

🔗 动手实战学Docker (/courses/498)

# 容器管理

## 1. 课程说明

课程为纯动手实验教程，为了能说清楚实验中的一些操作会加入理论内容。理论内容我们不会写太多，已经有太多好文章了，会精选最值得读的文章推荐给你，在动手实践的同时扎实理论基础。

实验环境中可以联网，不受实验楼网络限制。

## 2. 学习方法

实验楼的 Docker 课程包含 14 个实验，每个实验都提供详细的步骤和截图，适用于有一定Linux系统基础，想快速上手 Docker 的同学。

学习方法是多实践，多提问。启动实验后按照实验步骤逐步操作，同时理解每一步的详细内容。

如果实验开始部分有推荐阅读的材料，请务必先阅读后再继续实验，理论知识是实践必要的基础。

## 3. 本节内容简介

容器是 Docker 的一个基本概念，每个容器中都运行一个应用并为该应用提供完整的运行环境。本实验将详细学习Docker 容器的创建，运行管理操作。需要依次完成下面几项任务：

1. docker 命令
2. 查看容器
3. 创建容器
4. 管理容器运行状态
5. 连接容器
6. 其他容器命令

## 4. docker 命令

### 4.1 查看系统信息

除了查看版本信息之外，在 docker 的命令组中还有一个较为常用的命令，查看系统的一些相关信息：

动手实战学 Docker (/courses/498)

或者使用命令

```
docker info
```

运行截图如下所示：

```
shiyancelou:~/ $  
shiyancelou:~/ $ docker info  
Containers: 1  
Running: 0  
Paused: 0  
Stopped: 1  
Images: 2  
Server Version: 17.05.0-ce  
Storage Driver: aufs  
Root Dir: /var/lib/docker/aufs  
Backing Filesystem: extfs  
Dirs: 8  
Dirperm1 Supported: true  
Logging Driver: json-file  
Cgroup Driver: cgroupfs  
Plugins:  
Volume: local  
Network: bridge host macvlan null overlay  
Swarm: inactive  
Runtimes: runc  
Default Runtime: runc  
Init Binary: docker-init
```

