

How to Install Apache Web Server on Ubuntu 24.04

Learn how to install and configure Apache web server on Ubuntu 24.04 with step-by-step instructions for beginners. Get your website up and running today.

Contents

01	Introduction	3
02	Prerequisites	3
03	Install Apache	3
04	Manage the Apache System Service	5
05	Create a new Apache Virtual Host	6
06	Secure the Apache Web Server	10
07	Conclusion	14

Introduction

Apache is an open-source web server application that enables the delivery of static and dynamic web applications on a server. It's highly customized, supports multiple modular extensions, and can work as a reverse proxy or load balancer to deliver web applications or backend services on your server.

This article explains how to install the Apache web server on Ubuntu 24.04 and securely deliver web applications using virtual host configurations on your server.

Prerequisites

Before you begin:

- Deploy an [Ubuntu 24.04 instance](#) on Vultr.
- Create a [new domain A record pointing to the server IP address](#). For example, `app.example.com`.
- Access the server [using SSH](#) and log in as a non-root user with sudo privileges.
- [Update the server](#).

Install Apache

The Apache web server package is available in the default repositories on Ubuntu 24.04. Follow the steps below to install the latest Apache version using the default APT package manager and enable the web server to automatically start at boot time.

1. Update the server's package index.

```
CONSOLE
```

```
$ sudo apt update
```

2. Install the Apache web server package.

CONSOLE

```
$ sudo apt install apache2 -y
```

3. View the installed Apache web server version.

CONSOLE

```
$ apachectl -v
```

Output:

```
Server version: Apache/2.4.58 (Ubuntu)  
Server built: 2024-07-11T14:41:54
```

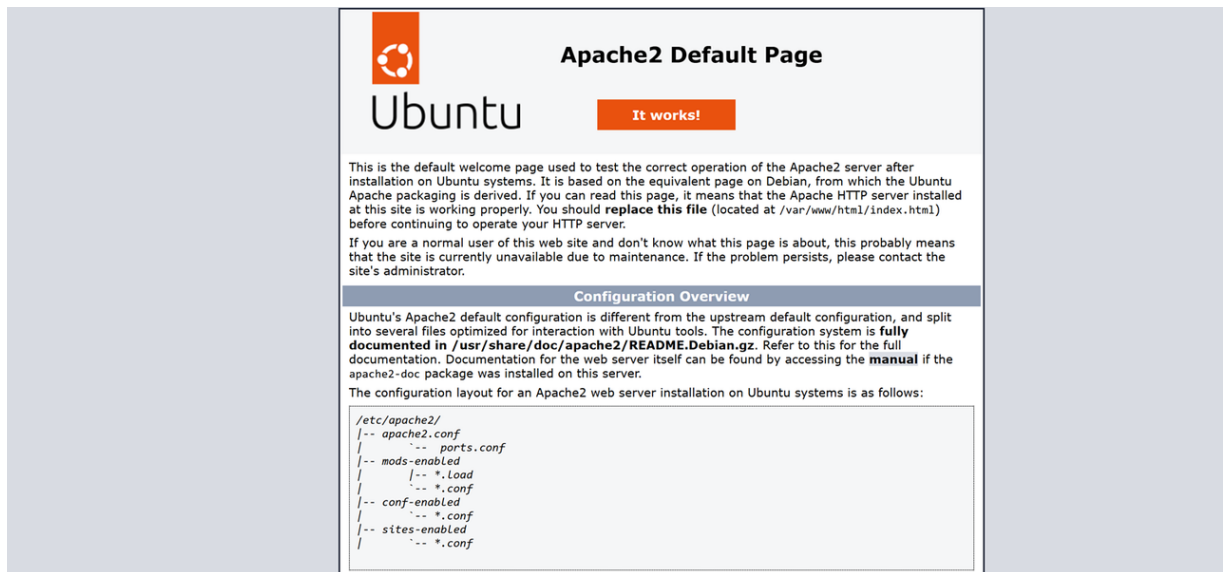
4. Allow connections on the HTTP port `80` through the default UFW firewall configurations.

CONSOLE

```
$ sudo ufw allow 80/tcp
```

5. Access your public server IP using a web browser such as Chrome and verify that the default Apache web page displays.

```
http://SERVER-IP
```



Manage the Apache System Service

Apache runs with the `apache` system service profile that manages the web server runtime processes on your server. Follow the steps below to enable the Apache web server to start at boot time, and verify the system service status.

