Web Server Configuration with DNS

Server

A Server is a computer, software, or device that provides data, resources, or services to other devices, known as clients, over a network. Servers handle requests and deliver responses, enabling websites, applications, and services to function.

Types of Servers

1. **Web Server**

   Purpose: Hosts websites and delivers web pages to clients (browsers) using HTTP/HTTPS.

   Examples: Apache, Nginx, Microsoft IIS, LiteSpeed.

Use Case: Serving websites, handling HTTP requests, managing static and dynamic content.

2. **Application Server**

   Purpose: Hosts and manages application logic and processes between the user and the backend database.

   Examples: Tomcat, JBoss, WebLogic, GlassFish.

   Use Case: Hosting enterprise applications, APIs, and web services.

3. **Database Server**

   Purpose: Stores, manages, and provides access to structured data.

   Examples: MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database**.**

Use Case: Managing large datasets, supporting web applications, and storing user information**.**

4. **File Server**

   Purpose: Provides centralized storage and file sharing for users and devices.

   Examples: Windows File Server, Samba, NAS (Network Attached Storage).

   Use Case: Storing and sharing files across a local network.

5. **Mail Server**

   Purpose: Handles the sending, receiving, and storage of email messages.

   Examples: Microsoft Exchange, Postfix, Sendmail.

   Use Case: Managing email communication for organizations.

6. **DNS Server (Domain Name System)**

   Purpose: Translates human-readable domain names (like google.com) into IP addresses.

   Examples: BIND, Microsoft DNS, Cloudflare DNS.

   Use Case: Resolving domain names, enabling internet access.

Introduction to Web Servers

A web server is a computer or software that delivers web pages to your browser when you type a website's address (like www.google.com). It handles requests from your browser and sends back the requested content (like text, images, or videos).

How It Works:

Request: You type a website URL in your browser.

Search: The browser finds the server where the website is stored.

Send Request: Your browser asks the server for the webpage.

Response: The server sends back the page.

Display: The browser shows the page to you.

Types of Web Servers:

Static Web Server: Sends fixed content (like HTML, images).

Dynamic Web Server: Processes and generates content before sending it (like PHP or Python apps).

Examples of Web Servers:

Apache: Most common, used for hosting websites.

Nginx: Fast and handles high traffic.

IIS (Internet Information Services): Used on Windows servers.

Why Are Web Servers Important?

- Deliver websites quickly.
- Handle multiple users at the same time.
- Make sure websites run smoothly.

In simple terms: A web server is like a waiter that brings you the right dish (webpage) after you place your order (URL request).

INSTALLATION OF DNS

Installing DNS on Windows Server

Step 1: Open Server Manager

1. Click on Start and open Server Manager.
2. Click on Manage (top-right corner).
3. Select Add Roles and Features.

Step 2: Start the Installation Wizard

1. Choose Role-based or feature-based installation and click Next.
2. Select the target server and click Next.

Step 3: Install DNS Server

1. Scroll down and select DNS Server.
2. Click Add Features if prompted.

3. Click Next, then Install.

Step 4: Complete Installation

1. Wait for the installation to complete.
2. Click Close when finished.

How to Install a DNS Server on Windows 11 (Simple Steps)

Windows 11 does not have a built-in DNS server like Windows Server, but you can configure a DNS resolver or use third-party DNS software.

Method 1: Change DNS Settings (Using Public DNS)

Step 1: Open Settings

1. Press Win + I to open Settings.
2. Click on Network & Internet.

Step 2: Select Your Network

1. Click on Wi-Fi (if wireless) or Ethernet (if wired).
2. Select Hardware Properties or Edit IP Assignment.

Step 3: Change DNS Settings

1. Click on Edit next to DNS Server Assignment.
2. Select Manual and enable IPv4.
3. Enter preferred DNS servers:

- Google DNS: 8.8.8.8 and 8.8.4.4
- Cloudflare DNS: 1.1.1.1 and 1.0.0.1

4. Click Save.

You've successfully set custom DNS servers.

Multiple web server using IP address

When hosting multiple websites on a single server, you can configure it to use different IP addresses to serve each website separately. This is known as IP-based virtual hosting.

IP-Based Hosting –

In IP-based virtual hosting, each website is associated with a different IP address. The server listens on multiple IP addresses and serves different websites based on the IP the request is sent to.

