EspoCRM

CRM open source completo

- Instalação e Configuração
  - EspoCRM with Docker
- EspoCRM with N8N integration

Instalação e Configuração

EspoCRM with Docker

Link: https://docs.espocrm.com/administration/docker/installation/#docker-composedemo: https://demo.us.espocrm.com/
Editado em 22/08/2026

In this article:

- Installing with Docker Run
- Installing with Docker Compose
- Installing with Traefik
- Installing with Caddy
- Upgrading
- Maintenance
- Running a shell
- Docker Secrets
- Environments
- Config Environments
- Image Variants
- Troubleshooting

Installing

You can deploy EspoCRM using either Docker Run or Docker Compose. For production environments, we recommend Docker Compose, as it is easier to manage and provides better configuration control. While Docker Run is quicker to set up, it is a good choice if you want to try the system out.

Docker Run

Docker Run allows you to launch EspoCRM containers directly using command-line commands. Each container is created and managed individually. You can run the setup commands as a single

operation or step by step, depending on your preference.

See the Maintenance and Upgrading sections for operational guidance.

Quick start

```
docker network create espocrm-network && \
docker volume create espocrm-db && \
docker volume create espocrm-data && \
docker volume create espocrm-custom && \
docker volume create espocrm-custom-client && \
docker run \
  --name espocrm-db \
  --network espocrm-network \
  --restart unless-stopped \
  -e MARIADB_DATABASE=espocrm \
  -e MARIADB_USER=espocrm \
  -e MARIADB_PASSWORD=your_database_password \
  -e MARIADB_ROOT_PASSWORD=your_root_password \
  -v espocrm-db:/var/lib/mysql \
  -d mariadb && \
docker run \
  --name espocrm \
  --network espocrm-network \
  --restart unless-stopped \
  -e ESPOCRM_DATABASE_PASSWORD=your_database_password \
  -e ESPOCRM_ADMIN_USERNAME=admin \
  -e ESPOCRM_ADMIN_PASSWORD=your_admin_password \
  -v espocrm-data:/var/www/html/data \
  -v espocrm-custom:/var/www/html/custom \
  -v espocrm-custom-client:/var/www/html/client/custom \
  -p 8080:80 \
  -d espocrm/espocrm && \
docker run \
  --name espocrm-daemon \
  --network espocrm-network \
  --restart unless-stopped \
  --volumes-from espocrm \
  --entrypoint docker-daemon.sh \
  -d espocrm/espocrm
```

Then, access it via `http://localhost:8080` or `http://YOUR_IP_ADDRESS:8080` with credentials `admin` and `your_admin_password`.

Step by step

If you prefer to run the commands individually rather than using the quick start approach above, follow the steps below.

1. Create network

```
docker network create espocrm-network
```

2. Create volumes

```
docker volume create espocrm-db && \
docker volume create espocrm-data && \
docker volume create espocrm-custom && \
docker volume create espocrm-custom-client
```

3. Run MariaDB

```
docker run \
  --name espocrm-db \
  --network espocrm-network \
  --restart unless-stopped \
  -e MARIADB_DATABASE=espocrm \
  -e MARIADB_USER=espocrm \
  -e MARIADB_PASSWORD=your_database_password \
  -e MARIADB_ROOT_PASSWORD=your_root_password \
  -v espocrm-db:/var/lib/mysql \
  -d mariadb
```

4. Run EspoCRM

```
docker run \
  --name espocrm \
  --network espocrm-network \
  --restart unless-stopped \
  -e ESPOCRM_DATABASE_PASSWORD=your_database_password \
  -e ESPOCRM_ADMIN_USERNAME=admin \
  -e ESPOCRM_ADMIN_PASSWORD=your_admin_password \
  -v espocrm-data:/var/www/html/data \
```

```
  -v espocrm-custom:/var/www/html/custom \
  -v espocrm-custom-client:/var/www/html/client/custom \
  -p 8080:80 \
  -d espocrm/espocrm
```

5. Run daemon

```
docker run \
  --name espocrm-daemon \
  --network espocrm-network \
  --restart unless-stopped \
  --volumes-from espocrm \
  --entrypoint docker-daemon.sh \
  -d espocrm/espocrm
```

Then, access it via `http://localhost:8080` or `http://YOUR_IP_ADDRESS:8080` with credentials `admin` and `your_admin_password`.

