

# Déploiement - EcoRide

Projet de covoiturage écoresponsable



## A. Choix de l'hébergeur

### i. introduction

Il est vrai qu'avec Symfony, il y a quelques petites spécificités lorsque l'on souhaite mettre en production son projet. La première chose à faire, c'est de se demander sur quel hébergeur, sur quelle plateforme, sur quel serveur ou service nous allons héberger notre site internet et nos différents fichiers et sa base de données.

Nous avons différentes possibilités. Tout d'abord, nous avons les hébergements dits classiques, c'est-à-dire que l'on va nous louer une partie d'un serveur ou même un serveur entier. Et ce sera à nous de gérer nos fichiers, nos dossiers mais aussi nos configurations, éventuellement même, ce sera à nous d'installer les logiciels nécessaires à l'utilisation de votre site. C'est ce que nous pouvons retrouver chez OVH, Hostinger ou 1 and 1.

Nous avons ensuite d'autres types d'hébergement, les hébergements qu'on pourrait appeler de type conteneur, comme on retrouve par exemple, chez AWS ou même chez HEROKU. Alors, l'avantage de ces types d'hébergement-là, c'est que nous n'avons presque pas besoin de connaissances concernant l'infrastructure. Tout va être un peu pré mâché pour nous et surtout l'autre avantage, c'est que, la plupart du temps, ce type d'offre nous donne l'accès à des options gratuites, à des fonctionnalités gratuites.

Alors, que ce soit dans des serveurs classiques comme OVH ou dans des conteneurs comme HEROKU, vous avez toujours, grosso modo, trois questions que vous devez vous poser, et vous allez voir que ces trois questions vont rester avec nous pendant toute cette section.

La première question, c'est l'**infrastructure**. Quand on a créé notre site, on a effectivement notre code, qui est le centre de tout. Mais on a utilisé aussi des outils qui venaient à côté, par exemple une base de données, MySQL. Autre chose : nous avons servi pendant tout le développement notre site, nous l'avons servi avec le Symfony serve, avec le serveur interne de PHP. Mais il nous faut bien un serveur web en production pour travailler là-dessus et le serveur interne de PHP n'est pas fait pour ça. Il existe des logiciels qui sont faits pour ça par exemple, Apache ou NGINX.

Donc, on voit que pour faire tourner notre site, finalement, ce n'est pas juste une question de code, il y a des logiciels qui doivent être installés à côté, MySQL, APACHE et peut-être même d'autres logiciels.

Deuxième question à se poser : quelles sont les **configurations nécessaires** ?

Nous avons différentes variables d'environnement qui sont nécessaires à être mises en place et qui doivent changer éventuellement quand on va passer le site dans un autre environnement, dans l'environnement de prod par exemple. Pour que notre site puisse réellement tourner, le code lui-même n'est pas suffisant. Il y a quelques scripts qu'on doit faire tourner, des scripts pour créer la base de données, des scripts pour migrer les tables, peut-être même des scripts pour remplir ces tables avec nos Fixtures, et peut-être même d'autres scripts. Donc, on voit que la deuxième question sur la configuration est aussi essentielle à comprendre quand on se prépare à une mise en ligne.

Et enfin, la troisième et dernière question : notre **code**, le cœur de notre application. On va se poser la question suivante : comment rendre ce code accessible de façon à ce que nos déploiements soient facilités, que l'on n'ait pas à passer par un copier-coller de dossier sur un FTP. Là aussi, c'est une question à laquelle on va répondre relativement simplement en ayant créé un dépôt Github.

En conclusion, nous choisissons Hostinger.

## B. Déploiement via GIT

### i. Documentation de Symfony

Avant toutes choses, nous allons sur la documentation de Symfony et voir ce que nous dit Symfony en matière de déploiement d'applications.

<https://symfony.com/doc/current/deployment.html>

Quand on descend un tout petit peu, Symfony nous propose plusieurs types de déploiement.

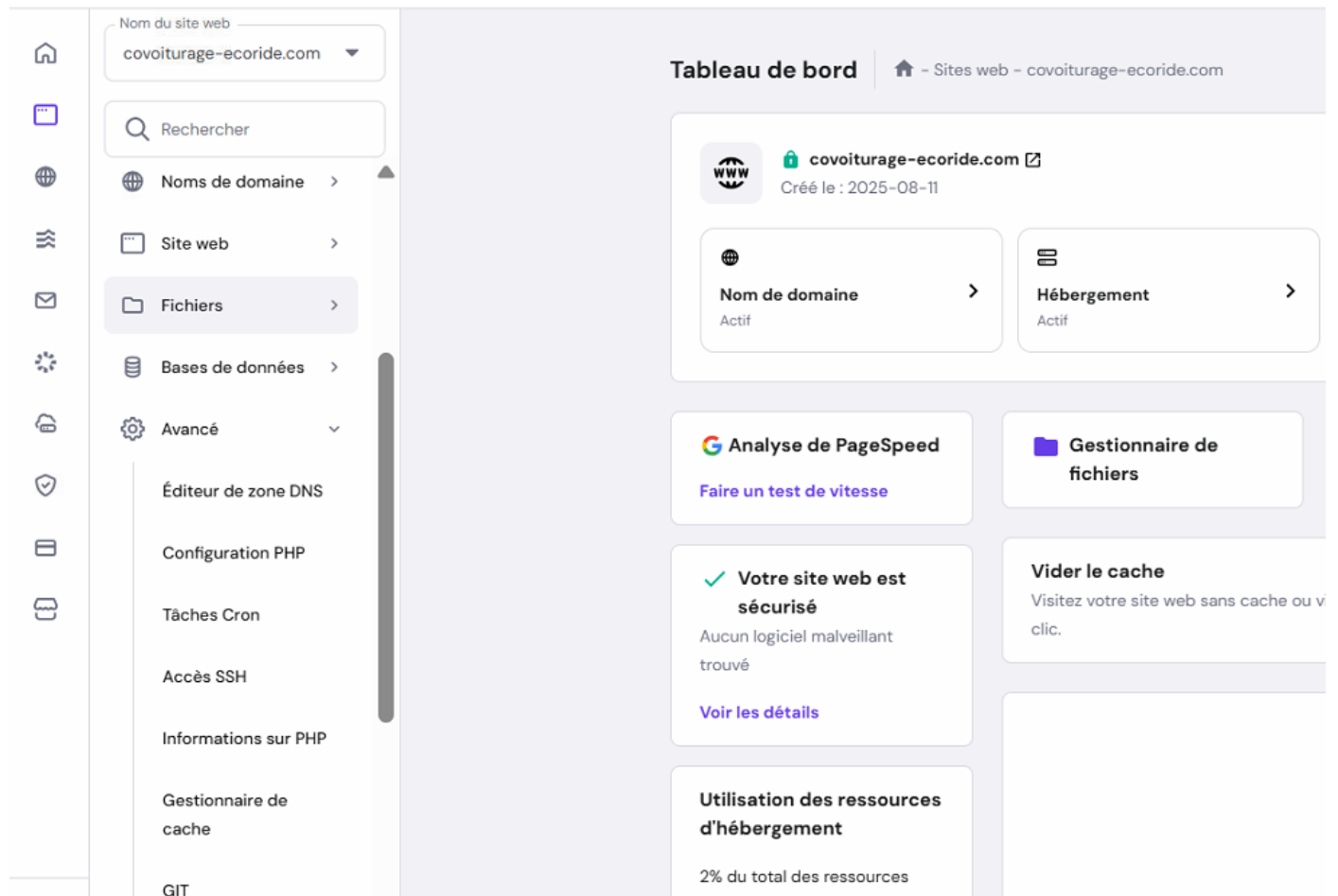
<https://symfony.com/doc/current/deployment.html#how-to-deploy-a-symfony-application-1>

