容器生命周期管...

这是本专栏的第二部分：容器篇，共 6 篇，帮助大家由浅入深地认识和掌握容器。这一篇和下一篇，我会为你介绍容器生命周期管理相关的内容，带你掌握容器生命周期。下面我们一起进入第一篇的内容，主要涉及容器状态的变化。

在第一部分第二篇《Docker 的基本使用》中，我为你介绍了 Docker 的基本使用，包括使用 `docker run` 启动一个容器，使用 `docker exec` 进入一个正在运行的容器。那你有没有考虑过容器的整个生命周期是什么样的？当我们不使用它的时候，它是什么状态？

我们启动一个 Redis 的容器作为示例。

复制

```
(MoeLove) ➜  ~ docker run -d redis
cf56a27e94f6142e7f69eb34de837b9bc091986006b1a7b563750317185376ea
```

使用 `docker ps` 命令查看容器当前的状态，在输出结果中包含一列 STATUS 状态信息，我们以此状态作为本篇的切入点。

复制

```
(MoeLove) ➜  ~ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
cf56a27e94f6        redis               "docker-entrypoint.s…"   5 seconds ago
```

创建一个容器（Created）

使用 `docker container create` 或者 `docker create` 命令可创建一个处于 Created 状态的容器，但是该容器并没有真正运行，我们是不能通过 `docker exec` 进入该容器的。

复制

```
# 创建一个容器
(MoeLove) ➜  ~ docker container create redis
2017ecbb4da676069ce0f57583765a7f2562475bb0f7dfab12de8d1e0a93f322
# 查询容器的状态
(MoeLove) ➜  ~ docker ps -l
CONTAINER ID        IMAGE               COMMAND                  CREATED
2017ecbb4da6        redis               "docker-entrypoint.s…"   11 seconds ago
# 尝试进入容器内部
(MoeLove) ➜  ~ docker exec -it 2017ecbb4da6 sh
Error response from daemon: Container 2017ecbb4da676069ce0f57583765a7f2562475bb0f
```

虽然它没有真正运行，但是它运行所需的相关文件却已经都创建好了，并且你也可以在 create 的时候给它传递各种可用于配置容器运行时的参数；这种方式通常是用来预配置容器，因为在执行完 `docker create` 命令后，会直接输出容器的 ID。你可以在一切准备就绪后，随时启动它。

另一种使用场景是，你可以用它来预挂载一个或多个存储卷，后续其他其他容器可借助此容器直接使用这些存储卷（在存储篇会详细介绍）。

运行一个容器（Running）

前面已经提到了我们可以使用 `docker create` 命令创建一个处于 Created 状态的容器，那我们如何能让它真正地开始运行呢？

复制

```
# 创建容器
(MoeLove) ➜  ~ docker create redis
b41a77be2fd9624824cc5b00b786f0236eba8d952f79b665a500fb7e649bb813
# 查询容器的状态
(MoeLove) ➜  ~ docker ps -l
CONTAINER ID        IMAGE               COMMAND                  CREATED
b41a77be2fd9        redis               "docker-entrypoint.s…"   2 seconds ago
```

我们可以使用 `docker start` 命令，让刚才 Created 状态的容器真正启动（运行）。

复制

```
# 启动该容器
(MoeLove) ➜  ~ docker start b41a77be2fd9
b41a77be2fd9
# 查询容器的状态
(MoeLove) ➜  ~ docker ps -l
CONTAINER ID        IMAGE               COMMAND                  CREATED
b41a77be2fd9        redis               "docker-entrypoint.s…"   33 seconds ago
# 进入容器执行命令
(MoeLove) ➜  ~ docker exec b41a77be2fd9 redis-cli ping
PONG
```