Custom site URL

To set a custom IP address or domain, pass the `ESPOCRM_SITE_URL` environment variable when running the container.

```
docker run \
  --name espocrm \
  --network espocrm-network \
  -p 8080:80 \
  -e ESPOCRM_DATABASE_HOST=espocrm-db \
  -e ESPOCRM_DATABASE_USER=espocrm \
  -e ESPOCRM_DATABASE_PASSWORD=your_database_password \
  -e ESPOCRM_ADMIN_USERNAME=admin \
  -e ESPOCRM_ADMIN_PASSWORD=your_admin_password \
  -e ESPOCRM_SITE_URL=http://192.168.0.100:8080 \
  -d espocrm/espocrm:latest
```

Then, access it via `http://192.168.0.100:8080` with credentials `admin` and `your_admin_password`.

Reset

If you'd like to start over or delete your data, refer to the reset and cleanup section.

Docker Compose

Docker Compose simplifies container orchestration by defining all services in a single configuration file. It automatically manages networking, dependencies, and health checks between containers. This approach is ideal for production deployments as it's easier to version-control, scale, and maintain your entire setup.

See the Maintenance and Upgrading sections for operational guidance.

1. Create an empty directory.

```
mkdir espocrm-docker
```

2. Change into this directory.

```
cd espocrm-docker/
```

3. Create a `docker-compose.yml` file:

```
services:

  espocrm-db:
    image: mariadb:latest
    container_name: espocrm-db
    environment:
      MARIADB_ROOT_PASSWORD: your_root_password
      MARIADB_DATABASE: espocrm
      MARIADB_USER: espocrm
      MARIADB_PASSWORD: your_database_password
    volumes:
      - espocrm-db:/var/lib/mysql
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "healthcheck.sh", "--connect", "--innodb_initialized"]
      interval: 20s
      start_period: 10s
      timeout: 10s
      retries: 3

  espocrm:
    image: espocrm/espocrm:latest
```

```
    container_name: espocrm
    environment:
      ESPOCRM_DATABASE_HOST: espocrm-db
      ESPOCRM_DATABASE_USER: espocrm
      ESPOCRM_DATABASE_PASSWORD: your_database_password
      ESPOCRM_ADMIN_USERNAME: admin
      ESPOCRM_ADMIN_PASSWORD: your_admin_password
      ESPOCRM_SITE_URL: "http://localhost:8080"
    volumes:
      - espocrm-data:/var/www/html/data
      - espocrm-custom:/var/www/html/custom
      - espocrm-custom-client:/var/www/html/client/custom
    restart: unless-stopped
    depends_on:
      espocrm-db:
        condition: service_healthy
    healthcheck:
      test: ["CMD", "bin/command", "app-check"]
      start_period: 20s
      interval: 60s
      timeout: 20s
      retries: 3
    ports:
      - 8080:80

  espocrm-daemon:
    image: espocrm/espocrm:latest
    container_name: espocrm-daemon
    volumes_from:
      - espocrm
    restart: unless-stopped
    entrypoint: docker-daemon.sh
    depends_on:
      espocrm:
        condition: service_healthy
    healthcheck:
      test: ["CMD", "bin/command", "app-check"]
      start_period: 20s
      interval: 180s
      timeout: 20s
```

```
      retries: 3

  espocrm-websocket:
    image: espocrm/espocrm:latest
    container_name: espocrm-websocket
    environment:
      ESPOCRM_CONFIG_USE_WEB_SOCKET: "true"
      ESPOCRM_CONFIG_WEB_SOCKET_URL: "ws://localhost:8081"
      ESPOCRM_CONFIG_WEB_SOCKET_ZERO_M_Q_SUBSCRIBER_DSN: "tcp://*:7777"
      ESPOCRM_CONFIG_WEB_SOCKET_ZERO_M_Q_SUBMISSION_DSN: "tcp://espocrm-websocket:7777"
    volumes_from:
      - espocrm
    restart: unless-stopped
    entrypoint: docker-websocket.sh
    depends_on:
      espocrm:
        condition: service_healthy
    healthcheck:
      test: ["CMD", "bin/command", "app-check"]
      start_period: 20s
      interval: 180s
      timeout: 20s
      retries: 3
    ports:
      - 8081:8080

volumes:
  espocrm-db:
  espocrm-data:
  espocrm-custom:
  espocrm-custom-client:
```

More about *Installation Environments* you can find here.

