docker-compose.yml

Cette configuration définit un ensemble de services Docker qui sont utilisés pour déployer l'application Castopod, une plateforme de gestion de podcasts.

```yaml
version: "3.7"

services:
  app:
    image: castopod/app:latest
    container_name: "castopod-app"
    volumes:
      - castopod-media:/opt/castopod/public/media
    environment:
      MYSQL_DATABASE: castopod
      MYSQL_USER: castopod
      MYSQL_PASSWORD: changeme
      CP_BASEURL: "https://lesconteslinux.mindcast.fr"
      CP_ANALYTICS_SALT: changeme
      CP_CACHE_HANDLER: redis
      CP_REDIS_HOST: redis
    networks:
      - castopod-app
    restart: unless-stopped

  redis:
    image: redis:7.0-alpine
    container_name: "castopod-redis"
    volumes:
      - castopod-cache:/data
    networks:
      - castopod-app

  # this container is optional
  # add this if you want to use the videoclips feature
  video-clipper:
    image: castopod/video-clipper:latest
    container_name: "castopod-video-clipper"
    volumes:
      - castopod-media:/opt/castopod/public/media
    environment:
      MYSQL_DATABASE: castopod
      MYSQL_USER: castopod
      MYSQL_PASSWORD: changeme
```

```
    restart: unless-stopped

volumes:
  castopod-media:
  castopod-cache:

networks:
  castopod-app:
```

Les services déclarés sont les suivants :

1. Le service **“app”** :

- Utilise l'image Docker **“castopod/app:latest”** pour exécuter l'application Castopod.
- Le nom du conteneur est défini comme **“castopod-app”**.
- Montre un volume nommé **“castopod-media”** dans le répertoire **“/opt/castopod/public/media”** du conteneur, qui est utilisé pour stocker les fichiers médias des podcasts.
- Définit plusieurs variables d'environnement liées à la base de données MySQL utilisée par Castopod, y compris le nom de la base de données, l'utilisateur et le mot de passe.
- Définit également d'autres variables d'environnement, telles que l'URL de base de Castopod, le sel pour les analyses, le gestionnaire de cache utilisé (Redis), et l'hôte Redis.
- Appartient au réseau **“castopod-app”**.
- Le redémarrage du conteneur est défini sur **“unless-stopped”**, ce qui signifie qu'il sera redémarré automatiquement sauf s'il est arrêté explicitement.

2. Le service **“redis”** :

- Utilise l'image Docker **“redis:7.0-alpine”** pour exécuter une instance Redis, qui est utilisée par Castopod comme gestionnaire de cache.
- Le nom du conteneur est défini comme **“castopod-redis”**.
- Montre un volume nommé **“castopod-cache”** dans le répertoire **“/data”** du conteneur, qui est utilisé pour stocker les données de cache de Redis.
- Appartient également au réseau **“castopod-app”**.

3. Le service **“video-clipper”** (optionnel) :

- Utilise l'image Docker **“castopod/video-clipper:latest”** pour exécuter un conteneur dédié à la fonctionnalité de découpage vidéo de Castopod.
- Le nom du conteneur est défini comme **“castopod-video-clipper”**.
- Montre le même volume **“castopod-media”** que le service **“app”**, ce qui permet au conteneur d'accéder aux fichiers médias des podcasts.
- Définit les mêmes variables d'environnement pour la base de données MySQL que le service **“app”**.
- Le redémarrage du conteneur est également défini sur **“unless-stopped”**.

En outre, la configuration définit deux volumes Docker nommés **“castopod-media”** et **“castopod-cache”**, qui sont utilisés pour stocker respectivement les fichiers médias des podcasts et les données de cache de Redis. De plus, un réseau nommé **“castopod-app”** est créé pour permettre la communication entre les services.

Cette configuration permet donc de déployer l'application Castopod avec une base de données

MySQL et une instance Redis pour la gestion du cache.

Concernant les variables à modifier

Voici les variables modifiables présentes dans la section `environment` de chaque service :

Pour le service “**app**” :

- `MYSQL_DATABASE` : Nom de la base de données MySQL utilisée par Castopod.
- `MYSQL_USER` : Nom d'utilisateur MySQL pour Castopod.
- `MYSQL_PASSWORD` : Mot de passe MySQL pour Castopod.
- `CP_BASEURL` : URL de base pour Castopod.
- `CP_ANALYTICS_SALT` : Sel utilisé pour les analyses.

