Projektarbeit: Praxissetellenbewerten

David Damke

4. September 2024

Inhaltsverzeichnis

1 Einführung

In dieser Pdf wird gezeigt wie ein Vue App mit Node Backend auf einem Debian Server eingerichtet werden müssen. Dazu noch welche Schritte für das Rechenzentrum Notwendig sind und welche Versionen und Plugins verwendet wurden. Zum erstellen der Tutorials wurde ChatGPT verwendet.

2 Plugins und Versionen

2.1 Node.js

- **Node.js**: v20.11.0
- **Abhängigkeiten**:
  - `cors`: 2.8.5
  - `express`: 4.18.2
  - `ldapjs`: 3.0.7
  - `mongodb`: 6.3.0

2.2 Vue.js

- **Vue.js**: 3.3.11
- **Abhängigkeiten**:
  - `axios`: 1.6.7
  - `vue-router`: 4.2.5
  - `vuetify`: 3.0.1
  - `vuex`: 4.1.0
  - `@mdi/font`: 7.4.47
  - `@vitejs/plugin-vue`: 4.5.2
  - `vite`: 5.0.10

3 Rechenzentrum

Rechenzentrum muss die eignenen Firewall Regeln so anpassen das die ServerIp auf den Ldap Server zugreiffen kann.

4 Einrichtung auf Debian Webserver

4.0.1 Verbindung zum Server

ssh name@ip password :

4.1 Apache2

4.1.1 Voraussetzungen

Bevor du mit der Einrichtung beginnst, benötigst du:

- Einen Debian-Server (Debian 10 oder 11 empfohlen)
- Root-Zugriff oder einen Benutzer mit sudo-Rechten
- Einen SSH-Client, um dich mit dem Server zu verbinden (z.B. PuTTY oder direkt über Terminal)

4.1.2 Schritt 1: Aktualisierung des Paketsystems

Bevor du neue Software installierst, solltest du das Debian-Paketsystem aktualisieren, um sicherzustellen, dass du die neuesten Versionen der Pakete erhältst.

```
sudo apt update
sudo apt upgrade
```

4.1.3 Schritt 2: Installation von Apache2

Apache2 ist in den Standard-Repositorys von Debian enthalten, daher kann die Installation mit dem folgenden Befehl durchgeführt werden:

```
sudo apt install apache2
```

Dieser Befehl installiert Apache2 sowie alle notwendigen Abhängigkeiten. Nach der Installation sollte der Apache-Webserver automatisch starten.

4.1.4 Schritt 3: Überprüfung des Apache-Dienstes

Nach der Installation kannst du überprüfen, ob der Apache-Dienst läuft, indem du den folgenden Befehl ausführst:

```
sudo systemctl status apache2
```

Du solltest eine Ausgabe erhalten, die bestätigt, dass der Dienst läuft. Zum Beispiel:

```
apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service;
        enabled; vendor preset: enabled)
     Active: active (running) since Mon 2021-08-16 10:10:02
        UTC; 1min 45s ago
```

4.1.5 Schritt 4: Zugriff auf den Webserver

Um sicherzustellen, dass der Webserver ordnungsgemäß funktioniert, öffne einen Webbrowser und gebe die IP-Adresse deines Servers ein. Wenn Apache korrekt installiert wurde, solltest du die Standard-Apache-Seite sehen, die besagt:

It works!

4.1.6 Schritt 5: Firewall-Konfiguration

Wenn auf deinem Server eine Firewall läuft, musst du sicherstellen, dass der Webserver-Verkehr (Port 80 für HTTP und 443 für HTTPS) zugelassen wird.

```
sudo ufw allow 'Apache Full'
```

Dieser Befehl öffnet die Ports 80 (HTTP) und 443 (HTTPS).

4.1.7 Schritt 6: Apache2-Konfiguration

Die Standardkonfiguration von Apache befindet sich in der Datei `/etc/apache2/apache2.conf`. Normalerweise musst du diese Datei nicht ändern, es sei denn, du möchtest spezielle Einstellungen vornehmen.

Um eine benutzerdefinierte Website hinzuzufügen, kannst du eine neue Konfigurationsdatei in `/etc/apache2/sites-available/` erstellen. Ein Beispiel für eine einfache Konfigurationsdatei:

```
<VirtualHost *:80>
    ServerAdmin webmaster@domain.com
    DocumentRoot /var/www/html
    ServerName www.example.com
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

4.1.8 Schritt 7: Neustart des Apache-Webservers

Wenn du Änderungen an der Konfiguration vorgenommen hast, musst du den Apache-Dienst neu starten, damit die Änderungen wirksam werden.