容器

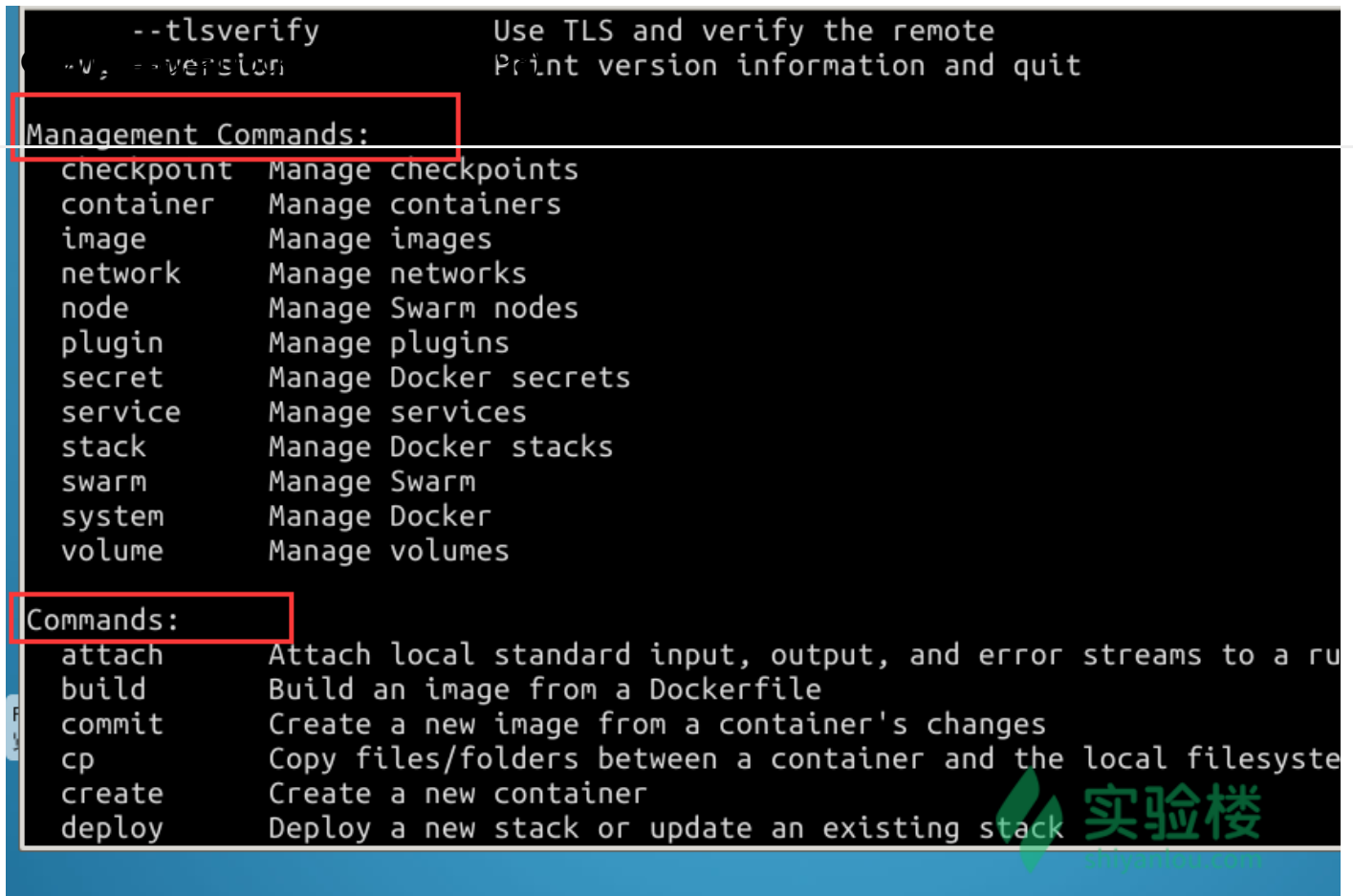
镜像

## 4.2 help

我们可以直接通过 `help` 或者使用 `man` 手册的方式查看相关命令的详细说明，例如我们直接使用如下命令：

```
$ docker --help
```

我们可以看到运行结果如下图所示。如果之前有学习过 docker 相关知识的同学，可能会发现一些不一样的地方。即下图中标出的 Management commands 和 Commands。在 1.13 版本之前，docker 并没有 Management commands。



## 4.3 Management Commands

在 Docker 1.12 CLI 中大约有四十个左右的顶级命令，这些命令没有经过任何组织，显得十分混乱，对于新手来说，学习它们并不轻松。

而在 Docker 1.13 中将命令进行分组，就得到如上图中所示的 Management Commands。例如经常使用的容器的一些相关命令：

```
# 创建一个新的容器，下面分别为 Commands 和 Management Commands，作用相同
docker create
docker container create

# 显示容器列表
docker ps
docker container ls

# 在一个新的容器中运行一个命令
docker run
docker container run

...
```

如上所示，对于新的命令而言相比于旧命令明显更具有可读性。并且在实验环境中的 docker 版本以及最新版本中两者都是有效的命令，所以在这里我们将一些常用的命令，及其对应的 Management Commands 命令都列举出来，方便大家在后续的学习过程中可以进行参考。

## 🔗 动手实战学 Docker (/courses/498)

Commands	Management Commands
docker attach	docker container attach
docker build	docker image build
docker commit	docker container commit
docker cp	docker container cp
docker create	docker container create
docker deploy	docker stack deploy
docker exec	docker container exec
docker images	docker image ls
docker import	docker image import
docker info	docker system info
docker inspect	docker container/image inspect
docker kill	docker container kill
docker port	docker container port
docker ps	docker container ls
docker restart	docker container restart
docker rm	docker container rm
docker rmi	docker image rm
docker run	docker container run
docker start	docker container start
docker top	docker container top
...	...



## 5. 容器

### 5.1 查看容器列表

查看容器列表可以使用如下命令：

```
docker container ls [OPTIONS]
```

或者旧的命令

```
docker ps [OPTIONS]
```

在使用命令时，我们可以使用一些可选的配置项 [OPTIONS]。

- -a 显示所有的容器
  - 🔗 动手实战学Docker (/courses/498)
  - -q 仅显示 ID
- 
- -s 显示总的文件大小

这些配置项对于上述的两个命令都是有效的，在后面的内容不会再特殊说明。

默认情况下，直接使用该命令仅显示正在运行的容器，如下所示：

```
$ docker container ls
```

```
shiyanolou:~/ $ docker container ls
CONTAINER ID        IMAGE               COMMAND
PORTS              NAMES
shiyanolou:~/ $
shiyanolou:~/ $
```