4. Build EspoCRM project from directory.

```
docker compose up -d
```

5. Access it at `http://localhost:8080` or `http://YOUR_IP_ADDRESS:8080` with credentials `admin` and `your_admin_password`.

Reset

If you'd like to start over or delete your data, refer to the reset and cleanup section.

Traefik

You can read the instructions for installing EspoCRM in conjunction with Traefik in the Docker Compose environment here.

Caddy

You can read the instructions for installing EspoCRM in conjunction with Caddy in the Docker Compose environment here.

Upgrading

Keeping your EspoCRM installation updated ensures you have the latest features, security patches, and bug fixes. The upgrade process differs depending on whether you're using Docker Run or Docker Compose.

Docker Run

To upgrade EspoCRM when using Docker Run:

1. Pull the latest images:

```
docker pull mariadb
docker pull espocrm/espocrm
```

2. Stop all running containers:

```
docker stop espocrm espocrm-db espocrm-daemon
```

3. Remove the old containers:

```
docker rm espocrm espocrm-db espocrm-daemon
```

4. Run the containers again with the same configuration as your original deployment (using the quick start or step by step commands from the Docker Run section).

Docker Compose

To upgrade EspoCRM when using Docker Compose:

1. Navigate to your EspoCRM container directory.

2. Run the command:

```
docker compose pull && docker compose up -d
```

Within a few minutes the container will be upgraded to the latest version.

Maintenance

After deploying EspoCRM, regular maintenance is essential for optimal performance and troubleshooting. Choose the commands appropriate for your deployment method.

Docker Run

When using Docker Run, you can manage your containers individually using Docker commands.

Status

```
# Running containers
docker ps -a

# Resource usage
docker stats espocrm espocrm-db espocrm-daemon
```

View logs

```
# For a specific container
docker logs espocrm

# Follow logs in real-time
```

```
docker logs -f espocrm
```

Reset and cleanup

```
# Restart all containers
docker restart espocrm espocrm-db espocrm-daemon

# Stop all containers
docker stop espocrm espocrm-db espocrm-daemon

# Remove all containers
docker rm espocrm espocrm-db espocrm-daemon

# Remove network
docker network rm espocrm-network

# Remove volumes (all data will be lost)
docker volume rm espocrm-db espocrm-data espocrm-custom espocrm-custom-client
```

Docker Compose

Docker Compose provides streamlined commands to manage all services together.

Navigate to your EspoCRM container directory (where your `docker-compose.yml` file is located) before running any of the commands below.

Status

```
# Running containers
docker compose ps

# Resource usage
docker compose stats
```

View logs

```
# For all services
docker compose logs

# For a specific service
```

```
docker compose logs espocrm

# Follow logs in real-time
docker compose logs -f espocrm
```

Reset and cleanup

```
# Restart all containers
docker compose restart

# Stop and remove all containers (preserves volumes and data)
docker compose down

# Stop and remove all containers, networks, and volumes (all data will be lost)
docker compose down --volumes
```

Running a shell

In order to enter the container and view the files, make a rebuild, etc., use the following command (`espocrm` is your container name):

```
docker exec -it espocrm bash
```

Docker Secrets

To securely pass sensitive information, append `_FILE` to any supported environment variable. When this suffix is used, the variable's value is read from a file inside the container instead of being specified directly. This can be used with Docker secrets stored in `/run/secrets/<secret_name>` files. For example:

```
docker run \
  --name espocrm \
  --network espocrm-network \
  -e ESPOCRM_DATABASE_PASSWORD_FILE=/run/secrets/espocrm_db_password \
  -e ESPOCRM_ADMIN_PASSWORD_FILE=/run/secrets/espocrm_admin_password \
  -d espocrm/espocrm
```

Installation Environments

These are one-time environment variables used only for a fresh installation. If you need to define configuration options on container startup, see the Config Environments.

ESPOCRM_DATABASE_PLATFORM

Database platform. The possible values: `Mysql` or `Postgresql`. The default value is `Mysql`.

