Web Server Configuration - Apache & Nginx

Module 1: Introduction to Web Server Configuration

1. Overview of Web Servers: Apache and Nginx

- **What is a Web Server?**
  - Software that delivers web content to users via HTTP or HTTPS protocols.
  - Acts as a bridge between client requests and server-side resources.
- **Introduction to Apache**
  - Developed by the Apache Software Foundation.
  - Known for its flexibility, module-based architecture, and widespread usage.
  - Supports dynamic content and multiple processing modes (e.g., MPM prefork, MPM worker).
  -
- **Introduction to Nginx ("engine x")**
  - Originally designed as a reverse proxy and load balancer.
  - Lightweight, high-performance web server with event-driven architecture.
  - Ideal for handling high traffic with low resource usage.

2. Role of Web Servers in Hosting Environments

- **Hosting Websites**
  - Serves static and dynamic content.
  - Handles client requests and delivers appropriate responses.
- **Reverse Proxy**

    - Balances traffic between backend servers.
    - Enhances security by hiding backend infrastructure.
- **Caching**
    - Improves website performance by storing frequently accessed data.
- **Logging and Monitoring**
    - Tracks server activity for troubleshooting and optimization.

3. Key Differences Between Apache and Nginx

Feature	Apache	Nginx
Architecture	Process-based	Event-driven
Performance	Suitable for dynamic content	Excels at serving static content
Configuration	.htaccess for per-directory configs	Centralized configuration
Resource Usage	Higher	Lower
Use Case	Flexible, dynamic content	High-traffic static content, reverse proxy

4. Installation and Initial Setup of Apache and Nginx

Using AlmaLinux

- **Update System Packages**

  sudo dnf update -y

- **Install Apache**

  sudo dnf install httpd -y

```
[root@king ~]# dnf install httpd -y
Last metadata expiration check: 0:01:33 ago on Tue 28 Jan 2025 08:06:38 EST.
Dependencies resolved.
================================================================================
 Package                  Architecture Version                                  Repository       Size
================================================================================
Installing:
 httpd                    x86_64       2.4.37-65.module_el8.10.0+3874+c2064c23.2 appstream       1.4 M
Installing dependencies:
 almalinux-logos-httpd    noarch       84.5-1.el8                               appstream        29 k
 apr                      x86_64       1.6.3-12.el8                             appstream       128 k
 apr-util                 x86_64       1.6.1-9.el8                              appstream       105 k
 httpd-filesystem         noarch       2.4.37-65.module_el8.10.0+3874+c2064c23.2 appstream        44 k
 httpd-tools              x86_64       2.4.37-65.module_el8.10.0+3874+c2064c23.2 appstream       111 k
 mailcap                  noarch       2.1.48-3.el8                             baseos           39 k
 mod_http2                x86_64       1.15.7-10.module_el8.10.0+3904+caabc2d4.1 appstream       155 k
Installing weak dependencies:
 apr-util-bdb             x86_64       1.6.1-9.el8                              appstream        24 k
 apr-util-openssl         x86_64       1.6.1-9.el8                              appstream        26 k
```

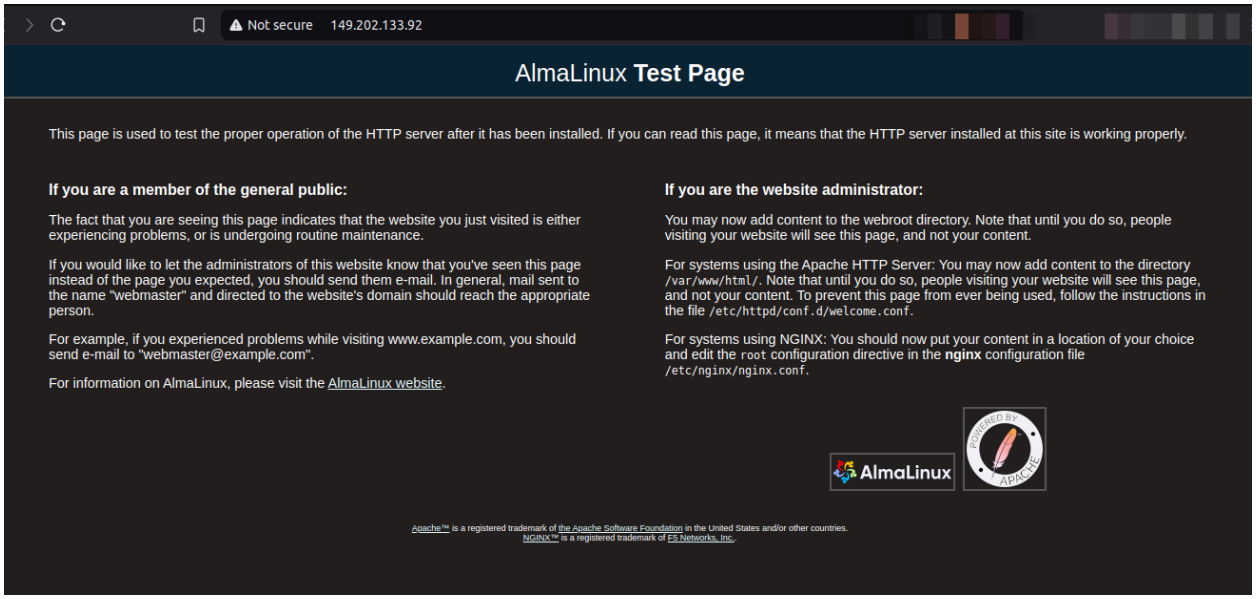
sudo systemctl enable httpd

sudo systemctl start httpd

sudo systemctl status httpd

```
[root@king ~]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@king ~]# systemctl start httpd
[root@king ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
     Active: active (running) since Tue 2025-01-28 08:10:16 EST; 8s ago
       Docs: man:httpd.service(8)
   Main PID: 50180 (httpd)
     Status: "Started, listening on: port 80"
      Tasks: 213 (limit: 12013)
     Memory: 28.6M
     CGroup: /system.slice/httpd.service
             ├─50180 /usr/sbin/httpd -DFOREGROUND
             ├─50181 /usr/sbin/httpd -DFOREGROUND
```

Confirm by visiting your server IP on the Browser.



- **Install Nginx**

sudo dnf install nginx -y

```
[root@king ~]#
[root@king ~]# sudo dnf install nginx -y
Last metadata expiration check: 2:29:28 ago on Tue 28 Jan 2025 23:05:10 EST.
Dependencies resolved.
================================================================================
 Package              Arch   Version                                    Repo       Size
================================================================================
Installing:
 nginx                x86_64 1:1.14.1-9.module_el8.3.0+2165+af250afe.alma appstream 568 k
Installing dependencies:
 dejavu-fonts-common  noarch 2.35-7.el8                                 baseos     73 k
 dejavu-sans-fonts    noarch 2.35-7.el8                                 baseos    1.5 M
 fontconfig           x86_64 2.13.1-4.el8                               baseos    273 k
 fontpackages-filesystem
                      noarch 1.44-22.el8                                baseos     16 k
 gd                   x86_64 2.2.5-7.el8                                appstream 143 k
 jbigkit-libs         x86_64 2.1-14.el8                                 appstream  54 k
```

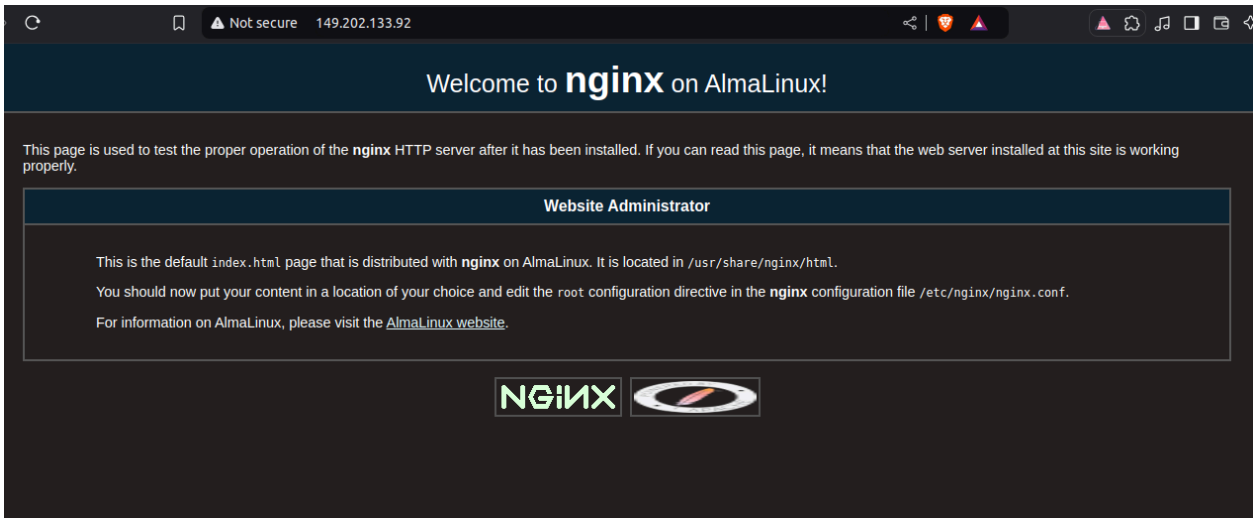
sudo systemctl enable nginx

sudo systemctl start nginx

sudo systemctl status nginx

```
Complete!
[root@king ~]#
[root@king ~]# sudo systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/syste
md/system/nginx.service.
[root@king ~]# sudo systemctl start nginx
[root@king ~]# sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disable
   Active: active (running) since Wed 2025-01-29 01:35:56 EST; 6s ago
  Process: 67563 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
  Process: 67561 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
  Process: 67560 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCES
 Main PID: 67565 (nginx)
    Tasks: 2 (limit: 12013)
   Memory: 3.7M
```

Confirm you can see the default Nginx page.



