1. Ente Photos

- **E2EE = YES**
- **Authentication:** Password, 2FA
- **Admin Metadata Access:** Access to social graph i.e who you share albums with
- **Website:** ente.com
- **Repository:** Github
- **Installation Documentation:** Docker Deployment
- **Installation Videos**: Jim's Garage
- **Community:**
- **Donations:**

1. Overview

Ente is one of these applications that is just a must have! It is not only E2EE but it also has amazing sharing capabilities between friends and family. Due to it's E2EE nature you can actually host all your friends and family on your server without them worrying that you as the admin has access to their private media. This is why this personal media organizer is my favorite and out performs Immich and Photoprism. The E2EE part is so important when it comes to private pictures that for immich and photoprism it only makes sense if you host it yourself and not for your spouse, kids any other friends.

2. Installation

Source: [Ente Docker Installation]

If you don't have enough resources you can install Ente on your DockerVM. However, if you do, then I would recommend to make a dedicated VM for your precious personal media, especially when it will also include media from your friends and family. But yeah, regardless the installation will be the same if you will use docker. Now, if you use Dockhand and create a new VM specifically for personal media then you should create another dockhand instance for that. Yes you can

1. So the first thing we will do is download all the files we need s.t we can then copy paste them in Dockhand. For this enter your VM and copy paste the following in the folder where Dockhand has access to.

```
git clone https://github.com/ente/ente
cd ente/server/config
```

2. Perfect, so now you are in the config file which contains the compose file, environment variables and the museum file. This is basically where the magic happens. So to make them effective we need to change there names as follows

```
# Inside the cloned repository's directory (usually `ente`)
cd server/config
cp example.env .env
cp example.yaml museum.yaml
```

3. Perfect now, we will start editing first the **.env** file. which at the moment of writing looks like this

.env

```
# Copy this file to .env in the same directory in which this file is present
# This file contains the environment variables needed for Ente's cluster to run properly.
# This file is split based on services declared in Docker Compose file
#
# Service: Postgres
# This is used for storing data in database pertaining to collections, files, users,
subscriptions, etc.
# These credentials are needed for accessing the database via Museum.
# Please set a strong password for accessing the database.
```

```
# Enter these values in museum.yaml file under `db`.
# This need not be defined if using external DB (i. e. no Compose service for PostgreSQL
is used)
POSTGRES_USER=pguser
POSTGRES_PASSWORD=<password>
POSTGRES_DB=ente_db

# Service: MinIO
# This is used for MinIO object storage service that's shipped by default with the compose
file
# to reduce need for an external S3-compatible bucket for quick testing.
# It is recommended to use an external bucket for long-term usage.
# The credentials required for accessing the object storage is documented below.
# It is not needed to configure these variables if you are using an external bucket
# Enter the user value into key and password in secret for buckets in museum.yaml under
`s3` section.
MINIO_ROOT_USER=minio-user
MINIO_ROOT_PASSWORD=password

# Service: Web
# Replace this with the API endpoint for Museum.
ENTE_API_ORIGIN=http://localhost:8080
```

4. They basically want you to create the following credentials for **Postgres** and for **Minio**

POSTGRES_USER	pguser
POSTGRES_PASSWORD	password
POSTGRES_DB	ente_db
MINIO_ROOT_USER	minio-user
MINIO_ROOT_PASSWORD	password
ENTE_API_ORIGIN	https://api-ente.example.org

5. Now, to make it easy you can generate some random passwords for **Postgres** and **MinIo** with the following commands

```
openssl rand -base64 32  # For Postgres Password
openssl rand -base64 32  # For MinIo Password
```

6. Then copy and paste these for the **Postgres** and **MinIo** passwords. Also the username you can adjust as well to make your instance even more secure as well.