ESPOCRM_DATABASE_HOST

Database host name for EspoCRM. The default value is `espocrm-db`.

ESPOCRM_DATABASE_PORT

Database port for EspoCRM. The default value is empty.

ESPOCRM_DATABASE_NAME

Database name for EspoCRM. The default value is `espocrm`.

ESPOCRM_DATABASE_USER

Database user for EspoCRM. The default value is `espocrm`.

ESPOCRM_DATABASE_PASSWORD

Database password for EspoCRM. The default value is `password`.

ESPOCRM_ADMIN_USERNAME

User name for an administrator of EspoCRM. The default value is `admin`.

ESPOCRM_ADMIN_PASSWORD

User password for an administrator of EspoCRM. The default value is `password`.

ESPOCRM_SITE_URL

The URL of EspoCRM. This option is very important for normal operating of EspoCRM. Examples: `http://192.168.0.100:8080`, `http://my-crm.local`.

Other optional options

The list of possible values and their default values can be found in EspoCRM Administrator panel > Settings.

- `ESPOCRM_LANGUAGE`
- `ESPOCRM_DATE_FORMAT`
- `ESPOCRM_TIME_FORMAT`
- `ESPOCRM_TIME_ZONE`
- `ESPOCRM_WEEK_START`
- `ESPOCRM_DEFAULT_CURRENCY`
- `ESPOCRM_THOUSAND_SEPARATOR`
- `ESPOCRM_DECIMAL_MARK`

Config Environments

These environment variables are using to define configuration parameters of the EspoCRM every time on the container startup. The parameters that can be changed are defined in the `data/config.php` or `data/config-internal.php`.

Naming

Config environment variables should be converted from the camel-case format. For example: The `exportDisabled` config option should be converted to `ESPOCRM_CONFIG_EXPORT_DISABLED`.

Logger

There are additional options to change the `logger`:

- `ESPOCRM_CONFIG_LOGGER__LEVEL: "DEBUG"`
- `ESPOCRM_CONFIG_LOGGER__MAX_FILE_NUMBER: 30`
- `ESPOCRM_CONFIG_LOGGER__PATH: "data/logs/espo.log"`

For more details, visit documentation.

Allowed types:

String

```
ESPOCRM_CONFIG_WEB_SOCKET_URL: "wss://my-espocrm.com:8080"
```

Integer

```
ESPOCRM_CONFIG_EMAIL_MESSAGE_MAX_SIZE: 10
```

Boolean

```
ESPOCRM_CONFIG_USE_WEB_SOCKET: "true"
```

Null¶

```
ESPOCRM_CONFIG_CURRENCY_DECIMAL_PLACES: "null"
```

Image Variants

The `espocrm` images come in many flavors, each designed for a specific use case.

- `espocrm:apache`
- `espocrm:fpm`
- `espocrm:fpm-alpine`
- `espocrm:<version>`
- `espocrm:<version>-apache`
- `espocrm:<version>-fpm`
- `espocrm:<version>-fpm-alpine`

Troubleshooting

Incompatible customizations

In most cases, this issue is caused by installed extensions that are not compatible with the upgraded version.

1. Stop your services

Navigate to your EspoCRM directory and run:

```
docker compose down
```

2. Update your `docker-compose.yml`

Pin each service to a specific image version. Replace the image tags:

```
espocrm:
  image: espocrm/espocrm:latest
  # ....

espocrm-daemon:
  image: espocrm/espocrm:latest
  # ....

espocrm-websocket:
  image: espocrm/espocrm:latest
  # ....
```

With versioned tags:

```
espocrm:
  image: espocrm/espocrm:VERSION
  # ....

espocrm-daemon:
  image: espocrm/espocrm:VERSION
  # ....

espocrm-websocket:
  image: espocrm/espocrm:VERSION
  # ....
```

Replace `VERSION` with the EspoCRM version you were running before the upgrade (e.g. `9.3.0`, `9.3.0-fpm`). This lets you revert to that version and adjust your customizations accordingly.

3. Start your services

```
docker compose up -d
```

Migration to EspoCRM 10

To upgrade to EspoCRM v10.0 or later, you must perform a one-time manual migration to update your volume configuration. Starting from v10.0, mounting the entire `/var/www/html` directory is no longer required.

