Installation

- Docker installation
- Adminrechte vergeben

Docker installation

Beschreibung:

Installation

Pakete installieren

```
apt install docker.io docker-compose apparmor apparmor-utils curl openssl
```

Verzeichnisse anlegen

```
mkdir -p /root/documenso/certs
mkdir -p /root/documenso/data-db
```

Wenn gewünscht Caddy anlege, sonst überspringen

```
mkdir -p /root/documenso/caddy_data
mkdir -p /root/documenso/caddy_config
```

```
nano /root/documenso/Caddyfile
```

Inhalt, wenn man es erst testen möchte mit Staging einfach den Kommentar entfernen.

```
example.com {
    reverse_proxy http://documenso:3000
    tls <deine_emailadresse> {
        #ca https://acme-staging-v02.api.letsencrypt.org/directory
    }
}
```

.env Datei anlegen und ausfüllen

Für NEXTAUTH_SECRET und NEXTAUTH_PRIVATE_ENCRYPTION_KEY und NEXTAUTH_PRIVATE_ENCRYPTION_SECONDARY_KEY
erstellen wir mit folgendem Befehl die Secrets. Für jede Variable einen neuen secret.

```
openssl rand -base64 32
```

Inhalt

```
POSTGRES_USER=docuuser
POSTGRES_PASSWORD=docupass
POSTGRES_DB=documenso
PORT=3000
#die url unter der das backend erreichbar ist, gleich mit public webapp
NEXTAUTH_URL=http://localhost
#wird mit dem Befehl: openssl rand -base64 32  erstellt
NEXTAUTH_SECRET=your-secret-key
#wird mit dem Befehl: openssl rand -base64 32  erstellt
NEXT_PRIVATE_ENCRYPTION_KEY=your-encryption-key
#wird mit dem Befehl: openssl rand -base64 32  erstellt
NEXT_PRIVATE_ENCRYPTION_SECONDARY_KEY=your-secondary-encryption-key
#wenn sich läute mit google mail anmelden sollen
#NEXT_PRIVATE_GOOGLE_CLIENT_ID=your-google-client-id
#NEXT_PRIVATE_GOOGLE_CLIENT_SECRET=your-google-client-secret
#die URL die von Außen erreichbar ist
NEXT_PUBLIC_WEBAPP_URL=http://localhost:3000
#muss gleich der public webapp url sein
NEXT_PRIVATE_INTERNAL_WEBAPP_URL=http://localhost:3000
#Die URL für eine Shop seite oder Dokemnetion.
NEXT_PUBLIC_MARKETING_URL=https://documenso.com
NEXT_PRIVATE_DATABASE_URL=postgres://docuuser:docupass@database:5432/documenso
NEXT_PRIVATE_DIRECT_DATABASE_URL=postgres://docuuser:docupass@database:5432/documenso
NEXT_PUBLIC_UPLOAD_TRANSPORT=database
NEXT_PRIVATE_SMTP_TRANSPORT=smtp
NEXT_PRIVATE_SMTP_HOST=smtp.example.com
NEXT_PRIVATE_SMTP_PORT=465
NEXT_PRIVATE_SMTP_SECURE="true"
NEXT_PRIVATE_SMTP_USERNAME=smtp-user
NEXT_PRIVATE_SMTP_PASSWORD=smtp-password
NEXT_PRIVATE_SMTP_FROM_NAME=Documenso Support
NEXT_PRIVATE_SMTP_FROM_ADDRESS=support@documenso.com
```

```
NEXT_PRIVATE_SIGNING_LOCAL_FILE_PATH=/opt/documenso/cert.p12
#die passphrase vom certificate
NEXT_PRIVATE_SIGNING_PASSPHRASE=signing-passphrase
#es durfen sich keine benutzer registrieren. False man darf sich NICHT registreiren, true man DARF
NEXT_PUBLIC_DISABLE_SIGNUP="false"
```

