***Virtualisation et Cloud***

**Application Java PetClinic construire avec Spring Boot**

**Membres du groupe**

**Superviseur**

* **Dr.KébaGueye**
* ***Mazaba Predona Doden***
* ***Odzongo Alexis***

**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**

1. **Introduction**
2. **Création de l’application**
3. **Déploiement de l’application sur GitHub**
4. **Conclusion**
5. **Introduction**

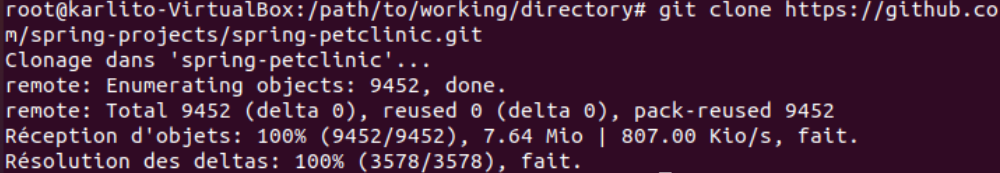
**Spring PetClinic** est un exemple d'application open source développé pour démontrer les capacités orientées base de données de Spring Boot, Spring MVC et Spring Data Framework. Il est basé sur cette pile Spring et construit avec Maven.

1. **Création de l’application**

**Etape 1** : Clonons notre application Pet Clinic.

Avec la commande qui suit :

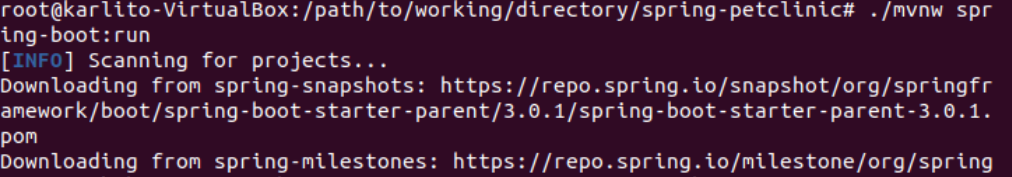
***git clone https://github.com/spring-projects/spring-petclinic.git***

****

**Etape 2** : Téléchargeons les dépendances liées a notre application

Avec la commande qui suit :

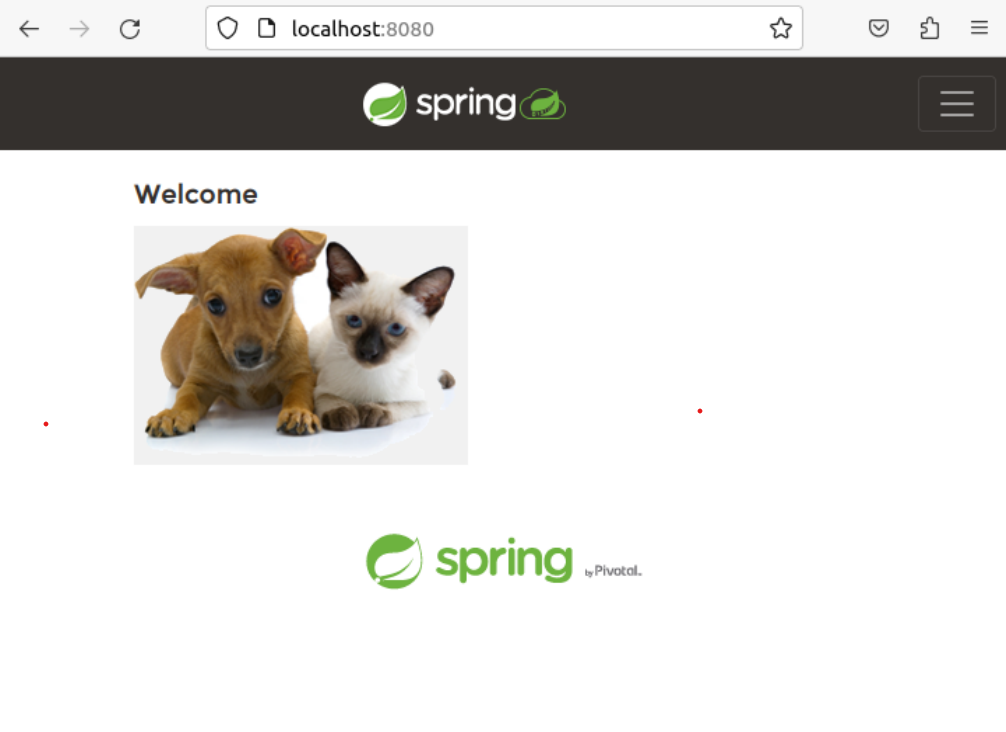
***./mvnw spring-boot:run***

******

**Etape 3** : Testons notre application

Ouvrons notre navigateur et tapons :

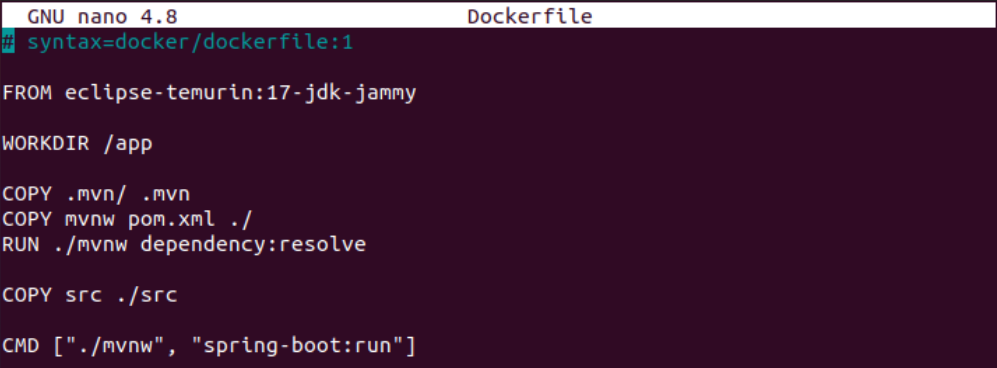
[***http://localhost:8080***](http://localhost:8080)

******

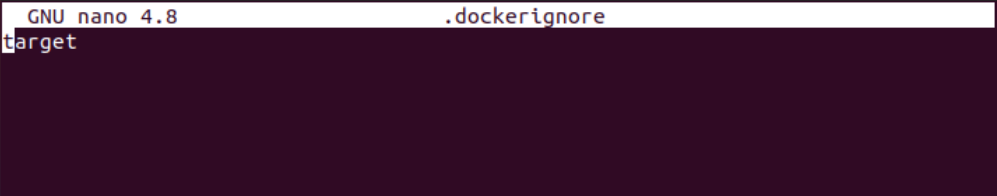
**Etape 4** : Construisons un **Dockerfile** et définissons notre image de base

Avece la commande qui suit :

***Nano Dockerfile***



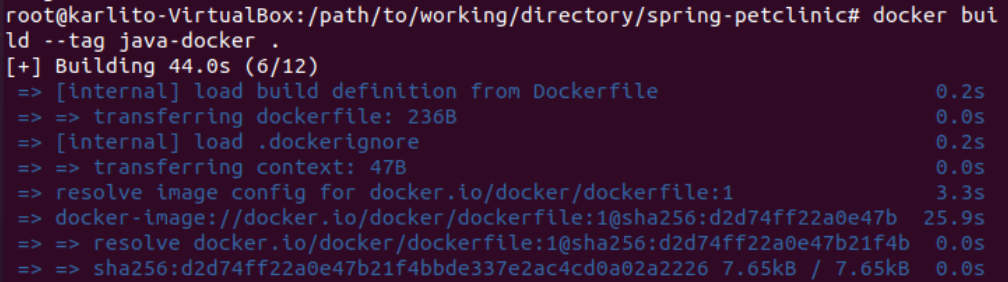
**Etape 5** : Créons un fichier **.dockerignore** et parametrage

***Nano .dockerignore***

**Etape 6** : Construisons notre image Docker

Avec la commande qui suit :