Using Ubuntu/Debian

- **Update System Packages**

  sudo apt update && sudo apt upgrade -y

- **Install Apache**

  sudo apt install apache2 -y

```
root@srv2:~#  apt install apache2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2 linux-headers-5.15.0-43 linux-headers-5.15.0-43-generic
  linux-image-5.15.0-43-generic linux-modules-5.15.0-43-generic
  linux-modules-extra-5.15.0-43-generic
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.3-0 mailcap mime-support ssl-cert
Suggested packages:
```

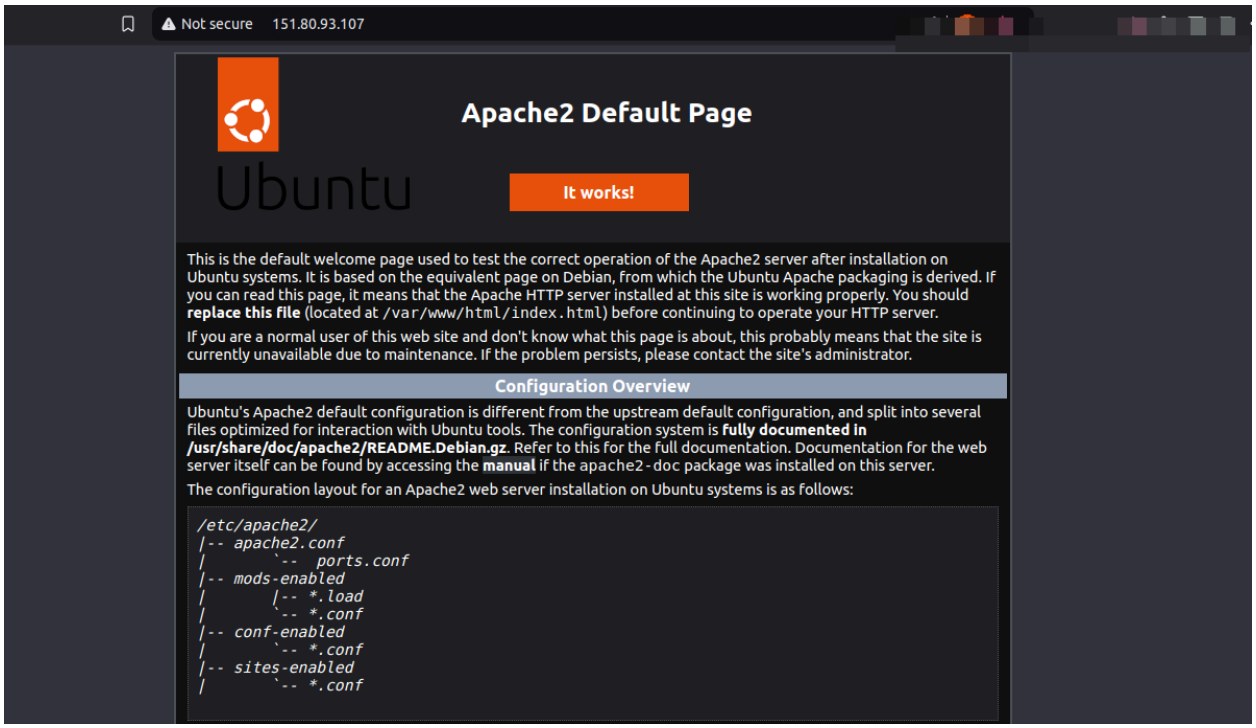
sudo systemctl enable apache2

sudo systemctl start apache2

systemctl status apache2* or *service apache2 status

```
root@srv2:~# sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-
sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable apache2
root@srv2:~# sudo systemctl start apache2
root@srv2:~# sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2025-01-28 13:08:38 UTC; 10min ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 65674 (apache2)
      Tasks: 55 (limit: 2180)
     Memory: 5.0M
        CPU: 67ms
```

Confirm by visiting your server IP on the Browser.



- **Install Nginx**

sudo apt install nginx -y

```
root@srv2:~# sudo apt install nginx -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2 linux-headers-5.15.0-43 linux-headers-5.15.0-43-generic
  linux-image-5.15.0-43-generic linux-modules-5.15.0-43-generic
  linux-modules-extra-5.15.0-43-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  fontconfig-config fonts-dejavu-core libdeflate0 libfontconfig1 libgd3 libjbig0
  libjpeg-turbo8 libjpeg8 libnginx-mod-http-geoip2 libnginx-mod-http-image-filter
  libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream
  libnginx-mod-stream-geoip2 libtiff5 libwebp7 libxpm4 nginx-common nginx-core
```

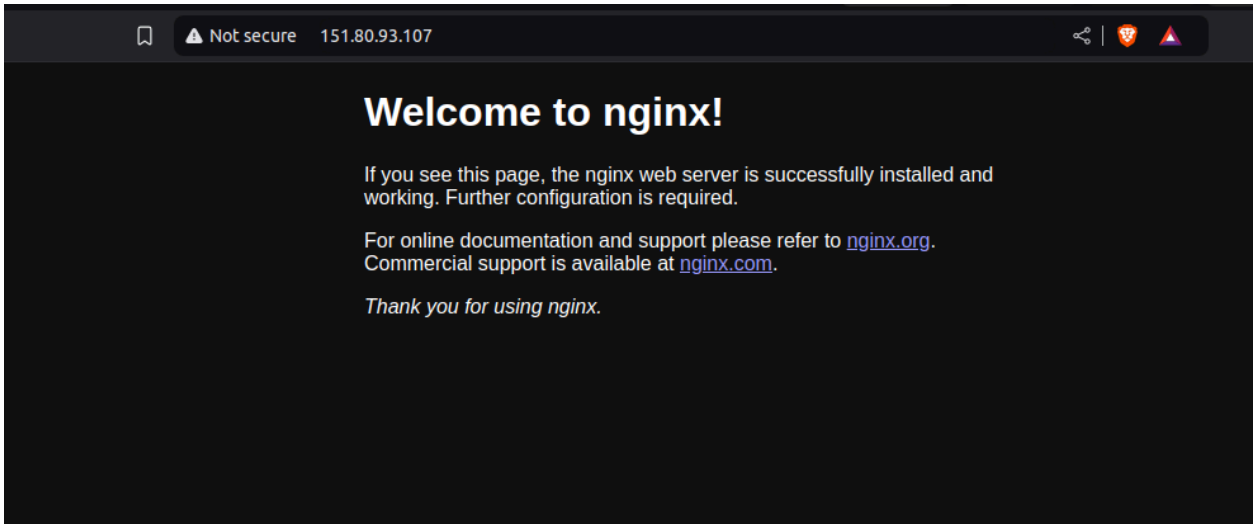
sudo systemctl enable nginx

sudo systemctl start nginx

sudo systemctl status nginx

```
root@srv2:~#
root@srv2:~# sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sy
sv-install.
Executing: /lib/systemd/systemd-sysv-install enable nginx
root@srv2:~# sudo systemctl start nginx
root@srv2:~# sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2025-01-29 06:41:11 UTC; 1min 23s ago
       Docs: man:nginx(8)
   Main PID: 78348 (nginx)
      Tasks: 2 (limit: 2180)
     Memory: 4.6M
        CPU: 20ms
     CGroup: /system.slice/nginx.service
```

Confirm the Nginx Page on Ubuntu is accessible.



5. Tools and Utilities for Managing Web Servers

- **Apache Utilities**
  - ***apachectl***: Controls Apache services.
  - ***htpasswd***: Manages basic HTTP authentication.
- **Nginx Utilities**
  - ***nginx -t***: Tests Nginx configurations.
  - ***nginx -s reload***: Reloads configurations without downtime.
- **Log Monitoring Tools**
  - ***tail, grep,*** and awk for analyzing log files.
  - Tools like GoAccess for real-time log analysis.
- **Performance Monitoring Tools**
  - ***top and htop*** for resource monitoring.

Tools and Utilities for Managing Web Servers (Detailed Overview)

1. Apache-Specific Utilities

- ***apachectl*** **(Apache Control Tool):**
  - A command-line tool to manage Apache processes.
- Common Commands:

```
apachectl start      # Starts the Apache server
```

apachectl stop # Stops the Apache server

apachectl restart # Restarts the Apache server

apachectl graceful # Restarts without disconnecting current clients

apachectl configtest # Tests the configuration file syntax

    - Useful for quickly applying changes or troubleshooting Apache configurations.
- **htpasswd:**
    - Used to manage HTTP basic authentication for Apache.
- Common Usage:

htpasswd -c /etc/apache2/.htpasswd username # Create a new user

htpasswd /etc/apache2/.htpasswd username # Add/update a user

    - Protect directories or files using ***.htpasswd*** with the .htaccess file.
- **mod_status:**
    - A module to monitor Apache activity and performance in real-time.

http://your-server-ip/server-status

    - Requires enabling the module and configuring access control.
    - ***Run : apachectl -M | grep status***
    - *The above command checks if the module is enabled.*

---

2. Nginx-Specific Utilities

- ***nginx -t*:**
    - Tests Nginx configuration files for syntax errors before reloading or restarting.
- Example:

  sudo nginx -t

    - Helps avoid downtime caused by faulty configurations.
- ***nginx -s*:**
    - Sends signals to control the Nginx process.
- Common Commands:

  nginx -s stop # Stops the server
- **nginx -s reload** # Reloads the configuration without downtime
- **nginx -s quit** # Gracefully shuts down the server
    -
- **Log Directories:**
    - Access and error logs provide insights into server behavior.
    - Default Locations:
        - Access logs: ***/var/log/nginx/access.log***
        - Error logs: ***/var/log/nginx/error.log***
- Commands for monitoring:

  tail -f /var/log/nginx/access.log

---

3. Log Monitoring Tools

- ***tail, grep, awk:***
- ***tail***: Monitors logs in real-time.

