Install Apache 2

Make sure your package repositories and installed programs are up to date by issuing the following commands:

```
apt-get update  apt-get upgrade --show-upgraded
```

Enter the following command to install the Apache 2 web server, its documentation and a collection of utilities.

```
apt-get install apache2 apache2-doc apache2-utils
```

Install Support for Scripting

The following commands are optional, and should be run if you want to have support within Apache for server-side scripting in PHP, Ruby, Python, or Perl.

To install Ruby support, issue the following command:

```
apt-get install libapache2-mod-ruby
```

To install Perl support, issue the following command:

```
apt-get install libapache2-mod-perl2
```

To install Python support, issue the following command:

```
apt-get install libapache2-mod-python
```

If you need support for MySQL in Python, you will also need to install Python MySQL support:

```
apt-get install python-mysqldb
```

Your PHP application may require additional dependencies included in Debian. To check for available PHP dependencies run "apt-cache search php", which will provide a list of package names and descriptions. To install, issue the following command:

```
apt-get install libapache2-mod-php5 php5 php-pear php5-xcache
```

Issue the following command to install the php5-suhosin package, which provides additional security to your PHP installation:

```
apt-get install php5-suhosin
```

If you're also hoping to run PHP with MySQL, then also install MySQL support:

```
apt-get install php5-mysql
```

Configure Apache for Named-Based Virtual Hosting

Apache supports both IP-based and name-based virtual hosting, allowing you to host multiple domains on a single server. To begin configuration, issue the following command to disable the default Apache virtual host.

```
a2dissite default
```

Each additional virtual host needs its own file in the /etc/apache2/sites-available/ directory. In this example, you'll create files for two
name-based
virtually hosted sites, "example.net" and "example.org".

First create example.net (/etc/apache2/sites-available/example.net) so that it resembles the following.

File:*/etc/apache2/sites-available/example.net*

ServerAdmin webmaster@example.net ServerName example.net ServerAlias www.example.net
DocumentRoot
/srv/www/example.net/public_html/

ErrorLog
/srv/www/example.net/logs/error.log

CustomLog
/srv/www/example.net/logs/access.log
combined

If you would like to enable Perl support, then add the following lines to the VirtualHost entry above.

File excerpt:*/etc/apache2/sites-available/bucknell.net*

```
Options ExecCGI  AddHandler cgi-script .pl
```

Next, create example.org (/etc/apache2/sites-available/example.org) so that it resembles this:

File:*/etc/apache2/sites-available/example.org*

```
ServerAdmin admin@example.org ServerName example.org ServerAlias
www.example.org
DocumentRoot
/srv/www/example.org/public_html/

ErrorLog
/srv/www/example.org/logs/error.log

CustomLog
/srv/www/example.org/logs/access.log
combined
```

You'll note that some basic options are specified for both sites, including where the files for the site will reside (under /srv/www/). You can add (or remove) additional configuration options, such as the Perl support, on a site-by-site basis to these files as your needs dictate.

Create required directories for these sites by issuing the following commands:

```
mkdir -p /srv/www/example.net/public_html  mkdir /srv/www/example.net/logs  mkdir -p
/srv/www/example.org/public_html  mkdir /srv/www/example.org/logs
```

Enable the sites by issuing these commands:

```
a2ensite example.net  a2ensite example.org
```

Finally, restart the Apache server to initialize all the changes, with this command:

```
/etc/init.d/apache2 restart
```

When you create or edit any virtual host file, you'll need to reload the config, which you can do without restarting the server with the following command:

```
/etc/init.d/apache2 reload
```

Congratulations! You now have Apache installed on your Debian Linux VPS and have configured the server for virtual hosting.

Install Apache Modules

One of Apache's prime strengths is its extreme customizability and flexibility. With its support for a large number of modules, there are few web serving tasks that Apache cannot fulfill. By default, modules and their configuration files are installed in the /etc/apache2/mods-available/ directory. Generating a list of this directory will tell you what modules are installed. To enable a module listed in this directory, use the following command:

```
a2enmod [module-name]
```

Note that in the /etc/apache2/mods-available/ directory, files have a .load and .conf extension. Module names do not include the extension.

To disable a module that is currently enabled, use the inverse command:

```
a2dismod [module-name]
```

To get a list of available Apache modules modules in the Debian repository use the following command:

```
apt-cache search libapache2*
```

To install one of these modules use the command:

```
apt-get install [module-name]
```

Modules should be enabled and ready to use following installation, though you may have to apply additional configuration options to have access to the modules' functionality. Consult the Apache module documentation
for more information regarding the configuration of specific modules.

Configuration Options

One of the strengths, and obstacles, of Apache is the immense amount of flexibility offered in its configuration files. In the default installation of Apache 2 on Debian, the main configuration is located in the /etc/apache2/apache2.conf files, but Apache configuration directives are loaded from files in a number of different locations, in a specific order. Configuration files are read in the following order, with items specified later taking precedence over earlier and potentially conflicting options:

1. /etc/apache2/apache2.conf
2. Files with .load or .conf extensions in /etc/apache2/mods-enabled/ directory.
3. /etc/apache2/httpd.conf (Blank by default.)
4. /etc/apache2/ports.conf
5. Files within the /etc/apache2/conf.d/ directory.
6. Files within the /etc/apache2/sites-enabled/ directory.
7. Per-directory .htaccess files in the directory.

Remember, later files take precedence over earlier-cited files. Within a directory of included configuration files, files will be read in order based on the sort of their file names.

Apache will follow symbolic links to read configuration files, so you can create links in these directories and locations to files that are actually located elsewhere in your file system.

Best practices for most installations dictate that we don't recommend modifying the following default configuration files: /etc/apache2/httpd.conf, files in /etc/apache2/mods-enabled/, and in most cases /etc/apache2/apache2.conf. This is to avoid unnecessary confusion and unintended conflicts in the future.

Generally, as specified in our LAMP guide for Debian 6 (Squeeze) and elsewhere, files that configure virtual hosts should be located in the
/etc/apache2/sites-available/
directory (and symbolically linked to
sites-enabled/
with the a2ensite tool. This allows for a clear and specific per-site configuration.

In practice, the vast majority of configuration options will probably be located in site-specific virtual host configuration files. If you need to set a system-wide configuration option or aren't using virtual hosting, the best practice is to specify options in files created beneath the conf.d/ directory.

Multi-Processing Module

The default Apache configuration uses a tool called MPM-worker, this multi-processing module can handle a large number of requests quickly by utilizing multiple threads per worker process. However, this use of multiple threads is not compatible with some PHP extensions. When PHP is installed MPM-worker is replaced with MPM-prefork, which allows Apache to handle requests without threading for greater compatibility with some software. Furthermore, using MPM-prefork allows Apache to isolate requests in separate processes so that if one request fails for some reason, other requests will be unaffected.

For more complex setups, however, we recommend that you consider using an alternate MPM module called "ITK." mpm-itk is quite similar to prefork, but it goes one step further and runs the processes for each site under a distinct user account. This is particularly useful in situations where you're hosting a number of distinct sites that you need to isolate sites on the basis of user privileges.

Begin by installing the mpm-itk module:

```
apt-get install apache2-mpm-itk
```

Now, in the entries for your sites (the site-specific files in /etc/apache2/sites-available/) add the following sub-block:

File excerpt:*Apache Virtual Host Configuration*

```
AssignUserId webeditor webgroup
```

In this example, webeditor is the name of the user of the specific site in question, and webgroup is the name of the particular group that "owns" the web server related files and processes. Remember that you must create the user accounts and groups using the useradd command.