

Installation

- [As a package in an existing Laravel app](#)
- [As a Standalone app](#)
- [Using Docker Image](#)

As a package in an existing Laravel app

Getting a license

If you already have a Laravel 10 application, you can install Mixpost as a package inside it.

To install Mixpost Enterprise, you'll need to [get a license](#) first.

Installation via Composer

First, add the `packages.inovector.com` repository to your `composer.json`.

```
"repositories": [  
    {  
        "type": "composer",  
        "url": "https://packages.inovector.com"  
    }  
],
```

Next, you need to create a file called `auth.json` and place it either next to the `composer.json` file in your project or the composer home directory. Using this command, you can determine the composer home directory on *nix machines.

```
composer config --list --global | grep home
```

This is the content you should put in `auth.json`:

```
{  
    "http-basic": {  
        "packages.inovector.com": {  
            "username": "<YOUR-MIXPOST-APP-ACCOUNT-EMAIL-ADDRESS-HERE>",  
            "password": "<YOUR-LICENSE-KEY-HERE>"  
        }  
    }  
}
```

```
}  
}  
}
```

To validate if Composer can read your `auth.json` you can run this command:

```
composer config --list --global | grep packages.inovector.com
```

If you did everything correctly, the above command should display your credentials. If that command doesn't display anything, verify that you created an `auth.json` as mentioned above.

With this configuration in place, you'll be able to install the package into your Laravel project using this command:

```
composer require inovector/mixpost-enterprise "^2.0"
```

After installing the Mixpost Enterprise package, you may execute these 2 commands:

```
php artisan mixpost:install  
php artisan mixpost-enterprise:install
```

To ensure that assets get republished each time Mixpost is updated, we strongly advise you to add the following command to the `post-update-cmd` of the `scripts` section of your `composer.json`.

```
"scripts": {  
    "post-update-cmd": [  
        "@php artisan mixpost:publish-assets --force=true",  
        "@php artisan mixpost-enterprise:publish-assets --force=true"  
    ]  
}
```

Exclude payment webhook handler route From CSRF Protection by adding the URI `"/payment-webhook"` to the ***\$except*** property of the `VerifyCsrfToken` middleware:

```
<?php  
  
namespace App\Http\Middleware;  
  
use Illuminate\Foundation\Http\Middleware\VerifyCsrfToken as Middleware;  
  
class VerifyCsrfToken extends Middleware  
{
```

```

/**
 * The URIs that should be excluded from CSRF verification.
 *
 * @var array<int, string>
 */
protected $except = [
    ' */payment- webhook'
];
}

```

Mixpost uses [Job Batching](#) and you should create a database migration to build a table to contain meta-information about your job batches.

If your application does not yet have this table, it may be generated using the:

```
php artisan queue: batches- table
```

Run the migrations with:

```
php artisan migrate
```

To make media files (images & videos) accessible from the web you should create a symbolic link from public/storage to storage/app/public

```
php artisan storage: link
```

You can publish the config file with:

```
php artisan vendor: publish -- tag=mixpost- config
```

Mixpost has the ability to generate images from video while uploading a video file. This would not be possible without FFmpeg installed on your server. You need to follow FFmpeg installation instructions on their [official website](#).

After installation, depending on the operating system, you need to set the `ffmpeg_path` and `ffprobe_path` in the Mixpost config file.

Default folder path: `/usr/bin/`. If FFmpeg is there, there is no need to change it.

```

/*
 * FFMPEG & FFProbe binaries paths, only used if you try to generate video thumbnails
 */

```

```
'ffmpeg_path' => env('FFMPEG_PATH', '/usr/bin/ffmpeg'),  
'ffprobe_path' => env('FFPROBE_PATH', '/usr/bin/ffprobe'),
```

Or, you can set them in your `.env` file

```
FFMPEG_PATH=/usr/bin/ffmpeg  
FFPROBE_PATH=/usr/bin/ffprobe
```

Error Reporting

Mixpost utilizes its unique internal exception handler rather than the default "

App\Exceptions\ExceptionHandler". To integrate external error reporting tools with your Mixpost setup, you should use the "**Mixpost::report**" method. Generally, this method is called from the register method of your app's "**App\Providers\AppServiceProvider**" class:

```
use Inovector\Mixpost\Mixpost;  
use Sentry\Laravel\Integration;  
  
Mixpost::report(function($exception) {  
    Integration::captureUnhandledException($exception);  
});
```

Install Horizon

Mixpost handles various tasks in a queued way via [Laravel Horizon](#). If your application doesn't have Horizon installed yet, follow [their installation instructions](#).

After Horizon is installed, don't forget to set `QUEUE_CONNECTION` in your `.env` file to `redis`.

