Web server Setup (Apache, Nginx)

Last updated by | Shakibe Hasan | Jun 19, 2023 at 12:43 PM GMT+6

What is a Web Server?

A web server is a software application or computer system that serves web pages to clients over the Internet or an intranet. It handles the delivery of web content, such as HTML documents, images, videos, and other resources, in response to client requests.

When a user enters a web address (URL) in a web browser, the browser sends a request to the web server hosting that website. The web server processes the request and sends back the requested content to the client, which is then displayed in the browser.

Web servers use the HTTP (Hypertext Transfer Protocol) to communicate with clients and transfer data. The most commonly used web server software is Apache HTTP Server, followed by Nginx, Microsoft IIS (Internet Information Services), and others.

Apache Web Server

Apache HTTP Server, commonly referred to as Apache, is one of the most widely used web server software applications in the world. It is an open-source and cross-platform web server that powers a significant portion(33%) of websites on the Internet.

Apache offers a robust and feature-rich platform for serving web content. It supports multiple operating systems, including Unix-based systems (such as Linux and macOS) and Windows. Apache is known for its stability, scalability, and extensive configuration options, making it suitable for a wide range of web hosting environments, from small personal websites to large enterprise setups.

Key features and capabilities of Apache web server include:

- **HTTP Protocol Support:** Apache supports the HTTP/1.1 protocol, which is the foundation of modern web communication. It handles client requests, delivers web content, and manages connections between clients and servers.
- **Virtual Hosting:** Apache allows hosting multiple websites on a single server by configuring virtual hosts. Each virtual host can have its own domain name, IP address, and configuration, enabling efficient use of server resources.
- **SSL/TLS Encryption:** Apache can be configured to enable secure communication using SSL/TLS protocols. This allows websites to use HTTPS, encrypting the data transmitted between the server and clients to ensure confidentiality and integrity.
- **URL Rewriting:** Apache provides powerful URL rewriting capabilities through modules like mod_rewrite. This feature enables the modification of URLs on-the-fly, which is useful for creating user-friendly and search engine optimized URLs.
- **Dynamic Content:** Apache supports server-side scripting languages like PHP, Perl, Python, and others. It integrates with these languages to execute scripts and generate dynamic web content, allowing the creation of interactive websites and web applications.
- **Authentication and Access Control:** Apache offers various authentication methods, including basic authentication, digest authentication, and more. It also supports access control through user and group permissions, allowing administrators to control access to specific directories or files.
- **Logging and Monitoring:** Apache logs detailed information about server activity, client requests, errors, and more. This logging functionality helps administrators troubleshoot issues, monitor performance, and analyze traffic patterns.

Apache web server provides a reliable and flexible platform for hosting websites and serving web content with a range of powerful features and customization options.

What Is Nginx?

Nginx (pronounced "engine-x") is a popular open-source web server and reverse proxy server software. Originally created to address performance issues with traditional web servers, Nginx has gained significant popularity due to its high performance, scalability, and efficient resource utilization.

Key features and capabilities of Nginx

1. **Web Server:** Nginx can function as a standalone web server similar to Apache. It can serve static content, such as HTML, CSS, JavaScript files, images, and videos, directly to clients without the need for additional modules or configurations. Nginx is known for its ability to handle a large number of concurrent connections efficiently.

2. **Reverse Proxy:** Nginx is often used as a reverse proxy server, acting as an intermediary between clients and backend servers. It receives client requests and forwards them to the appropriate backend server to process the request. This enables load balancing, improved security, and easier management of multiple backend servers.

3. **Load Balancer:** Nginx includes load balancing capabilities, allowing it to distribute incoming client requests across multiple backend servers. This improves the overall performance and reliability of websites and applications by distributing the workload evenly across the servers.

4. **Caching:** Nginx can cache frequently requested content, such as images or static files, to improve response times and reduce the load on backend servers. This feature is particularly useful for serving static content in high-traffic scenarios.

5. **SSL/TLS Termination:** Nginx can handle SSL/TLS encryption and decryption, acting as an SSL/TLS termination point. It offloads the computational burden of SSL/TLS encryption from backend servers, improving their performance and reducing resource usage.

6. **High Scalability:** Nginx is designed to be highly scalable and efficient. It uses an event-driven, asynchronous architecture that allows it to handle a large number of concurrent connections with minimal resource consumption.

7. **Configuration:** Nginx's configuration is based on a declarative syntax that is easy to understand and work with. The configuration files define how Nginx should process incoming requests, handle different types of content, and interact with backend servers.

