Az alábbiakban részletes útmutató található az Apache HTTP szerver telepítéséhez és konfigurálásához Debian Linux rendszeren.

---

1. Apache telepítése

1. Frissítsük a csomaglistát: `sudo apt update`
2. Telepítsük az Apache szervert: `sudo apt install apache2`
3. Ellenőrizzük az Apache telepítését: `apache2 -v`

---

2. Szolgáltatás kezelése

- **Apache indítása:** `sudo systemctl start apache2`
- **Apache automatikus indításának engedélyezése rendszerinduláskor:** `sudo systemctl enable apache2`
- **Apache állapotának ellenőrzése:** `sudo systemctl status apache2`

---

3. Tűzfal beállítása

- Ha tűzfalat használ (pl. UFW), engedélyezze az Apache forgalmat:
- Alapértelmezett profil engedélyezése: `sudo ufw allow 'Apache'`
- A tűzfal állapotának ellenőrzése: `sudo ufw status`

---

4. Webszerver tesztelése

Nyisson meg egy böngészőt, és írja be a szerver IP-címét vagy a localhost címet: `http://<IP-cím>`
Ha az Apache helyesen működik, az alapértelmezett "*Apache2 Debian Default Page*" jelenik meg.

---

5. Alapértelmezett konfigurációs fájlok

- **Apache fő konfigurációs fájl:** `/etc/apache2/apache2.conf`
- **Virtuális hosztok konfigurációs fájlok:** `/etc/apache2/sites-available/`
- **Engedélyezett hoszt fájlok:** `/etc/apache2/sites-enabled/`
- **Engedélyezett modulok listája:** `/etc/apache2/mods-enabled/`

---

6. Virtuális hosztok beállítása

Az Apache virtuális hoszt (Virtual Host) egy olyan konfigurációs mechanizmus, amely lehetővé teszi, hogy egyetlen Apache webszerver több weboldalt szolgáljon ki, akár ugyanazon az IP-címen, akár különböző IP-címeken. Ezt használják például arra, hogy több domain vagy aldomain különböző webhelyeket szolgáljon ki ugyanazon a szerveren.

6.1 Virtuális hosztok típusai

IP-alapú virtuális hoszt

Minden webhelyhez külön IP-címet rendel. Hasznos, ha SSL-t (külön tanúsítványokat) kell használni minden domainhez.

Példa: (`192.168.1.10` → `example.com`; `192.168.1.11` → `example.org`)

Példa konfiguráció külön IP-címekhez:

```
<VirtualHost 192.168.1.10:80>
    ServerName example.com
    DocumentRoot /var/www/example.com
</VirtualHost>

<VirtualHost 192.168.1.11:80>
    ServerName example.org
    DocumentRoot /var/www/example.org
</VirtualHost>
```

Név-alapú virtuális hoszt

Egyetlen IP-címet használ több domain kiszolgálására. A webhelyeket a domainnév alapján különbözteti meg. Ez a leggyakoribb és leginkább ajánlott megoldás.

Példa konfiguráció két domainhez (`example.com és example.org`):

```
<VirtualHost *:80>
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/example.com
    ErrorLog ${APACHE_LOG_DIR}/example.com-error.log
    CustomLog ${APACHE_LOG_DIR}/example.com-access.log combined
</VirtualHost>

<VirtualHost *:80>
    ServerName example.org
    ServerAlias www.example.org
    DocumentRoot /var/www/example.org
    ErrorLog ${APACHE_LOG_DIR}/example.org-error.log
    CustomLog ${APACHE_LOG_DIR}/example.org-access.log combined
</VirtualHost>
```

ServerName: Az elsődleges domain neve.

ServerAlias: Alternatív nevek, például www.example.com.

DocumentRoot: Az a mappa, ahol az adott webhely fájljai találhatók.

