Optimizing a Docker build process

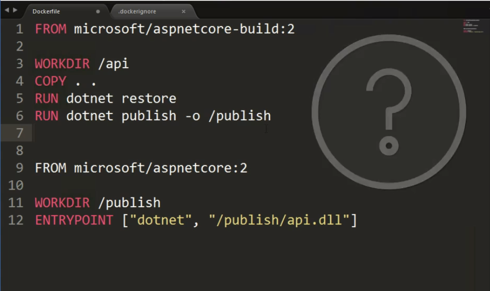
## Pre-requisite:

Refer to [this doc](file:///C:\Docs\core-Docker\Multi-staged%20Docker%20Build%20guideline.docx) to know how to create a multi-staged Dockerfile.

## Optimization:

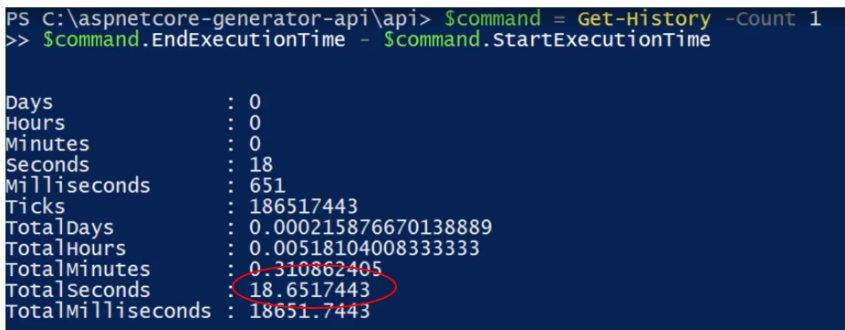
### Problem statement 1:

There is a change in source code but there is NO change in package dependencies at all. Still, image creation process skips the cache copy and executes the “*restore*” command all over again. That’s not necessary.



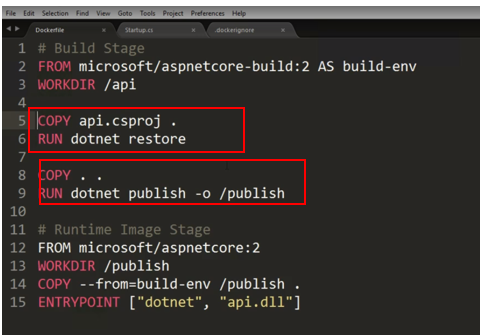
Even on re-run with caching enabled, after minor source code edit, the following image build command took 18 seconds





### Solution:

Let’s separate out the source code copy process from build process. Since, the dependency-restore is linked with changes in .csproj file NOT the source code edits, let’s re-organize the instructions appropriately. See the following DockerFile:



*RUN dotnet restore* is contingent upon COPY api.csproj .

In case you have multiple project files, COPY api.csproj can also be written as COPY \*.csproj

That means dotnet restore will use cache until there is a change in .csproj file.

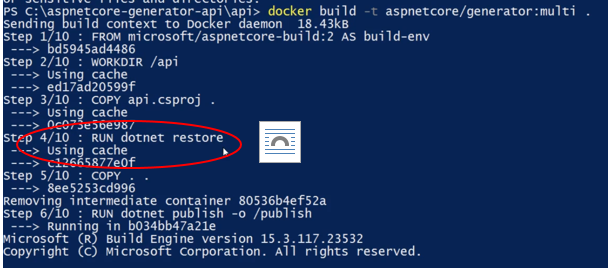
COPY instruction of rest of code is in following segment. That separates out the dependencies and does the optimization during image creation.

Optimization benefit:

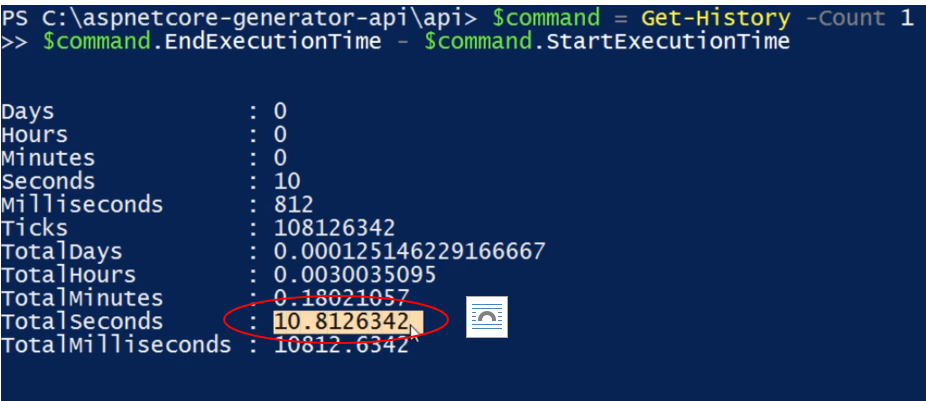
After editing the Dockerfile, when ran the same image build command (obviously after cache is created), it provided the optimization of almost 10 seconds:



See, the dotnet restore command uses cache



Run the next command to see how much it saved the time consumed to create the same image:



### Problem Statement 2:

Package dependency has actually changed but is limited to only one project in Solution and for that the image build process is *restoring* the dependencies for all the other projects that were not changed. Can we optimize the process to restore only for changed dependencies not for other projects?

### Solution:

The package restore can be kept limited to that particular project by separating out the build stages as following:

