Virtualisation et Cloud

Application Java PetClinic construire avec Spring Boot

Membres du groupe

- ***Mazaba Predona Doden***
- ***Odzongo Alexis***

Superviseur

- **Dr.KébaGueye**

Objectif

- Cloner et exécuter une application Spring Boot avec Maven
- Créer un nouveau **Dockerfile** contenant les instructions nécessaires pour créer une image Java
- Configurer un environnement de développement local pour connecter une base de données au conteneur
- Utiliser Docker Compose pour exécuter l'application Spring Boot
- Configurez un pipeline CI/CD pour votre application à l'aide de GitHub Actions

Plan

I. Introduction

II. Création de l’application

III. Déploiement de l’application sur GitHub

IV. Conclusion

I. Introduction

Spring PetClinic est un exemple d'application open source développé pour démontrer les capacités orientées base de données de Spring Boot, Spring MVC et Spring Data Framework. Il est basé sur cette pile Spring et construit avec Maven.

II. Création de l'application

Etape 1 : Clonons notre application Pet Clinic.

Avec la commande qui suit :

```
git clone https://github.com/spring-projects/spring-petclinic.git
```

```
root@karlito-VirtualBox:/path/to/working/directory# git clone https://github.co
m/spring-projects/spring-petclinic.git
Clonage dans 'spring-petclinic'...
remote: Enumerating objects: 9452, done.
remote: Total 9452 (delta 0), reused 0 (delta 0), pack-reused 9452
Réception d'objets: 100% (9452/9452), 7.64 Mio | 807.00 Kio/s, fait.
Résolution des deltas: 100% (3578/3578), fait.
```

Etape 2 : Téléchargeons les dépendances liées a notre application

Avec la commande qui suit :

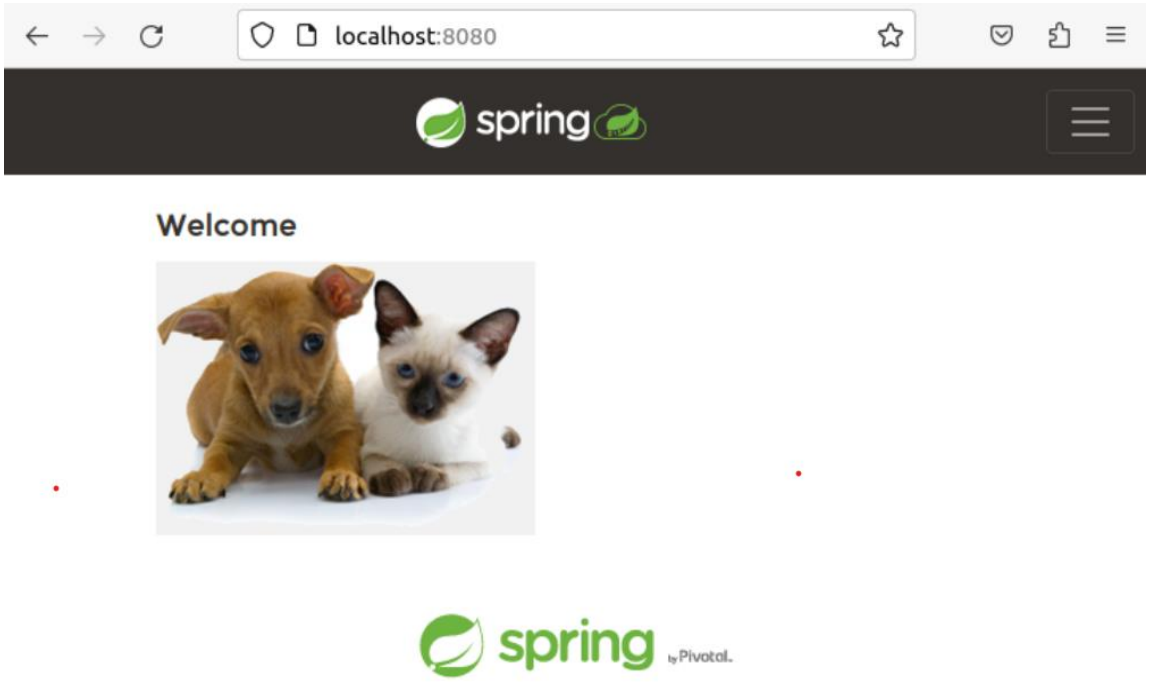
```
./mvnw spring-boot:run
```

```
root@karlito-VirtualBox:/path/to/working/directory/spring-petclinic# ./mvnw spr
ing-boot:run
[INFO] Scanning for projects...
Downloading from spring-snapshots: https://repo.spring.io/snapshot/org/springfr
amework/boot/spring-boot-starter-parent/3.0.1/spring-boot-starter-parent-3.0.1.
pom
Downloading from spring-milestones: https://repo.spring.io/milestone/org/spring
```

