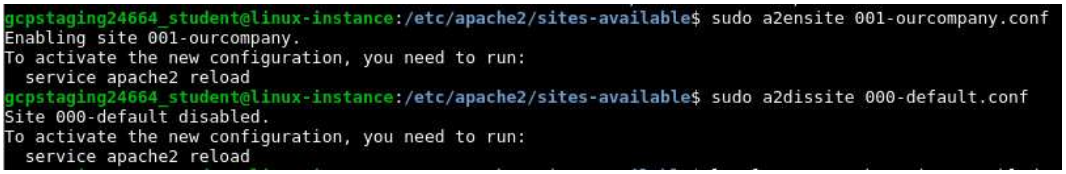
We have now added a site that points to the right location, but this site is not yet enabled. The default site is still currently enabled. Apache2 allows us to have sites that are available and not necessarily enabled, to avoid disruptive changes. The enabled sites are managed in `/etc/apache2/sites-enabled`.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

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

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
sudo a2dissite 000-default.conf
```



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

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

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
```

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?

Success!

Click *Check my progress* to verify the objective.

Enable Company Site

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
```

```
gcpstaging21730_student@linux-instance:/etc/apache2/sites-available$ cat /var/www/ourcompany/index.html
<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
```

```
gcpstaging21730_student@linux-instance:/etc/apache2/sites-available$ ls /etc/apache2/mods-available
access_compat.load    cgid.load                ldap.conf            proxy_scgi.load
actions.conf          cgi.load                 ldap.load            proxy_wstunnel.load
actions.load          charset_lite.load        log_debug.load       ratelimit.load
alias.conf            data.load                log_forensic.load    reflector.load
alias.load            dav_fs.conf              lua.load             remoteip.load
allowmethods.load     dav_fs.load              macro.load           reqtimeout.conf
asis.load             dav.load                 mime.conf            reqtimeout.load
auth_basic.load       dav_lock.load            mime.load            request.load
auth_digest.load      dbd.load                 mime_magic.conf      rewrite.load
auth_form.load        deflate.conf             mime_magic.load      sed.load
authn_anon.load       deflate.load             mpm_event.conf       session_cookie.load
authn_core.load       dialup.load              mpm_event.load       session_crypto.load
authn_dbd.load        dir.conf                 mpm_prefork.conf     session_dbd.load
authn_dbm.load        dir.load                 mpm_prefork.load     session.load
authn_file.load       dump_io.load             mpm_worker.conf      setenvif.conf
authn_socache.load    echo.load                mpm_worker.load      setenvif.load
authnz_fcgi.load      env.load                 negotiation.conf     slotmem_plain.load
authnz_ldap.load      expires.load             negotiation.load     slotmem_shm.load
authz_core.load       ext_filter.load          proxy_ajp.load       socache_dbm.load
authz_dbd.load        file_cache.load          proxy_balancer.conf  socache_memcache.load
authz_dbm.load        filter.load              proxy_balancer.load  socache_shmcb.load
authz_groupfile.load  headers.load             proxy.conf           speling.load
authz_host.load       heartbeat.load           proxy_connect.load   ssl.conf
authz_owner.load      heartmonitor.load        proxy_express.load   ssl.load
authz_user.load       http2.load               proxy_fcgi.load      status.conf
autoindex.conf        ident.load               proxy_fdpass.load    status.load
autoindex.load        imagemap.load            proxy_ftp.conf       substitute.load
buffer.load           include.load             proxy_ftp.load       suexec.load
cache_disk.conf       info.conf                proxy_hcheck.load    unique_id.load
cache_disk.load       info.load                proxy_html.conf      userdir.conf
cache.load            lbmethod_bybusyness.load proxy_html.load      userdir.load
cache_socache.load    lbmethod_byrequests.load proxy_http2.load     usertrack.load
cern_meta.load        lbmethod_bytraffic.load  proxy_http.load      vhost_alias.load
cgid.conf             lbmethod_heartbeat.load  proxy.load           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
```

```
gcpstaging24664_student@linux-instance:/etc/apache2/sites-available$ sudo a2enmod include
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
```

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.

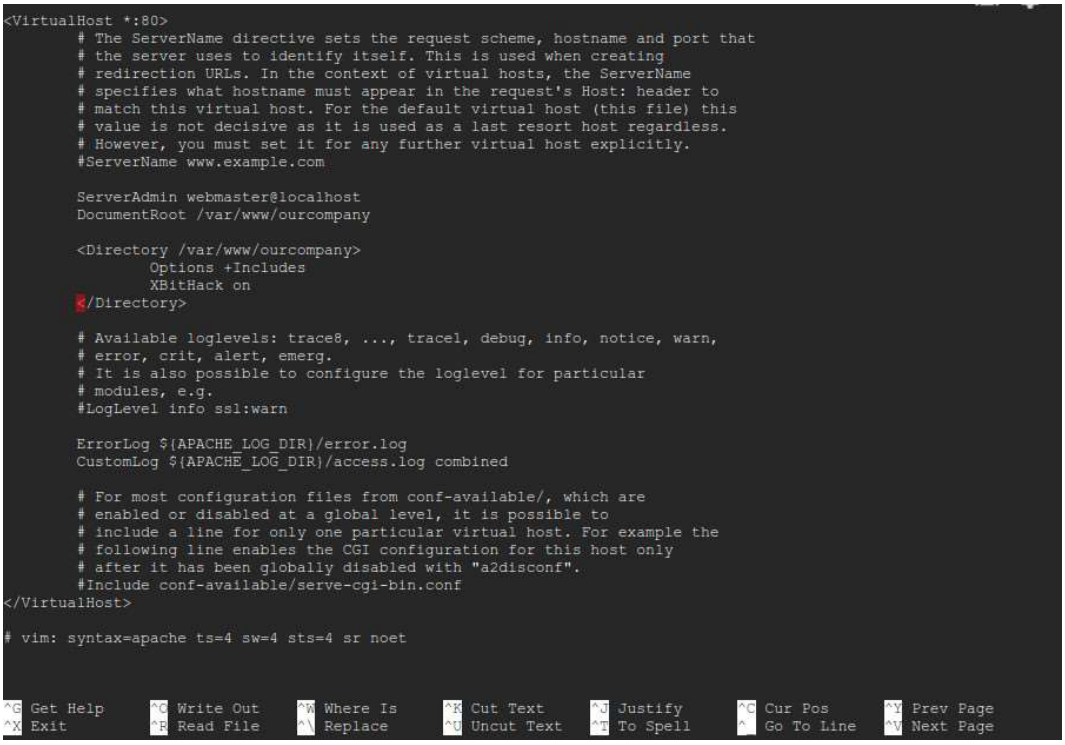
```
sudo nano /etc/apache2/sites-available/001-ourcompany.conf
```

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

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

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.



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

```
sudo service apache2 reload
```

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

Click *Check my progress* to verify the objective.

Enable Server-Side Includes

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.

End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.