

Erstellung einer eigenen CA (Certificate Authority) und manuelle Verteilung der Zertifikate

Worum geht es?

Wofür benötigen wir Zertifikate in der IT?

In der IT werden Zertifikate verwendet, um die Identität von Geräten, Diensten oder Personen eindeutig und vertrauenswürdig nachzuweisen. Sie ermöglichen außerdem die verschlüsselte und sichere Kommunikation über Netzwerke, z. B. per HTTPS oder VPN. Auch in einem geschlossenen Netzwerk sollte man eigene Zertifikate verwenden. Da öffentliche Zertifizierungsstellen (CAs) keinen Zugriff auf interne Systeme haben benötigen wir eine eigene CA.

Eigene Zertifikate ermöglichen die verschlüsselte und authentifizierte Kommunikation über Protokolle wie HTTPS, SMTPS, LDAPS, FTPS, sowie bei Datenbanken wie MariaDB, PostgreSQL und Message Brokern wie MQTT oder Kafka. Dadurch behält man die volle Kontrolle über Sicherheit, Gültigkeit und Verteilung der Zertifikate innerhalb des Netzwerks. **Lokal im Anwendungscontainer erstellte Zertifikate vs. eigener CA (Certificate Authority)**. Es ist möglich in jedem Anwendungscontainer (LXC) eigene Zertifikate zu erstellen - Wir haben uns aber dagegen entschieden. Hierfür bauen wir eine eigene CA auf, um die Komplexität des Verteilens der einzelnen Zertifikate zu minimieren. Wir haben durch eine eigene CA einheitliche zentral verwaltete Zertifikate.

Diese Anleitung zeigt dir, wie du mit **OpenSSL** auf deinem Server **ssl-ca140** (IP: 192.168.137.140) eine eigene **Zertifizierungsstelle (CA)** und ein **einzelnes Zertifikat für alle Dienste** erstellst. Anschließend verteilt man die Zertifikate manuell an die verschiedenen Dienste und Client-Rechner (Windows, Mac, Linux).

Warum ein einzelnes Zertifikat — und warum mehrere?

- **Ein einziges Zertifikat (SAN-Zertifikat)** ist praktisch:
Man braucht nur eine Datei, die für alle deine Dienste gültig ist.
Das spart Aufwand und ist für kleine oder Test-Umgebungen oft ausreichend.
 - **Mehrere Zertifikate (empfohlen für Produktion):**
Jeder Dienst bekommt ein eigenes Zertifikat, was die Sicherheit verbessert, da bei Kompromittierung nur dieser Dienst betroffen ist.
Auch die Verwaltung (Erneuerung, Widerruf) wird so flexibler, ist aber deutlich aufwändiger oder muss automatisiert werden.
-

Voraussetzungen

- Ein LXC Container **ssl-ca140** mit OpenSSL installiert
- Grundlegende Linux-Kommandos sind dir vertraut
- SSH-Zugang zum Server

```
sudo apt install openssl
```

1. Ordnerstruktur anlegen

```
sudo mkdir -p ~/myCA/{certs,crl,newcerts,private}
sudo chmod 700 ~/myCA/private
sudo touch ~/myCA/index.txt
echo 1000 | sudo tee ~/myCA/serial
```

```
pdal@CA-ssl140:~$ sudo mkdir -p ~/myCA/{certs,crl,newcerts,private}
[sudo] password for pdal:
pdal@CA-ssl140:~$ ls
myCA
```

```
pdal@CA-ssl140:~$ sudo chmod 700 ~/myCA/private
pdal@CA-ssl140:~$ █
```

```
pdal@CA-ssl140:~$ sudo touch ~/myCA/index.txt
pdal@CA-ssl140:~$ █
```

```
pdal@CA-ssl140:~/myCA$ echo 1000 | sudo tee ~/myCA/serial
1000
pdal@CA-ssl140:~/myCA$ ls
certs  crl  index.txt  newcerts  private  serial
```

2. CA (Zertifizierungsstelle) erstellen

2.1 OpenSSL-Konfigurationsdatei erstellen openssl.cnf

Erstelle im Ordner ~/myCA die Datei openssl.cnf mit folgendem Inhalt (vereinfachtes Beispiel):

```
sudo nano ~/myca/openssl.cnf
```

```
[ ca ]
default_ca = CA_default

[ CA_default ]
dir          = /home/pdal/myCA
certs        = $dir/certs
crl_dir      = $dir/crl
new_certs_dir = $dir/newcerts
database     = $dir/index.txt
serial       = $dir/serial
RANDFILE     = $dir/private/.rand
```

```
private_key      = $dir/private/ca.key.pem
certificate     = $dir/certs/ca.cert.pem

default_days    = 3650
default_crl_days = 30
default_md      = sha256

policy          = policy_loose

[ policy_loose ]
countryName      = optional
stateOrProvinceName = optional
localityName     = optional
organizationName = optional
organizationalUnitName = optional
commonName       = supplied
emailAddress     = optional

[ req ]
default_bits    = 4096
distinguished_name = req_distinguished_name
string_mask      = utf8only

[ req_distinguished_name ]
countryName      = Country Name (2 letter code)
stateOrProvinceName = State or Province Name
localityName     = Locality Name
organizationName = Organization Name
organizationalUnitName = Organizational Unit Name
commonName       = Common Name
emailAddress     = Email Address

[ v3_ca ]
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid(always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3_server ]
basicConstraints = CA:FALSE
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = apache.local
IP.1 = 192.168.137.101
DNS.2 = mariadb.local
IP.2 = 192.168.137.120
DNS.3 = pgadmin4.local
IP.3 = 192.168.137.130
DNS.4 = CA-ssl.local
IP.4 = 192.168.137.140
DNS.5 = kafka.local
```

```
IP.5 = 192.168.137.150
DNS.6 = postgresql.local
IP.6 = 192.168.137.160
DNS.7 = mqtt.local
IP.7 =192.168.137.170
```

```
[ ca ]  
default_ca = CA_default  
  
[ CA_default ]  
dir = /home/pdal/myCA  
certs = $dir/certs  
crl_dir = $dir/crl  
new_certs_dir = $dir/newcerts  
database = $dir/index.txt  
serial = $dir/serial  
RANDFILE = $dir/private/.rand  
  
private_key = $dir/private/ca.key.pem  
certificate = $dir/certs/ca.cert.pem  
  
default_days = 3650  
default_crl_days = 30  
default_md = sha256  
  
policy = policy_loose
```

```
[ policy_loose ]  
countryName = optional  
stateOrProvinceName = optional  
localityName = optional  
organizationName = optional  
organizationalUnitName = optional  
commonName = supplied  
emailAddress = optional
```

```
[ req ]
```

```
[ req ]  
default_bits = 4096  
distinguished_name = req_distinguished_name  
string_mask = utf8only  
  
[ req_distinguished_name ]  
countryName = Country Name (2 letter code)  
stateOrProvinceName = State or Province Name  
localityName = Locality Name  
organizationName = Organization Name
```

```

v.organizationName           = Organization Name
organizationUnitName        = Organizational Unit Name
commonName                  = Common Name
emailAddress                = Email Address

[ v3_ca ]
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3_server ]
basicConstraints = CA:FALSE
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

```

```

[ alt_names ]
DNS.1 = apache.local
DNS.2 = mariadb.local
DNS.3 = postgresql.local
DNS.4 = mosquitto.local
DNS.5 = kafka.local
IP.1 = 192.168.137.140
IP.2 = 127.0.0.1

```

Hinweis: Passe Pfade und Namen bei Bedarf an. Ersetze `/home/username` mit deinem tatsächlichen Pfad. Erklärung zur Konfigurationsdatei:

[ca]

```
default_ca = CA_default
```

Legt fest, welche CA-Konfiguration verwendet wird.

CA_default verweist auf die folgende Sektion [CA_default].

[CA_default]

```
dir      = /home/pdal/myCA
```

'dir' ist das Stammverzeichnis deiner CA-Dateien (Passe 'username' an deinen Benutzernamen an).

Alle folgenden Pfade bauen darauf auf.

```

certs          = $dir/certs
crl_dir       = $dir/crl
new_certs_dir = $dir/newcerts
database      = $dir/index.txt
serial         = $dir/serial
RANDFILE       = $dir/private/.rand

```

certs: Speichert ausgestellte Zertifikate

crl_dir: Für gesperrte Zertifikate (Certificate Revocation Lists)

new_certs_dir: Interner Ordner für neue Zertifikate

database: Datei, in der OpenSSL die ausgestellten Zertifikate dokumentiert

serial: Datei mit der aktuellen Seriennummer

RANDFILE: Hilfsdatei für zufällige Daten (z.B. bei Schlüsselerzeugung)

```

private_key     = $dir/private/ca.key.pem
certificate    = $dir/certs/ca.cert.pem

```

Speicherorte des CA-Schlüssels und CA-Zertifikats

```

default_days   = 3650
default_crl_days = 30
default_md     = sha256

```

default_days: Gültigkeitsdauer eines neuen Zertifikats (hier: 10 Jahre)

default_crl_days: Wie lange eine Sperrliste gültig ist

default_md: Hash-Algorithmus (hier: sha256, sicherer als z.B. md5)

```

policy        = policy_loose

```

Legt fest, wie streng OpenSSL die eingegebenen Felder prüft (z.B. ob Land oder Organisation zwingend notwendig ist)

[**policy_loose**]

```

countryName          = optional
stateOrProvinceName = optional
localityName         = optional
organizationName     = optional
organizationalUnitName = optional
commonName           = supplied
emailAddress         = optional

```

Das ist eine lockere Richtlinie: Die meisten Felder sind optional, nur 'commonName' (der Servername wie 'apache.local') muss angegeben werden.

[req]

```

default_bits      = 4096
distinguished_name = req_distinguished_name
string_mask       = utf8only

```

Konfiguration für neue Certificate Signing Requests (CSRs).

default_bits: Schlüssellänge in Bit (je mehr, desto sicherer – 4096 ist stark).

distinguished_name: Welche Felder beim CSR abgefragt werden.

string_mask = utf8only: Nur UTF-8-Zeichen erlaubt.

[req_distinguished_name]

```

countryName          = Country Name (2 letter code)
stateOrProvinceName = State or Province Name
...

```

Hier wird definiert, welche Felder bei der Erstellung eines Zertifikats abgefragt werden (z. B. Land, Organisation, Common Name).

[v3_ca]

```

subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid(always,issuer)
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

```

Erweiterungen für CA-Zertifikate

basicConstraints = CA:true: Macht klar, dass dieses Zertifikat Zertifikate ausstellen darf.

keyUsage: Welche Aktionen mit dem Schlüssel erlaubt sind.

[v3_server]

```
basicConstraints = CA:FALSE
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names
```

Erweiterungen für Server-Zertifikate:

- **CA:FALSE:** Dieses Zertifikat ist kein CA-Zertifikat,
- **keyUsage:** Für digitale Signatur und Verschlüsselung,
- **extendedKeyUsage:** Darf als TLS-Serverzertifikat verwendet werden.
- **subjectAltName:** Verweist auf die untenstehende Liste mit alternativen Namen.

[alt_names]

```
DNS.1 = apache.local
IP.1 = 192.168.137.101
DNS.2 = mariadb.local
IP.2 = 192.168.137.120
DNS.3 = pgadmin4.local
IP.3 = 192.168.137.130
DNS.4 = CA-ssl.local
IP.4 = 192.168.137.140
DNS.5 = kafka.local
IP.5 = 192.168.137.150
DNS.6 = postgresql.local
IP.6 = 192.168.137.160
DNS.7 = mqtt.local
IP.7 = 192.168.137.170
```

Dies ist die SAN-Liste (Subject Alternative Name).

Sie erlaubt dem Zertifikat, mehrere Namen und IP-Adressen abzusichern, z.B. **apache.local** für Apache, **mariadb.local** für MariaDB usw.

Dadurch braucht man nur ein einziges Zertifikat für alle Dienste – das ist das sogenannte pdal-Zertifikat.

Fazit

Diese Konfiguration definiert eine eigene CA mit:

lockeren Anforderungen an Eingaben,

starkem Schlüssel (4096 Bit),
SAN-Unterstützung für viele Dienste,
1 Zertifikat für alle Server (pdal-Zertifikat).

2.2 CA-Schlüssel und CA-Zertifikat erstellen

```
sudo openssl genrsa -aes256 -out private/ca.key.pem 4096
```

```
pdal@CA-ssl140:~/myCA$ sudo openssl genrsa -aes256 -out private/ca.key.pem 4096  
[sudo] password for pdal:  
Enter PEM pass phrase:[]
```

zur

Bestätigung gibt man ein Passwort ein. Wir wählen in unserem Beispiel **JadeHS20**, nach der Eingabe muss man das Passwort noch einmal bestätigen.

```
pdal@CA-ssl140:~/myCA$ sudo openssl genrsa -aes256 -out private/ca.key.pem 4096  
[sudo] password for pdal:  
Enter PEM pass phrase:  
Verifying - Enter PEM pass phrase:  
pdal@CA-ssl140:~/myCA$ []
```

```
sudo chmod 400 private/ca.key.pem
```

```
pdal@CA-ssl140:~/myCA$ sudo chmod 400 private/ca.key.pem  
pdal@CA-ssl140:~/myCA$ []
```

```
sudo openssl req -config openssl.cnf \  
-key private/ca.key.pem \  
-new -x509 -days 3650 -sha256 -extensions v3_ca \  
-out certs/ca.cert.pem
```

Dabei wird man nach Angaben wie Land, Organisation etc. gefragt. Diese kann man passend ausfüllen.

```
dal@CA-ssl140:~/myCA$ sudo openssl req -config openssl.cnf \
  -key private/ca.key.pem \
  -new -x509 -days 3650 -sha256 -extensions v3_ca \
  -out certs/ca.cert.pem
Enter pass phrase for private/ca.key.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
if you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []:DE
State or Province Name []:Niedersachsen
Locality Name []:Wilhelmshaven
Organization Name []:Jade-Hochschule
Organizational Unit Name []:FB-MIT
Common Name []:pdal
Email Address []:
dal@CA-ssl140:~/myCA$
```

3. Server-Zertifikat mit SAN (für alle Dienste) erstellen

3.1 Privaten Schlüssel für Server erstellen

```
sudo openssl genrsa -out private/server.key.pem 2048
sudo chmod 400 private/server.key.pem
```

```
pdal@CA-ssl140:~/myCA$ sudo openssl genrsa -out private/server.key.pem 2048
pdal@CA-ssl140:~/myCA$ sudo chmod 400 private/server.key.pem
```

3.2 Zertifikatsantrag (CSR) mit SAN erstellen

Erstelle eine Datei `server.ext` mit folgendem Inhalt (SANs für alle Dienste):

```
sudo nano server.ext
```

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = apache.local
IP.1 = 192.168.137.101
DNS.2 = mariadb.local
IP.2 = 192.168.137.120
DNS.3 = pgadmin4.local
```

```

IP.3 = 192.168.137.130
DNS.4 = CA-ssl.local
IP.4 = 192.168.137.140
DNS.5 = kafka.local
IP.5 = 192.168.137.150
DNS.6 = postgresql.local
IP.6 = 192.168.137.160
DNS.7 = mqtt.local
IP.7 = 192.168.137.170

```

Wenn die Liste der einbezogenen Server identisch mit der SAN-Liste ist, kann die nur Kurzschreibweise `subjectAltName = @alt_names` verwendet werden.