可以看到容器此时已经处于正在运行的状态了。那我们是否还有其他办法可以运行一个容器？

想必聪明的你已经想到了，我们也可以使用 `docker run` 命令直接运行一个容器。

复制

```
# 在后台运行一个容器
(MoeLove) ➜  ~ docker run -d redis
b566506f243bf22275397d5d73194e72a94775a0c5673240f60ac30539c276e5
# 查询容器状态
(MoeLove) ➜  ~ docker ps -l
CONTAINER ID        IMAGE               COMMAND                  CREATED
b566506f243b        redis               "docker-entrypoint.s…"   4 seconds ago
```

`docker run` 是运行容器最常用的一个命令，因为我们很多时候通过传入参数的方式便可完成大多数预期的配置了，并且使用 `docker run` 也更加直接一些。

你可能也发现了 `docker run -d` 和 `docker create` 在命令执行后，都会将容器 ID 输出，我们可以利用此 ID 完成后续对容器操作的需求。

暂停一个容器（Paused）

pause

容器在运行时，是否有办法让其暂停？或者让它暂时不提供服务？

答案是使用 `docker pause`，我们先来个实际例子再逐步进行解释。

复制

```
# 在后台运行一个容器
(MoeLove) ➜  ~ docker run -d redis
074322e2467c1f280d8f1dfe90be45a1fa0673ec7f97220ec2c695e812179e30
# 查看容器状态
(MoeLove) ➜  ~ docker ps -l
CONTAINER ID        IMAGE               COMMAND                  CREATED
074322e2467c        redis               "docker-entrypoint.s…"   3 seconds ago
# 暂停容器
(MoeLove) ➜  ~ docker pause 074322e2467c
074322e2467c
# 查看容器状态
(MoeLove) ➜  ~ docker ps -l
CONTAINER ID        IMAGE               COMMAND                  CREATED
074322e2467c        redis               "docker-entrypoint.s…"   19 seconds ago
```

可以看到容器已经处于 Paused 状态，但请同时**注意 Running （或者说 Up）与 Paused 并不互斥**，或者说 Paused 是 Running 的一个分支。

复制

```
# 尝试进入容器执行命令
(MoeLove) ➜  ~ docker exec 074322e2467c redis-cli ping
Error response from daemon: Container 074322e2467c is paused, unpause the contain
```

可以看到处于 Paused 状态的容器是无法 `docker exec` 进入其中的。

我们可以查看一下当前容器进程的状态：

复制

```
# 使用 docker inspect 可获取容器进程的 ID
(MoeLove) ➜  ~ docker inspect --format "{{.State.Pid}}" 074322e2467c
9599
# 使用 ps 命令查看进程的状态
(MoeLove) ➜  ~ ps -eo pid,state,command |grep 9599 |grep -v grep
 9599 D redis-server *:6379
```

可以看到当前进程状态是 D，表示进程确实是在休眠状态了。

unpause

那我们是否有办法让它恢复正常呢？

当然有，使用 `docker unpause` 即可。

复制

```
# 将容器从 Paused 状态恢复
(MoeLove) ➜  ~ docker unpause 074322e2467c
074322e2467c
# 进入容器内执行命令
(MoeLove) ➜  ~ docker exec 074322e2467c redis-cli ping
PONG
# 使用 ps 命令查看进程的状态
(MoeLove) ➜  ~ ps -eo pid,state,command |grep 9599 |grep -v grep
 9599 S redis-server *:6379
```

可以看到已经恢复正常。

停止一个容器（Exited）

如果想要将一个运行中的容器停止该如何操作？

使用 `docker stop` 操作即可。我们来看个具体例子：

复制

```
# 启动一个容器
(MoeLove) ➜  ~ docker run -d redis
fffb13b75e22ca4eb035721df1660d25b9ef8379b8fd9a90c4a0258ea54c560c
# 查询容器状态
(MoeLove) ➜  ~ docker ps -l
CONTAINER ID        IMAGE               COMMAND                  CREATED
fffb13b75e22        redis               "docker-entrypoint.s…"   2 seconds ago

# 停止一个容器
# docker ps -ql 用于获取最近一个容器的 ID
(MoeLove) ➜  ~ docker stop $(docker ps -ql)
fffb13b75e22

# 查询容器状态
(MoeLove) ➜  ~ docker ps -l
CONTAINER ID        IMAGE               COMMAND                  CREATED
fffb13b75e22        redis               "docker-entrypoint.s…"   17 seconds ago
```

