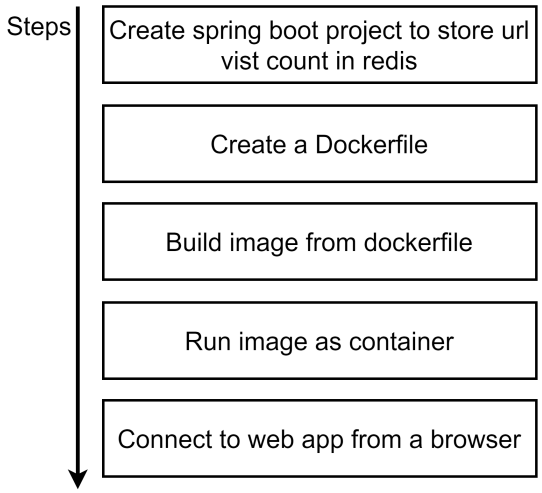
4. Developing a real world projects

In this section we will be building step by step a simple spring boot project that uses redis to store the count of the number of visits on a particular url.

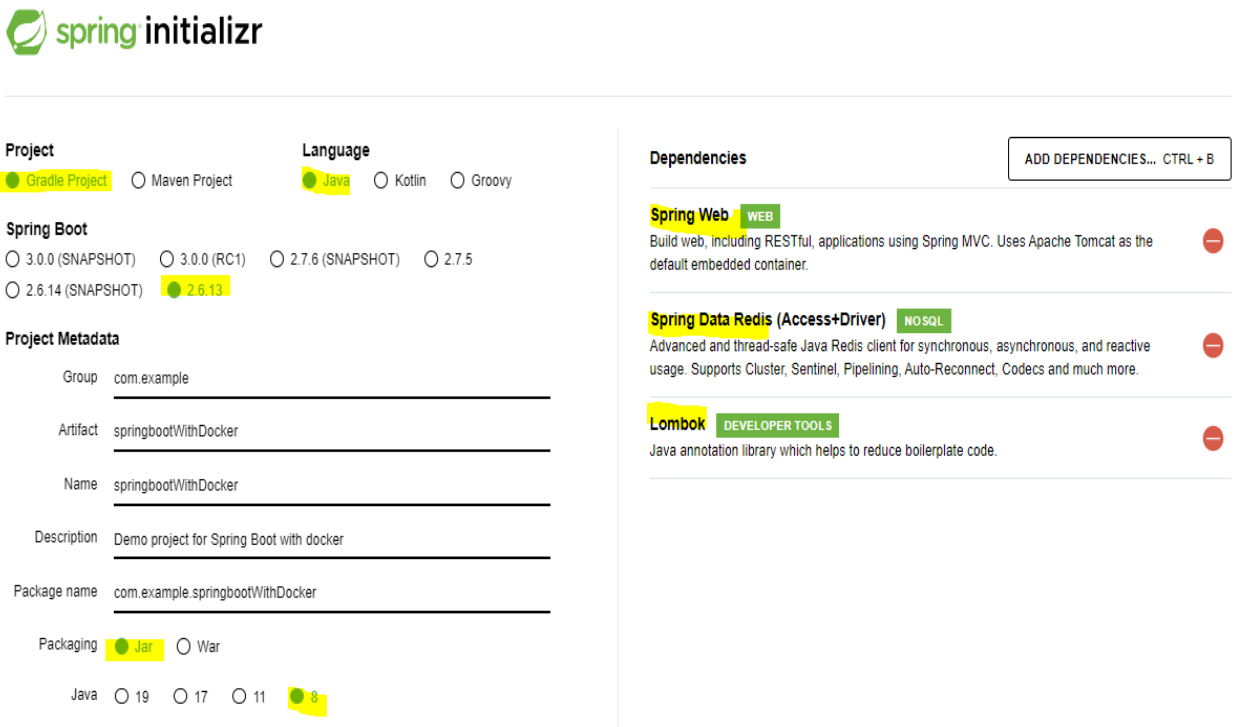
Steps:

1. Create initial spring boot project without redis
    a. Build the project
    b. Test using manually running jar file
2. Containerize the basic spring boot project without redis
3. Add redis to project and manual test the project
4. Containerizers the entire project with redis



4..1 Steps for manually initialize the project:

1. Navigate to https://start.spring.io. This service pulls in all the dependencies you need for an application and does most of the setup for you.
2. Choose either Gradle or Maven and the language you want to use. This guide assumes that you chose Java.
3. Click **Dependencies** and select **Spring Web**(For Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.**) ,Spring Data Redis** (For Advanced and thread-safe Java Redis client for synchronous, asynchronous, and reactive usage. Supports Cluster, Sentinel, Pipelining, Auto-Reconnect, Codecs and much more), **Lombok**(Java annotation library which helps to reduce boilerplate code.)
4. Click **Generate**.
5. Download the resulting ZIP file, which is an archive of a web application that is configured with your choices.



Source code :
<Link>

4.1.1 Code Snippet with explanation:

```java
@SpringBootApplication
@RestController
public class SpringbootWithDockerApplication {
  @RequestMapping("/")
  public String home() {
     return "Hello Docker World";
  }

  public static void main(String[] args) {
     SpringApplication.run(SpringbootWithDockerApplication.class, args);
  }

}
```

The class is flagged as a `@SpringBootApplication` and as a `@RestController`, meaning that it is ready for use by Spring MVC to handle web requests. `@RequestMapping` maps `/` to the `home()` method, which sends a `Hello World` response. The `main()` method uses Spring Boot's `SpringApplication.run()` method to launch an application.

4.1.2 Step to build and run the application[manual]

- If you use Gradle, run the following command:
    - **`./gradlew build && java -jar build/libs/<name of the jar file>.jar`**
- If you use Maven, run the following command:
    - **`./mvnw package && java -jar target/<name of the jar file>.jar`**
- Then go to localhost:8080 to see your "Hello Docker World" message.

Note: jar file will in build/libs/ folder

4.1.3 Few Commonly observed issues and solution for the same

While running the command ***`./gradlew build && java -jar build/libs/<name of the jar file>.jar`*** if following issue might be observed

- “***Could not find tools.jar. Please check that C:\Program Files\Java\jre1.8.0_151 contains a valid JDK installation.*** *If this* is observed then we need to set $JAVA_HOME variable properly in the environment variable section.
    - Few solution mentioned in below link will work
        - https://stackoverflow.com/questions/47291056/could-not-find-tools-jar-please-check-that-c-program-files-java-jre1-8-0-151-c

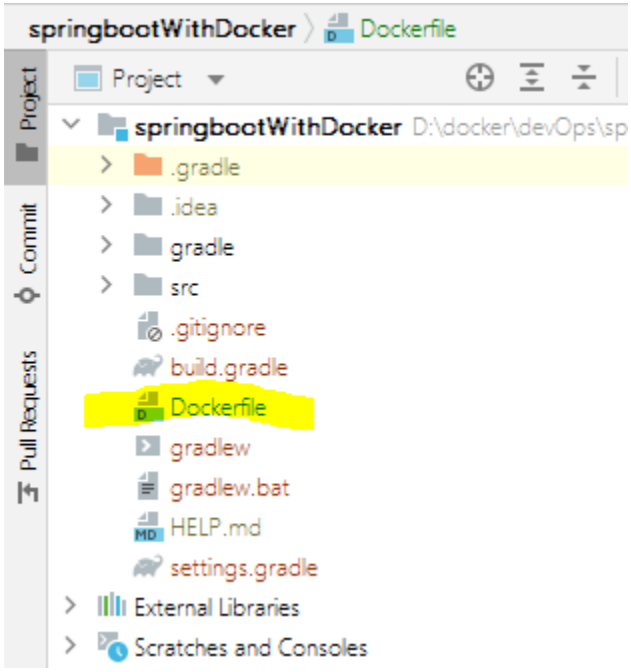
- **The token '&&' is not a valid statement separator in this version**

  If this issue is observed then run in cmd or git bash.
  https://stackoverflow.com/questions/65627536/the-token-is-not-a-valid-statement-separator-in-this-version

4.2 Containerizing application

4.2.1 Create a docker file

Step 1 : Project → New → File
Name the file as Dockerfile



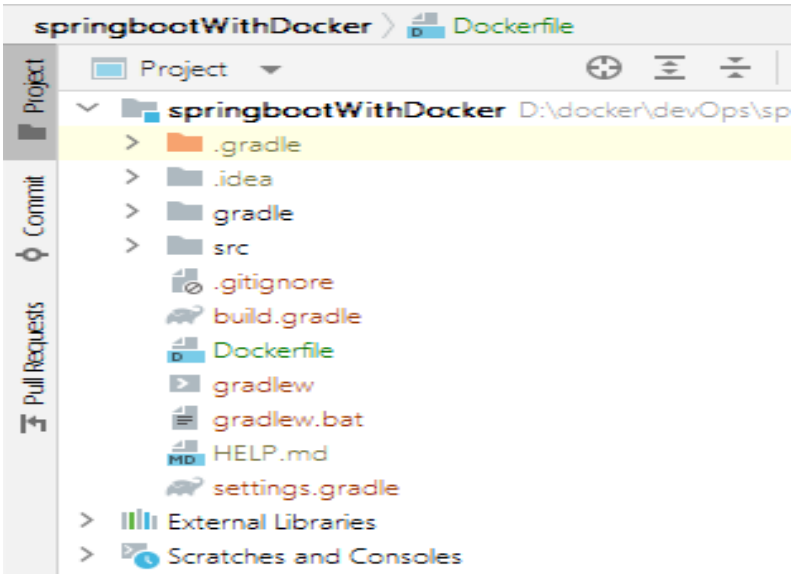
Step 2 : Dockerfile:

```
FROM openjdk:8-jdk-alpine
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} springbootWithDocker.jar
EXPOSE 8080
ENTRYPOINT ["java","-jar","/springbootWithDocker.jar"]
```

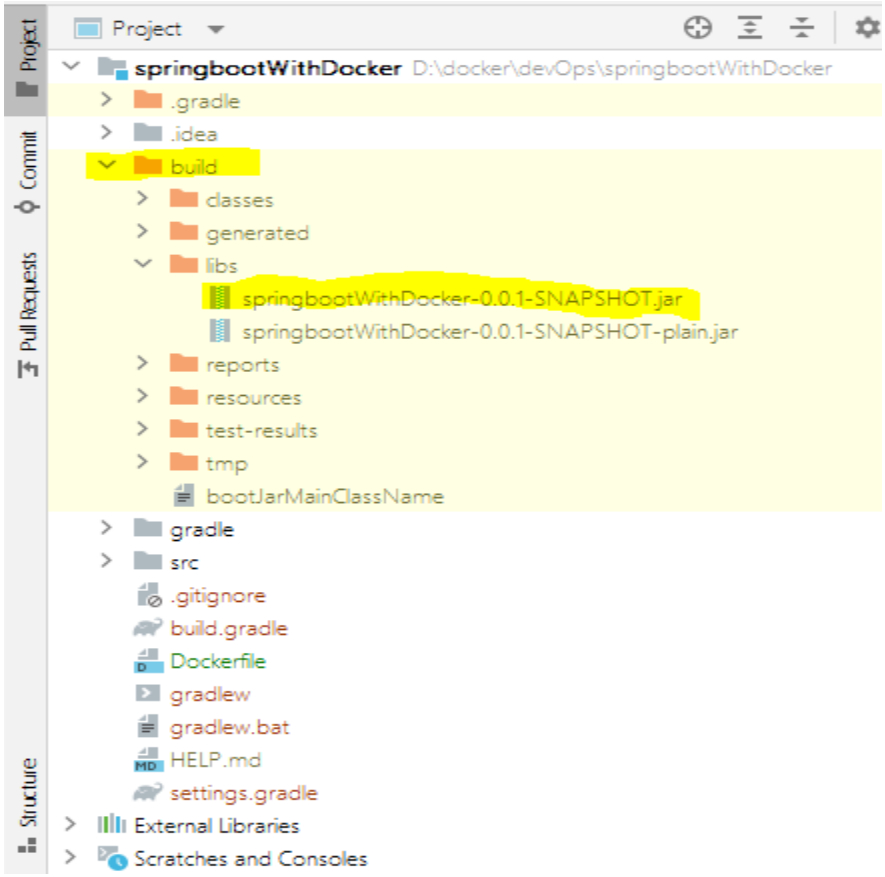
- **FROM:** We are building this image using openjdk:8 alpine image. JDK has many more images, we are using this version. You can check more images from the docker hub link
- **ARG:** We are taking jar path as ARG variable. Here our jar file is in **build/libs/springbootWithDocker-0.0.1-SNAPSHOT..jar** . ARG is only available during the build of a Docker image (RUN etc), not after the image is created and containers are started from it (ENTRYPOINT, CMD)
- **Copy:** Copying this jar file as springbootWithDocker.jar. We will execute the run command on this jar file with this name
- **ENTRYPOINT:** This will be executable to start when the container is booting. We must define them as *JSON-Array* because we will use an *ENTRYPOINT* in combination with a *CMD* for some application arguments. Here we are passing the jar file run commands
- **EXPOSE** The expose keyword in a Dockerfile tells Docker that a container listens for traffic on the specified port. So, for a container running a web server, you might add this to your Dockerfile: EXPOSE 8080 This tells Docker your web server will listen on port 8080 for TCP connections since TCP is the default. For UDP, specify the protocol after the port. **`EXPOSE 35/udp.`** If expose is not specified in the docker file each time when we run image we need to map the port. Example: docker run -p 8080:8080