此时并没有处于运行中的容器，所以显示为空。我们可以使用 -a 参数，来显示所有的容器，并加上 -s 选项，显示大小，命令如下：

```
$ docker container ls -a -s
```

限于界面大小，为了图片的格式更加友好，这里仅 截取部分输出结果：

```
shiyanolou:~/ $ docker container ls -a -s
CONTAINER ID        IMAGE               COMMAND
d26f5ec4183c        hello-world         "/hello"
shiyanolou:~/ $
shiyanolou:~/ $
```

CREATED About an hour ago STATUS Exited (0)



## 5.2 创建一个容器

### docker run

首先，我们回顾在上一节使用到的 docker run hello-world 命令，该命令的格式为：

```
docker run [OPTIONS] IMAGE [COMMAND]
```

对应于 Management Commands 的命令为：

```
docker container run [OPTIONS] IMAGE [COMMAND]
```

上述两个命令的作用相同，docker run 命令会在指定的镜像 IMAGE 上创建一个可写的容器（因为镜像只是只读的），然后开始运行指定的命令 [COMMAND]。

一些常用的配置项为：

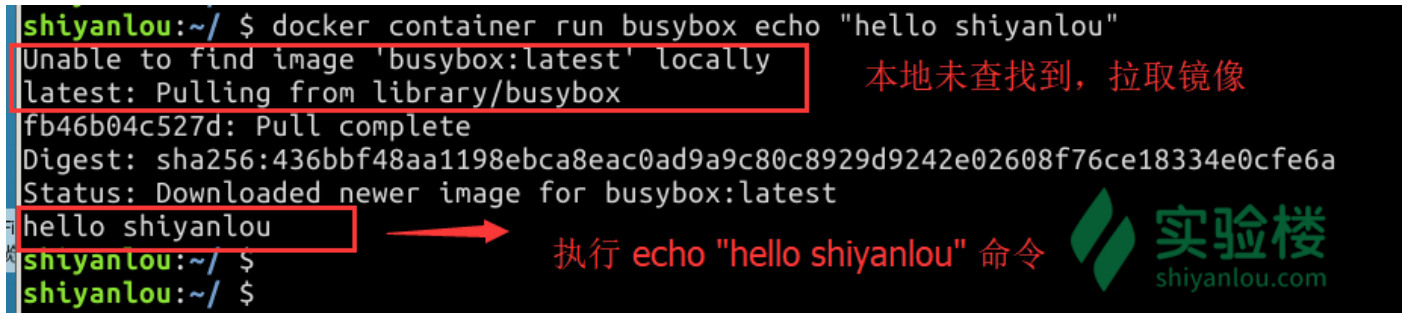
👉 动手实战学Docker (/courses/498)

- -i 或 --interactive , 交互模式
- -t 或 --tty , 分配一个 pseudo-TTY , 即伪终端
- --rm 在容器退出后自动移除
- -p 将容器的端口映射到主机
- -v 或 --volume , 指定数据卷

关于该命令的详细参数较多，并且大多数参数在很多命令中的意义是相同的，将在后面的内容中使用到时进行相应的介绍。

我们指定 busybox 镜像，然后运行命令 echo "hello shiyanlou" 命令，如下所示：

```
$ docker container run busybox echo "hello shiyanlou"
```



```
shiyanlou:~/ $ docker container run busybox echo "hello shiyanlou"
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
fb46b04c527d: Pull complete
Digest: sha256:436bbf48aa1198ebca8eac0ad9a9c80c8929d9242e02608f76ce18334e0cfe6a
Status: Downloaded newer image for busybox:latest
hello shiyanlou
shiyanlou:~/ $
shiyanlou:~/ $
```

本地未查找到，拉取镜像

执行 echo "hello shiyanlou" 命令

实验楼  
shiyanlou.com

在上图中，我们可以看到该命令执行的过程：

1. 对于指定镜像而言，首先会从本地查找，找不到时将会从镜像仓库中下载该镜像
2. 镜像下载完成后，通过镜像启动容器，并运行 echo "hello shiyanlou" 命令，输出运行结果之后退出。

在执行命令之后，容器就会退出，如果我们需要一个保持运行的容器，最简单的方法就是给这个容器一个可以保持运行的命令或者应用，比如 bash，例如我们在 ubuntu 容器中运行 /bin/bash 命令：

```
$ docker container run -i -t ubuntu /bin/bash
```