Le déploiement qu'on va appeler basique, c'est à dire que nous avons un compte FTP et nous déployons à la main les fichiers un par un ou en tout cas dossier par dossier sur le FTP. C'est une méthode qui fonctionne mais qui est très longue, et nos fichiers ne sont pas versionnés. Et à chaque fois que nous allons vouloir faire une modification dans notre application, nous allons devoir passer par le FTP. Et, ce n'est pas forcément la meilleure méthode.

Nous, on va plutôt utiliser ce que Symfony appelle le **Using Source Control**. Effectivement, nous allons utiliser notre dépôt Git pour déployer notre application.

## ii. Paramétrages d'Hostinger

Nous allons nous rendre sur Hostinger :



En cliquant sur « Avancé » puis sur « GIT »

Une étape très importante, c'est ici de générer une clé SSH que nous allons mettre directement dans notre projet GitHub et dans notre compte GitHub.



Pour cela, nous allons copier cette clé pour donner l'autorisation finalement à GitHub de déployer vers Hostinger. Car, il faut bien que Hostinger s'authentifie auprès de GitHub.

 **Dépôt Git Privé**



Ajoutez cette clé SSH à votre Github, Bitbucket ou tout autre service pour déployer des dépôts privés

```
ssh-rsa
AAAAB3NzaClyc2EAAAADAQABAAQgQDSgtVIUtOYVcLn/OrfEULim/ErkJ65nByv+QDyewMXsARIWtMc948+7RJ4QBHIJ1NOA6gjnGpalA2DjK17VRT8QdYU4Kus2Qe
uqGQRhYGVs4B0XPebDZCZeNE7xAbKyO/AtsQJdR/KRizdc6/VbIKHYRUWTJw+rD8syCeBsZke3SmOTpqpZMJgcKSillW45DgYPnalRlij6egDQT6CfvaGw18qW5XiaEij
3RjyGn6VQtdc19LjFBD9mO1C57JnKcpBuz72AyeJsGbGiLO4ZB9sNSdLinherhQxyFSOI5VqJg4OQFYVtOdCCQywi2NRLmqCD2c24E5zoU8LXogRdlxu8P2UBA052SK
SCmEdKjd4jYkZh8q5QlWxKI70Do8jsuFE4xpRUo8NzynG8MeS+Hzc7FhRfVGseOZH7ST12m0H4ZL2AdNCf8apov2sNbar3xEVzoXm1ABKF+LovKFrLvk96LbE/zXWP
mt5bDdmc3LWQhV+o/UhzXx+/ChjEoW/U= u537822312@fr-int-web1179.main-hosting.eu
```

 Copier

Supprimer la clé SSH

Sur GitHub, il faut se rendre sur l'onglet « Settings » :


  olivier435 / sym-ecoride


Type / to search


[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Puis, dans le menu à gauche, il faut se rendre sur « Deploy Keys » :

Security

 Advanced Security

 Deploy keys

 Secrets and variables

Nous arrivons sur cette page et il faut cliquer sur le bouton « Add Deploy Key » :

## Deploy keys

[Add deploy key](#)

[Deploy keys](#) use an SSH key to grant readonly or write access to a single repository. They are not protected by a passphrase and can be a security risk if your server is compromised. If you have a complex project or want more fine-grain control over permissions, consider using [GitHub Apps](#) instead.

Une fois cliqué :

## Deploy keys / Add new

Title

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'.

☐ Allow write access

Can this key be used to push to this repository? Deploy keys always have pull access.

Add key

Nous tombons sur ce formulaire, nous postons notre clé ici, celle que nous a donnée du coup Hostinger. On lui donne un titre, puis on clique sur « Add Key ».

Devons-nous cocher "Allow write access" ?

- **Sans** cocher : la clé pourra seulement **lire** le dépôt (faire git clone, git pull), utile par exemple pour un déploiement automatisé en lecture seule.
- **Avec** la case cochée : la clé pourra **lire et écrire** (faire git push, créer des branches, etc.), donc Hostinger pourra envoyer des commits directement sur GitHub.

Par sécurité, Hostinger ne doit servir **qu'à** récupérer le code pour le déployer.