7. Finally, the **ENTE_API_ORIGIN** value should be a publicly accessible domain which will be used as your connection point when you launch Ente applications. This is both for the App, Desktop and Web app. If your base domain is example.org you could make it simple and use api-ente.example.org for instance. Note! This is not the URL you use in your browser to access your Ente instance! We will set that value later.

8. Next we will enter the compose file and do a couple of small adjustments. Basically, we want to add the postgres and minio data volumes to be added in the same ente directory. I simply prefer to keep all data under neath one folder. To do this just add ./ in front of the volumes and remove the volumes at the bottom. Then it looks something like this

compose.yml

```
services:
  museum:
    build:
      context: ..
    ports:
      - 8080:8080 # Museum/API
    depends_on:
      postgres:
        condition: service_healthy
    env_file:
      - .env
    volumes:
      - ./museum.yaml:/museum.yaml:ro
      - ./data:/data:ro

  # Resolve "localhost:3200" in the museum container to the minio container.
  socat:
    image: alpine/socat
    network_mode: service:museum
    depends_on: [museum]
    command: "TCP-LISTEN:3200,fork,reuseaddr TCP:minio:3200"
```

```
  postgres:
    image: postgres:15-trixie
    env_file:
      - .env
    # Wait for postgres to accept connections before starting museum.
    healthcheck:
      test: pg_isready -q -d ${POSTGRES_DB} -U ${POSTGRES_USER}
      start_period: 30s
      start_interval: 1s
    volumes:
      - ./postgres-data:/var/lib/postgresql/data

  minio:
    image: minio/minio
    ports:
      - 3200:3200 # MinIO API
      # - 3201:3201 # MinIO Console (uncomment to access externally)
    env_file:
      - .env
    command: server /data --address ":3200" --console-address ":3201"
    volumes:
      - ./minio-data:/data
    post_start:
      - command: |
          sh -c '
          #!/bin/sh

          while ! mc alias set h0 http://minio:3200 ${MINIO_ROOT_USER}
${MINIO_ROOT_PASSWORD} 2>/dev/null
          do
            echo "Waiting for minio..."
            sleep 0.5
          done

          cd /data

          mc mb -p b2-eu-cen
          mc mb -p wasabi-eu-central-2-v3
          mc mb -p scw-eu-fr-v3
```

```
          '


  ente-web:
    build:
      context: ../..
      dockerfile: web/Dockerfile
    ports:
    ports:
      - 3000:3000 # Photos web app
      - 3001:3001 # Accounts
      - 3002:3002 # Public albums
      - 3003:3003 # Auth
      - 3004:3004 # Cast
      - 3005:3005 # Share
      - 3006:3006 # Embed
      - 3008:3008
      - 3009:3009
      - 3010:3010 # Memories
    # Modify these values to your custom subdomains, if using any
    environment:
      - NODE_ENV=development
    env_file:
      - .env

#volumes:
#  postgres-data:
#  minio-data:
```

9. Now, the minio-data folder will be the object storage where all the media is living. It is up to you if this will be local or external via an NFS share. Now, if you already have an minio-data object storage running somewhere then you can also connect to this as well. However, for this basic setup we assume that you don't have this setup yet and will generate a new MiniO object storage container. In the future I would like to add a RustFS container instead since MinIO stopped being open source in 2026 unfortunately.