  tail -f /var/log/apache2/error.log

- ***grep***: Searches for specific patterns in logs.

  grep "error" /var/log/nginx/error.log
  -
- **awk**: Processes and formats log data.

  awk '{print $1, $7}' /var/log/apache2/access.log
  -
- **GoAccess:**
  - Real-time log analyzer for web servers.
  - Features:
    - Visualizes data in a terminal or web interface.
    - Provides metrics like visitor counts, popular URLs, and HTTP status codes.
- Installation:

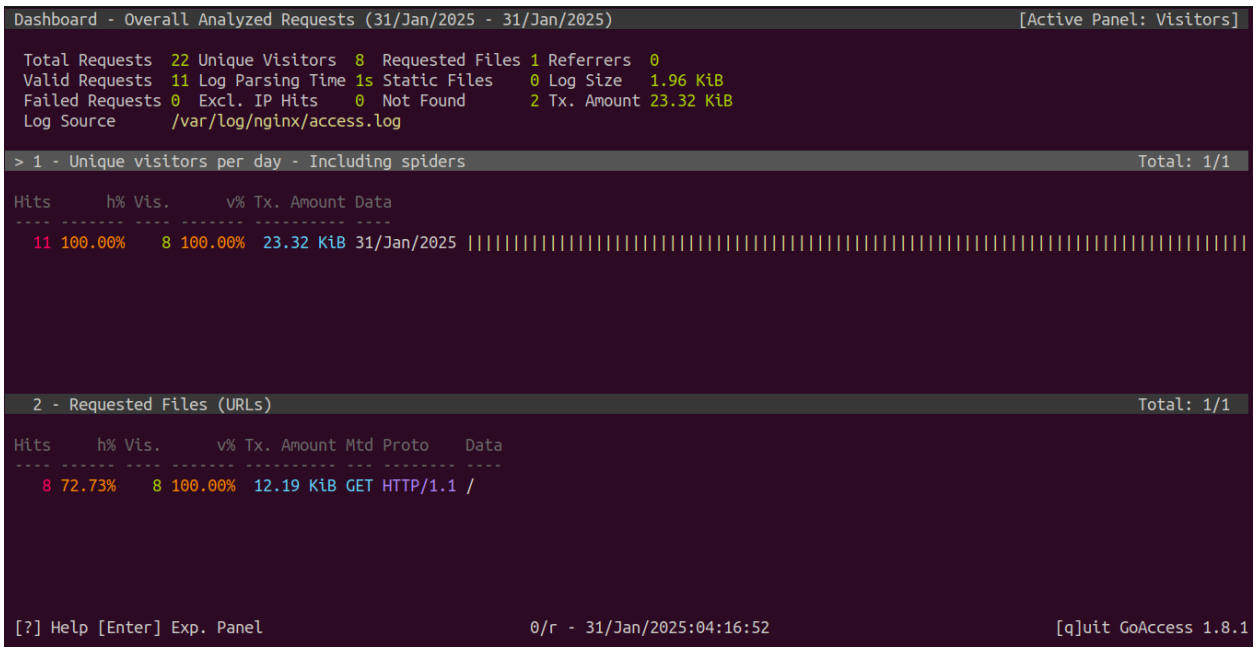
  sudo apt install goaccess # On Ubuntu/Debian

  sudo dnf install goaccess # On AlmaLinux

```
[root@king ~]# sudo dnf install goaccess
Last metadata expiration check: 1:31:00 ago on Fri 31 Jan 2025 02:44:37 EST.
Dependencies resolved.
================================================================================
 Package                 Architecture            Version                 Repository              Size
================================================================================
Installing:
 goaccess                x86_64                  1.8.1-1.el8             epel                   404 k
Installing dependencies:
 GeoIP                   x86_64                  1.6.12-7.el8            epel                   124 k
 GeoIP-GeoLite-data      noarch                  2018.06-5.el8           epel                   552 k

Transaction Summary
================================================================================
Install  3 Packages

Total download size: 1.1 M
Installed size: 2.9 M
```

- Run:

  goaccess /var/log/nginx/access.log



---

4. Performance Testing Tools

- **ApacheBench (ab):**
  - A command-line tool to benchmark web servers.
- Example:

  ab -n 100 -c 10 http://your-server-ip/

  - Tests 100 requests with 10 concurrent users.

  ***wrk*:**

    - A modern and powerful load testing tool for web servers.
- Example:

  wrk -t12 -c400 -d30s http://your-server-ip/

    - Simulates high traffic with 12 threads, 400 connections, for 30 seconds.

---

5. Configuration Editors

- ***vim*** **and** ***nano*****:**
    - Text editors for modifying configuration files.
- Example:

  sudo vim /etc/apache2/apache2.conf

  sudo nano /etc/nginx/nginx.conf

- **Certbot:**
    - Automates the installation of SSL certificates from Let's Encrypt.

Module 2: Configuring Virtual Hosts and Reverse Proxies

1. Understanding Virtual Hosts in Apache and Nginx

What are Virtual Hosts?

- Virtual Hosts allow a single web server to host multiple websites/domains.
- Each website has a separate configuration, including ***domain name, document root, and log files.***

Types of Virtual Hosts

- **Apache:**
  - **Name-based Virtual Hosts:** Multiple websites on one IP using different domain names.
  - **IP-based Virtual Hosts:** Separate IP addresses for each website.
- **Nginx:**
  - Uses **server blocks** (equivalent to Apache virtual hosts).
  - Supports multiple domain configurations in a single Nginx instance.

When to Use Virtual Hosts?

- Hosting multiple websites on one server.
- Separating development, staging, and production environments.
- Enforcing different configurations per site (e.g., SSL, caching, redirections).

2. Step-by-Step Guide to Configuring Virtual Hosts

Apache Configuration on AlmaLinux

Install Apache (if not installed)

sudo dnf install httpd -y

sudo systemctl enable --now httpd

Create a New Virtual Host File

Before you create your virtual hosts, you will need to create a **sites-available** directory to store them in. You will also create the ***sites-enabled*** directory that tells Apache that a virtual host is ready to serve to visitors. The sites-enabled directory will hold symbolic links to virtual hosts that we want to publish. Create both directories with the following command:

sudo mkdir /etc/httpd/sites-available /etc/httpd/sites-enabled

Next, you will tell Apache to look for virtual hosts in the sites-enabled directory. To accomplish this, edit Apache's main configuration file and add a line declaring an optional directory for additional configuration files:

sudo vi /etc/httpd/conf/httpd.conf

The httpd.conf is the main httpd(Apache configuration file here)

Add the line below to the end of the file:

IncludeOptional sites-enabled/*.conf

Add Virtual Host Configuration

Start by creating a new virtual hosts file in the **sites-available** directory:

sudo vi /etc/httpd/sites-available/your_domain.conf

Add the virtual host settings below (Replace with your actual domain name which must be pointed to the server IP already)

*<VirtualHost *:80>*

ServerName example.com

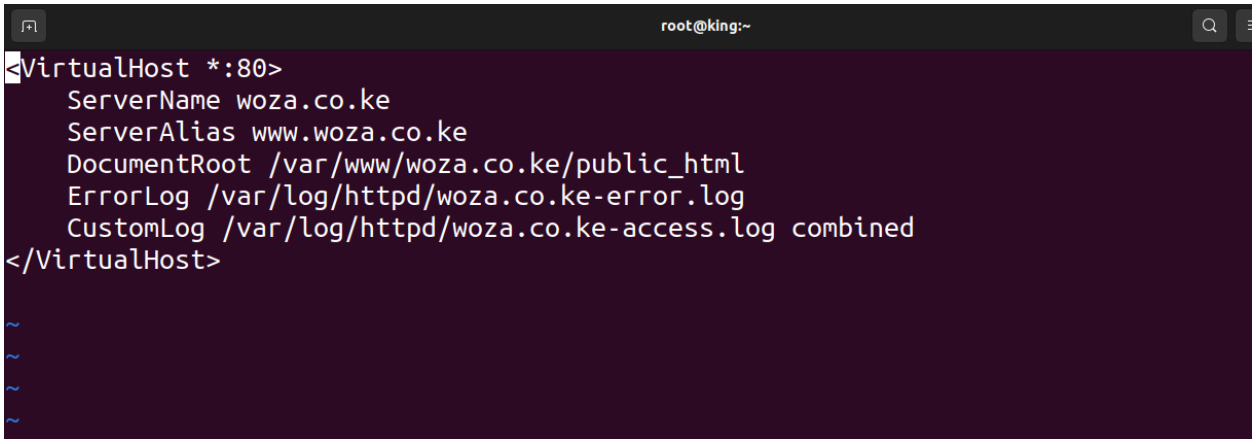
ServerAlias www.example.com

DocumentRoot /var/www/example.com/public_html

ErrorLog /var/log/httpd/example.com-error.log

CustomLog /var/log/httpd/example.com-access.log combined

</VirtualHost>



Create the Website Directory & Set Permissions

sudo mkdir -p /var/www/example.com/public_html

sudo chown -R apache:apache /var/www/example.com

First enable the site.

Now that you have created the virtual host files, you will enable them so that Apache knows to serve them to visitors. To do this, create a symbolic link for each virtual host in the sites-enabled directory:

sudo ln -s /etc/httpd/sites-available/your_domain.conf /etc/httpd/sites-enabled/your_domain.conf

```
[root@king ~]# sudo vi /etc/httpd/sites-available/woza.co.ke.conf
[root@king ~]# sudo ln -s /etc/httpd/sites-available/woza.co.ke.conf /etc/httpd/sites-enab
led/woza.co.ke.conf
[root@king ~]#
```

Adjusting Apache Policies Universally

sudo setsebool -P httpd_unified 1 (Only If you have selinux enabled)

In my case, it is disabled so no need to run the command above. You can check here using the command below.

