title: Docker(四)容器 tags: Docker categories: 技术

Docker 容器

镜像（Image）和容器（Container）的关系，就像是面向对象程序设计中的类和实例一样，镜像是静态的定义，容器是镜像运行时的实体。容器可以被创建、启动、停止、删除、暂停等。

容器的实质是进程，但与直接在宿主执行的进程不同，容器进程运行于属于自己的独立的 命名空间。因此容器可以拥有自己的 root 文件系统、自己的网络配置、自己的进程空间，甚至自己的用户 ID 空间。容器内的进程是运行在一个隔离的环境里，使用起来，就好像是在一个独立于宿主的系统下操作一样。这种特性使得容器封装的应用比直接在宿主运行更加安全。也因为这种隔离的特性，很多人初学 Docker 时常常会把容器和虚拟机搞混。

镜像使用的是分层存储，容器也是如此。每一个容器运行时，是以镜像为基础层，在其上创建一个当前容器的存储层，我们可以称这个为容器运行时读写而准备的存储层为**容器存储层**。

容器存储层的生存周期和容器一样，容器消亡时，容器存储层也随之消亡。因此，任何保存于容器存储层的信息都会随容器删除而丢失。

按照 Docker 最佳实践的要求，容器不应该向其存储层内写入任何数据，容器存储层要保持无状态化。所有的文件写入操作，都应该使用 数据卷（Volume）、或者绑定宿主目录，在这些位置的读写会跳过容器存储层，直接对宿主(或网络存储)发生读写，其性能和稳定性更高。

数据卷的生存周期独立于容器，容器消亡，数据卷不会消亡。因此，使用数据卷后，容器可以随意删除、重新 run，数据却不会丢失。

操作 Docker 容器

容器是 Docker 又一核心概念。

简单的说，容器是独立运行的一个或一组应用，以及它们的运行态环境。对应的，虚拟机可以理解为模拟运行的一整套操作系统（提供了运行态环境和其他系统环境）和跑在上面的应用。

启动容器

启动容器有两种方式，一种是基于镜像新建一个容器并启动，另外一个是将在终止状态（stopped）的容器重新启动。

因为 Docker 的容器实在太轻量级了，很多时候用户都是随时删除和新创建容器。

新建并启动

所需要的命令主要为 docker run。

例如，下面的命令输出一个 “Hello World”，之后终止容器。

```
$ sudo docker run ubuntu:14.04 /bin/echo 'Hello world'
Hello world
```

这跟在本地直接执行 /bin/echo 'hello world' 几乎感觉不出任何区别。

下面的命令则启动一个 bash 终端，允许用户进行交互。

```
$ sudo docker run -t -i ubuntu:14.04 /bin/bash
root@af8bae53bdd3:/#
```

其中，-t 选项让Docker分配一个伪终端（pseudo-tty）并绑定到容器的标准输入上， -i 则让容器的标准输入保持打开。

在交互模式下，用户可以通过所创建的终端来输入命令，例如

```
root@af8bae53bdd3:/# pwd
/
root@af8bae53bdd3:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp
usr var
```

当利用 docker run 来创建容器时，Docker 在后台运行的标准操作包括：

- 检查本地是否存在指定的镜像，不存在就从公有仓库下载
- 利用镜像创建并启动一个容器
- 分配一个文件系统，并在只读的镜像层外面挂载一层可读写层
- 从宿主主机配置的网桥接口中桥接一个虚拟接口到容器中去
- 从地址池配置一个 ip 地址给容器
- 执行用户指定的应用程序
- 执行完毕后容器被终止

启动已终止容器

可以利用 docker start 命令，直接将一个已经终止的容器启动运行。

容器的核心为所执行的应用程序，所需要的资源都是应用程序运行所必需的。除此之外，并没有其它的资源。可以在伪终端中利用 ps 或 top 来查看进程信息。

```
root@ba267838cc1b:/# ps
  PID TTY          TIME CMD
    1 ?        00:00:00 bash
   11 ?        00:00:00 ps
```

可见，容器中仅运行了指定的 bash 应用。这种特点使得 Docker 对资源的利用率极高，是货真价实的轻量级虚拟化。

后台(background)运行

更多的时候，需要让 Docker在后台运行而不是直接把执行命令的结果输出在当前宿主机下。此时，可以通过添加 -d 参数来实现。

下面举两个例子来说明一下。

如果不使用 -d 参数运行容器。

```
$ sudo docker run ubuntu:14.04 /bin/sh -c "while true; do echo hello world;
sleep 1; done"
hello world
hello world
hello world
hello world
```

