Why can't I use Docker CMD multiple times to run multiple services?

Asked 10 years, 9 months ago Modified 4 months ago Viewed 226k times

129

I have built a base image from Dockerfile named centos+ssh. In centos+ssh's Dockerfile, I use CMD to run ssh service.

Then I want to build a image run other service named rabbitmq,the Dockerfile:

```
FROM centos+ssh
EXPOSE 22
EXPOSE 4149
CMD /opt/mq/sbin/rabbitmq-server start
```

To start rabbitmq container，run：

```
docker run -d -p 222:22 -p 4149:4149 rabbitmq
```

but ssh service doesn't work, it sense rabbitmq's Dockerfile CMD override centos's CMD.

1. How does CMD work inside docker image?
2. If I want to run multiple service, how to? Using supervisor?

docker **docker-cmd**

Share Improve this question Follow

edited Apr 17, 2024 at 17:22
Anil
2,554 ● 6 ● 33 ● 45

asked May 16, 2014 at 4:06

edwardsbean
3,779 ● 5 ● 23 ● 25

6 Answers

Sorted by: Highest score (default)

162

You are right, the second Dockerfile will overwrite the `CMD` command of the first one. Docker will always run a single command, not more. So at the end of your Dockerfile, you can specify *one* command to run. Not more.

But you can execute both commands in one line:

```
FROM centos+ssh
EXPOSE 22
EXPOSE 4149
CMD service sshd start && /opt/mq/sbin/rabbitmq-server start
```

What you could also do to make your Dockerfile a little bit cleaner, you could put your CMD commands to an extra file:

```
FROM centos+ssh
EXPOSE 22
EXPOSE 4149
CMD sh /home/centos/all_your_commands.sh
```

And a file like this:

```
service sshd start &
/opt/mq/sbin/rabbitmq-server start
```

Share Improve this answer Follow

answered May 16, 2014 at 6:21

Thomas Uhrig
31.6k ● 13 ● 63 ● 80

1 thanks, i think use supervisor is better.but why docker only run one CMD?what happen inside? – edwardsbean May 17, 2014 at 2:38

3 I do not know what is happening inside. But I think it is just designed like that. When you have an image and run a command in it (e.g. with CMD), it starts a container. The container runs as long as the command runs. And as soon as the command finishes, the container also stops. So each container represents one single (running) command. – Thomas Uhrig May 17, 2014 at 9:48

i think maybe it because of lcx or something's limit – edwardsbean May 21, 2014 at 15:54

2 @Tyguy7 .. because ..... ? – Michael M Feb 27, 2019 at 23:00

6 `&&` technique will work only with non-interactive services (that can start in background) else only the first one will run. – noraj Nov 11, 2019 at 17:03

73

Even though CMD is written down in the Dockerfile, it really is runtime information. Just like EXPOSE, but contrary to e.g. RUN and ADD. By this, I mean that you can override it later, in an extending Dockerfile, or simple in your run command, which is what you are experiencing. At all times, there can be only one CMD.

If you want to run multiple services, I indeed would use supervisor. You can make a supervisor configuration file for each service, ADD these in a directory, and run the supervisor with `supervisord -c /etc/supervisor` to point to a supervisor configuration file which loads all your services and looks like

```
[supervisord]
nodaemon=true

[include]
files = /etc/supervisor/conf.d/*.conf
```

If you would like more details, I wrote a blog on this subject here: http://blog.trifork.com/2014/03/11/using-supervisor-with-docker-to-manage-processes-supporting-image-inheritance/

Share Improve this answer Follow

answered May 18, 2014 at 11:17

qkrijger
27.2k ● 6 ● 38 ● 37

Thanks,supervisor is a good idea,but i'm wonder how does CMD work inside docker image – edwardsbean May 18, 2014 at 12:51

2 You asked two question, 2. on running multiple services. On wondering how CMD works, please elaborate on what you want to know specifically. I already mentioned it being runtime information and being overwritten by any new CMD. – qkrijger May 20, 2014 at 8:36

30

While I respect the answer from qkrijger explaining how you can work around this issue I think there is a lot more we can learn about what's going on here ...

To actually answer your question of "**why**" ... I think it would for helpful for you to understand how the `docker stop` command works and that *all processes should be shutdown cleanly* to prevent problems when you try to restart them (file corruption etc).

