Dokumentation LB2 - Fileserver

Autor	**Raveendran Shajiran**
Erstell Datum	12. März 2021
Klasse	ST18e
Modul	300 / Plattformübergreifende Dienste im Netzwerk integrieren
Lehrperson	Berger Marco

Inhaltsverzeichnis

Einleitung

In diesem Projekt befassen wir uns mit **Infrastructure as Code (IaC)** an. Auf Basis von **VirtualBox / Vagrant** wollen wir nun einen Fileserver mittels Samba automatisch aufsetzen lassen.

Umsetzung

Tools

Bevor wir dem Code zuwenden, benötigen wir folgende Tools:

- VirtualBox
- Vagrant
- VisualStudioCode
- GitBash

Vagrantfile

Den ganzen Code findet man im Repository. Wir werden hier nun die einzelnen Schritte genauer anschauen.

Vagrant Konfiguration

Die "2" in der ersten Zeile steht für die Version des Konfigurationsobjekts **config**, das zur Konfiguration für diesen Block verwendet wird (der Abschnitt zwischen dem **do** und dem **end**). Dieses Objekt kann von Version zu Version sehr unterschiedlich sein. Derzeit gibt es nur zwei unterstützte Versionen: "1" und "2", wobei die "2" die neuere Version ist. Dies enthält neue und weitere Konfigurationsmöglichkeiten als die Vesion "1".

```
Vagrant.configure("2") do |config|
    ...
    ...
    ...
end
```

VM Konfiguration

Die Einstellungen in **config.vm** ändern die Konfiguration der Maschine, die Vagrant verwaltet. Hier wird konfiguriert, welche Maschine hochfahren sollte. Für unser Projekt verwenden wir eine Ubuntu Maschnine (**ubuntu/trusty64**) und holen dazu die entsprechende Image. Dieses Image ist bereits mit einem vorgegebenen Benutzer ausgestattet *(Username: **vagrant** / Password: **vagrant**)*.

```
    config.vm.box = "ubuntu/trusty64"
```

Für die Virtualisierung bestimmen wir hier, welchen Provider / Virtualisierungsanwendungen wir benutzen wollen. In diesem Projekt richten wir eine einfache VM (Virtuelle Maschine) mit VirtualBox ein. Mittels dem Befehle im GIT Bash z.B. `vagrant up` können wir die Maschine starten und mittels `vagrant destroy` stoppt es die Maschine und löscht alle Daten.

```
    config.vm.provider "virtualbox" do |vb|
```

Man kann nun auch noch die Virtuelle Maschnine anpassen. Man kann z.B. einen Namen für die Maschine bestimmen, Memory Speicher und Anzahl CPUs vergeben, etc. Die GUI funktion ist nicht notwendig. Dient dazu, dass nachdem man ein `vagrant up` gestartet hat, direkt in die Maschine gelangt. Wir setzen diese Funktion auf `true` für spätige Testzwecken, kann aber in unserem Fall auch auf `false` gesetzt sein.

```
        vb.name = "Fileserver (Samba)"
        vb.memory = "2048"
        vb.cpus = 2
        vb.gui = true
```

Netzwerk Konfiguration

Für unser Fileserver benötigen wir Internetzugang, um die Verbindung auch von aussen zu ermöglichen. Hier konfigurieren wir das Netzwerk auf der Maschine. Wir unterscheiden zwischen zwei Netzwerke `private` und `public`. Die Idee dahinter ist, dass private Netzwerke niemals der Öffentlichkeit Zugang zu Ihrem Rechner gewähren sollten, public Netzwerke aber schon. Wir können zugleich auch eine statische IP-Adresse vergeben, mit der wir dann vom lokalen Rechner auf dem Fileserver zugreifen können.

```
config.vm.network "public_network", ip: "192.168.1.200"
```

