



WEBSITE FOR STUDENTS

Learn Ubuntu Linux, Windows and CMS



How to Install Moodle with Apache2 and Let's Encrypt on Ubuntu 18.04 | 16.04

 by Richard Zayzay  06/07/2019

Moodle LMS, an open source course manage system (CMS) is great for professionals, businesses and colleges who want to run and manage their courses or training materials online via the web...

It is probably the best and most popular open source learning management platform available today... However, if you're going to be setting up Moodle in your own environment, make sure to run it over HTTPS...

Websites and apps running over HTTPS perform better than those that do not!

Google and other search engines also rank HTTPS websites better than HTTP and your user privacy will be ensured as well... Going forward, always build your sites to be HTTPS compliant!

This brief tutorial will show students and new user a step by step guide on how to setup Moodle websites with Apache2 and use Let's Encrypt free SSL/TLS certificates and security features to help improve their websites performance and protect against malicious actors..

This setup might take a while to complete and the process below should work on other websites as well... It doesn't have to be Moodle... This setup should work on other CMSes and plain HTML sites out of the box... When you're ready to setup Moodle and Let's Encrypt, follow the steps below:

Step 0: Get your Domain Name

Let's Encrypt works with valid domain and a working server that the domain is pointing to... This setup assumes that your domain name is called **example.com** and is pointing to your server with IP address **192.168.1.2**

Don't forget to also make sure **www** CNAME is pointing to the domain name.... Should look like something below:

example.com	A	=====>	192.168.1.2
www	CNAME	=====>	example.com

Step 1: Install and Configure Moodle

Now that you've configured your domain to point to your server, continue below to setting up Moodle and Let's Encrypt...

First install Apache2 HTTP server since we're using Apache2 for this post.. To install Apache2 server, run the commands below:

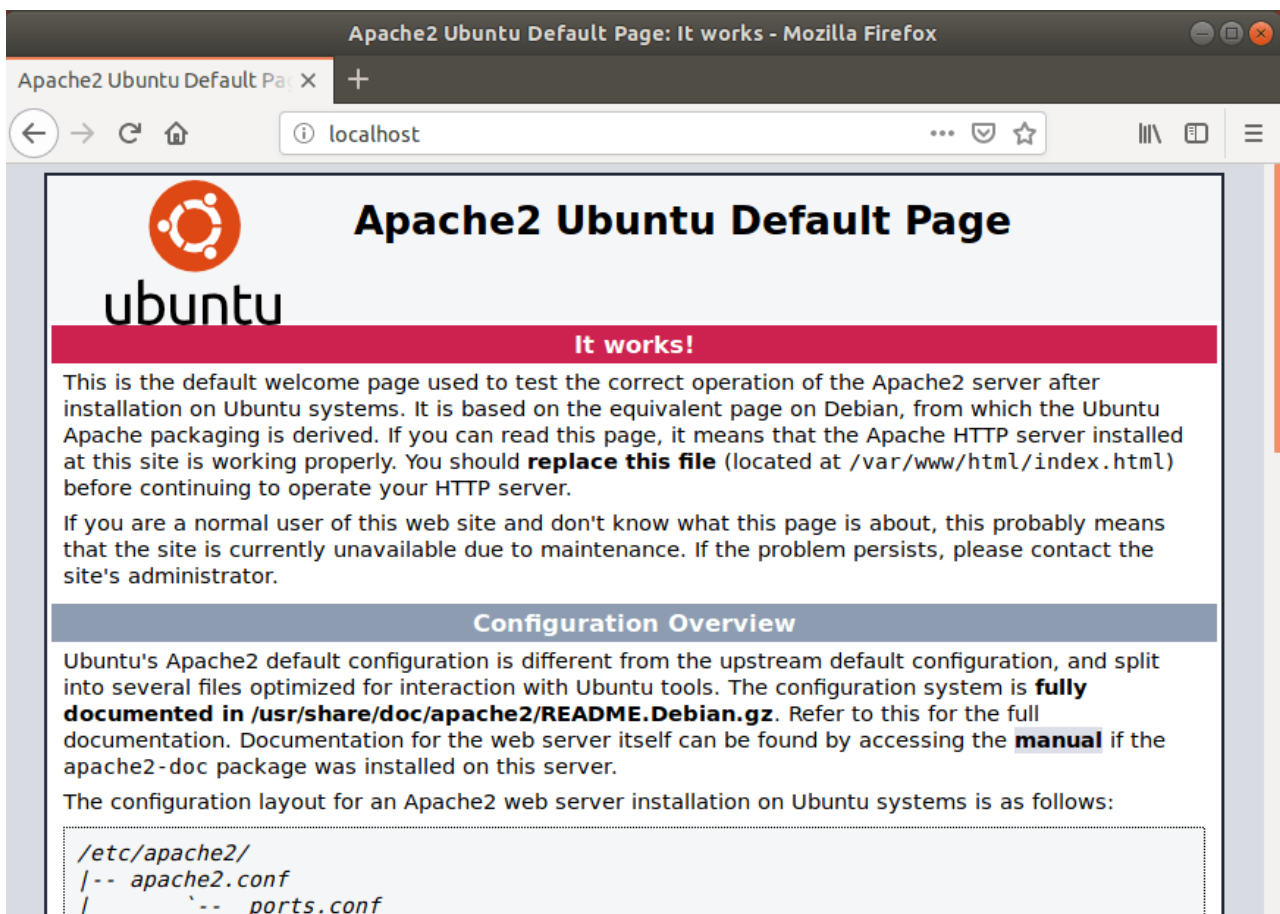
```
sudo apt update
sudo apt install apache2
```