Docker compose Datei anlegen

ohne caddy

```
nano /root/documenso/docker-compose.yml
```

Inhalt

```
version: '3.8'

services:
  database:
    image: postgres:15
    environment:
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
      - POSTGRES_DB=${POSTGRES_DB}
    healthcheck:
      test: ['CMD-SHELL', 'pg_isready -U ${POSTGRES_USER}']
      interval: 10s
      timeout: 5s
      retries: 5
    volumes:
      - ./data-db:/var/lib/postgresql/data  # Speichert Daten außerhalb des Containers
    restart: always

  documenso:
    image: documenso/documenso:latest
    depends_on:
      database:
        condition: service_healthy
    environment:
      - PORT=${PORT:-3000}
```

```
      - NEXTAUTH_URL=${NEXTAUTH_URL:-${NEXT_PUBLIC_WEBAPP_URL}
      - NEXTAUTH_SECRET=${NEXTAUTH_SECRET:?err}
      - NEXT_PRIVATE_ENCRYPTION_KEY=${NEXT_PRIVATE_ENCRYPTION_KEY:?err}
      - NEXT_PRIVATE_ENCRYPTION_SECONDARY_KEY=${NEXT_PRIVATE_ENCRYPTION_SECONDARY_KEY:?err}
      #wir wollen kein google
      #- NEXT_PRIVATE_GOOGLE_CLIENT_ID=${NEXT_PRIVATE_GOOGLE_CLIENT_ID}
      #- NEXT_PRIVATE_GOOGLE_CLIENT_SECRET=${NEXT_PRIVATE_GOOGLE_CLIENT_SECRET}
      - NEXT_PUBLIC_WEBAPP_URL=${NEXT_PUBLIC_WEBAPP_URL:?err}
      - NEXT_PRIVATE_INTERNAL_WEBAPP_URL=${NEXT_PRIVATE_INTERNAL_WEBAPP_URL:-
http://localhost:$PORT}
      - NEXT_PUBLIC_MARKETING_URL=${NEXT_PUBLIC_MARKETING_URL:-https://documenso.com}
      - NEXT_PRIVATE_DATABASE_URL=${NEXT_PRIVATE_DATABASE_URL:?err}
      - NEXT_PRIVATE_DIRECT_DATABASE_URL=${NEXT_PRIVATE_DIRECT_DATABASE_URL:-
${NEXT_PRIVATE_DATABASE_URL}
      - NEXT_PUBLIC_UPLOAD_TRANSPORT=${NEXT_PUBLIC_UPLOAD_TRANSPORT:-database}
      #Diese Variablen werden nur benötigt wenn die Daten nicht in einer Datenbank sondern in einem S3
      #Bucket gespeichert werden sollen
      #- NEXT_PRIVATE_UPLOAD_ENDPOINT=${NEXT_PRIVATE_UPLOAD_ENDPOINT}
      #- NEXT_PRIVATE_UPLOAD_FORCE_PATH_STYLE=${NEXT_PRIVATE_UPLOAD_FORCE_PATH_STYLE}
      #- NEXT_PRIVATE_UPLOAD_REGION=${NEXT_PRIVATE_UPLOAD_REGION}
      #- NEXT_PRIVATE_UPLOAD_BUCKET=${NEXT_PRIVATE_UPLOAD_BUCKET}
      #- NEXT_PRIVATE_UPLOAD_ACCESS_KEY_ID=${NEXT_PRIVATE_UPLOAD_ACCESS_KEY_ID}
      #- NEXT_PRIVATE_UPLOAD_SECRET_ACCESS_KEY=${NEXT_PRIVATE_UPLOAD_SECRET_ACCESS_KEY}
      - NEXT_PRIVATE_SMTP_TRANSPORT=${NEXT_PRIVATE_SMTP_TRANSPORT:?err}
      - NEXT_PRIVATE_SMTP_HOST=${NEXT_PRIVATE_SMTP_HOST}
      - NEXT_PRIVATE_SMTP_PORT=${NEXT_PRIVATE_SMTP_PORT}
      - NEXT_PRIVATE_SMTP_SECURE=${NEXT_PRIVATE_SMTP_SECURE:?err}
      - NEXT_PRIVATE_SMTP_USERNAME=${NEXT_PRIVATE_SMTP_USERNAME}
      - NEXT_PRIVATE_SMTP_PASSWORD=${NEXT_PRIVATE_SMTP_PASSWORD}
      - NEXT_PRIVATE_SMTP_FROM_NAME=${NEXT_PRIVATE_SMTP_FROM_NAME:?err}
      - NEXT_PRIVATE_SMTP_FROM_ADDRESS=${NEXT_PRIVATE_SMTP_FROM_ADDRESS:?err}
      - NEXT_PRIVATE_SIGNING_LOCAL_FILE_PATH=${NEXT_PRIVATE_SIGNING_LOCAL_FILE_PATH:-
/opt/documenso/cert.p12}
      - NEXT_PRIVATE_SIGNING_PASSPHRASE=${NEXT_PRIVATE_SIGNING_PASSPHRASE}
      - NEXT_PUBLIC_DISABLE_SIGNUP=${NEXT_PUBLIC_DISABLE_SIGNUP}
    ports:
      - ${PORT:-3000}:${PORT:-3000}
    volumes:
      - ./certs/cert.p12:/opt/documenso/cert.p12  # Zertifikate lokal speichern
    restart: always
```