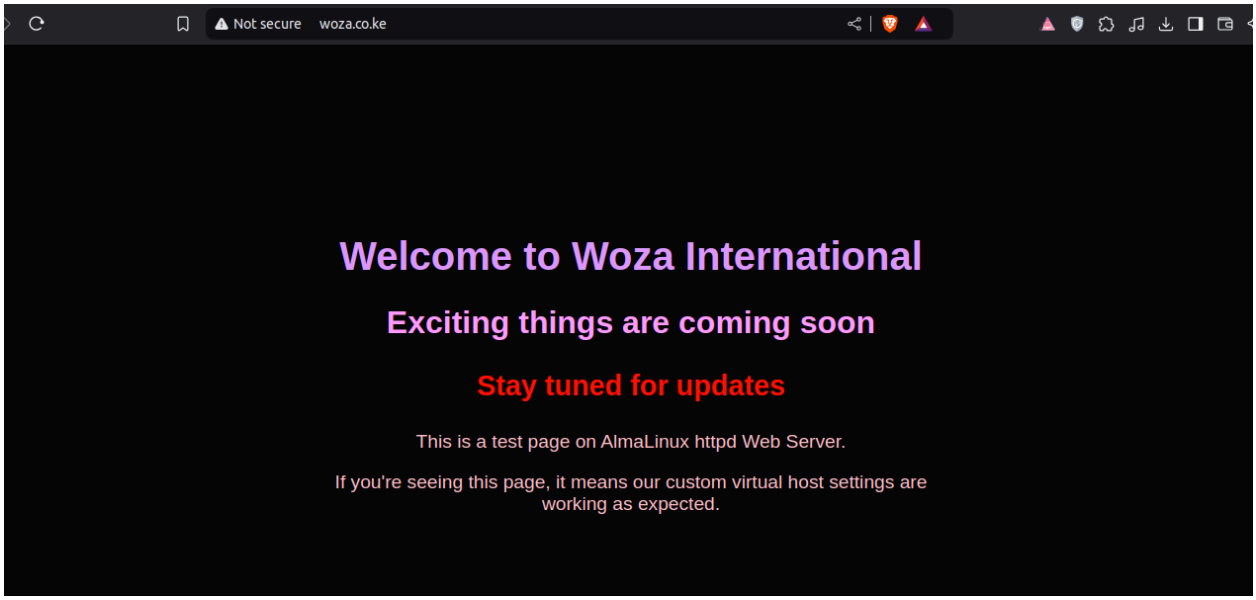
cat /etc/selinux/config

```
[root@king ~]# cat  /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted


[root@king ~]#
```

sudo systemctl restart httpd

Reload your website. You should now see your domain loading on the browser.

Note at the moment, our domain does not yet have an SSL certificate.



Apache Configuration on Ubuntu/Debian

Install Apache

sudo apt install apache2 -y

sudo systemctl enable --now apache2

Create a directory structure for your site.
sudo mkdir -p /var/www/your_domain.com/public_html

Adjust the permissions.

sudo chown -R $USER:$USER /var/www/ubuntu.woza.co.ke/public_html

sudo chmod -R 755 /var/www

Create Default Page for the Virtual Host

nano /var/www/yourdomain.com/public_html/index.html

```
root@srv2:~# sudo mkdir -p /var/www/ubuntu.woza.co.ke/public_html
root@srv2:~# sudo chown -R $USER:$USER /var/www/ubuntu.woza.co.ke/public_html
root@srv2:~# sudo chmod -R 755 /var/www
root@srv2:~# nano /var/www/
html/              ubuntu.woza.co.ke/
root@srv2:~# nano /var/www/ubuntu.woza.co.ke/public_html/index.html
root@srv2:~#
```

Creating New Virtual Host Files

sudo nano /etc/apache2/sites-available/your_domain_1.conf

Add Virtual Host Configuration

*<VirtualHost *:80>*

ServerName example.com

ServerAlias www.example.com

DocumentRoot /var/www/example.com/public_html

ErrorLog ${APACHE_LOG_DIR}/example.com-error.log

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

</VirtualHost>

```
  GNU nano 6.2        /etc/apache2/sites-available/ubuntu.woza.co.ke.conf
<VirtualHost *:80>
    ServerName ubuntu.woza.co.ke
    ServerAlias www.ubuntu.woza.co.ke
    DocumentRoot /var/www/ubuntu.woza.co.ke/public_html
    ErrorLog ${APACHE_LOG_DIR}/ubuntu.woza.co.ke-error.log
    CustomLog ${APACHE_LOG_DIR}/ubuntu.woza.co.ke-access.log combined
</VirtualHost>
```

1.

Enable the Virtual Host

First disable the default configuration

sudo a2dissite 000-default.conf

Enable your custom site.

sudo a2ensite example.com.conf

Reload the configuration

systemctl reload apache2

```
root@srv2:/etc/apache2/sites-available# sudo a2dissite 000-default.conf

Site 000-default disabled.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@srv2:/etc/apache2/sites-available# sudo a2ensite ubuntu.woza.co.ke.conf
Site ubuntu.woza.co.ke already enabled
root@srv2:/etc/apache2/sites-available# systemctl reload apache2
root@srv2:/etc/apache2/sites-available#
```

Test the configuration.

sudo apache2ctl configtest

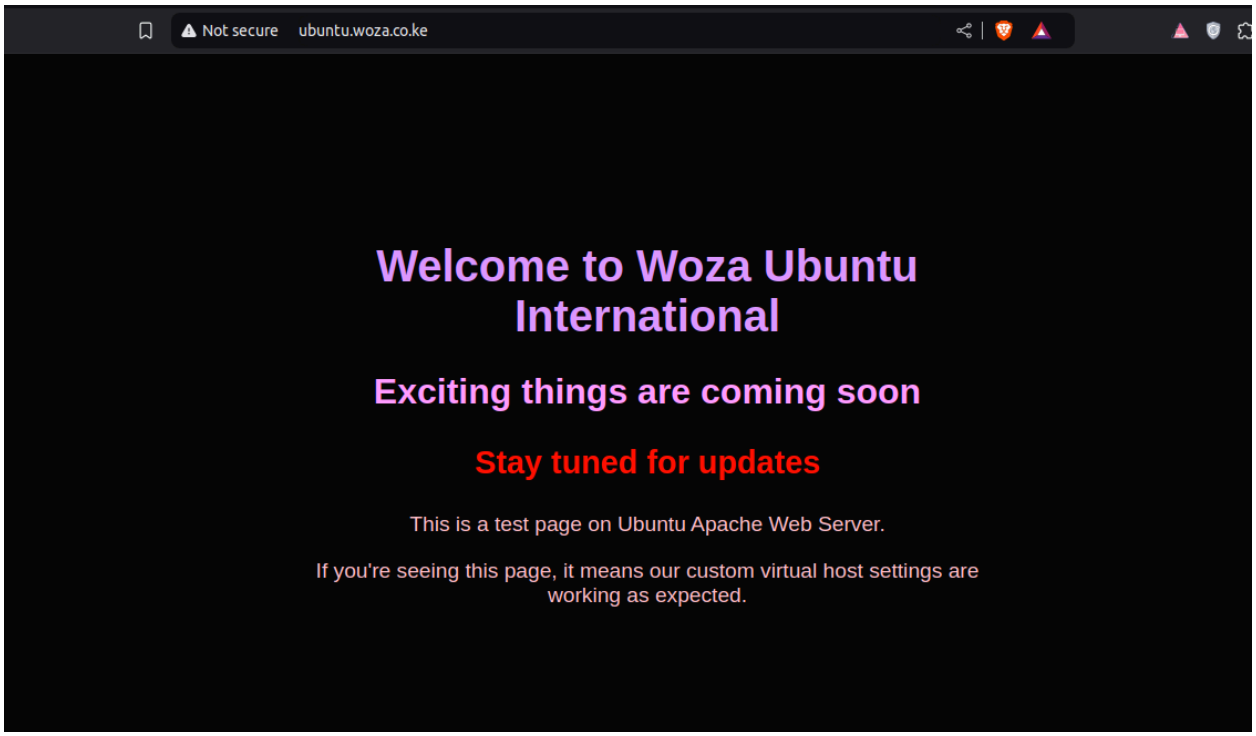
2.

Restart Apache
sudo systemctl restart apache2

Access your site.

Reload your website. You should now see your domain loading on the browser.

Note at the moment, our domain does not yet have an SSL certificate.



