

# Développement microservice

# Table of Contents

Guide pour créer une application SpringBoot dans un container Linux Docker .....	1
Installation recommandé .....	1
Application Spring Boot .....	1
Containeriser l'application SpringBoot. ....	2
Dockerfile .....	2
Construire une image Docker avec Maven .....	2
La configuration de Maven doit contenir informations .....	3
Utilisé Maven pour construire l'image Docker .....	3
Construire une image Docker avec Gradle .....	3
Utiliser Gradle pour construire une image Docker .....	4
Demonstration .....	4

# Guide pour créer une application SpringBoot dans un container Linux Docker

Ce document énumère les étapes de création d'une application Springboot dans un container Docker, un container Linux.

## Installation recommandé

- docker (1.6.0 ou supérieur)
- jdk 1.8
- Gradle 2.3+ or Maven 3.0+
- Install the required software

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
$ java -version
java version "1.8.0_45"
Java(TM) SE Runtime Environment (build 1.8.0_45-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)
$ sudo apt-get install gradle
$ sudo apt-get install maven
```

## Application Spring Boot

Ce fichier java retourne "Hello world" sur localhost:8080/

```
`src/main/java/hello/Application.java`
```

```

package hello;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.bind.RelaxedPropertyResolver;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@SpringBootApplication
@RestController
public class Application {
    @RequestMapping("/")
    public String home() {
        return "Hello Docker World";
    }
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}

```

Cette classe contient l'annotation `@SpringBootApplication` et `@RestController` pour paramétrer Springboot en Spring MVC. Quand l'utilisateur appel `localhost:8080/` Springboot renvoi sur la page "Hello Docker World"

## Containeriser l'application SpringBoot

Pour contenariser une application SpringBoot un Dockerfile est necessaire à la racine du projet. Lors de la construction de l'application, l'artefact "app.jar"

We will containerize the above application using the DockerFile. The project JAR file is added to the container as `app.jar` and then executed in the ENTRYPOINT. We have added a VOLUME pointing to /tmp because it is where a Spring Boot application creates working directories for Tomcat by default.

### Dockerfile

This file is needed to create a docker image;

```

FROM java:8
VOLUME /tmp
ADD gs-spring-boot-docker-0.1.0.jar app.jar
RUN bash -c 'touch /app.jar'
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]

```

## Construire une image Docker avec Maven

Maven support la construction d'une application Springboot dans un container Docker. Il suffit de préciser un plugins dans `pom.xml` (rajouter la dépendance Maven / Docker).

Dans la configuration, il est nécessaire de : . préciser le nom et le tag de l'application . le dossier où est situé le Dockerfile . les fichiers ressources pour construire l'image.

## La configuration de Maven doit contenir informations

`pom.xml`

```
<properties>
    <docker.image.prefix>springio</docker.image.prefix>
</properties>
<build>
    <plugins>
        <plugin>
            <groupId>com.spotify</groupId>
            <artifactId>docker-maven-plugin</artifactId>
            <version>0.2.3</version>
            <configuration>
                <imageName>${docker.image.prefix}/${project.artifactId}</imageName>
                <dockerDirectory>src/main/docker</dockerDirectory>
                <resources>
                    <resource>
                        <targetPath>/</targetPath>
                        <directory>${project.build.directory}</directory>
                        <include>${project.build.finalName}.jar</include>
                    </resource>
                </resources>
            </configuration>
        </plugin>
    </plugins>
</build>
```

## Utilisé Maven pour construire l'image Docker

```
$ mvn package docker:build
```

## Construire une image Docker avec Gradle

Pour la configuration de la construction de Docker image avec Gradle, il faut rajouter dans la configuration de Gradle

```

build.gradle
buildscript {
    ...
    dependencies {
        ...
        classpath('se.transmode.gradle:gradle-docker:1.2')
    }
}
group = 'springio'
...
apply plugin: 'docker'

```

```

task buildDocker(type: Docker, dependsOn: build) {
    push = true
    applicationName = jar.baseName
    dockerfile = file('src/main/docker/Dockerfile')
    doFirst {
        copy {
            from jar
            into stageDir
        }
    }
}

```

## Utiliser Gradle pour construire une image Docker

```
$ ./gradlew build buildDocker
```

## Demonstration

Maintenant l'application doit tourner sur **localhost:8080/** (application SpringBoot par default).

```

$ docker images

```

REPOSITORY	TAG	IMAGE ID	CREATED
gregturn/gs-spring-boot-docker	latest	3e70f57df702	21 hours ago

```

VIRTUAL SIZE
841.4 MB

```

```
$ docker run -p 8080:8080 -t gregturn/gs-spring-boot-docker
```