By following these instructions, your customizations and data will be preserved.

Warning

Before performing the following steps, be sure to make a backup.

Note

This applies only to users who mounted data to a local directory (e.g. `./espocrm:/var/www/html`). If you are using Docker volumes, see the instructions below.

1. Stop your services

Navigate to your EspoCRM directory and run:

```
docker compose down
```

2. Update your `docker-compose.yml`

Replace the existing volume mount:

```
espocrm:
  image: espocrm/espocrm
  container_name: espocrm
  # ....
  volumes:
    - ./espocrm:/var/www/html
  restart: unless-stopped
```

With the following targeted mounts:

```
espocrm:
  image: espocrm/espocrm
  container_name: espocrm
  # ....
  volumes:
    - ./espocrm/data:/var/www/html/data
    - ./espocrm/custom:/var/www/html/custom
    - ./espocrm/client/custom:/var/www/html/client/custom
```

```
  restart: unless-stopped
```

Repeat this for all EspoCRM service containers.

3. Remove obsolete files (optional)

This step is optional.

If your local `./espocrm` directory still contains image-provided files from previous versions, removing them is recommended to keep your mount clean. EspoCRM will still work if you keep them.

To remove them, run the following command:

```
rm -rf ./espocrm/application ./espocrm/vendor ./espocrm/bin ./espocrm/html ./espocrm/install
&& \
rm -f ./espocrm/bootstrap.php ./espocrm/clear_cache.php ./espocrm/command.php
./espocrm/cron.php ./espocrm/daemon.php ./espocrm/extension.php ./espocrm/index.php
./espocrm/preload.php ./espocrm/rebuild.php ./espocrm/upgrade.php ./espocrm/websocket.php
./espocrm/LICENSE.txt
```

4. Start your services

```
docker compose up -d
```

Migration to EspoCRM 10 with Docker volumes

To upgrade to EspoCRM v10.0 or later, you must perform a one-time manual migration to update your volume configuration. Starting from v10.0, mounting the entire `/var/www/html` directory is no longer required.

By following these instructions, your customizations and data will be preserved.

Warning

Before performing the following steps, be sure to make a backup.

Note

This applies only to users who mounted data to Docker volumes (e.g. `espocrm:/var/www/html`). If you are using a local directory, see the instructions above.

1. Stop your services

Navigate to your EspoCRM directory and run:

```
docker compose down
```

2. Create the required volumes

```
docker volume create espocrm-data; \
docker volume create espocrm-custom; \
docker volume create espocrm-custom-client
```

3. Migrate your data

```
docker compose run --rm \
  -v espocrm:/source:ro \
  -v espocrm-data:/dest-data \
  -v espocrm-custom:/dest-custom \
  -v espocrm-custom-client:/dest-client-custom \
  --entrypoint sh \
  espocrm -c "
    cp -a /source/data/. /dest-data/ &&
    cp -a /source/custom/. /dest-custom/ &&
    cp -a /source/client/custom/. /dest-client-custom/
  "
```

4. Update your `docker-compose.yml`

Replace the existing volume mount:

```
espocrm:
  image: espocrm/espocrm
  container_name: espocrm
  # ....
  volumes:
    - espocrm:/var/www/html
  restart: unless-stopped
```

```
# ...

volumes:
  espocrm-db:
  espocrm:
```

With the following targeted mounts:

```
espocrm:
  image: espocrm/espocrm
  container_name: espocrm
  # ....
  volumes:
    - espocrm-data:/var/www/html/data
    - espocrm-custom:/var/www/html/custom
    - espocrm-custom-client:/var/www/html/client/custom
  restart: unless-stopped

# ...

volumes:
  espocrm-db:
  espocrm-data:
    external: true
  espocrm-custom:
    external: true
  espocrm-custom-client:
    external: true
```

5. Remove obsolete files (optional)

This step is optional.

If the old `espocrm` Docker volume still contains image-provided files from previous versions, removing them is recommended to keep the volume clean. EspoCRM will still work if you keep them.

