DOCUMENTATION TECHNIQUE

PROJET SQUARE – INGENIEUR INFORMATIQUE II 2019-2020

INFORMATIONS

Nom du projet	**Square**
Type de document	Documentation technique
Date	02/12/2019
Version	1.0
Mots-clés	Java - Quarkus - Docker - Conteneurisation - Déploiement - Automatisation – Persistance - REST – http
Auteurs	FAU Nicolas – nfau1 MARTI Emilie - emarti

TABLE DES MATIERES

1 – RESUME DU DOCUMENT

Ce document est la documentation technique du programme Square, réalisé dans le cadre du cours de Java Avancé de la deuxième année du cursus d'Ingénieur Informatique à l'ESIPE, Marne la Vallée.

Ce document est divisé en trois parties :

- Documentation technique du serveur : Application Square ;
- Documentation technique de la librairie cliente ;
- Bugs connus du programme.

2 - FONCTIONNEMENT DE L'APPLICATION SQUARE

2.1 DESCRIPTION DU LOGICIEL

Le projet Square a pour but de créer un logiciel qui va être capable d'automatiser le déploiement et le contrôle de conteneurs d'applications en utilisant l'outil de conteneurisation Docker.

Le programme doit être également en mesure de sauvegarder des informations sur les applications conteneurisées sous la forme de logs. Pour ce faire, une librairie cliente est déployée dans le docker et va communiquer avec l'application Square via des services REST.

Le logiciel permet à l'utilisateur de gérer dynamiquement des conteneurs Docker avec les applications désirées via des requêtes http. L'utilisateur peut ainsi demander le déploiement et l'arrêt de conteneurs mais aussi peut demander une gestion automatisée du nombre de conteneurs contenant l'application de son choix en route à tout moment.

Indépendamment de l'utilisateur, Square va sauvegarder à tout moment des informations sous forme de logs sur les applications existantes. Ces informations vont être persistées dans une base de données qui va permettre à Square de fonctionner dynamiquement et garder une trace des applications qu'il a pu gérer.

Ce logiciel a été testé et développé sur Linux 2.6 et sur Microsoft Windows 10 Professionnel.

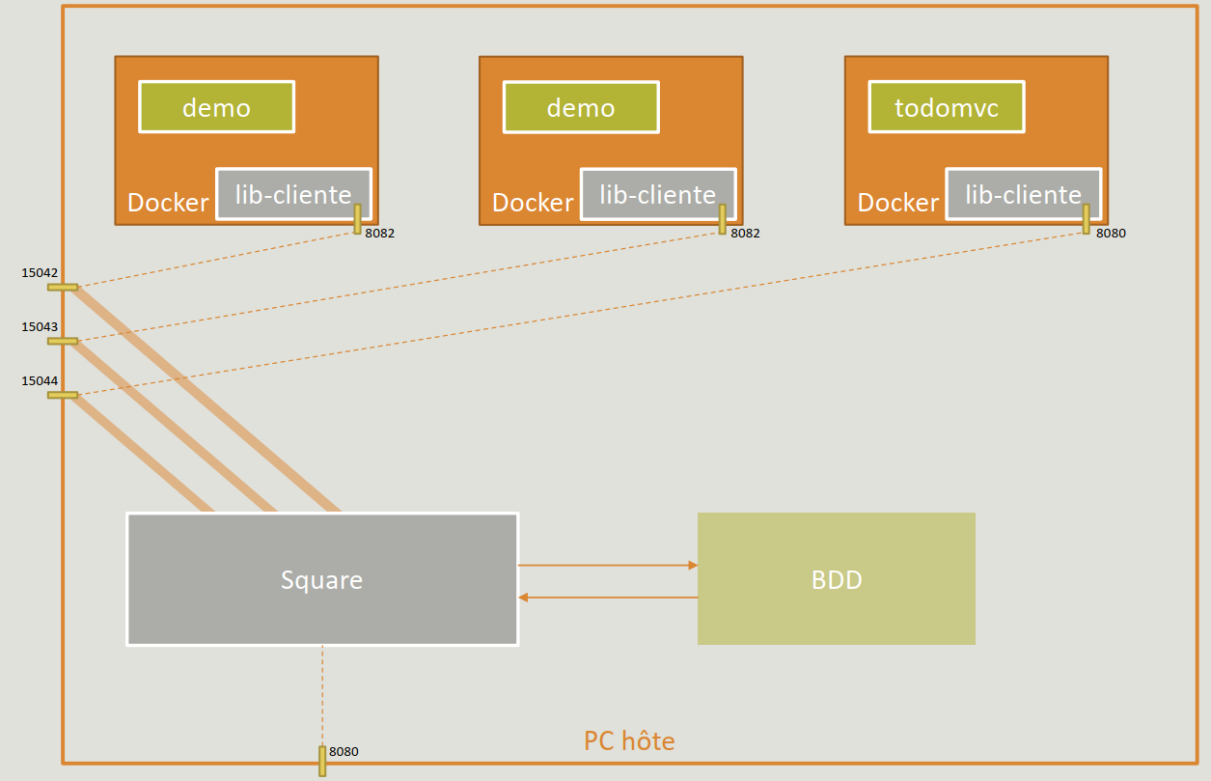
2.2 DECOMPOSITION DU PROJET

Le projet se décompose en différentes parties :

- L'application serveur ou Application Square, qui est l'application principale qui va gérer les conteneurs Docker depuis la machine hôte ;
- La librairie cliente, qui va servir de proxy dans le conteneur, se positionnant sur le port exposé du conteneur et en dialoguant avec l'application Square, gérant la persistance des logs et témoignant de la santé de l'application déployée.

Chaque partie sera développée dans la suite de ce document, dans les sections 3 et 4 respectivement.

2.3 ARCHITECTURE GLOBALE



Square est une application qui va tourner sur le port 8080 de la machine hôte. L'utilisateur va pouvoir communiquer avec Square via des requêtes de type GET et/ou POST vers le port 8080 de la machine hôte (par exemple avec Postman) afin de mettre en œuvre la gestion des conteneurs Docker correspondants.

Afin de pouvoir récupérer les logs, Square va exposer une route POST qui va permettre l'envoi de données de la librairie Cliente vers Square. La librairie cliente va se charger d'envoyer les logs vers Square périodiquement à travers ces routes, et cette dernière va sauvegarder ces informations dans la base de données.

3 - APPLICATION SQUARE

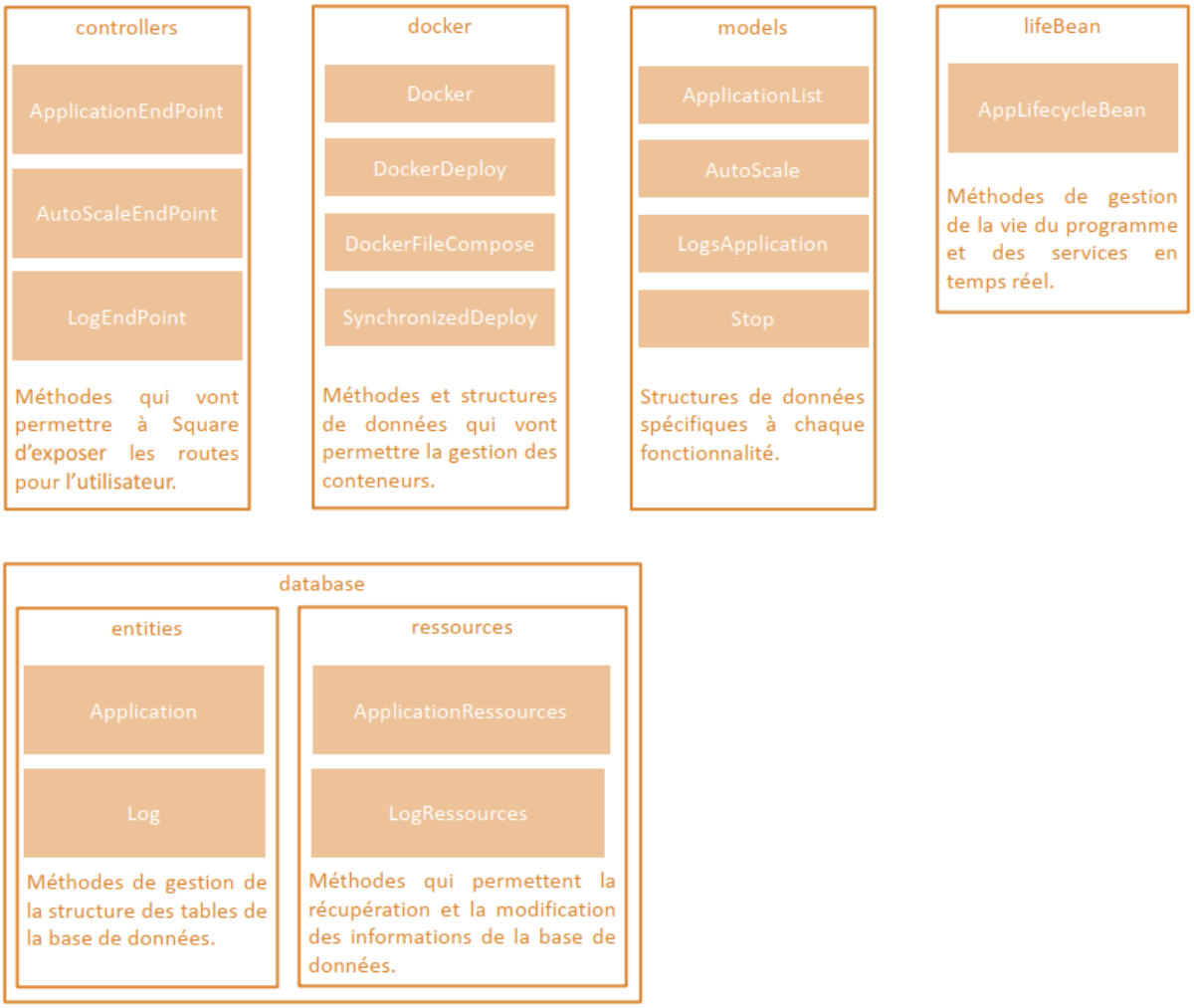
3.1 ARCHITECTURE

Cette application peut se diviser en plusieurs parties selon les fonctionnalités qu'elle propose :

- Gestion des applications
- Gestion des conteneurs
- Gestion des logs

En parallèle à ces trois grosses fonctions, Square va pouvoir exposer des routes à l'utilisateur afin qu'il puisse faire des requêtes pour récupérer des informations ou envoyer des instructions concernant les trois grosses fonctionnalités précédentes.

Ci-dessous la description générique des classes dans leurs package.



3.2 TECHNOLOGIES UTILISEES

3.2.1 QUARKUS

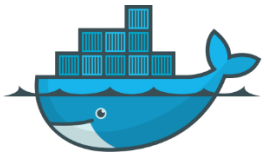
Le logiciel est codé en Java 11 et utilise le Framework Quarkus.

Quarkus est un framework natif full stack très récent qui est actuellement en version 1.0.1.Final. Il permet de créer très simplement des API Rest en Java et offre une utilisation simplifiée de l'ORM hibernate grâce à Hibernate-with-panache.

On appelle ORM un programme se plaçant comme une interface entre un programme applicatif et une base de données relationnelle. L'ORM possède des méthodes qui permettent l'ajout, la modification, la récupération ou encore la suppression d'éléments dans une base de données.

Panache est une couche ajoutée par Quarkus pour gérer les bases de données à l'aide d'hibernate.

3.2.2 DOCKER



L'outil utilisé pour conteneuriser nos applications est Docker.

Docker est logiciel libre qui permet d'empaqueter une application avec ses dépendances dans un conteneur isolé. Ce conteneur possèdera l'environnement le plus petit et nécessaire pour faire tourner l'application et pourra être exécuté sur n'importe quelle machine. Il s'appuie sur les fonctionnalités du système d'exploitation de la machine hôte pour fonctionner, contrairement aux machines virtuelles, qui ont leurs propres systèmes d'exploitation.

Docker est largement utilisé dans le monde de l'informatique pour de nombreuses raisons : un logiciel open source et gratuit, permettant une grande portabilité des applications et il est en développement continu depuis 2013. Il est multiplateforme et peut être utilisé sur Linux, Windows et MacOS.

Docker va servir à conteneuriser nos applications demo et todomvc et va être la raison principale d'existence de l'application Square, qui va devoir déployer et gérer les conteneurs docker.

3.3 STRUCTURE DU PROJET

La structure du projet va être détaillé selon les grandes fonctionnalités du projet.

3.3.1 GESTION DES APPLICATIONS

Les applications vont avoir une série d'informations qui vont être sauvegardées dans un objet de type *Application* ainsi que dans la base de données, pour avoir une trace persistante de l'existence de cette application.

Une application est associée à un port (lié au conteneur), un port de service (lié à l'hôte), une instance de docker ainsi qu'à un temps de vie. Une application est déployée dans un conteneur docker qui possède l'environnement nécessaire à son bon fonctionnement ainsi qu'une librairie cliente faite par nos soins dans ce projet, décrite ci-dessous.

