

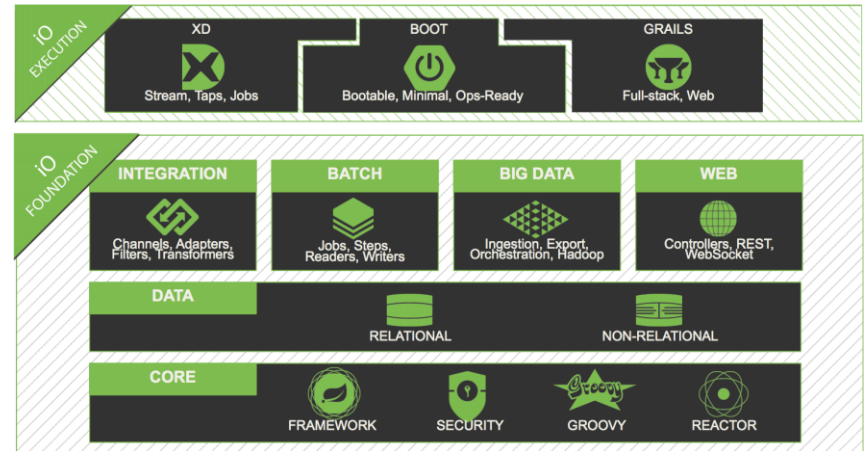


# LẬP TRÌNH JAVA SPRING BOOT

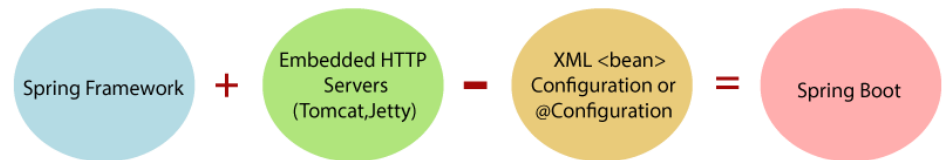
## BÀI 1: LÀM QUEN VỚI SPRING BOOT

- ❑ Kết thúc bài học này bạn có khả năng
    - ❖ Hiểu cấu trúc lập trình Java Spring Boot
    - ❖ Sử dụng công cụ IntelliJ IDEA
    - ❖ Biết cách cài đặt và sử dụng Docker
    - ❖ Hiểu mô hình giao tiếp 2 phía Client và Server
-

- ❑ Spring Boot là một dự án phát triển bởi JAVA (ngôn ngữ java) trong hệ sinh thái Spring framework.



- ❑ Nó giúp cho các lập trình viên chúng ta **đơn giản hóa** quá trình lập trình một ứng dụng với **Spring**, chỉ tập trung vào việc phát triển **ng nghiệp vụ** cho ứng dụng.



- ❑ Truy cập vào <https://www.jetbrains.com/idea/download/#section=windows> để tải về phiên bản phù hợp với máy và tiến hành cài đặt

IntelliJ IDEA

Coming in 2022.2 What's New Features Resources

Pricing

Download



Version: 2022.1.3  
Build: 221.5921.22  
21 June 2022

[Release notes](#) ↗

## Download IntelliJ IDEA

Windows macOS Linux

### Ultimate

For web and enterprise development

Download

.exe ▼

Free 30-day trial available

### Community

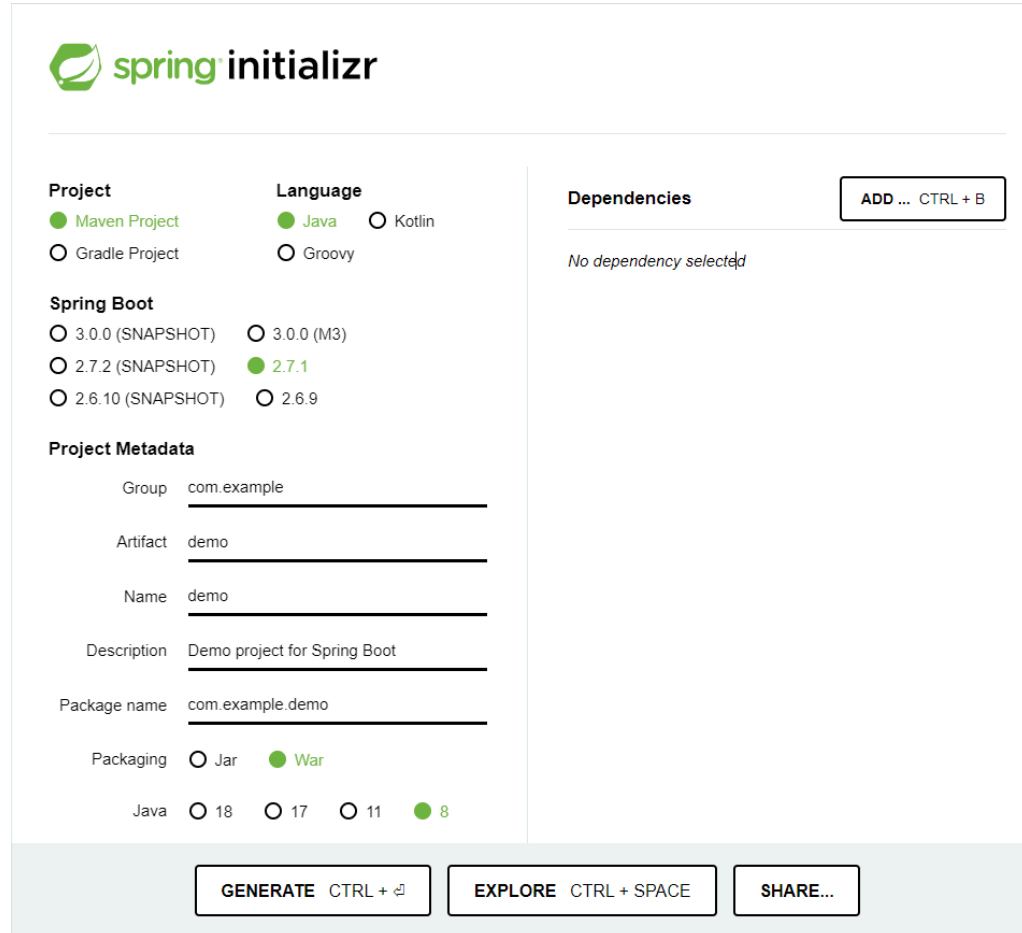
For JVM and Android development

Download

.exe ▼

Free, built on open source

- ❑ Tạo project spring boot đơn giản với Spring Initializr
- ❑ Truy cập vào <https://start.spring.io/> để tạo project

