Dockerfile difines the contents and start up (by entry point command) behavior of a single container.

Создаём image с помощью Dockerfile

1)-FROM alpine

2)-RUN apk update && apk add nano && apk add openjdk17

3)-ADD Hello.java Hello.java

4)-ENTRYPOINT javac Hello.java && java Hell

1)-Create our image using alpine as a base\_image

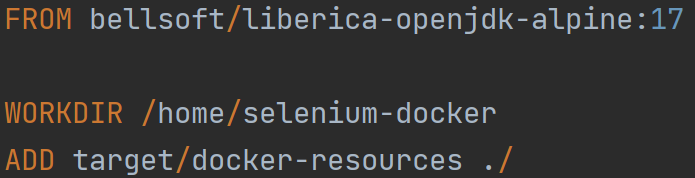
2)-The commands that we want to be executed by alpine. Whatever you want to run in terminal

3)-Add the file to the alpine (it is copying the files from host to container, we are doing it to be able to use them, since alpine originally does not have these files and we need them).

4)-It is what is to be done after all the settings are finished

**THE FILE SHOULD BE UPDATED, BUT** THERE IS INAF INFO IN INTERNET ON <HOW TO BUILD DOCKERFILE> USE INTERNET

When the dockerFIle is ready, for example:



WORKDIR - directory in a docker root container, where we would like to store docker-resources, that is mentioned in an ADD instruction.

**docker build -t some\_name\_of\_the\_prototip\_app** . ← command to build the image (you need to be in the directory where Docker file is located.

**-t** flag tags your image. Think of this as a human-readable name for the image. Since you named the image some\_name\_of\_the\_prototip\_app, you can refer to that image when you run a container. In another words it will create image from that Dockerfile with the name some\_name\_of\_the\_prototip\_app**.**

**.** ← dot represents the current directory where docker should look for the Dockerfile.