**Test Project**

**Contents**

1. Introduction
2. Environment
3. Project Installation
4. Project Explanation
5. **Introduccion**

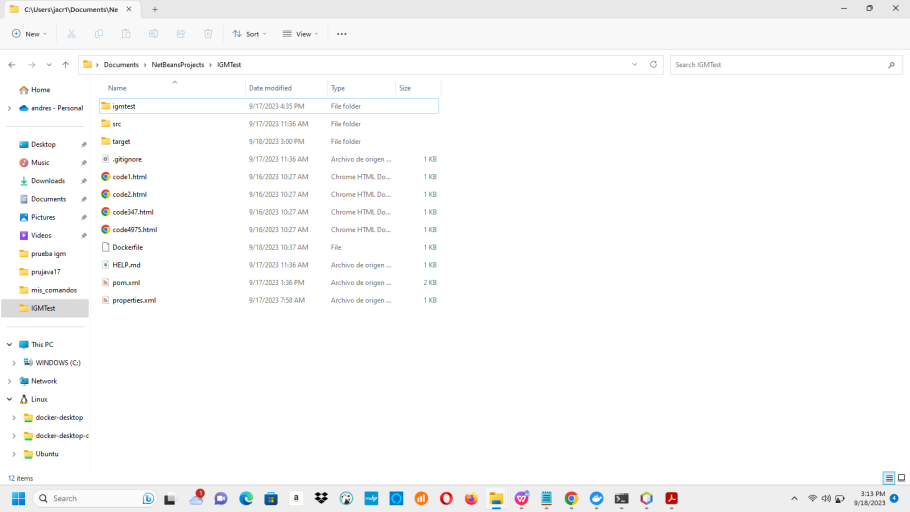
This document explains the organization of the archives of the test project.

1. **Enviroment**

* Hardware: HP (2,8 GHz i7 4 núcleos, 16 GB 1600 MHz DDR3)
* Operative System: Windows 11 Home
* Development environment: Netbeans 16
* Java 17 (openjdk:17.0.2-jdk)
* SpringBoot (3.1.3)
* Maven (3.8.6)
* Docker Desktop (4.19)
* Git (2.33.0.windows.2)

1. **Project Installation**

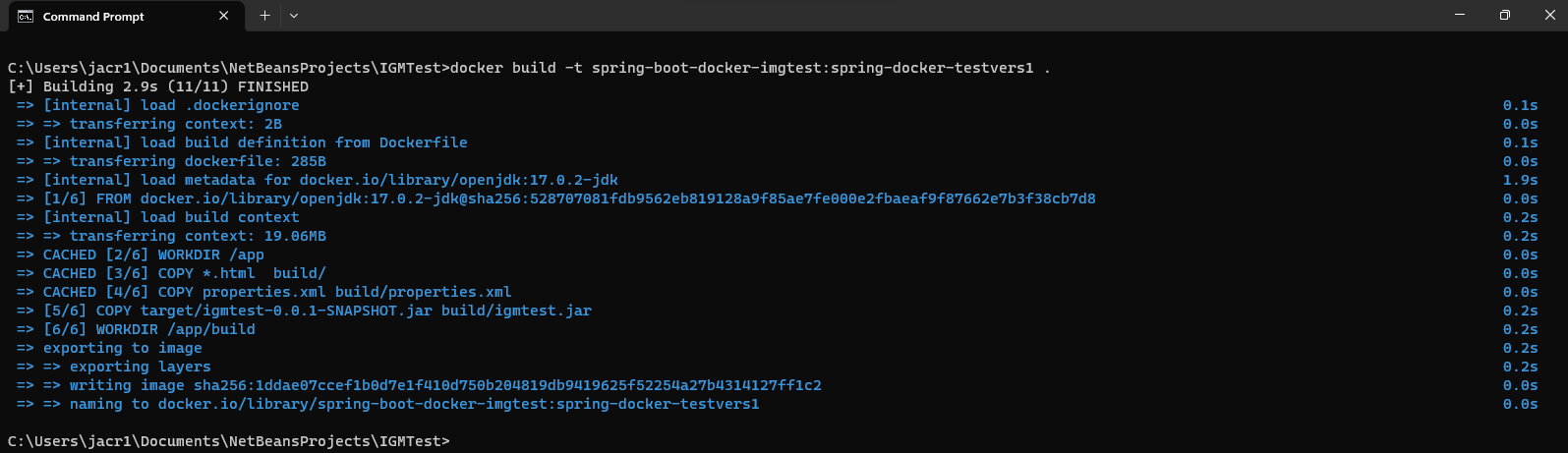
Folder of archives:



To run the Dockerfile it must be located in the folder shown in the image above.

