

# HTTP Server

## Inhaltsverzeichnis

1. Apache HTTP Server - Anleitung
2. Anpassen und Anzeigen deiner eigenen Webseite
3. Ports
4. Nginx Server - Anleitung
5. Deinstallieren von Apache

## 1. Warum WSL für Docker bereits installiert ist

Docker benötigt WSL 2, um auf einem echten Linux-Kernel zu laufen. Deshalb wird WSL 2 automatisch durch Docker auf deinem Windows-Rechner installiert.

## 2. Wie WSL mit Ubuntu unter Windows zu nutzen ist

Da WSL bereits installiert ist (durch Docker), kannst du nun Ubuntu direkt unter Windows verwenden. Hier ist eine Schritt-für-Schritt-Anleitung:

### Schritt 1: Installiere Ubuntu aus dem Microsoft Store (falls noch nicht installiert)

1. Öffne den **Microsoft Store** auf deinem Windows-PC.
2. Suche nach "**Ubuntu**" in der Suchleiste.
3. Wähle die gewünschte Ubuntu-Version aus (z. B. Ubuntu 20.04 LTS oder 22.04 LTS) und klicke auf **Installieren**.

Nachdem du Ubuntu im Microsoft Store installiert hast, kannst du einfach auf die Ubuntu-Option im Startmenü klicken, um die erste Einrichtung abzuschließen und einen Benutzer anzulegen.

### Schritte zum Erstellen eines Benutzers nach dem Start von Ubuntu in WSL:

1. **Ubuntu starten:**

- Klicke im **Startmenü** auf „**Ubuntu**“ oder suche nach „Ubuntu“, um es zu öffnen.
- Beim ersten Start wird Ubuntu einige Minuten benötigen, um die Umgebung zu initialisieren.

## 2. Benutzer und Passwort einrichten:

- Beim ersten Start wirst du aufgefordert, einen **Benutzernamen** und ein **Passwort** zu erstellen.
- Gib einen **Benutzernamen** ein (dies ist der Name, den du verwenden wirst, um dich in der Ubuntu-Shell anzumelden).
- Danach wirst du nach einem **Passwort** gefragt. Gib ein sicheres Passwort ein, das du dir merken kannst.

## 3. Abschließen der Einrichtung:

- Nach der Eingabe des Benutzernamens und Passworts wird Ubuntu die Installation abschließen und du bist mit deinem neuen Benutzer in der Ubuntu-Umgebung.

## 4. Fertig!:

- Nun bist du bereit, Ubuntu unter WSL zu verwenden. Du kannst Linux-Kommandos ausführen und das System nach Belieben konfigurieren.

## Beispiel:

Nach dem Öffnen von Ubuntu könnte es so aussehen:

```
Enter new UNIX username: meinbenutzer
Enter new UNIX password: *****
Retype new UNIX password: *****
```

Nachdem du die Eingaben gemacht hast, wird Ubuntu die Umgebungsdateien einrichten und du kannst den Linux-Befehl `sudo` verwenden, um mit Administratorrechten zu arbeiten.

## Weiterarbeiten:

- Wenn du möchtest, kannst du auch den **Standardbenutzer** festlegen, der bei jedem Start automatisch eingeloggt wird.
- Falls du später Änderungen an deinem Benutzer oder Passwort vornehmen möchtest, kannst du dies mit den üblichen Linux-Befehlen tun (z. B. `passwd` für das Passwort ändern).

Jetzt kannst du Ubuntu unter WSL wie gewohnt verwenden!

## Zusammenfassung:

- **Warum WSL für Docker bereits installiert ist:** Docker benötigt WSL 2, um auf einem echten Linux-Kernel zu laufen. Deshalb wird WSL 2 automatisch durch Docker auf deinem Windows-Rechner installiert.
- **Wie du Ubuntu unter WSL verwendest:** Nach der Installation von Ubuntu im Microsoft Store kannst du WSL 2 aktivieren, Ubuntu starten und alle gewohnten Linux-Kommandos innerhalb von Windows ausführen.