Pour le service “**video-clipper**” (optionnel) :

- `MYSQL_DATABASE` : Nom de la base de données MySQL utilisée par Castopod.
- `MYSQL_USER` : Nom d'utilisateur MySQL pour Castopod.
- `MYSQL_PASSWORD` : Mot de passe MySQL pour Castopod.

Vous pouvez modifier ces variables en fonction de vos besoins spécifiques en leur attribuant les valeurs souhaitées. Par exemple, vous pouvez changer le nom de la base de données, l'utilisateur et le mot de passe MySQL, ainsi que l'URL de base pour Castopod, le sel pour les analyses. Assurez-vous de respecter les contraintes et les configurations requises par l'application Castopod lors de la modification de ces variables.

Concernant les volumes

Pour définir les volumes dans la configuration Docker Compose, vous utilisez la section `volumes` du fichier YAML. Voici comment faire :

```yaml
version: "**3.7"

services:
  app:
    image: castopod/app:latest
    container_name: "**castopod-app"
    volumes:
      - castopod-media:/opt/castopod/public/media
    # ...
  redis:
    image: redis:7.0-alpine
    container_name: "**castopod-redis"
    volumes:
      - castopod-cache:/data
    # ...

volumes:
```

```
  castopod-media:
  castopod-cache:
```

Dans cet exemple, nous avons ajouté la section `volumes` à la fin du fichier YAML. À l'intérieur de cette section, vous déclarez vos volumes en utilisant un nom de volume suivi des deux-points `:`. Les noms de volume ici sont “**castopod-media**” et “**castopod-cache**”.

Ensuite, vous pouvez référencer ces volumes dans la section `volumes` des services concernés en utilisant la syntaxe `<nom du volume>:<chemin du montage dans le conteneur>`. Dans l'exemple ci-dessus, le volume “**castopod-media**” est monté dans le répertoire `/opt/castopod/public/media` du conteneur du service “**app**”, et le volume “**castopod-cache**” est monté dans le répertoire `/data` du conteneur du service “**redis**”.

En définissant les volumes de cette manière, Docker va créer les volumes persistants nécessaires pour les conteneurs et les associer à leurs répertoires respectifs à chaque exécution.

N'oubliez pas que si vous avez plusieurs services utilisant les mêmes volumes, ils auront accès aux mêmes données persistantes, ce qui peut être utile pour le partage de données entre les conteneurs.

Voici un exemple de configurations que vous pouvez utiliser pour les volumes “**castopod-media**” et “**castopod-cache**” dans votre fichier Docker Compose :

1. Utilisation d'un chemin local sur la machine hôte :

```
volumes:
  castopod-media:
    driver: local
    driver_opts:
      type: none
      o: bind
      device: /chemin/vers/le/dossier/media
  castopod-cache:
    driver: local
    driver_opts:
      type: none
      o: bind
      device: /chemin/vers/le/dossier/cache
```

Dans cet exemple, nous utilisons des volumes de type “**local**” pour lier des dossiers locaux sur la machine hôte aux conteneurs. Vous devez remplacer “**/chemin/vers/le/dossier/media**” et “**/chemin/vers/le/dossier/cache**” par les chemins réels vers les dossiers que vous souhaitez utiliser pour stocker les données du volume.

2. Utilisation de volumes nommés :

```
volumes:
  castopod-media:
  castopod-cache:
```

Dans cet exemple, nous déclarons simplement les volumes “**castopod-media**” et “**castopod-cache**” sans spécifier de configuration supplémentaire. Dans ce cas, Docker va créer et gérer les

volumes automatiquement dans un emplacement par défaut sur le système de fichiers du système d'hébergement Docker.

Concernant l'option restart

Dans le contexte de la configuration Docker Compose, la ligne `restart: unless-stopped` est spécifiée pour les services “**app**” et “**video-clipper**”.

Lorsque vous utilisez `restart: unless-stopped`, cela signifie que Docker va automatiquement redémarrer le conteneur en cas d'arrêt, sauf si vous arrêtez explicitement le conteneur manuellement en utilisant une commande Docker, par exemple `docker stop castopod-app`.

Cela garantit que le conteneur est toujours en cours d'exécution, sauf si vous décidez de l'arrêter de manière explicite. Cela peut être utile pour s'assurer que les services sont toujours disponibles et fonctionnent de manière continue, même après un redémarrage du système ou un arrêt inattendu.

Cette option de redémarrage automatique peut être configurée avec d'autres valeurs, telles que `always`, `on-failure`, `unless-stopped`.