Procédure de Déploiement : Application Symfony sur Wampserver

Stack technique :

- **Serveur :** Wampserver
- **PHP :** 8.3.14
- **Framework :** Symfony 7.3.4
- **Assets :** AssetMapper
- **Base de données :** MySQL (via Wampserver)

Prérequis

Avant de commencer, assurez-vous que :

1. **Wampserver est installé** et fonctionnel.
2. La version de **PHP 8.3.14 est active** dans Wampserver (vérifiez via l'icône Wampserver dans la barre des tâches).
3. **Composer 2** est installé globalement sur votre machine Windows.
4. **Git** est installé (recommandé pour récupérer le code).
5. Votre projet est versionné sous Git et prêt à être déployé.

Étape 1 : Configuration de l'environnement Wampserver

1. **Activer `mod_rewrite` d'Apache :**
   - Clic gauche sur l'icône Wampserver.
   - Allez dans `Apache` -> `Modules Apache`.
   - Assurez-vous que `rewrite_module` est coché. S'il ne l'est pas, cochez-le. Wampserver redémarrera Apache.

2. **Créer la base de données de production :**
   - Ouvrez `phpMyAdmin` depuis la page d'accueil de Wampserver.
   - Connectez-vous (par défaut `root` / mot de passe vide, ou `root` / `root` comme dans votre `.env.local`).
   - Créez une nouvelle base de données (par exemple, `taskflow_prod`) avec l'interclassement `utf8mb4_general_ci`.
   - (Optionnel mais recommandé) Créez un utilisateur spécifique pour cette base de données avec des droits limités, au lieu d'utiliser `root`.

Étape 2 : Récupération du code source

Placez le code de votre application dans le répertoire `www` de Wampserver. Nous utiliserons un dossier dédié (par exemple, `taskflow`).

Méthode recommandée (avec Git) :

1. Ouvrez un terminal (PowerShell ou Git Bash) dans votre répertoire Wampserver : `bash cd C:\wamp64\www`
2. Clonez votre projet (utilisez la branche principale ou un tag de release) : `bash # Remplacez <url_du_repository> par l'URL de votre dépôt Git git clone <url_du_repository> taskflow cd taskflow # Si vous n'êtes pas sur la branche principale (ex: 'main' or 'master') # git checkout main`

Méthode alternative (copie manuelle) :

- Copiez l'intégralité de votre dossier de projet (sauf le dossier `vendor/` et `var/`) vers `C:\wamp64\www\taskflow`.

Étape 3 : Installation des dépendances

Installez les dépendances Composer en mode production. Cela exclut les paquets de développement (`require-dev`) et optimise l'autoloader.

1. Ouvrez un terminal dans le dossier du projet : `bash cd C:\wamp64\www\taskflow`
2. Lancez l'installation : `bash composer install --no-dev --optimize-autoloader`

Étape 4 : Configuration des variables d'environnement

En production, nous n'utilisons **pas** le fichier `.env.local` (qui contient vos configurations de développement). Nous allons créer un fichier `.env.prod.local` pour les secrets de production.

1. **Créez le fichier `.env.prod.local`** à la racine de votre projet (`C:\wamp64\www\taskflow\.env.prod.local`).
2. **Remplissez ce fichier** avec vos configurations de production. Ce fichier *ne doit pas* être versionné (il est dans `.gitignore` par défaut).
3. `# C:\wamp64\www\taskflow\.env.prod.local`
4.
5. `###> symfony/framework-bundle ###`
6. `# Générez un secret fort pour la production !`
7. `# Vous pouvez le générer avec : php bin/console secrets:set APP_SECRET --env=prod`
8. `# ou copier la sortie de : openssl rand -base64 32`
9. `APP_SECRET=VOTRE_VRAI_SECRET_DE_PRODUCTION_TRES_LONG`
10. `###< symfony/framework-bundle ###`
11.
12. `###> doctrine/doctrine-bundle ###`
13. `# Mettez ici les accès à votre base de données de production (celle créée à l'étape 1)`

```
# Utilisez le nom de la BDD de prod (ex: taskflow_prod) et l'utilisateur/mot de passe de prod
DATABASE_URL="mysql://root:root@127.0.0.1:3306/taskflow_prod?serverVersion=8.0.32&charset=utf8mb4"
###< doctrine/doctrine-bundle ###

###> symfony/mailer ###
# Configurez votre vrai service d'envoi d'emails (ex: SendGrid, Mailgun, ou un serveur SMTP)
# Exemple pour un SMTP Gmail (à adapter) :
# MAILER_DSN=smtp://votre_email@gmail.com:votre_mot_de_passe_app@smtp.gmail.com:587
MAILER_DSN=null://null
###< symfony/mailer ###
```

Étape 5 : Configuration du Virtual Host Apache

Pour que Wampserver pointe correctement vers votre application (et non vers la racine `www`), nous devons créer un "Virtual Host".

