

**Apertura:** martes, 27 de febrero de 2024, 14:30

**Cierre:** martes, 27 de febrero de 2024, 17:30

En este taller profundizamos los conceptos de modulación por medio de virtualización usando Docker y AWS.

### Pre requisitos

1. El estudiante conoce Java, Maven
2. El estudiante sabe desarrollar aplicaciones web en Java
3. Tiene instalado Docker en su máquina

## DESCRIPCIÓN

El taller consiste en crear una aplicación web pequeña usando el micro-framework de Spark java (<http://sparkjava.com/>). Una vez tengamos esta aplicación procederemos a construir un container para docker para la aplicación y los desplegaremos y configuraremos en nuestra máquina local. Luego, cerramos un repositorio en DockerHub y subiremos la imagen al repositorio. Finalmente, crearemos una máquina virtual en AWS, instalaremos Docker, y desplegaremos el contenedor que acabamos de crear.

### Primera parte crear la aplicación web

1. Cree un proyecto java usando maven.
2. Cree la clase principal

```
package co.edu.escuelaing.sparkdockerdemolive;

public class SparkWebServer {

    public static void main(String... args){
        port(getPort());
        get("hello", (req,res) -> "Hello Docker!");
    }

    private static int getPort() {
        if (System.getenv("PORT") != null) {
            return Integer.parseInt(System.getenv("PORT"));
        }
        return 4567;
    }
}
```

3. Importe las dependencias de spark Java en el archivo pom

```

<dependencies>
  <!-- https://mvnrepository.com/artifact/com.sparkjava/spark-core -->
  <dependency>
    <groupId>com.sparkjava</groupId>
    <artifactId>spark-core</artifactId>
    <version>2.9.4</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-api -->
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>2.0.12</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-simple -->
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>2.0.12</version>
  </dependency>
</dependencies>

```

4. Asegúrese que el proyecto esté compilando hacia la versión 8 de Java

```

<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>8</maven.compiler.source>
  <maven.compiler.target>8</maven.compiler.target>
</properties>

```

5. Asegúrese que el proyecto este copiando las dependencias en el directorio target al compilar el proyecto. Esto es necesario para poder construir una imagen de contenedor de docker usando los archivos ya compilados de java. Para hacer esto use el plugin de dependencias de Maven.

```

<!-- build configuration -->
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-dependency-plugin</artifactId>
      <version>3.0.1</version>
      <executions>
        <execution>
          <id>copy-dependencies</id>
          <phase>package</phase>
          <goals><goal>copy-dependencies</goal></goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>

