

Creating a Custom CA (Certificate Authority) and Manually Distributing Certificates

What is this about?

Why do we need certificates in IT?

In IT, certificates are used to uniquely and reliably prove the identity of devices, services, or individuals. They also enable encrypted and secure communication over networks, such as via HTTPS or VPN. Even in a closed network, you should use your own certificates. Since public Certificate Authorities (CAs) do not have access to internal systems, we need our own CA.

Custom certificates enable encrypted and authenticated communication via protocols like HTTPS, SMTPS, LDAPS, FTPS, as well as for databases like MariaDB, PostgreSQL, and message brokers like MQTT or Kafka. This allows you to retain full control over security, validity, and distribution of certificates within the network.

Certificates created locally in the application container vs. a custom CA (Certificate Authority). It is possible to create custom certificates in every application container (LXC) – but we have decided against it. We are setting up our own CA to minimize the complexity of distributing individual certificates. A custom CA provides us with unified, centrally managed certificates.

This guide shows you how to use **OpenSSL** on your server **ssl-ca140** (IP: 192.168.137.140) to create your own **Certificate Authority (CA)** and a **single certificate for all services**. The certificates are then manually distributed to the various services and client computers (Windows, Mac, Linux).

Why a Single Certificate — and Why Multiple?

- **A single certificate (SAN Certificate)** is practical: You only need one file that is valid for all your services. This saves effort and is often sufficient for small or test environments.
 - **Multiple certificates (recommended for production):** Each service receives its own certificate, which improves security, as only that service is affected if compromised. Management (renewal, revocation) also becomes more flexible this way, but is significantly more complex or must be automated.
-

Prerequisites

- An LXC container **ssl-ca140** with OpenSSL installed
- Familiarity with basic Linux commands
- SSH access to the server

```
sudo apt install openssl
```

Create a new user **pdal** with sudo rights. Switch the user to **pdal** with **su pdal**.

1. Create Folder Structure

```
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. Create CA (Certificate Authority)

2.1 Create OpenSSL configuration file openssl.cnf

In the folder `~/myCA`, create the file `openssl.cnf` with the following content (simplified example):

```
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

```

Hint: Adjust paths and names as needed. Replace `/home/username` with your actual path.

Explanation of the Configuration File:

[ca]

```
default_ca = CA_default
```

Defines which CA configuration is used.

CA_default refers to the following section [CA_default].

[CA_default]

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

'\$dir' is the root directory of your CA files (Adjust 'username' to your username).

All subsequent paths are built upon this.

```

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

```

certs: Stores issued certificates

crl_dir: For revoked certificates (Certificate Revocation Lists)

new_certs_dir: Internal folder for new certificates

database: File where OpenSSL documents the issued certificates

serial: File with the current serial number

RANDFILE: Helper file for random data (e.g., during key generation)

```

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

```

Storage locations of the CA key and CA certificate

```

default_days    = 3650
default_crl_days = 30
default_md      = sha256

```

default_days: Validity period of a new certificate (here: 10 years)

default_crl_days: How long a revocation list is valid

default_md: Hash algorithm (here: sha256, more secure than e.g., md5)

```

policy          = policy_loose

```

Defines how strictly OpenSSL checks the entered fields (e.g., whether country or organization is mandatory)

[policy_loose]

```

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

```

This is a loose policy: most fields are optional, only 'commonName' (the server name like 'apache.local') must be provided.

[req]

```

default_bits      = 4096
distinguished_name = req_distinguished_name
string_mask       = utf8only

```

Configuration for new Certificate Signing Requests (CSRs).

default_bits: Key length in bits (the more, the more secure – 4096 is strong).

distinguished_name: Which fields are queried during the CSR.

string_mask = utf8only: Only UTF-8 characters allowed.

[req_distinguished_name]

```

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

```

This defines which fields are queried when creating a certificate (e.g., Country, Organization, Common Name).

[v3_ca]

```

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

```

Extensions for CA Certificates

basicConstraints = CA:true: Clarifies that this certificate is allowed to issue certificates.

keyUsage: Which actions are allowed with the key.

[v3_server]

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

Extensions for Server Certificates:

- **CA:FALSE:** This certificate is not a CA certificate,
- **keyUsage:** For digital signature and encryption,
- **extendedKeyUsage:** May be used as a TLS server certificate.
- **subjectAltName:** Refers to the list of alternative names below.

[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
```

This is the SAN list (Subject Alternative Name).