***docker build --tag java-docker .***

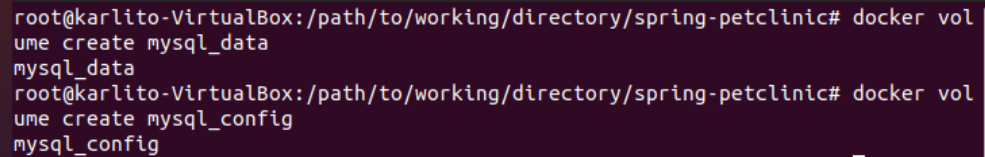


**Etape 7** : Créons nos volumes pour les données et pour la configuration

Avec les commandes qui suivent :

***docker volume create mysql\_data***

***docker volume create mysql\_config***

******

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

Avec la commande qui suit :

***docker network create mysqlnet***

******

**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 \***

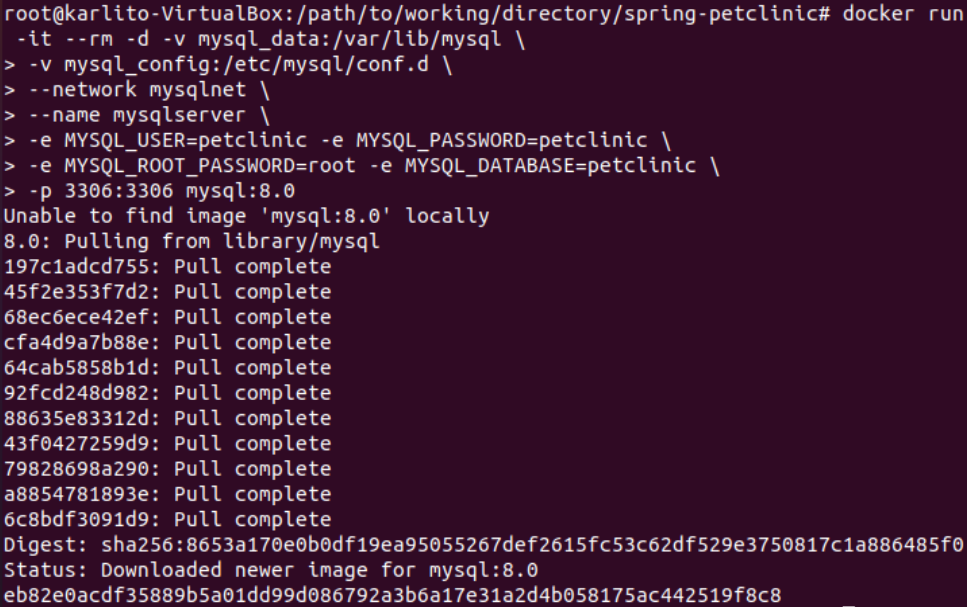
***--network mysqlnet \***

***--name mysqlserver \***

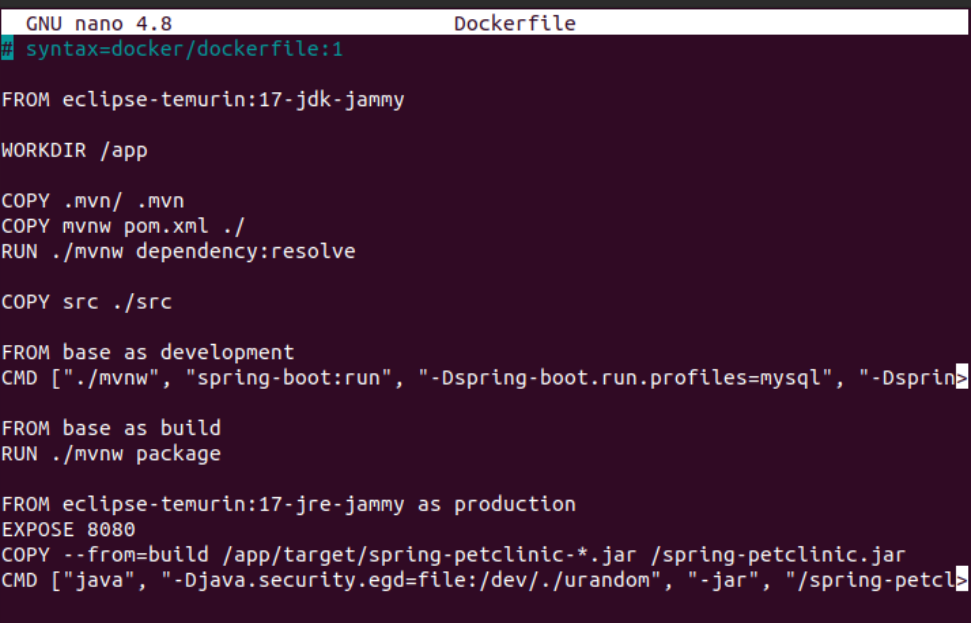
***-e MYSQL\_USER=petclinic -e MYSQL\_PASSWORD=petclinic \***

