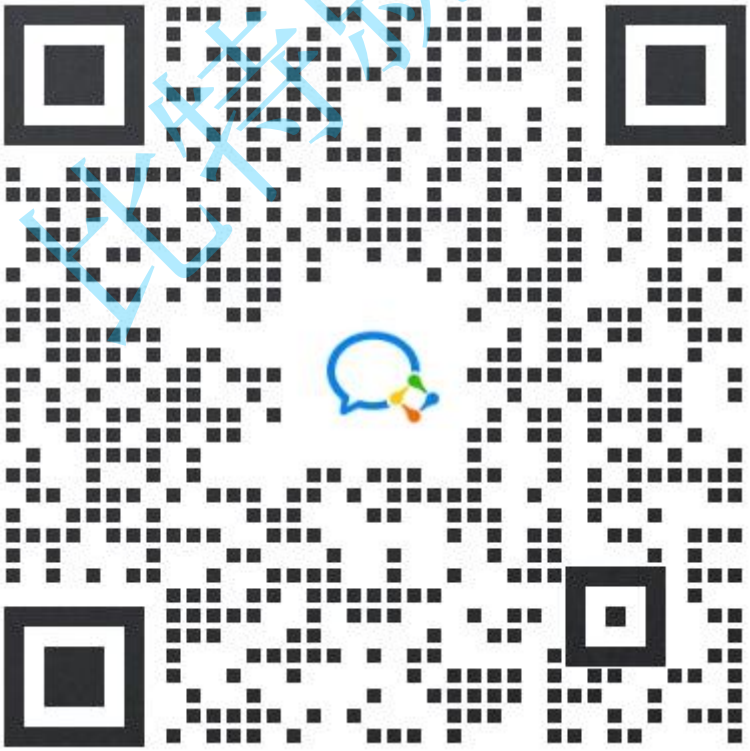
CMD 与 EntryPoint 实战

版权说明

对比特项目感兴趣，可以联系这个微信。



基础知识

ENTRYPOINT 和 CMD 都是在 docker image 里执行一条命令, 但是他们有一些微妙的区别.一般来说两个大部分功能是相似的都能满足。

比如执行运行一个没有调用 ENTRYPOINT 或者 CMD 的 docker 镜像, 返回错误，一般的镜像最后都提供了 CMD 或者 EntryPoint 作为入口。

覆盖

在写 Dockerfile 时, ENTRYPOINT 或者 CMD 命令会自动覆盖之前的 ENTRYPOINT 或者 CMD 命令.

在 docker 镜像运行时, 用户也可以在命令指定具体命令, 覆盖在 Dockerfile 里的命令.

如果你希望你的 docker 镜像只执行一个具体程序, 不希望用户在执行 docker run 的时候随意覆盖默认程序. 建议用 ENTRYPOINT.

Shell 和 Exec 模式

ENTRYPOINT 和 CMD 指令支持 2 种不同的写法: shell 表示法和 exec 表示法.

CMD 命令语法

```
Shell
# EXEC 语法
CMD ["executable","param1","param2"] (exec form, this is the
preferred form)
#用于给 ENTRYPOINT 传入参数，推荐使用
CMD ["param1","param2"] (as default parameters to ENTRYPOINT)
#shell 语法
CMD command param1 param2 (shell form)
```

ENTRYPOINT 语法

```
Shell
# EXEC 语法
ENTRYPOINT ["executable", "param1", "param2"] (exec form,
preferred)
# Shell 语法
ENTRYPOINT command param1 param2 (shell form)
```

当使用 shell 表示法时, 命令行程序作为 sh 程序的子程序运行, docker 用/bin/sh -c 的语法调用. 如果我们用 docker ps 命令查看运行的 docker, 就可以看出实际运行的是 /bin/sh -c 命令。这样运行的结果就是我们启动的程序的 PID 不是 1，如果从外部发送任何 POSIX 信号到 docker 容器, 由于/bin/sh 命令不会转发消息给实际运行的命令, 则

不能安全得关闭 docker 容器。

EXEC 语法没有启动/bin/sh 命令, 而是直接运行提供的命令, 命令的 PID 是 1. 无论你用的是 ENTRYPOINT 还是 CMD 命令, 都强烈建议采用 exec 表示法。

组合模式

组合使用 ENTRYPOINT 和 CMD, ENTRYPOINT 指定默认的运行命令, CMD 指定默认的运行参数.ENTRYPOINT 和 CMD 同时存在时, docker 把 CMD 的命令拼接到 ENTRYPOINT 命令之后, 拼接后的命令才是最终执行的命令.