1. **Modifiez `httpd-vhosts.conf` :**
   - Ouvrez le fichier : `C:\wamp64\bin\apache\apache[VERSION]\conf\extra\httpd-vhosts.conf`
   - Ajoutez ce bloc à la fin du fichier (adaptez les chemins si nécessaire) :

```
# Virtual Host pour l'application Taskflow
<VirtualHost *:80>
    # Nom de domaine local pour votre app
    ServerName taskflow.local
# Chemin vers le dossier "public" de votre projet Symfony
DocumentRoot "C:/wamp64/www/taskflow/public"

&lt;Directory "C:/wamp64/www/taskflow/public"&gt;
    AllowOverride All
    Require all granted
&lt;/Directory&gt;

# (Optionnel) Fichiers de log séparés pour ce projet
ErrorLog "C:/wamp64/logs/taskflow-error.log"
CustomLog "C:/wamp64/logs/taskflow-access.log" common
</VirtualHost>
```

5. **Modifiez le fichier `hosts` de Windows :**
   - Ouvrez le Bloc-notes **en tant qu'administrateur**.
   - Ouvrez le fichier : `C:\Windows\System32\drivers\etc\hosts`
   - Ajoutez cette ligne à la fin pour faire correspondre le `ServerName` à votre machine locale :

```
127.0.0.1   taskflow.local
::1         taskflow.local
```

8. **Redémarrez Wampserver :**
   - Clic droit sur l'icône Wampserver -> `Redémarrer les services`.

Étape 6 : Finalisation (Compilation et Base de données)

Ces commandes préparent l'application pour l'environnement de production (`--env=prod`).

1. Ouvrez un terminal dans le dossier du projet (`C:\wamp64\www\taskflow`).
2. **Compiler les variables d'environnement :** Cette commande lit tous les fichiers `.env` (y compris `.env.prod.local`) et crée un fichier `.env.runtime.php` optimisé pour la production.
3. `composer dump-env prod`
4. **Exécuter les migrations de base de données :** Cela va créer les tables dans votre base de données `taskflow_prod`.
5. `php bin/console doctrine:migrations:migrate --env=prod`
    - Confirmez par `yes` lorsque la commande vous le demande.
6. **Compiler les assets (AssetMapper) :** Cette commande va "compiler" (copier et versionner) vos assets dans le dossier `public/assets`.
7. `php bin/console asset-map:compile --env=prod`
8. **Nettoyer et chauffer le cache (Warmup) :** Bien que `dump-env` gère une partie du cache, il est bon de s'assurer que tout est propre. `bash php bin/console cache:clear --env=prod`

Étape 7 : Vérification

1. Ouvrez votre navigateur et accédez à l'URL que vous avez définie à l'étape 5 : **http://taskflow.local**
2. Votre application devrait maintenant fonctionner en mode production.
3. En cas d'erreur (`500 Internal Server Error`), consultez les logs d'erreurs spécifiés dans votre Virtual Host (`C:/wamp64/logs/taskflow-error.log`) ou les logs de Symfony (`C:/wamp64/www/taskflow/var/log/prod.log`) pour diagnostiquer le problème.

Annexe : Procédure de Mise à Jour (Déploiement continu)

Pour mettre à jour votre application, les étapes sont similaires mais plus courtes :

1. (Recommandé) Activer le mode maintenance de Symfony si nécessaire.
2. Récupérer les dernières modifications du code : `bash cd C:\wamp64\www\taskflow git pull origin main`
3. Installer les dépendances (si `composer.json` a changé) : `bash composer install --no-dev --optimize-autoloader`
4. Mettre à jour les variables d'environnement (si `.env.prod.local` doit changer).
5. Recompiler les variables d'environnement : `bash composer dump-env prod`
6. Lancer les migrations de base de données (s'il y en a de nouvelles) : `bash php bin/console doctrine:migrations:migrate --env=prod`
7. Recompiler les assets (si les assets ont changé) : `bash php bin/console asset-map:compile --env=prod`
8. Nettoyer le cache : `bash php bin/console cache:clear --env=prod`
9. (Recommandé) Désactiver le mode maintenance.