Technical Installation Manual

Team : CacheME

Project : Secure File Sharing Platform

Version 2

Table of Contents

Overview

The Secure Share platform is an end-to-end encrypted (E2EE) file sharing system built with modern web

technologies. The platform consists of:

- **Frontend**: Next.js application with React
- **Backend**: Go-based file service and Node.js API
- **Key Management**: Python Flask service with HashiCorp Vault
- **Database**: Supabase (PostgresSQL) and PostgreSQL with PgAdmin
- **File Storage**: OwnCloud
- **Authentication**: Google OAuth and custom JWT

Prerequisites

Before beginning the installation, ensure you have administrative access to your system and stable

internet connectivity. You will also need accounts for the following external services:

- **Supabase**: For user management and authentication database
- **Google Cloud Console**: For OAuth authentication
- **Cloudinary**: For image/avatar upload functionality

Required Software Versions

- **Node.js**: v18.20.5
- **Python**: 3.10.12
- **Go**: v1.24.5
- **Docker**: Latest stable version
- **Git**: Latest stable version

Software Installation

1. **Node.js installation**

   Download and install Node.js v18.20.5 from nodejs.org

   Verification:

```
node --version
npm --version
```

2. **Python Installation**

   Download and install Python 3.10.12 from python.org

   Verification:

```
python --version
pip --version
```

3. **Go installation**

   Download and install Go from golang.org

   Verification:

```
go version
```

4. **Docker installation**

   Download and install Docker from docker.com

   Verification:

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

Repository Setup

Cloning the Repository

1. Open your terminal/command prompt
2. Navigate to your desired project directory
3. Clone the repository;

```
gh repo clone COS301-SE-2025/Secure-File-Sharing-Platform
cd Secure-File-Sharing-Platform
```

Frontend Installation

1. Navigate to Frontend Directory

```
cd sfsp-ui
```

2. Install Dependencies

```
npm install
```

3. Create a .env file in the sfsp-ui directory with the following variables:

```
# Supabase Configuration
NEXT_PUBLIC_SUPABASE_URL=your_supabase_url
NEXT_PUBLIC_SUPABASE_ANON_KEY=your_supabase_anon_key

# Cloudinary Configuration
NEXT_PUBLIC_CLOUDINARY_CLOUD_NAME=your_cloud_name
NEXT_PUBLIC_CLOUDINARY_API_KEY=your_api_key
CLOUDINARY_API_SECRET=your_api_secret
NEXT_PUBLIC_CLOUDINARY_UPLOAD_PRESET=avatar

# Google OAuth Configuration
GOOGLE_CLIENT_ID=your_google_client_id
GOOGLE_CLIENT_SECRET=your_google_client_secret
NEXT_PUBLIC_GOOGLE_CLIENT_ID=your_google_client_id
NEXTAUTH_URL=http://localhost:3000
NEXTAUTH_SECRET=your_nextauth_secret

# API Configuration
NEXT_PUBLIC_API_URL=http://localhost:5000

# JWT Configuration
JWT_SECRET=your_jwt_secret

# Email Configuration
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your_email
SMTP_PASS=your_app_password
SMTP_FROM=your_email
FROM_NAME=SecureShare
```

Note: Replace all the placeholder values with your actual service credentials

Backend Installation

1. **Main API Service**

   Navigate to the backend directory:

```
cd ../sfsp-api
```

   Create a .env file in the sfsp-api directory:

```
# Supabase Configuration
SUPABASE_URL=your_supabase_url
SUPABASE_ANON_KEY=your_supabase_anon_key

# JWT Configuration
JWT_SECRET=your_jwt_secret
JWT_EXPIRES_IN=1h
PORT=5000

# PostgreSQL Configuration
POSTGRES_URI=postgres://admin:admin@localhost:5432/file_service_db?sslmode=disable
POSTGRES_USER=admin
POSTGRES_PASSWORD=admin

# PgAdmin Configuration
PGADMIN_DEFAULT_EMAIL=admin@example.com
PGADMIN_DEFAULT_PASSWORD=your_pgadmin_password

# AES Encryption
AES_KEY=your_32_character_aes_key

# OwnCloud Configuration
OWNCLOUD_URL=http://localhost:8080/remote.php/webdav/
OWNCLOUD_USERNAME=admin
OWNCLOUD_PASSWORD=admin

# Email Configuration
EMAIL_USER=your_email
EMAIL_PASS=your_app_password
EMAIL_RECEIVER=your_email

# SMTP Configuration
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your_email
SMTP_PASS=your_app_password

# Sender Information
FROM_NAME=SecureShare
FROM_EMAIL=your_email

# Vault Configuration
VAULT_URL=http://localhost:8200
VAULT_TOKEN=your_vault_token

# Flask Configuration
FLASK_PORT=8443
FLASK_DEBUG=False
FLASK_ENV=development
```

2. Key Service Setup

Navigate to the key service directory:

```
cd services/keyservice
```

Install Python Dependencies:

```
pip install -r requirements.txt
pip install hvac
```

3. File Service Setup

Navigate to the file service directory:

```
cd ../fileService
```

Create .env file in the fileService directory:

```
# MongoDB Configuration
MONGO_URI=your_mongodb_connection_string

# PostgreSQL Configuration
POSTGRES_URI=postgres://admin:admin@localhost:5432/file_service_db?sslmode=disable
POSTGRES_PASSWORD=admin
POSTGRES_USER=admin

# PGAdmin Configuration
PGADMIN_DEFAULT_EMAIL=admin@example.com
PGADMIN_DEFAULT_PASSWORD=your_pgadmin_password

# AES Encryption
AES_KEY=your_32_character_aes_key

# OwnCloud Configuration
OWNCLOUD_URL=http://localhost:8080/remote.php/webdav/
OWNCLOUD_USERNAME=admin
OWNCLOUD_PASSWORD=admin
```

Database Setup

---

1. **PostgreSQL Setup**

   Create a new directory for PostgreSQL:

```
mkdir postgres-setup
cd postgres-setup
```

   Create docker-compose.yml

```yaml
version: '3.8'

services:
  postgres:
    image: postgres:latest
    container_name: postgres_container
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: admin
      POSTGRES_DB: file_service_db
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./schema.sql:/docker-entrypoint-initdb.d/schema.sql

  postgres-admin:
    image: dpage/pgadmin4:latest
    container_name: pgadmin_container
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@example.com
      PGADMIN_DEFAULT_PASSWORD: admin
    ports:
      - "5050:80"
    depends_on:
      - postgres

volumes:
  postgres_data:
```

Create schema.sql in the same directory:

postgres > schema.sql

```sql
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE TABLE IF NOT EXISTS users (
    id UUID PRIMARY KEY
);
CREATE TABLE IF NOT EXISTS files (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(), -- Unique identifier for the file
    owner_id UUID NOT NULL REFERENCES users(id),
    file_name TEXT NOT NULL,
    file_type TEXT,
    file_size BIGINT,
    cid TEXT, -- Storage path or IPFS CID
    nonce TEXT,
    description TEXT,
    tags TEXT[], -- Optional: store tags as a Postgres array
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    file_hash TEXT,
    allow_view_sharing BOOLEAN DEFAULT FALSE
);
CREATE TABLE IF NOT EXISTS sent_files (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    sender_id UUID NOT NULL REFERENCES users(id),
    recipient_id UUID NOT NULL REFERENCES users(id),
    file_id UUID NOT NULL REFERENCES files(id) ON DELETE CASCADE,
    encrypted_file_key TEXT,
    x3dh_ephemeral_pubkey TEXT,
    sent_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
CREATE TABLE IF NOT EXISTS received_files (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    recipient_id UUID NOT NULL REFERENCES users(id),
    sender_id UUID NOT NULL REFERENCES users(id),
    file_id UUID NOT NULL REFERENCES files(id) ON DELETE CASCADE,
    received_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    accepted BOOLEAN DEFAULT FALSE,
    expires_at TIMESTAMP,
    metadata JSONB
);
CREATE TABLE IF NOT EXISTS access_logs (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    file_id UUID NOT NULL REFERENCES files(id) ON DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES users(id),
    action TEXT NOT NULL, -- for "viewed", "downloaded", "deleted" but UI can trigger this with any action
    message TEXT,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    view_only BOOLEAN DEFAULT FALSE
);
CREATE TABLE IF NOT EXISTS notifications (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    type TEXT NOT NULL,
    "from" UUID NOT NULL REFERENCES users(id),
    "to" UUID NOT NULL REFERENCES users(id),
    file_name TEXT NOT NULL,
    file_id UUID NOT NULL REFERENCES files(id) ON DELETE CASCADE,
    received_file_id UUID REFERENCES received_files(id),
    message TEXT,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status TEXT NOT NULL CHECK (status IN ('pending', 'accepted', 'declined')),
    read BOOLEAN DEFAULT FALSE
);

CREATE TABLE IF NOT EXISTS shared_files_view (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    sender_id UUID NOT NULL REFERENCES users(id),
    recipient_id UUID NOT NULL REFERENCES users(id),
    file_id UUID NOT NULL REFERENCES files(id) ON DELETE CASCADE,
    metadata JSONB NOT NULL,
    shared_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP,
    revoked BOOLEAN DEFAULT FALSE,
    revoked_at TIMESTAMP,
    access_granted BOOLEAN DEFAULT TRUE,
    -- index for fatser lookups
    UNIQUE(sender_id, recipient_id, file_id)
);

CREATE INDEX IF NOT EXISTS idx_shared_files_view_recipient ON shared_files_view(recipient_id, revoked);
CREATE INDEX IF NOT EXISTS idx_shared_files_view_sender ON shared_files_view(sender_id, file_id);
```

2. Supabase Schema

Execute the following SQL in your Supabase SQL editor:

```sql
create table public.users (
  id uuid not null default gen_random_uuid (),
  email text not null,
  password text null,
  "resetPasswordPIN" text null,
  ik_public text null,
  username text null,
  "resetPINExpiry" timestamp with time zone null,
  spk_public text null,
  opks_public text null,
  nonce text null,
  "signedPrekeySignature" text null,
  salt text null,
  notification_settings jsonb null default '{"news": {"newFeatures": true, "feedbackSurveys": true, "secureShareTips": false}, "files": {"sharedFolderActivity": true}, "alerts": {"newAppConnected": true,
  "newDeviceLinked": true, "deleteLargeFiles": true, "newBrowserSignIn": true, "runningOutOfSpace": true}}'::jsonb,
  avatar_url text null,
  google_id text null,
  is_verified boolean null,
  constraint users_pkey primary key (id),
  constraint unique_email unique (email),
  constraint users_google_id_key unique (google_id)
) TABLESPACE pg_default;
```

Open in SQL Editor

```sql
create table public.verification_codes (
  id uuid not null default gen_random_uuid (),
  user_id uuid not null,
  code character varying(6) not null,
  type character varying(20) not null,
  expires_at timestamp with time zone not null,
  used boolean null default false,
  created_at timestamp with time zone null default now(),
  constraint verification_codes_pkey primary key (id),
  constraint verification_codes_user_id_fkey foreign KEY (user_id) references users (id) on delete CASCADE
) TABLESPACE pg_default;

create index IF not exists idx_verification_codes_user_code on public.verification_codes using btree (user_id, code) TABLESPACE pg_default;

create index IF not exists idx_verification_codes_expires on public.verification_codes using btree (expires_at) TABLESPACE pg_default;
```

External Service Setup

1. **OwnCloud Setup**
2. **In your project root directory, create compose.yml:**

```
Run All Services
services:
  Run Service
  owncloud:
    image: owncloud/server:10.13
    restart: always
    ports:
      - 8080:8080
    environment:
      OWNCLOUD_DOMAIN: localhost
      ADMIN_USERNAME: admin
      ADMIN_PASSWORD: admin
      OWNCLOUD_DB_TYPE: sqlite
    volumes:
      - owncloud_files:/mnt/data

volumes:
  owncloud_files:
```

3. **HashiCorp Vault Setup**

   Create vault configuration directory:

```
mkdir -p vault/config
```

   Create vault/config/vault.hcl:

```
storage "file" {
  path = "/vault/file"
}

listener "tcp" {
  address     = "0.0.0.0:8200"
  tls_disable = true
}

disable_mlock = true    "mlock": Unknown word.
ui = true
```

Update your compose.yml to include Vault:

```yaml
Run Service
services:
Run Service
vault:
  image: hashicorp/vault:latest
  container_name: vault-server
  ports:
    - "8200:8200"
  environment:
    VAULT_ADDR: http://0.0.0.0:8200
  cap_add:
    - IPC_LOCK
  command: >
    vault server -config=/vault/config/vault.hcl
  volumes:
    - vault_data:/vault/file
    - ./vault/config:/vault/config

volumes:
  vault_data:
    driver: local
```

Vault Initialization:

1. Start the vault service: docker compose up -d
2. Access the Vault UI at http://localhost:8200
3. Initialize with 5 key shares and 3 key threshold
4. Download the generated keys and store them securely
5. Use the root token in your environment variables

Deployment and Running

1. **Start External Services**
   Start all Docker services:

```
# In project root
docker compose up -d
# In postgres-setup directory (if separate)
cd postgres-setup
docker compose up -d
```

2. **Start Backend Services**
   Main API Services:

```
cd sfsp-api
npm start
```

   Key Service:

```
cd sfsp-api/services/keyservice
python3 app.py
```

   File Service:

```
cd sfsp-api/services/fileService
go run main.go
```

3. **Start Frontend Service**

```
cd sfsp-ui
npm run dev
```

4. **Verify Services**
   Check that all services are running:

- **Frontend**: http://localhost:3000
- **Main API**: http://localhost:5000
- **Key Service**: http://localhost:8443
- **File Service**: http://localhost:8080 (Go service port)
- **OwnCloud**: http://localhost:8080 (WebDAV)
- **Vault UI**: http://localhost:8200
- **PostgreSQL**: http://localhost:5432
- **PGAdmin**: http://localhost:5050

Using the secure file sharing platform

To see how to use the system, you can refer to the user manual.

Support

For additional support or issues not covered in this manual:

1. Check the project repository issues
2. Review service documentation for external dependencies
3. Contact the development team

Document Version: 2

Last Updated: September 2025

Team: CacheMe

Platform: Secure Share E2EE File Sharing Platform