## Deploy keys / Add new

Title

Hostinger

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGD5SgtVIUtoOYVcLn/0rfEULim/ErkJ65nByv+QDyewMXsARIWtMc948+7RJ4QBHU1N
OA6gjnGpalA2DjK17VRT8QdYU4Kus2QeuqGQRhYGVs4B0XPebDZCZeNE7xAbKy0/AjsQjdR/KRizdc6/VbIKHYRUWTJw+rd8
syCeBsZke3SmOTpqwpZMJgcKsIlW45DgYPnalRlj6egDQT6CfvaGw18qW5XiaEij3RjyGn6VQtdc19LjFBD9mO1C57JnKcpBuz
72AyeJsGbGilo4ZB9sNSdLInherhQxyFSO15VqJg40QFYVt0dCCQywi2NRLmqCD2c24E5zoU8LXogRdlx8P2UBA052SKSCm
EdKjd4jYkZh8q5QIWXxKI70Do8jsuFEd4xpRUo8NzynG8MeS+Hzc7FhRfVgse0ZH7ST12m0H4ZL2AdNCf8apov2sNbar3xEVz
oXm1ABKF+LovKFrLvk96LbE/zXWPmt5bDmc3LWQhV+o/UhzXx+/ChEoW/U= u537822312@fr-int-web1179.main-
hosting.eu
```

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'.

☐ Allow write access

Can this key be used to push to this repository? Deploy keys always have pull access.

Add key

1 deploy key



SSH

Hostinger


SHA256:RqypGgSD1nHCOSwzaqU63Ygu3Ki+KimhTokv1LtCcQ

Added on Aug 13, 2025 by @olivier435

Never used — Read-only

Delete

Nous revenons sur Hostinger :

 **Créer un nouveau dépôt**

Déployer une application directement à partir d'un dépôt public Git. Entrez l'URL http de votre dépôt public, le nom de la branche et le chemin d'installation. Le chemin d'installation peut être laissé vide, l'application sera déployée directement dans le répertoire public\_html. Le chemin d'installation ne doit pas contenir de fichiers ou de dossiers, sinon le déploiement échouera.

Dépôt \*

Exemples :  
Pour les référentiels publics : `https://github.com/WordPress/WordPress.git`  
Pour les référentiels privés : `git@github.com:WordPress/WordPress.git`

Branche \*

master

Généralement, il s'agit de la branche maître

Répertoire

Laissez vide pour déployer directement dans public\_html. Le dossier doit être vide.

✓ Créer

Là, Hostinger nous demande l'URL du dépôt GIT en précisant que si le dépôt GIT est en **public**, alors, il faut copier l'URL **HTTPS** :

Local

Codespaces

Clone

?

HTTPS

SSH

GitHub CLI

`https://github.com/olivier435/sym-ecoride.git`

Clone using the web URL.

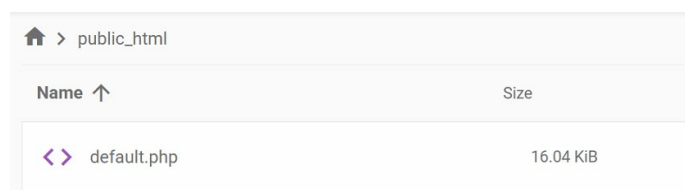
A contrario, si le dépôt est privé, alors, il demande :

`git@github.com:olivier435/sym-ecoride.git`

Use a password-protected SSH key.

Hostinger nous demande ensuite quelle branche nous allons venir chercher. Alors attention, par défaut il nous met master, mais ce n'est plus master les branches maîtres mais **main** côté GitHub.

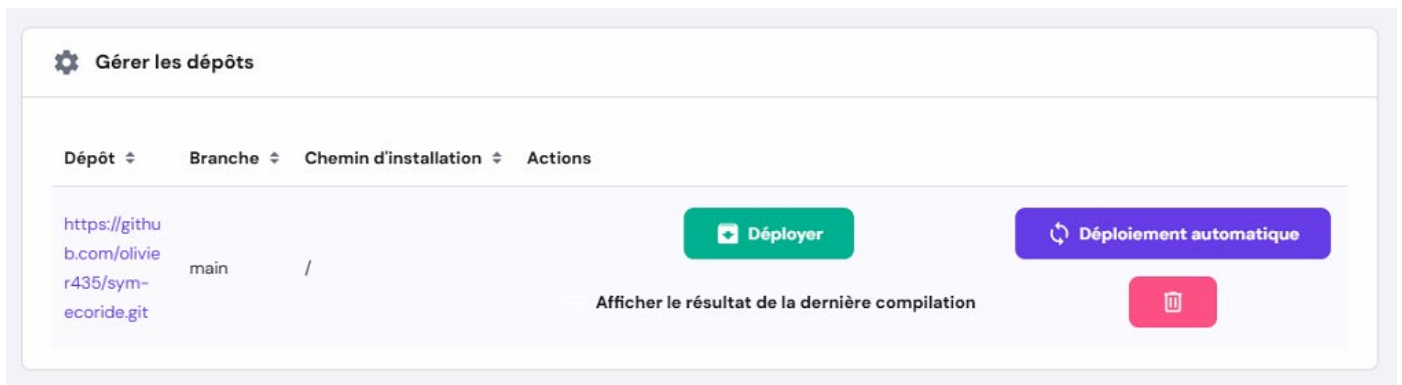
Et dans le répertoire, il nous dit de le laisser vide. Pour déployer directement dans public HTML, le dossier doit être vide. Bien entendu, il faut vérifier via le **Gestionnaire de fichier**, retourner dans le dossier **public\_html**, et on va supprimer **default.php** :



Pour déployer directement dans **public\_html**, le dossier doit être vide.

Nous cliquons sur « **Créer** » et on a une première notification d'Hostinger qui nous dit « dépôt

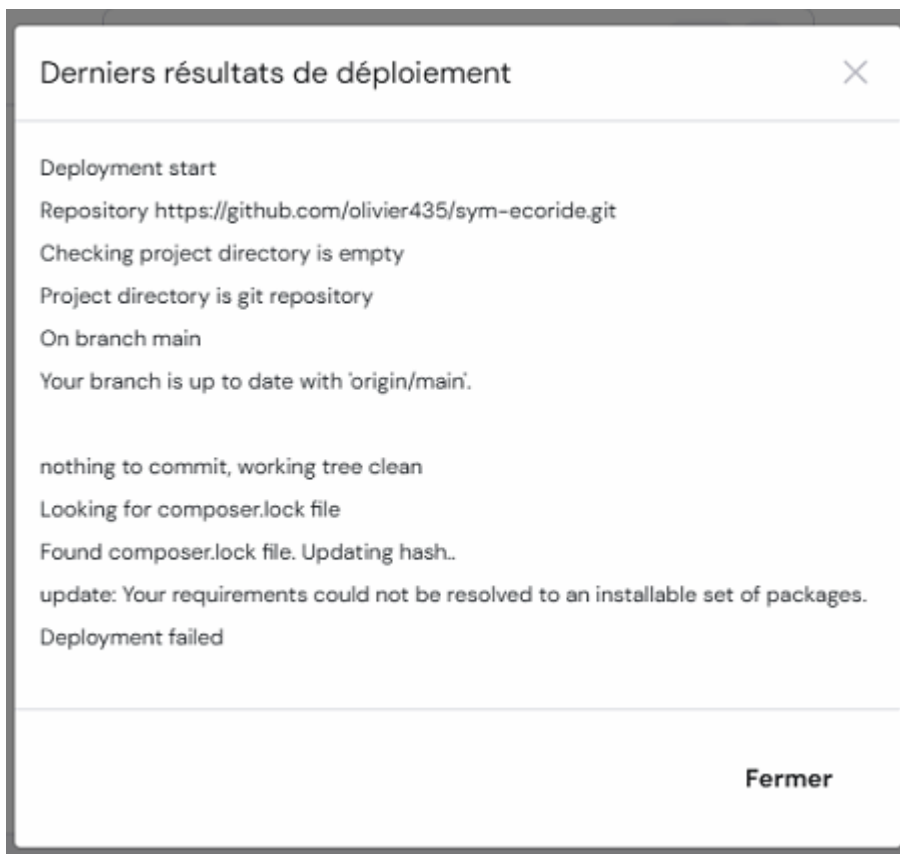
git correctement ajouté » :



Maintenant, nous allons le déployer :



Oui, je suis certain de vouloir le déployer !



Nous allons venir rafraîchir notre dossier dans le « **Gestionnaire de Fichier** » du dossier « **Fichier** » pour voir si les choses fonctionnent bien. Et effectivement, nous nous retrouvons bien avec la copie de notre dépôt GitHub sur notre espace de stockage :



The screenshot shows a web-based file browser interface. The address bar displays the URL: `https://srv1179-files.hstgr.io/b955e7c3dba70841/files/public_html/`. The left sidebar contains navigation options: 'My files', 'New folder', 'New file', 'Trash bin', 'Space' (3.87 MiB / 50 GiB, 0.01%), and 'Inodes' (205 / 600000, 0.03%). The main area shows a directory listing for 'public\_html' with columns for 'Name', 'Size', and 'Last modified'. The files listed are:

Name	Size	Last modified
.git	-	a minute ago
assets	-	4 minutes ago
bin	-	4 minutes ago
config	-	4 minutes ago
doc	-	4 minutes ago
migrations	-	4 minutes ago
public	-	4 minutes ago
sql	-	4 minutes ago
src	-	4 minutes ago
stubs	-	4 minutes ago

Alors qu'est-ce qu'il se passe si maintenant nous retournons dans notre tableau de bord et que nous ouvrons l'application ?

The screenshot shows the Hostinger dashboard for the website 'covoiturage-ecoride.com'. The top navigation bar includes the Hostinger logo, a search bar labeled 'Rechercher', and a 'Ctrl' button. The main content area is titled 'Tableau de bord' and shows the website's status. A red circle highlights the website's icon and name. Below this, there are two cards: 'Nom de domaine' (Active) and 'Hébergement' (Active).

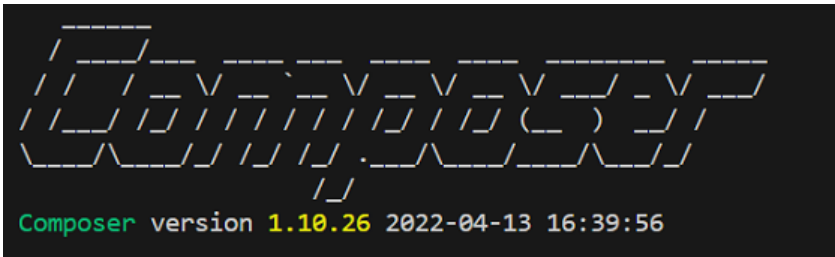
Evidemment nous ne voyons strictement rien et c'est tout à fait logique. Nous allons devoir faire ensemble quelques modifications.

## C. Configuration pour la production

### i. Installer Composer sur Hostinger

Composer est préinstallé sur les plans d'hébergement mutualisé Premium et Business de Hostinger, ce qui est notre cas. Néanmoins, nous allons procéder à la vérification de sa version. Nous nous connectons à notre compte d'hébergement en utilisant une connexion SSH. Dans le terminal :

```
$ composer
```



Composer version 1.10.26 2022-04-13 16:39:56

Nous allons mettre à jour Composer en exécutant ces commandes. Nous vérifions que nous sommes à la racine de notre hébergement :

```
$ ls
```

```
domains
```

Nous téléchargeons Composer depuis le site officiel (<https://getcomposer.org/download/>) en utilisant la commande suivante :

```
$ php -r "copy('https://getcomposer.org/installer', 'composer-setup.php')  
php -r "if (hash_file('sha384', 'composer-setup.php') ===  
'dac665fdc30fdd8ec78b38b9800061b4150413ff2e3b6f88543c636f7cd84f6db9189d43a81e5503cda  
447da73c7e5b6') { echo 'Installer verified'.PHP_EOL; } else { echo 'Installer  
corrupt'.PHP_EOL; unlink('composer-setup.php'); exit(1); }"
```

```
Installer verified
```

Nous pouvons installer Composer localement ou globalement. L'installation locale signifie que le gestionnaire de dépendances sera stocké dans notre répertoire actuel.

```
$ php composer-setup.php --2
```

```
All settings correct for using Composer  
Downloading...
```

```
Composer (version 2.8.10) successfully installed to: /home/u537822312/composer.phar  
Use it: php composer.phar
```

Une fois que c'est fait, nous supprimons l'installateur :

```
$ php -r "unlink('composer-setup.php');"
```

Nous créons un alias

```
$ alias composer="php ~/composer.phar"
```

Nous procédons à une ultime vérification

```
$ composer
```

```
Composer version 2.8.10 2025-07-10 19:08:33
```

## ii. Installer le dossier Vendor

Toujours dans VSCode et avec le terminal et notre connexion SSH :

```
PS C:\Users\DELL\sym-ecoride> ssh -p 65002 u537822312@178.16.128.168
u537822312@178.16.128.168's password:
#####

Vendor: HPE
Model: ProLiant DL325 Gen10 Plus v2

OS Version: CloudLinux 8

Link to hPanel:
https://hpanel.hostinger.com/

[u537822312@fr-int-web1179 ~]$
```

Nous nous rendons dans notre projet :

```
[u537822312@fr-int-web1179 ~]$ cd domains/covoiturage-ecoride.com/public_html
[u537822312@fr-int-web1179 public_html]$ ls
assets                composer.lock          importmap.php          README.md             symfony.lock
bin                   compose.yaml           migrations             sql                  templates
compose.override.yaml config                 phpunit.dist.xml      src                   tests
composer.json         doc                   public                 stubs                 translations
[u537822312@fr-int-web1179 public_html]$
```

Avant de taper dans la console `composer install` et pour éviter l'erreur **ext-sodium**, il faut penser à activer l'extension dans Hostinger :

- Allons dans **hPanel** → **Avancé** → **Extensions PHP**
- Cochons **sodium**
- Sauvegardons

**Extensions PHP à activer dans hPanel → Avancé → Extensions PHP (Passer en PHP 8.3)**

### 1 Obligatoires (cœur Symfony et composants Doctrine)

- **ctype** (*gestion des types de caractères, validation*)
- **iconv** (*conversion de chaînes*)
- **json** (*manipulation JSON*)

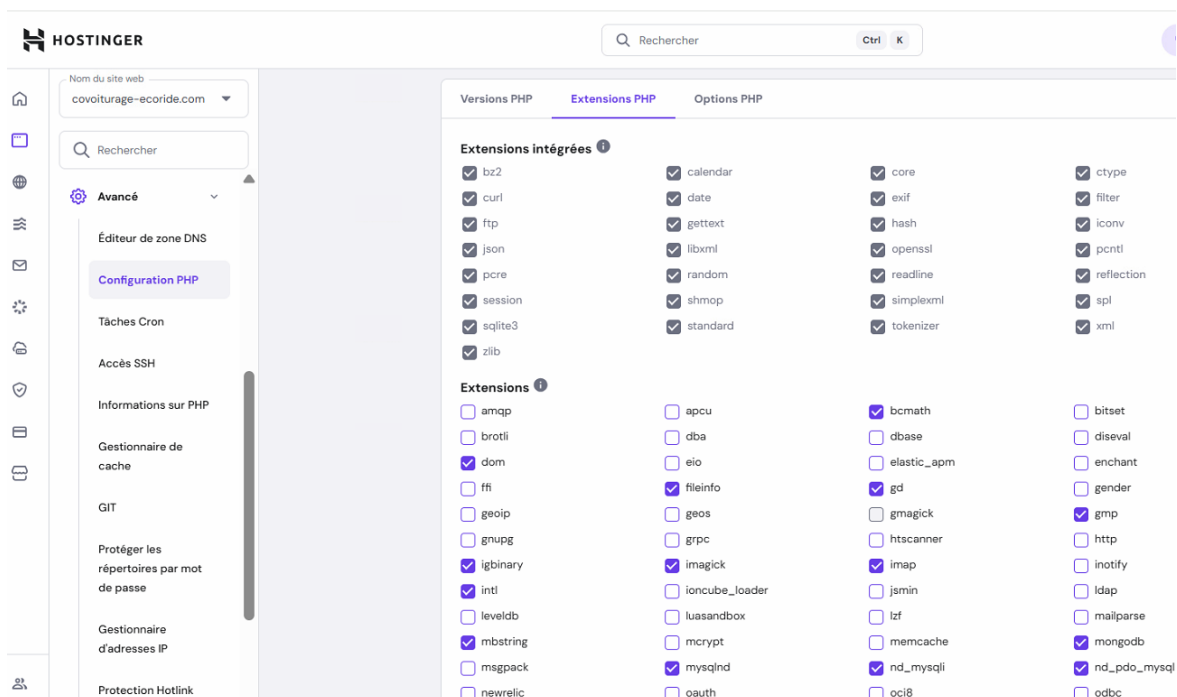
- **mbstring** (gestion multibyte, UTF-8)
- **openssl** (HTTPS, sécurité)
- **pdo** (accès BDD générique)
- **pdo\_mysql** (accès MySQL/MariaDB)
- **session** (gestion des sessions PHP)
- **tokenizer** (analyse syntaxique pour Twig et autres)

## 2 Fortement recommandées (optimisations, sécurité, fonctionnalités supplémentaires)

- **curl** (requêtes HTTP externes)
- **fileinfo** (détection type MIME, VichUploader, etc.)
- **intl** (formatage dates, nombres, traductions)
- **sodium** (cryptographie moderne, JWT, sécurité)
- **xml** (DOMDocument, parsing XML)
- **zip** (lecture/écriture ZIP, install via composer)

## 3 Selon le projet

- **gd** ou **imagick** (manipulation d'images, thumbnails)
- **mongodb** (si MongoDB utilisé en prod)
- **simplexml** (lecture XML simplifiée)
- **exif** (métadonnées images)



Nous devons corriger une autre erreur qui vient du fait que notre projet demande **l'extension MongoDB PHP en version  $\geq 2.1$** , mais sur Hostinger c'est la **1.18.1** qui est installée.

Nous devons assouplir la contrainte dans le `composer.json` pour supporter 1.18.1.

Et comme le Gestionnaire de fichiers ne donne pas les droits d'édition sur certains fichiers, surtout ceux créés via SSH ou uploadés par FTP avec des permissions restreintes, nous pouvons quand même modifier ton `composer.json` via SSH avec un éditeur en ligne de commande :

```
$ nano composer.json
```

- On utilise les flèches pour naviguer,
- On modifie la ligne "mongodb/mongodb": "2.1.0"  
en  
"mongodb/mongodb": "^1.18 || ^2.0"
- On sauvegarde avec CTRL + O, puis **Entrée**
- On quitte avec CTRL + X

```
$ composer update mongodb/mongodb
```

```
Loading composer repositories with package information
Updating dependencies
Lock file operations: 0 installs, 1 update, 1 removal
  - Removing symfony/polyfill-php85 (v1.32.0)
  - Downgrading mongodb/mongodb (2.1.0 => 1.19.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 155 installs, 0 updates, 0 removals
  - Downloading symfony/flex (v2.8.1)
  - Downloading symfony/runtime (v7.2.8)
  ...
  - Installing symfony/flex (v2.8.1): Extracting archive
  - Installing symfony/runtime (v7.2.8): Extracting archive
  ...

Generating autoload files
124 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

Run composer recipes at any time to see the status of your Symfony recipes.

Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
124 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

Run composer recipes at any time to see the status of your Symfony recipes.

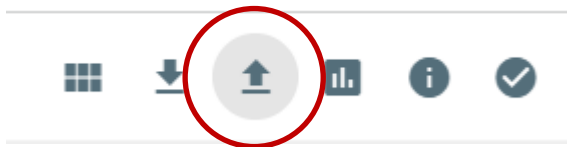
Executing script cache:clear [OK]
Executing script assets:install public [OK]
Executing script importmap:install [OK]

No security vulnerability advisories found.

No security vulnerability advisories found.
Bumping dependencies
./composer.json has been updated (1 changes).
```

### iii. Configurez les variables d'environnement

#### 1. Uploader le `.env.prod` dans le « Gestionnaire de fichiers »



#### 2. Générer le `.env.local.php` optimisé à partir du `.env.prod` en SSH :

```
$ composer dump-env prod
```

```
Successfully dumped .env files in .env.local.php
```

Cette commande va lire `.env`, `.env.prod` et fusionner les valeurs dans un fichier compilé `.env.local.php` que Symfony chargera directement sans relecture des fichiers `.env` à chaque requête.

### iv. Mise à jour des vendors

```
$ composer install --no-dev --optimize-autoloader
```

Le flag `--optimize-autoloader` améliore considérablement les performances du chargeur automatique de Composer en créant une « carte de classes ».

### v. Installation du package `symfony/apache-pack`

```
$ composer require symfony/apache-pack
```

Il a créé dans le dossier public un fichier `.htaccess`.

Et à la racine de `public_html`, on crée un autre fichier `.htaccess` qui va nous permettre de rediriger vers le dossier public :

```
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /

RewriteCond %{THE_REQUEST} /public/([^\s?]*) [NC]
RewriteRule ^ %1 [L,NE,R=302]

RewriteRule ^((?!public/).*)$ public/$1 [L,NC]
</IfModule>
```

## vi. Dernières tâches

### 1. Exécution de toutes les migrations de bases de données