## 1. Apache HTTP Server - Anleitung

Apache ist unter den meisten Unix-Systemen standardmäßig vorhanden, trotzdem werden hier die Installationsschritte dargestellt.

### Schritt 1: Installation von Apache

1. **Installiere Apache:** Apache kann über den Paketmanager deiner Linux-Distribution installiert werden. Die Befehle variieren je nach Distribution. Zum Beispiel:

```
# Für Ubuntu/Debian
sudo apt update
sudo apt install apache2
```

Dieser Befehl aktualisiert zunächst die Paketlisten und installiert dann Apache2 auf dem System. Apache2 ist der Standardname für den Apache HTTP Server in Ubuntu und Debian.

```
# Für CentOS/RHEL
```

```
sudo yum install httpd
```

`httpd` ist der Paketname für Apache in CentOS und RHEL (Red Hat Enterprise Linux).

## Schritt 2: Starten und Aktivieren des Apache-Dienstes

1. **Starte den Apache-Service:** Nach der Installation muss der Apache-Service gestartet werden, damit er läuft und Anfragen verarbeiten kann.

```
sudo systemctl start apache2    # Für Ubuntu/Debian
```

```
sudo systemctl start httpd      # Für CentOS/RHEL
```

Dieser Befehl startet den Apache-Service.

2. **Aktiviere den Apache-Service:** Damit Apache beim Start des Systems automatisch gestartet wird, muss der Service aktiviert werden.

```
sudo systemctl enable apache2   # Für Ubuntu/Debian
```

```
sudo systemctl enable httpd     # Für CentOS/RHEL
```

Indem du Apache aktivierst, stellst du sicher, dass er beim Booten des Systems automatisch gestartet wird.

## Schritt 3: Konfiguration von Apache

1. **Dokumentwurzel festlegen:** Die Dokumentwurzel ist das Hauptverzeichnis, in dem Apache nach Webinhalten sucht, die er an die Benutzer sendet. Standardmäßig ist die Dokumentwurzel auf den Pfad `/var/www/html` festgelegt.

```
sudo cat /etc/apache2/apache2.conf    # Für Ubuntu/Debian
```

```
sudo cat /etc/httpd/conf/httpd.conf   # Für CentOS/RHEL
```

Die Dokumentwurzel in einer Apache-Serverkonfiguration ist das Verzeichnis, in dem der Server nach Dateien sucht, um Anfragen von Clients zu bedienen. In dieser Konfigurationsdatei wird die Dokumentwurzel innerhalb der `<Directory>`-Direktive für `/var/www/` festgelegt:

```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

In diesem Fall ist die Dokumentwurzel also `/var/www/`. Das bedeutet, dass alle Dateien oder Verzeichnisse innerhalb von `/var/www/` von Clients, die Anfragen an den Server stellen, abgerufen werden können.

```
Require all denied
</Directory>

<Directory /usr/share>
    AllowOverride None
    Require all granted
</Directory>

<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

#<Directory /srv/>
#     Options Indexes FollowSymLinks
#     AllowOverride None
#     Require all granted
#</Directory>
```

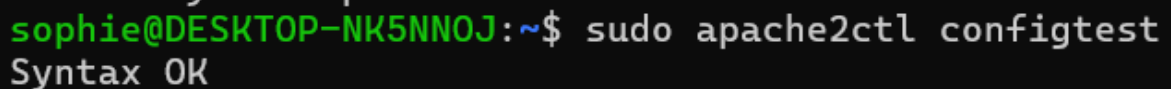
#### Schritt 4: Überprüfung der Konfiguration und Neustart von Apache

1. **Überprüfe die Apache-Konfiguration auf Fehler:** Vor dem Neustart von Apache ist es ratsam, die Konfiguration auf Fehler zu überprüfen.

```
sudo apache2ctl configtest    # Für Ubuntu/Debian
```

```
sudo apachectl configtest     # Für CentOS/RHEL
```

Dieser Befehl überprüft die Syntax der Apache-Konfigurationsdateien auf Fehler. Falls Fehler auftreten, werden sie angezeigt, andernfalls erhältst du eine Bestätigung, dass die Syntax in Ordnung ist.



```
sophie@DESKTOP-NK5NNOJ:~$ sudo apache2ctl configtest
Syntax OK
```

2. **Starte oder lade die Apache-Konfiguration neu:** Nachdem du die Konfiguration überprüft hast, kannst du den Apache-Service neu starten oder die Konfiguration neu laden, damit die Änderungen wirksam werden.

```
sudo systemctl reload apache2    # Für Ubuntu/Debian
```

```
sudo systemctl reload httpd      # Für CentOS/RHEL
```

Dieser Befehl lädt die Apache-Konfiguration neu, ohne den Apache-Dienst zu stoppen. Dadurch werden die Änderungen sofort wirksam.

Nachdem du diese Schritte ausgeführt hast, sollte Apache gemäß deiner Konfiguration betriebsbereit sein und bereit sein, Webseiten zu servieren. Du kannst deine Webseite in einem Webbrowser anzeigen, indem du die IP-Adresse oder den Domainnamen deines Servers besuchst.

### Verwenden von `localhost` in WSL

`localhost` wird verwendet, um eine Webseite im Browser anzuzeigen, wenn Ubuntu in WSL verwendet wird. In der Regel ist `localhost` ein Alias für die lokale IP-Adresse deines eigenen Systems (127.0.0.1), unabhängig davon, ob Windows oder WSL genutzt wird.

1. Öffne deinen bevorzugten Webbrowser auf deinem Windows-System.
2. Gib `http://localhost` in die Adressleiste des Browsers ein und drücke die Eingabetaste.
3. Dadurch wird der Browser eine Anfrage an den Apache-Webserver in deinem WSL-Ubuntu senden und die Webseite anzeigen, die auf deinem WSL-System gehostet wird.

Wenn Apache auf einem anderen Port als Port 80 läuft, musst du den Port spezifizieren, indem du ihn an die URL anhängst. Zum Beispiel, wenn Apache auf Port 8080 läuft, gib `http://localhost:8080` ein.

Die Verwendung von `localhost` ist eine praktische Möglichkeit, auf die lokal gehostete Webseite zuzugreifen, da du keine IP-Adresse kennen musst und es in den meisten Fällen einfach funktioniert.

### Was passiert hierbei?

Weiter oben haben wir folgendes gelesen: `/var/www/html` , aber was heißt das nun?

`/var/www/html` : Dies ist der tatsächliche Pfad auf dem Dateisystem, in dem die Dateien Ihrer Webseite gespeichert werden. In der Regel ist dies der Standardpfad für Webinhalte auf vielen Linux-Systemen. Das Verzeichnis `/var/www` wird häufig als Stammverzeichnis für Webinhalte verwendet, und innerhalb dieses Verzeichnisses wird das Unterverzeichnis `html` als DocumentRoot festgelegt.

Zusammengefasst bedeutet `/var/www/html` , dass der Apache-Server nach Dateien im Verzeichnis `/var/www/html` sucht, wenn eine Anfrage an Ihre Webseite gestellt wird. Wenn Sie also Ihre Webseite konfigurieren und Dateien wie HTML, CSS, Bilder usw. bereitstellen möchten, sollten Sie sie in dieses Verzeichnis platzieren.

### Befehlssequenz zum Untersuchen von `/var/www/html`

Um das Verzeichnis `/var/www/html` genauer zu betrachten, kannst du die folgende Befehlssequenz verwenden:

```
cd /var
ls
cd www
ls
cd html
```

```
ls  
cat index.html
```

**1. Wechsel in das Verzeichnis `/var` :**

```
cd /var
```

**2. Liste den Inhalt des Verzeichnisses `/var` auf:**

```
ls
```

Ausgabe

```
backups  cache  crash  lib  local  lock  log  mail  opt  run  
snap  spool  tmp  www
```

**3. Wechsel in das Verzeichnis `/var/www` :**

```
cd www
```

**4. Liste den Inhalt des Verzeichnisses `/var/www` auf:**

```
ls
```

Ausgabe

```
html
```

**5. Wechsel in das Verzeichnis `/var/www/html` :**

```
cd html
```

**6. Liste den Inhalt des Verzeichnisses `/var/www/html` auf:**



```
ls
```

Ausgabe

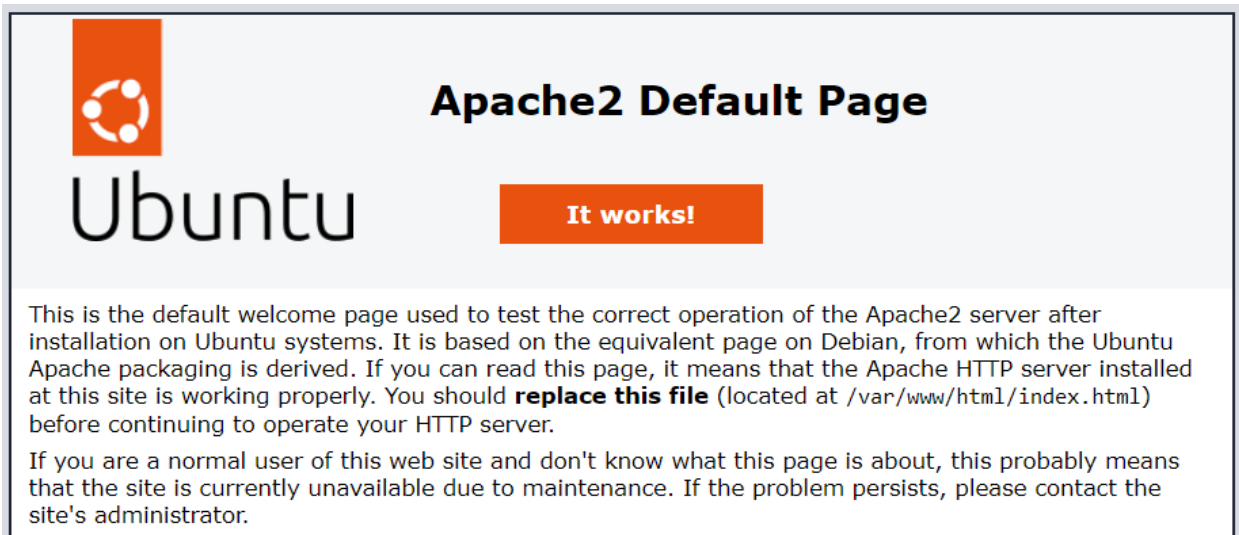
```
index.html
```

## 7. Zeige den Inhalt der Datei `index.html` an:

```
cat index.html
```

Nach dem Ausführen des Befehls `cat index.html` sollte der Inhalt der Datei `index.html` angezeigt werden.

Schließlich kann die Webseite auch über `http://localhost/` im Browser aufgerufen werden.



[1]

## 2. Anpassen und Anzeigen deiner eigenen Webseite

Um deine eigene Webseite zu erstellen und sie über den Apache-Server anzuzeigen, kannst du die folgende Anleitung befolgen:

1. Öffne dein Terminal und führe den Befehl `sudo vim newSite.html` aus, um eine neue HTML-Datei mit dem Namen `newSite.html` zu erstellen. Du kannst

alternativ einen Texteditor deiner Wahl verwenden.

```
sudo vim newSite.html
```

1. Füge den folgenden HTML-Code in die `newSite.html` Datei ein:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Meine Webseite</title>
</head>
<body>
  <h1>Willkommen auf meiner Webseite!</h1>
  <p>Hier ist mein erster Absatz.</p>
</body>
</html>
```

1. Speichere und schließe den Texteditor. Wenn du `vim` verwendest, drücke `Escape`, gib `:wq` ein und drücke die Eingabetaste.
2. Starte oder aktualisiere deinen Apache-Server, damit er die neue Webseite berücksichtigt.
3. Öffne deinen Webbrowser und navigiere zur Adresse `http://localhost/newSite.html`.

Dadurch wird deine neue Webseite angezeigt, und du solltest den Text "Willkommen auf meiner Webseite!" und "Hier ist mein erster Absatz." sehen können.

Das ist alles! Du hast erfolgreich deine eigene Webseite erstellt und sie über den Apache-Server zugänglich gemacht. Du kannst nun beliebige Änderungen an deiner Webseite vornehmen, indem du die `newSite.html` Datei bearbeitest und den Apache-Server bei Bedarf neu startest.

# Willkommen auf meiner Webseite!

Hier ist mein erster Absatz.

Um deine Webseite benutzerdefiniert zu gestalten und sie in einem eigenen Verzeichnis mit einem dazugehörigen Stylesheet zu organisieren, führe die folgenden Schritte aus:

1. Öffne dein Terminal und führe die folgenden Befehle aus:

```
sudo mkdir mySite
```

Dieser Befehl erstellt ein neues Verzeichnis mit dem Namen `mySite` im Verzeichnis `/var/www/html`.

```
sudo mv newSite.html mySite
```

Durch diesen Befehl verschiebst du die Datei `newSite.html` in das Verzeichnis `mySite`. Falls du eine Berechtigungsfehlermeldung erhältst, füge `sudo` vor den Befehl ein, um die nötigen Berechtigungen zu erhalten.

1. Öffne die Webseite `http://localhost/mySite/newSite.html` in deinem Webbrowser.

Dadurch wird die Webseite angezeigt, die den Inhalt der `newSite.html` Datei darstellt, welche sich jetzt im Verzeichnis `mySite` befindet.

1. Optional: Um das Design deiner Webseite anzupassen, kannst du eine CSS-Datei erstellen. Führe dazu die folgenden Befehle aus:

```
sudo touch styles.css
```

Dieser Befehl erstellt eine neue CSS-Datei mit dem Namen `styles.css` im Verzeichnis `mySite`.

1. Füge den folgenden CSS-Code in die `styles.css` Datei ein:

```

/* styles.css */
.container {
    display: grid;
    background-color: #666; /* Hintergrundfarbe des Containers */
    height: 100vh; /* Vollständige Bildschirmhöhe für den Container */
    place-items: center; /* Zentriert den Inhalt horizontal und vertikal */
}

h1 {
    color: #3dbdbd; /* türkis */
}

p {
    color: #FFAC33; /* Sonnenblumengelb */
}

```

Dieser CSS-Code definiert das Aussehen der Elemente auf deiner Webseite.

1. Aktualisiere die `newSite.html` Datei, um die `styles.css` Datei einzubinden. Füge dazu den folgenden `<link>` Tag im `<head>` Bereich der Datei ein:

```
<link rel="stylesheet" href="styles.css">
```

1. Speichere und schließe die Dateien.

Durch diese Schritte hast du erfolgreich deine eigene Webseite erstellt, sie in das Verzeichnis `mySite` verschoben und sie mithilfe von Apache angezeigt. Du kannst jetzt weitere Anpassungen an deiner Webseite vornehmen, indem du die Dateien in `mySite` bearbeitest und den Browser aktualisierst, um die Änderungen anzuzeigen.



**Willkommen auf meiner Webseite!**

Hier ist mein erster Absatz.

Schließlich wird über den Browser die URL `http://localhost/mySite/newSite.html` aufgerufen und der Browser zeigt den Inhalt der `newSite.html` an, die sich jetzt im Verzeichnis `mySite` befindet.

## 4. Ports

Ports sind numerische Bezeichnungen, die verwendet werden, um verschiedene Netzwerkdienste auf einem Computer zu identifizieren. Sie sind Teil des TCP/IP-Protokolls (Transmission Control Protocol/Internet Protocol), das die Grundlage des Internets bildet. Jeder Port ist einer bestimmten Anwendung oder einem bestimmten Dienst auf einem Host zugeordnet, wodurch die Kommunikation

zwischen verschiedenen Anwendungen ermöglicht wird. Hier sind einige häufig verwendete Ports und ihre typischen Anwendungen:

1. **Port 80 (HTTP):** Dieser Port wird standardmäßig für HTTP (Hypertext Transfer Protocol) verwendet, das für den Austausch von Webseiten im World Wide Web verwendet wird. Webserver wie Apache und Nginx verwenden diesen Port, um Webinhalte an Benutzer zu senden.
2. **Port 443 (HTTPS):** HTTPS (Hypertext Transfer Protocol Secure) ist die verschlüsselte Version von HTTP, die zur sicheren Übertragung von Webseiteninhalten verwendet wird. HTTPS nutzt den Port 443 für die verschlüsselte Kommunikation zwischen dem Webserver und dem Client, wobei SSL/TLS-Zertifikate zur Authentifizierung und Verschlüsselung eingesetzt werden.
3. **Port 22 (SSH):** Dieser Port wird für SSH (Secure Shell) verwendet, ein Protokoll zur sicheren Remote-Verbindung zu einem anderen Computer. SSH ermöglicht eine verschlüsselte Kommunikation und wird oft von Systemadministratoren verwendet, um sich auf entfernten Servern anzumelden und Befehle auszuführen.
4. **Port 25 (SMTP):** SMTP (Simple Mail Transfer Protocol) ist ein Protokoll zum Senden von E-Mails zwischen Servern. Port 25 wird verwendet, um ausgehende E-Mails zu senden, und wird von Mailservern verwendet, um E-Mails zwischen verschiedenen Domänen zu übertragen.
5. **Port 53 (DNS):** DNS (Domain Name System) ist ein Protokoll, das zur Auflösung von Domainnamen in IP-Adressen verwendet wird. Port 53 wird für die DNS-Anfragen und -Antworten zwischen DNS-Servern und Clients verwendet.
6. **Port 21 (FTP):** FTP (File Transfer Protocol) ist ein Protokoll zum Übertragen von Dateien zwischen einem Client und einem Server über ein Netzwerk. Port 21 wird für die Steuerung der FTP-Verbindungen verwendet, während Port 20 für die eigentliche Datenübertragung verwendet wird.
7. **Port 3306 (MySQL):** Dieser Port wird standardmäßig von MySQL-Datenbanken verwendet. MySQL ist ein relationales Datenbankmanagementsystem, das für die Speicherung und Verwaltung von Daten in vielen Webanwendungen verwendet wird.

Dies sind nur einige Beispiele für die vielen Ports, die im TCP/IP-Protokoll verwendet werden. Jeder Port hat eine spezifische Bedeutung und wird von entsprechenden Anwendungen oder Diensten genutzt, um die Kommunikation über das Netzwerk zu ermöglichen.

Wenn der Standard-HTTP-Port 80 bereits von einem anderen Dienst belegt ist oder aus Sicherheitsgründen geändert werden soll, gibt es mehrere alternative Ports, die für HTTP verwendet werden können. Hier sind einige davon:

1. **Port 8080:** Port 8080 ist einer der häufigsten alternativen Ports für HTTP. Er wird oft verwendet, um Webserver aufzusetzen und zu testen, ohne die Standardkonfiguration zu ändern.
2. **Port 8000:** Ähnlich wie Port 8080 wird auch Port 8000 häufig als alternativer HTTP-Port verwendet. Er wird oft für lokale Entwicklungsserver oder für das Hosting von Webanwendungen verwendet.
3. **Port 8888:** Port 8888 ist ein weiterer beliebter alternativer HTTP-Port. Er wird häufig für verschiedene Anwendungen und Entwicklungsumgebungen verwendet.
4. **Port 8081, 8001, 8889 usw.:** Es gibt viele weitere Ports im Bereich von 8000 bis 9000, die alternativ für HTTP genutzt werden können. Diese können je nach Bedarf ausgewählt werden, solange sie nicht bereits von anderen Diensten belegt sind.

[2][3]

## 5. Nginx Server - Anleitung

### Schritt 1: Installation von Nginx

1. **Installiere Nginx:** Öffne dein Terminal in WSL und führe die folgenden Befehle aus, um Nginx zu installieren:

```
sudo apt update  
sudo apt install nginx
```

Dies installiert Nginx auf deinem System.