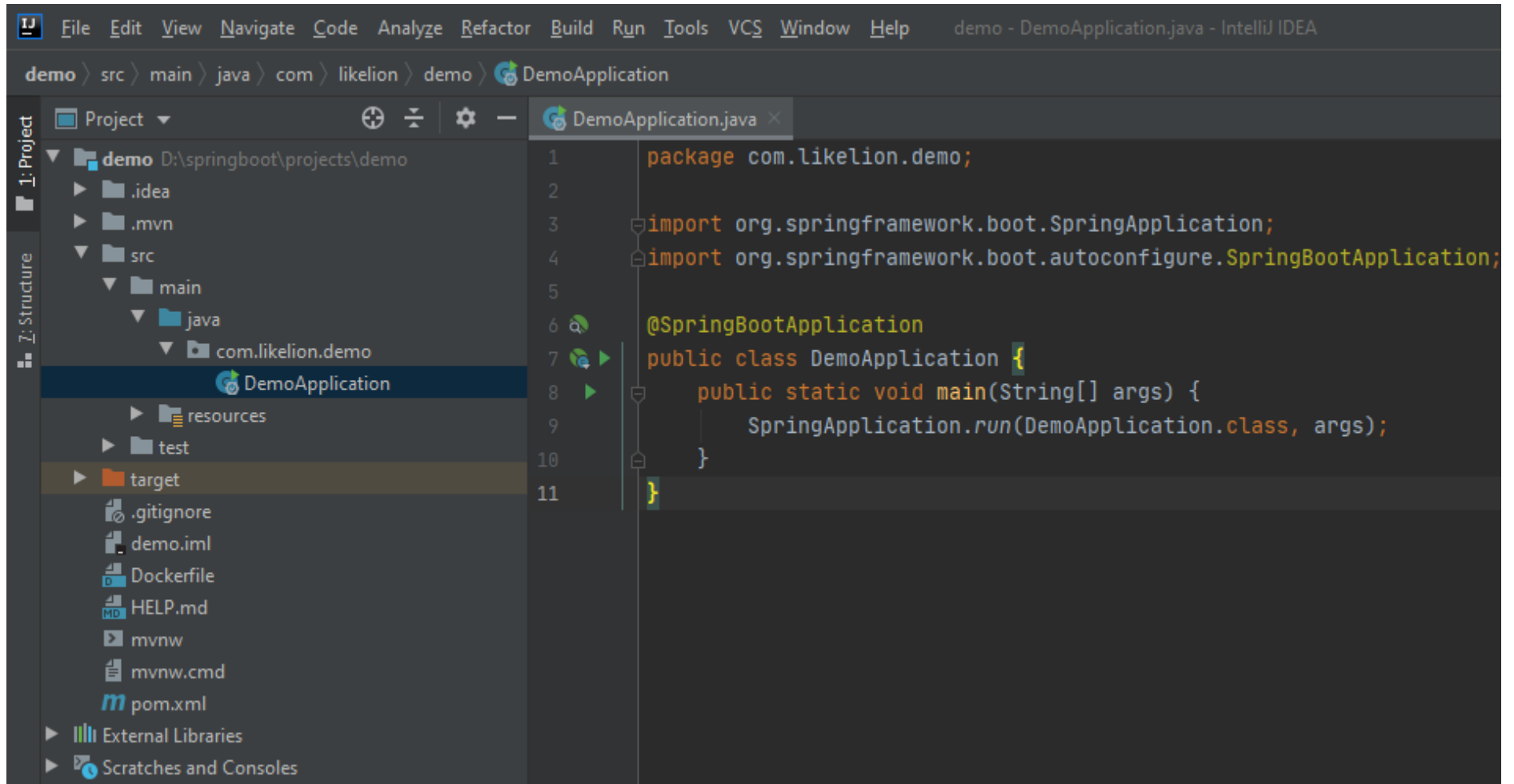


The screenshot displays the Spring Initializr web interface. At the top left is the Spring logo and the text "spring initializr". The form is divided into several sections:

- Project:** Includes radio buttons for "Maven Project" (selected) and "Gradle Project".
- Language:** Includes radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot:** Includes radio buttons for versions: "3.0.0 (SNAPSHOT)", "3.0.0 (M3)", "2.7.2 (SNAPSHOT)", "2.7.1" (selected), "2.6.10 (SNAPSHOT)", and "2.6.9".
- Project Metadata:** A series of text input fields for:
  - Group: com.example
  - Artifact: demo
  - Name: demo
  - Description: Demo project for Spring Boot
  - Package name: com.example.demo
- Packaging:** Includes radio buttons for "Jar" and "War" (selected).
- Java:** Includes radio buttons for versions: "18", "17", "11", and "8" (selected).
- Dependencies:** A section with the text "No dependency selected" and an "ADD ... CTRL + B" button.

At the bottom of the form are three buttons: "GENERATE CTRL + G", "EXPLORE CTRL + SPACE", and "SHARE...".

## ❏ Cấu trúc mặc định của project



- ❑ Nội dung file ***pom.xml***:
- ❑ `<parent>` là một project sẵn có trong Spring Boot. Các thư viện phụ thuộc cơ bản đã được khai báo trong **parent**, project của bạn chỉ cần thừa kế nó.
- ❑ Các "Starter" khác chỉ đơn giản là cung cấp phụ thuộc mà bạn có khả năng cần thiết khi phát triển một loại hình cụ thể của ứng dụng. Chẳng hạn khi bạn phát triển một ứng dụng web, bạn cần một phụ thuộc **spring-boot-starter-web**.
- ❑ plugin là plugin cung cấp các thư viện cần thiết giúp project của bạn có thể chạy trực tiếp mà không cần triển khai trên một Web Server. Nó giúp tạo ra một file jar có thể thực thi.

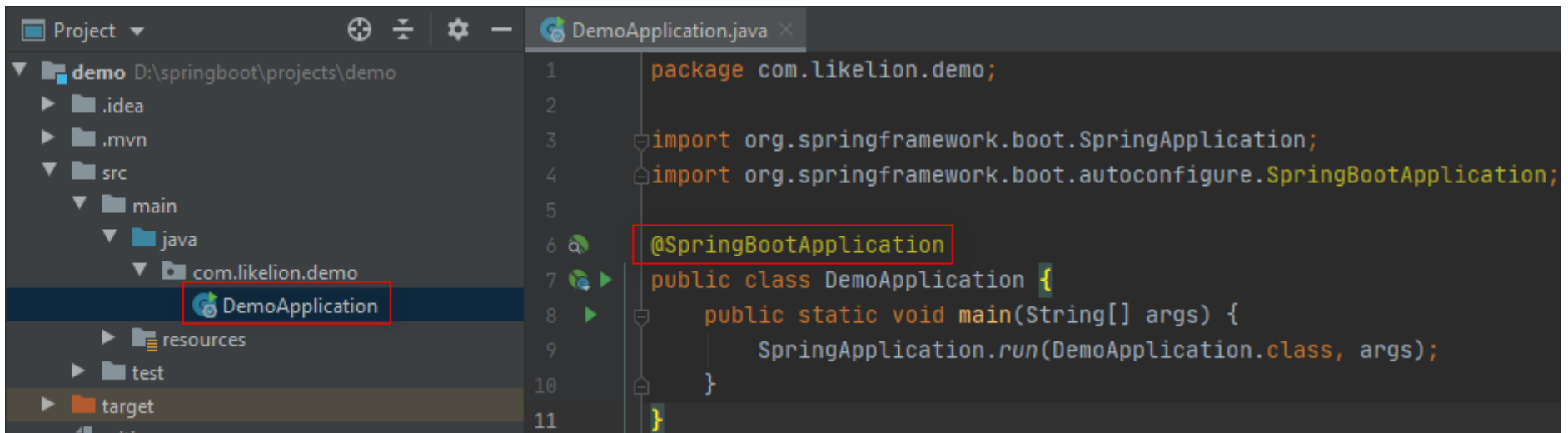
```
<modelVersion>4.0.0</modelVersion>
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.7.1</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.likelion</groupId>
<artifactId>demo</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>demo</name>
<description>Demo project for Spring Boot</description>
<properties>
  <java.version>1.8</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

1

2

3

- ❑ Khi bạn tạo một "Spring Boot Web App", có 1 class được tạo ra (generate). Và class này được chú thích bởi **@SpringBootApplication**
- ❑ Ví dụ **DemoApplication**

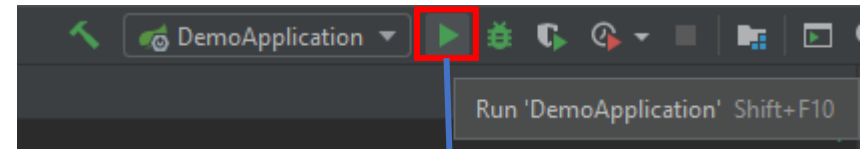
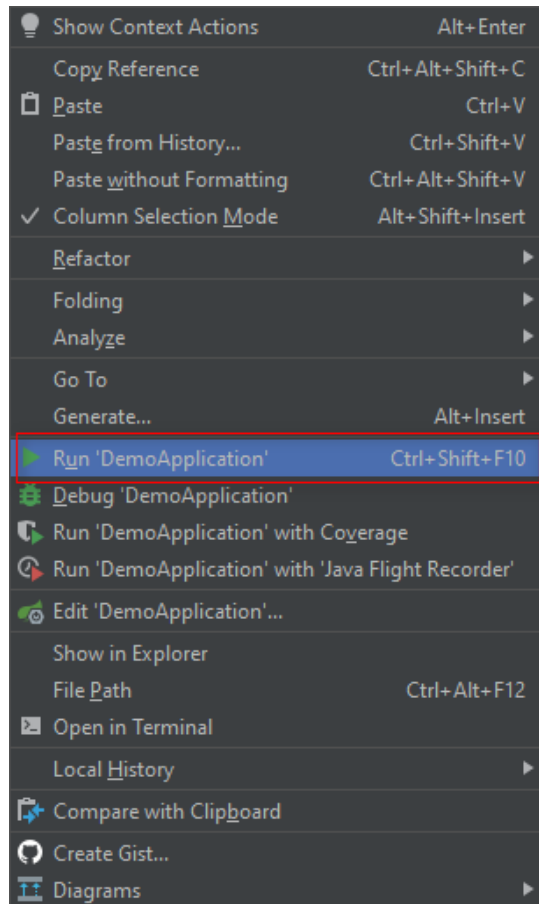


```
1 package com.likelion.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class DemoApplication {
8     public static void main(String[] args) {
9         SpringApplication.run(DemoApplication.class, args);
10    }
11 }
```

- ❑ Ứng dụng của bạn được bắt đầu bởi việc thực thi class được chú thích bởi **@SpringBootApplication**.



- ❑ Click chuột phải vào file có chú thích **@SpringBootApplication**
- ❑ **Run 'DemoApplication'**



Hoặc click vào nút **run** trên thanh **toolbar**

❑ Project chạy với cổng **8080** mặc định như hình dưới

```

  ____  _
 / ___|| | | |
| |___| |_| |
 \___|_||_|_|_|

:: Spring Boot ::                (v2.7.1)

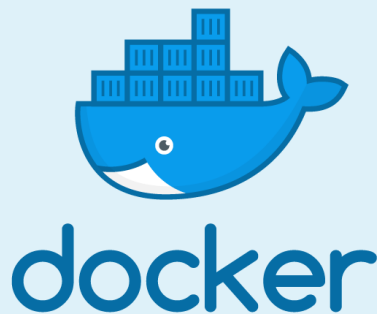
2022-07-08 09:31:30.815 INFO 15112 --- [main] com.likelion.demo.DemoApplication : Starting DemoApplication using Java 1.8.0_202 on MSI with PID 15112
2022-07-08 09:31:30.815 INFO 15112 --- [main] com.likelion.demo.DemoApplication : No active profile set, falling back to 1 default profile: "default"
2022-07-08 09:31:31.442 INFO 15112 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-07-08 09:31:31.442 INFO 15112 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-07-08 09:31:31.442 INFO 15112 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.64]
2022-07-08 09:31:31.521 INFO 15112 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-07-08 09:31:31.521 INFO 15112 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 674 ms
2022-07-08 09:31:31.745 INFO 15112 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-07-08 09:31:31.745 INFO 15112 --- [main] com.likelion.demo.DemoApplication : Started DemoApplication in 1.192 seconds (JVM running for 1.681)
2022-07-08 09:31:37.229 INFO 15112 --- [nio-8080-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-07-08 09:31:37.229 INFO 15112 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-07-08 09:31:37.229 INFO 15112 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms
```

❑ Mở trình duyệt và truy cập vào <http://localhost:8080/>



Chạy project thành công

- ❑ Theo nguồn wikipedia: "Docker là một dự án mã nguồn mở giúp **tự động** triển khai các ứng dụng **Linux** và **Windows** vào trong các container **ảo hóa**"
- ❑ Theo các trang mạng: "Docker là một open platform cung cấp công cụ và service để người sử dụng có thể **đóng gói** và **chạy chương trình** trên các môi trường khác nhau một cách nhanh nhất."



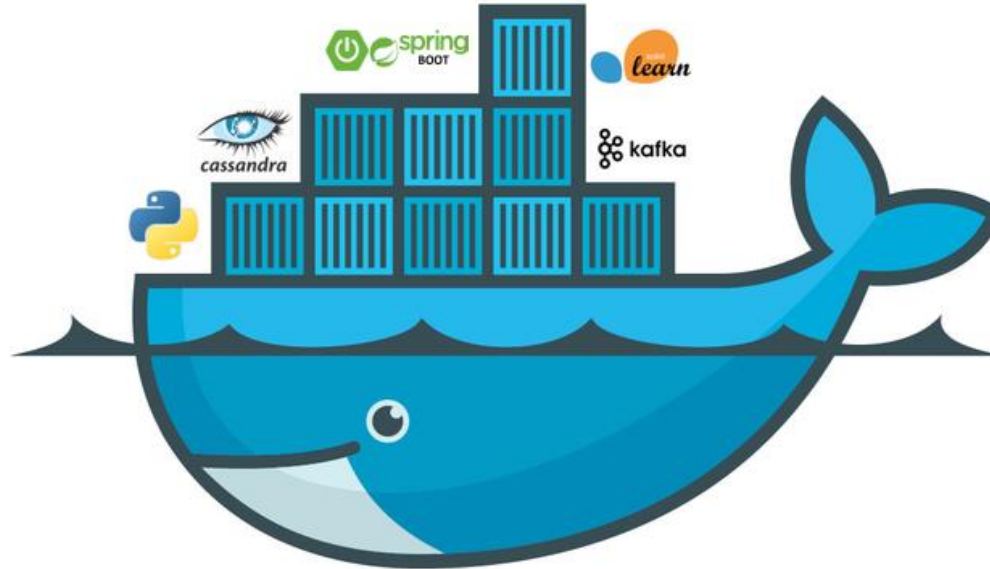
Tăng tốc cách bạn xây dựng, chia sẻ và chạy các ứng dụng hiện đại.

**13 triệu +**  
nhà phát triển

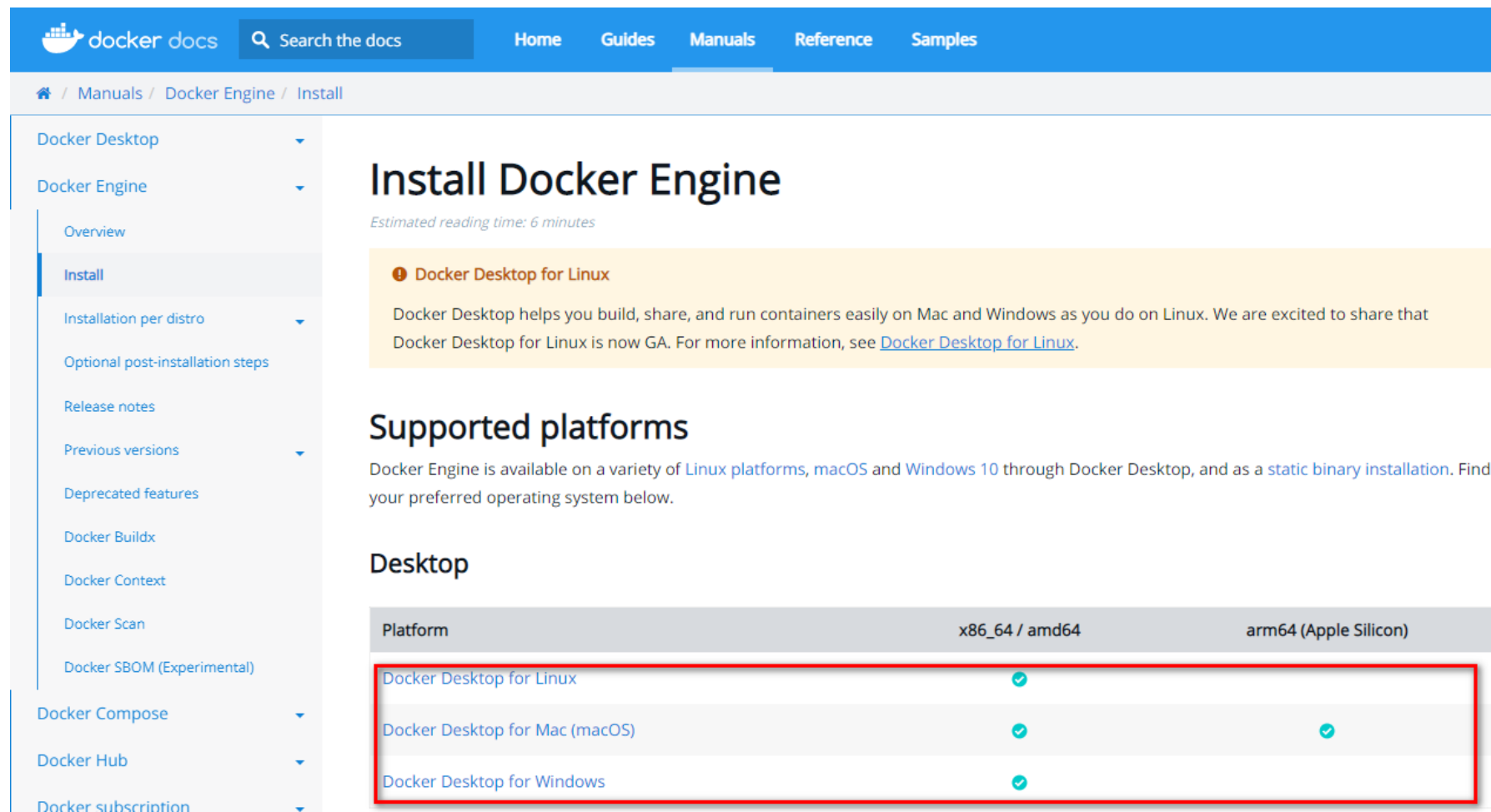
**7 triệu +**  
đơn đăng ký

**13 tỷ +**  
lượt tải xuống hình ảnh hàng tháng

- ❑ Lại có định nghĩa rằng: "Docker là một phương thức để đóng gói và sắp xếp phần mềm"
- **Vậy Docker có thể làm gì?**
- ❑ Khi ai đó muốn chạy app thì **chỉ cần** chạy docker mà **không cần** cài môi trường vì docker đã giúp ta rồi. Bạn **không cần biết** về laravel, ruby on rails hay thậm chí là java cũng như các **môi trường** của nó mà vẫn chạy được app, thật là đơn giản đúng không nào!



- ❑ Truy cập vào <https://docs.docker.com/engine/install/> để tải về bản phù hợp với máy, và bắt đầu cài đặt.



The screenshot shows the Docker Docs website. The top navigation bar includes the Docker logo, a search bar, and links for Home, Guides, Manuals, Reference, and Samples. The breadcrumb trail indicates the current location: Home / Manuals / Docker Engine / Install.

The main content area is titled "Install Docker Engine" with an estimated reading time of 6 minutes. Below the title is a yellow callout box for "Docker Desktop for Linux" announcing its GA status. The "Supported platforms" section follows, with a sub-section for "Desktop" containing a table of supported operating systems.

Platform	x86_64 / amd64	arm64 (Apple Silicon)
Docker Desktop for Linux	✓	
Docker Desktop for Mac (macOS)	✓	✓
Docker Desktop for Windows	✓	

- ❑ **Docker Engine**: dùng để tạo ra Docker image và chạy Docker container.
- ❑ **Docker Hub**: dịch vụ lưu trữ giúp chứa các Docker image.



### Một số khái niệm khác:

- ❑ **Docker Machine**: tạo ra các Docker engine trên máy chủ
  - ❑ **Docker Compose**: chạy ứng dụng bằng cách định nghĩa cấu hình các Docker container thông qua tệp cấu hình
  - ❑ **Docker Image**: một dạng tập hợp các tệp của ứng dụng, được tạo ra bởi Docker engine. Nội dung của các Docker image sẽ không bị thay đổi khi di chuyển. Docker image được dùng để chạy các Docker container
  - ❑ **Docker Container**: một dạng runtime của các Docker image, dùng để làm môi trường chạy ứng dụng
-

❑ Docker Images: Là một template chỉ **cho phép đọc**, ví dụ một image có thể chứa hệ điều hành Ubuntu và web app. Images được dùng để **tạo** Docker container. Docker cho phép chúng ta **build** và **cập nhật** các image có sẵn một cách cơ bản nhất, hoặc bạn có thể download Docker images của người khác.

❑ Docker Container: Docker container có nét giống với các **directory**. Một Docker container giữ mọi thứ chúng ta cần để chạy một app. Mỗi container **được tạo** từ Docker image. Docker container có thể có các trạng thái **run, started, stopped, moved** và **deleted**.

---

- ❑ **Dockerfile** là một tập tin dạng text chứa một tập các **câu lệnh** để tạo mới một **Image** trong Docker.

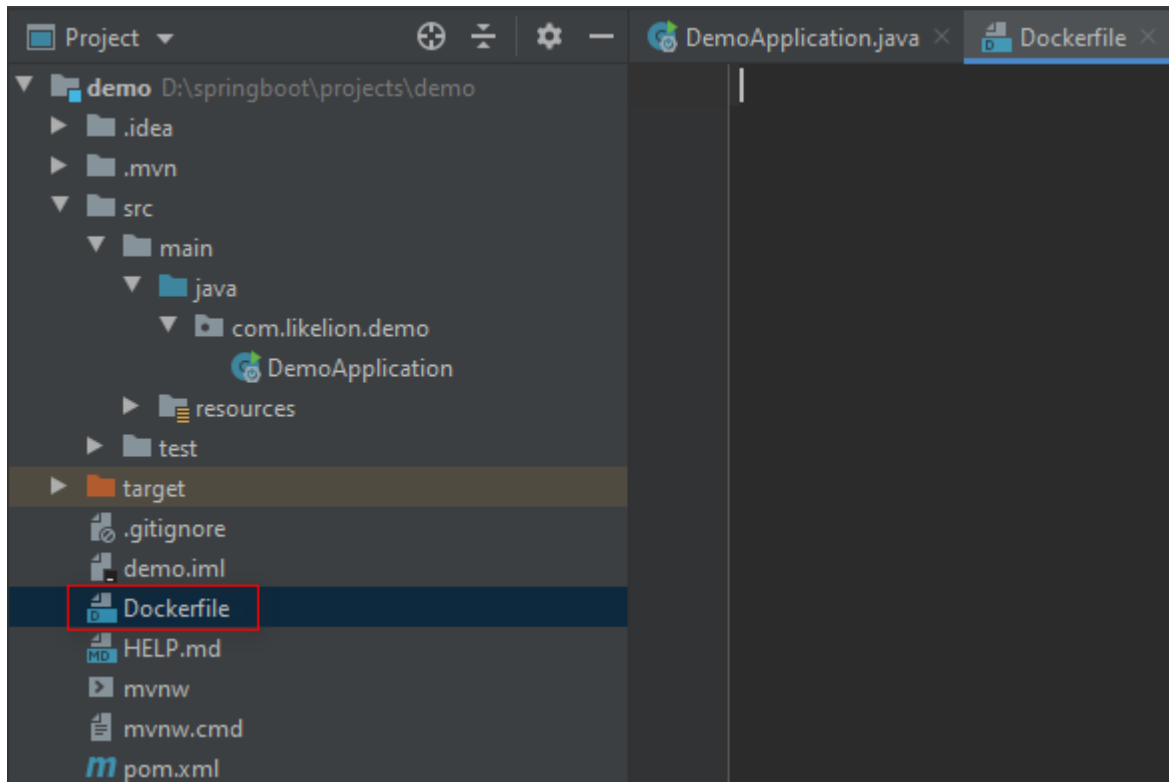


### Một số lệnh trong Dockerfile:

- ❑ **FROM <base\_image>:<phiên\_bản>**: đây là câu lệnh bắt buộc phải có trong bất kỳ Dockerfile nào. Nó dùng để khai báo base Image mà chúng ta sẽ build mới Image của chúng ta.
- ❑ **RUN <câu\_lệnh>**: chúng ta sử dụng lệnh này để chạy một command cho việc cài đặt các công cụ cần thiết cho Image của chúng ta.
- ❑ **ENV <tên\_biến>**: định nghĩa biến môi trường trong Container.
- ❑ **ENTRYPOINT <câu\_lệnh>**: định nghĩa những command mặc định, cái mà sẽ được chạy khi container running.
- ❑ **VOLUME <tên\_thư\_mục>**: dùng để truy cập hoặc liên kết một thư mục nào đó trong Container.



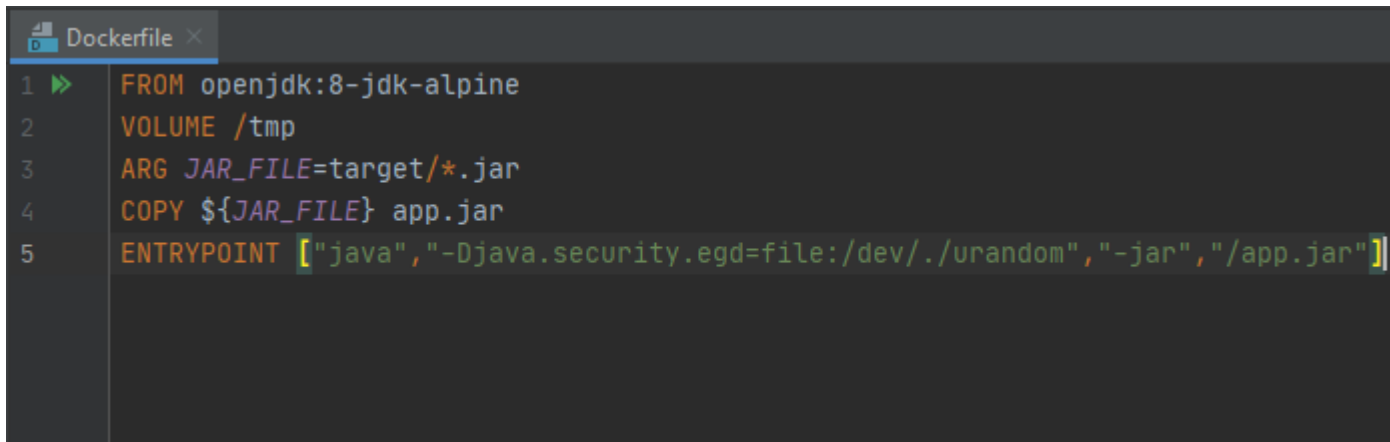
- ❑ Viết Dockerfile: Tạo một tập tin Dockerfile nằm trong thư mục của project



- ❑ Sửa Dockerfile này để xây dựng một Docker Image

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

- ❑ File Dockerfile