```

6. Asegúrese que el proyecto compila

```
$> mvn clean install
```

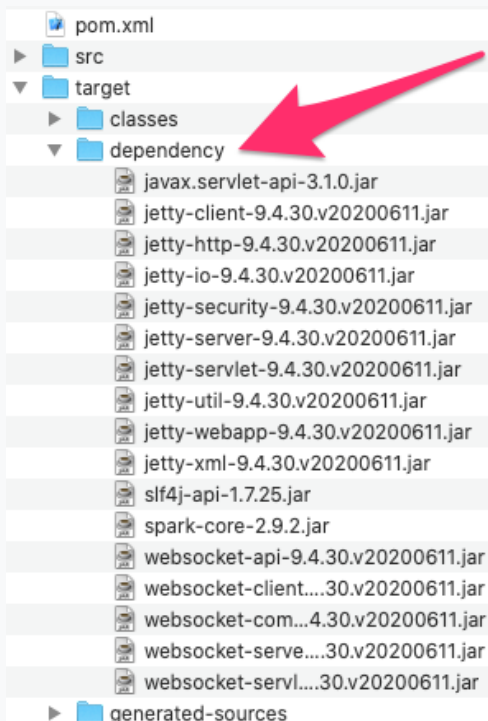
Debería obtener algo como esto:

```
[INFO] Scanning for projects...
[INFO]
[INFO] -----< co.edu.escuelaing:sparkDockerDemoLive >-----
[INFO] Building sparkDockerDemoLive 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ sparkDockerDemoLive ---
[INFO] Deleting /Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ sparkDockerDemoLive ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO]
    skip non existing resourceDirectory
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ sparkDockerDemoLive ---
[INFO] Changes detected - recompiling the module!
[INFO]
    Compiling 1 source file to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ sparkDockerDemoLive ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO]
    skip non existing resourceDirectory
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ sparkDockerDemoLive ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ sparkDockerDemoLive ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ sparkDockerDemoLive ---
[INFO]
    Building jar:
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/sparkDockerDemoLive-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- maven-dependency-plugin:3.0.1:copy-dependencies (copy-dependencies) @ sparkDockerDemoLive ---
[INFO]
    Copying spark-core-2.9.2.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/spark-core-2.9.2.jar
[INFO]
    Copying slf4j-api-1.7.25.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/slf4j-api-1.7.25.jar
[INFO]
    Copying jetty-server-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/jetty-server-9.4.30.v20200611.jar
[INFO]
    Copying javax.servlet-api-3.1.0.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/javax.s
api-3.1.0.jar
[INFO]
    Copying jetty-http-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/jetty-
http-9.4.30.v20200611.jar
```

```
[INFO]
Copying jetty-util-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/jetty-
util-9.4.30.v20200611.jar
[INFO]
Copying jetty-io-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/jetty-
io-9.4.30.v20200611.jar
[INFO]
Copying jetty-webapp-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/jetty-
webapp-9.4.30.v20200611.jar
[INFO]
Copying jetty-xml-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/jetty-
xml-9.4.30.v20200611.jar
[INFO]
Copying jetty-servlet-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/jetty-
servlet-9.4.30.v20200611.jar
[INFO]
Copying jetty-security-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/jetty-
security-9.4.30.v20200611.jar
[INFO]
Copying websocket-server-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/websock
server-9.4.30.v20200611.jar
[INFO]
Copying websocket-common-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/websock
common-9.4.30.v20200611.jar
[INFO]
Copying websocket-client-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/websock
client-9.4.30.v20200611.jar
[INFO]
Copying jetty-client-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/jetty-
client-9.4.30.v20200611.jar
[INFO]
Copying websocket-servlet-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/websock
servlet-9.4.30.v20200611.jar
[INFO]
Copying websocket-api-9.4.30.v20200611.jar to
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/dependency/websock
api-9.4.30.v20200611.jar
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ sparkDockerDemoLive ---
[INFO]
Installing
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/target/sparkDockerDemoLiv
1.0-SNAPSHOT.jar
to
/Users/dnielben/.m2/repository/co/edu/escuelaing/sparkDockerDemoLive/1.0-SNAPSHOT/sparkDockerDemoLive-1.0-
SNAPSHOT.jar
[INFO]
Installing
/Users/dnielben/Dropbox/01Escritorio/03Programas/AREP2020Talleres/sparkDockerDemoLive/pom.xml
to
```

```
/Users/dnielben/.m2/repository/co/edu/escuelaing/sparkDockerDemoLive/1.0-SNAPSHOT/sparkDockerDemoLive-1.0-SNAPSHOT.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.971 s
[INFO] Finished at: 2020-09-11T18:47:46-05:00
[INFO] -----
```

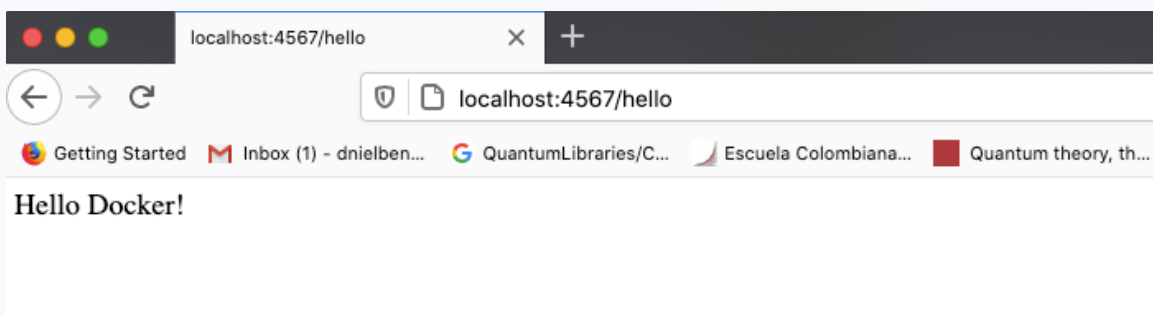
7. Asegúrese que las dependencias están en el directorio target y que continentemente las dependencia, es decir las librerías necesarias para correr en formato jar. En este caso solo son las dependencias necesarias para correr SparkJava.