6.1 Két különböző virtuális hoszt létrehozása:

6.1.1 Weboldalak könyvtárainak létrehozása: hozzunk létre külön könyvtárakat a két weboldal számára, például:

```
sudo mkdir -p /var/www/site1.com/public_html
sudo mkdir -p /var/www/site2.com/public_html
```

6.1.2 Könyvtárak jogosultságainak beállítása:

Debian rendszereken az Apache általában a **www-data** felhasználóval fut. Ennek a felhasználónak olvasási jogosultsággal kell rendelkeznie a weboldal könyvtárára és fájljaira.

```
sudo chown -R www-data:www-data /var/www/site1.com/public_html
sudo chown -R www-data:www-data /var/www/site2.com/public_html
sudo chmod -R 755 /var/www
```

6.1.3 Weboldal fájlok létrehozása: hozzunk létre egy-egy tesztfájlt mindkét könyvtárban, például:

```
sudo nano /var/www/site1.com/public_html/index.html
sudo nano /var/www/site2.com/public_html/index.html
```

6.1.4 Virtuális hoszt konfigurációs fájlok létrehozása:

site1.com.conf: `sudo nano /etc/apache2/sites-available/site1.com.conf`

A fájl tartalma:

```
<VirtualHost *:80>
    ServerName site1.com
    ServerAlias www.site1.com
    DocumentRoot /var/www/site1.com/public_html

    <Directory /var/www/site1.com/public_html>
        Options -Indexes +FollowSymLinks
        AllowOverride All
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/site1.com_error.log
    CustomLog ${APACHE_LOG_DIR}/site1.com_access.log combined
</VirtualHost>
```

site2.com.conf: `sudo nano /etc/apache2/sites-available/site2.com.conf`

A fájl tartalma:

```
<VirtualHost *:80>
    ServerName site2.com
    ServerAlias www.site2.com
    DocumentRoot /var/www/site2.com/public_html

    <Directory /var/www/site2.com/public_html>
        Options -Indexes +FollowSymLinks
        AllowOverride All
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/site2.com_error.log
    CustomLog ${APACHE_LOG_DIR}/site2.com_access.log combined
</VirtualHost>
```

6.1.5 Virtuális hosztok engedélyezése:

```
sudo a2ensite site1.com.conf
sudo a2ensite site2.com.conf
```

6.1.6 Apache újraindítása:

```
sudo systemctl reload apache2
```

6.1.7 Helyi tesztelés: Ha a domainnevek még nincsenek DNS-rekordhoz kötve, tesztelhetjük őket a hosts fájl módosításával: `sudo nano /etc/hosts`

Adjuk hozzá a következő 2 sort:

```
127.0.0.1 site1.com
127.0.0.1 site2.com
```

Nyissunk meg egy böngészőt, és írjuk be:

```
http://site1.com
http://site2.com
```

---

7. Modulok engedélyezése

1. **Rewrite modul engedélyezése:** A mod_rewrite szükséges lehet URL-ek átírásához:
   ```
   sudo a2enmod rewrite sudo
   systemctl reload apache2
   ```
2. **SSL modul engedélyezése (HTTPS támogatáshoz):**
   ```
   sudo a2enmod ssl
   sudo systemctl reload apache2
   ```
3. **PHP támogatás hozzáadása:** Telepítse a PHP-t, ha dinamikus tartalmat szeretne kiszolgálni:
   ```
   sudo apt install php libapache2-mod-php sudo
   systemctl restart apache2
   ```

---

8. SSL tanúsítvány beállítása (opcionális)

1. Telepítse a Let's Encrypt kliensprogramot: `sudo apt install certbot python3-certbotapache`
2. Tanúsítvány generálása: `sudo certbot --apache`
3. Automatikus megújítás ellenőrzése: `sudo certbot renew --dry-run`

---

9. Konfiguráció ellenőrzése

Az Apache konfigurációs hibáinak ellenőrzése: `sudo apache2ctl configtest`
Ha az eredmény Syntax OK, a konfiguráció helyes.

---

10. Hibaelhárítás

- Ellenőrizze az Apache naplófájljait: `sudo tail -f /var/log/apache2/error.log`
- Győződjön meg róla, hogy a megfelelő portok nyitva vannak: `sudo netstat -tuln | grep 80`

---

11. Jelszavas védelem beállítása weboldalhoz:

11.1 A .htaccess használatának engedélyezése:

Először ellenőrizzük, hogy az Apache **AllowOverride All** be van-e állítva a megfelelő könyvtárra. Nyissuk meg az adott virtuális host fájlt:

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

Írjuk be:

```
<Directory /var/www/html>
    AllowOverride All
</Directory>
```

11.2 A .htpasswd fájl létrehozása

A .htpasswd fájlban lesznek a jelszavak. Futtassuk az alábbi parancsot a fájl létrehozásához és egy felhasználó hozzáadásához (pl. admin):

```
sudo htpasswd -c /etc/apache2/.htpasswd admin
```

A -c kapcsoló új fájlt hoz létre. Ha már létezik a .htpasswd fájl ezt hagyjuk el, hogy ne írjuk felül az előzőt! Ezután kéri a jelszót, írjuk be kétszer.
Ha további felhasználót akarunk hozzáadni:

```
sudo htpasswd /etc/apache2/.htpasswd másikfelhasználó
```

11.3 A .htaccess fájl beállítása

Most a weboldal könyvtárában (/var/www/html/ vagy ahol a weboldal van) hozzunk létre egy .htaccess fájlt:

```
sudo nano /var/www/html/.htaccess
```

Írjuk bele a következőket:

```
AuthType Basic
AuthName "Protected Area"
AuthUserFile /etc/apache2/.htpasswd
Require valid-user
```

Magyarázat:

- **AuthType Basic** → Alapvető HTTP-hitelesítést használunk.
- **AuthName** → Az üzenet, amit a böngésző megjelenít a bejelentkezésnél.
- **AuthUserFile** → Az útvonal a .htpasswd fájlhoz.
- **Require valid-user** → Csak a hitelesített felhasználók férhetnek hozzá.

11.4 Ellenőrzés és az Apache újraindítása

Futtassuk az alábbi parancsokat, hogy az Apache betöltse az új beállításokat:

```
sudo apachectl configtest
sudo systemctl restart apache2
```

Ha a configtest hiba nélkül lefut, az Apache újraindítása után működni fog a jelszavas védelem.

Mire használható a mod_rewrite modul?

1. **Felhasználóbarát URL-ek létrehozása (SEO-barát URL-ek):**
    - Átalakíthatod a dinamikus URL-eket egyszerűbb, könnyebben olvasható formára.
    - Példa:
        - Eredeti URL: https://example.com/index.php?page=products&category=shoes
        - Átalakított URL: https://example.com/products/shoes
2. **Átirányítások kezelése:**
    - Átirányíthatod a régi URL-eket új URL-ekre, például ha egy oldal URL-je megváltozott.
    - Példa:
        - Régi URL: https://example.com/old-page
        - Új URL: https://example.com/new-page
3. **HTTPS kényszerítése:**
    - Biztosíthatod, hogy minden kérés automatikusan a HTTPS protokollt használja.
    - Példa:
        - http://example.com → https://example.com
4. **WWW előtag hozzáadása vagy eltávolítása:**
    - Eldöntheted, hogy a weboldalad URL-je tartalmazza-e a www előtagot, vagy anélkül jelenjen meg.
    - Példa:
        - http://example.com → http://www.example.com
        - Vagy fordítva: http://www.example.com → http://example.com
5. **Tartalomirányítás különböző nyelvekhez vagy területekhez:**
    - Nyelvi vagy régiós alapú URL-ek létrehozása.
    - Példa:
        - https://example.com/en/ → angol tartalom
        - https://example.com/hu/ → magyar tartalom