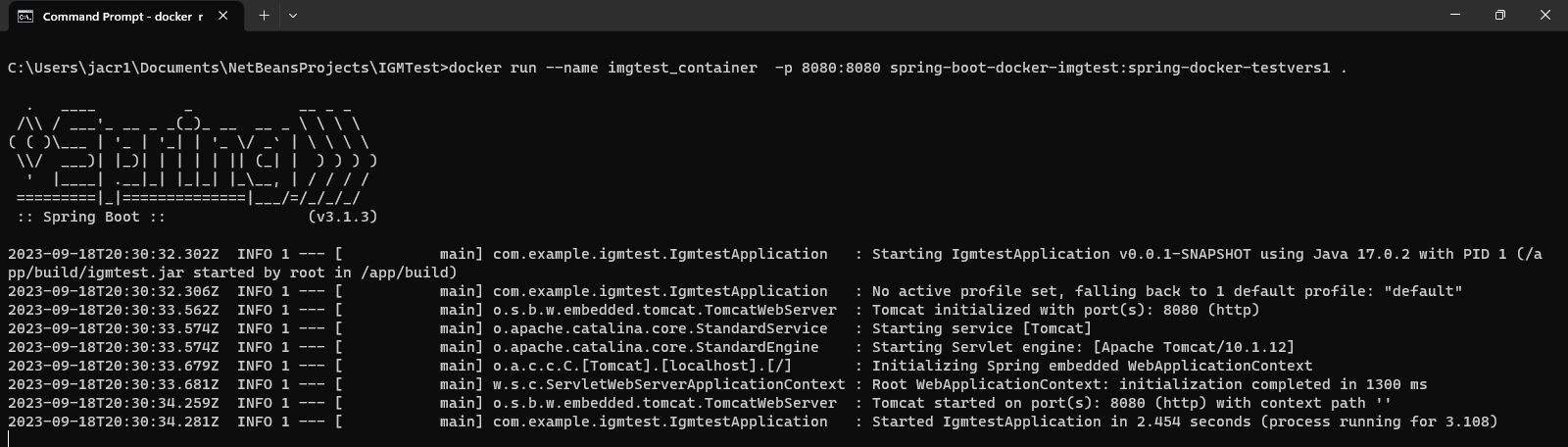
Command to create the docker image:

docker build -t spring-boot-docker-imgtest:spring-docker-testvers1 .



Command to create the docker container and run it:

docker run --name imgtest\_container -p 8080:8080 spring-boot-docker-imgtest:spring-docker-testvers1 .



1. **Project Explanation**

The project is made in SpringBoot 3.1.3 and deploys a jar file, its name is imgtest.

In the controller package:

com.example.igmtest.controller

There is a warning that is displayed in the console or cmd

It has 4 classes.

Class that contains the controller methods

FinalFileController.java

It has two REST endpoints

<http://localhost:8080/> This method allows the process of taking several html fragments (cloned with independent threads) and putting them together in a Result.html file using independent threads with a synchronized code block.

<http://localhost:8080/retrieveFile> This method allows you to call the endpoint http://localhost:8080/ several times, trying to complete the process correctly. For this, use the Backoff.java class which implements exponential backoff methods.

Class that implements exponential backoff methods.

