

Documenso

Lien : <https://docs.documenso.com/>

Architecture réalisée

```
Documenso
├── MinIO (S3 API)
│   ├── stockage temporaire local
│   └── Proxy SharePoint (Python)
│       ├── Microsoft Graph (AppOnly, cert)
│       └── SharePoint Online (Documents d'équipe)
│           ├── Site
│           └── Documents
│               ├── Dossiers = Buckets
│               └── Fichiers = Objets
```

Self-hosting avec Docker compose

Lien : <https://docs.documenso.com/docs/self-hosting/deployment/docker-compose>

Prérequis

- Docker 20.10 or later
- Docker Compose v2.0 or later
- SMTP credentials for sending emails
- At least 2GB of available RAM
- A domain name (for production deployments)

Installation des prérequis

- conteneur LXC : 2 Gio RAM ; 2 coeurs ; DD de 20 Gio
- modifier le fichier `/etc/apt/sources.list.d/debian.sources` pour avoir ce contenu (<http://security.debian.org> trixie-security remplacé par <http://deb.debian.org/debian-security>)

```
Types: deb
URIs: http://deb.debian.org/debian-security
Suites: trixie-security
Components: contrib main
Signed-By: /usr/share/keyrings/debian-archive-keyring.gpg
```

```
Types: deb
URIs: http://deb.debian.org/debian
Suites: trixie trixie-updates
Components: contrib main
Signed-By: /usr/share/keyrings/debian-archive-keyring.gpg
```

- ajouter les dépôts

```
# Add Docker's official GPG key:
apt update
apt install ca-certificates curl
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc
chmod a+r /etc/apt/keyrings/docker.asc
```

```
# Add the repository to Apt sources:
tee /etc/apt/sources.list.d/docker.sources <<EOF
Types: deb
URIs: https://download.docker.com/linux/debian
Suites: $(. /etc/os-release && echo "$VERSION_CODENAME")
Components: stable
```

```
Signed-By: /etc/apt/keyrings/docker.asc
EOF
```

- mettre à jour

```
apt update && apt upgrade -y
```

- installer Docker

```
apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

- Vérifier l'installation:

```
docker --version
docker compose version
```

Cloner and Configurer

- installer git

```
apt install git -y
```

Cloner le dépôt

```
git clone https://github.com/documenso/documenso.git
cd documenso/docker/production
```

Générer les secrets

```
# Generate NEXTAUTH_SECRET
echo "NEXTAUTH_SECRET=$(openssl rand -base64 32)"
# Generate encryption keys
echo "NEXT_PRIVATE_ENCRYPTION_KEY=$(openssl rand -base64 32)"
echo "NEXT_PRIVATE_ENCRYPTION_SECONDARY_KEY=$(openssl rand -base64 32)"
# Generate database password
echo "POSTGRES_PASSWORD=$(openssl rand -base64 24)"
```

Définir un accès Admin

Tous les comptes créés sont de simple utilisateur.

Pour donner les droits admin à un compte existant, il faut modifier directement dans la base de données le rôle du compte choisi :

```
docker compose -f compose.yml exec database \
  psql -U documenso -d documenso -c "UPDATE \"User\" SET roles = '{ADMIN}' WHERE email =
  'your@email.com';"
```

Utiliser minIO

Prérequis :

- MinIO en fonctionnement (conteneur Docker)
- Un bucket dédié : documenso. Le proxy sharepoint-point fera le mappage vers les dossiers pédagogiques (administration, bts-sio, etc.)
- Une clé d'accès + secret
- L'API S3 activée (par défaut)

Le bucket doit être créé dans minio sinon Documenso ne le voit pas. Documenso ne découvre pas les buckets automatiquement :

```
mc mb minio/documenso
```

Variables d'environnement Documenso

Documenso utilise le SDK S3 standard.

```
STORAGE_PROVIDER=s3

S3_ENDPOINT=http://minio:9000
S3_REGION=us-east-1
S3_BUCKET_NAME=documenso

S3_ACCESS_KEY_ID=compteadmin
S3_SECRET_ACCESS_KEY=motdepasse

S3_FORCE_PATH_STYLE=true
S3_USE_SSL=false
```

Créer une application Entra ID autorisée à mettre à jour le site Sharepoint

- application à créer dans Entra ID : **minio-sharepoint** ;
- création d'une équipe Teams **Signatures numériques** ;
- configurer l'accès en modification de l'application **minio-sharepoint** au site Sharepoint **Signatures numériques**

Lien : [Configurer une application enregistrée dans Entra ID pour gérer une équipe Sharepoint](#)

Docker compose