```
sudo systemctl restart apache2
```

4.1.9 Schritt 8: Überprüfung der Konfiguration

Du kannst die Apache-Konfiguration auf Fehler überprüfen, bevor du den Server neu startest, indem du den folgenden Befehl ausführst:

```
sudo apachectl configtest
```

Wenn alles in Ordnung ist, solltest du die Meldung `Syntax OK` erhalten.

4.2 Webapp

npm run build

dist ordner mittels scp an Server schicken

unter /var/www/ Speichern

Unter /etc/apach2/ .... neue Conf anlegen

Hier beschreiben was in Conf rein muss.

4.2.1 Voraussetzungen

- Lokale Vue.js-Anwendung, die mit `npm` verwaltet wird.
- Ein Debian-Server mit Apache2.
- SSH-Zugang zu deinem Server.
- `scp` zum Übertragen der Dateien.

4.2.2 Schritt 1: Lokale Vue.js-Anwendung bauen

1. Öffne das Terminal und navigiere in das Verzeichnis deiner Vue.js-Anwendung:

```
cd /pfad/zu/deiner/vue-app
```

2. Installiere die Abhängigkeiten:

```
npm install
```

3. Baue die Anwendung für die Produktion:

```
npm run build
```

   Dieser Befehl erstellt ein optimiertes `dist/`-Verzeichnis, das alle für die Bereitstellung notwendigen Dateien enthält.

4.2.3 Schritt 2: Übertragung der Vue.js-App auf den Debian-Server

1. Verwende `scp`, um das `dist/`-Verzeichnis auf den Server zu kopieren. Ersetze `username` durch deinen Benutzernamen und `192.168.1.100` durch die IP-Adresse deines Servers:

```
scp -r ./dist username@192.168.1.100:/tmp/vuewebapp
```

   Dies überträgt den Inhalt des `dist`-Ordners auf den Server in das temporäre Verzeichnis `/tmp/vuewebapp`.

4.2.4 Schritt 3: Vue.js-Dateien in das Apache-Verzeichnis verschieben

1. Melde dich per SSH auf dem Server an:

```
ssh username@192.168.1.100
```

2. Erstelle das Zielverzeichnis für die Webanwendung:

```
sudo mkdir -p /var/www/vuewebapp
```

3. Verschiebe die Dateien von `/tmp/vuewebapp` nach `/var/www/vuewebapp`:

```
sudo mv /tmp/vuewebapp/* /var/www/vuewebapp/
```

4. Ändere die Berechtigungen, sodass der Apache-Benutzer Zugriff auf die Dateien hat:

```
sudo chown -R www-data:www-data /var/www/vuewebapp
sudo chmod -R 755 /var/www/vuewebapp
```

4.2.5 Schritt 4: Apache2-Konfiguration

1. Erstelle eine neue Konfigurationsdatei für die Vue.js-Webanwendung in `/etc/apache2/sites-ava`

```
sudo nano /etc/apache2/sites-available/vuewebapp.conf
```

2. Füge die folgende Konfiguration ein:

```
<VirtualHost *:80>
    ServerAdmin admin@example.com
    ServerName example.com
    ServerAlias www.example.com

    DocumentRoot /var/www/vuewebapp

    <Directory /var/www/vuewebapp>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/vuewebapp_error.log
    CustomLog ${APACHE_LOG_DIR}/vuewebapp_access.log
       combined
</VirtualHost>
```

   Ersetze `example.com` mit deiner tatsächlichen Domain.

3. Aktiviere die neue Apache2-Site:

```
sudo a2ensite vuewebapp.conf
```

4. (Optional) Deaktiviere die Standardseite, falls sie noch aktiviert ist:

```
sudo a2dissite 000-default.conf
```

5. Überprüfe die Apache-Konfiguration:

```
sudo apachectl configtest
```

   Wenn alles korrekt ist, sollte die Ausgabe `Syntax OK` lauten.

6. Starte den Apache-Dienst neu, um die Änderungen zu übernehmen:

```
sudo systemctl restart apache2
```

4.2.6 Schritt 5: Überprüfung der Bereitstellung

Rufe deine Domain oder die IP-Adresse des Servers in einem Webbrowser auf:

```
http://example.com
```

Du solltest die Startseite deiner Vue.js-Anwendung sehen.

4.3 Node Backend

Auf Server
Als Service von Ubuntu Starten
unter /etc/apache2 ein Reverse Proxy erstellen

4.3.1 Voraussetzungen

- Eine Node.js-Anwendung, die lokal läuft.
- Ein Debian-Server mit Apache2 und `scp`-Zugang.
- Grundlegende Kenntnisse in der Verwaltung von Systemdiensten und der Apache-Konfiguration.

4.3.2 Schritt 1: Node.js-Anwendung lokal bauen

1. Öffne das Terminal und navigiere in das Verzeichnis deiner Node.js-Anwendung:

```
    cd /pfad/zu/deiner/node-app
```

2. Stelle sicher, dass alle Abhängigkeiten installiert sind:

```
    npm install
```

3. Überprüfe, ob die Anwendung lokal ordnungsgemäß läuft:

```
    npm start
```

4.3.3 Schritt 2: Übertragung der Node.js-App auf den Debian-Server

1. Verwende `scp`, um die Node.js-Anwendung auf den Debian-Server zu kopieren. Ersetze `username` durch deinen Benutzernamen und `192.168.1.100` durch die IP-Adresse deines Servers:

```
    scp -r ./node-app username@192.168.1.100:/tmp/backend
```

   Dies überträgt den Inhalt des Anwendungsverzeichnisses auf den Server nach `/tmp/backend`.

2. Melde dich per SSH auf dem Server an:

```
    ssh username@192.168.1.100
```

3. Erstelle das Zielverzeichnis für die Anwendung und verschiebe die Dateien:

```
sudo mkdir -p /var/www/backend
sudo mv /tmp/backend/* /var/www/backend/
```

4. Ändere die Berechtigungen des Verzeichnisses:

```
sudo chown -R www-data:www-data /var/www/backend
sudo chmod -R 755 /var/www/backend
```

4.3.4 Schritt 3: Einrichtung der Node.js-Anwendung als Systemdienst

1. Erstelle eine neue `service`-Datei für die Node.js-Anwendung:

```
sudo nano /etc/systemd/system/node-backend.service
```

2. Füge den folgenden Inhalt ein:

```
[Unit]
Description=Node.js Backend Application
After=network.target

[Service]
ExecStart=/usr/bin/node /var/www/backend/app.js
WorkingDirectory=/var/www/backend
Restart=always
User=www-data
Environment=PORT=8080

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

   Hierbei solltest du den Pfad zu deiner Hauptdatei (`app.js`) anpassen.

3. Lade den Systemdienst neu, um die neue Konfiguration zu laden:

```
sudo systemctl daemon-reload
```

4. Starte die Node.js-Anwendung als Dienst:

```
sudo systemctl start node-backend
```

5. Stelle sicher, dass die Anwendung beim Systemstart automatisch gestartet wird:

```
sudo systemctl enable node-backend
```

6. Überprüfe den Status des Dienstes:

```
sudo systemctl status node-backend
```

   Du solltest eine Ausgabe erhalten, die bestätigt, dass der Dienst aktiv läuft.

4.3.5 Schritt 4: Konfiguration von Apache als Reverse Proxy

1. Stelle sicher, dass die Apache-Module für den Reverse-Proxy aktiviert sind:

```
sudo a2enmod proxy
sudo a2enmod proxy_http
```

2. Erstelle oder bearbeite die Apache-Konfigurationsdatei für die Webseite:

```
sudo nano /etc/apache2/sites-available/vuewebapp.conf
```

3. Füge den folgenden Reverse-Proxy-Abschnitt in die VirtualHost-Konfiguration ein:

```
<VirtualHost *:80>
    ServerAdmin admin@example.com
    ServerName example.com
    ServerAlias www.example.com

    DocumentRoot /var/www/vuewebapp

    <Directory /var/www/vuewebapp>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ProxyPass /api http://localhost:8080/api
    ProxyPassReverse /api http://localhost:8080/api

    ErrorLog ${APACHE_LOG_DIR}/vuewebapp_error.log
    CustomLog ${APACHE_LOG_DIR}/vuewebapp_access.log
       combined
</VirtualHost>
```

   Dieser Abschnitt leitet alle Anfragen, die mit `/api` beginnen, an die Node.js-Anwendung auf Port 8080 weiter.

