Difference between RUN and CMD in a Dockerfile

Asked 5 years ago Active 1 year, 1 month ago Viewed 125k times

387

I'm confused about when should I use `CMD` vs `RUN`. For example, to execute bash/shell commands (i.e. `ls -la`) I would always use `CMD` or is there a situation where I would use `RUN`? Trying to understand the best practices about these two similar `Dockerfile` directives.

docker dockerfile

90

Share Edit Follow

edited Oct 14 '19 at 8:45
flow2k
2,649 24 40

asked May 26 '16 at 13:11
TakeSoUp
4,627 5 13 19

docs.microsoft.com/en-us/virtualization/windowscontainers/... – Manish Jain Mar 6 '20 at 19:09

9 Answers

Active | Oldest | Votes

566

RUN is an image build step, the state of the container after a `RUN` command will be committed to the container image. A Dockerfile can have many `RUN` steps that layer on top of one another to build the image.

CMD is the command the container executes by default when you launch the built image. A Dockerfile will only use the final `CMD` defined. The `CMD` can be overridden when starting a container with `docker run $image $other_command`.

ENTRYPOINT is also closely related to `CMD` and can modify the way a container starts an image.

Share Edit Follow

edited Apr 19 '20 at 11:15

answered May 26 '16 at 13:25
Matt

51.6k 6 118 123

24 you do all the `RUN` needed to setup your environment, and your (only) CMD launches the process running in your container, example, for nginx, extract from github.com/nginxinc/docker-nginx/blob/... you see the line `CMD ["nginx", "-g", "daemon off;"]` – user2915097 May 26 '16 at 14:00

3 "A Dockerfile can only have one CMD" - not technically true, but effectively all but one will be ignored. See the answer of GingerBeer. – Colm Bhandal Apr 19 '20 at 10:42

"A Dockerfile will only use the final CMD defined"? actually, the final CMD defined will be used in launching image as a container, right? – paul cheung May 29 '20 at 13:19

2 Yes @paulcheung the final command in the dockerfile is written to the image and is the command the container executes by default when you launch the built image. – Matt May 31 '20 at 23:35

"A Dockerfile will only use the final CMD defined." -- I just wasted the past hour because I did not realize this. Why on earth wouldn't they at least give you a warning if they are going to ignore these? – Pat Niemeyer Apr 28 at 4:57

172

`RUN` - command triggers while we build the docker image.

`CMD` - command triggers while we launch the created docker image.

Share Edit Follow

edited Dec 12 '19 at 15:34

tgogos
16.5k 14 77 108

answered Jul 17 '17 at 6:56

Nisal Edu
5,401 3 22 31

87

I found this article very helpful to understand the difference between them:

RUN - RUN instruction allows you to install your application and packages required for it. It executes any commands on top of the current image and creates a new layer by committing the results. Often you will find multiple RUN instructions in a Dockerfile.

CMD - CMD instruction allows you to set a default command, which will be executed only when you run container without specifying a command. If Docker container runs with a command, the default command will be ignored. If Dockerfile has more than one CMD instruction, all but last
CMD instructions are ignored.

Share Edit Follow

edited Sep 18 '17 at 4:50

rjdkolb
8,257 8 57 77

answered Jan 1 '17 at 13:14

fay
1,566 1 10 32

20

RUN - Install Python , your container now has python burnt into its image
CMD - python hello.py , run your favourite script

Share Edit Follow

answered Feb 20 '18 at 12:10

Rohit Salecha
496 5 6

CMD - Install Python, my container now doest not hast python burnt into its image? – Carlos Fontes Oct 23 '18 at 3:40

RUN will create an image layer of python , CMD will simply execute the command not create the image – Rohit Salecha Oct 25 '18 at 13:25

13

The existing answers cover most of what anyone looking at this question would need. So I'll just cover some niche areas for CMD and RUN.

CMD: Duplicates are Allowed but Wasteful

GingerBeer makes an important point: you won't get any errors if you put in more than one CMD - but it's wasteful to do so. I'd like to elaborate with an example:

```
FROM busybox
CMD echo "Executing CMD"
CMD echo "Executing CMD 2"
```

If you build this into an image and run a container in this image, then as GingerBeer states, only the last CMD will be heeded. So the output of that container will be:

```
Executing CMD 2
```

The way I think of it is that "CMD" is setting a single global variable for the entire image that is being built, so successive "CMD" statements simply overwrite any previous writes to that global variable, and in the final image that's built the last one to write wins. Since a Dockerfile executes in order from top to bottom, we know that the bottom-most CMD is the one gets this final "write" (metaphorically speaking).

RUN: Commands May not Execute if Images are Cached

A subtle point to notice about RUN is that it's treated as a pure function even if there are side-effects, and is thus cached. What this means is that if RUN had some side effects that don't change the resultant image, and that image has already been cached, the RUN won't be executed again and so the side effects won't happen on subsequent builds. For example, take this Dockerfile:

```
FROM busybox
RUN echo "Just echo while you work"
```

First time you run it, you'll get output such as this, with different alphanumeric IDs:

```
docker build -t example/run-echo .
Sending build context to Docker daemon  9.216kB
Step 1/2 : FROM busybox
 ---> be5888e67be6
Step 2/2 : RUN echo "Just echo while you work"
 ---> Running in ed37d558c505
Just echo while you work
Removing intermediate container ed37d558c505
 ---> 6f46f7a393d8
Successfully built 6f46f7a393d8
Successfully tagged example/run-echo:latest
```

Notice that the echo statement was executed in the above. Second time you run it, it uses the cache, and you won't see any echo in the output of the build:

```
docker build -t example/run-echo .
Sending build context to Docker daemon  9.216kB
Step 1/2 : FROM busybox
```

```
 ---> be5888e67be6
Step 2/2 : RUN echo "Just echo while you work"
 ---> Using cache
 ---> 6f46f7a393d8
Successfully built 6f46f7a393d8
Successfully tagged example/run-echo:latest
```

Share Edit Follow

edited Apr 19 '20 at 11:03

answered Apr 19 '20 at 10:45

Colm Bhandal
1,848 1 11 21

11

Note: Don't confuse RUN with CMD. RUN actually runs a command and commits the result; CMD does not execute anything at build time, but specifies the intended command for the image.

from docker file reference

https://docs.docker.com/engine/reference/builder/#cmd

Share Edit Follow

answered May 1 '17 at 9:59

Elsayed
1,908 2 19 34

10

RUN Command: RUN command will basically, execute the default command, when we are building the image. It also will commit the image changes for next step.

There can be more than 1 RUN command, to aid in process of building a new image.

CMD Command: CMD commands will just set the default command for the new container. This will not be executed at build time.

If a docker file has more than 1 CMD commands then all of them are ignored except the last one. As this command will not execute anything but just set the default command.

Share Edit Follow

answered Oct 29 '18 at 10:15

GingerBeer

703 8 10

8

RUN: Can be many, and it is used in **build** process, e.g. install multiple libraries

CMD: Can only have 1, which is your **execute** start point (e.g. `["npm", "start"]`, `["node", "app.js"]`)

Share Edit Follow

answered Jan 23 '20 at 23:05

Xin

22.9k 12 72 68

4

There has been enough answers on **RUN** and **CMD**. I just want to add a few words on **ENTRYPOINT**. **CMD** arguments can be overwritten by command line arguments, while **ENTRYPOINT** arguments are always used.

This article is a good source of information.

Share Edit Follow

answered Nov 21 '19 at 2:39

Milo Lu

2,603 1 26 39