3.3.2 GESTION DES CONTENEURS

Une application va être déployée dans un conteneur en utilisant l'outil Docker.

Le déploiement d'un conteneur se fait en deux étapes :

BUILD

Pour construire une image, il faut créer un dockerfile.

Ce dockerfile va avoir les informations importantes telles que l'image de base (openjdk:11), le port qui va être exposé ainsi que la copie du binaire de l'application dans le conteneur et sa ligne de commande pour l'exécuter. D'autres informations supplémentaires peuvent être rajoutées.

L'image peut être sauvegardée dans un dossier. Lorsqu'une image est créée, elle peut être réutilisée pour toutes les instances docker qui vont être lancées avec les mêmes informations.

RUN

On peut lancer un conteneur docker à partir d'une image construite en amont. L'image peut être chargée si on l'a sauvegardé dans un dossier sous la forme d'un tar.gz.

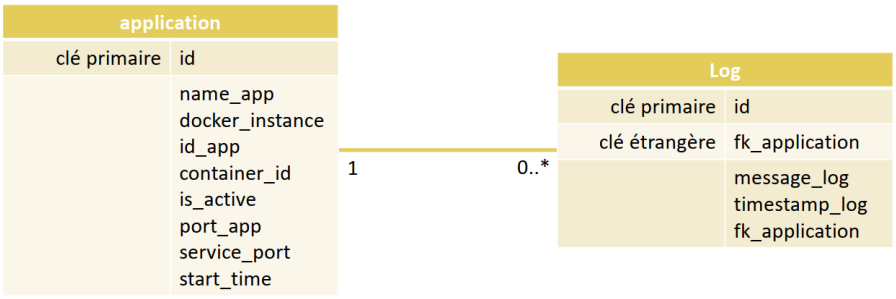
Les conteneurs peuvent être également listés et/ou arrêtés à la demande de l'utilisateur. L'auto-scale va permettre à l'utilisateur de programmer un nombre d'instances de docker nécessaires pour chaque application. L'autoscale, lorsqu'il est activé, va s'assurer qu'à tout moment ce nombre est respecté.

3.3.3 GESTIONS DES LOGS

Square est capable de persister des logs venant des applications qui tournent dans les instances de docker actives. L'application va exposer des routes qui vont être utilisées par la librairie cliente pour qu'elle puisse envoyer les logs périodiquement. Square va ensuite sauvegarder ces logs dans la base de données.

3.4 BASE DE DONNEES

La base de données utilisée dans l'application va permettre de sauvegarder les informations sur les applications et conteneurs gérés par Square ainsi que les logs qu'on va récupérer de ces applications.



3.5 SERVICES REST

Square permet à l'utilisateur de lui envoyer des instructions via des requêtes http.

3.5.1 DOCKER

URL	Type	Description
/app/deploy	POST	Déploie une instance de docker avec l'application donnée.
/app/list	GET	Renvoie la liste des instances de docker actives gérées par Square.
/app/stop	POST	Stoppe une instance de docker avec l'application donnée.

3.5.2 LOG

URL	Type	Description
/logs/:time	GET	Renvoie la liste des logs existants depuis <time> minutes.
/logs/:time/:filter	GET	Renvoie la liste filtrée des logs existants depuis <time> minutes.

3.5.3 AUTOSCALE

URL	Type	Description
/auto-scale/update	POST	Démarre l'autoscale ou modifie la configuration courante.
/auto-scale/status	GET	Renvoie la liste d'actions à réaliser à l'instant par l'autoscale.
/auto-scale/stop	GET	Stoppe le service d'autoscale.

4 - LIBRAIRIE CLIENTE

4.1 DESCRIPTION

Il s'agit d'une petite librairie déployée sur docker avec l'application qui va récupérer ses informations et les renvoyer sous forme de log vers Square, en utilisant les route que ce dernier lui a exposée. Cette librairie a été codée en Java 11.

4.3 STRUCTURE DU PROJET

Le projet est composé de trois classes.

APPLIFECYCLEBEAN

Cette classe va lancer un thread qui va créer des logs toutes les 15 secondes à partir de l'application du conteneur et va l'envoyer à Square.

LOG

Objet qui sert de structure de données pour un log, avec un message et un timestamp.

LOGREADER

Classe principale avec toutes les méthodes pour créer les logs à partir d'un message et d'un timestamp ainsi que pour les envoyer vers l'application Square, en utilisant l'adresse IP du système hôte et le port sur lequel tourne Square qu'on aura donné en tant que variable d'environnement au conteneur.