Nginx Configuration on AlmaLinux

Install Nginx

sudo dnf install nginx -y

sudo systemctl enable --now nginx

Create the Directory Structure

sudo mkdir -p /var/www/example.com/html

Grant Permissions

sudo chown -R nginx:nginx /var/www/example.com/html

sudo chmod -R 755 /var/www/example.com

Create a sample page for your site.

nano /var/www/example.com/html/index.html

Create server blocks

sudo nano /etc/nginx/conf.d/example.com.conf

Add the following configuration:

```
server {
    listen 80;
    server_name example.com www.example.com;
```

```
    root /var/www/example.com/public_html;
    index index.html index.htm index.php;

    access_log /var/log/nginx/example.com_access.log;
    error_log /var/log/nginx/example.com_error.log;

    location / {
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include fastcgi_params;
        fastcgi_pass unix:/run/php-fpm/www.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
    }
}
```

```
 GNU nano 2.9.8                    /etc/nginx/conf.d/woza.co.ke.conf

server {
    listen 80;
    server_name woza.co.ke www.woza.co.ke;

    root /var/www/woza.co.ke/public_html;
    index index.html index.htm index.php;

    access_log /var/log/nginx/woza.co.ke.access.log;
    error_log /var/log/nginx/woza.co.ke.error.log;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        include fastcgi_params;
        fastcgi_pass unix:/run/php-fpm/www.sock; # Update if using a different PHP version
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    }

                                [ Read 27 lines ]
^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      M-U Undo
^X Exit         ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell     ^_ Go To Line   M-E Redo
```

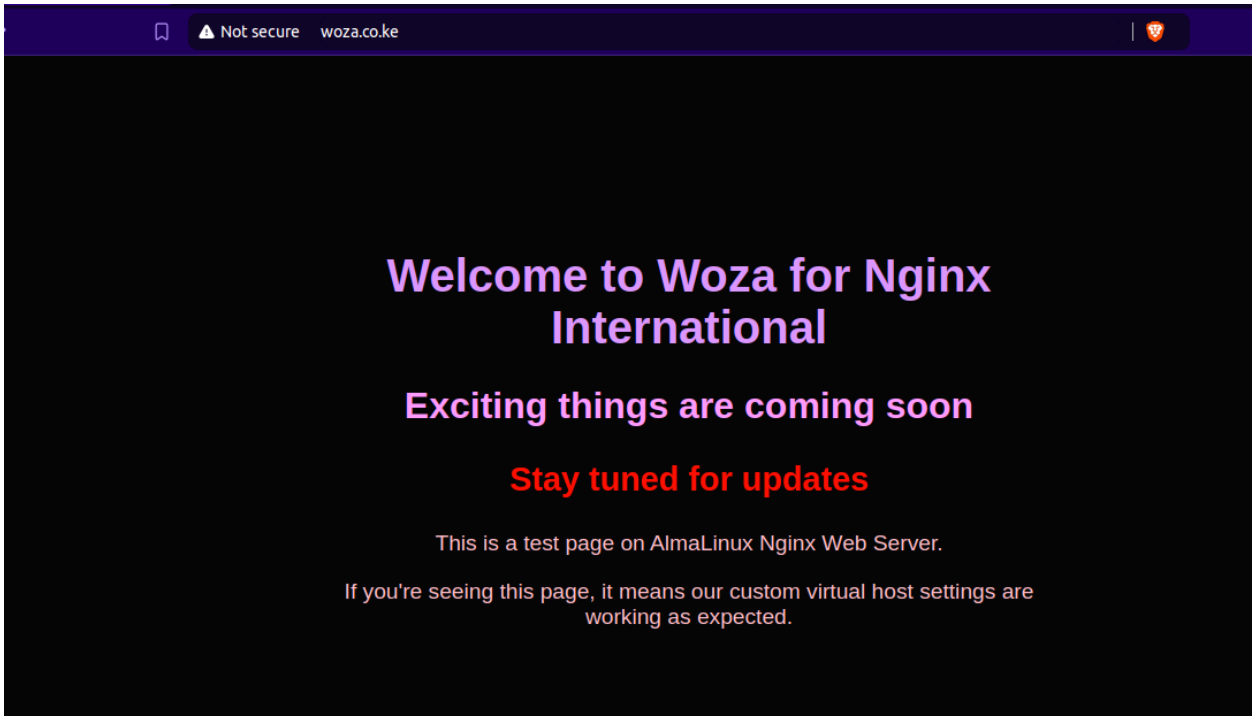
Test and Restart Nginx

sudo nginx -t

sudo systemctl restart nginx

```
[root@king ~]# sudo nano /etc/nginx/conf.d/woza.co.ke.conf
[root@king ~]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@king ~]# systemctl restart nginx
[root@king ~]#
```

Access the site.



Nginx Configuration on Ubuntu/Debian

Install Nginx

sudo apt install nginx -y

sudo systemctl enable --now nginx

Create the Directory Structure

sudo mkdir -p /var/www/example.com/public_html

Set the correct permissions:

sudo chown -R www-data:www-data /var/www/example.com

sudo chmod -R 755 /var/www/example.com

Create an Nginx Server Block

sudo nano /etc/nginx/sites-available/woza.co.ke

Add Configuration

```
server {
    listen 80;
    server_name example.com www.example.com;

    root /var/www/example.com/public_html;
    index index.html index.htm index.php;

    access_log /var/log/nginx/example.com_access.log;
    error_log /var/log/nginx/example.com_error.log;

    location / {
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include fastcgi_params;
        fastcgi_pass unix:/run/php/php-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
```

```
    }
}
```

Enable the Server Block

sudo ln -s /etc/nginx/sites-available/woza.co.ke /etc/nginx/sites-enabled/

Test the Nginx configuration:

sudo nginx -t

Restart Nginx:

sudo systemctl restart nginx

---

3. Reverse Proxy Basics

What is a Reverse Proxy?

- A reverse proxy sits between client requests and backend servers.
- Directs incoming traffic to the appropriate server, improving security and performance.

Common Use Cases

- Load balancing requests across multiple backend servers.
- Caching static files to improve speed.
- Protecting backend applications from direct exposure to the internet.

---

4. Configuring a Reverse Proxy

In this course, we shall not be configuring Reserver Proxy. This shall be covered in subsequent courses.

Summary

- Virtual Hosts/Server Blocks allow hosting multiple websites on one server.
- Apache and Nginx have different approaches but serve the same purpose.

Module 3: SSL/TLS Certificates and Secure Domain (Letsencrypt)

On AlmaLinux (Apache & Nginx)

sudo dnf install epel-release -y

```
[root@king ~]#
[root@king ~]# sudo dnf install epel-release -y
Last metadata expiration check: 2:33:44 ago on Wed 29 Jan 2025 23:20:37 EST.
Dependencies resolved.
================================================================================
 Package                Architecture      Version                Repository         Size
================================================================================
Installing:
 epel-release           noarch            8-21.el8               extras             24 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 24 k
Installed size: 34 k
Downloading Packages:
epel-release-8-21.el8.noarch.rpm                          907 kB/s |  24 kB     00:00
--------------------------------------------------------------------------------
```

sudo dnf install certbot python3-certbot-apache

```
[root@king ~]#
[root@king ~]# sudo dnf install certbot python3-certbot-apache -y

Extra Packages for Enterprise Linux 8 - x86_64           7.1 MB/s |  14 MB     00:01
Last metadata expiration check: 0:00:15 ago on Thu 30 Jan 2025 01:55:12 EST.
Dependencies resolved.
================================================================================
 Package                   Arch   Version                                Repo       Size
================================================================================
Installing:
 certbot                   noarch 1.22.0-1.el8                           epel       54 k
 python3-certbot-apache    noarch 1.22.0-1.el8                           epel      145 k
Installing dependencies:
 augeas-libs               x86_64 1.12.0-8.el8                           baseos    436 k
 mod_ssl                   x86_64 1:2.4.37-65.module_el8.10.0+3874+c2064c23.2
```

python3-certbot-nginx -y

-

On Ubuntu/Debian (Apache & Nginx)

bash

CopyEdit

```
sudo apt update

sudo apt install certbot python3-certbot-apache
python3-certbot-nginx -y
```

- **Step 2: Obtain SSL Certificate**

  For Apache

  ```
  sudo certbot --apache -d example.com -d
  www.example.com
  ```

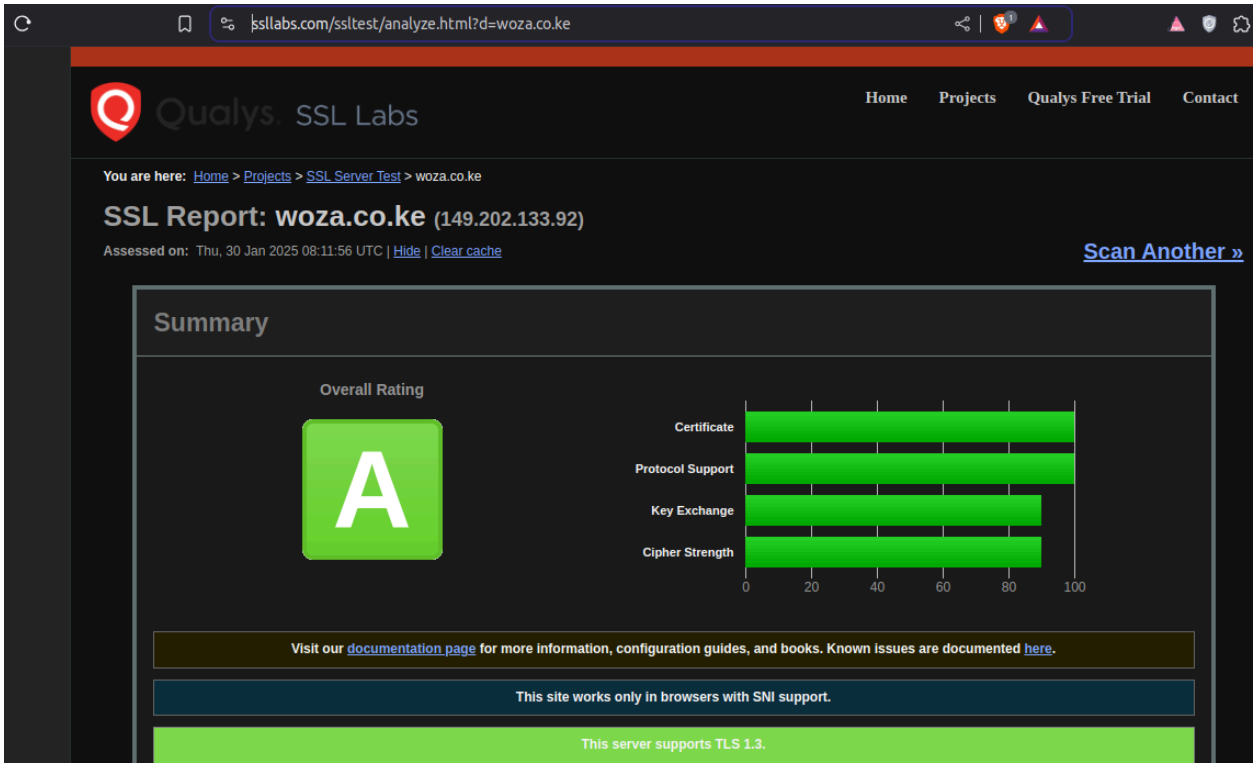
```
[root@king ~]# sudo certbot --apache -d woza.co.ke -d www.woza.co.ke
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
 (Enter 'c' to cancel): admin@woza.co.ke

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.4-April-3-2024.pdf. You must agree in
order to register with the ACME server. Do you agree?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: y

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: y
Account registered.
```