10. Now, that is all settled what is left is to configure the **museum.yaml** file, which looks like this

museum.yaml

```
# Copy this file to museum.yaml in the same directory in which this file is present

# This section is meant for configuration of database.
# Museum uses these values and credentials for connecting
# to the database.
# Set a strong password and if using PostgreSQL Docker container
# provided in Docker Compose file, ensure db.password is same
# as POSTGRES_PASSWORD
# Similarly ensure db.user and db.name are same as POSTGRES_USER and
# POSTGRES_DB respectively
db:
    host: postgres
    port: 5432
    name: ente_db
    user: pguser
    password: <password>

# This section is for configuring storage buckets. Omit this section if
# you only intend to use Ente Auth
s3:
    # Change this to false if enabling SSL
    are_local_buckets: true
    # Only path-style URL works if disabling are_local_buckets with MinIO
    use_path_style_urls: true
    b2-eu-cen:
      # Uncomment the below configuration to override the top-level configuration
      # are_local_buckets: true
      # use_path_style_urls: true
      key: <key>
      secret: <secret>
      endpoint: localhost:3200
      region: eu-central-2
      bucket: b2-eu-cen
    wasabi-eu-central-2-v3:
      # are_local_buckets: true
      # use_path_style_urls: true
      key: <key>
```

```
      secret: <secret>
      endpoint: localhost:3200
      region: eu-central-2
      bucket: wasabi-eu-central-2-v3
      compliance: false
    scw-eu-fr-v3:
      # are_local_buckets: true
      # use_path_style_urls: true
      key: <key>
      secret: <secret>
      endpoint: localhost:3200
      region: eu-central-2
      bucket: scw-eu-fr-v3

# Specify the base endpoints for various web apps
apps:
    # If you're running a self hosted instance and wish to serve public links,
    # set this to the URL where your albums web app is running.
    public-albums: http://localhost:3002
    # If you're running a self hosted instance and wish to serve public links
    # for locker, set this to the URL where your share web app is running.
    public-locker: http://localhost:3005
    # If you're running a self hosted instance and wish to serve Ente Paste,
    # set this to the URL where your paste web app is running.
    public-paste: http://localhost:3008
    # If you're running a self hosted instance and wish to serve public memory
    # shares, set this to the URL where your memories web app is running.
    public-memories: http://localhost:3010
    cast: http://localhost:3004
    embed-albums: http://localhost:3006
    # Set this to the URL where your accounts web app is running, primarily used for
    # passkey based 2FA.
    accounts: http://localhost:3001
    # Set this to the URL where your legacy recovery web app is running.
    legacy: http://localhost:3011

# Key used for encrypting customer emails before storing them in DB
#
# To make it easy to get started, some randomly generated (but fixed) values are
```

```
# provided here. But if you're really going to be using museum, please generate
# new keys. You can use `go run tools/gen-random-keys/main.go` for that.
#
# Replace values in key and JWT for security
key:
    encryption: yvmG/RnzKrbCb9L3mgsmoxXr9H7i2Z4qlbT0mL3ln4w=
    hash:
KXYiG07wC7GIgvCSdg+WmyWdXDAn6XKYJtp/wkEU7x573+byBRAYtpTP0wwvi8i/4l37uicX1dVTUzwH3sLZyw==
# JWT secrets
jwt:
    secret: i2DecQmfGreG6q1vBj5tCokhlN41gcfS2cjOs9Po-u8=
```

11. Now, the first thing we need to do is change the **encryption**, **hash** and **secret** key. Just as with the postgres and minio passwords we will again generate them with

```
openssl rand -base64 32  # For encryption key
openssl rand -base64 64  # For hash key
openssl rand -base64 32  # For JWT secret
```

12. Then just replace them in the museum.yaml file for security reasons.

13. Perfect, now that's done we are almost there, we only need to add the domain names for all the different applications. Also since this app will be publicly accessible I would host the application on Cloudflare DNS. In this way you can use their proxy to prevent you from DDOS attacks and what not. Okey so as of writing you should create the following domain names and replace the http://localhost:port with your domain name. Don't forget to change HTTP ---> HTTPS !!!

Domain Name	Port
Your Ente Instance: https://ente.example.org	3000
api: https://api-ente.example.org	8080
public-albums: https://albums.example.org	3002
public-locker: https://locker.example.org	3005
public-paste: https://paste.example.org	3008
public-memories: https://memories.example.org	3010
cast: https://cast.example.org	3004
embed-albums: https://embed.example.org	3006

accounts: https://accounts.example.org	3001
legacy: https://legacy.example.org	3011
space: https://space.rfeyn.org	3013

14. Now, to make this working you need to have your reverse proxy working of course. If you use OPNsense as your router with Caddy then it will be fairly simple to do this. Just go to Caddy and add all the domain names pointing to the IP address of the VM you are hosting on and to the right port ;)

Storage

At this moment Ente uses MinIO S3 block storage for storing all the media. Now, this might change in the future but that is what it is today. The reason I say his is because MinIO stopped being open source since it Archived it's repository in 2026. Now, Ente offers you the option to store everything in an external MinIO bucket if you have one running or you can create a new one if you don't have a MinIO S3 pool running. I guess most people don't have an object storage running at this moment so we will walk you through setting one up.

```
# Copy this file to museum.yaml in the same directory in which this file is present

# This section is meant for configuration of database.
# Museum uses these values and credentials for connecting
# to the database.
# Set a strong password and if using PostgreSQL Docker container
# provided in Docker Compose file, ensure db.password is same
# as POSTGRES_PASSWORD
# Similarly ensure db.user and db.name are same as POSTGRES_USER and
# POSTGRES_DB respectively
db:
    host: postgres
    port: 5432
    name: ente_db
    user: pguser
    password: jSZxfR5+WSp4oOhHFev+jzs1Dg+b


# This section is for configuring storage buckets. Omit this section if
```

```
# you only intend to use Ente Auth
s3:
    # Change this to false if enabling SSL
    are_local_buckets: true
    # Only path-style URL works if disabling are_local_buckets with MinIO
    use_path_style_urls: true
    b2-eu-cen:
      # Uncomment the below configuration to override the top-level configuration
      # are_local_buckets: true
      # use_path_style_urls: true
      key: <key>
      secret: <secret>
      endpoint: localhost:3200
      region: eu-central-2
      bucket: b2-eu-cen
    wasabi-eu-central-2-v3:
      # are_local_buckets: true
      # use_path_style_urls: true
      key: <key>
      secret: <secret>
      endpoint: localhost:3200
      region: eu-central-2
      bucket: wasabi-eu-central-2-v3
      compliance: false
    scw-eu-fr-v3:
      # are_local_buckets: true
      # use_path_style_urls: true
      key: <key>
      secret: <secret>
      endpoint: localhost:3200
      region: eu-central-2
      bucket: scw-eu-fr-v3

# Specify the base endpoints for various web apps
apps:
    # If you're running a self hosted instance and wish to serve public links,
    # set this to the URL where your albums web app is running.
    public-albums: https://albums.rfeyn.org
    # If you're running a self hosted instance and wish to serve public links
    # for locker, set this to the URL where your share web app is running.
```

```
    public-locker: https://locker.rfeyn.org
    # If you're running a self hosted instance and wish to serve Ente Paste,
    # set this to the URL where your paste web app is running.
    public-paste: https://public.rfeyn.org
    # If you're running a self hosted instance and wish to serve public memory
    # shares, set this to the URL where your memories web app is running.
    public-memories: https://memories.rfeyn.org
    cast: https://cast.rfeyn.org
    embed-albums: https://embed.rfeyn.org
    # Set this to the URL where your accounts web app is running, primarily used for
    # passkey based 2FA.
    accounts: https://accounts.rfeyn.org
    # Set this to the URL where your legacy recovery web app is running.
    legacy: https://legacy.rfeyn.org


# Key used for encrypting customer emails before storing them in DB
#
# To make it easy to get started, some randomly generated (but fixed) values are
# provided here. But if you're really going to be using museum, please generate
# new keys. You can use `go run tools/gen-random-keys/main.go` for that.
#
# Replace values in key and JWT for security
key:
      encryption: d4DU5WiP7LafNtVU4YXmDq0PQ5mpT6cZuox84yE4W9E=
      hash:
W+fyKVK2tBqkm5HZnMp6eITMYXyAgLFyhDkSnuEz7UmaDayD/SBKZf+1iUlKEuOy3bRfM2PBuByeO2UKhZsOGg==

jwt:
      secret: 9DiDZPPKId57tXbtPnCGEl_Tuc0LbNMkpv32DPTA8tQ=
```