```

[authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = apache.local
IP.1 = 192.168.137.101
DNS.2 = mariadb.local
IP.2 = 192.169.137.120
DNS.3 = pgadmin4.local
IP.3 = 192.168.137.130
DNS.4 = CA-ssl.local
IP.4 = 192.168.137.140
DNS.5 = kafka.local
IP.5 = 192.168.137.150
DNS.6 = postgresql.local
IP.6 = 192.168.137.160
DNS.7 = mqtt.local
IP.7 = 192.168.137.170

```

Erstelle den CSR(Certificate Signing Request): Was ist ein CSR?

Ein CSR ist eine Datei, die:

- einen öffentlichen Schlüssel enthält,
- zusammen mit Angaben zur Identität (z.B. Common Name, Organisation, Ort, SANs) erstellt wurde,
- von einer Zertifizierungsstelle (CA) signiert werden soll, um daraus ein gültiges Zertifikat zu erzeugen.

```

sudo openssl req \
-new \
-key private/server.key.pem \
-out server.csr.pem \
-config /home/pdal/myCA/openssl.cnf \
-subj "/C=DE/ST=Niedersachsen/L=Wilhelmshaven/O=Jade-Hochschule/OU=FB-
MIT/CN=all-services.local"

```

```
pdal@CA-ss1140:~/myCA$ sudo openssl req -new -key private/server.key.pem -out server.csr.pem -config openssl.cnf -subj "/C=DE/ST=Niedersachsen/L=Wilhelmshaven/O=Jade-Hochschule/OU=FB-MIT/CN=all-services.local"
pdal@CA-ss1140:~/myCA$
```

Im

Anschluss kann man noch überprüfen ob der key auch wirklich erstellt wurde - mit folgendem Befehl:

```
ls -l server.csr.pem
openssl req -in server.csr.pem -noout -text
```

```
pdal@CA-ss1140:~/myCA$ openssl req -in server.csr.pem -noout -text
Certificate Request:
Data:
Version: 1 (0x0)
Subject: C = DE, ST = Niedersachsen, L = Wilhelmshaven, O = Jade-Hochschule,
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
            Modulus:
                00:bd:fd:8d:2f:9d:cd:e1:6c:7e:45:cf:88:74:d5:
                70:2d:4f:e3:b1:0d:57:dd:b6:c0:49:f3:91:8e:87:
                ea:de:09:35:60:69:1b:a4:9f:98:95:5c:30:70:1e:
                1a:61:1a:00:ec:97:1c:39:bc:65:da:21:65:ba:80:
                ed:bd:27:ee:8e:f9:b5:8a:31:c8:89:11:6a:f7:6a:
                44:e5:7c:c2:e3:33:14:46:91:2d:06:cd:d0:87:43:
                70:ef:b1:1c:28:03:3e:16:96:a8:2e:0d:84:8d:32:
                aa:d7:1e:23:2d:e0:5f:79:a1:c3:fa:4a:12:6b:e6:
                a5:75:65:7c:64:26:7f:3a:bc:81:5e:e1:03:e5:db:
                7a:df:74:46:42:64:23:39:ff:80:1b:76:96:74:5d:
                ff:0f:fb:60:de:87:2f:82:23:80:4c:0e:c5:4a:db:
                97:2d:e8:f0:7f:be:8b:4a:02:e3:67:72:3a:95:95:
                be:c9:fc:ea:3d:7f:0b:1d:fd:05:e4:bf:11:0a:dc:
                da:60:1d:50:ae:38:fc:c2:2e:1d:d6:59:ee:de:d6:
                12:32:56:ef:3c:af:5c:6f:84:e3:32:e4:9f:cb:ce:
                68:cf:5d:a9:0b:88:29:d8:45:5d:f7:70:af:66:3f:
                e9:f4:5b:39:f5:59:62:50:5f:63:03:a6:c7:03:82:
                35:87
            Exponent: 65537 (0x10001)

Attributes:
    (none)
Requested Extensions:
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
50:26:c4:88:4c:f8:3c:26:f7:93:66:b7:f8:34:fb:ce:f8:5c:
4e:87:07:a9:04:ef:30:cf:b7:2f:77:1c:c2:78:e6:b5:91:12:
82:f1:bc:83:7b:51:bf:6e:eb:0d:d1:95:e0:93:43:7f:47:c8:
57:36:a1:a2:c1:27:37:1f:70:14:3f:66:77:af:9f:a6:0c:73:
94:cc:fa:09:56:af:90:26:3d:48:2c:f1:04:3a:ca:e5:0c:6e:
94:a3:90:27:e6:c1:e5:15:65:60:c2:17:0d:07:54:b3:a6:15:
a9:56:07:79:fe:eb:1f:08:83:fb:aa:eb:e7:40:85:b2:8e:8a:
d3:7b:a0:13:39:38:2d:4f:7f:de:10:bb:b6:3a:22:28:0a:0c:
ee:d0:c1:aa:6b:d1:97:69:40:64:da:f4:d7:c7:dc:e3:af:47:
e2:74:ac:5c:11:0e:af:a1:a9:dc:6e:a0:01:2d:1b:98:b0:05:
c5:93:3e:45:f8:02:d8:ab:ce:b2:18:aa:f0:34:07:3f:4d:33:
c7:07:14:99:30:9c:47:b0:f6:c2:66:fe:67:3f:01:f1:9b:26:
4d:7b:5f:12:c6:70:92:a9:18:5b:77:2d:e4:82:e6:0f:07:67:
b1:1e:e0:47:32:80:61:4f:f8:4c:c7:7a:83:3f:f2:03:cf:03:
65:29:a1:05
```

3.3 Zertifikat mit SAN signieren

```
sudo openssl ca -config openssl.cnf -extensions v3_server -days 825 -notext -md sha256 -in server.csr.pem -out certs/server.cert.pem
```

Man wird gefragt, ob man das Zertifikat signieren möchtest – mit y bestätigen. Anschließend muss man noch einmal mit y bestätigen, damit das Zertifikat erstellt und in die Datenbank geschrieben wird.

```
pdal@CA-ssl140:~/myCA$ sudo openssl ca -config openssl.cnf -extensions v3_server
m
Using configuration from openssl.cnf
Enter pass phrase for /home/pdal/myCA/private/ca.key.pem:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName          :PRINTABLE:'DE'
stateOrProvinceName  :ASN.1 12:'Niedersachsen'
localityName         :ASN.1 12:'Wilhelmshaven'
organizationName     :ASN.1 12:'Jade-Hochschule'
organizationalUnitName:ASN.1 12:'FB-MIT'
commonName           :ASN.1 12:'all-services.local'
Certificate is to be certified until Oct  5 12:23:07 2027 GMT (825 days)
Sign the certificate? [y/n]:
```

Anschliessend muss man noch die Berechtigungen auf das Zertifikat setzen.

```
sudo chmod 444 certs/server.cert.pem
```

```
pdal@CA-ssl140:~/myCA$ sudo chmod 444 /home/pdal/myCA/certs/server.cert.pem
[sudo] password for pdal:
pdal@CA-ssl140:~/myCA$
```

4. Zertifikate an Dienste verteilen und konfigurieren

Das Zertifikat `server.cert.pem`, den Schlüssel `server.key.pem` und das CA-Zertifikat `ca.cert.pem` braucht man auf allen Servern/Diensten.

Ziel

Wir möchten:

- Die Dateien `ca.cert.pem`, `server.cert.pem` und `server.key.pem` im CA-Container nach `/home/pdal/download` verschieben.
- Diese Dateien mit WinSCP per SFTP auf deinen lokalen Rechner in `C:\tmp` herunterladen.
- Die Dateien mit WinSCP per SFTP auf den Apache2-Container hochladen nach `/home/pdal/download`.
- Die Dateien von dort aus in die entsprechenden Verzeichnisse im Apache2-Container verschieben.

Voraussetzungen

- LXC-Container für CA (`ca-container`) und Apache2 (`apache110`) sind aktiv.
- Benutzer `pdal` ist in beiden Containern vorhanden.
- SFTP-Zugriff über IP oder Hostname ist möglich.
- SFTP Client wie z.B. WinSCP ist auf deinem Windows-Rechner installiert(funktioniert aber auch ohne separatem Client über die Commando Zeile).

- ◊ 1. Dateien im CA-Container vorbereiten

Download Verzeichnis erstellen

Wir erstellen zunächst im CA Container ein Verzeichnis mit der Bezeichnung download im home Verzeichnis von dem User pdal.

```
sudo mkdir -p /home/pdal/download
```

```
pdal@CA-ssl140:~/myCA$ sudo mkdir -p /home/pdal/download
```

Pfade anpassen, wenn nötig

Jetzt kopieren wir die einzelnen Zertifikate und den Server Schlüssel in das Download-Verzeichnis. Da das Verzeichnis `/home/pdal/myca/private` root gehört und nur root auf dieses Verzeichnis zugreifen darf, müssen wir uns später im container als `root` anmelden. Dies machen wir wie folgt. Wir sind derzeit als pdal angemeldet und müssen um Zugriff auf das private Verzeichnis zu erlangen, `sudo -i` eingeben um zum Benutzer `root` zu wechseln.

```
sudo cp /home/pdal/myca/certs/ca.cert.pem /home/pdal/download/
sudo cp /home/pdal/myca/certs/server.cert.pem /home/pdal/download/
sudo -i
cp /home/pdal/myca/private/server.key.pem /home/pdal/download/
```

Dies folgt der Syntax `sudo cp dieZuKopierendeDateiMitPfad dasZielVerzeichnis`.

```
pdal@CA-ssl140:~$ sudo cp /home/pdal/myCA/certs/ca.cert.pem /home/pdal/download/
pdal@CA-ssl140:~$ sudo cp /home/pdal/myCA/certs/server.cert.pem /home/pdal/download/
pdal@CA-ssl140:~$ sudo -i
root@CA-ssl140:~# cp /home/pdal/myCA/private/server.key.pem /home/pdal/download/
root@CA-ssl140:~#
```

Rechte anpassen (optional, damit der Nutzer pdal Zugriff hat)

Wir befinden uns nach wie vor als root in dem Container. Falls nicht bitte auf die root Ebene mit `sudo -i` wechseln.

Die eben kopierten Dateien gehören alle dem User Root und haben sehr strikte Berechtigungen. Um diese Dateien über einen SFTP Client herunter laden zu können, müssen wir die Berechtigungen und auch den Owner der Dateien ändern. Das machen wir mit dem Befehl `chown` und mit dem Befehl `chmod`.

```
root@CA-ssl140:~# ls -l /home/pdal/download/
total 12
-rw-r--r-- 1 root root 2098 Jul  7 09:10 ca.cert.pem
-r--r--r-- 1 root root 1984 Jul  7 09:10 server.cert.pem
-r----- 1 root root 1704 Jul  7 09:11 server.key.pem
root@CA-ssl140:~#
```

```
chown pdal:pdal /home/pdal/download/*.pem  
chmod 600 /home/pdal/download/*.pem
```

```
root@CA-ssl1140:~# chown pdal:pdal /home/pdal/download/*.pem  
root@CA-ssl1140:~# chmod 600 /home/pdal/download/*.pem  
root@CA-ssl1140:~# ls -l /home/pdal/download/  
total 12  
-rw----- 1 pdal pdal 2098 Jul  7 09:10 ca.cert.pem  
-rw----- 1 pdal pdal 1984 Jul  7 09:10 server.cert.pem  
-rw----- 1 pdal pdal 1704 Jul  7 09:11 server.key.pem  
root@CA-ssl1140:~#
```

◊ 2. Dateien mit WinSCP per SFTP vom CA-Container herunterladen. Schritte:

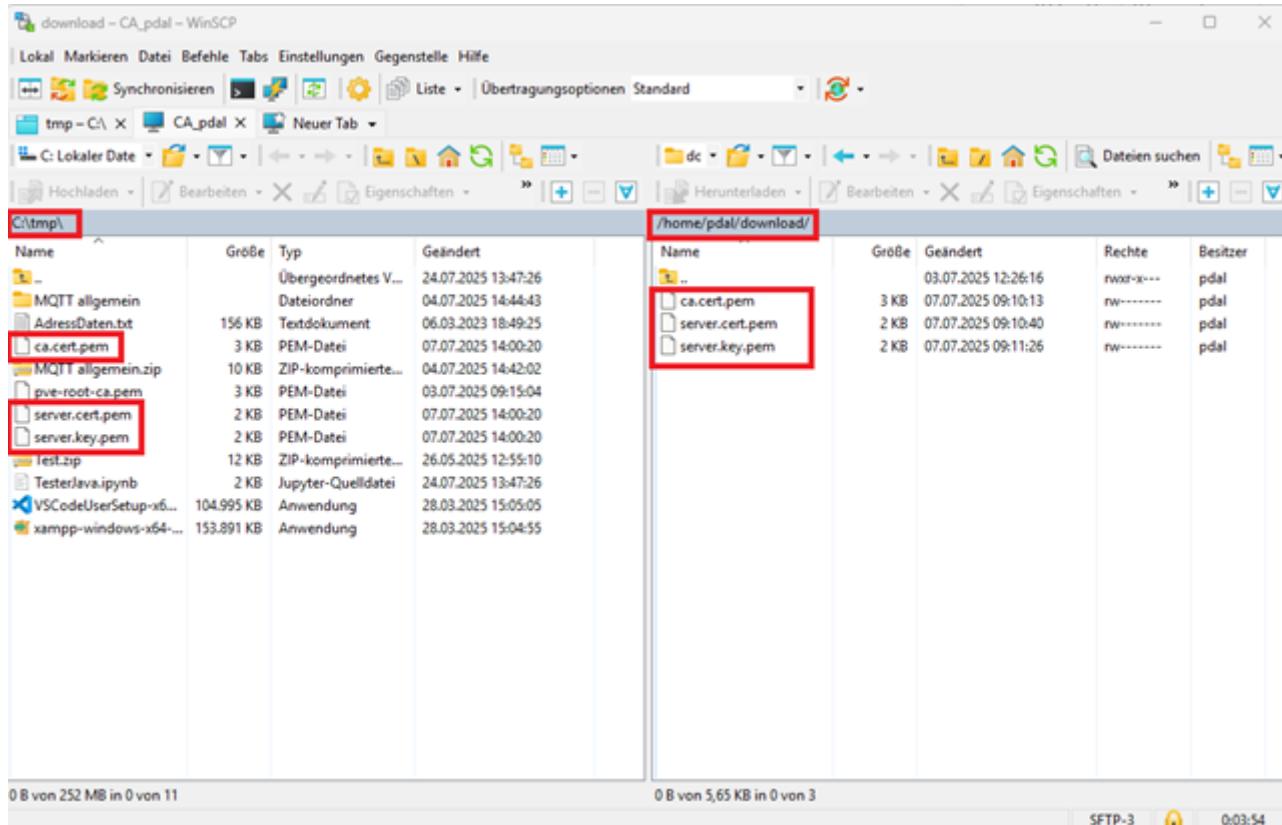
- Starte WinSCP.
- Öffne eine neue SFTP-Sitzung:
- Host: IP deines CA-Containers
- Benutzer: pdal
- Passwort: Passwort
- Navigiere im rechten Fenster (Remote-Browser) zu:

```
/home/pdal/download/
```

Markiere die Dateien **ca.cert.pem**, **server.cert.pem**, **server.key.pem**.

Ziehe sie in das lokale Verzeichnis auf deinem PC:

C:\tmp



Es geht aber auch komplett ohne extra Apps wie WinSCP. Ab Windows 10/11 habe ich die Möglichkeit über die Eingabeaufforderung über den Befehl `scp` die Dateien herunter zu laden. Hierzu die einzelnen Schritte wie das funktioniert. Wir öffnen auf unserem Windows Client die Eingabeaufforderung als Administrator.

```
scp pdal@192.168.137.140:/home/pdal/download/*.pem C:\tmp\
```

```
C:\Users\kalk>scp pdal@192.168.137.140:/home/pdal/download/*.pem C:\tmp\
The authenticity of host '192.168.137.140 (192.168.137.140)' can't be established.
ED25519 key fingerprint is SHA256:27QBEuxin+B4M5EkeqLNCr000bSwAJV9K8pE6GlhZPc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Please type 'yes', 'no' or the fingerprint:
Warning: Permanently added '192.168.137.140' (ED25519) to the list of known hosts.
pdal@192.168.137.140's password:
scp: Connection closed

C:\Users\kalk>scp pdal@192.168.137.140:/home/pdal/download/*.pem C:\tmp\
pdal@192.168.137.140's password:
ca.cert.pem          100% 2098      2.1KB/s  00:00
server.cert.pem      100% 1984      1.9KB/s  00:00
server.key.pem       100% 1704      1.7KB/s  00:00
```

Hier werden wir gefragt ob wir die Verbindung fortsetzen möchten, da der Host nicht erkannt wird. Wir müssen hier mit `yes` bestätigen. Es ist durchaus möglich das wir hier `yes` eingeben müssen und **Achtung!** uns wird nicht angezeigt was wir schreiben. Die 3 Dateien befinden sich nun im `tmp` Verzeichnis unter `C:`

- ◊ 3. Dateien per SFTP auf den Apache2-Container hochladen.

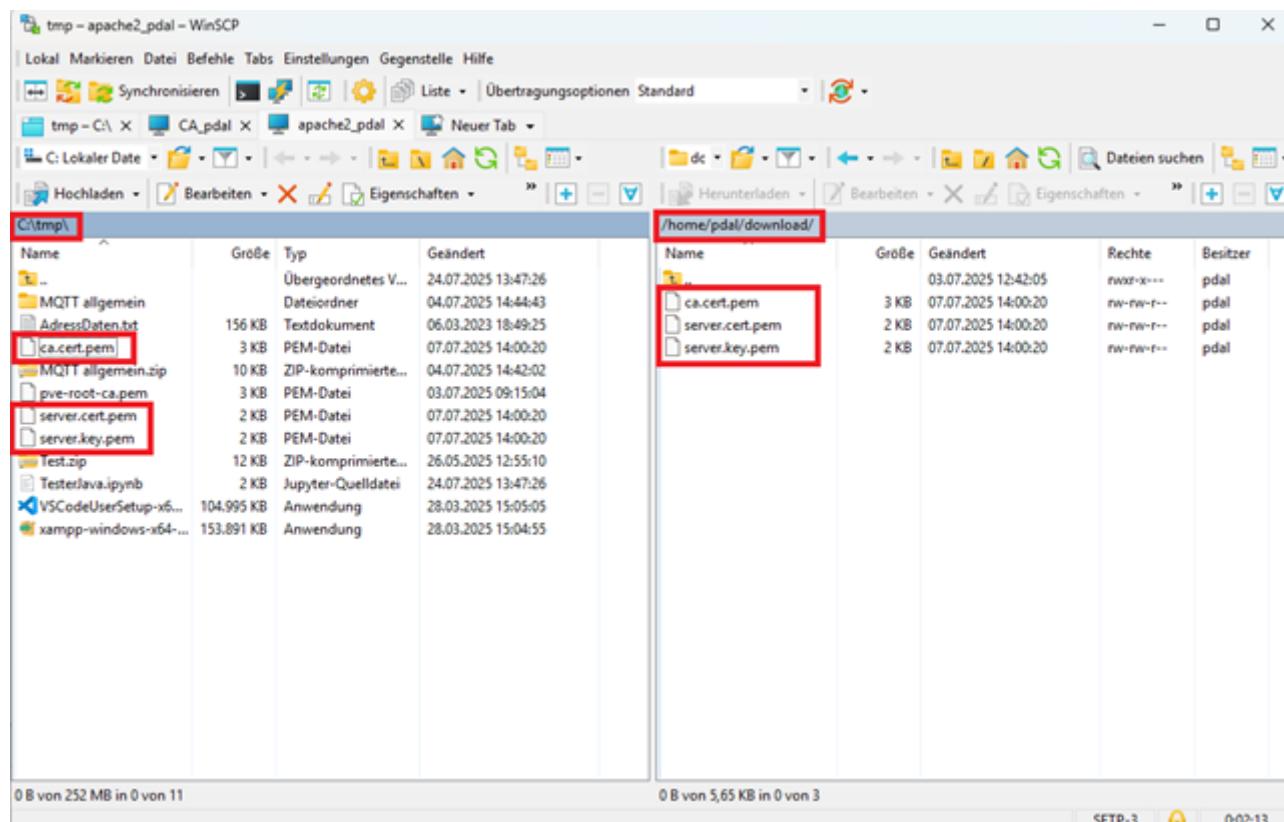
In WinSCP, öffne eine neue SFTP-Sitzung zum Apache2-Container:

- Host: 192.168.137.101
- Benutzer: pdal
- Passwort: JadeHS20

Navigiere im Remote-Browser zum Ziel:

```
/home/pdal/download/
```

Ziehe im lokalen Fenster (C:\tmp) die drei .pem-Dateien hinein.



- ◊ 4. Dateien im Apache2-Container an Zielorte verschieben Beispiel: Du meldest dich im Apache2-Container über die Proxmox-Webgui in der Konsole vom apache2 Container an:

Zertifikatsverzeichnis

```
/etc/ssl/certs  
/etc/ssl/private
```

Dateien überprüfen/ verschieben

```
ls -l /home/pdal/download/  
sudo mv /home/pdal/download/ca.cert.pem /etc/ssl/certs/
```

```
sudo mv /home/pdal/download/server.cert.pem /etc/ssl/certs/
sudo mv /home/pdal/download/server.key.pem /etc/ssl/private/
```

```
pdal@apache101:~$ ls -l /home/pdal/download/
total 12
-rw-rw-r-- 1 pdal pdal 2098 Jul  7 12:15 ca.cert.pem
-rw-rw-r-- 1 pdal pdal 1984 Jul  7 12:15 server.cert.pem
-rw-rw-r-- 1 pdal pdal 1704 Jul  7 12:15 server.key.pem
pdal@apache101:~$ sudo mv /home/pdal/download/ca.cert.pem /etc/ssl/certs/
[sudo] password for pdal:
pdal@apache101:~$ sudo mv /home/pdal/download/server..cert.pem /etc/ssl/certs/
mv: cannot stat '/home/pdal/download/server..cert.pem': No such file or directory
pdal@apache101:~$ sudo mv /home/pdal/download/server.cert.pem /etc/ssl/certs/
pdal@apache101:~$ sudo mv /home/pdal/download/server.key.pem /etc/ssl/private/
pdal@apache101:~$ ls -l /home/pdal/download/
total 0
pdal@apache101:~$ █
```

Rechte anpassen (wichtig für Apache2)

```
chmod 644 /etc/ssl/certs/ca.cert.pem
chmod 644 /etc/ssl/certs/server.cert.pem
sudo -i # wechselt auf root, da das Verzeichnis /etc/ssl/private nur von root
gelesen werden darf
chmod 600 /etc/ssl/private/server.key.pem
```

Anhand der Bilder kann man auch deutlich erkennen, dass wir nach jeder Änderung die Berechtigungen mit `ls -l` überprüfen.

```
pdal@apache101:~$ ls -l /etc/ssl/certs/ca.cert.pem
-rw-rw-r-- 1 pdal pdal 2098 Jul  7 12:15 /etc/ssl/certs/ca.cert.pem
pdal@apache101:~$ chmod 644 /etc/ssl/certs/ca.cert.pem
pdal@apache101:~$ ls -l /etc/ssl/certs/ca.cert.pem
-rw-r--r-- 1 pdal pdal 2098 Jul  7 12:15 /etc/ssl/certs/ca.cert.pem

-rw-rw-r-- 1 pdal pdal 1984 Jul  7 12:15 /etc/ssl/certs/server.cert.pem
pdal@apache101:~$ chmod 644 /etc/ssl/certs/server.cert.pem
pdal@apache101:~$ ls -l /etc/ssl/certs/server.cert.pem
-rw-r--r-- 1 pdal pdal 1984 Jul  7 12:15 /etc/ssl/certs/server.cert.pem
pdal@apache101:~$ █

pdal@apache101:~$ chmod 600 /etc/ssl/private/server.key.pem
chmod: cannot access '/etc/ssl/private/server.key.pem': Permission denied
pdal@apache101:~$ sudo -i
[sudo] password for pdal:
root@apache101:~# chmod 600 /etc/ssl/private/server.key.pem
root@apache101:~# ls -l /etc/ssl/private/server.key.pem
-rw----- 1 pdal pdal 1704 Jul  7 12:15 /etc/ssl/private/server.key.pem
root@apache101:~# █
```

Jetzt können wir in der Konsole `exit` eingeben um wieder zum User pdal zurück zu kehren. 🗝 Hinweis zur Apache2-Konfiguration

Vergiss nicht, in der Apache2-Konfiguration (z.B. `/etc/apache2/sites-available/default-ssl.conf`) die Pfade korrekt anzugeben:

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

```
SSLCertificateFile /etc/ssl/certs/server.cert.pem  
SSLCertificateKeyFile /etc/ssl/private/server.key.pem  
SSLCACertificateFile /etc/ssl/certs/ca.cert.pem
```

```
# SSL Engine Switch:  
# Enable/Disable SSL for this virtual host.  
SSLEngine on  
  
# A self-signed (snakeoil) certificate can be created by installing  
# the ssl-cert package. See  
# /usr/share/doc/apache2/README.Debian.gz for more info.  
# If both key and certificate are stored in the same file, only the  
# SSLCertificateFile directive is needed.  
SSLCertificateFile /etc/ssl/certs/server.cert.pem  
SSLCertificateKeyFile /etc/ssl/private/server.key.pem  
SSLCertificateChainFile /etc/ssl/certs/ca.cert.pem  
  
# Server Certificate Chain:  
# Point SSLCertificateChainFile at a file containing the  
# concatenation of PEM encoded CA certificates which form the  
# certificate chain for the server certificate. Alternatively  
# the referenced file can be the same as SSLCertificateFile  
# when the CA certificates are directly appended to the server  
# certificate for convinience.  
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt
```

Dann:

```
a2enmod ssl  
a2ensite default-ssl  
sudo systemctl restart apache2
```

```
pdal@apache101:~$ a2enmod ssl  
a2ensite default-ssl  
sudo systemctl restart apache2  
Considering dependency mime for ssl:  
Module mime already enabled  
Considering dependency socache_shmcb for ssl:  
Module socache_shmcb already enabled  
Module ssl already enabled  
Site default-ssl already enabled  
pdal@apache101:~$ █
```

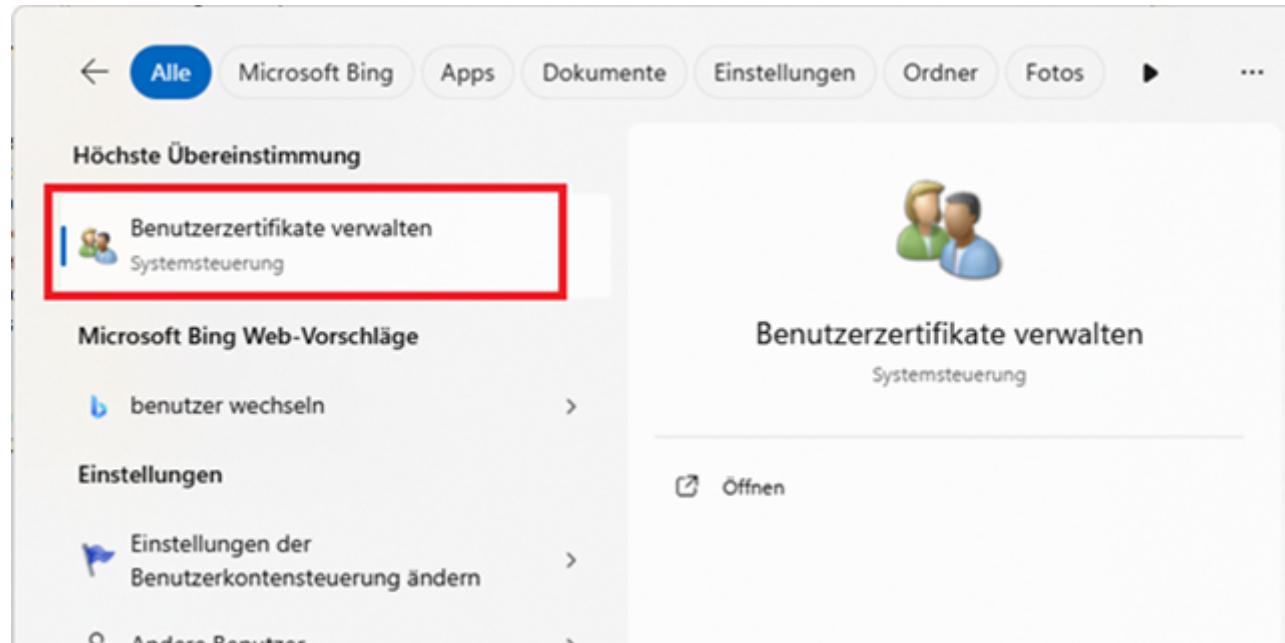
4.1 Apache2 (Beispiel)

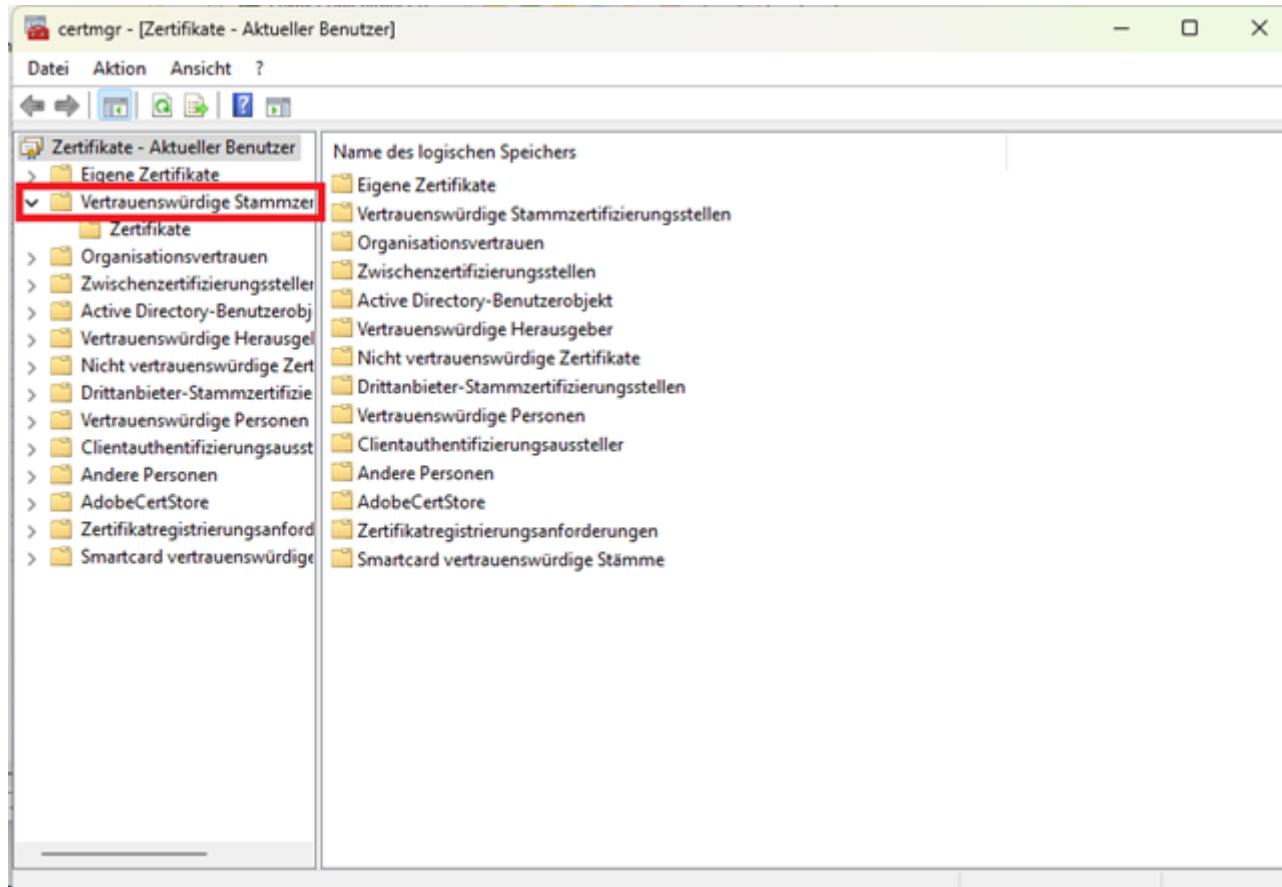
```
SSLCertificateFile /etc/ssl/certs/server.cert.pem  
SSLCertificateKeyFile /etc/ssl/private/server.key.pem  
SSLCACertificateFile /etc/ssl/certs/ca.cert.pem
```

5. Zertifikat auf Windows, Mac, Linux installieren

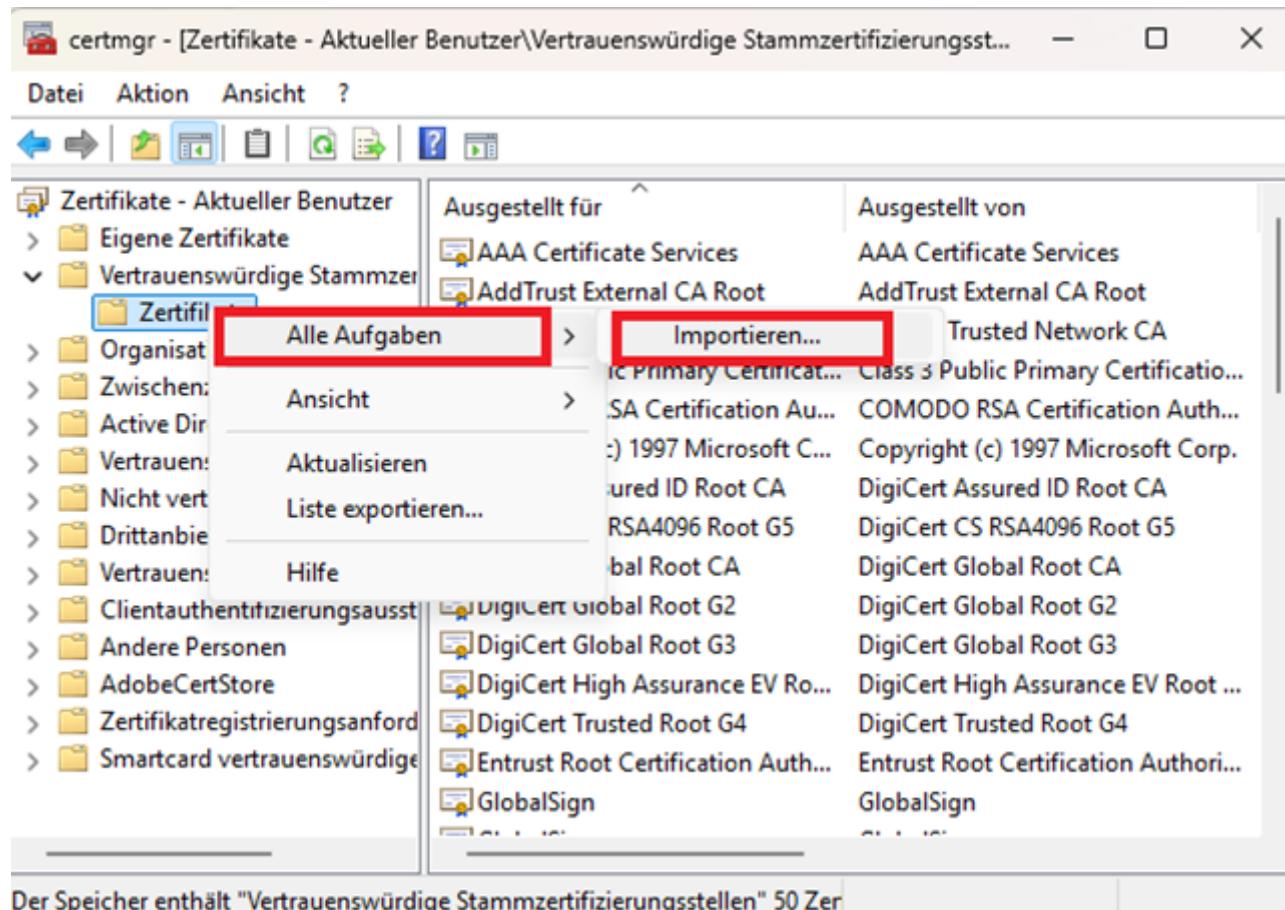
Importiere das CA-Zertifikat (ca.cert.pem) als vertrauenswürdige Zertifizierungsstelle in den jeweiligen Betriebssystemen. Diesen Schritt machen wir, damit Browser oder andere Tools eine sichere Verbindung ohne Zertifikatsfehler aufbauen können.

So erkennen deine Clients alle Zertifikate, die von deiner CA stammen, als vertrauenswürdig.





certmgr - [Zertifikate - Aktueller Benutzer\Vertrauenswürdige Stammzertifizierungsstellen]		
Datei	Aktion	Ansicht
<img alt="		



Der Speicher enthält "Vertrauenswürdige Stammzertifizierungsstellen" 50 Zer



← Zertifikatimport-Assistent

Willkommen

Dieser Assistent hilft Ihnen beim Kopieren von Zertifikaten, Zertifikatvertrauenslisten und Zertifikatssperrenlisten vom Datenträger in den Zertifikatspeicher.

Ein von einer Zertifizierungsstelle ausgestelltes Zertifikat dient der Identitätsbestätigung. Es enthält Informationen für den Datenschutz oder für den Aufbau sicherer Netzwerkverbindungen. Ein Zertifikatspeicher ist der Systembereich, in dem Zertifikate gespeichert werden.

Speicherort

- Aktueller Benutzer
 Lokaler Computer

Klicken Sie auf "Weiter", um den Vorgang fortzusetzen.

Weiter

Abbrechen



← Zertifikatimport-Assistent

Zu importierende Datei

Geben Sie die Datei an, die importiert werden soll.

Dateiname:

 Durchsuchen...

Hinweis: Mehrere Zertifikate können in einer Datei in folgenden Formaten gespeichert werden:

Privater Informationsaustausch - PKCS #12 (.PFX,.P12)

Syntaxstandard kryptografischer Meldungen - "PKCS #7"-Zertifikate (.P7B)

Microsoft Serieller Zertifikatspeicher (.SST)

Weiter

Abbrechen



← Zertifikatimport-Assistent

Zu importierende Datei

Geben Sie die Datei an, die importiert werden soll.

Dateiname:

C:\tmp\ca.cert.pem

[Durchsuchen...](#)

Hinweis: Mehrere Zertifikate können in einer Datei in folgenden Formaten gespeichert werden:

Privater Informationsaustausch - PKCS #12 (.PFX,.P12)

