

Local Docker Setup & Validation – Documents Signature Project

This document describes the steps followed to set up, run, and validate the Documents Signature application locally using Docker, prior to moving toward a production environment.

The purpose of this setup was to ensure that:

- The Docker configuration works correctly
- The application runs successfully inside containers
- Dependencies install without issues
- The environment is consistent with a future production setup

Commands Used & Explanation

1. Build and Start Containers :

sudo docker compose up -d --build

Purpose:

- Builds Docker images using the Dockerfile
- Creates application and database containers
- Starts containers in detached (background) mode

Result:

- tmx-app (Laravel application) container starts
- tmx-db (MySQL) container starts

2. Verify Running Containers

sudo docker ps

Purpose:

Confirms that required containers are running successfully

3. Access the Application Container

sudo docker exec -it tmx-app bash

Purpose:

- Opens a terminal session inside the running application container
- Allows execution of Laravel and Composer commands in the correct environment

Why this is required:

- Composer and PHP are installed inside the container
- Ensures environment consistency with production

4. Environment File Setup (.env)

At the initial stage, the project did not contain a .env file, which is required for Laravel to run.

The following steps were performed: **cp .env.example .env**

Purpose:

- Creates the required environment configuration file
- Allows application-specific settings such as database credentials and app URL to be defined
- After creating the file, database and application values were configured to match the Docker setup (MySQL service, ports, credentials, etc.).

5. Install Laravel Dependencies

composer install

Purpose:

- Installs PHP dependencies defined in composer.json
- Generates the vendor/ directory
- Prepares the Laravel application to run

6. Laravel Post-Install Commands

php artisan key:generate

php artisan config:clear

php artisan cache:clear

Purpose:

- Generates application encryption key.
- Clears cached configuration and application cache.
- Ensures a clean and predictable runtime state.

7. PSR-4 Autoloading Warning & Resolution

Issue Identified

Composer reported PSR-4 warnings for duplicate middleware files:
checkLogin(1).php
checkLogin_9_9_2025.php

Root Cause

File names did not match class names
Duplicate backup files violated PSR-4 autoloading standards

Resolution Steps

- Removed duplicate middleware files
- Renamed the valid file to: CheckLogin.php
- Updated class name to match file name
- Regenerated Composer autoload files

composer dump-autoload

This ensured full PSR-4 compliance.

8. Application Verification

After completing the above steps, the application was successfully accessed in the browser using the following URL:

<http://localhost:8080>