***-e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=petclinic \***

***-p 3306:3306 mysql:8.0***



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



**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:***

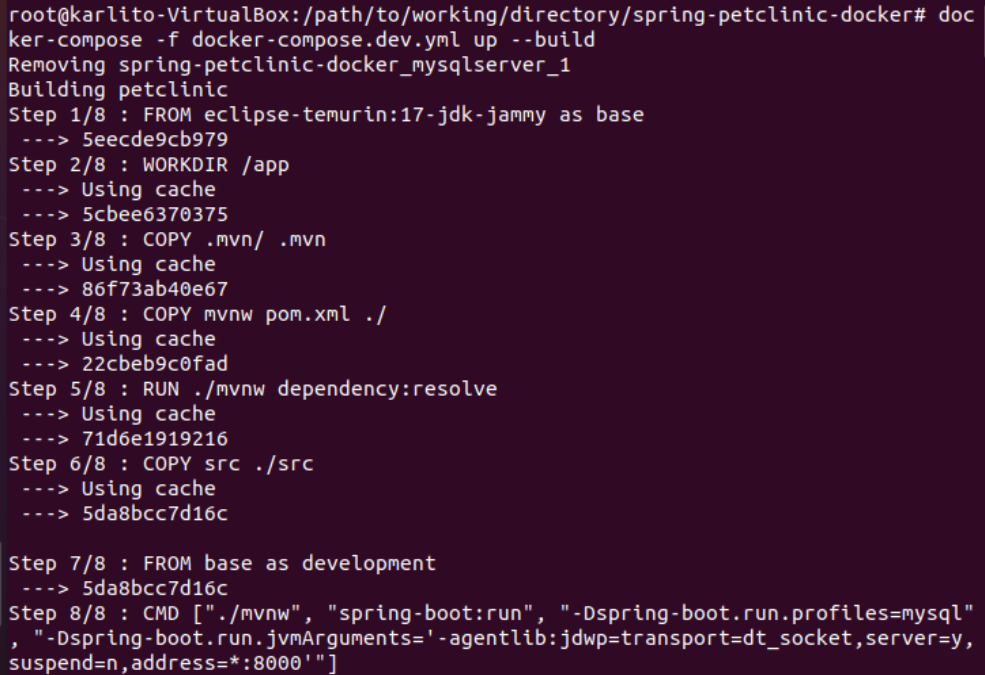
***mysql\_config:***

******

**Etape 12** : Démarrons notre application et confirmons qu’elle fonctionne

Avec la commande qui suit :

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



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

Avec la commande qui suit :