Etape 3 : Testons notre application

Ouvrons notre navigateur et tapons :

http://localhost:8080



Etape 4 : Construisons un **Dockerfile** et définissons notre image de base

Avece la commande qui suit :

Nano Dockerfile

```
  GNU nano 4.8                          Dockerfile
# syntax=docker/dockerfile:1

FROM eclipse-temurin:17-jdk-jammy

WORKDIR /app

COPY .mvn/ .mvn
COPY mvnw pom.xml ./
RUN ./mvnw dependency:resolve

COPY src ./src

CMD ["./mvnw", "spring-boot:run"]
```

Etape 5 : Créons un fichier **.dockerignore** et parametrage

Nano .dockerignore

```
  GNU nano 4.8                          .dockerignore
target
```

Etape 6 : Construisons notre image Docker

Avec la commande qui suit :

```
docker build --tag java-docker .
```

```
root@karlito-VirtualBox:/path/to/working/directory/spring-petclinic# docker bui
ld --tag java-docker .
[+] Building 44.0s (6/12)
 => [internal] load build definition from Dockerfile                       0.2s
 => => transferring dockerfile: 236B                                       0.0s
 => [internal] load .dockerignore                                          0.2s
 => => transferring context: 47B                                           0.0s
 => resolve image config for docker.io/docker/dockerfile:1                 3.3s
 => docker-image://docker.io/docker/dockerfile:1@sha256:d2d74ff22a0e47b  25.9s
 => => resolve docker.io/docker/dockerfile:1@sha256:d2d74ff22a0e47b21f4b    0.0s
 => => sha256:d2d74ff22a0e47b21f4bbde337e2ac4cd0a02a2226 7.65kB / 7.65kB    0.0s
```

Etape 7 : Créons nos volumes pour les données et pour la configuration

Avec les commandes qui suivent :

```
docker volume create mysql_data

docker volume create mysql_config
```

```
root@karlito-VirtualBox:/path/to/working/directory/spring-petclinic# docker vol
ume create mysql_data
mysql_data
root@karlito-VirtualBox:/path/to/working/directory/spring-petclinic# docker vol
ume create mysql_config
mysql_config
```

Etape 8 : Créons un réseau que notre application et notre base de données utiliseront

Avec la commande qui suit :

```
docker network create mysqlnet
```

```
root@karlito-VirtualBox:/home/karlito# docker network create mysqlnet
c25ea6a892f63122161d9e41628d7d3ae6d19a4cb142d514fad31a849dcca793
```

Etape 9 : Exécutons MYSQL dans un conteneur

Avec la suite des commandes qui suit :

```
docker run -it --rm -d -v mysql_data:/var/lib/mysql \

-v mysql_config:/etc/mysql/conf.d \

--network mysqlnet \
```

```
--name mysqlserver \
-e MYSQL_USER=petclinic -e MYSQL_PASSWORD=petclinic \
-e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=petclinic \
-p 3306:3306 mysql:8.0
```

```
root@karlito-VirtualBox:/path/to/working/directory/spring-petclinic# docker run
 -it --rm -d -v mysql_data:/var/lib/mysql \
> -v mysql_config:/etc/mysql/conf.d \
> --network mysqlnet \
> --name mysqlserver \
> -e MYSQL_USER=petclinic -e MYSQL_PASSWORD=petclinic \
> -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=petclinic \
> -p 3306:3306 mysql:8.0
Unable to find image 'mysql:8.0' locally
8.0: Pulling from library/mysql
197c1adcd755: Pull complete
45f2e353f7d2: Pull complete
68ec6ece42ef: Pull complete
cfa4d9a7b88e: Pull complete
64cab5858b1d: Pull complete
92fcd248d982: Pull complete
88635e83312d: Pull complete
43f0427259d9: Pull complete
79828698a290: Pull complete
a8854781893e: Pull complete
6c8bdf3091d9: Pull complete
Digest: sha256:8653a170e0b0df19ea95055267def2615fc53c62df529e3750817c1a886485f0
Status: Downloaded newer image for mysql:8.0
eb82e0acdf35889b5a01dd99d086792a3b6a17e31a2d4b058175ac442519f8c8
```