```
$ php bin/console doctrine:migrations:migrate
```

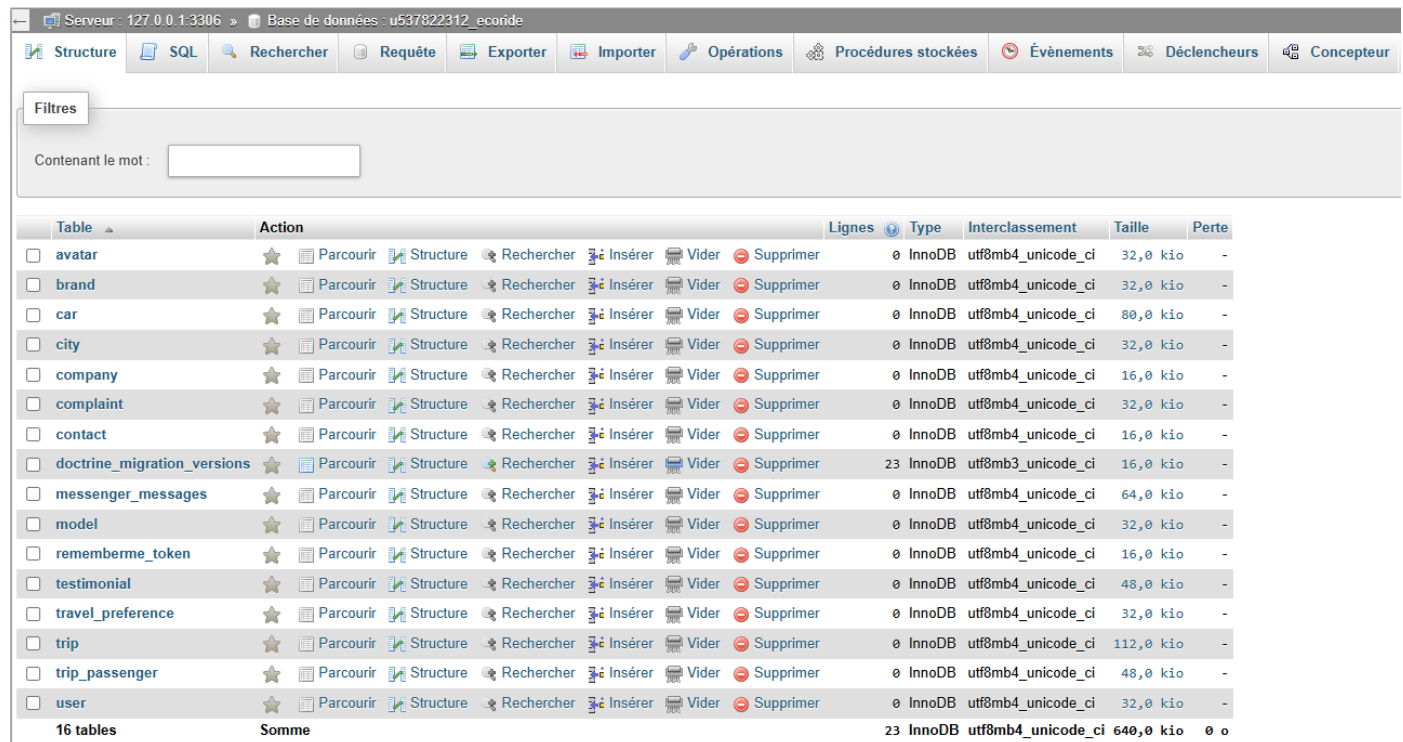


Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> avatar	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> brand	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> car	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	80,0 kio	-
<input type="checkbox"/> city	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> company	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
<input type="checkbox"/> complaint	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> contact	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
<input type="checkbox"/> doctrine_migration_versions	★ Parcourir Structure Rechercher Insérer Vider Supprimer	23	InnoDB	utf8mb3_unicode_ci	16,0 kio	-
<input type="checkbox"/> messenger_messages	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	64,0 kio	-
<input type="checkbox"/> model	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> rememberme_token	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
<input type="checkbox"/> testimonial	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
<input type="checkbox"/> travel_preference	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
<input type="checkbox"/> trip	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	112,0 kio	-
<input type="checkbox"/> trip_passenger	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
<input type="checkbox"/> user	★ Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
16 tables	Somme	23	InnoDB	utf8mb4_unicode_ci	640,0 kio	0 o

### 2. On copie l'intégralité du fichier sql\ecoride\_donnees.sql



Exécuter une ou des requêtes SQL sur la base de données « u537822312\_ecoride »:

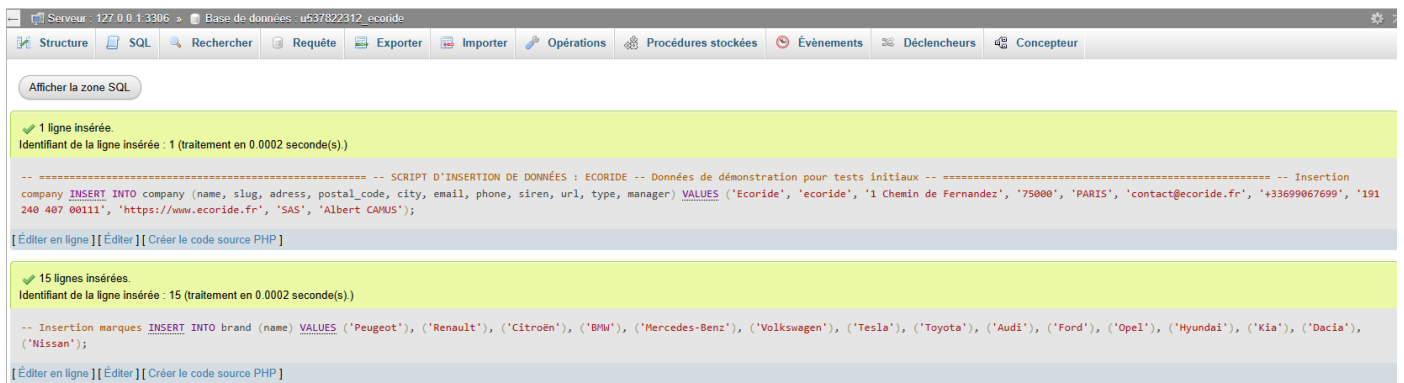
```
1
2 -- =====
3 -- SCRIPT D'INSERTION DE DONNÉES : ECORIDE
4 -- Données de démonstration pour tests initiaux
5 -- =====
6
7 -- Insertion company
8 INSERT INTO company (name, slug, adress, postal_code, city, email, phone, siren, url, type, manager) VALUES
9 ('Ecoride', 'ecoride', '1 Chemin de Fernandez', '75000', 'PARIS', 'contact@ecoride.fr', '+33699067699', '191 240 407 00111', 'https://www.ecoride.fr', 'SAS', 'Albert CAMUS');
10
11 -- Insertion marques
12 INSERT INTO brand (name) VALUES
13 ('Peugeot'), ('Renault'), ('Citroën'), ('BMW'), ('Mercedes-Benz'), ('Volkswagen'), ('Tesla'), ('Toyota'), ('Audi'), ('Ford'), ('Opel'), ('Hyundai'), ('Kia'), ('Dacia'), ('Nissan');
14
15 -- Insertion modèles
```

Effacer Format Récupérer la requête auto-sauvegardée

☐ Lier les paramètres

Délimiteur : ☐ Afficher à nouveau la requête après exécution ☐ Conserver la boîte de requêtes ☐ ROLLBACK à la fin ☒ Activer la vérification des clés étrangères Exécuter

Nous cliquons sur « Exécuter »



### 3. Compilation des assets