Fill in your details

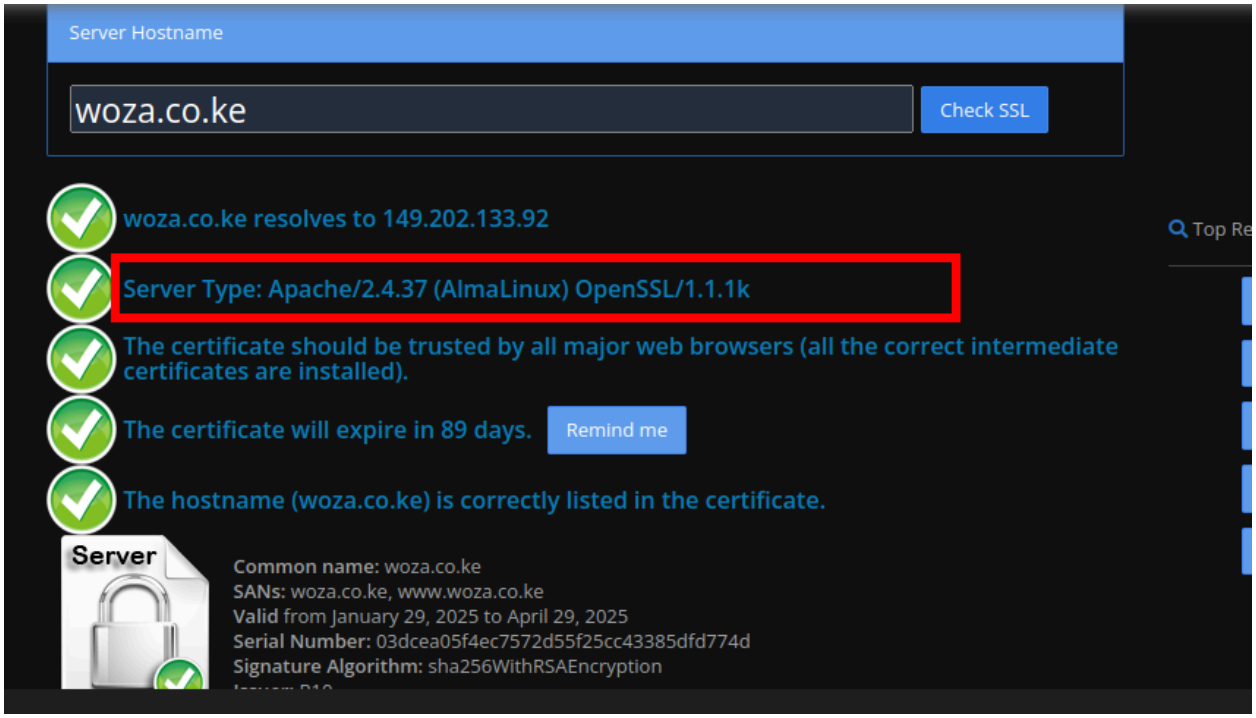
You will at the end see the output below.

```
Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/woza.co.ke/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/woza.co.ke/privkey.pem
This certificate expires on 2025-04-30.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the backgro
und.

Deploying certificate
Successfully deployed certificate for woza.co.ke to /etc/httpd/conf.d/woza.co.ke-le-ssl.co
nf
Successfully deployed certificate for www.woza.co.ke to /etc/httpd/conf.d/woza.co.ke-le-ss
l.conf
Congratulations! You have successfully enabled HTTPS on https://woza.co.ke and https://www
.woza.co.ke
```

Test your ssl here: https://www.ssllabs.com/ssltest/



Or here: https://www.sslshopper.com/ssl-checker.html



For Nginx

```
[root@king ~]# sudo dnf install certbot python3-certbot-nginx -y

Last metadata expiration check: 1:24:37 ago on Fri 31 Jan 2025 02:44:37 EST.
Package certbot-1.22.0-1.el8.noarch is already installed.
Dependencies resolved.
================================================================================
 Package                  Architecture        Version                 Repository
================================================================================
Installing:
 python3-certbot-nginx    noarch              1.22.0-1.el8            epel
Installing dependencies:
 python3-pyparsing        noarch              2.1.10-7.el8            baseos

Transaction Summary
```

sudo certbot --nginx -d example.com -d www.example.com

```
[root@king ~]# sudo certbot --nginx -d woza.co.ke -d www.woza.co.ke
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Certificate not yet due for renewal

You have an existing certificate that has exactly the same domains or certificate name you requested and isn't close to e
(ref: /etc/letsencrypt/renewal/woza.co.ke.conf)

What would you like to do?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: Attempt to reinstall this existing certificate
2: Renew & replace the certificate (may be subject to CA rate limits)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
Renewing an existing certificate for woza.co.ke and www.woza.co.ke

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/woza.co.ke/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/woza.co.ke/privkey.pem
This certificate expires on 2025-05-01.
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 woza.co.ke to /etc/nginx/conf.d/woza.co.ke.conf
Successfully deployed certificate for www.woza.co.ke to /etc/nginx/conf.d/woza.co.ke.conf
Your existing certificate has been successfully renewed, and the new certificate has been installed.
```

Step 3: Verify SSL Installation

- Open your website using **HTTPS** (***https://example.com***).

  Use:

  sudo certbot certificates

```
[root@king public_html]# sudo certbot certificates
Saving debug log to /var/log/letsencrypt/letsencrypt.log

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Found the following certs:
  Certificate Name: woza.co.ke
    Serial Number: 3dcea05f4ec7572d55f25cc43385dfd774d
    Key Type: RSA
    Domains: woza.co.ke www.woza.co.ke
    Expiry Date: 2025-04-30 06:18:18+00:00 (VALID: 89 days)
    Certificate Path: /etc/letsencrypt/live/woza.co.ke/fullchain.pem
    Private Key Path: /etc/letsencrypt/live/woza.co.ke/privkey.pem
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
[root@king public_html]#
```

  Check expiration:

```
openssl x509 -noout -dates -in
/etc/letsencrypt/live/example.com/fullchain.pem
```

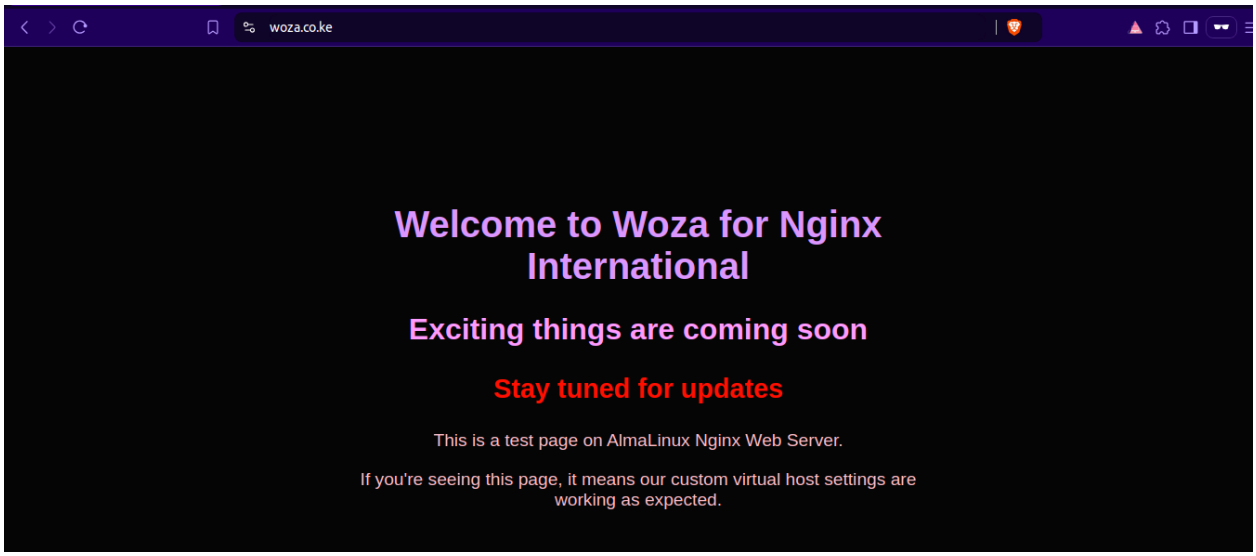
Step 4: Auto-Renew SSL

Let's Encrypt certificates expire every **90 days**, so enable automatic renewal:

```
sudo systemctl status certbot.timer
```

You can manually renew with:

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



---

3. Installing SSL/TLS Certificates: Managing Premium SSL Certificates

Premium SSL certificates are issued by commercial Certificate Authorities (CAs) like:

- **DigiCert**, **Sectigo**, **GlobalSign**, **GoDaddy**, **Namecheap**

Premium SSL offers:

✔ **Domain Validation** : Domain authentication.
✔ **Extended Validation (EV SSL)**: Strongest authentication.
✔ **Organization Validation (OV SSL)**: Verifies business legitimacy.
✔ **Wildcard SSL**: Secures **multiple subdomains**.