8. **Extensions and Modules:** Nginx supports various extensions and modules that extend its functionality. These include additional load balancing algorithms, security features, caching mechanisms, and integration with other technologies such as PHP-FPM, WebSocket, and more.

Apache vs NGINX

- Since Apache uses the thread-based structure, owners of high-traffic websites may encounter performance problems. Nginx is one of the web servers that address the c10k problem and probably the most successful one.
- Nginx has an event-driven architecture that doesn’t create a new process for each request. Instead, it handles every incoming request in a single thread. This master process manages several worker processes that perform the actual processing of requests. The event-based model of Nginx distributes user requests among worker processes in an efficient way, therefore leading to much better scalability.
- If we need to manage a high-traffic website, Nginx is an excellent choice, as it can do that by using minimal resources. It cannot be a coincidence that it’s used by many high-visibility websites such as Netflix, Hulu, Pinterest, and Airbnb.

- However, for small and medium players, Apache comes with a handful of advantages over Nginx, such as its easy configuration, lots of modules, and a beginner-friendly environment.

How to install Apache web server in Linux

Step 1: Log in to Linux Server

First access the command terminal of your Linux server where you want to install the Apache webserver.

Step 2: Run system update

- {sudo apt update && sudo apt upgrade}

Step 3: Install Apache on Linux

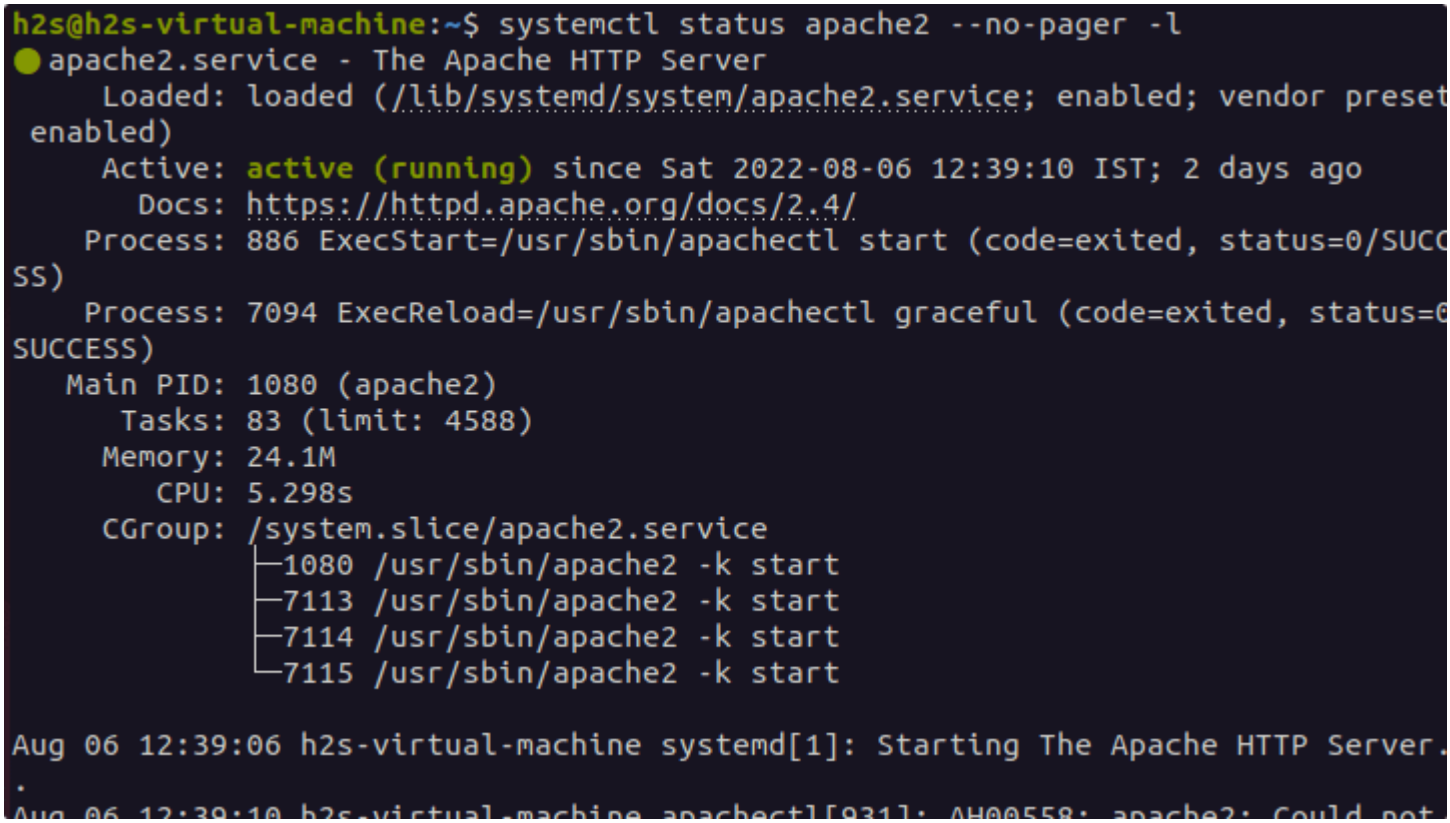
- {sudo apt install apache2}

Step 4: Enable and check service

- {sudo systemctl enable --now apache2}

To check whether the service is running absolutely fine or not, run:

- {systemctl status apache2 --no-pager -l}



Here, "--no-pager" used in Unix-like operating systems to disable the pager program, which is a utility that allows users to view files or command output one page at a time. By using "--no-pager", we're instructing the command to bypass the pager and display the entire output on the console without pausing.
"-l": This option is commonly used with the "ls" command, which lists directory contents.
By combining "--no-pager" and "-l", it seems we intend to execute a command that lists directory contents in a long format while ensuring that the output is displayed entirely without invoking a pager.

How to host a website in Apache:-

Step 1: Install Apache Server on Linux
{sudo apt-get update