To remove them, run the following command:

```
docker run --rm \
  -v espocrm:/source \
```

```
  --entrypoint sh \
  espocrm -c "rm -rf /source/application /source/vendor /source/bin /source/html
/source/install && rm -f /source/bootstrap.php /source/clear_cache.php /source/command.php
/source/cron.php /source/daemon.php /source/extension.php /source/index.php
/source/preload.php /source/rebuild.php /source/upgrade.php /source/websocket.php
/source/LICENSE.txt"
```

6. Start your services

```
docker compose up -d
```

Undefined volume: invalid compose project

If you encounter the following error when starting your containers:

```
service "espocrm" refers to undefined volume espocrm/data: invalid compose project
```

Follow the Migration to EspoCRM 10 with Docker volumes instructions to resolve it.

Switching to MySQL 8.4

In MySQL 8.4 there were changes in the authentication procedure, so you may encounter authentication related errors while upgrading EspoCRM. In this case, it is recommended to take the following steps:

1. Change *authentication plugin* to `caching_sha2_password` for your MySQL users:

Notes:

- Replace the `YOUR_ROOT_PASSWORD` with your MySQL root password.
- Replace the `YOUR_ESPOCRM_DB_PASSWORD` with your MySQL espocrm user password.

```
sudo docker exec -i mysql mysql --user=root -p -e "
  ALTER USER IF EXISTS 'root'@'localhost' IDENTIFIED WITH caching_sha2_password BY
'YOUR_ROOT_PASSWORD';
  ALTER USER IF EXISTS 'root'@'%' IDENTIFIED WITH caching_sha2_password BY
'YOUR_ROOT_PASSWORD';
  ALTER USER IF EXISTS 'espocrm'@'localhost' IDENTIFIED WITH caching_sha2_password BY
```

```
'YOUR_ESPOCRM_DB_PASSWORD';
  ALTER USER IF EXISTS 'espocrm'@'%' IDENTIFIED WITH caching_sha2_password BY
'YOUR_ESPOCRM_DB_PASSWORD';"
```

2. Remove from *docker-compose.yml* file the following line: `command: --default-authentication-plugin=mysql_native_password`.

3. Restart and build `mysql` container:

```
sudo docker stop mysql && sudo docker rm mysql
docker compose up -d --build
```

EspoCRM with N8N integration

Link: https://github.com/traien/n8n-nodes-espocrm

Alt: https://www.npmjs.com/package/@traien/n8n-nodes-espocrm
Em 24/08/20206

@traien/n8n-nodes-espocrm

@traien/n8n-nodes-espocrm

This is a community node for n8n that enables seamless integration with EspoCRM, a powerful open-source CRM platform. With this node, you can automate your CRM operations and integrate EspoCRM with other services in your n8n workflows.

n8n is a fair-code licensed workflow automation platform.

Features

The EspoCRM node provides comprehensive access to EspoCRM's API functionality:

- **Entity Operations**
  - Create new records (Leads, Contacts, Accounts, etc.)
  - Retrieve entity details
  - Update records (full or partial updates)
  - Delete records
  - List/search entities with advanced filtering
  - Supported core entities: Account, Contact, Lead, Meeting, Task, Call, Opportunity, Case
- **Dynamic Entity Support**
  - Work with any entity type in your EspoCRM instance
  - Automatic field detection and validation
  - Custom field support

- **Advanced Filtering**
  - Complex search queries
  - Date-based filtering
  - Relationship filters
  - Array field operations
  - Custom boolean filters
- **Performance Optimizations**
  - Pagination support
  - Selective field loading
  - Skip total count for large datasets
  - Skip duplicate verification checks on record creation (using `X-Skip-Duplicate-Check` header)
- **AI Agent Tooling**
  - Dedicated `EspoCRM Tool` node that outputs an AI Tool connection for n8n Agents
  - Configure which entities and operations the agent may call plus safe pagination defaults
  - Agent requests are translated into authenticated EspoCRM REST calls automatically

Prerequisites

- Active EspoCRM instance (v6.0 or later recommended)
- API access enabled in EspoCRM
- API user with appropriate permissions
- n8n installation (v0.170.0 or later)

Installation

Follow these steps to install this package in your n8n instance:

1. Open your n8n instance
2. Go to Settings > Community Nodes
3. Select "Install"
4. Enter `@traien/n8n-nodes-espocrm`
5. Click "Install"

For a manual installation, you can use:

```
npm install @traien/n8n-nodes-espocrm
```

Or if you have installed n8n globally:

```
npm install -g @traien/n8n-nodes-espocrm
```