4. Generating a CSR via SSH in Linux

🔗 Truehost Guide: How to Generate a CSR via SSH in Linux

Step 1: Create a Private Key & CSR

Navigate to SSL directory.

```
cd /etc/ssl/certs/
```

Run the following command:

```
openssl req -new -newkey rsa:2048 -nodes -keyout example.com.key -out example.com.csr
```

Enter the required details:

Country Name (2 letter code) [XX]: KE

State or Province Name (full name) [Some-State]: Nairobi

Locality Name (eg, city) [Default City]: Nairobi

Organization Name (eg, company) [Default Company Ltd]: Truehost

Organizational Unit Name (eg, section) []: IT Department

Common Name (eg, your domain name) []: example.com

Email Address []: admin@example.com

```
[root@king ~]# cd /etc/ssl/certs/
[root@king certs]# ls
ca-bundle.crt  ca-bundle.trust.crt  localhost.crt
[root@king certs]# openssl req -new -newkey rsa:2048 -nodes -keyout woza.co.ke.key -out woza.co.ke.csr
Generating a RSA private key
.....................................................+++++
.....................................................................................................
.........................+++++
writing new private key to 'woza.co.ke.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:KE
State or Province Name (full name) []:NAIROBI
Locality Name (eg, city) [Default City]:NAIROBI
Organization Name (eg, company) [Default Company Ltd]:Woza Int
Organizational Unit Name (eg, section) []:IT
Common Name (eg, your name or your server's hostname) []:woza.co.ke
Email Address []:admin@woza.co.ke

Please enter the following 'extra' attributes
```

View the generated certs

```
[root@king certs]# ls -la
total 20
drwxr-xr-x. 2 root root 4096 Jan 30 04:40 .
drwxr-xr-x. 5 root root 4096 Jan 28 07:05 ..
lrwxrwxrwx  1 root root   49 Aug 21 15:20 ca-bundle.crt -> /etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem
lrwxrwxrwx  1 root root   55 Aug 21 15:20 ca-bundle.trust.crt -> /etc/pki/ca-trust/extracted/openssl/ca-bundle.t
rust.crt
-rw-r--r--  1 root root 3871 Jan 30 02:01 localhost.crt
-rw-r--r--  1 root root 1050 Jan 30 04:40 woza.co.ke.csr
-rw-------  1 root root 1704 Jan 30 04:39 woza.co.ke.key
[root@king certs]#
```

Step 2: Submit CSR to SSL Provider

- The **CSR (example.com.csr)** is submitted to the **SSL provider** to issue the certificate.

  To view the csr, type the command below;.

  cat example.com.csr

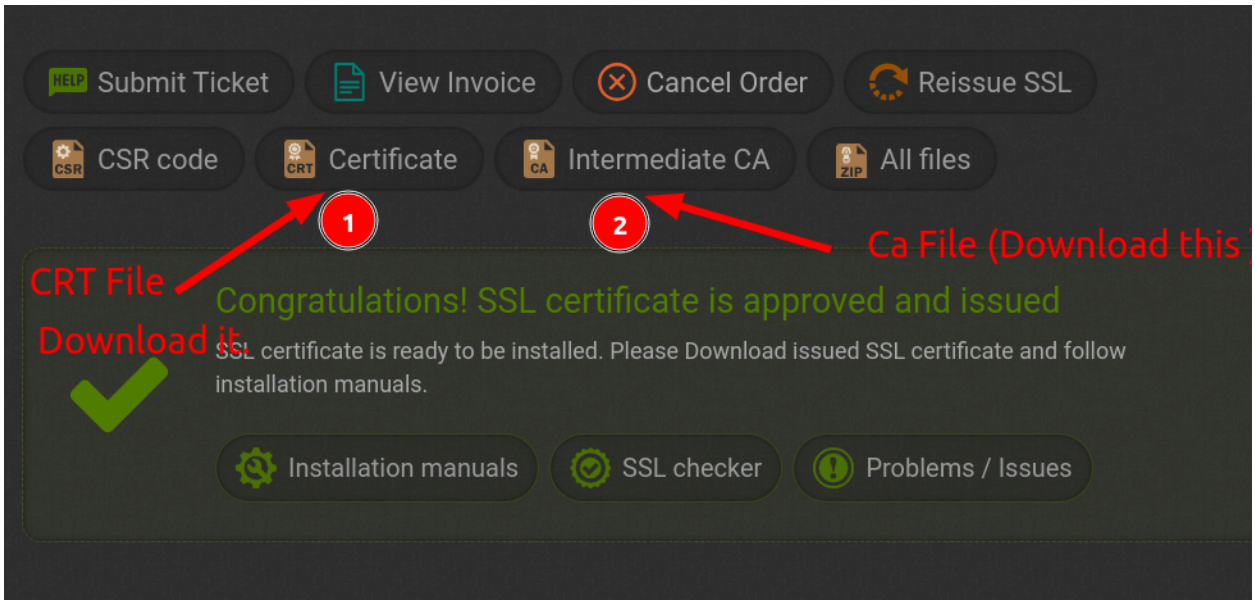
```
[root@king certs]# cat woza.co.ke.csr
-----BEGIN CERTIFICATE REQUEST-----
MIICzTCCAbUCAQAwgYcxCzAJBgNVBAYTAktFMRAwDgYDVQQIDAdOQUlST0JJMRAw
DgYDVQQHDAdOQUlST0JJMREwDwYDVQQKDAhXb3phIEludDELMAkGA1UECwwCSVQx
EzARBgNVBAMMCndvemEuY28ua2UxHzAdBgkqhkiG9w0BCQEWEGFkbWluQHdvemEu
Y28ua2UwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCexCS9WxYqO9GT
IJyGf64UuK6cxjhkvClZRZx9g14Z4m2hSP1GTufbHmBOJqYCYThg55cQkUXiHqKC
MwxWo8qW/AmT1oBjN2rxabDzuq4vsIafZmAWJtbLBRsAqD3xbT3QCqcY3mkCwFK1
j9AFLRisbwGFSrXBor9ZQSXlk7mUyPaADb1NvUjDEQMI823fNjHqk0lvgWjWx7/Y
paRwBF4wYBtrORSYPT0m1lSlt58m6rg8Y6uRJ0g3UDSTcx2cBBGffGDbDISFIi50
NRW2dUGz1s4OZyjLSIYleg69pYYKBv7VcQjZ8LsYPr0BEy8fmYGJQLjoZFBAuY4S
lhi9LUQfAgMBAAGgADANBgkqhkiG9w0BAQsFAAOCAQEAWX/L4BZ1mbK8uKw4v0no
Ge5rbWsLO7JDVPSSk3ptC2E7QnVSKf0EJNu3yotOOuBae3ZES6J0vyn1cJrGzVyi
lCYTfzxF0RXClU4qnhT+/RCT198tgatNvvzCdf8UbZ7KO7wRAVfD9Z0Egf9fQsF5
GwMDeeXbrHrrvg3ZzjQv6opszRb62FJdNDxmwcKu1nfcmFdLY2UfF3BininQgTuS
ZYjA7yv01zDASOnbTCF6A7w5PgMDJeClRNwD6MMQFFiBTH0CjSyHZd2uylwUNzTd
dP7w6WbDzoXXQ07Z2XqBrGg500s7xCPxQ1TCt8tQfqUsf1i3/yNwbJzr9hTuuRXv
Uw==
-----END CERTIFICATE REQUEST-----
[root@king certs]#
```

- The SSL provider sends back a **certificate file (.crt)**.
- Still inside the /etc/ssl/certs folder, create the two files as below.
  - touch example.com.ca
  - touch example.com.crt

```
[root@king certs]#
[root@king certs]# touch woza.co.ke.ca
[root@king certs]# touch woza.co.ke.crt
[root@king certs]# ls -la
total 20
drwxr-xr-x. 2 root root 4096 Jan 30 04:52 .
drwxr-xr-x. 5 root root 4096 Jan 28 07:05 ..
lrwxrwxrwx  1 root root   49 Aug 21 15:20 ca-bundle.crt -> /etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem
lrwxrwxrwx  1 root root   55 Aug 21 15:20 ca-bundle.trust.crt -> /etc/pki/ca-trust/extracted/openssl/ca-bundle.t
rust.crt
-rw-r--r--  1 root root 3871 Jan 30 02:01 localhost.crt
-rw-r--r--  1 root root    0 Jan 30 04:52 woza.co.ke.ca
-rw-r--r--  1 root root    0 Jan 30 04:52 woza.co.ke.crt
-rw-r--r--  1 root root 1050 Jan 30 04:40 woza.co.ke.csr
-rw-------  1 root root 1704 Jan 30 04:39 woza.co.ke.key
[root@king certs]#
```

Copy the content of your Intermediate Certificate file into example.com.ca and that of your Certificate file into example.com.crt

Since I am using Gogetssl, see sample below.



Now, we need to add the paths to the certificates above to the vhost so that they are loaded by our web server and used to secure our website.