sudo apt-get install apache }

Step 2: Verify Apache Service Status
If we are unsure about our server's address,by running the {hostname -i} command we can print the details.

```
File  Edit  View  Search  Terminal  Help
devops@devops-Lenovo-G410:~$ hostname -i
127.0.1.1
devops@devops-Lenovo-G410:~$ apache2 -version
Server version: Apache/2.4.52 (Ubuntu)
Server built:   2023-03-01T22:43:55
devops@devops-Lenovo-G410:~$
```

Step 3: Configure Firewall to Allow Apache Server Access
Following command to list all Apache application profiles:
{sudo ufw app list}

To allow UFW access on only port 80 as follows:
{sudo ufw allow 'Apache'}

To, check the firewall status by running:
{sudo ufw status}

Step 4: Understand Apache Directories and Files

- /var/log/apache2/error.log: Logs all the errors encountered
- /var/log/apache2/access.log: Logs all the access requests made to the server
- /etc/apache2/sites-available: Directory that contains virtual hosts
- /etc/apache2/sites-enabled: Stores ready to serve websites per virtual host. It cannot work without linking the configuration file inside the sites-available directory using the a2ensite command.

An Example to Set Up a Virtual Host

- Creating a domain directory:

```
devops@devops-Lenovo-G410: ~/Desktop
File  Edit  View  Search  Terminal  Help
devops@devops-Lenovo-G410:~/Desktop$ sudo cp -r QUICKLY /var/www
```

- Changing the ownership and file permissions of the directory using chown

```
devops@devops-Lenovo-G410: ~/Desktop
File  Edit  View  Search  Terminal  Help
devops@devops-Lenovo-G410:~/Desktop$ sudo cp -r QUICKLY /var/www
devops@devops-Lenovo-G410:~/Desktop$ sudo chown -R $current_user:$current_user /
var/www/QUICKLY
devops@devops-Lenovo-G410:~/Desktop$ sudo chmod -R 755 /var/www/QUICKLY
devops@devops-Lenovo-G410:~/Desktop$
```

- Creating a configuration folder that serves as a storage place to contain a record of the virtual hosts.[The default configuration file is /etc/apache2/sites-available/000-default.conf.]

```
devops@devops-Lenovo-G410:~/Desktop$ touch quickly.conf
devops@devops-Lenovo-G410:~/Desktop$ sudo cp quickly.conf /etc/apache2/sites-ava
ilable
```

devops@devops-Lenovo-G410: /etc/apache2/sites-available

File Edit View Search Terminal Help

```
  GNU nano 6.2                          quickly.conf
<VirtualHost *:80>
#ServerAdmin admin@host_example
ServerName quickly.local.com
#ServerAlias www.host_example
DocumentRoot /var/www/QUICKLY
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

- Activating Domain Configuration File

devops@devops-Lenovo-G410: /

File Edit View Search Terminal Help