```
name: documenso-production

services:
  database:
    image: postgres:15
    env_file:
      - .env
    healthcheck:
      test: ['CMD-SHELL', 'pg_isready -U ${POSTGRES_USER}']
      interval: 10s
      timeout: 5s
      retries: 5
    volumes:
      - ./data/postgres:/var/lib/postgresql/data

  documenso:
    image: documenso/documenso:latest
    depends_on:
      database:
        condition: service_healthy
    env_file:
      - .env
    ports:
      - ${PORT:-3000}:${PORT:-3000}
    volumes:
      - /opt/documenso/cert.p12:/opt/documenso/cert.p12:ro

  minio:
    image: minio/minio:latest
    container_name: minio
    command: server /data --console-address ":9001"
    env_file:
      - .env
    volumes:
      - minio-data:/data
    ports:
      - "9000:9000"
      - "9001:9001"

  sharepoint-proxy:
```

```
build: ./sharepoint-proxy
container_name: sharepoint-proxy
volumes:
  - ./certs:/certs:ro
working_dir: /app
env_file:
  - .env
depends_on:
  - minio
ports:
  - "8080:8080"
volumes:
  database:
```

Créer le fichier d'environnement

Créer le fichier d'environnement **.env** dans le même dossier que celui qui contient **compose.yml** (documenso/docker/production)

```
touch .env
```

- contenu du fichier .env

```
# Database (used by both database and documenso services)
POSTGRES_USER=documenso
POSTGRES_PASSWORD=your-secure-database-password
POSTGRES_DB=documenso

# Application secrets (generate with: openssl rand -base64 32)
NEXTAUTH_SECRET=your-nextauth-secret
NEXT_PRIVATE_ENCRYPTION_KEY=your-encryption-key-min-32-characters
NEXT_PRIVATE_ENCRYPTION_SECONDARY_KEY=your-secondary-key-min-32-characters

# Public URL where Documenso is accessible
NEXT_PUBLIC_WEBAPP_URL=https://sign.example.com
NEXT_PRIVATE_INTERNAL_WEBAPP_URL=http://localhost:3000

# Database connection (uses Docker service name)
NEXT_PRIVATE_DATABASE_URL=postgresql://documenso:your-secure-database-password@database:5432/documenso

# Email configuration pour Microsoft 365 / outlook
# Transport SMTP simple vers ton Postfix du réseau qui est relais SMTP
NEXT_PRIVATE_SMTP_TRANSPORT="smtp-auth"
NEXT_PRIVATE_SMTP_HOST="relaissmtp.domaine.local"
NEXT_PRIVATE_SMTP_PORT="25"

# Pas de TLS entre serveurs internes
NEXT_PRIVATE_SMTP_SECURE="false"
NEXT_PRIVATE_SMTP_UNSAFE_IGNORE_TLS="true"

# Pas d'auth car le relais SMTP Postfix du réseau n'en demande pas
NEXT_PRIVATE_SMTP_USERNAME=""
NEXT_PRIVATE_SMTP_PASSWORD=""

# Adresse FROM (celle utilisée par le script Graph API)
NEXT_PRIVATE_SMTP_FROM_ADDRESS="your-email@yourdomain.com"
NEXT_PRIVATE_SMTP_FROM_NAME="Documenso"

# configuration minio
MINIO_ROOT_USER="compteadmmin"
MINIO_ROOT_PASSWORD="motdepasse"

# configuration stockage S3 pour documenso
NEXT_PUBLIC_UPLOAD_TRANSPORT=s3
NEXT_PRIVATE_UPLOAD_ENDPOINT=http://minio:9000
NEXT_PRIVATE_UPLOAD_FORCE_PATH_STYLE=true
NEXT_PRIVATE_UPLOAD_REGION=us-east-1
NEXT_PRIVATE_UPLOAD_BUCKET=lycee
NEXT_PRIVATE_UPLOAD_ACCESS_KEY_ID="compte"
NEXT_PRIVATE_UPLOAD_SECRET_ACCESS_KEY="secret"
```

```
# configuration stockage S3 pour minio
STORAGE_PROVIDER=s3

S3_ENDPOINT=http://minio:9000
S3_REGION=us-east-1
S3_BUCKET_NAME=lycee

S3_ACCESS_KEY_ID=compteadmin
S3_SECRET_ACCESS_KEY=motdepasse

S3_FORCE_PATH_STYLE=true
S3_USE_SSL=false

# configuration de sharepoint-proxy
MINIO_ROOT_USER=admin
MINIO_ROOT_PASSWORD=motdepasse

TENANT_ID=<ID du tenant>
CLIENT_ID=<ID de l'application dans Entra ID>
SITE_PATH=tenant.sharepoint.com:/sites/site
CERT_PATH=/certs/minio-sharepoint.pem
KEY_PATH=/certs/minio-sharepoint.key