Wir könnten noch ein **Port-Forwarding** machen, wäre aber nicht nötig für unser Fileserver. Hätten wir z.B. noch einen einfachen **Webserver**, hätten wir unter `http://localhost:8000` diesen Webserver erreichen. Davor müsste man aber noch `apache2` installieren, um dann in die index.html Datei (Startdatei Apache Web Server) zu erlangen. Diese Datei kann man dann beliebig abändern. **SSH** wäre eine weitere Port-Weiterleitung, welche uns eine sichere Möglichkeit bietet, über ein ungesichertes Netzwerk auf eine Maschine zuzugreifen z.B. über Putty. Dabei gibt man die IP des Localhosts `127.0.0.1` an und den weitergeleiteten Port `2200` an, welche dann auf den Port `22` umgeleitet wird.

```
config.vm.network :forwarded_port, guest: 80, host: 8000
config.vm.network :forwarded_port, guest: 22, host: 2200, id: "ssh"
```

Provisionierung

Mit der Provisionierung kann man in Vagrant als Teil des Vagrant-Up-Prozesses automatisch Software installieren, Konfigurationen ändern, usw. Die Befehle die man sonst auf dem Bash in der Maschine eintippt, können nun unter dem Abschnitt `SCRIPT` eingefügt werden. Diese werden in der Variable `SCRIPT_INSTALL` gespeichert. In der unteren Zeile erkennt dann die Provisionierung, dass es sich um eine `Shell / Bash` Provision handelt. Wir sagen also der Shell, dass folgende Befehle von `SCRIPT_INSTALL` ausgeführt werden soll, bzw. automatisch eingetippt werden soll. Man könnte auch die Bash Befehle in einem getrennten Datei speichern, sodass man anstatt die Variable im Code `inline:`, einen relativen Pfad mittels `path:` die Datei im Vagrantfile verweist. So hätte man z.B. die Provisionierung und die Netzwerk Konfiguration in seperaten Dateien, um eine besseren Übersicht zu erhalten. Dies benötigt man jedoch bei grösseren Vagrantfiles. Da unser Vagrantfile jedoch klein ist, ist es in unserem Fall in Ordnung, wenn alles im Vagrantfile befindet.

```
SCRIPT_INSTALL = <<-SCRIPT
    ...
    ...
    ...
SCRIPT

config.vm.provision :shell, inline: SCRIPT_INSTALL
```

Samba Installation

Bevor wir **Samba** installieren, nutzen wir die Befehle `sudo apt-get update` um die Paketlisten neu einzulesen und zu aktualisieren. Der Befehl `sudo apt-get upgrade` führt das eigentliche „Update“ aus. Somit sind wir auf dem aktuellsten Stand und können dann unser Samba installieren, welche uns für die Freigabe und das Teilen von Ordner und Dateien ermöglicht.

```
        sudo apt-get -y update
        sudo apt-get -y upgrade
        sudo apt-get -y install samba
```

Samba Konfiguration

Sobald man Samba erfolgreich installiert hat, konfigurieren wir nun die `config` Datei. Zuerst erstellen wir eine **Backup Datei** vom originalen bzw. eine Kopie, falls wenn wir etwas falsches konfiguriert haben, wieder auf diese zurück zugreifen können.

```
        sudo cp /etc/samba/smb.conf /etc/samba/smb.conf-backup
```

Nun mittels Vagrant ist es schwierig die `config` Datei zu ändern. Was wir also machen werden ist, wir löschen die `config` Datei in der Maschine und fügen eine vorkonfigurierte `config` Datei ein. Diese vorkonfigurierte Datei befindet sich in unserem Repository. Mit dem Befehl `wget` können wir nun die Datei via dem Internet herunterladen. Die Option `-P` erlaubt die Angabe eines Verzeichnisses, wo die heruntergeladene Datei gespeichert werden soll.

```
        sudo rm /etc/samba/smb.conf
        sudo wget -P /etc/samba/
https://raw.githubusercontent.com/shajiran/m300_lb/main/lb2/smb.conf
```