Configuration

API Authentication

1. In EspoCRM:
   - Navigate to Administration > API Users
   - Create a new API User
   - Generate an API Key
   - Note down the API Key and API Secret
2. In n8n:
   - Add new credentials of type 'EspoCRM API'
   - Enter your EspoCRM instance URL
   - Input the API Key and API Secret
   - Save the credentials

Usage Examples

Creating a Contact

Create a new contact record with customized field values:

1. Add an "EspoCRM" node to your workflow
2. Select "Contact" as the Resource
3. Choose "Create" as the Operation
4. Fill in the required fields:
   - First Name
   - Last Name
   - Email Address
5. Add any additional fields as needed
6. (Optional) Under **Options**, enable **Skip Duplicate Check** if you wish to bypass EspoCRM's duplicate verification rules
7. Connect to other nodes in your workflow

Creating a Meeting

1. Add an "EspoCRM" node to your workflow
2. Select `Meeting` as the Resource
3. Choose `Create` as the Operation
4. Fill in the required fields:

- `Name`
- `Start Date`

5. Optionally set `End Date`, `Status`, `Assigned User ID`, or parent (`parentType` + `parentId`)
6. Execute to create the meeting in EspoCRM

Working with Dynamic Entities

The Dynamic resource allows you to work with any entity type in your EspoCRM system:

1. Add an "EspoCRM" node to your workflow
2. Select "Dynamic" as the Resource
3. Choose your desired Operation (Create, Update, Get, etc.)
4. Select the Entity Type from the dropdown
5. The available fields will be automatically loaded based on the entity type
6. Complete the required fields and connect to your workflow

Advanced Filtering Example

```
{
  "operation": "getAll",
  "entityType": "Lead",
  "filterOptions": {
    "where": [
      {
        "type": "and",
        "value": [
          {
            "type": "equals",
            "field": "status",
            "value": "New"
          },
          {
            "type": "greaterThan",
            "field": "createdAt",
            "value": "2024-01-01"
          }
        ]
```

```
      }
    ],
    "orderBy": "createdAt",
    "order": "desc",
    "maxSize": 50
  }
}
```

Using EspoCRM as an AI Agent Tool

The package now ships with an `EspoCRM Tool` node that emits an AI Tool connection compatible with n8n's Agent/LLM nodes. Use it to let an agent autonomously read and modify EspoCRM data:

1. Drop the **EspoCRM Tool** node into the same workflow as your Agent node and select your Espo credentials.
2. Provide a concise tool description plus the entity types/operations you want to expose (e.g. `Account,Contact,Opportunity`).
3. Connect the node to the Agent's **Tools** input. At runtime the agent can call EspoCRM by sending structured parameters that map to the API.

The tool expects JSON with the following shape:

```
{
  "entityType": "Contact",
  "operation": "getAll",
  "filters": {
    "where": [
      { "type": "equals", "field": "accountId", "value": "ACCOUNT_ID" }
    ],
    "orderBy": "createdAt",
    "order": "desc"
  },
  "limit": 25
}
```

Supported operations are `get`, `getAll`, `create`, `update`, and `delete`. For single-record actions include `recordId`; for creates/updates add a `data` object with Espo field names. List requests inherit the default/max limits you configure on the node, and you can set `returnAll: true` when the agent needs the full dataset.

> ℹ When invoking the tool from an AI Agent, `data` and `filters` can be provided either as native JSON objects or as JSON strings (for example `"{\\"name\\":\\"Acme\\"}"`). The node automatically parses strings into objects before sending the EspoCRM REST call. You can also override the requested operation by including `operation` or `operations` in the payload (for example `{ "operation": "getAll" }` or `{ "operations": ["get", "update"] }`); requested values are validated against the operations enabled on the node. When no override is provided, the operations selected in the node panel act as defaults.

Attachments & Documents

You can upload files to EspoCRM as `Attachment` records and then create a `Document` that references the uploaded file. You can also download attachments to binary output in n8n.

Upload Methods

The Attachment upload operation supports two input sources:

Option A: Binary Field (default)

Upload from a binary property on the input item. Use this when you have binary data from a previous node (e.g., HTTP Request, Read Binary File, Email trigger).

