**WEB HOSTING APPLICATION**

**Web hosting application** can be understood in two ways:

**1. Web Hosting Management Tools**

These are applications that help users manage web hosting services, allowing them to configure domains, databases, security, and server resources.

🔹 **Examples:**

* **cPanel & WHM** – User-friendly hosting control panel
* **Plesk** – Web hosting automation for Linux & Windows
* **WHMCS** – Web hosting billing and automation system
* **CyberPanel** – Open-source control panel for LiteSpeed servers

**👉 Used by:** Web hosting companies, system administrators, and businesses managing multiple websites.

**2. Web Applications Hosted on a Server**

A **web hosting application** can also refer to any website or web-based application that is hosted on a server. This can be a simple static website or a complex dynamic application.

🔹 **Examples of Hosting Platforms:**

* **AWS (Amazon Web Services)** – Offers EC2, S3, Elastic Beanstalk, etc.
* **Google Cloud Platform (GCP)** – App Engine, Compute Engine
* **Microsoft Azure** – Web Apps, Virtual Machines
* **Heroku, Vercel, Netlify** – Serverless hosting solutions

**👉 Used by:** Developers, businesses, and organizations to deploy and run websites or applications.

**Web Server Functionality:**

* IIS allows a Windows Server to host websites and web applications. This means it can deliver web content (like HTML pages, images, and videos) to users over the internet or an intranet.
* It supports various protocols, including:
  + **HTTP (Hypertext Transfer Protocol)**
  + **HTTPS (HTTP Secure)**
  + **FTP (File Transfer Protocol)**
  + **SMTP (Simple Mail Transfer Protocol**)

**Common web server software’s including:**

* Microsoft IIS (Internet Information Services)
* Apache HTTP Server
* Nginx

### ****Microsoft IIS (Internet Information Services)****

**IIS (Internet Information Services)** is a **web server developed by Microsoft** for hosting websites, applications, and services on **Windows Server and Windows OS**. It is widely used for **ASP.NET applications, enterprise environments, and Windows-based web hosting**.

 IIS is a core component of the Windows Server operating system.

 it’s primarily used for hosting websites and web applications on Windows-based servers.

 its tight integration with the Windows environment makes it a popular choice for organizations that rely on Microsoft technologies.

 It has evolved over many versions, and Microsoft continues to update and improve IIS.

1. **Apache HTTP Server (Apache)**

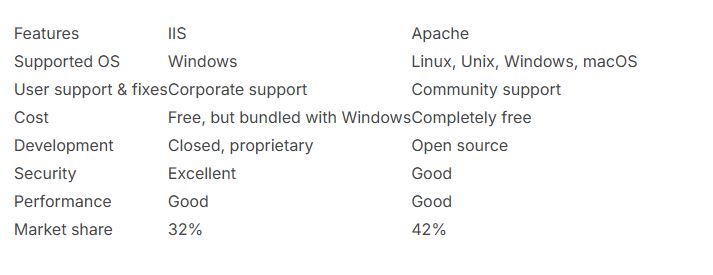
The **Apache HTTP Server** (commonly called **Apache**) is one of the most widely used open-source web server software. It is known for its flexibility, stability, and extensive module support.

✅ **Open-source & Free** – Developed and maintained by the **Apache Software Foundation**  
✅ **Cross-platform** – Runs on **Linux, Windows, and macOS**  
✅ **Highly Configurable** – Uses **modules** to extend functionality (e.g., security, caching, authentication)  
✅ **Supports Multiple Protocols** – Works with **HTTP, HTTPS, and FTP**  
✅ **Compatible with Many Technologies** – Supports **PHP, Python, Perl, Ruby, and more**  
✅ **Virtual Hosting** – Allows multiple websites to run on a single server

### ****NGINX – High-Performance Web Server & Reverse Proxy****

**NGINX** (pronounced "engine-x") is a powerful, high-performance web server known for handling **high traffic loads, acting as a reverse proxy, load balancer, and caching server**. It is widely used by large-scale websites like Netflix, Airbnb, and GitHub.

* While both Apache and Nginx are popular web servers, they have different architectural designs.
* Apache uses a process-driven approach, while Nginx uses an event-driven approach. This difference in architecture gives Nginx an advantage in handling high concurrency.