User und Share einrichten

Nachdem Samba erfolgreich installiert und konfiguriert wurde, erstellen wir nun einen Ordner, welche die User später dann Dateien untereinander teilen können. Wir benennen den Ordner `share`.

```
        sudo mkdir /home/share
```

Jetzt haben wir einen Ordner, fehlt nun noch ein User. Wir erstellen einen User mit Passwort. Dieser dient zur **Sicherheit**, sodass kein anderer User auf dem Share gelangt. Mittels Variablen erleichtert es uns die Aufgabe für zukünftig weitere Änderungen am Code selber. Wir setzen hier also ein `Username: test` und ein `Passwort: mypassword` in jeweils eine Variable.

```
        LOGIN=test
        PASS=mypassword
```

Wir erstellen hier jetzt einen User und wie wir wissen, müssen wir danach das Passwort zwei mal tippen. Mittels dem Befehl `echo` können wir dem Vagrant-Prozess unser Passwort mitteilen, sodass dies automatisch einrichtet. Im nachhinein fügen wir den erstellten User in eine exestierende Gruppe.

```
        echo -ne "$PASS\n$PASS\n" | sudo adduser $LOGIN
        sudo addgroup $LOGIN $LOGIN
```

Der folgende Befehl legt ein smb-Passwort für den bestehenden User "test" an. Dass dieser Schritt für einen bestehenden User durchgeführt wird ist insofern wichtig, da jeder smb-User einen gültigen Account am Server benötigt. Mit dem vergebenen Passwort kann später auf das smb-Share zugegriffen werden. Die Option `-a` gibt an, dass der folgende Benutzername der lokalen Datei smbpasswd hinzugefügt werden soll, wobei das neue Passwort eingegeben wird. Da das smb-Passwort nicht mit dem Passwort des eigentlichen Accounts übereinstimmen muss, kann es durchaus aufwendig sein, die unterschiedlichen Passwörter zu verwalten. Die Option `-s` soll Benutzern beim Schreiben von Skripten helfen, smbpasswd zu betreiben. Wir geben hier nun dasselbe Passwort wie vorhin.

```
        echo -ne "$PASS\n$PASS\n" | sudo smbpasswd -a -s $LOGIN
```

Nun möchten wir einen weiteren Aspekt der **Sicherheit** beachten: `chown` steht für change owner und erlaubt das Ändern des Eigentümer-Benutzers und/oder der Eigentümer-Gruppe von Dateien. Normalerweise ist der Benutzer, der eine Datei oder ein Verzeichnis erstellt, auch der Besitzer. Mit chown kann der Besitzer jedoch nachträglich geändert werden. `chmod` verändert die Zugriffsrechte von Dateien, bzw. kann man hier die Schreibe- und Lesebrechte vergeben. Die Ziffer-Ordnung `2770` haben eine Bedeutung:

- **1. Ziffer**: Besitzer der Datei
- **2. Ziffer**: Gruppe der Datei
- **3. Ziffer**: andere Benutzer
- **4. Ziffer**: Besitzer, Gruppe und Andere

Die einzelnen Ziffer haben ebenfalls eine Bedeutung:

- **0**: Keine Rechte
- **1**: Nur Ausführrechte
- **2**: Nur Schreibrechte
- **3**: Nur Ausführ- und Schreibrechte
- **4**: Nur Leserechte
- **5**: Nur Schreib- und Ausführrechte
- **6**: Nur Schreib- und Leserechte
- **7**: Alle Rechte

Unserem Share geben wir also nun die Berechtigung, dass der Besitzer sowie die Gruppe alle Rechte haben, die anderen jedoch keine Rechte haben. So kann man noch mit der **Sicherheit** "herum spielen".

```
sudo chown $LOGIN:$LOGIN /home/$LOGIN
sudo chmod 2770 /home/$LOGIN
```

Fileserver aktivieren

Sobald alle Konfigurationen vorgenommen sind, können wir unser Fileserver aktivieren.

```
sudo /etc/init.d/samba restart
```