It allows the certificate to secure multiple names and IP addresses, e.g., `apache.local` for Apache, `mariadb.local` for MariaDB, etc.

This means only a single certificate is needed for all services – this is the so-called pdal certificate.

Conclusion

This configuration defines a custom CA with:

```
loose input requirements,
strong key (4096 bit),
```

SAN support for many services,

1 certificate for all servers (pdal certificate).

2.2 Create CA Key and CA Certificate

Go to the `~/myCA` folder with the command `cd ~/myCA/`.

```
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:[]
```

You enter a password for confirmation. We choose `JadeHS20` in our example, after which you have to confirm the password again.

```
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
```

You will be asked for details like country, organization, etc. You **must** fill these in appropriately.

```
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
Or 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. Create Server Certificate with SAN (for all services)

3.1 Create Private Key for Server

```
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 Create Certificate Signing Request (CSR) with SAN

Create the CSR (Certificate Signing Request):

What is a CSR?

A CSR is a file that:

- contains a public key,
- was created along with identity information (e.g., Common Name, Organization, Location, SANs),
- is intended to be signed by a Certificate Authority (CA) to generate a valid certificate.

```
sudo openssl req \
  -new \
  -key private/server.key.pem \
  -out server.csr.pem \
  -config ~/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$
```

Afterward, you can check if the key was actually created with the following command:

```
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)

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 Sign Certificate with SAN

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

You will be asked if you want to sign the certificate – confirm with y. Then you must confirm once more with y for the certificate to be created and written to the database.

```
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]:
```

Afterward, you must set the permissions on the certificate.

```
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. Distribute and Configure Certificates to Services

The certificate `server.cert.pem`, the key `server.key.pem`, and the CA certificate `ca.cert.pem` are needed on all servers/services.



We want to:

- Copy the files `ca.cert.pem`, `server.cert.pem`, and `server.key.pem` in the CA container to `/home/pdal/download`.
- Download these files to your local computer in `C:\tmp` using WinSCP via SFTP.
- Upload the files later to the Apache2 container to `/home/pdal/download` using WinSCP via SFTP. (This will be explained in a separate document.)
- Move the files from there into the corresponding directories, e.g., in the Apache2 container.



- LXC container for CA (ca-container) is active.
- User `pdal` exists.

- SFTP access via IP or hostname is possible.
 - SFTP client such as WinSCP is installed on your Windows computer (but also works without a separate client via the command line).
- ◊ 1. Prepare files in the CA container

Create Download Directory

We first create a directory named `download` in the home directory of the user `pdal` in the CA container.

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

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

Adjust Paths, if necessary

Now we copy the individual certificates and the server key into the download directory. Since the `/home/pdal/myCA/private` directory belongs to root and only root is allowed to access this directory, we must later log in to the container as `root`. We do this as follows. We are currently logged in as `pdal` and must enter `sudo -i` to switch to the user `root` to gain access to the private directory.

```
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/
```

This follows the syntax `sudo cp theFileToCopyWithPath theTargetDirectory`.

```
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:~#
```

Adjust Permissions (optional, so that user pdal has access)

We are still in the container as root. If not, please switch to the root level with `sudo -i`.

The files just copied all belong to the user Root and have very strict permissions. To be able to download these files via an SFTP client, we must change the permissions and also the owner of the files. We do this with the `chown` command and the `chmod` command.

```
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
----- 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-ssl140:~# chown pdal:pdal /home/pdal/download/*.pem  
root@CA-ssl140:~# chmod 600 /home/pdal/download/*.pem  
root@CA-ssl140:~# 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-ssl140:~#
```

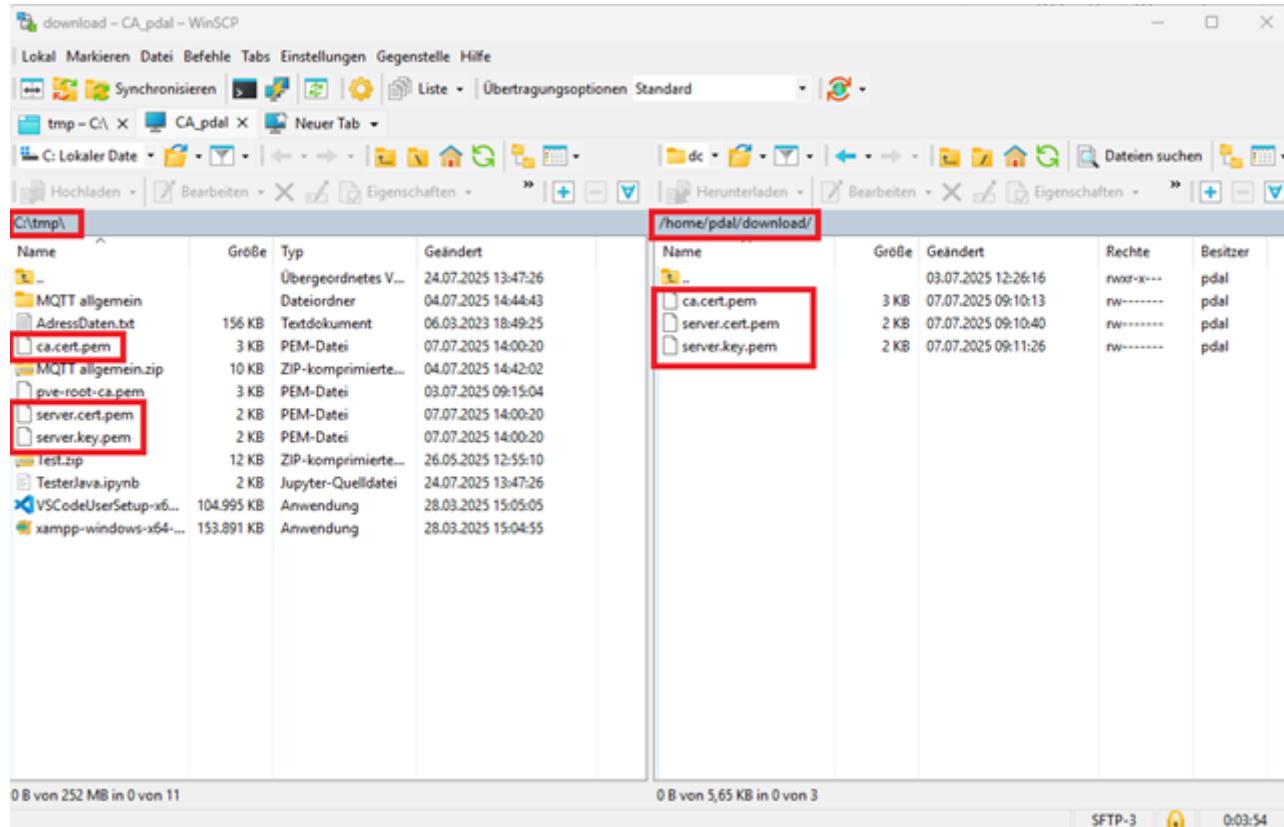
◊ 2. Download files from the CA container via SFTP using WinSCP. Steps:

- Start WinSCP.
- Open a new SFTP session:
 - Host: IP of your CA container
 - User: **pdal**
 - Password: Password
- Navigate in the right window (Remote Browser) to:

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

- Select the files **ca.cert.pem**, **server.cert.pem**, **server.key.pem**.
- Drag them to the local directory on your PC:

C:\tmp



It also works completely without extra apps like WinSCP. Starting with Windows 10/11, I have the option to download the files via the command prompt using the `scp` command. Here are the individual steps on how this works. We open the command prompt on our Windows client as 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
```

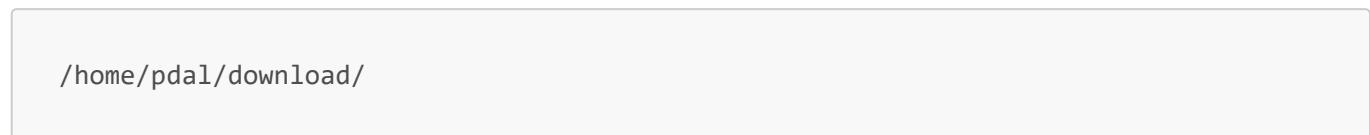
Here we are asked if we want to continue the connection because the host is not recognized. We must confirm with `yes`. The 3 files are now in the `tmp` directory under `C:`.

- ◊ 3. Upload files to the Apache2 container via SFTP.

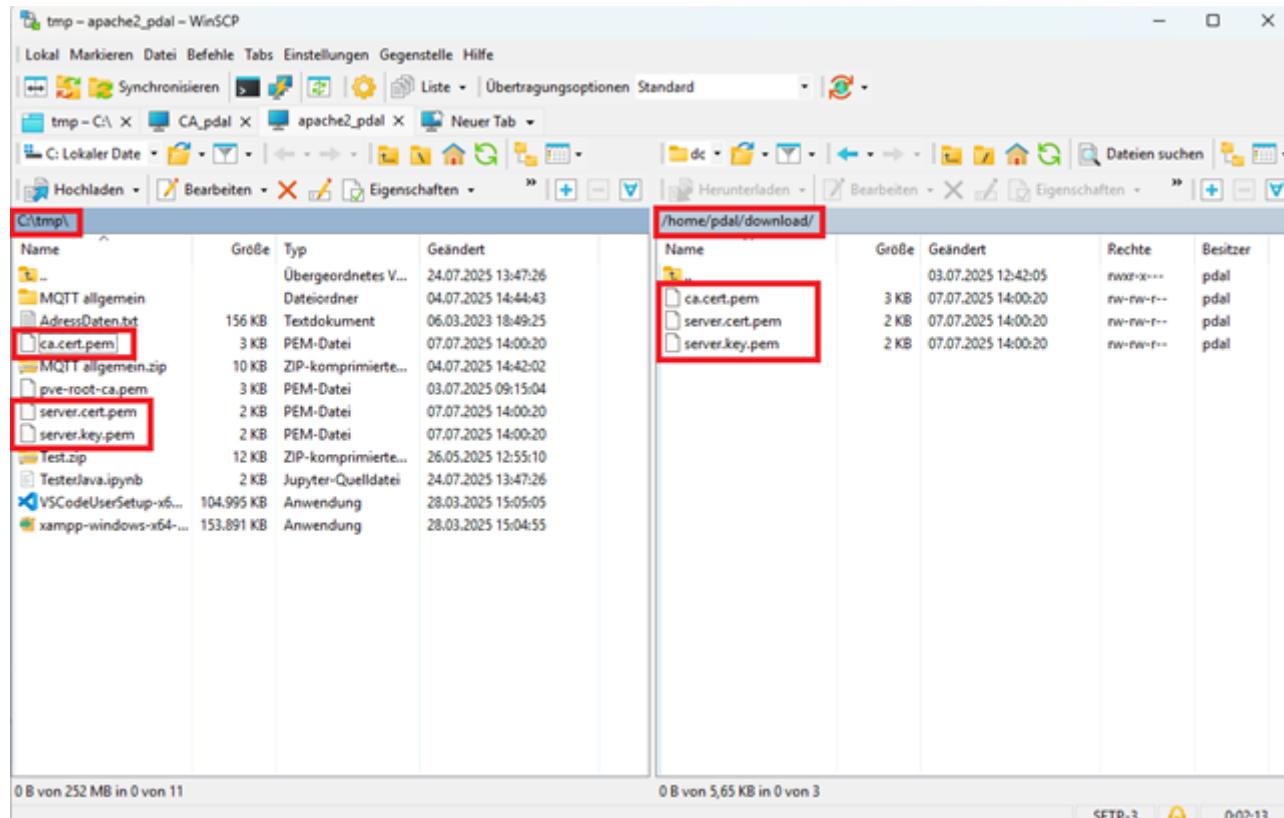
In WinSCP, open a new SFTP session to the Apache2 container:

- Host: 192.168.137.101
- User: pdal
- Password: JadeHS20

Navigate in the Remote Browser to the destination:



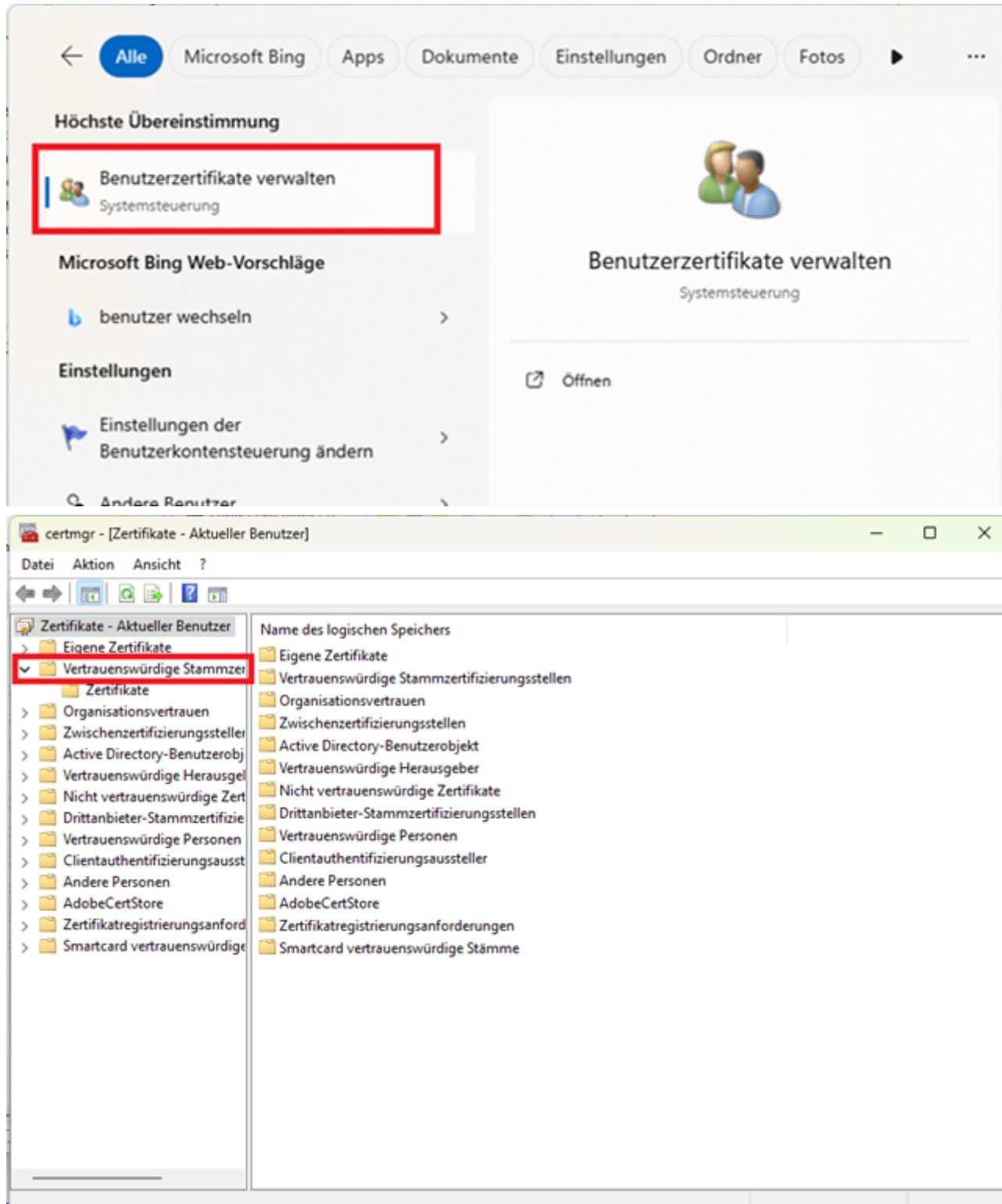
Drag the three .pem files from the local window (C:\tmp) into it.



5. Install Certificate on Windows, Mac, Linux

Import the CA certificate ([ca.cert.pem](#)) as a trusted root certificate authority in the respective operating systems. We do this so that browsers or other tools can establish a secure connection without certificate errors.

This way, your clients will recognize all certificates originating from your CA as trustworthy.



certmgr - [Zertifikate - Aktueller Benutzer\Vertrauenswürdige Stammzertifizierungsstellen]

Datei Aktion Ansicht ?

Zertifikate - Aktueller Benutzer

- > Eigene Zertifikate
- > Vertrauenswürdige Stammzertifizierungsstellen
 - Zertifikate**
 - > Organisationsvertrauen
 - > Zwischenzertifizierungsstellen
 - > Active Directory-Benutzerobjekte
 - > Vertrauenswürdige Herausgeber
 - > Nicht vertrauenswürdige Zertifikate
 - > Drittanbieter-Stammzertifizierungsstellen
 - > Vertrauenswürdige Personen
 - > Clientauthentifizierungsaussteller
 - > Andere Personen
 - > AdobeCertStore
 - > Zertifikatregistrierungsanfordungen
 - > Smartcard vertrauenswürdige Zertifikate

Ausgestellt für	Ausgestellt von
AAA Certificate Services	AAA Certificate Services
AddTrust External CA Root	AddTrust External CA Root
Certum Trusted Network CA	Certum Trusted Network CA
Class 3 Public Primary Certificate Authority	Class 3 Public Primary Certification Authority
COMODO RSA Certification Authority	COMODO RSA Certification Authority
Copyright (c) 1997 Microsoft Corporation	Copyright (c) 1997 Microsoft Corp.
DigiCert Assured ID Root CA	DigiCert Assured ID Root CA
DigiCert CS RSA4096 Root G5	DigiCert CS RSA4096 Root G5
DigiCert Global Root CA	DigiCert Global Root CA
DigiCert Global Root G2	DigiCert Global Root G2
DigiCert Global Root G3	DigiCert Global Root G3
DigiCert High Assurance EV Root CA	DigiCert High Assurance EV Root CA
DigiCert Trusted Root G4	DigiCert Trusted Root G4
Entrust Root Certification Authority	Entrust Root Certification Authority
GlobalSign	GlobalSign

Der Speicher enthält "Vertrauenswürdige Stammzertifizierungsstellen" 49 Zertifikate

certmgr - [Zertifikate - Aktueller Benutzer\Vertrauenswürdige Stammzertifizierungsstellen]

Datei Aktion Ansicht ?

Zertifikate - Aktueller Benutzer

- > Eigene Zertifikate
- > Vertrauenswürdige Stammzertifizierungsstellen
 - Zertifikate**
 - Alle Aufgaben**
 - > Importieren...
- > Organisationsvertrauen
- > Zwischenzertifizierungsstellen
- > Active Directory-Benutzerobjekte
- > Vertrauenswürdige Herausgeber
- > Nicht vertrauenswürdige Zertifikate
- > Drittanbieter-Stammzertifizierungsstellen
- > Vertrauenswürdige Personen
- > Clientauthentifizierungsaussteller
- > Andere Personen
- > AdobeCertStore
- > Zertifikatregistrierungsanfordungen
- > Smartcard vertrauenswürdige Zertifikate

Ausgestellt für	Ausgestellt von