Syntaxstandard kryptografischer Meldungen - "PKCS #7"-Zertifikate (.P7B)

Microsoft Serieller Zertifikatspeicher (.SST)

[Weiter](#)

[Abbrechen](#)



← Zertifikatimport-Assistent

Zertifikatspeicher

Zertifikatspeicher sind Systembereiche, in denen Zertifikate gespeichert werden.

Windows kann automatisch einen Zertifikatspeicher auswählen, oder Sie können einen Speicherort für die Zertifikate angeben.

- Zertifikatspeicher automatisch auswählen (auf dem Zertifikattyp basierend)
- Alle Zertifikate in folgendem Speicher speichern

Zertifikatspeicher:

Vertrauenswürdige Stammzertifizierungsstellen

Durchsuchen...

Weiter

Abbrechen



← Zertifikatimport-Assistent

Fertigstellen des Assistenten

Das Zertifikat wird importiert, nachdem Sie auf "Fertig stellen" geklickt haben.

Sie haben folgende Einstellungen ausgewählt:

Vom Benutzer gewählter Zertifikatspeicher	Vertrauenswürdige Stammzertifizierungsstelle
Inhalt	Zertifikat
Dateiname	C:\tmp\ca.cert.pem

Fertig stellen

Abbrechen

Sicherheitswarnung



Sie sind im Begriff, ein Zertifikat von einer Zertifizierungsstelle zu installieren, die sich wie folgt darstellt:

pdal

Es wird nicht bestätigt, dass das Zertifikat wirklich von "pdal" stammt. Wenden Sie sich an "pdal", um die Herkunft zu bestätigen. Die folgende Zahl hilft Ihnen bei diesem Prozess weiter:

Fingerabdruck (sha1): 106AA179 DDB97C64 6A5C4E30
5DF92539 2093E3AD

Warnung:

Wenn Sie dieses Stammzertifikat installieren, wird automatisch allen Zertifikaten vertraut, die von dieser Zertifizierungsstelle ausgestellt werden. Die Installation mit einem unbestätigten Fingerabdruck stellt ein Sicherheitsrisiko dar. Falls Sie auf "Ja" klicken, nehmen Sie dieses Risiko in Kauf.

Möchten Sie dieses Zertifikat installieren?

Zertifikatimport-Assistent



Der Importvorgang war erfolgreich.

6. Zusammenfassung und Empfehlungen

Ein SAN-Zertifikat für alle Dienste ist für kleine Umgebungen schnell und einfach.

Für Produktion und mehr Sicherheit: Erstelle für jeden Dienst ein eigenes Zertifikat (mit eigenem CSR und Signierung).

Private Schlüssel immer sicher aufbewahren!

Verteile CA-Zertifikat an alle Clients, damit die TLS-Verbindungen als sicher erkannt werden.

Quellen

- „Eigene Zertifikate mit openssl — Linux und Open Source“. Zugegriffen: 26. August 2025. [Online]. Verfügbar unter: [OpenSSL Grundwissen](#)
 - „CA > Wiki > ubuntuusers.de“. Zugegriffen: 26. August 2025. [Online]. Verfügbar unter: [CA Wiki Ubuntuusers](#)
-

Lizenz

Dieses Werk ist lizenziert unter der **Creative Commons - Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz**.

[Zum Lizenztext auf der Creative Commons Webseite](#)