Nantes Université

Master Informatique *1er année*

Module MP :XMA1IU3042 Développement, données, exploitation

Rapport Boardgamemanager

Abdelillah El Khotri , Nicolas Gouget

19 décembre 2023

Table des matières

1 Analyse **3**
1.1 Découpage de l'application . . . 3
1.2 Les images à construire . . . 3
1.2.1 Construction de la première image . . . 3
1.2.2 Utilisation de la première image . . . 3
1.2.3 Construction de la seconde image . . . 4
1.2.4 Utilisation de la seconde image . . . 4
1.3 Le déploiement des conteneurs . . . 4
1.3.1 Les réseaux virtuels . . . 4
1.3.2 Les montages de répertoire . . . 4
1.3.3 Les variables d'environnements . . . 5
1.3.4 Les publications de ports . . . 5
1.4 Synthèse des déploiements . . . 5

2 Technique **5**
2.1 Front . . . 5
2.2 Back . . . 6
2.3 Base de données . . . 6

3 Conclusion **6**

1 Analyse

1.1 Découpage de l'application

Nous avons décidé de déployer le boardgamemanager en utilisant 3 conteneurs :

- un conteneur pour le Front avec une image nginx.
  Va servir a host partie web du boardgamemanager celle avec laquelle l'utilisateur interagi.

- Un conteneur pour le Back avec une image maven pour compiler.
  Va servir pour host l'API les requêtes arriverons sur ce conteneur et il servira de pont entre le front et la base de données.

- Et un conteneur pour la Base de données qui utilisera une image Postgres.
  Va servir à stocker les données reçues par le back

1.2 Les images à construire

Pour réalisé le déploiement, nous allons procéder à la réalisation de deux images de conteneurs.

1.2.1 Construction de la première image

La 1re image que nous allons créer est celle du backend :

- On utilisera maven en image de base car le back du boardgamemanager est écrit en java et contient un fichier pom.xml pour la compilation. Plus précisément, on utilise maven avec l'étiquette 3-eclipse-temurin-17. Le code étant compatible java 17 et eclipse temurin est une machine virtuelle java performante.
- Notre image va récupérer les fichiers nécessaires à la construction du back. Puis compiler les fichiers avec maven. Pour enfin lancer le serveur Java qui contient l'API nécessaire à la gestion du boardgamemanager.
- Il nous faut donc inclure le dossier backend dans l'image. Car ce dernier contient tous les fichiers dont nous aurons besoin.
- Pour limiter la taille de l'image, on peut supprimer les fichiers inutiles après compilation ou utiliser une certaine image pour compiler et une autre image plus légère avec juste de quoi lancer le serveur.
- Notre point d'entrée dans l'image est l'exécution du serveur java en mode production

1.2.2 Utilisation de la première image

Pour utiliser notre image, il y a quelques paramètres à ajuster :

- Dans une optique d'isolation et de respect de philosophie des conteneurs, il faut connecter le backend a deux réseaux. Un pour le lien avec le front et un autre pour communiquer avec la base de données. Dans notre cas, ils s'appellent data_network et back_network. On aura donc –network data_network et –network back_network

- Il est aussi nécessaire d'effectuer une publication de port pour que le front puisse appeler l'API. Le port utilisé est le 8080. On aura donc –publish 8080 :8080 (Le premier port sera celui du front qui associe son port 8080 a celui du back. On aurait aussi pu avoir un autre numéro à droite.).

1.2.3 Construction de la seconde image

La 2e image que nous allons créer est celle du frontend :

- On utilisera nginx en image de base, car le front du boardgamemanager est un site web, nginx nous permet donc de host le site internet. Plus précisément, on utilise nginx avec l'étiquette mainline-alpine. On utilise une version alpine plus légère.
- Notre image doit installer de quoi compilé le code du front. Puis va récupérer les fichiers nécessaires à la construction du front. Ensuite, on compile les fichiers avec npm. Pour enfin mettre les fichiers compilés dans les dossiers de nginx et on laisse la commande d'exécution par défaut de nginx.
- Il nous faut donc inclure le dossier frontend dans l'image. Car ce dernier contient tous les fichiers dont nous aurons besoin.
- Pour limiter la taille de l'image, on peut supprimer les fichiers inutiles après compilation ou désinstaller les dépendances installées pour compiler.
- Notre point d'entrée dans l'image est l'exécution par défaut de nginx

1.2.4 Utilisation de la seconde image

Pour utiliser notre image, il y a quelques paramètres à ajuster :

- Dans une optique d'isolation et de respect de philosophie des conteneurs, il faut connecter le frontend a un réseau, pour communiquer avec le back. Dans notre cas, ils s'appellent back_network. On aura donc –network back_network
- Il est aussi nécessaire d'effectuer une publication de port pour que l'utilisateur puisse accéder au boardgamemanager. Le port utilisé est le 80. On aura donc –publish 8480 :8080 (Le premier port sera celui du serveur de déploiement qui associe son port 8080 a celui du back. On aurait aussi pu avoir un autre numéro à droite selon qui exécute l'image.).