Etape 10 : Mettons notre Dockerfile a jour pour produire une image finale

```
  GNU nano 4.8                          Dockerfile
# syntax=docker/dockerfile:1

FROM eclipse-temurin:17-jdk-jammy

WORKDIR /app

COPY .mvn/ .mvn
COPY mvnw pom.xml ./
RUN ./mvnw dependency:resolve

COPY src ./src

FROM base as development
CMD ["./mvnw", "spring-boot:run", "-Dspring-boot.run.profiles=mysql", "-Dsprin>

FROM base as build
RUN ./mvnw package

FROM eclipse-temurin:17-jre-jammy as production
EXPOSE 8080
COPY --from=build /app/target/spring-petclinic-*.jar /spring-petclinic.jar
CMD ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/spring-petcl>
```

Etape 11 : Utilisons Docker Compose pour développer localement et paramétrons le

Avec la commande qui suit :

```
nano docker-compose.dev.yml
```

Voici les paramètres :

```
version: '3.8'
services:
  petclinic:
    build:
      context: .
      target: development
    ports:
      - "8000:8000"
      - "8080:8080"
    environment:
      - SERVER_PORT=8080
      - MYSQL_URL=jdbc:mysql://mysqlserver/petclinic
    volumes:
      - ./:/app
    depends_on:
      - mysqlserver

  mysqlserver:
    image: mysql:8.0
    ports:
      - "3306:3306"
    environment:
      - MYSQL_ROOT_PASSWORD=
      - MYSQL_ALLOW_EMPTY_PASSWORD=true
      - MYSQL_USER=petclinic
      - MYSQL_PASSWORD=petclinic
      - MYSQL_DATABASE=petclinic
    volumes:
      - mysql_data:/var/lib/mysql
      - mysql_config:/etc/mysql/conf.d
```

```
volumes:
  mysql_data:
  mysql_config:
```

```
 GNU nano 4.8                    docker-compose.dev.yml
version: '3.8'
services:
  petclinic:
    build:
      context: .
      target: development
    ports:
      - "8000:8000"
      - "8080:8080"
    environment:
      - SERVER_PORT=8080
      - MYSQL_URL=jdbc:mysql://mysqlserver/petclinic
    volumes:
      - ./:/app
    depends_on:
      - mysqlserver

  mysqlserver:
    image: mysql:8.0
    ports:
      - "3306:3306"
    environment:
      - MYSQL_ROOT_PASSWORD=
      - MYSQL_ALLOW_EMPTY_PASSWORD=true
      - MYSQL_USER=petclinic
```

Etape 12 : Démarrons notre application et confirmons qu'elle fonctionne

Avec la commande qui suit :

```
docker-compose -f docker-compose.dev.yml up --build
```