对于交互式的进程而言（例如这里的 bash），必须将 -i 和 -t 参数一起使用，才能为容器进程分配一个伪终端，通常会直接使用 -it。

```
shiyancelou:~/ $ docker container run -i -t ubuntu /bin/bash
root@dfc1f4c50197:/#
root@dfc1f4c50197:/# ls
bin  dev  home  lib64  mnt  proc  run  srv  tmp  var
boot  etc  lib  media  opt  root  sbin  sys  usr
root@dfc1f4c50197:/#
root@dfc1f4c50197:/#
```



如上所示，我们已经进入到分配的终端中了，这时如果我们需要退出 `bash`，可以使用以下两种方式，它们的效果完全不同：

1. 直接使用 `exit` 命令，这时候 `bash` 程序终止，容器进入到停止状态
2. 使用组合键退出，容器仍然保持运行的状态，可以再次连接到这个 `bash` 中，组合键是 `ctrl + p` 和 `ctrl + q`。即先同时按下 `ctrl` 和 `p` 键，再同时按 `ctrl` 和 `q` 键。就可以退出

这里我们使用第二种方式，然后使用 `docker container ls` 命令，可以看到该容器仍然处于运行中。

## docker container create

严格意义上来讲，`docker run` 命令的作用并不是创建一个容器，而是在一个新的容器中运行一个命令。而用于创建一个新容器的命令为

```
docker container create [OPTIONS] IMAGE [COMMAND] [ARG...]
```

或者使用旧的

```
docker create [OPTIONS] IMAGE [COMMAND] [ARG...]
```

该命令会在指定的镜像 `IMAGE` 上创建一个可写容器层，并 **准备** 运行指定的命令。需要着重强调的是，这里是准备运行，并不是立即运行。即该命令只创建容器，并不会运行容器。

一些常见的配置项如下所示：

- `--name` 指定一个容器名称，未指定时，会随机产生一个名字。
- `--hostname` 设置容器的主机名
- `--mac-address` 设置 MAC 地址
- `--ulimit` 设置 Ulimit 选项。



关于上述提到的 `ulimit`，我们可以通过其对容器运行时的一些资源进行限制。 `ulimit` 是一种 linux 系统的内建功能，一些简单的描述，可以参考 <https://www.ibm.com/developerworks/cn/linux/l-cn-ulimit/> (<https://www.ibm.com/developerworks/cn/linux/l-cn-ulimit/>)，而对于在下面我们将要设置的部分值的含义，可以参考 <https://access.redhat.com/solutions/61334>。

除此之外，关于创建容器，我们还可以设置有关存储和网络的全部内容，将会在下一节的内容中进行介绍。

如下示例，我们指定容器的名字为 `shianlou`，主机名为 `shianlou`，设置相应的 MAC 地址，并通过 `ulimit` 设置最大进程数（1024:2048 分别代表软硬资源限制，详细内容可以参考上面的链接），使用 `ubuntu` 的镜像，并运行 `bash`：

```
$ docker container create --name shianlou --hostname shianlou --mac-address 00:01:02:03:04:05 --ulimit nproc=1024:2048 -it ubuntu /bin/bash
```

```
shianlou:~/ $ docker container create --name shianlou --hostname shianlou --mac-address 00:01:02:03:04:05 --ulimit nproc=1024:2048 -it ubuntu /bin/bash
2dcbec8a85e20f7e5be545c80f76647003730ca68e90f63026fa0c68c2989e29
shianlou:~/ $
shianlou:~/ $
```

打印容器的 ID

此时，容器创建成功后，会打印该容器的 ID，这里需要简单说明一下，在 `docker` 中，容器的标识有三种比较常见的标识方式：

- UUID 长标识符，例如 `1f6789f885029dbdd4a6426d7b950996a5bcc1cccec9f8185240313aa1badeaf`
- UUID 短标识符，从长标识符开始，只要不与其它标识符冲突，可以从头开始，任意选用位数，例如针对上面的长标识符，可以使用 `1f`，`1f678` 等等
- Name 最后一种方式即是使用容器的名字