How It Works:

1. **Multiple IP Assignment:**

   A single physical or virtual server is configured with multiple IP addresses.

   Example:

   192.168.1.100 for site1.com

   192.168.1.101 for site2.com

2. **DNS Configuration:**

   DNS records for each domain point to their respective IP addresses.

   Example:

   site1.com -> 192.168.1.100

   site2.com -> 192.168.1.101

3. **Web Server Configuration:**

- The web server (like Apache or Nginx) is configured with virtual hosts that bind each website to a different IP address.
- Incoming requests are directed to the correct website based on the IP address.

Key Components:

Server Configuration: Set up virtual hosts with unique IP addresses.

DNS Configuration: Map domain names to the correct IPs.

Firewall Rules: Open necessary ports (e.g., 80 for HTTP and 443 for HTTPS).

Benefits of IP-Based Hosting:

Isolation: Each website is isolated by IP, improving security.

SSL Certificates: Easier to use separate SSL certificates for each site.

Traffic Management: Better control over incoming traffic.

Challenges/Considerations:

IP Scarcity: Each website requires a unique public IP, which can be costly.

Configuration Complexity: Requires additional DNS and network configuration.

Use Cases:

- Hosting multiple high-traffic websites.
- Hosting websites that require unique SSL certificates.
- Managing enterprise-level applications with different IPs.

How to Download and Install Apache Server on Windows 11 (Simple Steps)

Step 1: Download Apache for Windows

- Open your browser and go to the official Apache website i.e.Apache Lounge
- Click on the latest version of Apache HTTP Server for Windows.
- Download the ZIP file for Apache (usually named like httpd-2.4.xx-win64-VS16.zip).

Step 2: Extract the ZIP File

- Right-click on the downloaded file and select Extract All.
- Extract the files to a directory (e.g., C:\Apache24).

Step 3: Configure Apache

- Open the extracted folder (C:\Apache24 or your chosen directory).
- Go to the conf folder and open (httpd.conf) with Notepad.
- Press Ctrl + F and search for Listen 80.
- Save and close the file.

Step 4: Install Apache as a Service

1. Open Command Prompt as Administrator:

- Press Win + X and select Command Prompt (Admin) or Terminal (Admin).

2. Navigate to the Apache **bin** directory:

cd C:\Apache24\bin

3. Install Apache as a service:

httpd.exe -k install

4. Start the Apache service:

httpd.exe -k start

Step 5: Verify Apache is Running

1. Open your browser.

2. Type:

http://localhost

3. If Apache is running, you will see the default "It works!" page.

To Stop or Restart Apache:

- To stop:

httpd.exe -k stop

- To restart:

httpd.exe -k restart

How to Install Node.js and Set It Up as a Server (Simple Steps)

Step 1: Download and Install Node.js

- Go to Node.js Official Website: ☞ https://nodejs.org/
- Download the LTS Version:
  - Choose the LTS (Long-Term Support) version for stability.
  - Download the installer for Windows, Mac, or Linux based on your system.

- Install Node.js:
- Open the downloaded file and follow the installation instructions.
- Make sure to check the box that says "Add to PATH" during installation.

Step 2: Verify Installation

- Open Command Prompt/Terminal.
- Check if Node.js is installed:

node –v

You will see the installed Node.js version.

- Check if npm (Node Package Manager) is installed:

npm -v

You will see the npm version.

Step 3: Create a Basic Node.js Server

1. Create a New Folder:

- Open Command Prompt/Terminal and create a project directory:

mkdir my-node-server
cd my-node-server

2. Create a JavaScript File:

Create a new file named **server.js** inside the folder.

Add the following code:

```
// Load HTTP module
const http = require('http');

// Create a server
const server = http.createServer((req, res) => {
   res.writeHead(200, { 'Content-Type': 'text/plain' });
   res.end('Hello, Node.js Server!\n');
});

// Set the server to listen on port 3000
server.listen(3000, () => {
   console.log('Server running at http://localhost:3000/');
});
```

Step 4: Run Your Node.js Server

1. Open your terminal and navigate to the project folder:

cd my-node-server

2. Run the server:

node server.js

You will see a message:

Step 5: Test Your Node.js Server

- Open your browser.
- Go to:

http://localhost:3000

- You will see the message:

Hello, Node.js Server!