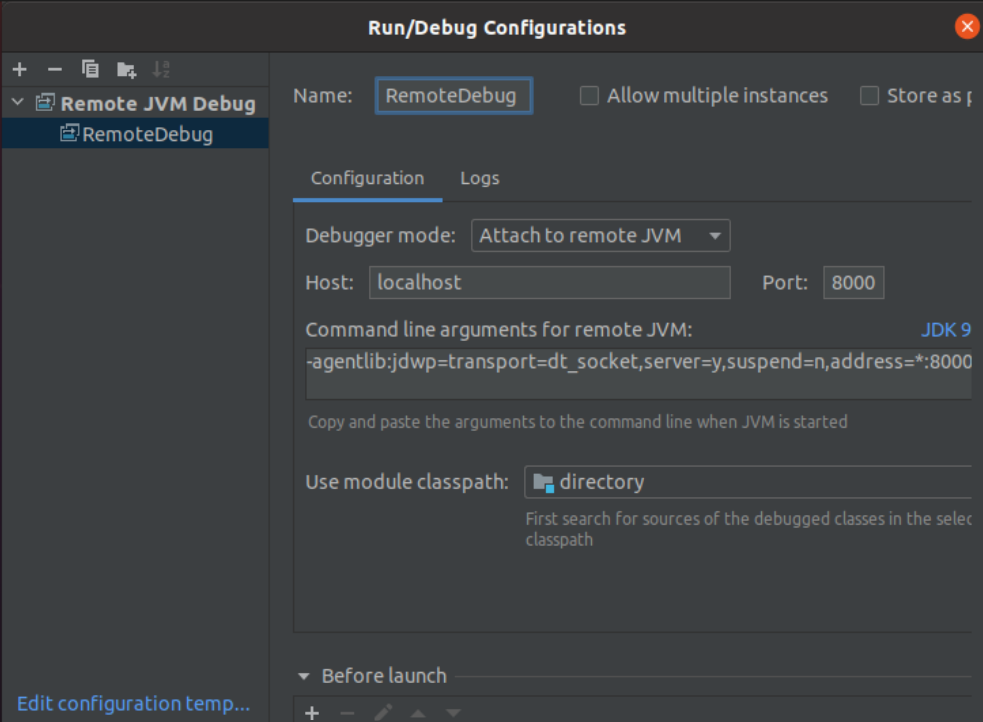
***curl --request GET \***

***--url http://localhost:8080/vets \***

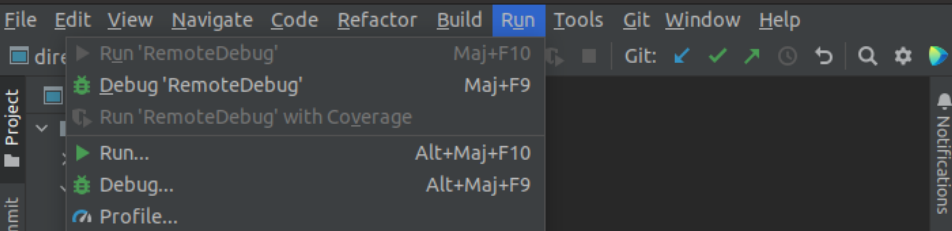
***--header 'content-type: application/json'***

****

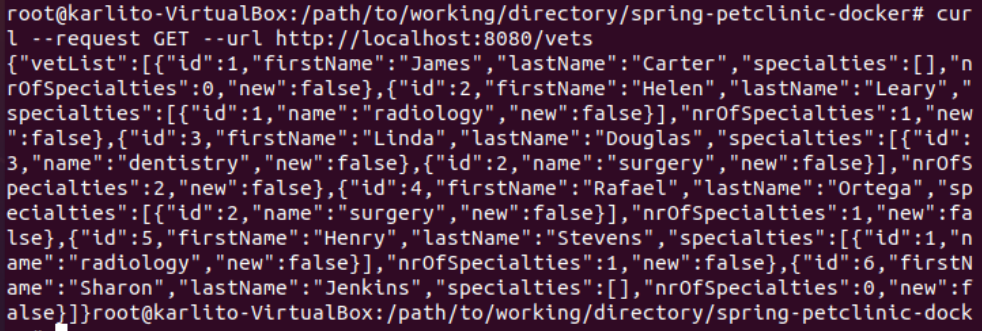
**Etape 14** : Connectons un débogueur et fixons un point d’arrêt