在容器创建成功后，我们可以查看其运行状态，使用如下命令：

```
# 此时该容器并未运行，需要使用 -a 参数
$ docker container ls -a
```

```
shianlou:~/ $ docker container ls -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS                    PORTS          NAMES
dfc1f4c50197   ubuntu    "/bin/bash"             3 minutes ago   Exited (0) About a minute ago           happy_lewin
ab38cb3ce0a0   busybox    "echo 'hello shiya..." 3 minutes ago   Exited (0) 3 minutes ago               ecstatic_roent
0f319cdb3a35   hello-world "/hello"                4 minutes ago   Exited (0) 4 minutes ago               awesome_brown
2dcbec8a85e2   ubuntu    "/bin/bash"             8 minutes ago   Created                                shianlou
```

新创建的容器的状态 (STATUS) 为 `Created`，并且其容器名被设置为对应的值。



## 5.3 查看容器的详细信息

动手实践 Docker (/courses/498)

查看容器的详细信息可以使用如下命令：

```
docker container inspect [OPTIONS] CONTAINER [CONTAINER...]
```

或者旧的

```
docker inspect [OPTIONS] CONTAINER [CONTAINER...]
```

例如我们查看刚刚创建的容器的详细信息就可以使用以下命令：

```
# 使用容器名
$ docker container inspect shiyanlou

# 使用 ID ， 因生成的 ID 不同，需要修改为相应的 ID
$ docker container inspect 1f6789

$ docker container inspect 1f6
```

例如，我们查看刚刚创建的名为 shiyanlou 的容器的 MAC 地址，就可以使用如下命令：

```
$ docker container inspect shiyanlou | grep "00:01"
```

```
shiyanlou:~/ $
shiyanlou:~/ $ docker container inspect shiyanlou | grep "00:01"
    "MacAddress": "00:01:02:03:04:05",
shiyanlou:~/ $
shiyanlou:~/ $
```

## 5.4 容器的启动和暂停及退出

容器的启动命令为：

```
docker container start [OPTIONS] CONTAINER [CONTAINER...]
```

对于上面我们创建的容器而言，此时处于 Created 状态，需要使用如下命令启动它：

```
$ docker container start shiyanlou
```

```
shiyanlou:~/ $ docker container start shiyanlou
shiyanlou
shiyanlou:~/ $
```

此时，运行一个容器我们分成了两个步骤，即创建和启动，使用的命令如下：

### 🔗 动手实战学 Docker (/courses/498)

```
$ docker container create --name shiyanlou --hostname shiyanlou --mac-address 00:01:02:03:04:05 --ulimit nproc=1024:2048 -it ubuntu /bin/bash
```

```
# 启动
```

```
$ docker container start shiyanlou
```

上述的两个命令如果我们使用 `docker container run` 只需要一步即可，即此时 `run` 命令同时完成了 `create` 及 `start` 操作：

```
$ docker container run --name shiyanlou --hostname shiyanlou --mac-address 00:01:02:03:04:05 --ulimit nproc=1024:2048 -it ubuntu /bin/bash
```

除此之外，上面的 `run` 命令还完成一些其它的操作，例如没有镜像时会 `pull` 镜像，使用 `-it` 参数时完成了 `attach` 操作（后面会学习该操作），使用 `--rm` 参数在容器退出后还会完成 `container rm` 操作。

`run` 命令是一个综合性的命令，如果能够熟练的使用它可以简化很多步骤，但是其使用方式较为复杂

启动之后，暂停容器可以使用如下命令：

```
# 暂停一个或多个容器
```

```
docker container stop [OPTIONS] CONTAINER [CONTAINER...]
```

```
# 暂停一个或多个容器中的所有进程
```

```
docker container pause CONTAINER [CONTAINER...]
```

上述两个命令的区别在于一个是暂停容器中的进程，而另外一个则是暂停容器，例如，我们使用 `stop` 停止刚刚启动的容器就可以使用如下命令：

```
$ docker container stop shiyanlou
```

```
# 查看容器的状态
```

```
$ docker container ls -a
```

```
shiyanlou:~/ $ docker container stop shiyanlou
shiyanlou
shiyanlou:~/ $ docker container ls -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
dfc1f4c50197	ubuntu	"/bin/bash"	11 minutes ago	Exited (0) 9 minutes ago	
ab38cb3ce0a0	busybox	"echo 'hello shiya...'"	11 minutes ago	Exited (0) 11 minutes ago	
0f319cdb3a35	hello-world	"/hello"	12 minutes ago	Exited (0) 12 minutes ago	
2dcbec8a85e2	ubuntu	"/bin/bash"	15 minutes ago	Exited (0) 5 seconds ago	