AUTHORITY=https://login.microsoftonline.com/<ID du tenant>
GRAPH_SCOPE=https://graph.microsoft.com/.default

WEBHOOK_SECRET="super-secret-minio"

DOCUMENT_LIBRARY="Documents"
```

Commentaires :

- S3FORCEPATHSTYLE=true ⇒ OBLIGATOIRE avec MinIO * S3USESSL=false ⇒ car MinIO est en HTTP interne * S3REGION ⇒ peut être n'importe quoi (MinIO n'en tins pas compte)

Utiliser un certificat signé

Un certificat signé est requis pour signer les documents. Généré un certificat autosigné .p12 sur le serveur et le monter dans le conteneur.

Générer une clé privée RSA 2048 bit

```
openssl genrsa -out private.key 2048
```

Créer un certificat autosigné pour une année

```
openssl req -new -x509 -key private.key -out certificate.crt -days 365
```

Au prompt, renseigner les informations de l'organisation :

Champ	Valeur
Country Name	FR
State or Province	France
Locality Name	Limoges
Organization Name	Lycée Suzanne Valadon
Organizational	Administration
Common Name	Lycée Suzanne Valadon
Email Address	0870019y.ac-limoges.fr

Créer le certificat .p12

Création du certificat .p12 et création du fichier PKCS#12 (.p12) contenant ce certificat et la clé privée. Un mot de passe doit être renseigné pour protéger le fichier .p12

```
openssl pkcs12 -export -out certificate.p12 -inkey private.key -in certificate.crt
```

- Pour saisir le mot de passe de manière non interactive :

```
# Set password securely (won't appear in command history)
read -s -p "Enter certificate password: " CERT_PASS
echo
openssl pkcs12 -export -out certificate.p12 -inkey private.key -in certificate.crt \
  -password env:CERT_PASS
```

Clean up

Supprimer les fichiers intermédiaires:

```
rm private.key certificate.crt
```

- Gardez certificate.p12 and le mot de passe

Vérifier le certificat

```
# Check certificate details
openssl pkcs12 -in certificate.p12 -info -nokeys
# Verify password works
openssl pkcs12 -in certificate.p12 -noout
```

Placer le certificat signé sur l'hôte

Placer le certificat sur l'hôte et définir les permissions pour que le conteneur puisse le lire (UID 1001):

```
mkdir -p /opt/documenso
cp /root/documenso/docker/production/certificate.p12 /opt/documenso/cert.p12
chown 1001:1001 /opt/documenso/cert.p12
chmod 400 /opt/documenso/cert.p12
```

Le compose.yml va monter ce chemin dans le conteneur. Ajoutez la passphrase au fichier **.env** :

```
NEXT_PRIVATE_SIGNING_PASSPHRASE=your-certificate-password
```

If file mounting is not available, you can set `NEXTPRIVATESIGNINGLOCALFILE_CONTENTS` with the base64-encoded certificate string instead.

Démarrer les services

```
docker compose --env-file .env up -d
```

- vérifier que les conteneurs s'exécutent:

```
docker compose ps
```

- sorties attendues :

NAME	STATUS	PORTS
documenso-production-database-1	running (healthy)	5432/tcp
documenso-production-documenso-1	running	0.0.0.0:3000->3000/tcp

Wait for the database to be healthy and for migrations to complete. Check the logs:

```
docker compose logs -f documenso
```

Lancer automatiquement le Docker Compose Documenso au démarrage de ta VM, avec systemd

créer un service systemd

- docker-compose.yml dans **/opt/documenso/**
- Créer le fichier pour le service systemd :

```
nano /etc/systemd/system/documenso.service
```

- contenu du fichier documenso.service

```
[Unit]
Description=Documenso Docker Compose Service
Requires=docker.service
After=docker.service

[Service]
Type=oneshot
WorkingDirectory=/opt/documenso
ExecStart=/usr/bin/docker compose up -d
ExecStop=/usr/bin/docker compose down
RemainAfterExit=yes
TimeoutStartSec=0

[Install]
WantedBy=multi-user.target
```

- Recharger systemd

```
systemctl daemon-reload
```

- Activer le service au démarrage

```
systemctl enable documenso.service
```

- Démarrer immédiatement

```
systemctl start documenso.service
```

- Vérifier

```
systemctl status documenso.service
```

From:
[/ - Les cours du BTS SIO](#)

Permanent link:
[/doku.php/systeme/documenso/autohebergement](#)

Last update: **2026/05/12 22:05**