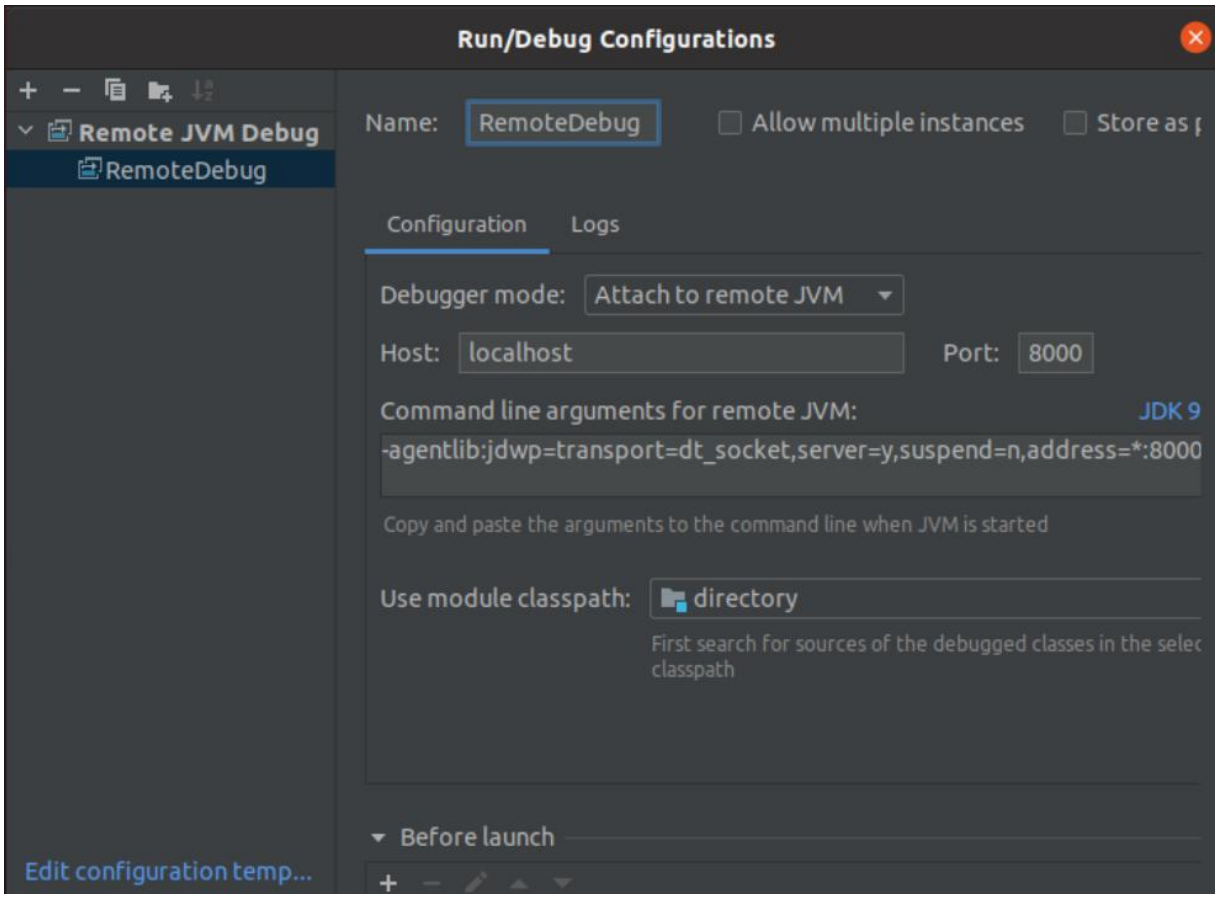
```
root@karlito-VirtualBox:/path/to/working/directory/spring-petclinic-docker# doc
ker-compose -f docker-compose.dev.yml up --build
Removing spring-petclinic-docker_mysqlserver_1
Building petclinic
Step 1/8 : FROM eclipse-temurin:17-jdk-jammy as base
 ---> 5eecde9cb979
Step 2/8 : WORKDIR /app
 ---> Using cache
 ---> 5cbee6370375
Step 3/8 : COPY .mvn/ .mvn
 ---> Using cache
 ---> 86f73ab40e67
Step 4/8 : COPY mvnw pom.xml ./
 ---> Using cache
 ---> 22cbeb9c0fad
```

Etape 13 : Testons notre point de terminaison **API**

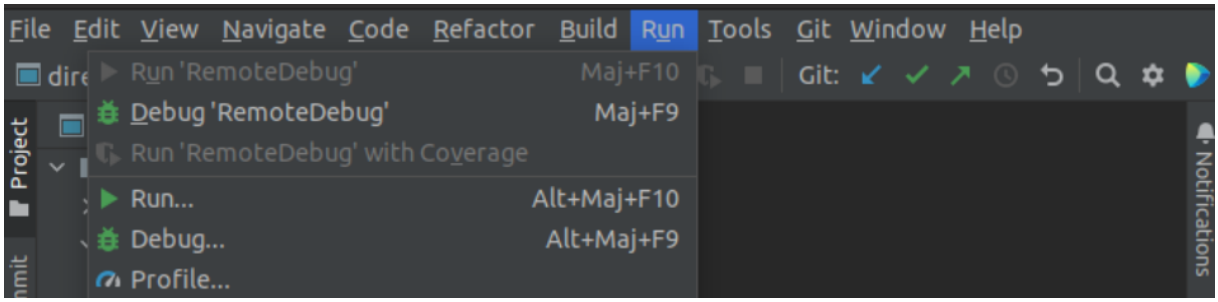
Avec la commande qui suit :

```
curl  --request GET \
--url http://localhost:8080/vets \
--header 'content-type: application/json'
```