```
Dockerfile x
1  FROM openjdk:8-jdk-alpine
2  VOLUME /tmp
3  ARG JAR_FILE=target/*.jar
4  COPY ${JAR_FILE} app.jar
5  ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

### ❑ Chú ý

- ❖ Sử dụng một Image trên Docker Hub để build Image
  - ❖ Image đó có tên là openjdk:8-jdk-alpine
  - ❖ Dùng câu lệnh FROM để thực hiện tạo image đó
    - *FROM openjdk:8-jdk-alpine*
  - ❖ Câu lệnh *VOLUME* /tmp dùng để dẫn thư mục Docker Container vào thư mục của Docker
  - ❖ Câu lệnh *ARG* dùng để tạo biến, chỉ khả dụng trong quá trình build docker image
-

## ❑ Những lệnh cơ bản trong Docker

- ❖ Build một image: *docker build -t <tên image>.*
  - ❖ Xem các image trong Docker: *docker image ls*
  - ❖ Chạy image: *docker run --name <tên container> -p8080:8080 <tên image>*
    - Trong đó: *-p8080:8080* là mapping cổng 8080 của container với cổng 8080 của máy
  - ❖ Tạo tag từ image: *docker tag <tên image> <account name>/<tên repository>:<tagname>*
  - ❖ Push lên Docker Hub: *docker push <account name>/<tên repository>:<tagname>*
  - ❖ Pull image từ Docker Hub: *docker pull <account name>/<tên repository>:<tagname>*
-

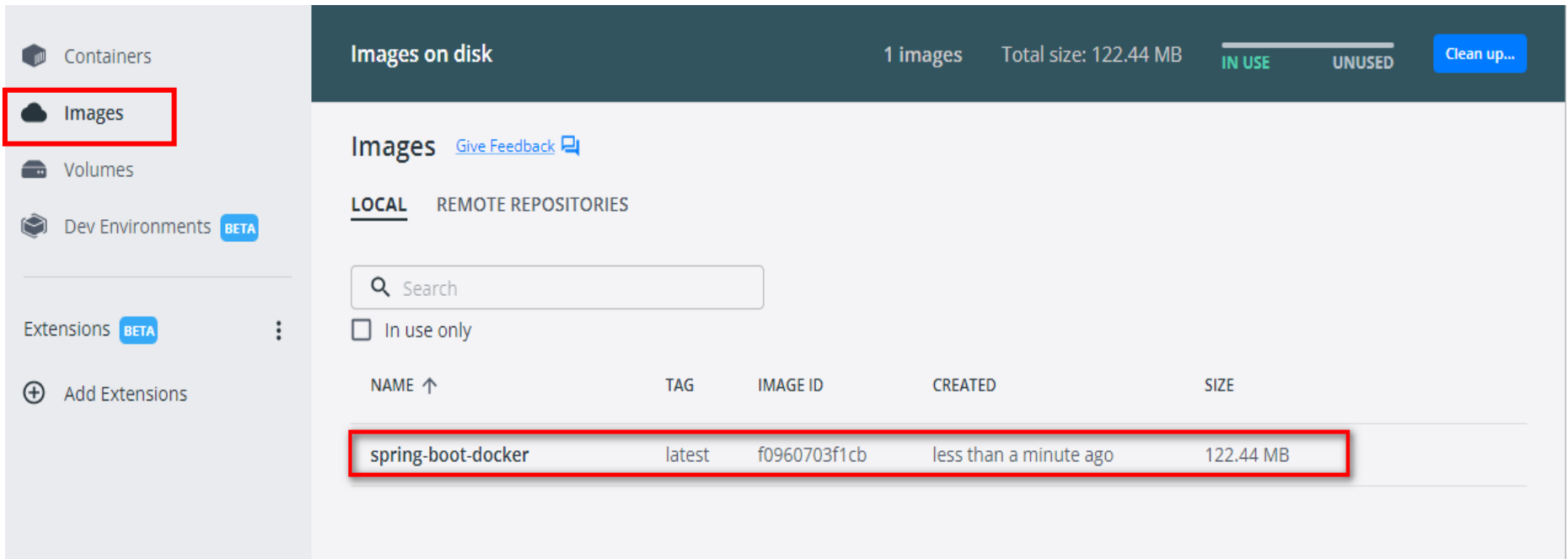
- ❑ Mở Terminal trong IntelliJ và chạy dòng lệnh sau

`mvn clean package -DskipTests && docker build -t spring-boot-docker .`

- ❑ Câu lệnh trên có 2 phần

- ❖ Phần đầu tiên là sẽ dùng Maven để build ứng dụng Spring Boot, skip tất cả unit test
- ❖ Phần tiếp theo là dùng Docker để build Docker Image từ Dockerfile
  - `spring-boot-docker` là tên của Image

📁 Mở Docker Desktop → vào tab Images để xem kết quả



The screenshot shows the Docker Desktop interface. On the left sidebar, the 'Images' menu item is highlighted with a red box. The main panel displays 'Images on disk' with a summary bar indicating '1 images' and 'Total size: 122.44 MB'. Below this, there is a search bar and a filter for 'In use only'. A table lists the images, with the first row 'spring-boot-docker' highlighted by a red box. The table has columns for NAME, TAG, IMAGE ID, CREATED, and SIZE.

NAME ↑	TAG	IMAGE ID	CREATED	SIZE
spring-boot-docker	latest	f0960703f1cb	less than a minute ago	122.44 MB

- ❑ Mở Terminal chạy câu lệnh sau để kiểm tra tất cả Image hiện có  
*docker image ls*

- ❑ Demo

```
D:\springboot\projects\demo>docker image ls
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
spring-boot-docker  latest      f0960703f1cb  11 minutes ago 122MB

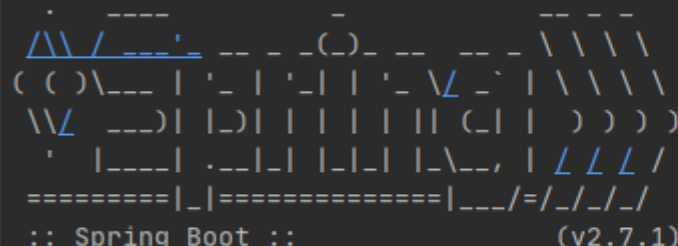
D:\springboot\projects\demo>
```

- ❑ Chạy Image vừa tạo bằng câu lệnh sau

*docker run --name springboot-container -p8080:8080 spring-boot-docker*

- ❑ Demo

```
D:\springboot\projects\demo>docker run --name springboot-container -p8080:8080 spring-boot-docker
```



```
:: Spring Boot ::                (v2.7.1)
```