5. Installing SSL on Apache Web Server

Step 1: Upload the SSL Certificate

Place the SSL files in ***/etc/ssl/certs/*** and ***/etc/ssl/private/: (In case you had done this in a different folder)***

```
sudo mv example.com.crt /etc/ssl/certs/

sudo mv example.com.key /etc/ssl/private/

sudo mv ca_bundle.crt /etc/ssl/certs/
```

Step 2: Edit the Apache Virtual Host File

```
sudo nano /etc/httpd/conf.d/example.com.conf
```

Add the following configuration:

```
<VirtualHost *:443>

    ServerName example.com

    DocumentRoot /var/www/example.com/public_html


    SSLEngine on

    SSLCertificateFile
/etc/ssl/certs/example.com.crt
```

```
    SSLCertificateKeyFile /etc/ssl/private/example.com.key

    SSLCertificateChainFile /etc/ssl/certs/ca_bundle.crt


    ErrorLog /var/log/httpd/example.com-error.log

    CustomLog /var/log/httpd/example.com-access.log combined

</VirtualHost>
```

Step 3: Restart Apache

```
sudo systemctl restart httpd
```

---

6. Installing SSL on Nginx Web Server

Step 1: Upload the SSL Certificate

Move the certificate files:

```
sudo mv example.com.crt /etc/ssl/certs/

sudo mv example.com.key /etc/ssl/private/
```

```
sudo mv ca_bundle.crt /etc/ssl/certs/
```

Step 2: Edit the Nginx Server Block

```
sudo nano /etc/nginx/conf.d/example.com.conf
```

Add the following configuration:

```
server {
    listen 443 ssl;
    server_name example.com www.example.com;

    ssl_certificate /etc/ssl/certs/example.com.crt;
    ssl_certificate_key
/etc/ssl/private/example.com.key;
    ssl_trusted_certificate
/etc/ssl/certs/ca_bundle.crt;

    location / {
        root /var/www/example.com/public_html;
        index index.html;
```

```
    }
}
```

Step 3: Restart Nginx

```
sudo systemctl restart nginx
```

---

7. Testing SSL Installation

✅ **Check SSL Configuration:**

```
openssl s_client -connect example.com:443
```

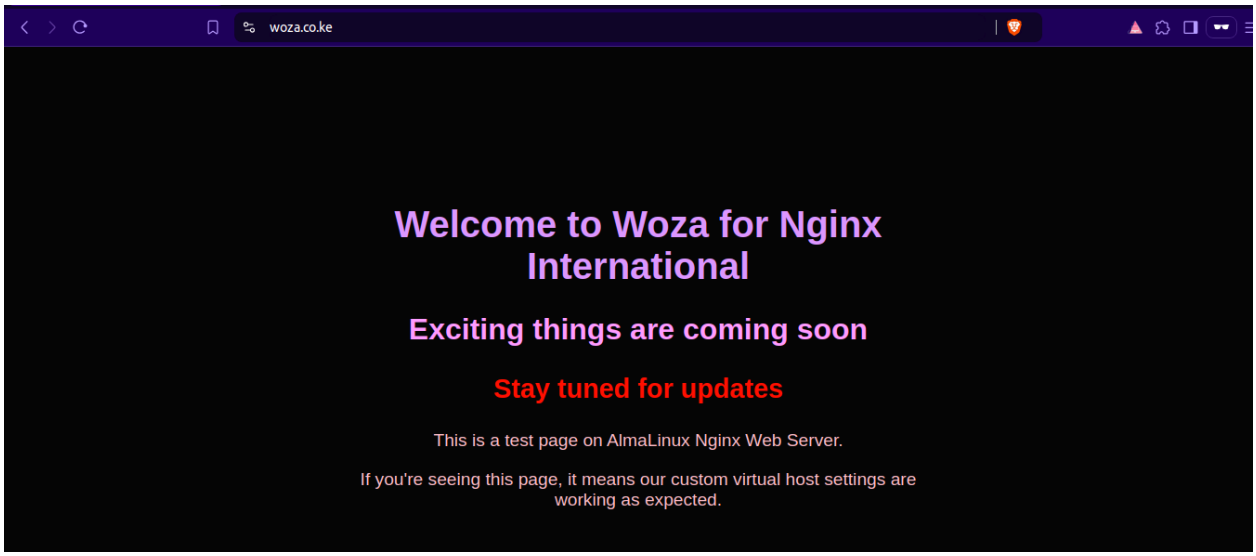
✅ **Use SSL Test Tools:**

- SSL Labs SSL Test
- https://www.sslshopper.com/ssl-checker.html

Summary

- **Let's Encrypt** provides free SSL certificates with **automatic renewal**.
- **Premium SSL** offers better security, wildcard support, and organization validation.
- **CSR generation** is required to obtain an SSL certificate.

- Apache and Nginx both support SSL, but the configuration differs.
- Always **test SSL installation** to ensure proper setup.



Module 4: Advanced Web Server Configuration and Optimization

1. Load Balancing with Apache and Nginx

What is Load Balancing?

Load balancing distributes incoming traffic across multiple backend servers to improve:
✅ **Performance** – Prevents a single server from being overwhelmed.
✅ **High Availability** – Ensures uptime if one server fails.
✅ **Scalability** – Allows handling of increased traffic.

Types of Load Balancing

- **Round Robin** – Requests are distributed sequentially.
- **Least Connections** – Sends traffic to the server with the fewest active connections.
- **IP Hashing** – Routes a specific client's request to the same backend server.

Load Balancing with Apache or Nginx.

NOTE: In this course, we shall not cover the setup of Load Balancing. This will be done in subsequent courses.

---

2. Implementing Caching Mechanisms for Improved Performance

What is Caching?

Caching stores frequently accessed data to reduce load times and server resource usage.

Types of Caching

- **Browser Caching** – Stores resources on the client's browser.
- **Page Caching** – Saves fully rendered pages to serve them faster.
- **Object Caching** – Stores database query results.
- **Opcode Caching** – Uses tools like OPcache to store compiled PHP scripts.

---

3. Configuring Caching

NOTE: In this course, we shall not cover the setup of Caching. This will be done in subsequent courses.

---

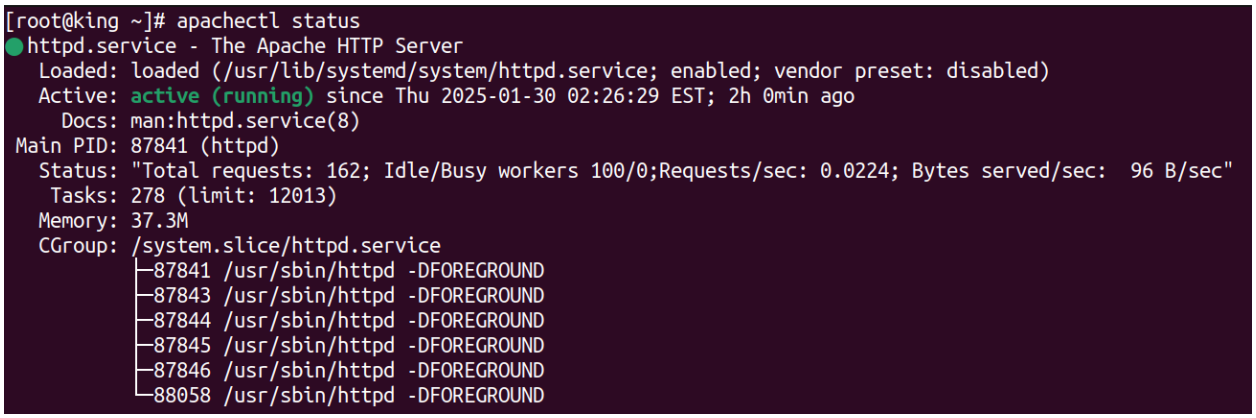
5. Monitoring and Troubleshooting Web Server Performance

Monitoring helps detect slow response times, high CPU usage, and potential failures.

Apache Performance Monitoring

✅ Check Active Connections

apachectl status



✅ Check Error Logs

sudo tail -f /var/log/httpd/error_log

```
[root@king ~]#
[root@king ~]# sudo tail -f /var/log/httpd/error_log
[Thu Jan 30 02:21:55.772464 2025] [mpm_event:notice] [pid 85899:tid 140129562442048] AH00492: caught SIGWINCH, s
hutting down gracefully
[Thu Jan 30 02:21:56.872356 2025] [suexec:notice] [pid 87606:tid 140268023720256] AH01232: suEXEC mechanism enab
led (wrapper: /usr/sbin/suexec)
[Thu Jan 30 02:21:56.907452 2025] [lbmethod_heartbeat:notice] [pid 87606:tid 140268023720256] AH02282: No slotme
m from mod_heartmonitor
[Thu Jan 30 02:21:56.913884 2025] [mpm_event:notice] [pid 87606:tid 140268023720256] AH00489: Apache/2.4.37 (Alm
aLinux) OpenSSL/1.1.1k configured -- resuming normal operations
```

✅ Monitor Requests Per Second

ab -n 1000 -c 10 http://example.com/

```
[root@king ~]# ab -n 1000 -c 10 http://woza.co.ke/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking woza.co.ke (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        Apache/2.4.37
Server Hostname:        woza.co.ke
Server Port:            80

Document Path:          /
Document Length:        227 bytes
```

Nginx Performance Monitoring

✅ Check Active Connections

sudo nginx -t

✅ Check Error Logs

```
sudo tail -f /var/log/nginx/error.log
```

✅ Monitor Real-Time Traffic

```
sudo apt install htop && htop
```

---

6. Best Practices for Securing Web Servers

🔒 Keep Software Updated

```
sudo apt update && sudo apt upgrade -y (Ubuntu/Debian)
sudo dnf update -y (RHEL/Almalinux/RockyLinux/Centos)
```

🔒 Restrict Access to Sensitive Directories

apache

```
<Directory /var/www/>
```

```
    Options -Indexes
</Directory>
```

🔒 Enable a Web Application Firewall (WAF)

Install **ModSecurity** for Apache

```
sudo apt install libapache2-mod-security2 -y
```

Install **Fail2Ban** for Nginx

```
sudo apt install fail2ban -y
```

🔒 Disable Unnecessary Modules

```
sudo a2dismod autoindex
```

🔒 Use Secure Headers

Edit the Apache or Nginx configuration:

apache

```
Header always set X-Frame-Options "SAMEORIGIN"
```

```
Header always set X-Content-Type-Options "nosniff"

Header always set Content-Security-Policy "default-src 'self'"
```

Summary

- **Load Balancing** improves availability and performance in Apache and Nginx.
- **Caching** reduces server load and speeds up page loads.
- **Monitoring tools** help detect performance bottlenecks.
- **Security Best Practices** protect against attacks.

The END.