### Schritt 2: Konfiguration von Nginx für die Nutzung eines anderen Ports

1. **Betrachten der Nginx-Konfigurationsdatei:** Lasse dir die Nginx-Konfigurationsdatei:

```
sudo cat /etc/nginx/sites-enabled/default
```

Die Ausgabe des Befehls `sudo cat /etc/nginx/sites-enabled/default` zeigt den Inhalt der Konfigurationsdatei für den Standard-Virtual-Host in Nginx an. Diese Datei enthält die Konfigurationseinstellungen für die Standard-Website, die von Nginx bedient wird, wenn keine spezifische Konfiguration für eine andere Website vorhanden ist.

In der Regel wird diese Datei beim Installieren von Nginx mitgeliefert und enthält eine grundlegende Konfiguration, um Webanfragen auf den Server zu behandeln. Der Inhalt kann je nach Nginx-Version und Distribution variieren, aber typischerweise sieht eine einfache Konfiguration für den Standard-Virtual-Host in Nginx so aus:

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    server_name _;

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

Diese Konfiguration teilt Nginx mit, auf Port 80 auf eingehende Anfragen zu lauschen. Die Dateien für die Website befinden sich im Verzeichnis `/var/www/html`, und die Standardindexdateien sind `index.html`, `index.htm` und `index.nginx-debian.html`. Die `server_name` ist als Platzhalter `_` angegeben, was



bedeutet, dass diese Konfiguration für alle Anfragen gilt, die nicht von anderen virtuellen Hosts abgedeckt werden.

Je nach den Anforderungen deines Systems kann die Konfiguration in dieser Datei angepasst werden, um verschiedene Websites zu hosten oder spezifische Anforderungen zu erfüllen.

1. **Öffne die Nginx-Konfigurationsdatei:** Verwende einen Texteditor wie Nano oder VIM, um die Nginx-Konfigurationsdatei zu öffnen:

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

oder

```
sudo vim /etc/nginx/sites-available/default
```

2. **Ändere den Port:** Suche nach der Zeile `listen 80 default_server;` und ändere sie in den gewünschten Port, zum Beispiel `listen 8080 default_server;`. Du kannst den Port nach Bedarf anpassen, jedoch wird oft Port 8080 als Alternative für HTTP verwendet.

### Schritt 3: Starten und Aktivieren des Nginx-Dienstes

1. **Starte den Nginx-Service:** Verwende die folgenden Befehle, um den Nginx-Service zu starten:

```
sudo systemctl start nginx
```

Dieser Befehl startet den Nginx-Service.

2. **Aktiviere den Nginx-Service:** Damit Nginx beim Start des Systems automatisch gestartet wird, aktiviere den Service:

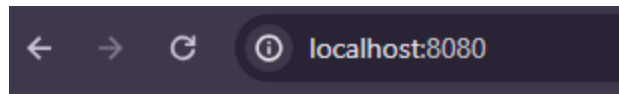
```
sudo systemctl enable nginx
```

Auf diese Weise wird sichergestellt, dass Nginx beim Hochfahren des Systems automatisch gestartet wird.

[4]

## Verwenden von `localhost` in WSL

Nutzen wir jetzt `http://localhost:8080/`, um auf die Nginx-Webseite zuzugreifen, sehen wir keinen Unterschied zu vorher.



**Herzlich willkommen!**

Dies ist eine Testseite.

Daher passen wir die Dokumentenwurzel an.

1. **Erstellen eines neuen Ordners für Nginx:** Zuerst erstelle einen neuen Ordner für Nginx. Du könntest zum Beispiel einen Ordner namens `nginx` im Verzeichnis `/var/www/html` erstellen:

```
sudo mkdir /var/www/html/nginx
```

2. **Ändern der Nginx-Konfiguration:** Du musst die Nginx-Konfigurationsdatei bearbeiten, um den neuen Pfad anzugeben, auf den Nginx zugreifen soll. Öffne die Nginx-Konfigurationsdatei mit einem Texteditor:

Über nano

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

Oder über vim

```
sudo vim /etc/nginx/sites-available/default
```

Ändere dann den `root`-Parameter in der Server-Konfiguration, um auf den neuen Ordner zu verweisen. Zum Beispiel:

```
server {  
    ...  
    root /var/www/html/nginx;  
    ...  
}
```

Speichere die Änderungen und schließe die Datei.