Etape 14 : Connectons un débogueur et fixons un point d'arrêt



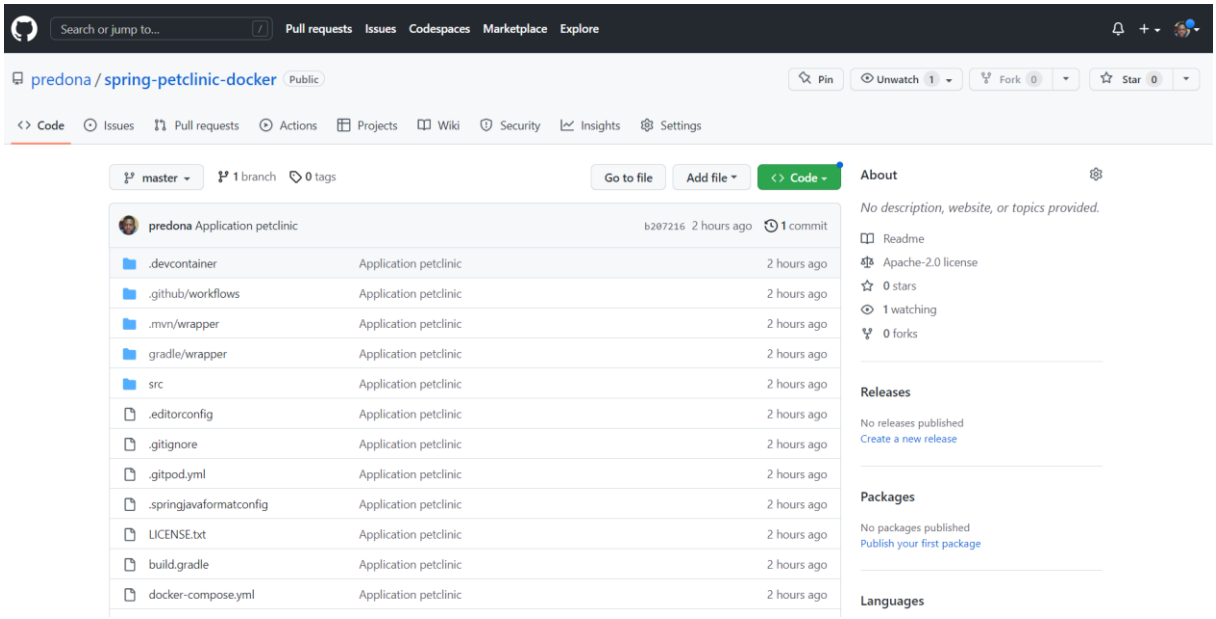
Ensuite :



Etape 15 : Appelons le point de terminaison du serveur

```
root@karlito-VirtualBox:/path/to/working/directory/spring-petclinic-docker# cur
l --request GET --url http://localhost:8080/vets
{"vetList":[{"id":1,"firstName":"James","lastName":"Carter","specialties":[],"n
rOfSpecialties":0,"new":false},{"id":2,"firstName":"Helen","lastName":"Leary","
specialties":[{"id":1,"name":"radiology","new":false}],"nrOfSpecialties":1,"new
":false},{"id":3,"firstName":"Linda","lastName":"Douglas","specialties":[{"id":
3,"name":"dentistry","new":false},{"id":2,"name":"surgery","new":false}],"nrOfS
pecialties":2,"new":false},{"id":4,"firstName":"Rafael","lastName":"Ortega","sp
ecialties":[{"id":2,"name":"surgery","new":false}],"nrOfSpecialties":1,"new":fa
lse},{"id":5,"firstName":"Henry","lastName":"Stevens","specialties":[{"id":1,"n
ame":"radiology","new":false}],"nrOfSpecialties":1,"new":false},{"id":6,"firstN
ame":"Sharon","lastName":"Jenkins","specialties":[],"nrOfSpecialties":0,"new":f
alse}]}root@karlito-VirtualBox:/path/to/working/directory/spring-petclinic-dock
```

III. Déploiement de l'application sur GitHub



Voici le lien :

https://github.com/predona/spring-petclinic-docker

IV. Conclusion

Nous avons appris avec succès comment conteneuriser une application **PetClinic** à l'aide de Docker. Avec une construction en plusieurs étapes, nous pouvons facilement réduire la taille de votre image Docker finale et améliorer les performances d'exécution. À l'aide d'un seul fichier **YAML**, nous avons démontré comment Docker Compose nous aide à créer et à déployer facilement notre application **PetClinic** en quelques secondes.