- ❑ Mở trình duyệt và truy cập vào <http://localhost:8080/> để kiểm tra kết quả



Chạy Image thành công



- ❑ Để dừng Image sử dụng câu lệnh sau

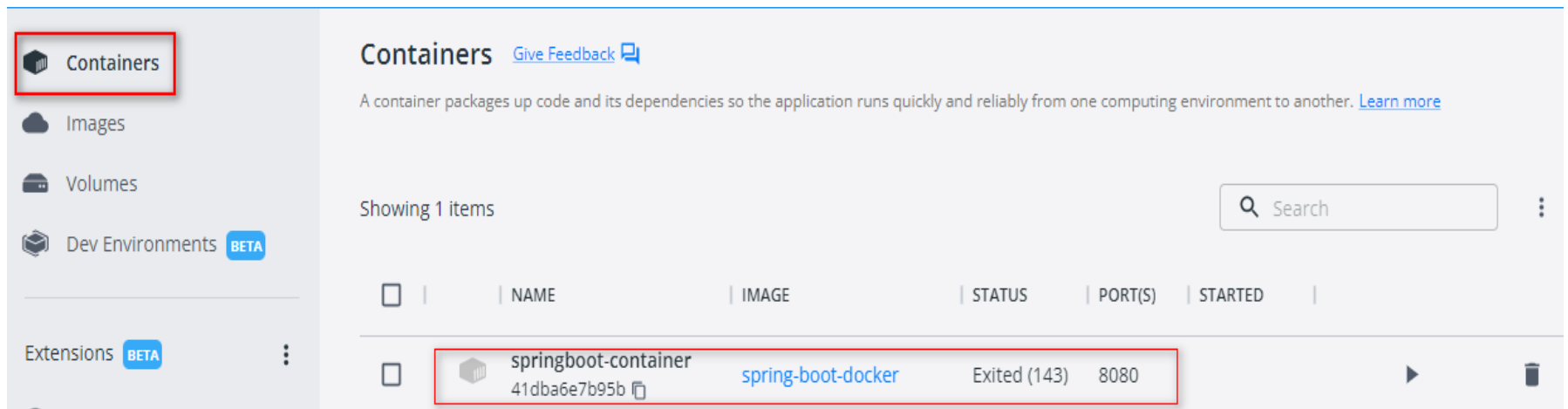
*docker stop springboot-container*

- ❑ Demo

```
D:\springboot\projects\demo>docker stop springboot-container  
springboot-container
```

- ❑ Lưu ý

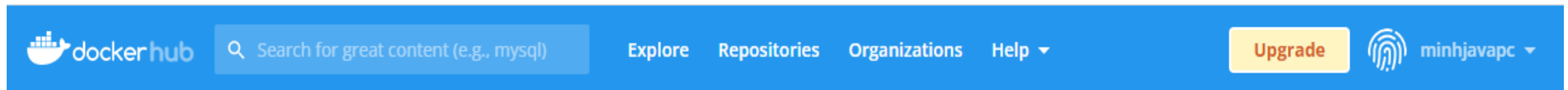
❖ Springboot-container là tên của container



The screenshot shows the Docker Desktop interface. On the left sidebar, the 'Containers' tab is selected and highlighted with a red box. The main panel displays the 'Containers' section with a search bar and a table of containers. The table has columns for NAME, IMAGE, STATUS, PORT(S), and STARTED. One container is listed: 'springboot-container' with ID '41dba6e7b95b', image 'spring-boot-docker', status 'Exited (143)', and port '8080'. The container name and ID are highlighted with a red box.

	NAME	IMAGE	STATUS	PORT(S)	STARTED
<input type="checkbox"/>	springboot-container 41dba6e7b95b	spring-boot-docker	Exited (143)	8080	

- ❑ Trup cập <https://hub.docker.com/signup> để tạo tài khoản
- ❑ Sau khi tạo tài khoản cần **ghi nhớ** account name để thực hiện push image lên Docker Hub



minhjavapc

[Edit profile](#)



Community User



Joined July 7, 2022

[Repositories](#)

[Starred](#)

[Contributed](#)

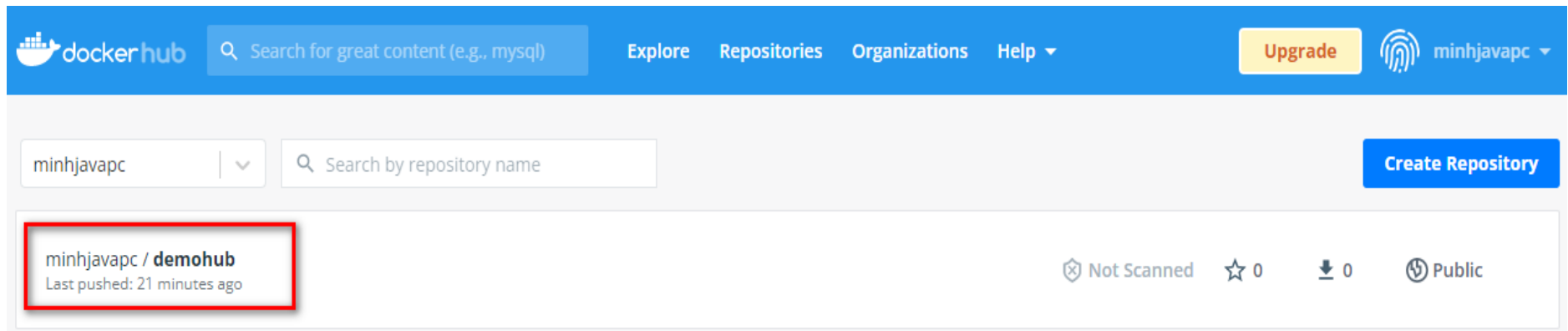
- ❑ Để push image lên Docker Hub trước tiên tạo “tag” từ image và repositories bằng câu lệnh sau

```
docker tag spring-boot-docker minhjavapc/demohub:part1
```

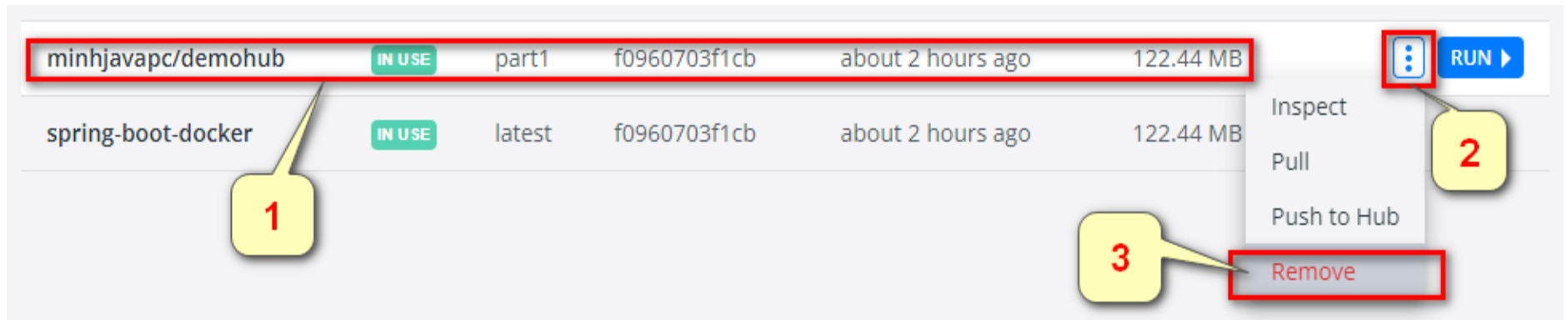
- ❑ Tiếp theo push lên Docker Hub bằng câu lệnh sau

```
docker push minhjavapc/demohub:part1
```

- ❑ Truy cập vào Docker Hub để kiểm tra



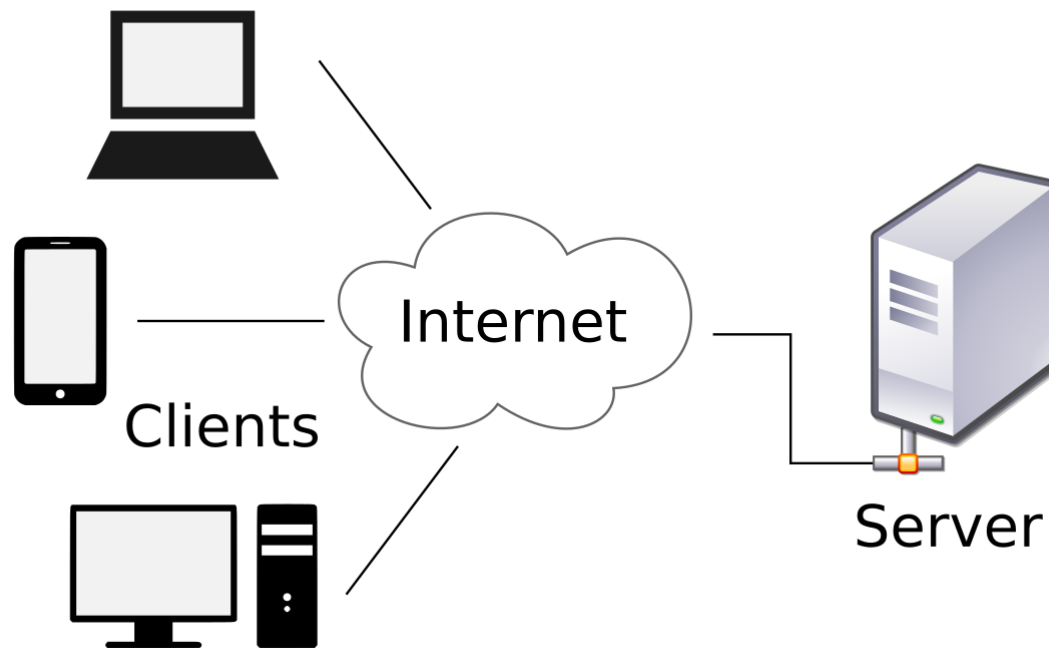
- ❑ Mở Docker Desktop, thực hiện xóa image vừa push lên Hub



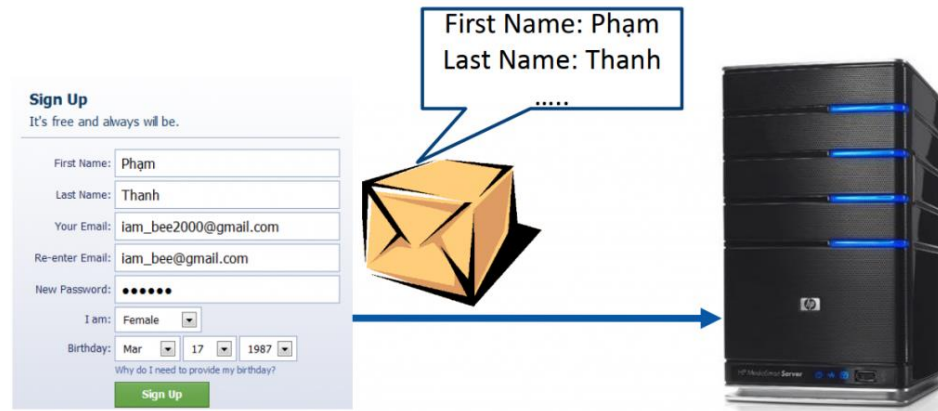
- ❑ Thực hiện pull image từ Docker Hub bằng câu lệnh sau

*docker pull minhjavapc/demohub:part1*

- ❑ Mô hình client - server là mô hình giúp các máy tính giao tiếp truyền tải dữ liệu cho nhau
- ❑ **Client:** Với vai trò là **máy khách**, chúng sẽ **không** cung cấp tài nguyên đến các máy tính khác mà chỉ sử dụng tài nguyên được cung cấp từ máy chủ
- ❑ **Server:** Là máy tính có khả năng **cung cấp** tài nguyên và các dịch vụ đến các máy khách khác trong hệ thống mạng.



- ❑ Trong mô hình Client Server, server **chấp nhận tất cả các yêu cầu** hợp lệ từ mọi nơi khác nhau trên Internet, sau đó trả kết quả về máy tính đã gửi yêu cầu đó
- ❑ Máy tính được coi là máy khách khi chúng làm nhiệm vụ **gửi yêu cầu** đến các máy chủ và đợi câu trả lời được gửi về
- ❑ Tuy nhiên, đối với Client và Server trên Web chúng ta cần phải có một phương thức **giao tiếp**, cụ thể là một **giao thức** để hai hệ thống có thể hiểu và **tương tác** với nhau. Giao thức này được gọi là giao thức truyền thông **HTTP**
- ❑ Trong đó hai phương thức phổ biến nhất là GET và POST
- ❑ **GET** và **POST** là hai phương thức của giao thức HTTP, đều là gửi dữ liệu đến server xử lý sau khi người dùng nhập thông tin vào form và thực hiện submit



### GET

- ❑ GET là phương thức của giao thức HTTP
- ❑ Trước khi gửi thông tin, nó sẽ được **mã hóa** bằng cách sử dụng một giản đồ gọi là url **encoding**. Giản đồ này là các cặp **name/value** được kết hợp với các **kí hiệu =** và các kí hiệu khác nhau được ngăn cách bởi dấu **&**
  - `http://www.example.com/index.htm?name=value1&name1=value1`

### POST

- ❑ Phương thức POST truyền thông tin thông qua HTTP header, thông tin này được **mã hóa** như phương thức GET
- ❑ Dữ liệu được gửi bởi phương thức POST rất **bảo mật** vì dữ liệu được gửi **ngầm, không** đưa lên URL, bằng việc sử dụng Secure HTTP
- ❑ Parameters được truyền trong **request body** nên có thể truyền dữ liệu **lớn**, hạn chế tùy thuộc vào cấu hình của Server
- ❑ POST không có bất kì hạn chế nào về **kích thước** dữ liệu sẽ gửi, có thể gửi dữ liệu nhị phân, hình ảnh

- ❑ Truy cập <https://start.spring.io/> để tạo project Spring Boot
  - ❑ Spring Boot sẽ chạy class có chú thích **@SpringBootApplication** đầu tiên
  - ❑ Sử dụng IntelliJ để lập trình Java
  - ❑ Cài đặt Docker, sử dụng CLI để thực hiện các lệnh build, run, stop... với Docker
  - ❑ Tạo image, thực hiện chạy container
  - ❑ Push image lên Docker Hub
  - ❑ Pull image từ Repository trên Docker Hub
  - ❑ Mô hình Client – Server là mô hình giúp các máy tính giao tiếp truyền tải dữ liệu cho nhau
  - ❑ Máy khách nhận tài nguyên - máy chủ cung cấp tài nguyên cho máy khách
  - ❑ Hai phương thức giao tiếp phổ biến giữa client và server là **GET** và **POST**
-



# Cảm Ơn Bạn Đã Chăm Chỉ!