1. Enable the Apache service to automatically start at boot time.

CONSOLE

```
$ sudo systemctl enable apache2
```

Output:

```
Synchronizing state of apache2.service with SysV service script with /usr/lib/  
systemd/systemd-sysv-install.
```

```
Executing: /usr/lib/systemd/systemd-sysv-install enable apache2
```

2. Start the Apache web server.

CONSOLE

```
$ sudo systemctl start apache2
```

3. View the Apache service status and verify that it's active on your server.

CONSOLE

```
$ sudo systemctl status apache2
```

Output:

```
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset:
   enabled)
   Active: active (running) since Thu 2024-07-18 00:04:53 UTC; 41s ago
     Docs: https://httpd.apache.org/docs/2.4/
  Main PID: 4137 (apache2)
    Tasks: 55 (limit: 1061)
   Memory: 6.0M (peak: 6.1M)
      CPU: 31ms
   CGroup: /system.slice/apache2.service
           └─4137 /usr/sbin/apache2 -k start
           └─4139 /usr/sbin/apache2 -k start
           └─4140 /usr/sbin/apache2 -k start
```

4. Stop the Apache web server.

CONSOLE

```
$ sudo systemctl stop apache2
```

5. Restart the Apache web server.

CONSOLE

```
$ sudo systemctl restart apache2
```

Create a new Apache Virtual Host

The Apache web server delivers web applications using virtual host configurations on your server. The default virtual host listens for incoming

connections using all server IP addresses on the default HTTP port `80`. Follow the steps below to disable the default virtual host and create a new virtual host to listen for connection requests using your domain on the default HTTP port `80`.

1. Create a new Apache virtual host configuration file in the `/etc/apache2/sites-available/` directory using a text editor such as `nano`. For example, `website.conf`.

CONSOLE

```
$ sudo nano /etc/apache2/sites-available/website.conf
```

2. Add the following contents to the file. Replace `app.example.com` with your domain and `webmaster@example.com` with your web administrator email.

APACHECONF

```
<VirtualHost *:80>
  ServerAdmin webmaster@example.com
  ServerName app.example.com

  DocumentRoot /var/www/html/website
  DirectoryIndex index.html index.php

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

  <Directory /var/www/html/website>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
  </Directory>
</VirtualHost>
```

Save and close the file.

The above Apache configuration creates a new virtual host that delivers web application files from the `/var/www/html/website` directory using your domain `app.example.com`. Within the configuration:

- `<VirtualHost *:80>`: Sets the virtual host listening address. The value `80` enables the HTTP port `80`.
- `ServerAdmin webmaster@example.com`: Specifies the web server administrator email address to send critical errors and alerts.
- `ServerName example.com`: Sets the virtual host domain to use for delivering web applications on your server.
- `DocumentRoot /var/www/html/website`: Sets the virtual host web root directory that contains the web application files to serve on your domain.
- `DirectoryIndex index.html`: Specifies the default index web application file to serve when the virtual host domain `app.example.com` is accessed.
- `ErrorLog ${APACHE_LOG_DIR}/error.log`: Sets a custom log file to store your virtual host error logs.
- `CustomLog ${APACHE_LOG_DIR}/access.log combined`: Sets the virtual host access log file location using the combined log format.
- `<Directory /var/www/html/example>`: Defines the configurations to apply on the web root directory requests.
- `Options Indexes FollowSymLinks`: Allows directory listing `Indexes` and symbolic links `FollowSymLinks`.
- `AllowOverride All`: Allows `.htaccess` files to override the Apache virtual host configurations.
- `Require all granted`: Allows all domain users to access the web root directory files.

3. Disable the default Apache virtual host configuration.

CONSOLE

```
$ sudo a2dissite 000-default
```

4. Enable the new Apache virtual host configuration.

CONSOLE

```
$ sudo a2ensite website
```

5. Test the Apache configuration for errors.

CONSOLE

```
$ sudo apachectl configtest
```

Output:

```
Syntax OK
```

6. Create the web root directory `/var/www/html/website` referenced in your virtual host configuration.

CONSOLE

```
$ sudo mkdir -p /var/www/html/website
```

7. Create a new sample HTML application `index.html` file in the web root directory.

CONSOLE

```
$ sudo nano /var/www/html/website/index.html
```

8. Add the following contents to the file.

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Apache Web Server</title>
</head>
<body>
  <h1>Greetings from Vultr</h1>
```

```
</body>
</html>
```

Save and close the file.

The above HTML code displays `Greetings from Vultr` message when the application is accessed using your virtual host domain `app.example.com`

9. Grant the web server user and group `www-data` ownership privileges to the `/var/www/html/website` web root directory.