- Add an `EspoCRM` node
- Resource: `Attachment`
- Operation: `Upload`
- Input Source: `Binary Field`
- Fields:
  - `Binary Property`: name of the binary key on the incoming item (default `data`)
  - `Related Type`: usually `Document` (but can be any entity supported by your instance)
  - `Field`: usually `file` for Document's File field
  - `Role`: defaults to `Attachment` (other roles: Inline Attachment)
- The node derives `name`, `type`, and `size` from the binary; you can override `name` and `type` in Additional Fields
- Output contains the created Attachment `id`

Option B: Base64 Direct

Provide base64-encoded file content directly. Use this when you have base64 data from an API response, AI agent, or other source.

- Add an `EspoCRM` node
- Resource: `Attachment`

- Operation: `Upload`
- Input Source: `Base64 Direct`
- Fields:
  - `Base64 Data`: the base64-encoded file content (without data URI prefix)
  - `File Name`: file name with extension (e.g., `document.pdf`)
  - `MIME Type`: optional, auto-detected from file extension if not provided
  - `Related Type`: usually `Document`
  - `Field`: usually `file`
  - `Role`: defaults to `Attachment`
- Output contains the created Attachment `id`

Create a Document linked to the uploaded file

- Add a second `EspoCRM` node
- Resource: `Document`
- Operation: `Create`
- Fields:
  - `Name`: document name
  - `File ID`: reference the Attachment `id` (from the previous step)
  - Optional: `Publish Date` (date only is expected; we normalize inputs), `Status`, `File Name`, `Folder ID`, `Description`, `Assigned User ID`

Notes:

- EspoCRM may restrict allowed file types by extension/MIME. If you receive `403 Not allowed file type`, verify your instance settings and the file's extension/MIME.
- For attachment-multiple fields (e.g., `Note.attachments`), upload first, then create/update the parent entity with `attachmentsIds` including the returned attachment ID (remember to include existing IDs when updating to avoid unlinking).

Resources

- EspoCRM API Documentation
- n8n Community Nodes Documentation

Support

- For EspoCRM-specific issues: EspoCRM Forum
- For node-specific issues: Create an issue in the GitHub repository

- For n8n-related questions: n8n Community

Development

Local Development on macOS (npm)

For local development on macOS with n8n installed globally via npm:

1. **Install n8n globally** (if not already installed):

```
npm install -g n8n
```

2. **Clone and build the package**:

```
git clone https://github.com/traien/n8n-nodes-espocrm.git
cd n8n-nodes-espocrm
npm install
npm run build
```

3. **Link the package to n8n's custom nodes folder**:

```
# Create package.json in n8n's nodes folder if it doesn't exist
mkdir -p ~/.n8n/nodes
cd ~/.n8n/nodes
echo '{"name": "installed-nodes", "private": true}' > package.json

# Install the local package (creates a symlink)
npm install /path/to/your/n8n-espocrm
```

4. **Start n8n**:

```
n8n start
```

   Access n8n at http://localhost:5678

5. **Development workflow**:

```
# After making changes to the source code:
cd /path/to/your/n8n-espocrm
npm run build

# Restart n8n to pick up changes (Ctrl+C to stop, then):
```

```
n8n start
```

6. **Watch mode** (optional - auto-rebuild on changes):

   ```
   npm run dev
   ```

   Note: You still need to restart n8n after rebuilds.

Docker Development

To run a local version of n8n with this node for development using Docker. Make sure to replace `/path/to/your/local/n8n-espocrm` with the absolute path to this repository on your machine.

```
docker run -it --rm \
  --name n8n \
  -p 5678:5678 \
  -e N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS=true \
  -e N8N_RUNNERS_ENABLED=true \
  -e N8N_LOG_LEVEL=debug \
  -e N8N_COMMUNITY_PACKAGES_ALLOW_TOOL_USAGE=true \
  -v n8n_data:/home/node/.n8n \
  -v /path/to/your/local/n8n-espocrm:/home/node/.n8n/custom/n8n-espocrm \
  n8nio/n8n
```

Useful Commands

Command	Description
`npm run build`	Compile TypeScript and copy icons
`npm run dev`	Watch mode for active development
`npm run lintfix`	Auto-fix ESLint issues
`npm run lint`	Check for linting errors

License

MIT