After installing Apache2, the commands below can be used to stop, start and enable Apache2 service to always start up with the server boots...

```
sudo systemctl stop apache2.service
sudo systemctl start apache2.service
sudo systemctl enable apache2.service
```

Now that Apache2 is installed.... to test whether the web server is working, open your browser and browse to the URL below...

<https://localhost>



If you see the page above, then Apache2 is successfully installed...

Step 2: Install MariaDB Database Server

Moodle also requires a database server to store its content... If you're looking for a truly open source database server, then MariaDB is a great place to start... To install MariaDB run the commands below:

```
sudo apt-get install mariadb-server mariadb-client
```

After installing MariaDB, the commands below can be used to stop, start and enable MariaDB service to always start up when the server boots...

Run these on Ubuntu 16.04 LTS

```
sudo systemctl stop mysql.service
sudo systemctl start mysql.service
sudo systemctl enable mysql.service
```

Run these on Ubuntu 19.04 and 18.04 LTS

```
sudo systemctl stop mariadb.service
sudo systemctl start mariadb.service
sudo systemctl enable mariadb.service
```

Next, run the commands below to secure the database server with a root password if you were not prompted to do so during the installation...

```
sudo mysql_secure_installation
```

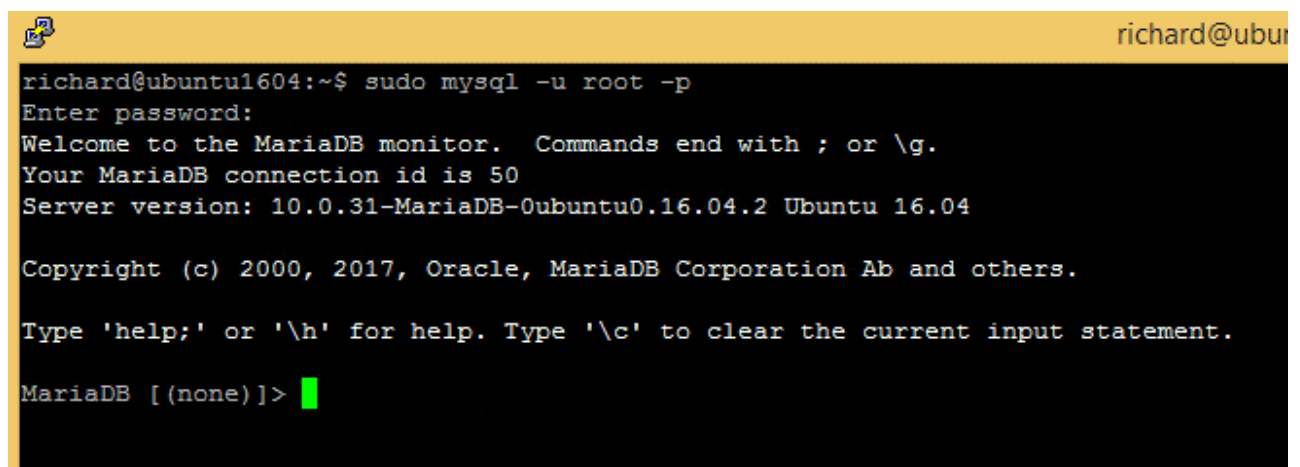
When prompted, answer the questions below by following the guide.

- Enter current password for root (enter for none): Just press the **Enter**
- Set root password? [Y/n]: **Y**
- New password: **Enter password**
- Re-enter new password: **Repeat password**
- Remove anonymous users? [Y/n]: **Y**
- Disallow root login remotely? [Y/n]: **Y**
- Remove test database and access to it? [Y/n]: **Y**
- Reload privilege tables now? [Y/n]: **Y**

Now that MariaDB is installed, to test whether the database server was successfully installed, run the commands below...

```
sudo mysql -u root -p
```

type the root password when prompted...

A terminal window with a yellow title bar. The title bar contains a small icon on the left and the text 'richard@ubun' on the right. The terminal content shows the command 'sudo mysql -u root -p' being executed. It prompts for a password, then displays a welcome message for the MariaDB monitor, including the connection ID (50) and server version (10.0.31-MariaDB-0ubuntu0.16.04.2). It also shows copyright information and instructions on how to use the monitor. The prompt 'MariaDB [(none)]>' is shown at the bottom with a green cursor.

```
richard@ubuntu1604:~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 50
Server version: 10.0.31-MariaDB-0ubuntu0.16.04.2 Ubuntu 16.04

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

If you see a similar screen as shown above, then the server was successfully installed...

Step 3: Install PHP 7.2 and Related Modules

Moodle CMS is a PHP based CMS and PHP is required... However, PHP 7.2 may not be available in Ubuntu default repositories... To run PHP 7.2 on Ubuntu 16.04 and previous, you may need to run the commands below:

```
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:ondrej/php
```

Then update and upgrade to PHP 7.2

```
sudo apt update
```

Next, run the commands below to install PHP 7.2 and related modules.

```
sudo apt install php7.2 libapache2-mod-php7.2 php7.2-common php7.2-mysql php7.2-gmp php7.2-curl
php7.2-intl php7.2-mbstring php7.2-xmlrpc php7.2-gd php7.2-xml php7.2-cli php7.2-zip
```

After installing PHP 7.2, run the commands below to open PHP default configuration file for Apache2...

```
sudo nano /etc/php/7.2/apache2/php.ini
```

The lines below is a good settings for most PHP based CMS... Update the configuration file with these and save....

```
file_uploads = On
allow_url_fopen = On
short_open_tag = On
memory_limit = 256M
upload_max_filesize = 100M
max_execution_time = 360
date.timezone = America/Chicago
```

Everytime you make changes to PHP configuration file, you should also restart Apache2 web server... To do so, run the commands below:

```
sudo systemctl restart apache2.service
```

Now that PHP is installed, to test whether it's functioning, create a test file called **phpinfo.php** in Apache2 default root directory.... (**/var/www/html/**)

```
sudo nano /var/www/html/phpinfo.php
```

Then type the content below and save the file.

```
<?php phpinfo( ); ?>
```

Next, open your browser and browse to the server's hostname or IP address followed by **phpinfo.php**

<http://localhost/phpinfo.php>

You should see PHP default test page...

Step 4: Create Moodle Database

Now that you've installed all the packages that are required for Moodle to function, continue below to start configuring the servers. First run the commands below to create a blank Moodle database.

To logon to MariaDB database server, run the commands below.

```
sudo mysql -u root -p
```

Then create a database called **moodle**

```
CREATE DATABASE moodle;
```

Create a database user called **moodleuser** with a new password

```
CREATE USER 'moodleuser'@'localhost' IDENTIFIED BY 'new_password_here';
```

Then grant the user full access to the database.

```
GRANT ALL ON moodle.* TO 'moodleuser'@'localhost' IDENTIFIED BY 'user_password_here' WITH GRANT OPTION;
```

Finally, save your changes and exit.

```
FLUSH PRIVILEGES;  
EXIT;
```

Step 5: Download Moodle Latest Release

To get Moodle latest release you may want to use Github repository... Install Curl and other dependencies to get started...

```
sudo apt install git curl
```

After installing git and curl above, change into the Apache2 root directory and download Moodle packages from Github... Always replace the **branch number** with the latest branch.... The current major version is **36**...

```
cd /var/www/html  
sudo git clone -b MOODLE_36_STABLE git://git.moodle.org/moodle.git example.com  
sudo mv moodle /var/www/html/
```

Then run the commands below to set the correct permissions for Moodle to function.

```
sudo mkdir -p /var/www/html/moodledata  
sudo chown -R www-data:www-data /var/www/html/  
sudo chmod -R 755 /var/www/html/
```

Step 6: Configure Apache2

Next, configure Apache2 site configuration file for Moodle.. This file will control how users access Moodle content. Run the commands below to create a new configuration file called **example.com.conf**

```
sudo nano /etc/apache2/sites-available/example.com.conf
```

Then copy and paste the content below into the file and save it. Replace the highlighted line with your own domain name and directory root location.

```
<VirtualHost *:80>
  ServerName example.com
  ServerAlias www.example.com
  ServerAdmin admin@example.com
  DocumentRoot /var/www/html/example.com

  <Directory /var/www/html/example.com/>
    Options FollowSymlinks
    AllowOverride All
    Require all granted
  </Directory>

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

  <Directory /var/www/html/example.com/>
    RewriteEngine on
    RewriteBase /
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^(.*) index.php [PT,L]
  </Directory>
</VirtualHost>
```

Save the file and exit.

Now the the **example.com** configuration file is created, run the commands below to enable it...

```
sudo a2ensite example.com.conf
```

At this point Apache2 should be configured and ready to respond over HTTP... It doesn't yet support HTTPS.

Step 7: Install and Configure Let's Encrypt

Now that our Apache2 site is enabled and ready to use, run the commands below to install and configure Let's Encrypt to secure the Apache2 website...

First install **Certbot**... Certbot is a fully featured and easy to use tool that can automate the tasks for obtaining and renewing Let's Encrypt SSL certificates...

To install it, run the commands below:

```
sudo apt install certbot
```

After installing Certbot, create a file to for Let's Encrypt to the Webroot plugin to validate our domain in the **\${webroot-path}/.well-known/acme-challenge** directory....

To do that, create the directory and give Apache2 access to it...

```
sudo mkdir -p /var/lib/letsencrypt/.well-known
sudo chgrp www-data /var/lib/letsencrypt
sudo chmod g+s /var/lib/letsencrypt
```

Next, create a well-known challenge file with the configurations below...

```
sudo nano /etc/apache2/conf-available/well-known.conf
```

Then copy and paste the content below into the file and save...

```
Alias /.well-known/acme-challenge/ "/var/lib/letsencrypt/.well-known/acme-challenge/"
<Directory "/var/lib/letsencrypt/">
    AllowOverride None
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    Require method GET POST OPTIONS
</Directory>
```

Save the file and exit

Step 8: Obtain Your Free Certificate

At this point, your domain should be pointing to your server IP... Apache2 HTTP server installed and configured and Certbot installed ready to obtain your certificate...

Before requesting your free certificate, open your [example.com](#) enable Apache2 configurations and modules by running the commands below...

The commands below enable Apache2 SSL, Headers, HTTPS/2 and the well-known configuration file we created above..

```
sudo a2enmod ssl
sudo a2enmod headers
sudo a2enmod http2
sudo a2enconf well-known
```

After enabling the modules and config file above, restart Apache2 server... To do that, run the commands below

```
sudo systemctl restart apache2
```

At this point all is set and you're ready to obtain your certificate... To do that run the commands below:

```
sudo certbot certonly --agree-tos --email admin@example.com --webroot -w /var/lib/letsencrypt/ -d
example.com -d www.example.com
```

Let's Encrypt should connect validate your domain and server, then install the domain certificate... If everything is successful, you should see a similar message as below:

IMPORTANT NOTES:

- Congratulations! Your certificate and chain have been saved at:
/etc/letsencrypt/live/example.com/fullchain.pem
Your key file has been saved at:
/etc/letsencrypt/live/example.com/privkey.pem
Your cert will expire on 2019-08-18. To obtain a new or tweaked version of this certificate in the future, simply run certbot again. To non-interactively renew *all* of your certificates, run "certbot renew"
- If you like Certbot, please consider supporting our work by:
Donating to ISRG / Let's Encrypt: <https://letsencrypt.org/donate>
Donating to EFF: <https://eff.org/donate-le>

At this point you have a certificate, now go and add it to Apache2 configuration for example.com domain...

First, let's generate a Diffie-Hellman key exchange (DH) certificate to securely exchange cryptographic keys... To do that, run the commands below to generate a certificate with 2048 bit...

```
sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
```

Next, open your example.com config file and make it so that it looks similar to the one below:

```
sudo nano /etc/apache2/sites-available/example.com.conf
```

Configure your file to look similar to the one below

```
<VirtualHost *:80>
    ServerName example.com
    ServerAlias www.example.com

    Redirect permanent / https://example.com/
</VirtualHost>

<VirtualHost *:443>
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/html/example.com

    Protocols h2 http/1.1

    <If "%{HTTP_HOST} == 'www.example.com'">
        Redirect permanent / https://example.com/
    </If>

    ErrorLog ${APACHE_LOG_DIR}/example.com-error.log
    CustomLog ${APACHE_LOG_DIR}/example.com-access.log combined
```

```

SSLEngine On
SSLCertificateFile /etc/letsencrypt/live/example.com/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/example.com/privkey.pem
SSLOpenSSLConfCmd DHParameters "/etc/ssl/certs/dhparam.pem"

SSLCipherSuite ECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH
SSLProtocol All -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
SSLCompression off
SSLUseStapling on

<Directory /var/www/html/example.com/>
    Options FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>

<Directory /var/www/html/example.com/>
    RewriteEngine on
    RewriteBase /
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^(.*) index.php [PT,L]
</Directory>
</VirtualHost>

```

Next you will need to configure a server cache for the OCSP status information. The best place for this would be in the Apache SSL configuration file.

```
sudo nano /etc/apache2/mods-available/ssl.conf
```

This file contains all the options that Apache uses for SSL. An additional option `SSLStaplingCache`, needs to be added to this file as below.

```

# Set the location of the SSL OCSP Stapling Cache
SSLStaplingCache shmcb:/tmp/stapling_cache(128000)

```

The `SSLStaplingCache` directive defines the location for the cache and a size value for the OCSP cache.

Save your changes above and restart Apache2 for the settings above to take effect..

```
sudo systemctl restart apache2
```

To setup a process to automatically renew the certificates, add a cron job to execute the renewal process.

```
sudo crontab -e
```

Then add the line below and save.

```
0 1 * * * /usr/bin/certbot renew & > /dev/null
```

The cron job will attempt to renew 30 days before expiring

Step 9: Enable the Moodle

After configuring the VirtualHost above, enable it by running the commands below

```
sudo a2ensite example.com.conf
sudo a2enmod rewrite
sudo systemctl restart apache2.service
```

Then open your browser and browse to the server domain name. You should see Moodle setup wizard to complete. Please follow the wizard carefully.

<http://example.com/>

Then follow the on-screen instructions and select the installation language here...

Next, select MariaDB connection driver and continue...

On the next screen, enter the database connection info you created above and continue...

Then create an admin account and the Moodle site info and finish the installation....

On this page you should configure your main administrator account which will have complete control over the site. Make sure you give it a secure username and password as well as a valid email address. You can create more admin accounts later on.

Congratulation! You have successfully installed Moodle on Ubuntu 16.04 | 18.04 and may work on upcoming 18.10...

In the future when you want to upgrade to a new released version, simply run the commands below to upgrade...

Upgrading Moodle

First stop the webserver...

```
sudo systemctl stop apache2
```

For students and new users who already have Moodle installed and wish to upgrade, assuming that you followed the steps above to install, run the commands below to backup your old Moodle folder...

```
sudo mv /var/www/html/example.com /var/www/html/example.com_bak
```

Then change into the webserver root directory and download the latest version of Moodle from Github..... always change the **version number** to the current (latest)

```
cd /var/www/html
sudo git clone -b MOODLE_37_STABLE git://git.moodle.org/moodle.git example.com
```

Next, copy Moodle config file, theme and data folder... If you updated your themes... a theme content should be there.... If you also installed additional modules... you should find them in the **/mod** directory... copy them to the new Moodle folder....


```
sudo cp /var/www/html/example.com_bak/config.php /var/www/html/example.com
sudo cp -pr /var/www/html/example.com_bak/theme/mytheme /var/www/html/example.com/theme/mytheme
sudo cp -pr /var/www/html/example.com_bak/mod/mymod /var/www/html/example.com/mod/mymod
```

After that, update the web server permissions...

```
sudo chown -R www-data:www-data /var/www/html/example.com/
sudo chmod -R 755 /var/www/html/example.com/
```

Restart your web server...

```
sudo systemctl start apache2
```

The last step is to trigger the upgrade processes within Moodle..... If you put your site into Maintenance mode earlier; take it out now!

Once you browse to the server IP or hostname, Moodle should prompt you to begin upgrading your database... After upgrading the database, logon to Moodle and go to:

Administration > Site administration > Notifications.

Moodle will automatically detect the new version and perform all the SQL database or file system upgrades that are necessary. If there is anything it can't do itself (very rare) then you will see messages telling you what you need to do.

Assuming all goes well (no error messages) then you can start using your new version of Moodle and enjoy the new features!

That's it!

Congratulations! You have successfully installed Moodle with Cloudflare support on Ubuntu 16.04 | 18.04

You may also like the post below:

Install Memcached on Ubuntu 18.04 | 16.04 with Apache2 and PHP-7.2

Posted in Applications, Labs, Linux Ubuntu • Tagged let's encrypt, Moodle, Ubuntu 16.04 LTS, Ubuntu 18.04

Published by Richard Zayzay

Hi, I'm Richard. In my spare time, I research topics that are interesting and worthwhile for users and students who want to try something new. I, too, am a student and my focus here is to help other students and new users get started with managing Ubuntu Linux, Windows, Content Management Systems (CMS) and others. I try to do my best explaining the topics and detailing the instructions so that anyone can understand. These tutorials may not work in all situations and for all users. However, if you run into trouble, please ask your questions below and I or someone from the community may help you resolve. Thanks for reading and hope you come back. ~Richard [View all posts by Richard Zayzay](#)

Prev

Install Memcached on Ubuntu 18.04 | 16.04 with Apache2 and PHP-7.2

4 Replies to “How to Install Moodle with Apache2 and Let’s Encrypt on Ubuntu 18.04 | 16.04”

Mark

06/26/2019 at 11:57 AM

Hi there

I get a The page isn't redirecting properly error when i try to load moodle.

Any ideas on how to fix this?

Thanks

[REPLY](#)

Ilir

11/23/2019 at 2:05 PM

Remove this from sites-enabled

RewriteEngine on

RewriteBase /

RewriteCond %{REQUEST_FILENAME} !-f

RewriteRule ^(.*) index.php [PT,L]

and it will work

[REPLY](#)

Najam

04/20/2020 at 4:18 AM

I have removed that from .conf but still I'm getting a non-css page. The login form does not login user and the css or js is not loading. However, before SSL setting, everything was working fine.

Any ideas what to change?

[REPLY](#)

dennis

06/24/2020 at 5:22 AM

i remove this part on mine

Redirect permanent / <https://example.com/>

[REPLY](#)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

POST COMMENT

This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)

Search ...



Currently Trending

How to Install SpiderFoot on Ubuntu

Working with Sudo and Su Commands in Ubuntu Linux

How to Install ELK Stack on Ubuntu

How to Install GNS3 on Ubuntu Linux

[How to Install Discord on Ubuntu](#)

[How to Boot into Windows 10 Safe Mode](#)

[How to Install ConfigServer Security & Firewall on Ubuntu](#)

[How to Install Chef Server on Ubuntu](#)

[How to Install Flectra on Ubuntu](#)

[How to Clear Edge Browser Data on Close](#)

[How to Install ReactJS on Ubuntu](#)

[How to use Chattr Command in Ubuntu Linux](#)