`config/horizon.php` should have been created in your project. In this config file, you must add a block named `mixpost-heavy` to both the `production` and `local` environment.

```
'environments' => [  
    'production' => [  
        'supervisor-1' => [  
            'maxProcesses' => 10,
```

```

        'balanceMaxShift' => 1,
        'balanceCooldown' => 3,
    ],
    'mixpost-heavy' => [
        'connection' => 'mixpost-redis',
        'queue' => ['publish-post'],
        'balance' => 'auto',
        'processes' => 8,
        'tries' => 1,
        'timeout' => 60 * 60,
    ],
],
],

'local' => [
    'supervisor-1' => [
        'maxProcesses' => 3,
    ],
    'mixpost-heavy' => [
        'connection' => 'mixpost-redis',
        'queue' => ['publish-post'],
        'balance' => 'auto',
        'processes' => 3,
        'tries' => 1,
        'timeout' => 60 * 60,
    ],
],
],
],

```

In the `config/queue.php` file you must add the `mixpost-redis` connection:

```

'connections' => [

    // ...

    'mixpost-redis' => [
        'driver' => 'redis',
        'connection' => 'default',
        'queue' => env('REDIS_QUEUE', 'default'),
        'retry_after' => 11 * 60,
        'block_for' => null,
    ],
],

```

```
],
```

Don't forget to run `php artisan horizon`. In production, you need a way to keep your `horizon` processes running. For this reason, you need to configure a process monitor [Supervisor](#) that can detect when your `horizon` processes exit and automatically restart them.

Example of supervisor config:

```
[program: mixpost_horizon]
process_name=%(program_name)s
command=php /path-to-your-project/artisan horizon
autostart=true
autorestart=true
user=your_user_name
stopwaitsecs=3600
```

Schedule the commands

In the console kernel (`app/Console/Kernel.php`), you should schedule these commands:

```
protected function schedule(Schedule $schedule)
{
    // ...
    \Inovector\MixpostEnterprise\Schedule::register($schedule);

    $schedule->command(' horizon: snapshot' )->everyFiveMinutes();
    $schedule->command(' queue: prune-batches' )->daily();
}
```

Don't forget to add a cron that runs the scheduler:

```
* * * * cd /path-to-your-project && php artisan schedule:run >> /dev/null 2>&1
```

Visit the UI

After performing all these steps, you should be able to visit the Mixpost UI at `/mixpost`.

As a Standalone app

If you don't know Laravel, but have basic PHP knowledge and know how to deploy to a server, you can follow these instructions.

You can create a new Laravel application with Mixpost Enterprise preinstalled using Composer.

Getting a license

In order to install Mixpost Enterprise, you'll need to [get a license](#) first.

Creating the application

You can create the application with Mixpost pre-installed using this command

```
composer create-project inovector/mixpost-enterprise-app
```

During the execution of this command, Composer will ask for a user and a password. The user is the email address of your [mixpost.app](#) account. The password is the key to your [Mixpost license](#).

Configure the app URL

You will need to modify the value of the APP_URL in the `.env` file to your project URL.

For example: `APP_URL=https://your-domain.com`

Configure the database

You will need to modify the values of the DB_* entries in the `.env` file to make sure they are aligned with your database.

Then, run the migration to create all tables.

Configure the SMTP

By configuring SMTP, Mixpost will be able to send emails such as (password reset link). You will need to modify your `.env` file.

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.mailgun.org
MAIL_PORT=587
MAIL_USERNAME=
MAIL_PASSWORD=
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS="hello@example.com"
MAIL_FROM_NAME="${APP_NAME}"
```

Integrate with Sentry - Error Reporting

[Read the integration guide](#)

Modify the value of "**SENTRY_LARAVEL_DSN**=" in the `.env` file.

File Permissions

Make sure you have the right file permissions by following this [tutorial](#).

Server configuration (Manual)

[Server configuration Guide](#)

Visit the UI

After performing all these steps, you should be able to visit the Mixpost UI at /mixpost.

Deploying to production

Change the value of APP_ENV from the .env file to **production**.

```
APP_ENV=production
```

Debug Mode

In your production environment, this value should always be false. If the APP_DEBUG variable is set to true in production, you risk exposing sensitive configuration values to your application's end users.

Caching Configuration

When deploying to production, you should make sure that you run the `config:cache` command during your deployment process:

```
php artisan config:cache
```

Once the configuration has been cached, everything you change in the .env file will have no effect. To have an effect, you must repeat the execution of the cache command.

Caching Routes

To improve Mixpost application performance, run:

```
php artisan route:cache
```

Once the route has been cached, you have to repeat it every time you deploy/update/upgrade the Mixpost application.

Deploying Horizon

During your Mixpost deployment process (update/upgrade of Mixpost or change some code), you should instruct the Horizon process to terminate so that it will be restarted by your process monitor and receive your code changes:

```
php artisan horizon:terminate
```

Using Docker Image

Docker Installation (Guide)

Have a VPS?

- [Install Docker Engine](#)

Desktop?

- [Docker Install documentation](#)
- [Docker-Compose Install documentation](#)

Just create a `docker-compose.yml` file on your server, and change the values (which start with the `example_*`) with real values.

For passwords, we recommend using strong values. You can use [this](#) tool to generate strong passwords:

Required:

- `LICENSE_KEY` (license key, see [Mixpost Enterprise license](#) page)
- `APP_URL` (your app URL, for example, <https://my-project.com>)
- `APP_KEY` (your app key. Generate a base64 secret with this [tool](#))
- `DB_DATABASE` (the database name of the linked MySQL container)
- `DB_USERNAME` (the user of the linked MySQL container)
- `DB_PASSWORD` (the user password of the linked MySQL container)

Optional:

- `APP_NAME` (Your project name. Default value: `'Mixpost'`)
- `APP_DEBUG` (Available options: `true`/`false`. Default value: `false`)
- `MIXPOST_CORE_PATH` (This is the path that Mixpost will use to load its core routes and assets. Default value: `mixpost`)
- `MIXPOST_PUBLIC_PAGES_PREFIX` (public pages have an endpoint directly after the URL domain (for example, <https://your-domain/privacy-policy>). Default value: `pages`, which means <https://your-domain/pages/privacy-policy>)
- `MIXPOST_FORGOT_PASSWORD` (Enable/Disable Forgot Password feature. Default value: `'true'`)

- `MIXPOST_TWO_FACTOR_AUTH` (Enable/Disable Two Factor Authentication feature. Default value: `'true'`)
- **`FORCE_CORE_PATH_CALLBACK_TO_NATIVE`** (true/false)- Force the callback URL to use the native "**mixpost**" core path. Default value: `'false'`
- **`SENTRY_LARAVEL_DSN`** - Send errors to sentry. [Read the integration guide](#)
- `MAIL_HOST` (Default value: *smtp.mailgun.org*)
- `MAIL_PORT` (Default value: *587*)
- `MAIL_USERNAME`
- `MAIL_PASSWORD`
- `MAIL_ENCRYPTION` (Default value: *tls*)
- `MAIL_FROM_ADDRESS` (Default value: *hello@example.com*)
- `MAIL_FROM_NAME` (Default value: *Example*)
- `DB_HOST` (The IP of the MySQL server. Default value: `mysql`)
- `DB_PORT` (The port of the MySQL server. Default value: `3306`)
- `REDIS_HOST` (The IP of Redis server. Default value: `redis`)
- `REDIS_PORT` (The port of the Redis server. Default value: `6379`)

```
version: '3.1'
```

```
services:
```

```
  mixpost:
```

```
    image: inovector/mixpost-enterprise:latest
```

```
    environment:
```

```
      LICENSE_KEY: 'example_license_key'
```

```
      APP_URL: https://your-domain.com
```

```
      APP_KEY: example_secret_key # Generate a base64 secret with this tool:
```

```
https://mixpost.app/encryption-key-generator
```

```
      DB_DATABASE: 'example_db_name'
```

```
      DB_USERNAME: 'example_db_user'
```

```
      DB_PASSWORD: 'example_db_password'
```

```
    ports:
```

```
      - 9000:80
```

```
    volumes:
```

```
      - storage:/var/www/html/storage/app
```

```
    depends_on:
```

```
      - mysql
```

```
      - redis
```

```
    restart: unless-stopped
```

```
  mysql:
```

```
    image: 'mysql/mysql-server:8.0'
```

```
    ports:
```

```
      - '3306:3306'
```

```
environment:
  MYSQL_DATABASE: 'example_db_name'
  MYSQL_USER: 'example_db_user'
  MYSQL_PASSWORD: 'example_db_password'
volumes:
  - 'mysql:/var/lib/mysql'
healthcheck:
  test: ["CMD", "mysqladmin", "ping", "-p example_db_password"]
  retries: 3
  timeout: 5s
restart: unless-stopped
redis:
  image: 'redis:latest'
  command: redis-server --appendonly yes --replica-read-only no
  volumes:
    - 'redis:/data'
  healthcheck:
    test: ["CMD", "redis-cli", "ping"]
    retries: 3
    timeout: 5s
  restart: unless-stopped
volumes:
  mysql:
    driver: local
  redis:
    driver: local
  storage:
    driver: local
```

Then execute this command:

```
docker-compose up -d
```

You can connect to Mixpost by accessing the URL address set in **APP_URL**

Important information

Mixpost uses encrypting and decrypting text via OpenSSL using AES-256 and AES-128 encryption to secure your credentials of services and connected social accounts. That said, we don't

recommend you change `APP_KEY`. By changing the `APP_KEY`, some functions in the applications will stop working, namely: service credentials and connected social account tokens will no longer be able to be decrypted. You will have to re-enter your service credentials and reconnect your social accounts. For Mastodon, you have to re-create a new app, see the [instructions](#).

Change the `API_KEY` only if you have a serious reason such as your access to the server has been compromised.