```
devops@devops-Lenovo-G410:/$ sudo a2ensite quickly.conf
Enabling site quickly.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

- Hosts

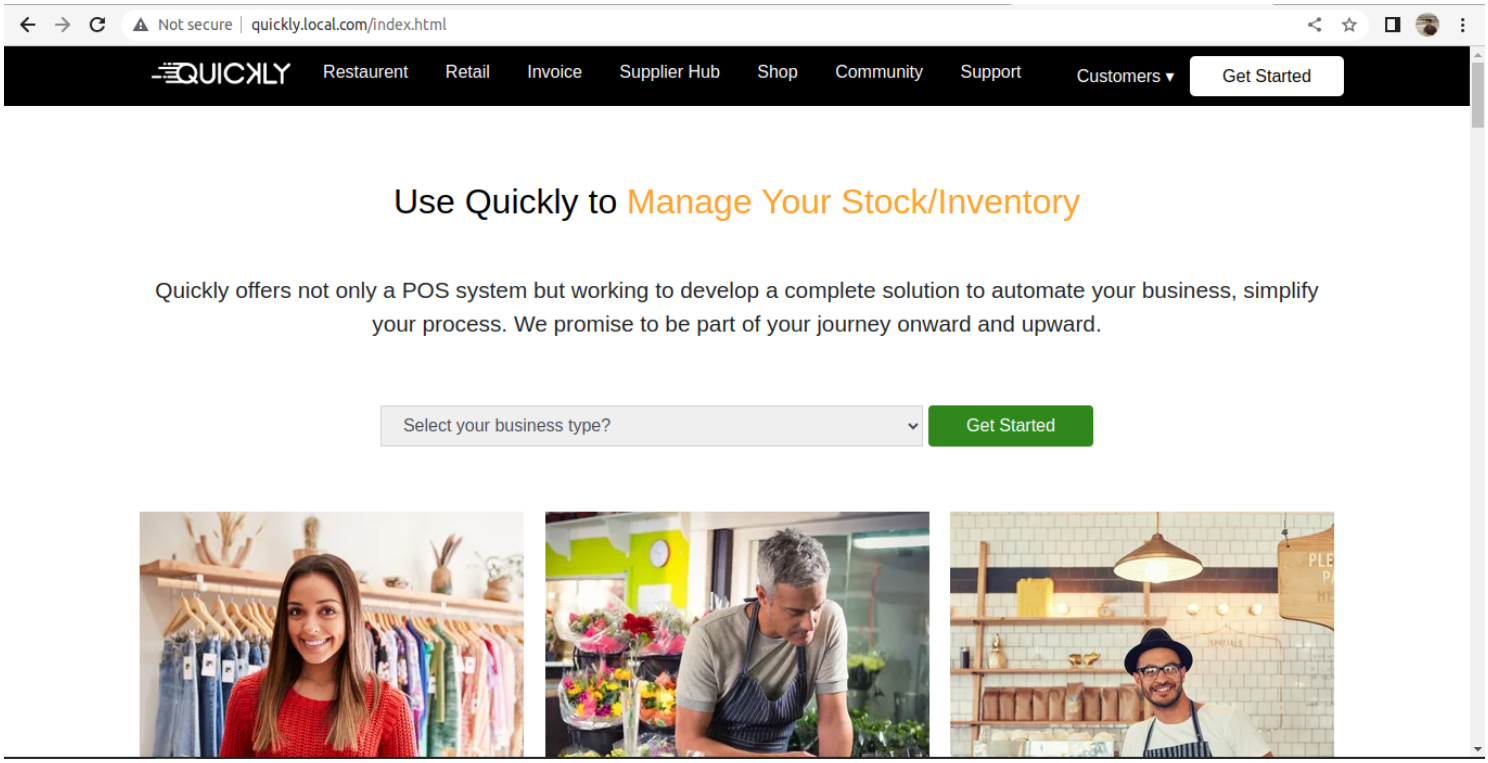
devops@devops-Lenovo-G410: /

File Edit View Search Terminal Help

```
  GNU nano 6.2                                                    /etc/hosts *
127.0.0.1       localhost host.localhost.com shakibe.localhost.com quickly.loacl.com
127.0.1.1       devops-Lenovo-G410

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

```
devops@devops-Lenovo-G410:/$ sudo nano /etc/hosts
devops@devops-Lenovo-G410:/$ sudo systemctl restart apache2
devops@devops-Lenovo-G410:/$
```



Ref: https://www.hostinger.com/tutorials/what-is-apache