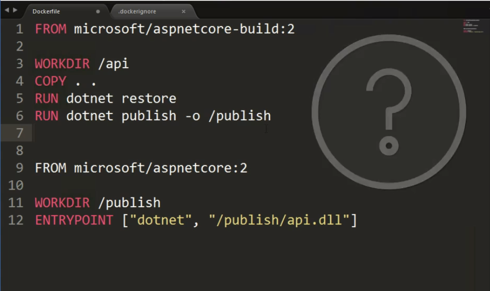
Building a multi-stage Dockerfile

Dockerfile can have multi-stage container creation and execution instructions where output of a container can be used in subsequent container being spinned-off.

## What is Multiple stages:

In the below example, same Dockerfile can use two different source images at different stages (each FROM instruction is a starting point for one stage with default index value of 0):



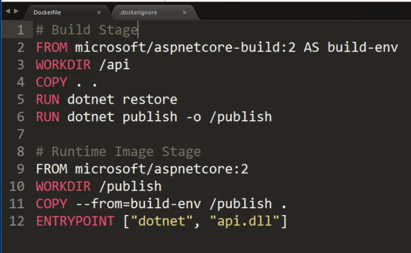
In the above example, line #1 marks stage 0 while line #9 marks the stage 1.

**Note:** In order to keep the published code from being bloated by debugging symbols, *in the above example, at line# 6, the publish command can be parameterized to pass a configuration of* ***Release*** *as following:*

*RUN dotnet publish –c Release –o /publish*

## Naming of Stages:

Each stage can be labelled with the usage of AS <stage name>.



In the above screen shot the stage 0 is labelled as *build-env.* Line number 11 shows how to copy content from previous stage to current stage container.

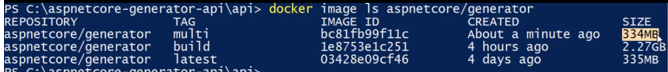
## Running the multi-stage docker:

On command line, run the following command by changing the tag of image:



Advantage of using multi-stage Dockerfile:

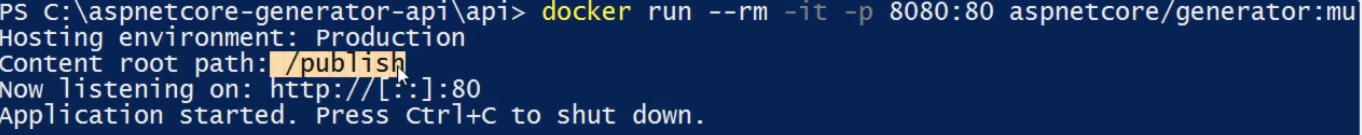
1. It allowed to have all the build tools included in docker image creation
2. Yet, the final container contains only runtime requisites, thus by significantly reducing the size of final image. See the comparison



*Build* image was the one that was built along with build tools and infrastructure (using only stage 0 base image of *microsoft:aspnetcore-build/2* which was huge in size.

On the other hand,The *multi* image uses the output of stage 0 and finally emits only the runtime requisite image (using *microsoft:aspnetcore* image as base image) and then using the publish folder from previous stage. This allows it to discard all the build tool dependencies in final image. This reduces the final image size from 2.27GB to 334MB.

Finally, run the container on port as desired:



Check the swagger, it should just come up on 8080 port of host machine:

