802.11N LINUX SERVER CONFIGURATIONS AND SERVICES

Part 1: Install 802.11N Drivers

Resources:

- GitHub Repository [https://github.com/gnab/rtl8812au]
- YouTube Video [https://www.youtube.com/watch?v=fVBE9gDs_aQ] 🚀🚀🚀

Part 2: Set Hotspot on the Wi-Fi Card (usually wlan1)

Follow the guide here: Create Wi-Fi Access Point/Hotspot in Linux 📡📡📡

Part 3: Install MySQL in Linux

```
sudo apt-get update
sudo apt-get install mysql-server mysql-client
```

🎉🎉🎉

Part 4: Configure MySQL in Linux

```
sudo mysql -u root

CREATE USER 'yourusername'@'localhost' IDENTIFIED BY 'yourpassword';
GRANT ALL PRIVILEGES ON HDSS.* TO 'hydot'@'localhost';
FLUSH PRIVILEGES;

exit;
```

🔥🔥🔥

Part 5: Make MySQL Start on Startup

```
sudo systemctl enable mysql
sudo systemctl start mysql
sudo systemctl status mysql
```

🛏️🛏️

Part 6: Configure .NET to Use MySQL Database and Allow Remote Connections

Steps:

1. Delete the `Migrations` folder (if it exists).
2. Install Entity Framework Core and MySQL packages:

   ```
   dotnet add package Microsoft.EntityFrameworkCore
   dotnet add package Pomelo.EntityFrameworkCore.MySql
   ```

3. Update your `DbContext` class:

   ```
   protected override void OnConfiguring(DbContextOptionsBuilder
   optionsBuilder)
   {
       if (!optionsBuilder.IsConfigured)
       {

   optionsBuilder.UseMySQL("Server=localhost;Port=3306;Database=HDSS;User=hyd
   ot;Password=SolDanKoHy1;");
       }
   }
   ```

4. In `Program.cs`, configure Kestrel:

   ```
   serverOptions.ListenAnyIp(5000);
   ```

📋📋📋

Part 7: Make the .NET Application Start Automatically

Steps:

1. Create a systemd service file:

   ```
   sudo nano /etc/systemd/system/hdss.service
   ```

2. Add the following content:

   ```
   [Unit]
   Description=Hydot School System
   After=network.target

   [Service]
   WorkingDirectory=/home/hydot/Desktop/HDSS_Backend
   ExecStart=/usr/bin/dotnet watch run
   Restart=always
   RestartSec=10

   [Install]
   WantedBy=multi-user.target
   ```

3. Enable and start the service:

   ```
   sudo systemctl enable hdss
   ```

```
sudo systemctl start hdss
```

Part 8: Install Webmin

Steps:

1. Update your package list:

   ```
   sudo apt update
   ```

2. Install dependencies:

   ```
   sudo apt install wget apt-transport-https software-properties-common
   ```

3. Add the Webmin repository:

   ```
   wget -qO- http://www.webmin.com/jcameron-key.asc | sudo apt-key add -
   sudo add-apt-repository "deb
   http://download.webmin.com/download/repository sarge contrib"
   ```

4. Install Webmin:

   ```
   sudo apt update
   sudo apt install webmin
   ```

5. Access Webmin:
   - Open a browser and navigate to `https://<server_ip>:10000`.
   - Log in with your kali linux login credentials

Part 9: Install phpMyAdmin

Steps:

1. Install phpMyAdmin:

   ```
   sudo apt install phpmyadmin
   ```

2. Configure Apache (or Nginx) for phpMyAdmin:
   - For Apache:

     ```
     sudo nano /etc/apache2/conf-enabled/phpmyadmin.conf
     ```

     Ensure the following lines exist:

     ```
     Alias /phpmyadmin /usr/share/phpmyadmin

     <Directory /usr/share/phpmyadmin>
         Options FollowSymLinks
         DirectoryIndex index.php
         Require all granted
     </Directory>
     ```

Restart Apache:

```
sudo systemctl restart apache2
```

🌐🌐🌐

Part 10: Create Virtual Hosts and Install Nginx

Steps:

1. Install Nginx:

   ```
   sudo apt install nginx
   ```

2. Create a virtual host:

   ```
   sudo nano /etc/nginx/sites-available/myapp
   ```

3. Add the following configuration:

   ```
   server {
       listen 80;
       server_name myapp.local;

       location / {
           proxy_pass http://127.0.0.1:3000; # Adjust port as needed
           proxy_set_header Host $host;
           proxy_set_header X-Real-IP $remote_addr;
           proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
       }
   }
   ```

4. Enable the site:

   ```
   sudo ln -s /etc/nginx/sites-available/myapp /etc/nginx/sites-enabled/
   ```

5. Restart Nginx:

   ```
   sudo systemctl restart nginx
   ```

✨✨✨

Part 11: Run the Frontend on Nginx on a Specific Port

Steps:

1. Build your frontend application (React/Vue/Angular):

   ```
   npm run build
   ```

2. Copy the build files to `/var/www/myfrontend`:

   ```
   sudo cp -r build/* /var/www/myfrontend/
   ```

3. Update the Nginx configuration:

```
sudo nano /etc/nginx/sites-available/frontend
```

4. Add the configuration:

```
server {
    listen 8080;
    server_name frontend.local;

    root /var/www/myfrontend;
    index index.html;

    location / {
        try_files $uri /index.html;
    }
}
```

5. Enable the site and restart Nginx:

```
sudo ln -s /etc/nginx/sites-available/frontend /etc/nginx/sites-enabled/
sudo systemctl restart nginx
```

🎨🎨🎨

Part 12: Configure Subdomains Locally

Steps:

1. Open the `hosts` file on your local machine:

```
sudo nano /etc/hosts
```

2. Add entries for your subdomains:

```
127.0.0.1 myapp.local
127.0.0.1 frontend.local
```

3. Save the file and test accessing the subdomains in your browser:
   - `http://myapp.local`
   - `http://frontend.local:8080` 🏠🏠🏠

Part 13: Install Dotnet

Steps:

4. Add the Microsoft Package Repository:

```
wget https://packages.microsoft.com/config/debian/11/packages-microsoft-prod.deb -O packages-microsoft-prod.deb

sudo dpkg -i packages-microsoft-prod.deb
```

5. Install Required Dependencies:

   ```
   sudo apt update
   sudo apt install -y apt-transport-https
   ```

6. Install .NET SDK or Runtime

   sudo apt install -y dotnet-sdk-8.0

   dotnet --version

Final Note

This guide covers setting up essential configurations for a Linux server to handle various services, applications, and subdomains efficiently. Ensure you adapt configurations as necessary to fit your specific needs. 📚📚📚