AAA Certificate Services	AAA Certificate Services
AddTrust External CA Root	AddTrust External CA Root
Class 3 Public Primary Certificate Authority	Class 3 Public Primary Certification Authority
COMODO RSA Certification Authority	COMODO RSA Certification Authority
Copyright (c) 1997 Microsoft Corporation	Copyright (c) 1997 Microsoft Corp.
DigiCert Assured ID Root CA	DigiCert Assured ID Root CA
DigiCert CS RSA4096 Root G5	DigiCert CS RSA4096 Root G5
DigiCert Global Root CA	DigiCert Global Root CA
DigiCert Global Root G2	DigiCert Global Root G2
DigiCert Global Root G3	DigiCert Global Root G3
DigiCert High Assurance EV Root CA	DigiCert High Assurance EV Root CA
DigiCert Trusted Root G4	DigiCert Trusted Root G4
Entrust Root Certification Authority	Entrust Root Certification Authority
GlobalSign	GlobalSign

Der Speicher enthält "Vertrauenswürdige Stammzertifizierungsstellen" 50 Zertifikate



← 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](#)

X

 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:



← 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. Summary and Recommendations

A SAN certificate for all services is quick and easy for small environments.

For production and greater security: Create a separate certificate for each service (with its own CSR and signing).

Always keep private keys secure!

Distribute the CA certificate to all clients so that TLS connections are recognized as secure.

Sources

- "Own certificates with openssl — Linux and Open Source". Accessed: August 26, 2025. [Online]. Available at: [OpenSSL Basic Knowledge](#)
 - "CA > Wiki > ubuntuusers.de". Accessed: August 26, 2025. [Online]. Available at: [CA Wiki Ubuntuusers](#)
-

License

This work is licensed under the **Creative Commons Attribution-ShareAlike 4.0 International License**.

[To the license text on the Creative Commons website](#)