```
shiyanlou:~/ $
shiyanlou:~/ $
```

如上图所示，容器被暂停后，此时处于 `Exited` 状态。

## 5.5 连接到容器

动手实战学 Docker (/courses/498)

上述操作我们启动的容器运行于后台，所以，我们需要使用 `attach` 操作将本地标准输入输出流连接到正在运行的容器，命令格式为：

```
docker container attach [OPTIONS] CONTAINER
```

如下示例，我们启动容器，并使用连接命令：

```
$ docker container start shiyanlou
$ docker container attach shiyanlou
```

```
shiyanlou:~/ $ docker container start shiyanlou
shiyanlou
shiyanlou:~/ $ docker container attach shiyanlou
root@shiyanlou:/#
root@shiyanlou:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:01:02:03:04:05
          inet addr:192.168.0.2  Bcast:0.0.0.0  Mask:255.255.240.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@shiyanlou:/#
```



连接到容器后，查看相应的主机名和 Mac 地址，可以判断我们连接到了刚刚创建的容器。

## 5.6 其它命令

除了上面介绍的一些命令之外，还有很多其它的命令，下面简单描述

### 获取日志

获取容器的输出信息可以使用如下命令：

```
docker container logs [OPTIONS] CONTAINER
```

常用的配置项有：

- -t 或 --timestamps 显示时间戳
- 🕒 动手实战学Docker (/courses/498)
- -f 实时输出，类似于 tail -f

如下所示，我们查看刚刚创建的容器的日志，使用如下命令：

```
$ docker container logs -tf shiyanlou
```

## 显示进程

除了获取日志之外，还可以显示运行中的容器的进程信息，例如查看刚刚创建的容器的进程信息：

```
$ docker container top shiyanlou
```

```
shiyanlou:~/ $ docker container top shiyanlou
UID          PID         PPID        C          STIME      TTY        TIME       CMD
root         28138      28118       0          16:23      pts/7      00:00:00   /bin/bash
shiyanlou:~/ $
```

需要注意的是，该命令对于并未运行的容器是无效的

## 查看修改

查看相对于镜像的文件系统来说，容器中做了哪些改变，可以使用如下命令：

```
docker container diff shiyanlou
```

例如我们在 shiyanlou 容器中创建一个文件，就可以使用 diff 命令查看到相应的修改：

```
shiyanlou:~/ $ docker attach shiyanlou
root@shiyanlou:/#
root@shiyanlou:/#
root@shiyanlou:/# ls
bin    dev    home  lib64  mnt    proc  run    srv    tmp    var
boot  etc    lib   media  opt    root  sbins  sys    usr
root@shiyanlou:/# touch test1
root@shiyanlou:/# %
shiyanlou:~/ $
shiyanlou:~/ $ docker container diff shiyanlou
A /test1
shiyanlou:~/ $
```

## 重启

重启容器可以使用如下命令：

动手实战学 Docker (courses/498)

## 执行命令

除了使用 `docker container run` 执行命令之外，我们还可以在一个运行中的容器中执行命令，使用如下格式：

```
docker container exec [OPTIONS] CONTAINER COMMAND [ARG...]
```

例如，我们在刚刚创建的容器中执行 `echo "test_exec"` 命令，就可以使用如下命令：

```
$ docker container exec shiyanlou echo "test_exec"
```

```
shiyanlou:~/ $ docker container exec shiyanlou echo "test_exec"
test_exec
shiyanlou:~/ $
```

## 删除容器

删除容器的命令：

```
docker container rm [OPTIONS] CONTAINER [CONTAINER...]
```

需要注意的是，在删除容器后，在容器中进行的操作并不会持久化到镜像中

## 6. 总结

本节实验中我们学习了以下内容：

1. docker 命令
2. 查看容器
3. 创建容器
4. 管理容器运行状态
5. 连接容器
6. 其他容器命令

请务必保证自己能够动手完成整个实验，只看文字很简单，真正操作的时候会遇到各种各样的问题，解决问题的过程才是收获的过程。

*\*本课程内容，由作者授权实验楼发布，未经允许，禁止转载、下载及非法传播。*

 [动手实战学Docker概念及基本用法 \(/courses/498/labs/1702/document\)](/courses/498/labs/1702/document)

下一节： Docker 镜像管理 (/courses/498/labs/1705/document)