Problem: What if docker *did* start SSH from it's command *and* started RabbitMQ from your Docker file? "The docker stop command attempts to stop a running container first by sending a SIGTERM signal to the root process (PID 1) in the container." Which process is docker tracking as PID 1 that will get the SIGTERM? Will it be SSH or Rabbit?? "According to the Unix process model, the init process -- PID 1 -- inherits all orphaned child processes and must reap them. Most Docker containers do not have an init process that does this correctly, and as a result their containers become filled with zombie processes over time."

Answer: Docker simply takes that last CMD as the **one** that will get launched as **the root process** with PID 1 and get the SIGTERM from `docker stop`.

Suggested solution: You should use (or create) a base image specifically made for running more than one service, such as phusion/baseimage

It should be important to note that tini exists exactly for this reason, and as of Docker 1.13 and up, tini is officially part of Docker, which tells us that running more than one process in Docker **IS VALID** .. so even if someone claims to be more skilled regarding Docker, and insists that you

absurd for thinking of doing this, know that you are not. There are perfectly valid situations for doing so.

Good to know:

- https://blog.phusion.nl/2015/01/20/docker-and-the-pid-1-zombie-reaping-problem/
- http://www.techbar.me/stopping-docker-containers-gracefully/
- https://www.ctl.io/developers/blog/post/gracefully-stopping-docker-containers/
- https://github.com/phusion/baseimage-docker#docker_single_process

Share Improve this answer Follow edited Aug 28, 2018 at 19:21 answered Jan 22, 2016 at 19:54

Michael M
8,763 ● 2 ● 37 ● 54

6

The official docker answer to Run multiple services in a container.

It explains how you can do it with an init system (systemd, sysvinit, upstart) , a script (`CMD ./my_wrapper_script.sh`) or a supervisor like `supervisord`.

The `&&` workaround can work only for services that starts in background (daemons) or that will execute quickly without interaction and release the prompt. Doing this with an interactive service (that keeps the prompt) and only the first service will start.

Share Improve this answer Follow edited Nov 11, 2019 at 17:29 answered Nov 11, 2019 at 17:10

noraj
4,642 ● 1 ● 36 ● 41

"The following is a naive example" – Klaas van Schelven Aug 27, 2024 at 10:46

1

Building on Michael's explanation, I'd like to suggest an alternative approach using a tool I wrote called `monofy`. It's a Python-based solution designed to manage multiple processes within a single Docker container, ensuring they start, run, and stop together.

How It Works

- **Single Parent Process:** `monofy` runs as the parent, starting all your services as child processes. This allows for proper signal handling, forwarding signals like `SIGTERM` to all child processes for a clean shutdown.
- **Unified Logging:** It consolidates stdout and stderr from all child processes, simplifying monitoring.

- **Connected Fate:** If one process exits, `monofy` terminates all other processes, ensuring they stop together.

Example Implementation

Consider you have two services, `service1` and `service2`, tightly coupled within your application. Here's how you could modify your Dockerfile:

```
FROM python:3.12

RUN pip install monofy

CMD ["monofy", "service1", "|||", "service2"]
```

This setup runs both `service1` and `service2` in a single container with proper process management, logging, and shutdown handling—without extra tools like `supervisor` or `phusion/baseimage`.

Share Improve this answer Follow

edited Oct 7, 2024 at 21:02

answered Aug 29, 2024 at 14:05

Klaas van Schelven
2,528 ● 1 ● 25 ● 40

0

To address why CMD is designed to run only one service per container, let's just realize what would happen if the secondary servers run in the same container are not trivial / auxiliary but "major" (e.g. storage bundled with the frontend app). For starters, it would break down several important containerization features such as horizontal (auto-)scaling and rescheduling between nodes, both of which assume there is only one application (source of CPU load) per container. Then there is the issue of vulnerabilities - more servers exposed in a container means more frequent patching of CVEs...

So let's admit that it is a 'nudge' from Docker (and Kubernetes/Openshift) designers towards good practices and we should not reinvent workarounds (SSH is not necessary - we have `docker exec / kubectl exec / oc rsh` designed to replace it).

- More info

https://devops.stackexchange.com/questions/447/why-it-is-recommended-to-run-only-one-process-in-a-container

Share Improve this answer Follow

edited Jan 8, 2020 at 10:19

answered Jan 8, 2020 at 9:59

mirekphd
6,821 ● 3 ● 55 ● 79