MINIO_ROOT_USER=minio-user-PwZZDZ4b
MINIO_ROOT_PASSWORD=yRpTHbJRKP0kAb1mBIkMfirDQTaZ

# Service: Web
# Replace this with the API endpoint for Museum.
ENTE_API_ORIGIN=http://localhost:8080

4. Now, what do they request from you here

5. You should consider generating values for JWT and encryption keys for emails if you intend to use for long-term needs. You can do by running the following command inside /ente/server folder, assuming you cloned the repository to the ente folder.

6. We will now configure the museum.yaml file even though the example file is named as example.yaml which can be found in the ente/server/config folder. Just nano into it and it should give you this

5. So what do we need to adjust in the **museum.yaml** file ? Well we need to make sure that we provide the correct URL's for the different apps that ente is providing. My recommendation today is to use one domain name and in front of it place the name of the app.

In order to run the cluster, you will have to provide environment variable values. Copy the configuration files for modification by the following command inside `server/config` directory of the repository. This allows you to modify configuration without having to face hassle while pulling in latest changes

```
# Copy this file to museum.yaml in the same directory in which this file is present


# This section is meant for configuration of database.
# Museum uses these values and credentials for connecting
# to the database.
```

```
# Set a strong password and if using PostgreSQL Docker container
# provided in Docker Compose file, ensure db.password is same
# as POSTGRES_PASSWORD
# Similarly ensure db.user and db.name are same as POSTGRES_USER and
# POSTGRES_DB respectively
db:
    host: postgres
    port: 5432
    name: ente_db
    user: pguser
    password: <password>

# This section is for configuring storage buckets. Omit this section if
# you only intend to use Ente Auth
s3:
    # Change this to false if enabling SSL
    are_local_buckets: true
    # Only path-style URL works if disabling are_local_buckets with MinIO
    use_path_style_urls: true
    b2-eu-cen:
      # Uncomment the below configuration to override the top-level configuration
      # are_local_buckets: true
      # use_path_style_urls: true
      key: <key>
      secret: <secret>
      endpoint: localhost:3200
      region: eu-central-2
      bucket: b2-eu-cen
    wasabi-eu-central-2-v3:
      # are_local_buckets: true
      # use_path_style_urls: true
      key: <key>
      secret: <secret>
      endpoint: localhost:3200
      region: eu-central-2
      bucket: wasabi-eu-central-2-v3
      compliance: false
    scw-eu-fr-v3:
      # are_local_buckets: true
      # use_path_style_urls: true
```

```
      key: <key>
      secret: <secret>
      endpoint: localhost:3200
      region: eu-central-2
      bucket: scw-eu-fr-v3

# Specify the base endpoints for various web apps
apps:
    # If you're running a self hosted instance and wish to serve public links,
    # set this to the URL where your albums web app is running.
    public-albums: https://albums.rfeyn.org
    # If you're running a self hosted instance and wish to serve public links
    # for locker, set this to the URL where your share web app is running.
    public-locker: https://locker.rfeyn.org
    # If you're running a self hosted instance and wish to serve Ente Paste,
    # set this to the URL where your paste web app is running.
    public-paste: https://paste.rfeyn.org
    # If you're running a self hosted instance and wish to serve public memory
    # shares, set this to the URL where your memories web app is running.
    public-memories: https://memories.rfeyn.org