Docker compose file mit caddy

```
version: '3.8'

services:
  database:
    image: postgres:15
    environment:
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
      - POSTGRES_DB=${POSTGRES_DB}
    healthcheck:
      test: ['CMD-SHELL', 'pg_isready -U ${POSTGRES_USER}']
      interval: 10s
      timeout: 5s
      retries: 5
    volumes:
      - ./data-db:/var/lib/postgresql/data  # Speichert Daten außerhalb des Containers
    restart: always

  documenso:
    image: documenso/documenso:latest
    depends_on:
      database:
        condition: service_healthy
    environment:
      - PORT=${PORT:-3000}
      - NEXTAUTH_URL=${NEXTAUTH_URL:-${NEXT_PUBLIC_WEBAPP_URL}
      - NEXTAUTH_SECRET=${NEXTAUTH_SECRET:?err}
      - NEXT_PRIVATE_ENCRYPTION_KEY=${NEXT_PRIVATE_ENCRYPTION_KEY:?err}
      - NEXT_PRIVATE_ENCRYPTION_SECONDARY_KEY=${NEXT_PRIVATE_ENCRYPTION_SECONDARY_KEY:?err}
      #wir wollen kein google.de
      #- NEXT_PRIVATE_GOOGLE_CLIENT_ID=${NEXT_PRIVATE_GOOGLE_CLIENT_ID}
      #- NEXT_PRIVATE_GOOGLE_CLIENT_SECRET=${NEXT_PRIVATE_GOOGLE_CLIENT_SECRET}
      - NEXT_PUBLIC_WEBAPP_URL=${NEXT_PUBLIC_WEBAPP_URL:?err}
      - NEXT_PRIVATE_INTERNAL_WEBAPP_URL=${NEXT_PRIVATE_INTERNAL_WEBAPP_URL:-
http://localhost:$PORT}
      - NEXT_PUBLIC_MARKETING_URL=${NEXT_PUBLIC_MARKETING_URL:-https://documenso.com}
      - NEXT_PRIVATE_DATABASE_URL=${NEXT_PRIVATE_DATABASE_URL:?err}
      - NEXT_PRIVATE_DIRECT_DATABASE_URL=${NEXT_PRIVATE_DIRECT_DATABASE_URL:-
```

```
${NEXT_PRIVATE_DATABASE_URL}
      - NEXT_PUBLIC_UPLOAD_TRANSPORT=${NEXT_PUBLIC_UPLOAD_TRANSPORT:-database}
      #Diese Variablen werden nur benötigt wenn die Daten nicht in einer Datenbank sondern in einem S3
      #Bucket gespeichert werden sollen
      #- NEXT_PRIVATE_UPLOAD_ENDPOINT=${NEXT_PRIVATE_UPLOAD_ENDPOINT}
      #- NEXT_PRIVATE_UPLOAD_FORCE_PATH_STYLE=${NEXT_PRIVATE_UPLOAD_FORCE_PATH_STYLE}
      #- NEXT_PRIVATE_UPLOAD_REGION=${NEXT_PRIVATE_UPLOAD_REGION}
      #- NEXT_PRIVATE_UPLOAD_BUCKET=${NEXT_PRIVATE_UPLOAD_BUCKET}
      #- NEXT_PRIVATE_UPLOAD_ACCESS_KEY_ID=${NEXT_PRIVATE_UPLOAD_ACCESS_KEY_ID}
      #- NEXT_PRIVATE_UPLOAD_SECRET_ACCESS_KEY=${NEXT_PRIVATE_UPLOAD_SECRET_ACCESS_KEY}
      - NEXT_PRIVATE_SMTP_TRANSPORT=${NEXT_PRIVATE_SMTP_TRANSPORT:?err}
      - NEXT_PRIVATE_SMTP_HOST=${NEXT_PRIVATE_SMTP_HOST}
      - NEXT_PRIVATE_SMTP_PORT=${NEXT_PRIVATE_SMTP_PORT}
      - NEXT_PRIVATE_SMTP_SECURE=${NEXT_PRIVATE_SMTP_SECURE:?err}
      - NEXT_PRIVATE_SMTP_USERNAME=${NEXT_PRIVATE_SMTP_USERNAME}
      - NEXT_PRIVATE_SMTP_PASSWORD=${NEXT_PRIVATE_SMTP_PASSWORD}
      - NEXT_PRIVATE_SMTP_FROM_NAME=${NEXT_PRIVATE_SMTP_FROM_NAME:?err}
      - NEXT_PRIVATE_SMTP_FROM_ADDRESS=${NEXT_PRIVATE_SMTP_FROM_ADDRESS:?err}
      - NEXT_PRIVATE_SIGNING_LOCAL_FILE_PATH=${NEXT_PRIVATE_SIGNING_LOCAL_FILE_PATH:-
/opt/documenso/cert.p12}
      - NEXT_PRIVATE_SIGNING_PASSPHRASE=${NEXT_PRIVATE_SIGNING_PASSPHRASE}
      - NEXT_PUBLIC_DISABLE_SIGNUP=${NEXT_PUBLIC_DISABLE_SIGNUP}
    ports:
      - ${PORT:-3000}:${PORT:-3000}
    volumes:
      - ./certs/cert.p12:/opt/documenso/cert.p12  # Zertifikate lokal speichern
    restart: always


  caddy:
      image: caddy:latest
      container_name: caddy-reverse-proxy
      ports:
        - "80:80"
        - "443:443"
      volumes:
        - ./Caddyfile:/etc/caddy/Caddyfile
        - ./caddy_data:/data
        - ./caddy_config:/config
      restart: always
```

