RUN, CMD, ENTRYPOINT Directives

RUN

The RUN directive is executed in a new layer, what does it mean? It is used to install packages and applications on top of an existing image layer and create a new layer on top of it. Docker images are used to build new ones and the RUN directive allows you to add more software / security updates on top of the based image, giving you the ability to customize it.

You have to understand that Docker images work in layers. You have the base images that are provided by Docker such as Ubuntu, Linux, or Windows images. They by default don't have much software installed, that's where you can use RUN to install other software to customize it to your need! By running RUN it will add a new layer on top of the existing layer with the software you want to install.

```
RUN [“apt-get”, “install”, “vim”]
```

This RUN command basically installs `vim` on top of your existing image which might not have it by default.

There can be multiple RUN command in your Dockerfile because it is used to customize the image to your need.

CMD

The CMD directive is similar to ENTRYPOINT both are used to run a command after your container has started. (Okay I got this environment setup already, with all the code and dependencies, what do you actually want me to run?).

Using the CMD command you set a default command. The default command is run if you run the container without specifying some command. In the case where you do specify command then the CMD line is ignored because the command user specifies override the default command.

> There can only be one CMD instruction in a Dockerfile, if you have more than one only the last CMD take effect.

There are three ways of using CMD directive

1. CMD <command> parameter1, parameter2... (Shell form)
2. CMD ["executable command", "parameter1", "parameter2"] (Executable form)
3. CMD ["parameter1", "parameter2"]

The third way is used to set additional default parameter when you are using ENTRYPOINT in executable form.

ENTRYPOINT

Very similar to CMD also used to tell what command to run after your container has started. Only the last ENTRYPOINT will have an effect.

However, if you use ENTRYPOINT command then you cannot override the instruction by adding command-line parameter to the `docker run` command. If you use ENTRYPOINT command then you are implying the container is built for a specific use-case and the command should not be overriden.

```
FROM ubuntu
ENTRYPOINT ["echo", "Hello World"]
```

If you have this as your Dockerfile and running

```
docker build -t entrypoint-instructions
docker run entrypoint-instructions
```

This will simply print out "Hello World". But what happens if you add command line arguments after the image name?

`docker run entrypoint-instructions printenv`

It will print out "Hello World printenv". So command-line arguments are simply appended as additional parameters to the ENTRYPOINT command.

If you do want to override ENTRYPOINT you would use the `--entrypoint` flag and then provide in the command you want to execute instead.

Combining ENTRYPOINT & CMD

If you have both ENTRYPOINT and CMD in your Dockerfile then CMD's parameter will just be appended to ENTRYPOINT as additional parameter just like what it would happen if you provide additional command-line arguments to ENTRYPOINT.

```
FROM ubuntu
ENTRYPOINT ["echo", "Hello"]
CMD ["Ricky"]
```

Running this container without any argument will print out "Hello Ricky"

However, if you run the container with argument such as

`docker run entrypoint-cmd hehexd`

Because you provided command-line argument it will override the CMD result in printing "Hello hehexd"

> When using both instructions use it in exec form.

Revision #2
Created 2023-02-20 16:53:45 UTC by Tamarine
Updated 2023-06-23 14:12:53 UTC by Tamarine