8. Ejecute el programa invocando la máquina virtual de Java desde la línea de comandos y acceda la url `http://localhost:4567/hello`:

```
java -cp "target/classes:target/dependency/*" co.edu.escuelaing.sparkdockerdemolive.SparkWebServer
```

Debería ver algo así:



## Segunda Parte: crear imagen para docker y subirla

1. En la raíz de su proyecto cree un archivo denominado Dockerfile con el siguiente contenido:

```
FROM openjdk:8

WORKDIR /usrapp/bin

ENV PORT 6000

COPY /target/classes /usrapp/bin/classes
COPY /target/dependency /usrapp/bin/dependency

CMD ["java", "-cp", "./classes:./dependency/*", "co.edu.escuelaing.sparkdockerdemolive.SparkWebServer"]
```

2. Usando la herramienta de línea de comandos de Docker construya la imagen:

```
docker build --tag dockersparkspring .
```

3. Revise que la imagen fue construida

```
docker images
```

Debería ver algo así:

```
%> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
dockersparkspring   latest             0c5dd4c040f2       49 seconds ago     514MB
openjdk              8                 db530b5a3ccf       39 hours ago       511MB
```

4. A partir de la imagen creada cree tres instancias de un contenedor docker independiente de la consola (opción “-d”) y con el puerto 6000 enlazado a un puerto físico de su máquina (opción -p):

```
docker run -d -p 34000:6000 --name firstdockercontainer dockersparkspring
docker run -d -p 34001:6000 --name firstdockercontainer2 dockersparkspring
docker run -d -p 34002:6000 --name firstdockercontainer3 dockersparkspring
```

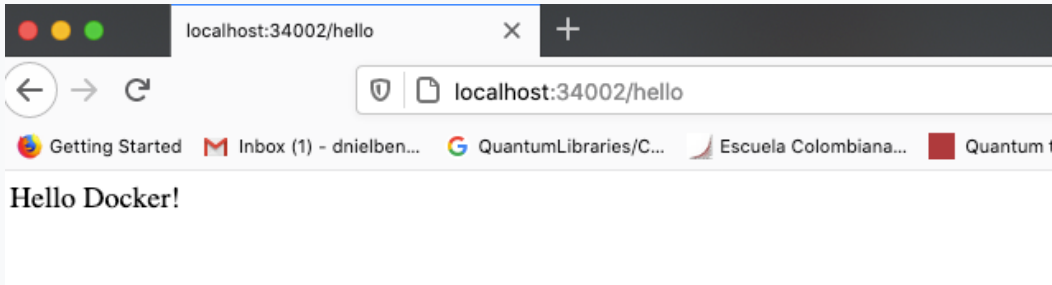
5. Asegúrese que el contenedor está corriendo

```
docker ps
```

Debería ver algo así:

```
%> docker ps
CONTAINER
ID                IMAGE                COMMAND              CREATED
STATUS           PORTS               NAMES
4e44267d49c0     dockersparkspring    "java -cp ./classes:..." 4 minutes ago
Up 3 minutes     0.0.0.0:34002->6000/tcp  firstdockercontainer3
dd96c59d9798     dockersparkspring    "java -cp ./classes:..." 4 minutes ago
Up 4 minutes     0.0.0.0:34001->6000/tcp  firstdockercontainer2
45f9b2769633     dockersparkspring    "java -cp ./classes:..." 6 minutes ago
Up 6 minutes     0.0.0.0:34000->6000/tcp  firstdockercontainer
```

6. Acceda por el browser a <http://localhost:34002/hello>, o a <http://localhost:34000/hello>, o a <http://localhost:34001/hello> para verificar que están corriendo.