Zertifikat anlegen

Privaten 2048-Bit-RSA-Schlüssel erstellen

```
openssl genrsa -out /root/documenso/cert.key 2048
```

Erstelle eine Zertifikatsignierungsanforderung (CSR)

```
openssl req -new -key /root/documenso/cert.key -out /root/documenso/cert.csr
```

Beantworte die Fragen (du kannst auch Platzhalter verwenden):

- **Country Name**: Zwei-Buchstaben-Ländercode (z. B. `DE`)
- **State or Province**: Bundesland oder Region
- **Locality**: Stadt
- **Organization Name**: Dein Unternehmens- oder Projektname
- **Organizational Unit Name**: Abteilung (falls nicht zutreffend, leer lassen)
- **Common Name**: Die Domain oder der Name (z. B. `localhost` oder `example.com`)
- **Email Address**: Deine E-Mail-Adresse
- **A chellenge Passwort: leer lassen #**Dieses Passwort muss dann auch das PK12 Passwort werden
- **An aoptional company name :** Enter (leer lassen, keins setzten)

Erstelle ein selbstsigniertes Zertifikat mit 100 Jahren Gültigkeit

```
openssl x509 -req -days 36500 -in /root/documenso/cert.csr -signkey /root/documenso/cert.key -out /root/documenso/cert.crt
```

Konvertiere das Zertifikat in ein PKCS#12-Format, setzte deinen Namen, hier heißt es Documenso Certificat

```
openssl pkcs12 -export -out /root/documenso/cert.p12 -inkey /root/documenso/cert.key -in /root/documenso/cert.crt -legacy -name "Documenso Certificate"
```

Du wirst aufgefordert, ein Passwort für die `.p12`-Datei festzulegen. Dieses Passwort musst du in der `.env`-Variable `NEXT_PRIVATE_SIGNING_PASSPHRASE` hinterlegen.

Nun das Zertifikat überprüfen

```
openssl x509 -in /root/documenso/cert.crt -text -noout
```

Das Zertifikat in certs kopieren

```
cp /root/documenso/cert.p12 /root/documenso/certs/cert.p12
```

rechte setzten

```
chown 1001 /root/documenso/certs/cert.p12
chmod 644 ./certs/cert.p12
```

die Conatiner starten

```
cd /root/documenso/
docker-compose up -d
```

Website aufrufen:

jednachdem http oder https :

http(s)://example.com oder ip:3000

PG Admin installieren (optional)

Wenn gewünscht kann man noch die Datenbankverwaltung dazu installieren

WIr fügen in die Docker compose einen weiteren Container hinzu und vergeben dann ein sicheres Kennwort:

```
pgadmin:
    image: dpage/pgadmin4:latest
    container_name: pgadmin
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@admin.com
      PGADMIN_DEFAULT_PASSWORD: supersichereskennwort
    ports:
      - "5050:80"
```

Nun kann via port 5050 pg admin aufgerufen werden.

Adminrechte vergeben

Beschreibung:

Von Haus aus, ist jeder neu registrierte Benutzer ein Benutzer.
Um einem Benutzer Admin rechte zu geben, müssen wir den Datenbankeintrag des Benutzers ändern

Rechte anpassen:

Ins Projekt Verzeichnis wechseln

```
cd /root/documenso
```

Im docker conatiner wo der Datenbankserver läuft anmelden

```
docker-compose exec <postgres container id> sh

Beispiel

docker-compose exec database sh
```

Nun an der Datenbank mit Datenbankname und Benutzername anmelden

-d Datenbankname
-u Benutzername

```
psql -d documenso -U docuuser
```

Nun den Benutzer Admin rechte geben. Als Identifier gillt die E-Mailadresse.

Also user@email durch die Benutzeremailadresse ersetzten.

```
UPDATE "User" SET roles = ARRAY_REMOVE(roles, 'USER') || ARRAY['ADMIN']::"Role"[] WHERE email = 'user@email';
```

Ausgabe:

```
documenso=# UPDATE "User" SET roles = ARRAY_REMOVE(roles, 'USER') || ARRAY['ADMIN']::"Role"[] WHERE
email = 'user@email';
UPDATE 1
```

Nun mit exit exit wieder raus.
Auf der Weboberfläche, haben wir jetzt Adminrechte