

Installation avec Docker

Instructions de Configuration

1. Prérequis

Pour commencer, assurez-vous d'avoir **Docker** installé sur votre système.

Vous n'avez pas besoin de connaissances préalables sur Docker pour suivre les prochaines étapes. Cependant, si vous souhaitez utiliser votre propre environnement, n'hésitez pas à le faire !

Ensuite, **clonez** le projet Castopod en utilisant la commande suivante :

```
git clone https://code.castopod.org/adaures/castopod.git
```

Créez un fichier `.env` avec la configuration minimale requise pour connecter l'application à la base de données et utiliser Redis comme gestionnaire de cache :

```
CI_ENVIRONMENT="development"
# Si défini sur développement, vous devez exécuter `npm run dev` pour démarrer le serveur d'assets statiques
vite.environment="development"

# Par défaut, cela est défini sur vrai dans la configuration de l'application.
# Pour le développement, cela doit être défini sur faux car c'est
# dans un environnement local
app.forceGlobalSecureRequests=false

app.baseURL="http://localhost:8080/"
media.baseURL="http://localhost:8080/"

admin.gateway="cp-admin"
auth.gateway="cp-auth"

database.default.hostname="mariadb"
```

```
database.default.database="castopod"
database.default.username="castopod"
database.default.password="castopod"
```

```
cache.handler="redis"
cache.redis.host = "redis"
```

```
# Vous ne souhaitez peut-être pas utiliser Redis comme gestionnaire de cache
# Commentez/supprimez les deux lignes ci-dessus et décommentez
# la ligne suivante pour le cache par fichier.
#cache.handler="file"
```

N'oubliez pas d'ajouter le référentiel que vous avez cloné à Docker Desktop dans Paramètres > Ressources > Partage de fichiers.

2. Développement à l'intérieur du Conteneur d'Application avec VSCode (Recommandé)

Si vous utilisez VSCode, vous pouvez configurer un conteneur de développement préconfiguré.

Pour ce faire, installez l'extension [VSCode Remote - Containers](#)

et exécutez la commande suivante :

```
Ctrl/Cmd + Shift + P > Ouvrir dans le conteneur
```

La fenêtre de VSCode se rechargera à l'intérieur du conteneur de développement. Attendez plusieurs minutes lors du premier chargement car il construit tous les services nécessaires.

Le conteneur de développement démarrera en exécutant le serveur PHP de Castopod. Pendant le développement, vous devrez démarrer le serveur de développement de [Vite](#) pour compiler le code TypeScript et les styles :

```
# exécutez le serveur de développement de Vite
npm run dev
```

Si le serveur PHP de Castopod ne fonctionne pas, vous pouvez le redémarrer en utilisant les commandes suivantes :

```
# exécutez le serveur Castopod
php spark serve - 0.0.0.0
```

Vous êtes maintenant prêt ! ☐☐

Vous êtes maintenant à l'intérieur du conteneur de développement, vous pouvez utiliser la console VSCode (Terminal > Nouveau terminal) pour exécuter n'importe quelle commande :

```
php -v

# Composer est installé
composer -V

# pnpm est installé
pnpm -v

# git est installé
git version
```

Pour plus d'informations, consultez les Conteneurs distants VSCode.

2-bis. Développement Hors du Conteneur d'Application(sans VScode)

Si vous préférez ne pas utiliser le devcontainer de VSCode, vous pouvez démarrer les conteneurs Docker manuellement. Accédez au dossier racine du projet et exécutez les commandes suivantes :

```
# démarre tous les services déclarés dans le fichier docker-compose.yml
# l'option -d démarre les conteneurs en arrière-plan
docker-compose up -d

# Voir tous les processus en cours d'exécution (vous devriez voir 3 processus en cours d'exécution)
docker-compose ps

# Alternativement, vous pouvez vérifier tous les processus Docker
docker ps -a
```

La commande `docker-compose up -d` démarrera 4 conteneurs en arrière-plan :

- `castopod_app` : un conteneur basé sur PHP avec les exigences de Castopod installées
- `castopod_redis` : une base de données Redis pour gérer les requêtes et le cache des pages
- `castopod_mariadb` : un serveur MariaDB pour les données persistantes
- `castopod_phpmyadmin` : un serveur phpMyAdmin pour visualiser la base de données MariaDB.

Exécutez n'importe quelle commande à l'intérieur des conteneurs en les préfixant par `docker-compose run --rm app` :

```
# use PHP
docker-compose run --rm app php -v

# use Composer
docker-compose run --rm app composer -V

# use pnpm
docker-compose run --rm app pnpm -v

# use git
docker-compose run --rm app git version
```

3. Commencez à Développer

Vous êtes maintenant prêt à commencer le développement. Utilisez la console VSCode pour exécuter des commandes telles que `php -v`, `composer -V`, `npm -v`, ou `git version` selon vos besoins.

Pour voir vos modifications, rendez-vous sur :

- `http://localhost:8080/` pour le site web Castopod
- `http://localhost:8080/cp-admin` pour l'interface d'administration de Castopod :
 - Adresse e-mail : **admin@castopod.local**
 - Mot de passe : **castopod**
- `http://localhost:8888/` pour l'interface de phpMyAdmin :
 - Nom d'utilisateur : **castopod**
 - Mot de passe : **castopod**

Pour des information complémentaire visiter la [documentation Castopod](#)