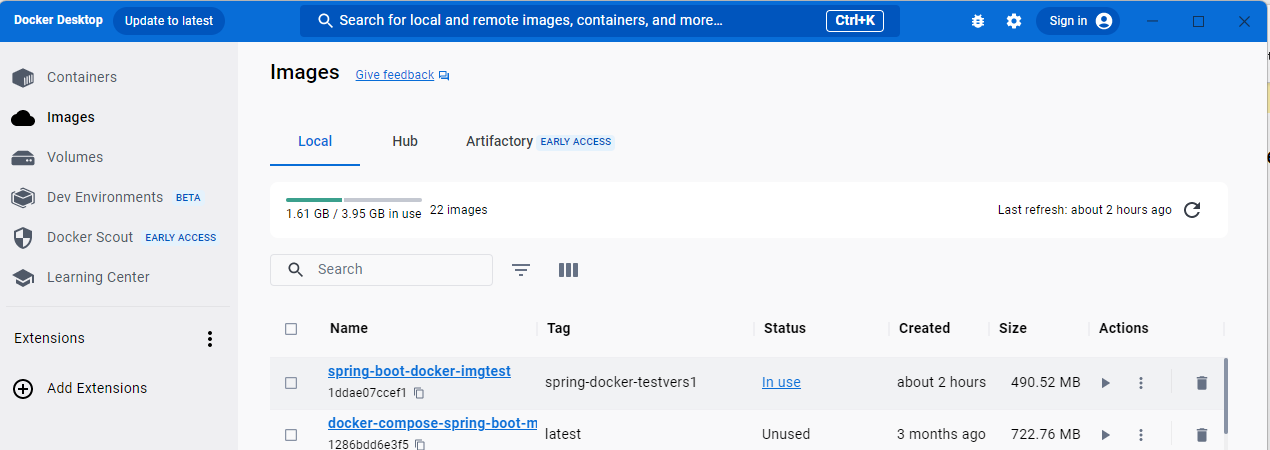
Backoff.java

Two kinds of Threads

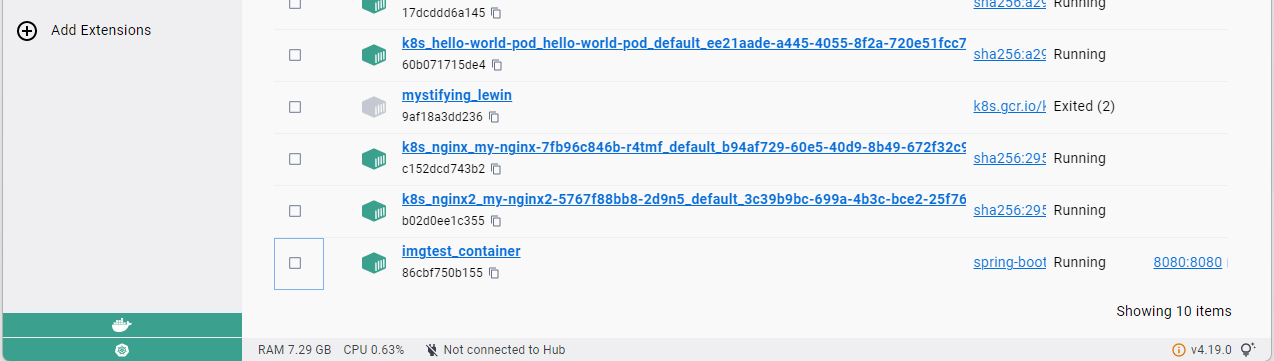
readThread.java Takes a fragment of html and saves it to a txt file.

finalFileThread.java Takes a txt file with a snippet of html code and inserts it into the Result.html file and does so in a synchronized block.

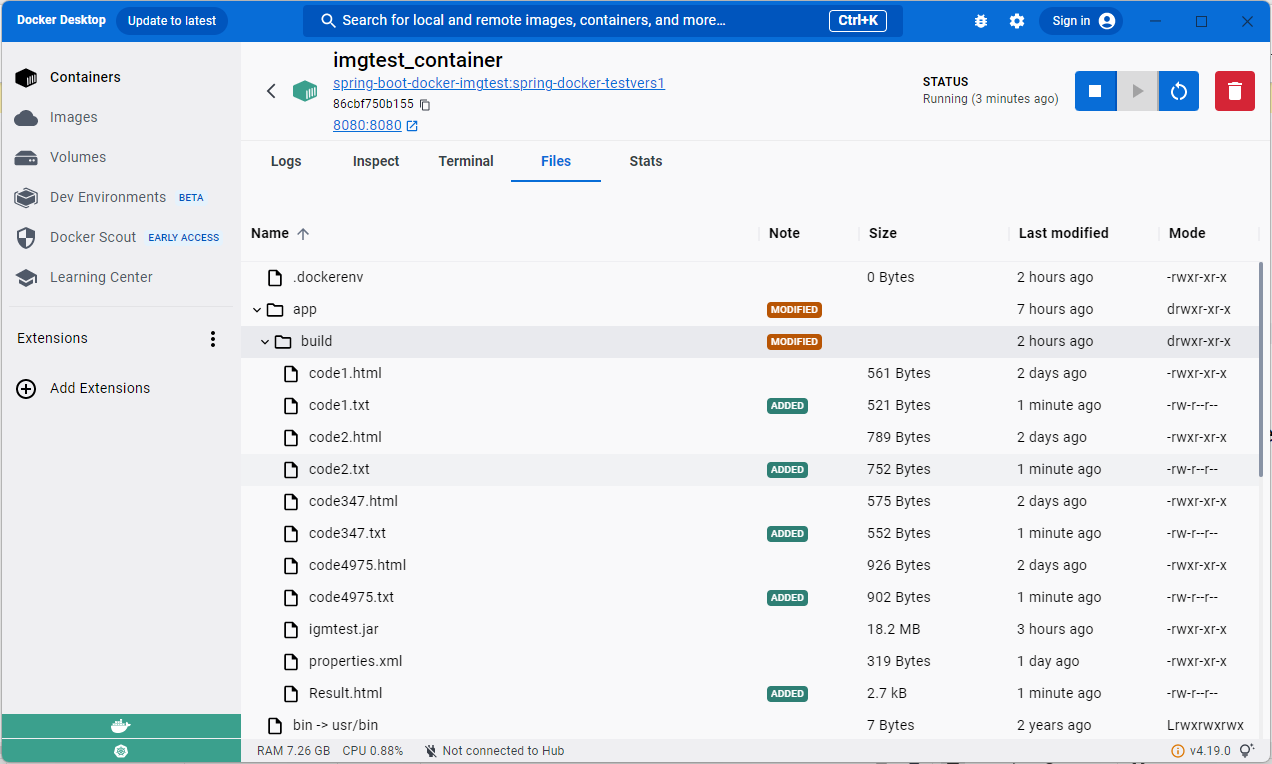
View with the spring-boot-docker-imgtest image on the Docker Desktop:



View with the imgtest\_container container on the Docker Desktop:



View with the html fragment files and the Result.html file when the process is successful in Docker Desktop:



View with the results of the exponential backoff mechanism in the console or cmd (There is a warning in the code, possible bug):