容器会把输出的结果(STDOUT)打印到宿主机上面

如果使用了 -d 参数运行容器。

```
$ sudo docker run -d ubuntu:14.04 /bin/sh -c "while true; do echo hello world;
sleep 1; done"
77b2dc01fe0f3f1265df143181e7b9af5e05279a884f4776ee75350ea9d8017a
```

此时容器会在后台运行并不会把输出的结果(STDOUT)打印到宿主机上面(输出结果可以用docker logs 查看)。

注： 容器是否会长久运行，是和docker run指定的命令有关，和 -d 参数无关。

使用 -d 参数启动后会返回一个唯一的 id，也可以通过 docker ps 命令来查看容器信息。

```
$ sudo docker ps
CONTAINER ID  IMAGE         COMMAND               CREATED        STATUS
PORTS NAMES
77b2dc01fe0f  ubuntu:14.04  /bin/sh -c 'while tr  2 minutes ago  Up 1 minute
agitated_wright
```

要获取容器的输出信息，可以通过 docker logs 命令。

```
$ sudo docker logs [container ID or NAMES]
hello world
hello world
hello world
. . .
```

终止容器

可以使用 docker stop 来终止一个运行中的容器。

此外，当Docker容器中指定的应用终结时，容器也自动终止。 例如对于上一章节中只启动了一个终端的容器，用户通过 exit 命令或 Ctrl+d 来退出终端时，所创建的容器立刻终止。

终止状态的容器可以用 docker ps -a 命令看到。例如

```
sudo docker ps -a
CONTAINER ID        IMAGE                     COMMAND                CREATED
STATUS                           PORTS               NAMES
ba267838cc1b        ubuntu:14.04              "/bin/bash"            30 minutes
ago      Exited (0) About a minute ago                       trusting_newton
98e5efa7d997        training/webapp:latest    "python app.py"        About an
hour ago    Exited (0) 34 minutes ago
backstabbing_pike
```

处于终止状态的容器，可以通过 docker start 命令来重新启动。

此外，docker restart 命令会将一个运行态的容器终止，然后再重新启动它。

进入容器

在使用 -d 参数时，容器启动后会进入后台。 某些时候需要进入容器进行操作，有很多种方法，包括使用 docker attach 命令或 nsenter 工具等。

attach 命令

docker attach 是Docker自带的命令。下面示例如何使用该命令。

```
$ sudo docker run -idt ubuntu
243c32535da7d142fb0e6df616a3c3ada0b8ab417937c853a9e1c251f499f550
$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
243c32535da7        ubuntu:latest       "/bin/bash"         18 seconds ago
Up 17 seconds                           nostalgic_hypatia
$sudo docker attach nostalgic_hypatia
root@243c32535da7:/#
```

但是使用 attach 命令有时候并不方便。当多个窗口同时 attach 到同一个容器的时候，所有窗口都会同步显示。当某个窗口因命令阻塞时,其他窗口也无法执行操作了。

导出和导入容器

导出容器

如果要导出本地某个容器，可以使用 docker export 命令。

```
$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS                      PORTS               NAMES
7691a814370e        ubuntu:14.04        "/bin/bash"         36 hours ago
Exited (0) 21 hours ago                         test
$ sudo docker export 7691a814370e > ubuntu.tar
```

这样将导出容器快照到本地文件。

导入容器快照

可以使用 docker import 从容器快照文件中再导入为镜像，例如

```
$ cat ubuntu.tar | sudo docker import - test/ubuntu:v1.0
$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
VIRTUAL SIZE
test/ubuntu         v1.0                9d37a6082e97        About a minute ago
171.3 MB
```

此外，也可以通过指定 URL 或者某个目录来导入，例如

```
$sudo docker import http://example.com/exampleimage.tgz example/imagerepo
```

*注：用户既可以使用 docker load 来导入镜像存储文件到本地镜像库，也可以使用 docker import 来导入一个容器快照到本地镜像库。这两者的区别在于容器快照文件将丢弃所有的历史记录和元数据信息（即仅保存容器当时的快照状态），而镜像存储文件将保存完整记录，体积也要大。此外，从容器快照文件导入时可以重新指定标签等元数据信息。

删除容器

可以使用 docker rm 来删除一个处于终止状态的容器。 例如

```
$sudo docker rm  trusting_newton
trusting_newton
```

如果要删除一个运行中的容器，可以添加 -f 参数。Docker 会发送 SIGKILL 信号给容器。

清理所有处于终止状态的容器

用 docker ps -a 命令可以查看所有已经创建的包括终止状态的容器，如果数量太多要一个个删除可能会很麻烦，用 docker rm $(docker ps -a -q) 可以全部清理掉。

*注意：这个命令其实会试图删除所有的包括还在运行中的容器，不过就像上面提过的 docker rm 默认并不会删除运行中的容器。