4. Überprüfe die Apache-Konfiguration:

```
sudo apachectl configtest
```

   Wenn alles korrekt ist, sollte die Ausgabe `Syntax OK` lauten.

5. Starte Apache neu, damit die Änderungen wirksam werden:

```
sudo systemctl restart apache2
```

4.3.6 Schritt 5: Überprüfung der Konfiguration

- Rufe deine Webseite auf, z. B. `http://example.com`.
- Die statischen Vue.js-Inhalte sollten korrekt angezeigt werden.
- Alle Anfragen an `/api` sollten automatisch an die Node.js-Anwendung auf `localhost:8080` weitergeleitet werden.

4.4 MongoDb

4.4.1 Repository hinzufügen

MongoDB ist in den offiziellen Debian-Repositories nicht in der neuesten Version verfügbar, daher fügen wir das offizielle MongoDB-Repository hinzu.

1. Importiere den öffentlichen GPG-Schlüssel für das MongoDB-Repository:

```
wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -
```

2. Füge das MongoDB-Repository zur Paketquelle hinzu:

```
echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/debian buster/mongodb-org/6.0 main" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list
```

3. Aktualisiere das Paket-Repository:

```
sudo apt update
```

4.4.2 MongoDB installieren und als Service konfigurieren

1. Installiere MongoDB:

```
sudo apt install -y mongodb-org
```

2. Starte den MongoDB-Dienst und aktiviere ihn, damit er beim Systemstart automatisch startet:

```
sudo systemctl start mongod
sudo systemctl enable mongod
```

3. Überprüfe, ob der Dienst läuft:

```
sudo systemctl status mongod
```

   Wenn alles richtig funktioniert, sollte der Status als `active (running)` angezeigt werden.

4.4.3 Schritt 2: MongoDB Datenbank und Collections erstellen

4.4.4 Verbindung zur MongoDB herstellen

Verwende das MongoDB-Shell-Tool `mongosh`, um eine Verbindung zu MongoDB herzustellen:

```
mongosh
```

4.4.5 Erstellen der Datenbank und Collections

1. Erstelle eine neue Datenbank (z.B. `myappdb`):

   ```
   use myappdb
   ```

2. Erstelle zwei Collections `user` und `companies`:

   ```
   db.createCollection("user")
   db.createCollection("companies")
   ```

3. Überprüfe, ob die Collections erstellt wurden:

   ```
   show collections
   ```

   Du solltest jetzt die Collections `user` und `companies` sehen.

4.4.6 Schritt 3: Benutzerrolle und Benutzer erstellen

4.4.7 Erstellen eines Benutzers mit Zugriffsrechten

Nun erstellen wir eine Benutzerrolle, die Zugriff auf beide Collections hat, und fügen der Datenbank einen Benutzer hinzu, der diese Rolle verwendet.

1. Erstelle einen neuen Benutzer mit der Rolle, um Zugriff auf die Datenbank zu erhalten:

   ```
   db.createUser({
       user: "appUser",
       pwd: "strongpassword",
       roles: [
           {
               role: "readWrite",
               db: "myappdb"
           }
       ]
   })
   ```

   Dieser Befehl erstellt den Benutzer `appUser` mit dem Passwort `strongpassword` und der Rolle `readWrite`, die Lese- und Schreibzugriff auf die Datenbank `myappdb` ermöglicht.

2. Überprüfe, ob der Benutzer korrekt erstellt wurde:

```
db.getUsers()
```

4.4.8 Verbindung als neuer Benutzer testen

Schließe die aktuelle MongoDB-Sitzung und verbinde dich erneut mit dem neu erstellten Benutzer:

```
mongosh -u appUser -p strongpassword --authenticationDatabase myappdb
```

Wenn die Verbindung erfolgreich ist, hast du den Zugriff auf die Datenbank mit den richtigen Berechtigungen eingerichtet.

4.4.9 Schritt 4: MongoDB-Sicherheitskonfiguration

1. Bearbeite die MongoDB-Konfigurationsdatei, um die Authentifizierung zu erzwingen:

```
sudo nano /etc/mongod.conf
```

2. Füge in der Datei folgende Zeile hinzu, um die Authentifizierung zu aktivieren:

```
security:
  authorization: "enabled"
```

3. Starte den MongoDB-Dienst neu, damit die Änderungen wirksam werden:

```
sudo systemctl restart mongod
```