**Docker Entrypoint vs CMD**

* CMD defines default commands and/or parameters for a container.
* CMD is an instruction that is best to use if you need a default command which users can easily override.
* If a Dockerfile has multiple CMDs, it only applies the instructions from the last one.
* ENTRYPOINT is preferred when you want to define a container with a specific executable.
* You cannot override an ENTRYPOINT when starting a container unless you add the --entrypoint flag.

Combine ENTRYPOINT with CMD if you need a container with a specified executable and a default parameter that can be modified easily. For example, when containerizing an application use ENTRYPOINT and CMD to set environment-specific variables.

**Shell and Exec Form**

Docker ENTRYPOINT and CMD can have two forms:

• Shell form

• Exec form

The syntax for any command in shell form is:

<instruction> <command>

The syntax for instructions in exec form is:

<instruction> [“executable”, “parameter”]

**Docker CMD/ENTRYPOINT instructions in both forms:**

• CMD echo “Hello World” (shell form)

• CMD ["echo", "Hello World"] (exec form)

• ENTRYPOINT echo "Hello World" (shell form)

• ENTRYPOINT ["echo", "Hello World"] (exec form)

**Docker CMD**

Docker CMD defines the default executable of a Docker image. You can run this image as the base of a container without adding command-line arguments. In that case, the container runs the process specified by the CMD command.

The CMD instruction is only utilized if there is no argument added to the run command when starting a container. Therefore, if you add an argument to the command, you override the CMD.

**Creating a Dockerfile with CMD and Building an Image**

**1. Start by creating a new project1 to store your images in:**

mkdir project1

**2. Move into that folder and create a new Dockerfile:**

cd project1

**3. Create a Docker file and add the following content to the file:**

**vi Dockerfile**

FROM ubuntu

MAINTAINER chaitanya

RUN apt-get update

CMD [“echo”, “Hello World”]

In the content above, you can see that we used the CMD instruction to echo the message Hello World when the container starts up without a specified command.

**4. Save and exit the file.**

**5. The next step is to build a Docker image from the newly made Dockerfile.**

docker build -t img1 .

**6. The output will tell you the name of the container. You can check to see whether it is available among the locally stored images by running:**

docker images

**7. Create a container**

**Running a Docker Container with CMD**

Run the container with the command:

docker run [image\_name]

docker run img1

Since there is no command-line argument, the container will run the default CMD instruction and display the Hello World message. However, if you add an argument when starting a container, it overrides the CMD instruction.

**For example, add the hostname argument to the docker run command:**

docker run [image\_name] hostname

Docker will run the container and the hostname command instead of the CMD’s echo command. You can see this in the output.

**Docker Entrypoint**

ENTRYPOINT is the other instruction used to configure how the container will run. Just like with CMD, you need to specify a command and parameters.

You cannot override the ENTRYPOINT instruction by adding command-line parameters to the docker run command. By opting for this instruction, you imply that the container is specifically built for such use.

**Creating a Dockerfile with ENTRYPOINT and Building an Image**

**1. Use the Dockerfile created in the CMD section and edit the file to change the instruction. Open the existing file with a text editor:**

sudo nano Dockerfile

**2. Edit the content by replacing the CMD command with ENTRYPOINT:**

FROM ubuntu

MAINTAINER chaitanya

RUN apt-get update

ENTRYPOINT [“echo”, “Hello World”]

**3. Save and close the file.**

**Running a Docker Container with ENTRYPOINT**

**1. Build a new image using the following command:**

docker build .

**2. The output should show you have successfully built the new image under a given name. Now let’s run it as a container without adding any command-line parameters:**

docker run [container\_name]

The output will be **“Hello World”** the same as with CMD. This is because we haven’t added any arguments to the run command.

**3. To see how ENTRYPOINT works, you need to add a parameter when starting a container. Use the same command as in the previous step and add something after the container name:**

docker run [container\_name] example

As you see, Docker did not override the initial instruction of echoing Hello World. It merely added the new parameter to the existing command.