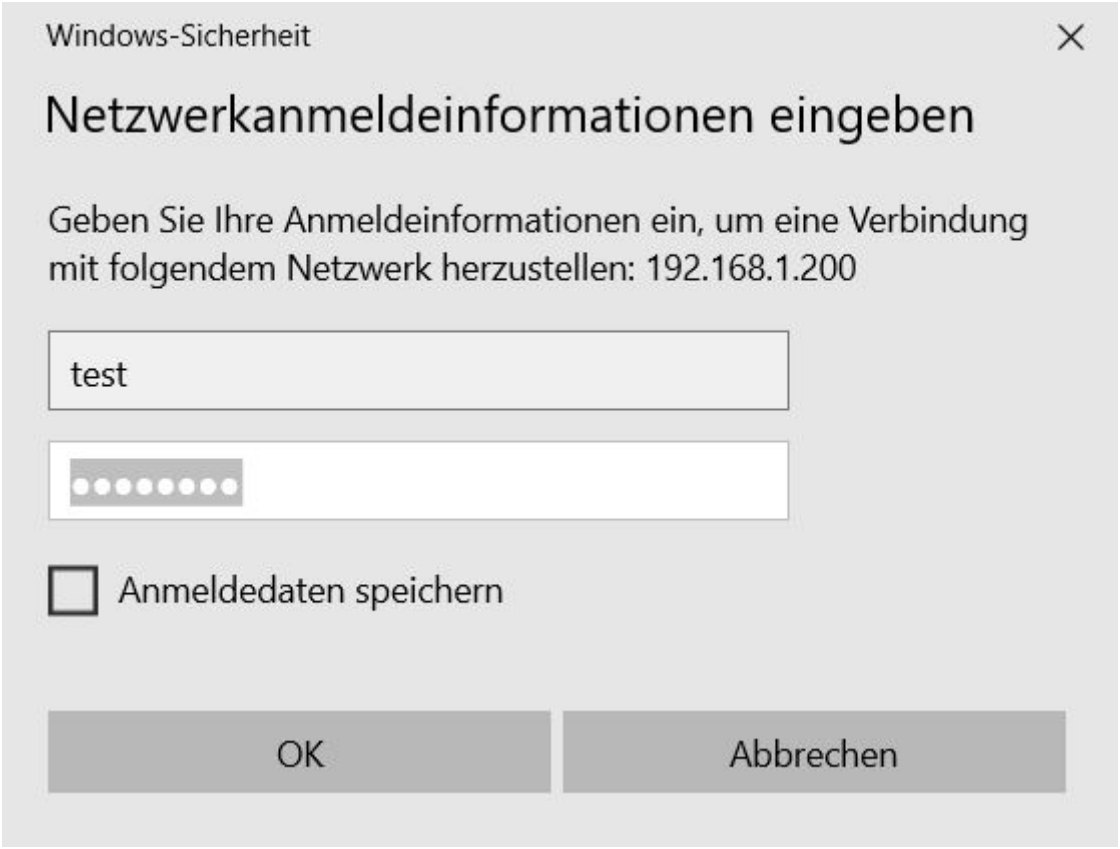
Config Datei

Die **smb.conf** Datei ist das Grundstück des ganzen. Hier bestimmen wir, welchen Ordner geshared werden soll, können weitere Berechtigungen festlegen, welche User oder Gruppe auf diesen Ordner Zugriff haben dürfen, etc. In dieser Config Datei können natürlich weitere Ordner zum sharen ergänzt werden.