```
$ php bin/console asset-map:compile
```

```
// Compiling and writing asset files to public
// Compiled 103 assets
// Manifest written to public/assets/manifest.json
// Import map data written to public/assets/importmap.json.
// Entrypoint metadata written for 8 entrypoints (app, register, resetpw, updatepw,
// deleteAccount, admin_pseudo, password_utils, admin_password_generator).
```

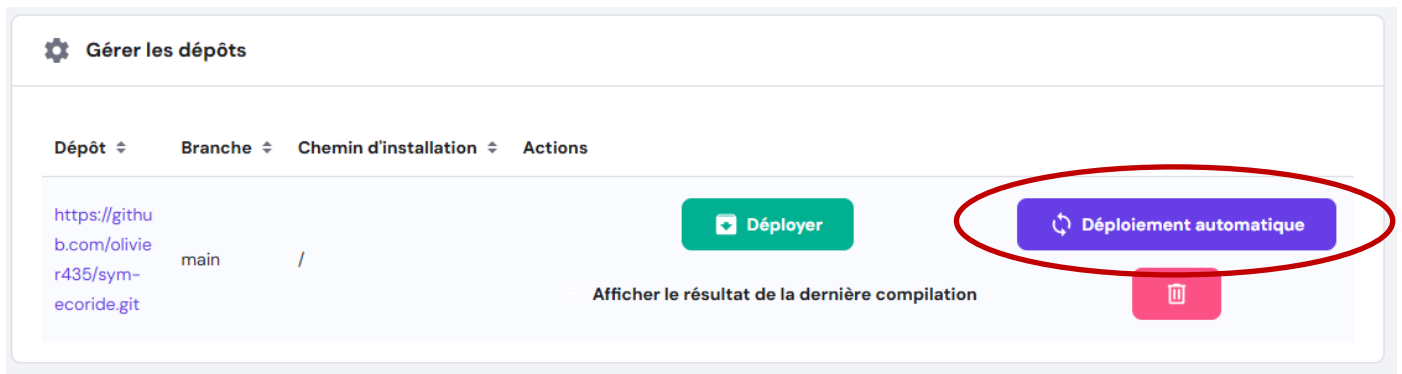
### 4. Videz le cache Symfony

```
$ APP_ENV=prod APP_DEBUG=0 php bin/console cache:clear
```

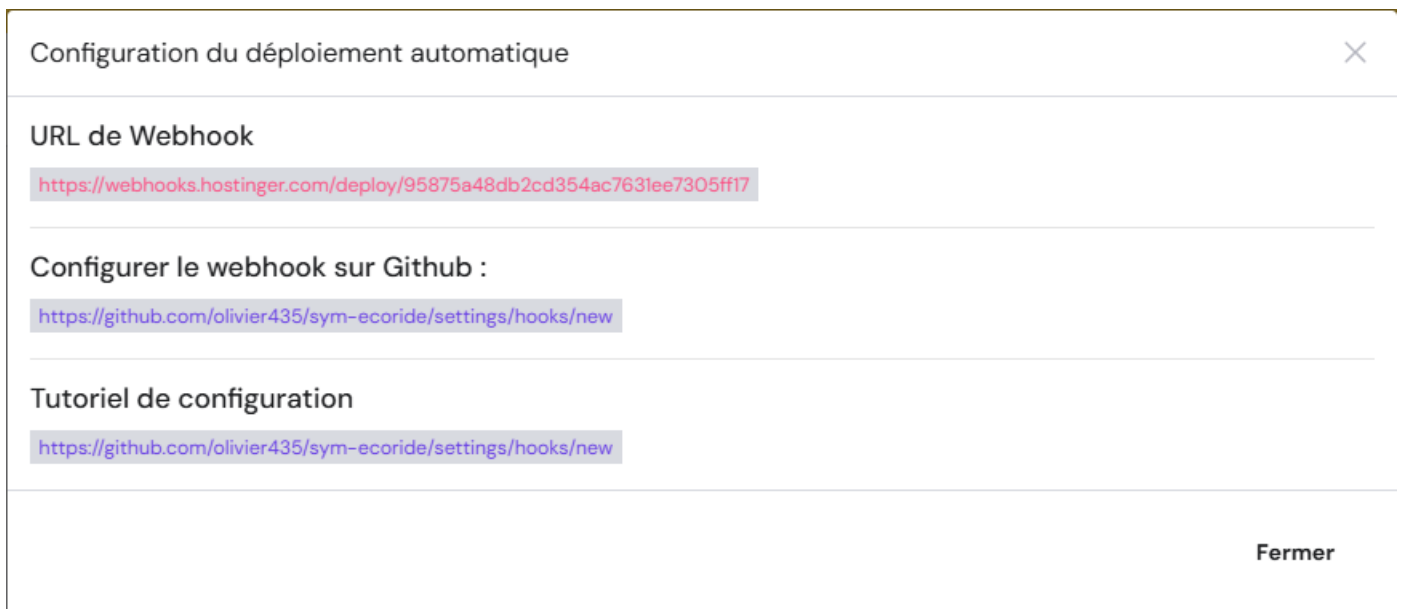


## D. Mise en place des déploiements automatiques

Nous allons pouvoir mettre en place les déploiements automatiques à l'aide notamment de Webhook. Dans le hPanel, on clique sur **Avancé** puis sur **GIT** :

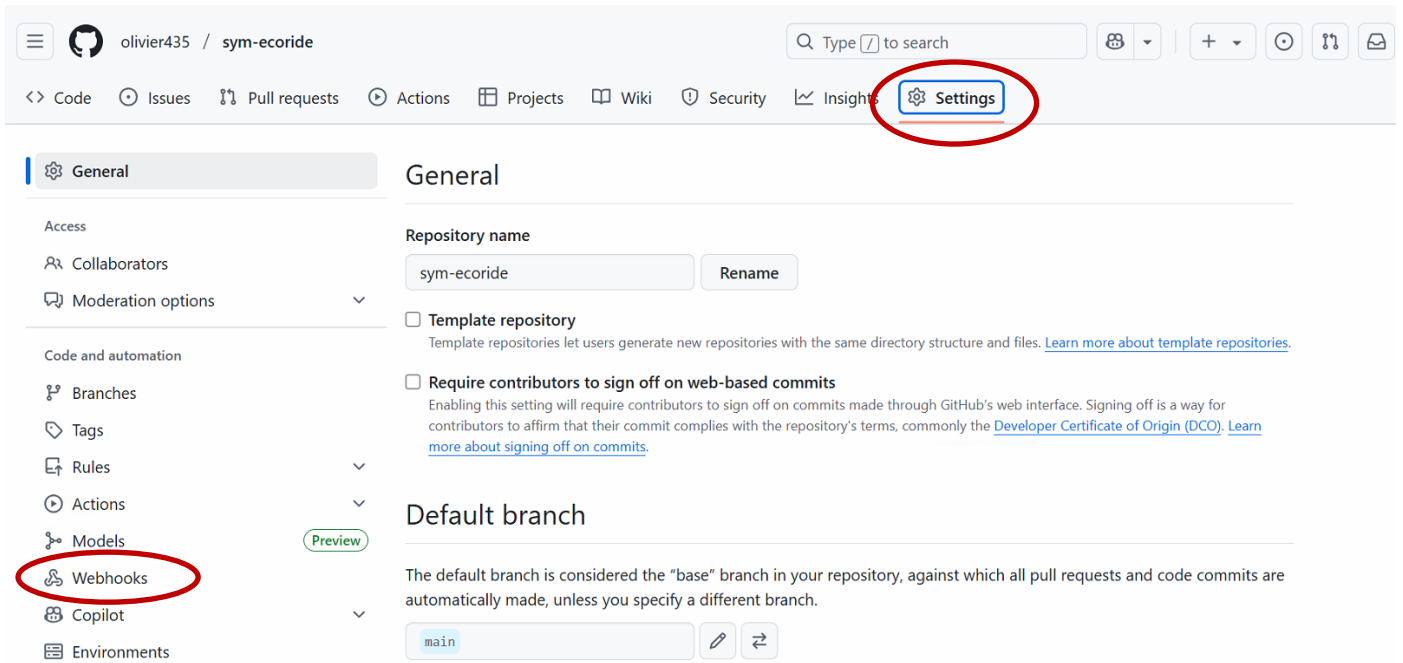


Il nous propose une configuration sur sur GitHub :



Nous allons pouvoir comme ça intégrer un déploiement automatique à chaque fois que nous « pushons » un nouveau fichier.

Nous nous rendons sur notre dépôt Github, « sym-ecoride », nous cliquons sur « **Settings** » dans la navigation et dans le menu latéral, nous cliquons sur « **Webhooks** » :



Nous sommes redirigés vers cette page et nous allons cliquer sur « **add Webhook** » :

## Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

Nous configurons en copiant-collant l'URL de Webhook fournie par Hostinger :

## Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

### Payload URL \*

### Content type \*

### Secret

### SSL verification

 By default, we verify SSL certificates when delivering payloads.

☒ Enable SSL verification ☐ Disable (not recommended)

### Which events would you like to trigger this webhook?

☒ Just the push event

Nous devons arriver à ce résultat :

## Webhooks

[Add webhook](#)



Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

● <https://webhooks.hostinger.com/de...> (push)

[Edit](#)[Delete](#)

This hook has never been triggered.

Nous réalisons un push et dans le « Gestionnaire de Fichiers » d'Hostinger :

🏠 > public_html > sql			
Nom ↑	Taille	Dernière modification	
 ecoride_donnees.sql	7.84 KiB	il y a quelques secondes	-rw-r--r--
 ecoride_structure.sql	6.09 KiB	il y a 3 heures	-rw-r--r--

## E. Configuration des tâches CRON

Dans Hostinger, le chemin complet vers PHP CLI est souvent :

