Web Server

Tujuan Pembelajaran:

1. Peserta didik mampu memahami pengertian, fungsi, dan prinsip kerja web server.
2. Peserta didik mampu menyebutkan contoh web server yang banyak digunakan serta menjelaskan perbedaannya.
3. Peserta didik mampu melakukan instalasi dan konfigurasi web server Apache dan Nginx di Debian.

---

1. Pengantar tentang Web Server

Web server adalah komponen penting dalam dunia internet yang memungkinkan website diakses oleh pengguna dari berbagai tempat. Setiap kali kita mengunjungi sebuah website, proses pengambilan data dilakukan oleh web server. Dalam pembelajaran ini, kita akan mempelajari bagaimana web server bekerja, apa saja jenis web server yang banyak digunakan, serta cara menginstal dan mengonfigurasinya.

2. Pengertian Web Server

Web server adalah perangkat lunak yang menerima permintaan (request) dari pengguna melalui browser, memprosesnya, dan mengirimkan kembali hasil dalam bentuk halaman web. Web server mengelola komunikasi antara client (browser) dan server (sistem yang menyediakan layanan web).

3. Fungsi Web Server

Web server memiliki beberapa fungsi utama:

a. **Melayani permintaan HTTP/HTTPS:** Web server menerima permintaan dari client dan mengirimkan respons dalam bentuk halaman web.

b. **Pengelolaan file statis dan dinamis:** Web server dapat menangani file statis seperti gambar, HTML, CSS, dan file dinamis melalui scripting.

c. **Keamanan dan otentikasi:** Web server sering dilengkapi fitur untuk membatasi akses berdasarkan otentikasi pengguna.

4. Prinsip dan Cara Kerja Web Server

Prinsip dasar kerja web server adalah sebagai perantara antara client (biasanya browser web) dengan server tempat data dan konten situs web disimpan. Berikut penjelasan lebih rinci mengenai cara kerja web server:

1. **Request dari Client:** Ketika seorang pengguna mengetikkan URL di browser atau mengklik sebuah tautan, browser akan mengirimkan permintaan HTTP (Hypertext Transfer Protocol) atau HTTPS (versi aman dari HTTP) ke web server. Permintaan ini dapat berupa permintaan halaman HTML, gambar, video, atau file lainnya.
2. **Penerimaan dan Pemrosesan Request oleh Web Server:** Web server menerima permintaan ini dan menentukan apa yang harus dilakukan berdasarkan alamat URL dan metode HTTP yang digunakan (misalnya GET, POST). Web server akan memeriksa apakah file yang diminta berada di dalam direktori yang diizinkan untuk diakses.
3. **Eksekusi Skrip atau Pengambilan File:**
    a. **Untuk file statis:** Jika file yang diminta berupa file statis (misalnya gambar, file HTML, atau CSS), web server langsung mengirimkan file tersebut ke client.
    b. **Untuk file dinamis:** Jika permintaan melibatkan file dinamis (misalnya PHP, Python, atau Ruby), web server menjalankan skrip yang sesuai dan menghasilkan halaman yang diminta sebelum mengirimkannya ke client.
4. **Mengirimkan Respons ke Client:** Setelah web server memproses permintaan, hasilnya dikirimkan kembali ke browser. Browser kemudian menampilkan hasil tersebut kepada pengguna dalam bentuk halaman web.
5. **Logging dan Monitoring:** Web server juga biasanya mencatat aktivitas yang terjadi, seperti alamat IP yang mengakses server, waktu akses, serta jenis permintaan yang dilakukan. Data ini penting untuk menganalisis trafik dan mendeteksi potensi masalah.

5. Contoh Web Server yang Banyak Digunakan

1. Apache HTTP Server

Apache dikembangkan pertama kali pada tahun 1995 oleh Apache Software Foundation. Ini adalah salah satu web server open-source paling populer dan paling banyak digunakan di dunia, terutama karena kemampuannya untuk berjalan di berbagai sistem operasi seperti Unix, Linux, dan Windows.

Kelebihan:

a. Dukungan komunitas yang sangat besar.
b. Mudah dikonfigurasi dan sangat fleksibel.
c. Mendukung banyak modul ekstensi untuk meningkatkan fungsionalitas, seperti mod_ssl untuk keamanan HTTPS.
d. Kompatibel dengan banyak sistem operasi.

Kekurangan:

a. Memerlukan sumber daya lebih banyak dibandingkan Nginx ketika menangani trafik yang sangat tinggi.

b. Kinerja menurun saat harus menangani permintaan dalam jumlah besar secara bersamaan.

2. Nginx (Engine-X)

Nginx dikembangkan oleh Igor Sysoev pada tahun 2004, awalnya dirancang untuk menyelesaikan masalah "C10k" (kesulitan menangani lebih dari 10.000 koneksi secara bersamaan). Karena fokusnya pada efisiensi dan kemampuan menangani banyak koneksi sekaligus, Nginx dengan cepat menjadi pilihan utama untuk situs web dengan trafik tinggi.

Kelebihan:

a. Dapat menangani ribuan koneksi secara bersamaan dengan penggunaan memori yang rendah.

b. Lebih cepat dalam melayani file statis daripada Apache.

c. Dukungan untuk reverse proxy, load balancing, dan caching bawaan.

Kekurangan:

a. Tidak memiliki kemampuan untuk menjalankan modul dinamis seperti Apache. Ini berarti konfigurasi dan fungsionalitas tambahan harus ditangani dengan solusi lain (misalnya, menggunakan PHP-FPM untuk mendukung PHP).

3. LiteSpeed

LiteSpeed dikembangkan oleh LiteSpeed Technologies pada tahun 2002. LiteSpeed dirancang untuk menawarkan peningkatan performa dan keamanan dibandingkan dengan Apache, serta kompatibel dengan berbagai teknologi yang digunakan pada Apache (seperti mod_rewrite dan mod_security).

Kelebihan:

a. Sangat cepat dan efisien dalam penggunaan sumber daya, lebih cepat dari Apache dalam melayani konten dinamis.

b. Mendukung protokol HTTP/3 terbaru untuk koneksi yang lebih cepat.

c. Kompatibel dengan file konfigurasi Apache sehingga mudah diintegrasikan.

Kekurangan:

a. Bukan sepenuhnya open-source, karena versi berbayar menawarkan lebih banyak fitur daripada versi gratisnya.

4. **Microsoft IIS (Internet Information Services)**

   IIS adalah web server yang dikembangkan oleh Microsoft dan dirilis pertama kali pada tahun 1995. Web server ini secara default terintegrasi dengan sistem operasi Windows Server, sehingga populer di kalangan pengguna platform Windows.

   Kelebihan:

   a. Integrasi yang baik dengan teknologi Windows lainnya seperti Active Directory dan .NET framework.

   b. Memiliki antarmuka pengguna grafis (GUI) yang mudah digunakan untuk manajemen server.

   Kekurangan:

   a. Hanya tersedia untuk sistem operasi Windows, sehingga kurang fleksibel dibandingkan Apache atau Nginx yang dapat berjalan di berbagai platform.

   b. Penggunaan lisensi berbayar untuk versi Windows Server.

6. **Instalasi Apache2 di Debian**

1. Update repository:
   `sudo apt update`
2. Instalasi Apache:
   `sudo apt install apache2`
3. Memeriksa Status Apache:
   `sudo systemctl status apache2`
4. Mengakses Apache:
   Buka browser dan ketik http://localhost atau alamat IP server untuk melihat halaman default Apache yang menandakan server sudah aktif.

Pengaturan Document Root pada Apache

Document Root adalah direktori tempat Apache menyimpan file website yang akan dilayani ke pengguna. Secara default, pada sistem Debian, document root Apache terletak di:

`/var/www/html`

Artinya, semua file HTML, CSS, JavaScript, gambar, dan konten website lainnya yang akan diakses melalui browser harus disimpan di direktori ini.

Cara Mengubah Document Root pada Apache

Untuk mengubah lokasi **Document Root**, ikuti langkah-langkah berikut:

1. Edit Virtual Host File
   - Apache menggunakan file konfigurasi Virtual Host untuk menentukan document root. File konfigurasi untuk Virtual Host default terletak di:
     `/etc/apache2/sites-available/000-default.conf`
   - Buka file ini dengan editor teks (misalnya nano):
     `sudo nano /etc/apache2/sites-available/000-default.conf`