To Stop the Server:

- Press Ctrl + C in the terminal.

DNS Zone Transfer

A DNS Zone Transfer is the process of copying DNS records from one DNS server to another. It is used to synchronize DNS data between a primary (master) DNS server and one or more secondary (slave) DNS servers.

How It Works:

1. Primary DNS Server (Master):

- Stores the original DNS zone data.
- Updates and manages DNS records.

2. Secondary DNS Server (Slave):

- Retrieves zone data from the primary server through a zone transfer.
- Acts as a backup in case the primary server fails.

3. Zone Transfer Process:

- The secondary server sends a request to the primary server for updated DNS records.
- The primary server responds with the zone data, synchronizing both servers.

Zone Transfer Process:

1. Zone File Update: Changes are made to the DNS zone on the primary server.
2. Notify Message: The primary server notifies the secondary server that updates are available.
3. Zone Transfer Request: The secondary server requests a zone transfer.
4. Zone Data Transfer:

- AXFR: Entire zone is transferred.
- IXFR: Only changes are transferred.

Imagine This:

You are the **owner of a company**, and you keep a **list of all your employees** (names, phone numbers, departments, etc.) in a notebook. This notebook is very important.

You have:

- **Main Notebook** in your office (this is the **Primary DNS server**).

- **Backup Notebook** in your assistant's office (this is the **Secondary DNS server**).

What is a Zone Transfer in this case?

A **zone transfer** is when your assistant **copies everything** from your notebook to their notebook.

Why?

- So if your notebook is lost or damaged (server is down), the assistant still has all the info.
- So people can still get the info they need from the assistant.

Real Life Example (Online):

You own a website called `mycompany.com`.

- Your main DNS server (like your notebook) has all the records:
  - Where the website is hosted
  - Email server info
  - Other services (like FTP, VPN, etc.)
- You also have a second DNS server that needs to have the **same records** so your site runs smoothly.

So, the second server says:

"Hey primary server, please send me all your DNS records for `mycompany.com`."

This process is called a **DNS Zone Transfer**.

Why Be Careful?

If you let **anyone** copy your notebook (zone transfer without security), a bad person might get:

- A full list of your servers
- Email server details (to try to send fake emails)
- Other secret info about your setup

That's why zone transfers should only be allowed **between trusted friends (servers)**.

Summary

- DNS Zone Transfer = **Copying your list of DNS records** from one server to another.
- Primary = Original copy, Secondary = Backup.
- It helps with **speed**, **backup**, and **consistency**.
- Only **trusted DNS servers** should be allowed to do this.

DNS Zone Transfer Process (Step-by-Step)

1. Two DNS Servers Are Set Up

- One is the **Primary DNS Server** (holds the main zone file).
- The other is the **Secondary DNS Server** (needs to copy that data).

Think of:

- Primary = Main notebook with all the important info.
- Secondary = Assistant's notebook that gets a copy.

2. Secondary Requests a Zone Transfer

- The **Secondary Server** asks the Primary:

  "Hey, do you have any updates for the zone example.com?"

- This is done using a special message (usually using **AXFR** for full copy or **IXFR** for updates only).

3. Primary Checks Permissions

- The Primary DNS server checks:
  - "Is this a trusted server?"
  - "Is it allowed to get zone transfers?"

If yes → continue
If no → deny the request (to prevent abuse)

4. Zone Data is Transferred

- If approved, the **Primary DNS server sends the full zone file** to the Secondary.
- The data includes all the DNS records: A, MX, CNAME, TXT, etc.

This is like sending **a full copy of your address book**.

5. Secondary Stores the Data

- The Secondary DNS server **saves the zone data**.
- Now it can answer DNS queries just like the Primary.

6. Regular Updates

- The Secondary will check the Primary regularly:

  “Hey, has anything changed?”

- If something changed (like a new IP or record), it does another transfer (usually a small update using **IXFR**).

Example in Technical Terms

Here’s how a full zone transfer (AXFR) looks like in practice using a DNS tool:

dig AXFR example.com @ns1.example.com

This command says:

"Give me a full copy of the zone example.com from the server ns1.example.com."