可以看到使用 `docker stop` 可以将一个容器从 Running 状态变为 Exited 状态。一般情况下处于 Exited 状态的容器仍存在于机器上。

启动容器

将容器从 Exited 状态到 Running，想必你已经猜到了，使用 `docker start` 即可。

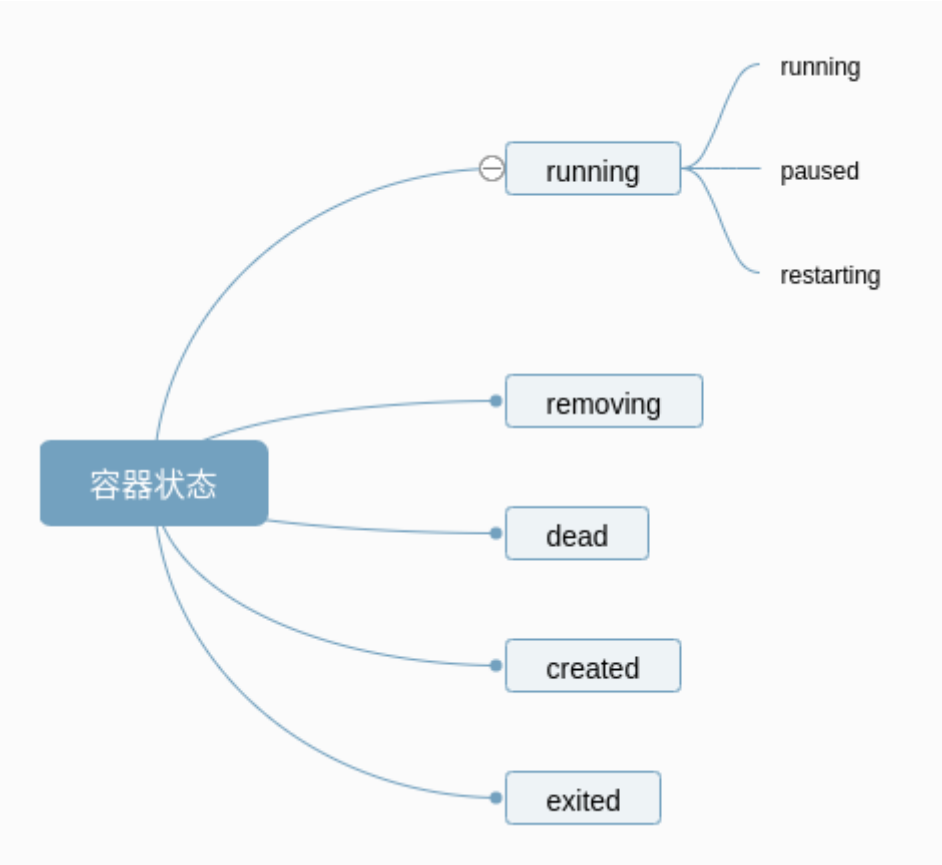
复制

```
# 启动一个容器
(MoeLove) ➜  ~ docker start $(docker ps -ql)
fffb13b75e22
# 查询容器状态
(MoeLove) ➜  ~ docker ps -l
CONTAINER ID        IMAGE               COMMAND                  CREATED
fffb13b75e22        redis               "docker-entrypoint.s…"   36 seconds ago
```

总结

本篇介绍了容器生命周期的基本管理操作，通过实际的实验来加深印象。

这里我总结了一张容器状态的概览图，供你参考。



其中 Dead 和 Removing 状态本篇未涉及，我将在下篇与你分享容器生命周期管理的剩余部分。

本篇的内容相对简单，但需要非常熟悉，建议多实践，会更利于后续专栏内容的学习。