2. Ubah Direktori Document Root
    - Cari baris yang mendefinisikan **DocumentRoot**, yang secara default terlihat seperti ini:
      `DocumentRoot /var/www/html`
    - Ubah jalur tersebut ke direktori baru yang ingin Anda gunakan sebagai document root. Misalnya, jika Anda ingin mengubahnya menjadi /home/user/mywebsite, ubah menjadi:
      `DocumentRoot /home/user/mywebsite`
3. Buat dan Setel Izin untuk Direktori Baru
    - Jika direktori baru belum ada, buatlah direktori tersebut:
      `sudo mkdir -p /home/user/mywebsite`
    - Atur kepemilikan dan izin untuk direktori tersebut agar Apache dapat mengaksesnya:
      ```
      sudo chown -R www-data:www-data /home/user/mywebsite
      sudo chmod -R 755 /home/user/mywebsite
      ```
4. Restart Apache
    - Setelah mengubah document root, restart Apache agar perubahan berlaku:
      `sudo systemctl restart apache2`
5. Akses Website dengan Document Root Baru
    - Sekarang, file website yang berada di direktori baru (misalnya /home/user/mywebsite) akan diakses oleh Apache dan dapat dilihat di browser melalui alamat http://localhost atau IP server.

7. Instalasi Nginx di Debian

1. Update repository:
   `sudo apt update`
2. Instalasi Nginx:
   `sudo apt install nginx`
3. Memeriksa Status Nginx:
   `sudo systemctl status nginx`
4. Mengakses Nginx:
   Buka browser dan ketik http://localhost atau alamat IP server untuk melihat halaman default Nginx yang menandakan server sudah aktif.

Pengaturan Document Root pada Nginx

Document Root adalah direktori tempat Nginx menyimpan file website yang akan dilayani ke pengguna. Secara default, pada sistem Debian, document root Nginx terletak di:

`/var/www/html`

Ini adalah lokasi di mana Anda perlu menyimpan file HTML, CSS, JavaScript, gambar, dan konten lainnya yang ingin dilayani oleh Nginx. File ini dapat diakses oleh pengguna melalui browser.

Cara Mengubah Document Root pada Nginx

Untuk mengubah **Document Root** pada Nginx, ikuti langkah-langkah berikut:

1. **Edit File Konfigurasi Server Block**
   - Nginx menggunakan **Server Block** (setara dengan Virtual Host di Apache) untuk mengatur server dan document root. Konfigurasi server block default terletak di: `/etc/nginx/sites-available/default`
   - Buka file ini menggunakan editor teks seperti nano:
     ```
     sudo nano /etc/nginx/sites-available/default
     ```
2. Ubah Direktori Document Root
   - Cari bagian yang berisi pengaturan root, yang secara default terlihat seperti ini:
     ```
     root /var/www/html;
     ```
   - Ubah jalur tersebut ke direktori baru yang ingin Anda gunakan sebagai document root. Misalnya, jika Anda ingin mengubahnya menjadi /home/user/mywebsite, ubah menjadi:
     ```
     root /home/user/mywebsite;
     ```
3. Buat dan Setel Izin untuk Direktori Baru
   - Jika direktori baru belum ada, buatlah direktori tersebut:
     ```
     sudo mkdir -p /home/user/mywebsite
     ```
   - Atur kepemilikan dan izin untuk direktori baru tersebut agar Nginx dapat mengakses dan melayani file dari sana:
     ```
     sudo chown -R www-data:www-data /home/user/mywebsite
     sudo chmod -R 755 /home/user/mywebsite
     ```
4. Uji Konfigurasi Nginx
   - Setelah mengedit file konfigurasi, Anda perlu memeriksa apakah ada kesalahan dalam konfigurasi dengan menjalankan perintah:
     ```
     sudo nginx -t
     ```
   - Jika tidak ada kesalahan, Anda akan mendapatkan pesan seperti "syntax is ok" dan "test is successful".
5. Restart Nginx
   - Restart Nginx agar perubahan pada konfigurasi berlaku:
     ```
     sudo systemctl restart nginx
     ```
6. Akses Website dengan Document Root Baru
   - Setelah restart, file website yang berada di direktori baru (misalnya /home/user/mywebsite) akan dilayani oleh Nginx, dan Anda dapat mengaksesnya melalui http://localhost atau IP server.

Contoh File Konfigurasi Nginx Setelah Mengubah Document Root

Setelah Anda mengubah **Document Root**, berikut adalah contoh dari file konfigurasi default setelah perubahan:

```
server {
    listen 80;
    server_name localhost;

    root /home/user/mywebsite;
    index index.html index.htm index.nginx-debian.html;

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

    error_page 404 /404.html;
    location = /404.html {
        internal;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
    }

    location ~ /\.ht {
        deny all;
    }
}
```