**Common Ways to Deploy a Web Server in AWS**

1. **Amazon EC2 (Elastic Compute Cloud)**
   * You can launch an **EC2 instance** and install a web server like **Apache, Nginx, or IIS** to serve web applications.
   * Best for: Full control over the web server, custom configurations.
2. **AWS Elastic Beanstalk**
   * A **Platform-as-a-Service (PaaS)** solution where AWS manages infrastructure, including load balancing and auto-scaling.
   * Supports popular platforms like **Java, Python, Node.js, .NET, PHP, Ruby, and Go**.
   * Best for: Developers who want to focus on application code without managing infrastructure.
3. **Amazon Lightsail**
   * A simple way to deploy a **pre-configured** web server with a fixed-cost model.
   * Supports WordPress, Joomla, and other CMS applications.
   * Best for: Small businesses, personal projects.
4. **AWS Lambda with API Gateway (Serverless)**
   * You can use **AWS Lambda** to run backend functions and **Amazon API Gateway** to handle HTTP requests.
   * No need to manage servers; AWS handles scaling.
   * Best for: Serverless web applications, RESTful APIs.
5. **Amazon S3 with Static Website Hosting**
   * Store and serve **static websites** (HTML, CSS, JavaScript) directly from an **S3 bucket**.
   * Best for: Static websites without a backend (e.g., company landing pages, documentation sites).
6. **AWS Amplify**
   * A managed service for **full-stack applications** that integrates hosting, authentication, and backend functions.
   * Best for: Frontend and mobile developers.
7. **Amazon CloudFront (CDN) + Web Server**
   * Use AWS **CloudFront** as a **Content Delivery Network (CDN)** to distribute web content globally with low latency.
   * Works with EC2, S3, or on-premise servers.
   * Best for: High-performance global websites.

**Deploying a web server on AWS using a Windows Server EC2 instance**

## ****Step 1: Launch a Windows Server EC2 Instance****

1. **Sign in** to the AWS Management Console.
2. Navigate to **EC2** > Click **Launch Instance**.
3. **Select an Amazon Machine Image (AMI):**
   * Choose **Windows Server 2022 Base** (or another version you prefer).
4. **Choose an Instance Type:**
   * Select a suitable instance type (e.g., t2.micro for testing, t3.medium for production).
5. **Configure Instance Details:**
   * Ensure Auto-assign **Public IP** is enabled.
6. **Add Storage:**
   * Allocate sufficient disk space (e.g., 30GB recommended).
7. **Configure Security Group:**
   * Allow **RDP (3389)** for remote access.
   * Allow **HTTP (80)** and **HTTPS (443)** for web traffic.
8. **Launch and Connect:**
   * Click **Launch** and create/select a key pair.
   * After the instance starts, connect using **Remote Desktop Protocol (RDP)**.

## ****Step 2: Install a Web Server (IIS) on Windows Server****

1. Open **Server Manager** in the Windows instance.
2. Click **Manage** > **Add Roles and Features**.
3. Select **Role-based or feature-based installation** > Click **Next**.
4. Choose your server and click **Next**.
5. Under **Server Roles**, select **Web Server (IIS)** and click **Next**.
6. Click **Install** and wait for the installation to complete.

## ****Step 3: Configure IIS and Host a Website****

1. Open **Internet Information Services (IIS) Manager**.
2. Expand the **Server Name** > Click **Sites**.
3. Right-click **Default Web Site** > **Manage Website** > **Browse**.
   * This should open the **IIS default page** in your browser.
4. To deploy a custom website:
   * Upload your **HTML, ASP.NET, or PHP files** to C:\inetpub\wwwroot.
   * Restart IIS (iisreset in **Command Prompt**).

## ****Step 4: Update Security Group to Allow Web Traffic****

1. Go to **EC2 Dashboard** > **Security Groups**.
2. Find the **Security Group** attached to your instance.
3. Click **Edit inbound rules** and **add the following rules**:
   * **Type:** HTTP | **Protocol:** TCP | **Port Range:** 80 | **Source:** Anywhere (0.0.0.0/0)
   * **Type:** HTTPS | **Protocol:** TCP | **Port Range:** 443 | **Source:** Anywhere (0.0.0.0/0)

## ****Step 5: Access Your Web Server****

1. Find your **EC2 Public IPv4 address** from the EC2 dashboard.
2. Open a browser and enter:

## ****Optional Enhancements****

* **Assign an Elastic IP** to prevent changing public IPs.
* **Set up an SSL Certificate** (AWS Certificate Manager + Load Balancer).
* **Use a Custom Domain** with **Route 53**.