1.3 Le déploiement des conteneurs

Pour déployer les 3 images, il y a quelques paramètres à définir.

1.3.1 Les réseaux virtuels

Il faut créer deux réseaux virtuels : data_network et back_network. Comme dit précédemment le front et le back sont connecté a back_network et le back et la base de données a data_network.

1.3.2 Les montages de répertoire

Pour la base de données, il faut effectuer un montage de fichier entre un dossier data du coter server podman et /var/libpostgresql/data du coter du conteneur.

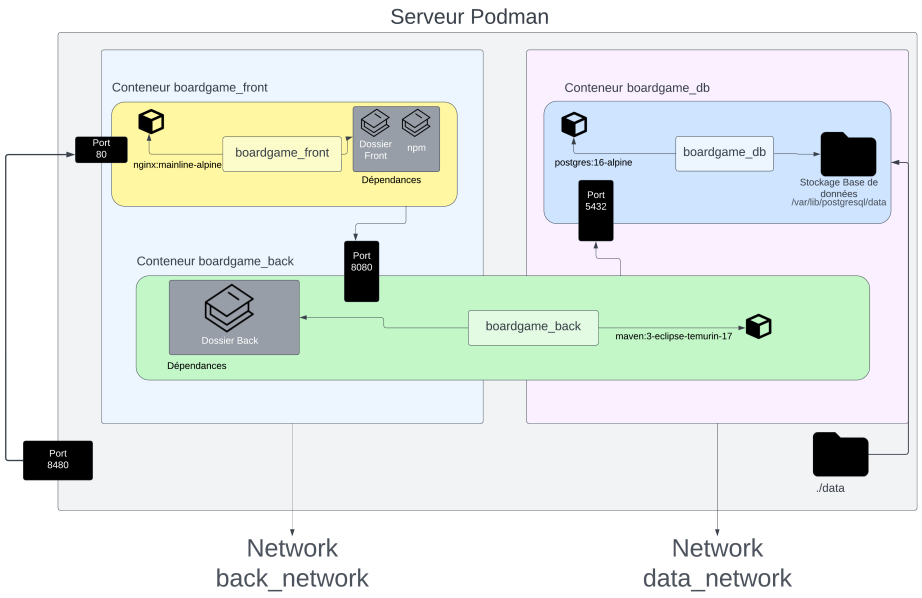
1.3.3 Les variables d'environnements

Pour le conteneur de base de données, il faut confiurer POSTGRES_USER POSTGRES_PASSWORD et POSTGRES_DB. Attention selon les valeurs choisie, il faut adapter les fichiers de configurations du backend.

1.3.4 Les publications de ports

Le conteneur du front doit avoir le port 80 publié, le back le port 8080 et la base de données le port 5432. Les ports associés côté utilisateur sont les mêmes par défaut à l'exception du front qui est nous est spécifique dû à la restriction de l'université. Mais tout les mappage peuvent être changé par l'utilisateur.

1.4 Synthèse des déploiements



2 Technique

2.1 Front

Pour utiliser l'image boardgamemanager_front il faut forcément effectuer une publication de port avec le port 80. Et connecter le conteneur au même réseau que le backend.

```
// Exemple de commande pour utiliser l'image boardgamemanager_front

podman container run --rm \
        --detach \
        --name boardgame_front \
        --network back_network \
        --publish 8480:80 \
        boardgamemanager_front:v1
```

2.2 Back

Pour utiliser l'image boardgamemanager_backend il faut effectuer une publication avec le port 8080. Le connecter aux mêmes réseaux que le front et la base de données.

```
// Exemple de commande pour utiliser l'image
    boardgamemanager_backend

podman container run --rm \
        --detach \
        --name boardgame_back \
        --network data_network \
        --network back_network \
        --publish 8080:8080 \
        boardgamemanager_backend:v1
```

2.3 Base de données

Pour lancer la base de données, il faut effectuer une publication avec le port 5432. Le conteneur doit être connecté au même réseau que le backend. Il faut également monter un volume pour stocker les données et configurer les variables d'environnement de postgres (POSTGRES_USER, POSTGRES_PASSWORD, POSTGRES_DB).

```
// Exemple de commande pour lancer la bdd
podman container run --rm \
                    --detach \
                    --name boardgame_db \
                    --network data_network \
                    --volume ./data:/var/lib/postgresql/data \
                    --publish 5432:5432 \
                    --env POSTGRES_USER=user1 \
                    --env POSTGRES_PASSWORD=pwd1234 \
                    --env POSTGRES_DB=board_db \
                    docker.io/postgres:16-alpine
```

3 Conclusion

Les images on été créer pour simplifier, mais ne pas limiter l'utilisation des images par des utilisateurs tiers. Des améliorations sont possibles notamment sur la taille des images et la possibilité de configuration pour des usages différents.

En l'état, tous les conteneurs peuvent être lancés et permettent d'utiliser toutes les fonctionnalités du boardgamemanager sans aucun souci.