docker

Connect to Container as root

```
$ docker exec -u 0 -it FW_pgAdmin4 /bin/ash
```

PostgreSQL DEV refresh

```
docker exec -it dev-container_devcontainer_db_1 /bin/bash
```

View filesystem

```
docker image save fueltracker_app >image.tgz
```

SSL/TLS connection

Generate a CA, server, and client certificate/key pair

1. Create a working directory

```
mkdir -p ~/.docker/tls && cd ~/.docker/tls
```

2. Generate the CA private key

```
openssl genrsa -aes256 -out ca-key.pem 4096
```

3. Generate the CA certificate

```
openssl req -new -x509 -days 365 -key ca-key.pem -sha256 -out ca.pem
```

You'll be prompted for a passphrase and subject info (Country, CN, etc.).

4. Generate the server private key

```
openssl genrsa -out server-key.pem 4096
```

5. Generate the server CSR (Certificate Signing Request)

```
openssl req -subj "/CN=<your-server-hostname-or-IP>" -sha256 -new -key
server-key.pem -out server.csr
```

Replace <your-server-hostname-or-IP> with your Docker host's hostname or IP (e.g., mydockerhost or 192.168.1.10).

6. Create a SANs (Subject Alternative Names) extension file

```
echo subjectAltName = DNS:<hostname>,IP:<IP>,IP:127.0.0.1 > extfile.cnf
echo extendedKeyUsage = serverAuth >> extfile.cnf
```

Add all hostnames/IPs clients will use to reach the server.

7. Sign the server certificate with the CA

```
openssl x509 -req -days 365 -sha256 \
  -in server.csr -CA ca.pem -CAkey ca-key.pem \
  -CAcreateserial -out server-cert.pem -extfile extfile.cnf
```

8. Generate the client private key

```
openssl genrsa -out key.pem 4096
```

9. Generate the client CSR

```
openssl req -subj '/CN=client' -new -key key.pem -out client.csr
```

10. Create a client extension file

```
echo extendedKeyUsage = clientAuth > extfile-client.cnf
```

11. Sign the client certificate with the CA

```
openssl x509 -req -days 365 -sha256 \
  -in client.csr -CA ca.pem -CAkey ca-key.pem \
  -CAcreateserial -out cert.pem -extfile extfile-client.cnf
```

12. Clean up CSRs and extension files, lock down permissions

```
rm -f client.csr server.csr extfile.cnf extfile-client.cnf
chmod 0400 ca-key.pem key.pem server-key.pem
chmod 0444 ca.pem server-cert.pem cert.pem
```

After this you'll have:

```
┌──────────────────────────────────┬──────────────────────────────────────────────────
──────────┐
│               File               │                      Purpose
│
├──────────────────────────────────┼──────────────────────────────────────────────────
──────────┤
│ ca.pem                           │ CA certificate (needed by both server
and client) │
├──────────────────────────────────┼──────────────────────────────────────────────────
──────────┤
│ server-cert.pem / server-key.pem │ Server certificate and key
│
├──────────────────────────────────┼──────────────────────────────────────────────────
```

```
──────────────┤
│ cert.pem / key.pem                │ Client certificate and key
│
└──────────────────────────────────┴──────────────────────────────────────────
──────────────┘
```

Configure /etc/docker/daemon.json with "tls": true, "tlscacert", "tlscert", and "tlskey"

1. Copy server certs to a permanent location

```
sudo mkdir -p /etc/docker/certs
sudo cp ~/.docker/tls/ca.pem /etc/docker/certs/
sudo cp ~/.docker/tls/server-cert.pem /etc/docker/certs/
sudo cp ~/.docker/tls/server-key.pem /etc/docker/certs/
sudo chmod 0444 /etc/docker/certs/ca.pem /etc/docker/certs/server-cert.pem
sudo chmod 0400 /etc/docker/certs/server-key.pem
```

2. Create or edit /etc/docker/daemon.json

```
sudo tee /etc/docker/daemon.json > /dev/null <<EOF
{
  "tls": true,
  "tlsverify": true,
  "tlscacert": "/etc/docker/certs/ca.pem",
  "tlscert": "/etc/docker/certs/server-cert.pem",
  "tlskey": "/etc/docker/certs/server-key.pem",
  "hosts": ["unix:///var/run/docker.sock", "tcp://0.0.0.0:2376"]
}
EOF
```

1. “tls”: true — enables TLS but allows unauthenticated clients
2. “tlsverify”: true — enforces mutual TLS (clients must present a valid cert)
3. Port 2376 is the standard TLS Docker port (2375 is unencrypted)

3. Handle the hosts conflict with systemd (WSL/Linux)

If Docker is managed by systemd, the -H flag in the service file conflicts with hosts in daemon.json. Override it:

```
sudo mkdir -p /etc/systemd/system/docker.service.d
sudo tee /etc/systemd/system/docker.service.d/override.conf > /dev/null
<<EOF
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd
EOF
```

Enable systemd in WSL2 (recommended).

Edit /etc/wsl.conf

```
sudo tee /etc/wsl.conf > /dev/null <<EOF
[boot]
systemd=true
EOF
```

4. Reload and restart Docker

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

5. Verify Docker is listening on port 2376

```
sudo ss -tlnp | grep 2376
```

You should see dockerd bound to 0.0.0.0:2376. That confirms TLS is active and the daemon is accepting remote connections.

References

Docker Exec Command With Examples - https://devconnected.com/docker-exec-command-with-examples/

Exploring Docker container's file system - https://stackoverflow.com/questions/20813486/exploring-docker-containers-file-system

[Docker]