****

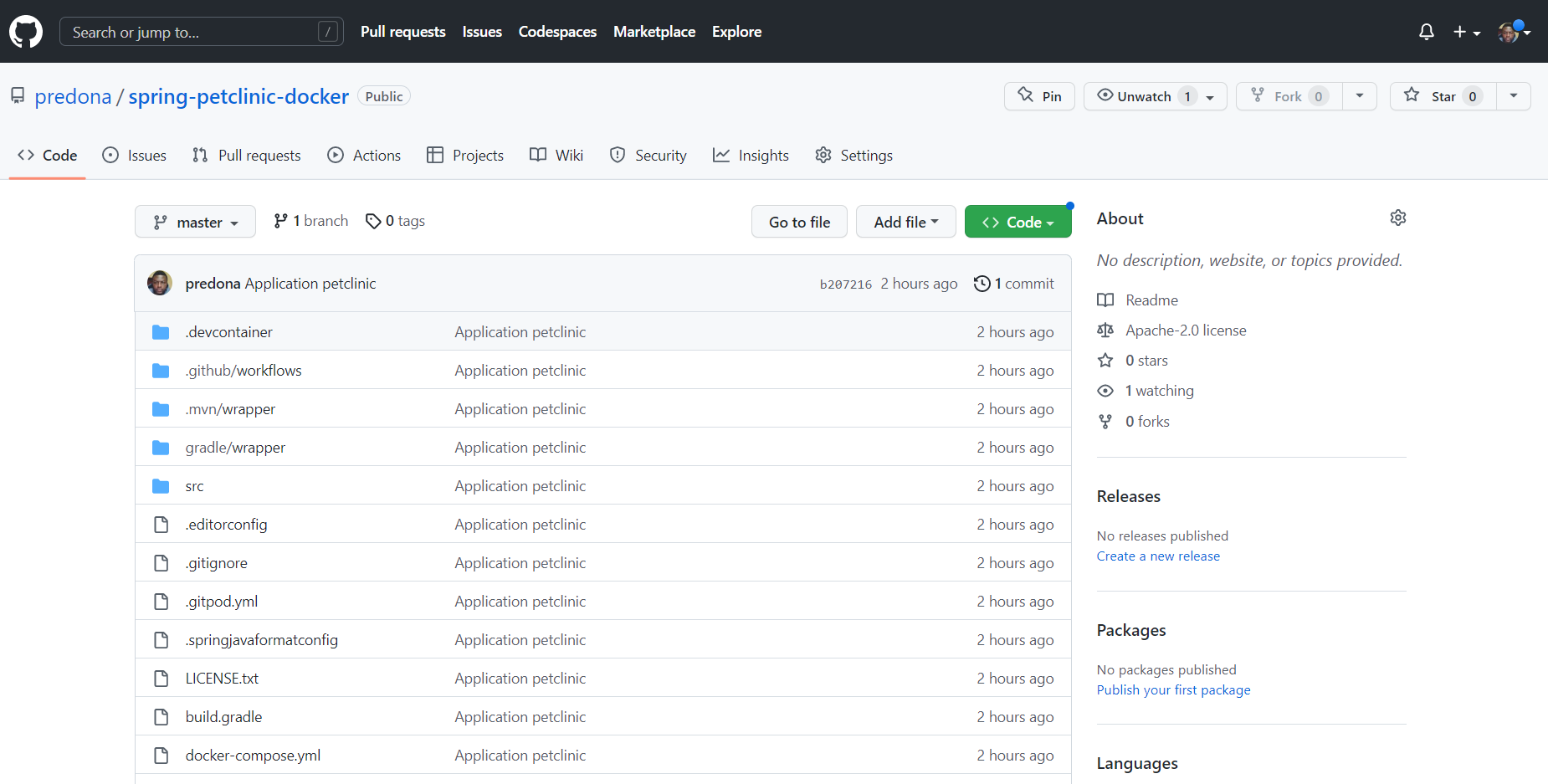
Ensuite :



**Etape 15** : Appelons le point de terminaison du serveur



1. **Déploiement de l’application sur GitHub**



1. **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.