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)"
```

Utiliser minIO

Prérequis :

- MinIO en fonctionnement (conteneur Docker)
- Un bucket dédié (ex : lycee)
- Une clé d’accès + secret
- L’API S3 activée (par défaut)

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=admin
S3_SECRET_ACCESS_KEY=Orpheus87

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 numéritiques** ;
- configurer l'accès en modification de l'application **minio-sharepoint** au site Sharepoint **Signatures numéritiques**

Lien : Configurer une application enregistrée dans Entra ID pour gérer une équipe Sharepoint

Docker compose

```
name: documenso-production

services:
  database:
    image: postgres:15
    environment:
      - POSTGRES_USER=${POSTGRES_USER:?err}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD:?err}
      - POSTGRES_DB=${POSTGRES_DB:?err}
    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
    environment:
      - PORT=${PORT:-3000}
      - NEXTAUTH_SECRET=${NEXTAUTH_SECRET:?err}
      - NEXT_PRIVATE_ENCRYPTION_KEY=${NEXT_PRIVATE_ENCRYPTION_KEY:?err}
      - NEXT_PRIVATE_ENCRYPTION_SECONDARY_KEY=${NEXT_PRIVATE_ENCRYPTION_SECONDARY_KEY:?err}
      - 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:$PORT}
      - NEXT_PRIVATE_DATABASE_URL=${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}
      - 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=${NEXT_PRIVATE_SMTP_PORT}
      - NEXT_PRIVATE_SMTP_USERNAME=${NEXT_PRIVATE_SMTP_USERNAME}
      - NEXT_PRIVATE_SMTP_PASSWORD=${NEXT_PRIVATE_SMTP_PASSWORD}
      - NEXT_PRIVATE_SMTP_APIKEY_USER=${NEXT_PRIVATE_SMTP_APIKEY_USER}
      - NEXT_PRIVATE_SMTP_APIKEY=${NEXT_PRIVATE_SMTP_APIKEY}
      - NEXT_PRIVATE_SMTP_SECURE=${NEXT_PRIVATE_SMTP_SECURE}
      - NEXT_PRIVATE_SMTP_APIKEY=${NEXT_PRIVATE_SMTP_APIKEY}
      - NEXT_PRIVATE_SMTP_SECURE=${NEXT_PRIVATE_SMTP_SECURE}
      - NEXT_PRIVATE_SMTP_UNSAFE_IGNORE_TLS=${NEXT_PRIVATE_SMTP_UNSAFE_IGNORE_TLS}
      - NEXT_PRIVATE_SMTP_FROM_NAME=${NEXT_PRIVATE_SMTP_FROM_NAME:?err}
      - NEXT_PRIVATE_SMTP_FROM_ADDRESS=${NEXT_PRIVATE_SMTP_FROM_ADDRESS:?err}
      - NEXT_PRIVATE_SMTP_SERVICE=${NEXT_PRIVATE_SMTP_SERVICE}
      - NEXT_PRIVATE_RESEND_API_KEY=${NEXT_PRIVATE_RESEND_API_KEY}
      - NEXT_PRIVATE_MAILCHANNELS_API_KEY=${NEXT_PRIVATE_MAILCHANNELS_API_KEY}
      - NEXT_PRIVATE_MAILCHANNELS_ENDPOINT=${NEXT_PRIVATE_MAILCHANNELS_ENDPOINT}
```

```
      - NEXT_PRIVATE_MAILCHANNELS_DKIM_DOMAIN=${NEXT_PRIVATE_MAILCHANNELS_DKIM_DOMAIN}
      - NEXT_PRIVATE_MAILCHANNELS_DKIM_SELECTOR=${NEXT_PRIVATE_MAILCHANNELS_DKIM_SELECTOR}
      - NEXT_PRIVATE_MAILCHANNELS_DKIM_PRIVATE_KEY=${NEXT_PRIVATE_MAILCHANNELS_DKIM_PRIVATE_KEY}
      - NEXT_PUBLIC_DOCUMENT_SIZE_UPLOAD_LIMIT=${NEXT_PUBLIC_DOCUMENT_SIZE_UPLOAD_LIMIT}
      - NEXT_PUBLIC_POSTHOG_KEY=${NEXT_PUBLIC_POSTHOG_KEY}
      - NEXT_PUBLIC_DISABLE_SIGNUP=${NEXT_PUBLIC_DISABLE_SIGNUP}
      - NEXT_PRIVATE_ALLOWED_SIGNUP_DOMAINS=${NEXT_PRIVATE_ALLOWED_SIGNUP_DOMAINS}
      - NEXT_PRIVATE_SIGNING_LOCAL_FILE_PATH=${NEXT_PRIVATE_SIGNING_LOCAL_FILE_PATH:-
/opt/documenso/cert.p12}
      - NEXT_PRIVATE_SIGNING_PASSPHRASE=${NEXT_PRIVATE_SIGNING_PASSPHRASE}
      - NEXT_PUBLIC_USE_INTERNAL_URL_BROWSERLESS=${NEXT_PUBLIC_USE_INTERNAL_URL_BROWSERLESS}
    ports:
      - ${PORT:-3000}:${PORT:-3000}
    volumes:
      - /opt/documenso/cert.p12:/opt/documenso/cert.p12:ro

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"
```

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 NEXT*PRIVATE*SIGNING*LOCAL*FILE_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
```