实战目的

了解 CMD 和 EntryPoint 的本质，能够区别两种，并了解官方的推荐用法。

实战步骤

覆盖

多次覆盖

1. 我们创建一个 Dockerfile，指定多个 CMD，创建目录如下

```
Shell
mkdir -p /data/myworkdir/dockerfile/cmd
```

创建 Dockerfile1

```
Shell
FROM busybox
CMD echo "hello world"
CMD echo "hello bit"
```

2. 然后编译镜像，运行查看结果，可以看到第一个 CMD 被覆盖了

```
Shell
docker build -t cmd1:v0.1 -f Dockerfile1 .

root@139-159-150-152:/data/myworkdir/dockerfile/cmd# docker run
cmd1:v0.1
hello bit
```

3. 我们创建一个 Dockerfile2，指定多个 EntryPoint 然后运行查看结果

```
Shell
FROM busybox
ENTRYPOINT echo "hello world"
ENTRYPOINT echo "hello bit"
```

4. 编译构建，运行查看结果

```
Shell
docker build -t cmd1:v0.2 -f Dockerfile2 .
root@139-159-150-152:/data/myworkdir/dockerfile/cmd# docker run
cmd1:v0.2
hello bit
```

参数覆盖

1. 我们通过指定后面的启动参数，可以覆盖 CMD 的指令，但是 EntryPoint 的无法覆盖，执行效果如下

```
Shell
root@139-159-150-152:/data/myworkdir/dockerfile/cmd# docker run
cmd1:v0.1 echo hello bit2
hello bit2
root@139-159-150-152:/data/myworkdir/dockerfile/cmd# docker run
cmd1:v0.2 echo hello bit2
hello bit
```

2. 如果我们指定参数--entrypoint 就可以完成对 entrypoint 的替换

```
Shell
root@139-159-150-152:/data/myworkdir/dockerfile/cmd# docker run --
entrypoint '/bin/sh' cmd1:v0.2  -c "echo hello bit2"
hello bit2
```

Shell VS Exec

1. 我们编写下面的 Dockerfile3，执行 ping 命令

```
Shell
FROM ubuntu:22.04
RUN apt-get update -y && apt install -y iputils-ping
CMD ping localhost
```

2. 然后编译镜像，运行

```
Shell
docker build -t cmd1:v0.3  -f Dockerfile3 .
docker run --name shell1 -d  cmd1:v0.3
```

3. 进入镜像里面查看，可以看到 PID 为 1 的其实是/bin/sh

```
Shell
root@139-159-150-152:/data/myworkdir/dockerfile/cmd# docker exec -
it shell1 bash
root@3f0c00e8dbb4:/# ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 13:41 ?        00:00:00 /bin/sh -c
ping localhost
root           6       1  0 13:41 ?        00:00:00 ping localhost
root           7       0  1 13:41 pts/0    00:00:00 bash
root          14       7  0 13:41 pts/0    00:00:00 ps -ef
```

4. 我们新建 Dockerfile4,这个里面我们采用 EXEC 模式

```
Shell
FROM ubuntu:22.04
RUN apt-get update -y && apt install -y iputils-ping
CMD ["ping", "localhost"]
```

5. 我们编译镜像，然后运行镜像

```
Shell
 docker build -t cmd1:v0.4 -f Dockerfile4 .
 docker run --name shell2 -d  cmd1:v0.4
```

6. 我们进入镜像查看，可以看到 PID 为 1 的进程是 ping 而不再是我们的/bin/sh

```
Shell
root@139-159-150-152:/data/myworkdir/dockerfile/cmd# docker exec -
it shell2 bash
root@c7d04f6cd184:/# ps -ef
```

```
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 13:46 ?        00:00:00 ping localhost
root           6       0  0 13:46 pts/0    00:00:00 bash
root          13       6  0 13:47 pts/0    00:00:00 ps -ef
root@c7d04f6cd184:/#
```

7. EntryPoint 也是一样大家可以尝试下。无论你用的是 ENTRYPOINT 还是 CMD 命令, 都强烈建议采用 exec 表示法,

组合

1. 我们新建 Dockerfile5,同时设置 ENTRYPOINT 和 CMD

```
Shell
FROM ubuntu:22.04
RUN apt-get update -y && apt install -y iputils-ping
ENTRYPOINT ["/bin/ping","-c","3"]
CMD ["localhost"]
```

2. 此时我们编译镜像然后启动运行看下效果是什么，可以看到 CMD 的内容作为 ENTRYPOINT 的参数添加到了后面

```
Shell
docker build -t cmd1:v0.5 -f Dockerfile5 .
root@139-159-150-152:/data/myworkdir/dockerfile/cmd# docker run --
name shell3  --rm cmd1:v0.5
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.113
ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.032
ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.032
ms
```

3. 因为 CMD 的内容可以被替换，如果我们运行的时候替换成另外的一个网站，我们可以看到他会 ping 的是另外一个网站，也就是说 CMD 的参数被替换掉了。

```
Shell
root@139-159-150-152:/data/myworkdir/dockerfile/cmd# docker run --
name shell3  --rm cmd1:v0.5 www.baidu.com
PING www.a.shifen.com (14.119.104.189) 56(84) bytes of data.
```

```
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=1 ttl=53
time=4.67 ms
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=2 ttl=53
time=4.54 ms
64 bytes from 14.119.104.189 (14.119.104.189): icmp_seq=3 ttl=53
time=4.55 ms

--- www.a.shifen.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 4.536/4.585/4.668/0.059 ms
```