```
/usr/bin/php
```

Et le projet est ici :

```
/home/u537822312/domains/covoiturage-ecoride.com/public_html
```

Nous allons vérifier les noms exacts des commandes avec :

```
$ php bin/console list | grep app:
```

```
app:test:mongo           Teste la connexion à MongoDB et la présence
de l'extension
app:trip-passengers:auto-validate  Valide automatiquement les passagers si
aucune réclamation après 1h du trajet terminé
app:trips:auto-complete  Passe automatiquement les trajets "en cours"
à "effectué" si l'heure d'arrivée est atteinte
app:trips:auto-start     Passe automatiquement les trajets à "en
cours" si l'heure de départ est atteinte
```

On va donc créer 3 tâches CRON dans le hPanel avec les noms exacts des commandes.

### 1. Préparer le dossier des logs

```
$ mkdir -p /home/u537822312/logs
```

Cela va créer le dossier logs à la racine de ton compte (en dehors de **public\_html**) où les CRON vont enregistrer leurs sorties.

### 2. Test manuel avant CRON

Toujours tester une commande avant de l'ajouter au CRON :

```
$ cd /home/u537822312/domains/covoiturage-ecoride.com/public_html
php bin/console app:trips:auto-start --env=prod
```

### 3. Création du script qui exécute :

- Le **démarrage auto des trajets à venir** (toutes les minutes)
- Le **passage des trajets en cours à "effectué"** (toutes les minutes)
- La **validation auto des passagers** (uniquement à l'heure pile)

On édite le fichier en SSH :

```
$ nano /home/u537822312/run_ecoride_cron.sh
```

On colle le script suivant :

```
#!/bin/bash
# Script CRON EcoRide - exécute les commandes Symfony
# Démarrage et complétion des trajets toutes les minutes
# Validation automatique des passagers à 1'heure pile

PROJECT_DIR="/home/u537822312/domains/covoiturage-ecoride.com/public_html"
PHP_BIN="/usr/bin/php"

# 1. Démarrer les trajets à venir
$PHP_BIN $PROJECT_DIR/bin/console app:trips:auto-start --env=prod

# 2. Compléter les trajets en cours
$PHP_BIN $PROJECT_DIR/bin/console app:trips:auto-complete --env=prod

# 3. Valider automatiquement les passagers seulement à 1'heure pile
if [ "$(date +%M)" == "00" ]; then
    $PHP_BIN $PROJECT_DIR/bin/console app:trip-passengers:auto-validate --env=prod
fi
```

On sauvegarde (**CTRL+O**, Entrée) puis quitte (**CTRL+X**).

On donne les droits d'exécution

```
$ chmod +x /home/u537822312/run_ecoride_cron.sh
ls
```

```
composer.phar  domains  logs  run_ecoride_cron.sh
```

Dans le hPanel, on clique sur « **Avancé** » puis « **Tâches Cron** »

**Création d'une nouvelle tâche Cron**

☐ PHP ☒ Personnalisé

Commande à exécuter

Options communes (facultatif) ▼

minute  
Chaque minute (\*) ▼

heure  
Chaque heure (\*) ▼

jour  
Tous les jours (\*) ▼

mois  
Tous les mois (\*) ▼

weekDay  
Tous les jours de la semaine (\*) ▼

**Sauvegarder**

- On sélectionne **Personnalisé** dans Hostinger
- On met `/home/u537822312/run_ecoride_cron.sh`
- Pour la planification :
  - **Minutes** → \*
  - **Heures** → \*
  - **Jour** → \*
  - **Mois** → \*
  - **Jour de la semaine** → \*

Liste des tâches cron		
Heure ↕	Commande à exécuter ↕	Actions
*****	/home/u537822312/run_ecoride_cron.sh	<a href="#">Afficher le résultat</a> <a href="#">Effacer</a>

Il faut aussi s'assurer que le fichier n'a pas d'encodage Windows (\r\n).

```
$ sed -i 's/\r$//' /home/u537822312/run_ecoride_cron.sh
```

Après conversion, on vérifie les droits d'exécution :

```
$ chmod +x /home/u537822312/run_ecoride_cron.sh
```

On teste manuellement :

```
$ /home/u537822312/run_ecoride_cron.sh
```

Si ça tourne sans erreur → le CRON pourra l'exécuter correctement.

Puis on teste :

```
$ tail -f /home/u537822312/logs/*.log
```

```
==> /home/u537822312/logs/trips-auto-complete.log <==
[2025-08-13 18:49:03] Complétion trajets en cours
[2025-08-13 18:50:03] Complétion trajets en cours
[2025-08-13 18:51:03] Complétion trajets en cours
[2025-08-13 18:52:03] Complétion trajets en cours

==> /home/u537822312/logs/trips-auto-start.log <==
[2025-08-13 18:49:03] Démarrage trajets à venir
[2025-08-13 18:50:03] Démarrage trajets à venir
[2025-08-13 18:51:02] Démarrage trajets à venir
[2025-08-13 18:52:03] Démarrage trajets à venir
```

Test de la validation passagers immédiatement (simulation) : comme le script ne la lance que quand minute == 00, on peut exécuter la commande directement :

```
$ /usr/bin/php /home/u537822312/domains/covoiturage-ecoride.com/public_html/bin/console app:trip-passengers:auto-validate --env=prod
```