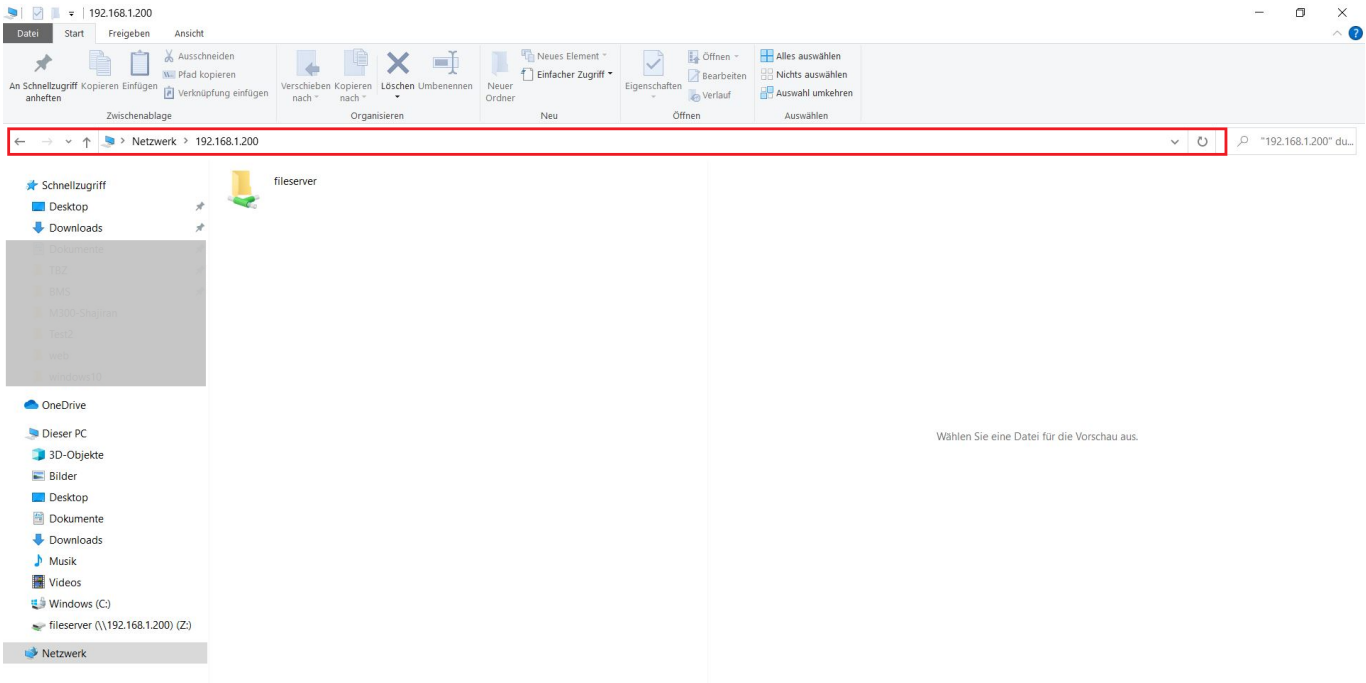
```
[fileserver]
    path = /home/share
    browseable = yes
    writeable = yes
    read only = no
    create mode = 0600
    directory mode = 0700
    valid users = @test
```

Testen

Nachdem das Vagrantfile fertig eingerichtet ist, können wir den Fileserver starten. Wie schon vorhin erklärt, können wir nun mittels dem Befehl `vagrant up` im GIT Bash eingeben, um die Maschine zu starten. Nachdem die Maschine, bzw. unser Fileserver läuft, können wir auf unser Share Ordner zugreifen. Dazu geben wir im **Explorer Pfad** die IP-Adresse unsers Fileservers an `\\192.168.1.200`. Es wird nach ein Username und ein Passwort gefordert *(Username: **test** / Password: **mypassword**)*.



Wenn die Login-Daten stimmen gelangen wir in unser Share Ordner. Somit haben wir es geschafft einen Fileserver automatisch aufzusetzen und mittels Passwörter und Berechtigungen in ein Share Ordner vom lokalen Rechner zu gelangen.



Quellenverzeichnis

- VagrantBox Katalog
- Provisioning
- Samba Installation und Konfiguration
- Samba Konfigurations File anpassen

- Samba Config File Download
- Samba Konfigurations File verschieben
- Samba Freigabe mit Authentifizierung
- Berechtigung Erklärung
- Berechtigung Optionen
- Berechtigungen definieren / setzen