3. **Ändern der Indexdatei:** Erstelle oder ändere die Indexdatei für Nginx im neuen Ordner, um die gewünschte Nachricht anzuzeigen:

```
sudo nano /var/www/html/nginx/index.html
```

oder

```
sudo vim /var/www/html/nginx/index.html
```

Ändere den Inhalt der Datei:

```
<html lang="de">  
<head>  
    <meta charset="UTF-8">  
    <title>Herzlich willkommen!</title>  
</head>  
<body>  
    <h1>Dies ist eine Testseite über Nginx, um zu schauen,  
    ob dies klappt.</h1>  
    <p>Hier kannst du beliebigen HTML-Inhalt einfügen.</p>  
</body>  
</html>
```

Speichere die Änderungen und schließe die Datei.

4. **Neuladen von Nginx:** Nachdem du die Konfiguration und die Indexdatei geändert hast, lade die Nginx-Konfiguration neu, damit die Änderungen wirksam werden:

```
sudo systemctl reload nginx
```

Nachdem diese Schritte durchgeführt wurden, sollte Nginx auf den neuen Ordner zugreifen und die Indexdatei anzeigen, wenn du `localhost:8080` in deinem Browser aufrufst.

5. **Überprüfe die Nginx-Konfiguration auf Fehler:** Führe den folgenden Befehl aus, um die Nginx-Konfiguration auf Fehler zu überprüfen:

```
sudo nginx -t
```

Falls Fehler auftreten, werden sie angezeigt. Andernfalls erhältst du eine Bestätigung, dass die Syntax in Ordnung ist.

```
sophie@DESKTOP-NK5NNOJ:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```



← → ↻ ⓘ localhost:8080

**Dies ist eine Testseite über Nginx, um zu schauen, ob dies klappt.**

Hier kannst du beliebigen HTML-Inhalt einfügen.

## 6. Deinstallieren von Apache

Um Apache zu deinstallieren und Nginx auf einem anderen Port als Port 80 zu verwenden, befolge diese Schritte:

1. **Stoppen des Apache-Servers:** Öffne dein Terminal und stoppe den Apache-Server mit dem Befehl:

```
sudo systemctl stop apache2
```

**Stoppen des Apache-Servers:** Dieser Schritt beendet den laufenden Apache-Server, um sicherzustellen, dass er nicht im Konflikt mit anderen Diensten steht,

die denselben Port verwenden sollen. Der Befehl `sudo systemctl stop apache2` stoppt den Apache-Dienst sofort.

1. **Deaktivieren des Apache-Servers beim Systemstart:** Deaktiviere Apache, damit er beim Systemstart nicht automatisch gestartet wird:

```
sudo systemctl disable apache2
```

**Deaktivieren des Apache-Servers beim Systemstart:** Durch das Deaktivieren des Apache-Servers wird sichergestellt, dass er beim nächsten Neustart des Systems nicht automatisch gestartet wird. Dies ist wichtig, um sicherzustellen, dass der Apache-Dienst nicht unerwartet wieder gestartet wird. Der Befehl `sudo systemctl disable apache2` deaktiviert den Apache-Dienst.

1. **Entfernen von Apache:** Deinstalliere Apache von deinem System:

```
sudo apt purge apache2 apache2-utils
```

**Entfernen von Apache:** Mit diesem Schritt wird Apache vollständig von deinem System entfernt. Der Befehl `sudo apt purge apache2 apache2-utils` entfernt nicht nur den Apache-Webserver, sondern auch zugehörige Dienstprogramme und Konfigurationsdateien. Dies stellt sicher, dass alle Überreste von Apache entfernt werden.