    cast: https://cast.rfeyn.org
    embed-albums: https://embed.rfeyn.org
    # Set this to the URL where your accounts web app is running, primarily used for
    # passkey based 2FA.
    accounts: https://accounts.rfeyn.org
    # Set this to the URL where your legacy recovery web app is running.
    legacy: https://legacy.rfeyn.org


# Key used for encrypting customer emails before storing them in DB
#
# To make it easy to get started, some randomly generated (but fixed) values are
# provided here. But if you're really going to be using museum, please generate
# new keys. You can use `go run tools/gen-random-keys/main.go` for that.
#
# Replace values in key and JWT for security
key:
    encryption: yvmG/RnzKrbCb9L3mgsmoxXr9H7i2Z4qlbT0mL3ln4w=
    hash:
KXYiG07wC7GIgvCSdg+WmyWdXDAn6XKYJtp/wkEU7x573+byBRAYtpTP0wwvi8i/4l37uicX1dVTUzwH3sLZyw==
```

```
# JWT secrets
jwt:
    secret: i2DecQmfGreG6q1vBj5tCokhlN41gcfS2cjOs9Po-u8=
```

```
services:
  museum:
    build:
      context: ..
    ports:
      - 8080:8080 # Museum/API
    depends_on:
      postgres:
        condition: service_healthy
    env_file:
      - .env
    volumes:
      - ./museum.yaml:/museum.yaml:ro
      - ./data:/data:ro

  # Resolve "localhost:3200" in the museum container to the minio container.
  socat:
    image: alpine/socat
    network_mode: service:museum
    depends_on: [museum]
    command: "TCP-LISTEN:3200,fork,reuseaddr TCP:minio:3200"

  postgres:
    image: postgres:15-trixie
    env_file:
      - .env
    # Wait for postgres to accept connections before starting museum.
    healthcheck:
      test: pg_isready -q -d ${POSTGRES_DB} -U ${POSTGRES_USER}
      start_period: 30s
      start_interval: 1s
    volumes:
      - postgres-data:/var/lib/postgresql/data

  minio:
    image: minio/minio
```

```
    ports:
      - 3200:3200 # MinIO API
      # - 3201:3201 # MinIO Console (uncomment to access externally)
    env_file:
      - .env
    command: server /data --address ":3200" --console-address ":3201"
    volumes:
      - minio-data:/data
    post_start:
      - command: |
          sh -c '
          #!/bin/sh

          while ! mc alias set h0 http://minio:3200 ${MINIO_ROOT_USER} ${MINIO_ROOT_PASSWORD}
2>/dev/null
          do
            echo "Waiting for minio..."
            sleep 0.5
          done

          cd /data

          mc mb -p b2-eu-cen
          mc mb -p wasabi-eu-central-2-v3
          mc mb -p scw-eu-fr-v3
          '

  ente-web:
    build:
      context: ../..
      dockerfile: web/Dockerfile
    ports:
    ports:
      - 3000:3000 # Photos web app
      - 3001:3001 # Accounts
      - 3002:3002 # Public albums
      - 3003:3003 # Auth
      - 3004:3004 # Cast
      - 3005:3005 # Share
      - 3006:3006 # Embed
```

```
      - 3008:3008
      - 3009:3009
      - 3010:3010 # Memories
    # Modify these values to your custom subdomains, if using any
    environment:
      - NODE_ENV=development
    env_file:
      - .env

volumes:
  postgres-data:
  minio-data:
```

3. Major Ente Update

MINI_ROOT_PASSWORD: yRpTHbJRKP0kAb1mBIkMfirDQTaZ

MINIO_SERVER_URL: https://minio-ente.rfeyn.org

4. Ente Legacy

Life is unpredictable. Legacy helps someone you trust recover your Ente account if you lose access, are incapacitated, or are no longer around.

There are two Legacy options in Ente Locker:

- **Trusted contacts**: Choose another Ente user who can recover your account after a waiting period.
- **Legacy Kits**: Create 3 physical recovery sheets. Any 2 sheets can start account recovery in a browser, and the helper does not need an Ente account.

Both options are account-level recovery tools. They are not item sharing, collection sharing, or direct login.