4.2.2 Create a jar file

- Gradle command : ./gradlew build
  Before running the command file structure:



After running the command



4.2.3 Execute docker build command

$docker build .

```
D:\docker\devOps\springbootWithDocker>docker build .
[+] Building 16.2s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 32B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/openjdk:8-jdk-alpine
 => [internal] load build context
 => => transferring context: 27.68MB
 => CACHED [1/2] FROM docker.io/library/openjdk:8-jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd524164a0f2283343547b3
 => [2/2] COPY build/libs/*.jar springbootWithDocker.jar
 => exporting to image
 => => exporting layers
 => => writing image sha256:061811a794dbaa426ab429ab1c16d73d328376877c5cffbb8cd0b4237a7f2a97
```

4.2.4 Run Docker image

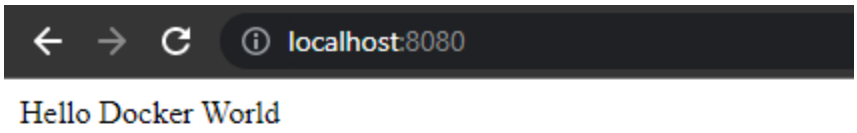
$docker run <container_id>

```
D:\docker\devOps\springbootWithDocker>docker run 061811a794dbaa426

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.6.13)

2022-10-23 15:42:22.977  INFO 1 --- [           main] c.e.s.SpringbootWithDockerApplication    : Starting SpringbootWithDockerApplication using Java 1.8.0_212 on 6f573883168b with PID 1 (/springbootWithDocker.jar start
ed by root in /)
2022-10-23 15:42:22.982  INFO 1 --- [           main] c.e.s.SpringbootWithDockerApplication    : No active profile set, falling back to 1 default profile: "default"
2022-10-23 15:42:24.473  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Multiple Spring Data modules found, entering strict repository configuration mode
2022-10-23 15:42:24.479  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data Redis repositories in DEFAULT mode.
2022-10-23 15:42:24.525  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 11 ms. Found 0 Redis repository interfaces.
2022-10-23 15:42:25.779  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2022-10-23 15:42:25.823  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-10-23 15:42:25.825  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.68]
2022-10-23 15:42:26.130  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-10-23 15:42:26.130  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2985 ms
2022-10-23 15:42:28.278  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2022-10-23 15:42:28.301  INFO 1 --- [           main] c.e.s.SpringbootWithDockerApplication    : Started SpringbootWithDockerApplication in 6.328 seconds (JVM running for 7.816)
```

Open the browser and run localhost:8080



Hello Docker World