9. Use docker-compose para generar automáticamente una configuración docker, por ejemplo un container y una instancia a de mongo en otro container. Cree en la raíz de su directorio el archivo docker-compose.yml con le siguiente contenido:

```
version: '2'

services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: web
    ports:
      - "8087:6000"
  db:
    image: mongo:3.6.1
    container_name: db
    volumes:
      - mongodb:/data/db
      - mongodb_config:/data/configdb
    ports:
      - 27017:27017
    command: mongod

volumes:
  mongodb:
  mongodb_config:
```

10 Ejecute el docker compose:

```
docker-compose up -d
```

11. Verifique que se crearon los servicios:

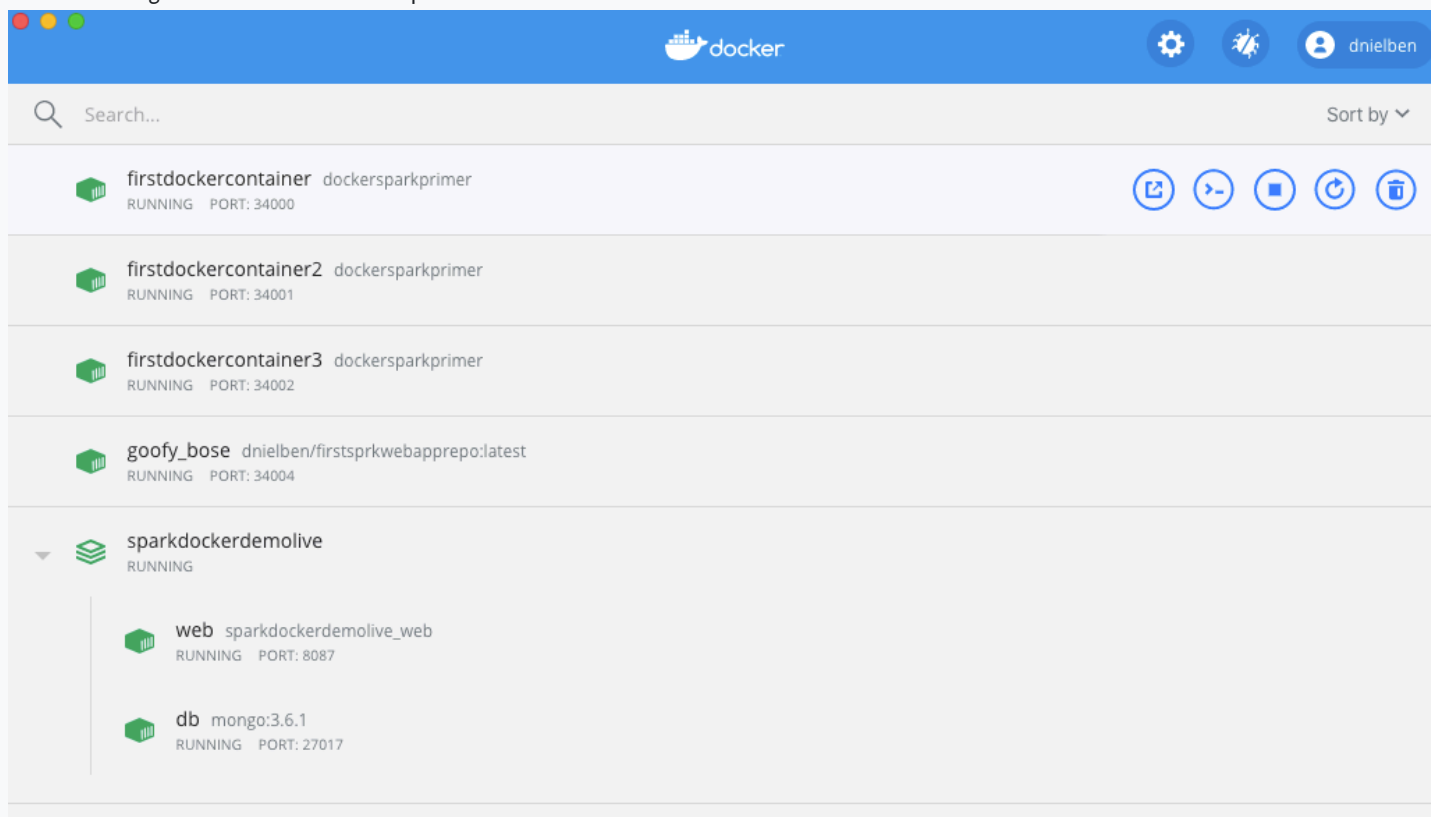
```
docker ps
```

Debería ver algo así en consola



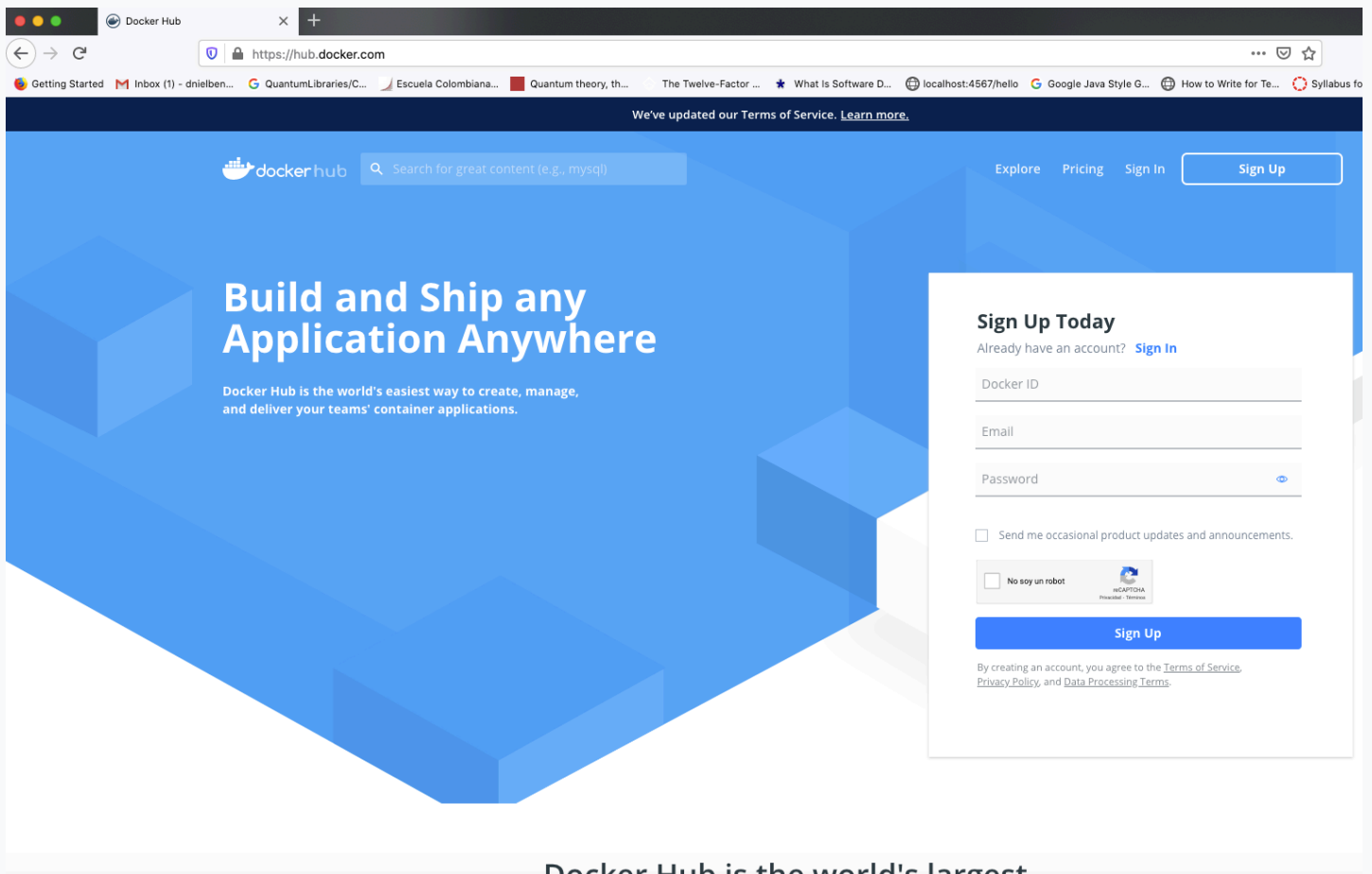
```
%> docker ps
CONTAINER
ID          IMAGE          COMMAND
CREATED    STATUS        PORTS
NAMES
498500a0c6c6
  mongo:3.6.1    "docker-entrypoint.s..." 2 hours
ago          Up 2 hours    0.0.0.0:27017->27017/tcp  db
394d835ccf8c
  sparkdockerdemolive_web    "java -cp ./classes:..." 2
hours ago    Up 2 hours    0.0.0.0:8087->6000/tcp    web
250ecaac59ca
  danielben/firstsprkwebapprepo:latest "java -cp ./classes:..." 3
hours ago    Up 3 hours    0.0.0.0:34004->6000/tcp
goofy_bose
4e44267d49c0
  0c5dd4c040f2    "java -cp ./classes:..." 3
hours ago    Up 3 hours    0.0.0.0:34002->6000/tcp
firstdockercontainer3
dd96c59d9798
  0c5dd4c040f2    "java -cp ./classes:..." 3
hours ago    Up 3 hours    0.0.0.0:34001->6000/tcp
firstdockercontainer2
45f9b2769633
  0c5dd4c040f2    "java -cp ./classes:..." 3
hours ago    Up 3 hours    0.0.0.0:34000->6000/tcp
firstdockercontainer
```

Debería ver algo así en el Docker Desktop dashboard:

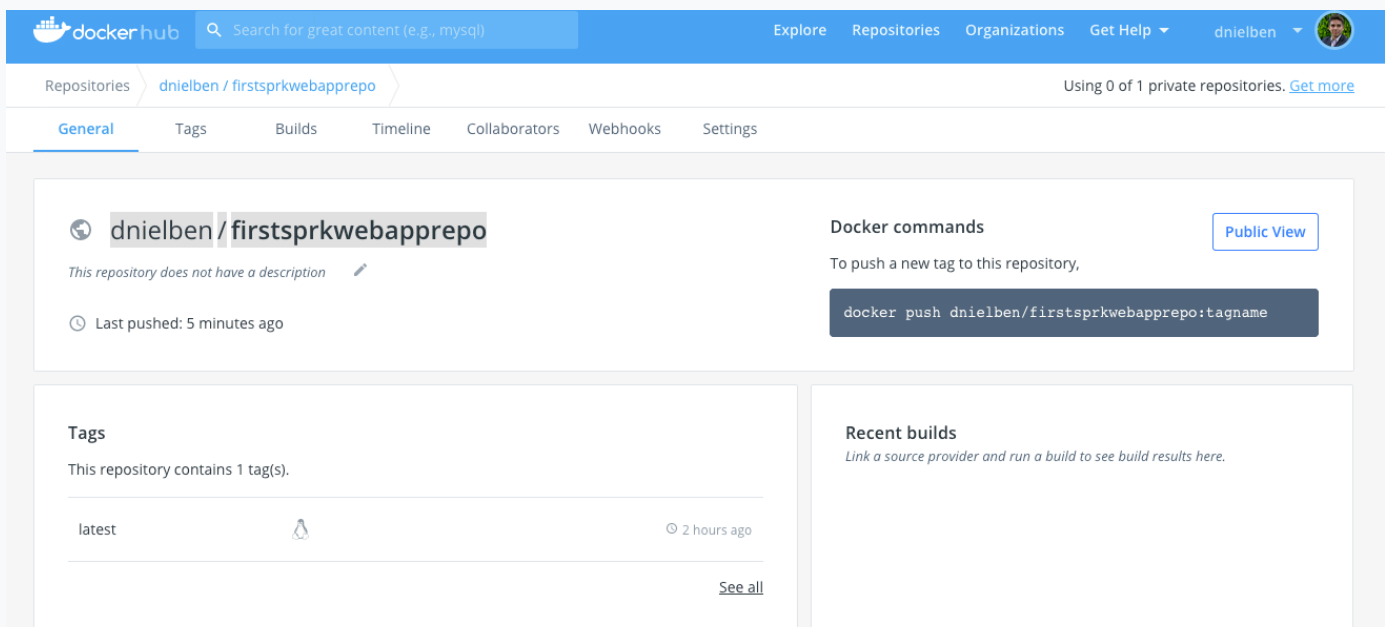


### Tercera para subir la imagen a Docker Hub

1. Cree una cuenta en Dockerhub y verifique su correo.



## 2. Acceda al menu de repositorios y cree un repositorio



## 3. En su motor de docker local cree una referencia a su imagen con el nombre del repositorio a donde desea subirla:

```
docker tag dockersparkprimer danielben/firstsprkwebapprepo
```

\*\* Si desea puede usar tags para poner nombre específicos, como solo tenemos una imagen simplemente creamos una referencia con el nombre de repositorio y dejamos el mismo nombre de tag, en este caso "latest"

## 4. Verifique que la nueva referencia de imagen existe

```
docker images
```

Debería ver algo así:

```
%> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
danielben/firstsprkwebapprepo  latest             0c5dd4c040f2       26 minutes ago     514MB
dockersparkprimer      latest             0c5dd4c040f2       26 minutes ago     514MB
openjdk               8                 db530b5a3ccf       39 hours ago       511MB
```

5. Autentíquese en su cuenta de dockerhub (ingrese su usuario y clave si es requerida):

```
docker login
```

6. Empuje la imagen al repositorio en DockerHub

```
docker push danielben/firstsprkwebapprepo:latest
```

En la solapa de Tags de su repositorio en Dockerhub debería ver algo así:

The screenshot shows the Docker Hub web interface. At the top is a blue navigation bar with the Docker Hub logo, a search bar, and links for Explore, Repositories, Organizations, Get Help, and a user profile for 'danielben'. Below the navigation bar, the breadcrumb 'Repositories > danielben / firstsprkwebapprepo' is visible. The main content area has tabs for General, Tags (selected), Builds, Timeline, Collaborators, Webhooks, and Settings. Under the 'Tags' tab, there's a filter section with an 'Action' dropdown and a 'Filter Tags' search box. The tags are sorted by 'Latest'. A single tag, 'latest', is listed. It shows 'Last updated 2 hours ago by danielben'. Below the tag name, the 'DIGEST' is '3805755c745c', the 'OS/ARCH' is 'linux/amd64', and the 'COMPRESSED SIZE' is '222.21 MB'. On the right side of the tag entry, there's a button that says 'docker pull danielben/firstsprkwebapprepo:latest' with a copy icon.

## TAREA

Modifique su proyecto publicando servicios REST para calcular:

- Sin
- Cos
- Determinar si una cadena es un palíndromo
- Retorna la magnitud de un vector real de dos dimensiones, es decir recibe dos parámetros reales.

Modifique el Cliente para que se puedan llamar asíncronamente estas funciones, por ejemplo con una formulario para cada función.

### Entregables:

1. El código del proyecto en un repositorio de GITHUB

2. Un README que explique un resumen del proyecto, la arquitectura, el diseño de clases y que muestre cómo generar las imágenes para desplegarlo. Además que muestre imágenes de cómo quedó desplegado cuando hicieron las pruebas.

Editar entrega

Borrar entrega

## ESTADO DE LA ENTREGA

Estado de la entrega	Enviado para calificar
Estado de la calificación	Calificado
Tiempo restante	La tarea fue enviada 1 día 22 horas después
Última modificación	jueves, 29 de febrero de 2024, 15:49
Texto en línea	<div><div>+</div><div><a href="https://github.com/ELS4NTA/AREP-LAB-Q5.git">https://github.com/ELS4NTA/AREP-LAB-Q5.git</a></div></div>
Comentarios de la entrega	<div><div></div><div>Comentarios (0)</div></div>

## COMENTARIO

Calificación	49,00 / 50,00
Calificado sobre	viernes, 22 de marzo de 2024, 21:32
Calificado por	<div><div></div><div>JULIAN DAVID CASTILLO SOTO</div></div>

	Referencia	Evaluación
Entregables	7	7
Desplegado en github	1	1
Tiene .gitignore completo	1	1
Tiene README.md	1	1
No contiene archivos o carpetas basura	1	1
Tiene POM.xml	1	1
Respeto estructura de maven	1	1
No contiene la carpeta target	1	1
Diseño y Arquitectura	28	27
El diseño del sistema parece razonable para el problema	3	3
El diseño está bien documentado en el README.md	3	3
El README contiene instrucciones de instalación y uso	3	3
El README evidencia pruebas	3	3
Tiene pruebas automáticas	3	3
El repositorio se puede clonar y ejecutar	3	3
El desarrollo es completo y de alta calidad	10	9
Total	35	34
Nota	5	4.85714286

[Biblioteca](#)

[Investigación e innovación](#)


[Enlace - Académico](#)


[Ministerio de Educación Nacional](#)

[Colombia Aprende](#)

[Red Latinoamericana de Portales Educativos](#)

[Red Universitarias Metropolitana de Bogotá](#)

 AK.45 No.205-59 (Autopista Norte).

 Teléfono: +57(1) 668 3600

 Email: [contactocc@escuelaing.edu.co](mailto:contactocc@escuelaing.edu.co)

Escuela Colombiana de Ingeniería Julio Garavito