```
CONSOLE
```

```
$ sudo chown -R www-data:www-data /var/www/html/website
```

10. Restart the Apache web server to apply your configuration changes.

```
CONSOLE
```

```
$ sudo systemctl restart apache2
```

Secure the Apache Web Server

The Apache web server listens for connection requests on the HTTP port `80` by default. HTTP enables unencrypted requests between your web server and a user's browser. HTTPS enables encrypted connections using valid SSL certificates to authenticate the web server with the user's browser. Follow the steps below to secure the Apache web server with trusted Let's Encrypt SSL certificates to enable encrypt network connections and forward all HTTP requests to HTTPS.

1. Install the Certbot Let's Encrypt client package using the Snap package manager.

```
CONSOLE
```

```
$ sudo snap install certbot --classic
```

2. Request a new Let's Encrypt SSL certificate on your server using your virtual host domain. Replace `app.example.com` with your actual domain and `hello@example.com` with your email address.

CONSOLE

```
$ sudo certbot --apache --agree-tos --redirect -d
app.example.com -m hello@example.com
```

Your output should be similar to the one below when successful.

```
Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/app.example.com/fullchain.pem
Key is saved at:          /etc/letsencrypt/live/app.example.com/privkey.pem
This certificate expires on 2024-10-15.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in
the background.

Deploying certificate
Successfully deployed certificate for app.example.com to /etc/apache2/sites-
available/website-le-ssl.conf
Congratulations! You have successfully enabled HTTPS on https://app.example.com
```

3. Test the Certbot SSL certificate renewal process.

CONSOLE

```
$ sudo certbot renew --dry-run
```

Output:

```
-----
Processing /etc/letsencrypt/renewal/publishqueue.ddns.net.conf
-----
Account registered.
Simulating renewal of an existing certificate for publishqueue.ddns.net

-----
Congratulations, all simulated renewals succeeded:
```

```
/etc/letsencrypt/live/publishqueue.ddns.net/fullchain.pem (success)  
-----
```

- Restart the Apache web server to apply your SSL configuration changes.

CONSOLE

```
$ sudo systemctl restart apache2
```

Set Up Firewall Rules

Uncomplicated Firewall (UFW) is active and enabled on Vultr Ubuntu 24.04 servers by default. Follow the steps below to configure the UFW utility and allow network connections using the Apache web server firewall profile.

- List all available UFW application profiles.

CONSOLE

```
$ sudo ufw app list
```

Verify that the default Apache profile `apache` is available similar to the following output:

```
Apache  
Apache Full  
Apache Secure  
OpenSSH
```

- Allow the `Apache Full` profile to enable both HTTP port `80` and HTTPS port `443` connections through the firewall.

CONSOLE

```
$ sudo ufw allow 'Apache Full'
```

- Reload UFW to apply the firewall changes.

```
CONSOLE
```

```
$ sudo ufw reload
```

4. View the firewall status and verify that the new rules are available.

```
CONSOLE
```

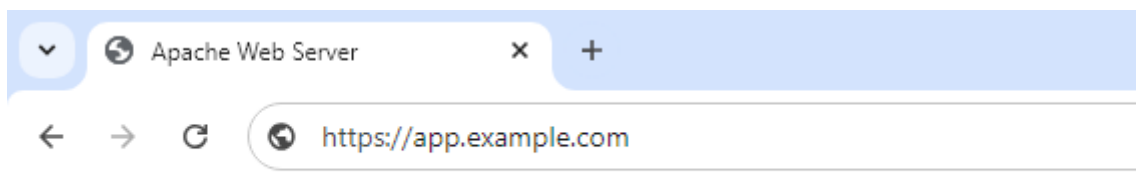
```
$ sudo ufw status
```

Output:

To	Action	From
--	-----	----
22/tcp	ALLOW	Anywhere
Apache Full	ALLOW	Anywhere
22/tcp (v6)	ALLOW	Anywhere (v6)
Apache Full (v6)	ALLOW	Anywhere (v6)

5. Access your virtual host domain `app.example.com` in a new web browser window and verify that Apache serves your HTML application with a `Greetings from Vultr` message.

```
https://app.example.com
```



Greetings from Vultr

Conclusion

You have installed the Apache web server on an Ubuntu 24.04 server. Apache enables you to host static websites and integrate with dynamic content processors such as PHP to serve modern web applications such as WordPress on your server. In addition, you can use the Apache web server as a reverse proxy to securely deliver your backend services using the `mod_proxy` extension on your server. For more information and configuration options, visit the [Apache documentation](#).



VULTR

