LU02a - Unterschied ENTRYPOINT vs CMD

Ziele

1. Ich kann die Kommandos `ENTRYPOINT` und `CMD` beschreiben.
2. Ich kann die unterschiedliche Anwendung von `ENTRYPOINT` und `CMD` beschreiben.
3. Ich kann `ENTRYPOINT` und `CMD` in einem Docker-Datei gezielt anwenden.
4. Ich kann mindestens eine Massnahme bei Fehler im Build- oder Run-Prozess vorschlagen und umsetzen.

How to Use Docker EntryPoint

Introduction

`ENTRYPOINT` is one of the many instructions you can write in a dockerfile. The `ENTRYPOINT` instruction is used to configure the executables that will always run after the container is initiated. For example, you can mention a script to run as soon as the container is started.

Docker `ENTRYPOINT` instructions can be written in both `shell` and `exec` forms, such as the following example below:

- Shell form: `ENTRYPOINT node app.js`
- Exec form: `ENTRYPOINT [node, app.js]`

Steps we'll cover:

How does docker ENTRYPOINT work?

When you specify command line arguments to the docker run command through `exec form`, these arguments are appended to the end of all elements of the exec form, so the specified commands will run after the executables mentioned in entrypoint. For example, if you pass the '-d' argument to the 'docker run' command, the Docker will consider these arguments as the arguments of entrypoint and will execute them in the background. By doing this, you can turn your container into an executable. You can even add additional arguments to the entrypoint to convert a complex command line tool into a single argument docker container. The exec form just runs the binary you provide with the arguments you include but without any features of the shell parsing.

The `shell form` will execute the shell parameters as parameters to `/bin/sh -c`. That allows you to expand variables, piping output, subcommands, chaining commands together, and other shell features. You cannot use shell form to specify any command or the docker run command-line arguments while starting the container because the `ENTRYPOINT` command runs as a subcommand

of /bin/sh –c. The executable has a different process ID than the container's 'PID 1'. If we pass any Unix signals like SIGTERM from the docker stop command, the executable will receive it.

You can override the entrypoint instruction when initiating the container using the –entrypoint flag. Note that if you have mentioned multiple ENTRYPOINT instructions in the Dockerfile, then the last entrypoint will be executed. The recommended forms to use are:

```
RUN: shell form
ENTRYPOINT: exec form
CMD: exec form
```

Examples

The following example shows an entrypoint instruction that launches the funny tool cowsay. Note that the Mooo message of cowsay is hardcoded inside the dockerfile. Alternatives to that will be shown later.

```
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y cowsay \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*
ENTRYPOINT [/usr/games/cowsay , Mooo ]
```

The next step is to build the image from this docker file. Assume the name of the above docker file is Dockerfile. The image name can be specified with -t parameter. In this example, the image name provided is testentrypoint. The parameter directoryName is the folder having the default Dockerfile.

```
docker build  <directoryName>  –t testentrypoint
```

During building the container image, the docker daemon will look for the entrypoint instruction and will specify it as a default program that will always run whether the command line input is added or not upon container start. Now that the image has been created, the next step is to run the container. Just mention the image name in the docker run command.

```
docker run testentrypoint
```

Difference between ''ENTRYPOINT'' and ''CMD''

Before we discuss the differences, let's look at some of the similarities between these two:

- Both the commands are executed during docker run.
- Both CMD and ENTRYPOINT instructions define the commands which will be executed upon running a container
- You can override both commands by passing an argument during docker run.
- If multiple declarations are made, only the last CMD and ENTRYPOINT will be valid.

Both commands are different in the following aspects:

- CMD command is ignored by Docker daemon if you specify parameters within the docker run command. ENTRYPOINT, on the other hand, is not ignored; instead, it is appended as command line parameters by considering those as arguments of the command.
- ENTRYPOINT specifies the executable to be invoked when the container is started. Whereas CMD specifies the arguments that are passed to the ENTRYPOINT (for arguments).

Examples

Let's extend the above entrypoint example to CMD. The below example will work the same way as the above example of entrypoint, however, this time using CMD.

```
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y cowsay \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*
CMD [/usr/games/cowsay,Mooo]
```

However, if you run

```
docker run testentrypoint /usr/games/cowsay Mooo
```

then the whole CMD command inside the dockerfile will be ignored, and the command arguments passed through the docker run command will be given preference.

That is not the case with entrypoint instruction. Whatever you provide through entrypoint actually appends to the existing entrypoint instruction in the dockerfile. Let's look at the example below:

```
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y cowsay \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*
ENTRYPOINT [/usr/games/cowsay]
```

Now you can run

```
docker run testentrypoint Mooo
```

to achieve the same results because the arguments to docker run will be appended to the instruction present in dockerfile and will not be overridden.

We can combine both CMD and entrypoint in the same image too. Look at the example below:

```
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y cowsay \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*
ENTRYPOINT [/usr/games/cowsay]
```

```
CMD [Mooo]
```

Note that when ENTRYPOINT and CMD are both used in the same Dockerfile, everything specified in the CMD will be appended to the ENTRYPOINT as an argument.

When to use docker ENTRYPOINT vs CMD

ENTRYPOINT instructions can be used for both single-purpose and multi-mode docker images where you want a specific command to run upon the container start. You can also use it to build wrapper container images that encapsulate legacy programs for containerization, ensuring that the program will always run.

The optimal use case for CMD instruction is to specify default programs that should run if you do not provide any input arguments in the command line.

To conclude:

- Use ENTRYPOINT instructions when creating an executable Docker image with commands that must always be executed.
- Use CMD instructions for an additional set of arguments that will serve as default instructions in case there is an explicit command line argument provided when the container runs.