Docker命令比较多,并且每个命令有很多参数项，这里仅列出比较常用的一些操作（镜像相关、容器相关、调试等），以便有一个直观的印象。

Docker的命令清单可以通过运行 docker，或者 docker help 命令得到。

镜像操作相关

查找镜像

我们可以从Docker Hub网站来搜索镜像，Docker Hub 网址为： https://hub.docker.com/

我们也可以使用 docker search 命令来搜索镜像。

比如我们需要一个mysql的镜像来作为我们的数据库。我们可以通过 docker search 命令搜索 mysql 来寻找适合我们的镜像。

```
[root@localhost ~]# docker search mysql
NAME       DESCRIPTION                                     STARS     OFFICIAL
AUTOMATED
mysql      MySQL is a widely used, open-source relati...   4753      [OK]
mariadb    MariaDB is a community-developed fork of M...   1463      [OK]
percona    Percona Server is a fork of the MySQL rela...   283       [OK]
```

输出字段含义：

1. NAME: 镜像仓库源的名称
2. DESCRIPTION: 镜像的描述
3. OFFICIAL: 是否docker官方发布

从docker仓库中获取镜像

使用 docker 必须自少有一个基础镜像包，镜像包可以从公共仓库中获取，也可以搭建私有的镜像仓库。

本例使用 docker 的默认配置，docker pull 命令会自动从docker.io 这个仓库去下载 busybox 镜像包。

```
[root@localhost ~]# docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
9e87eff13613: Pull complete
Digest:
sha256:2605a2c4875ce5eb27a9f7403263190cd1af31e48a2044d400320548356251c4
Status: Downloaded newer image for busybox:latest
```

后续会搭建私有仓库，并针对私有仓库的使用另外做一个说明。

查看本机的镜像包

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
busybox             latest              efe10ee6727f        2 weeks ago
1.13MB
hello-world         latest              1815c82652c0        7 weeks ago
1.84kB
```

通过 docker images 命令，可以查看本机所有的镜像包。这里解释下命令输出中每个字段的意义：

1. REPOSITORY：该字段包括两部分，第一部分是仓库名字（docker.io），第二部分是镜像包的名字。
2. TAG：用来表征镜像的版本号，默认情况下拖取最新版本（latest），也可以指定镜像的具体版本。
3. IMAGE_ID：一个哈希值，用来唯一标志一个镜像文件。
4. CREATED：镜像创建的时间。
5. VIRTUAL SIZE：镜像的大小。

利用镜像包启动容器

```
[root@localhost hello]# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://cloud.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/
```

docker run 命令的作用是用来启动一个容器。

如上图所示，通过 hello-world 镜像包启动一个容器。

查看镜像包的历史信息

如前所述，镜像包是累加制作的，通过docker history命令，可以看到当前镜像包的整体叠加构建的历史。

```
[root@localhost hello]# docker history busybox
IMAGE               CREATED             CREATED BY
SIZE                COMMENT
efe10ee6727f        2 weeks ago         /bin/sh -c #(nop)  CMD ["sh"]
0B
<missing>           2 weeks ago         /bin/sh -c #(nop) ADD
file:0516fc7a5988 ...   1.13MB
```

容器操作相关

运行一个带交互的容器

```
[root@localhost ~]# docker run -i -t busybox
/ # ls
bin   dev   etc   home  proc  root  sys   tmp   usr   var
/ #
```

本例中，docker run 的-i 选项使得容器可以接收外部的标准输入，-t 选项使得容器内部开启一个伪终端，加上这两个参数后，就可以通过命令行接口和容器进行交互。

查看运行态容器的信息

```
[root@localhost blog]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
1575c6863d8b        busybox             "sh"                3 minutes ago
Up 3 minutes                            keen_fermat
```

docker ps 命令可以查看处于运行态的容器信息，这里解释一下命令输出中各个字段 的意义：

1. CNTAINER ID：容器的 ID 号，一个哈希值，用于唯一确认一个容器。
2. IMAGE： 此容器对应的镜像包。
3. COMMAND： 该容器启动后，在该容器内运行的最近的一个命令。
4. CREATED：容器创建的时间。
5. STAUTS：容器当前的状态。
6. PORTS：容器所开放的端口（本例未分配端口给该容器）。
7. NAMES： docker 会给容器随机分配一个名字，以方便引用。也可以通过 run 命令的--name 选项手动指定。

跟踪容器对镜像做出的修改

```
[root@localhost ~]# docker attach 1575c6863d8b
/ # cd /home
/home # ls
/home # touch test
/home # read escape sequence
[root@localhost ~]# docker diff 1575c6863d8b
C /home
A /home/test
C /root
A /root/.ash_history
```

docker的底层使用了 AUFS文件系统，镜像包是累进叠加的，最终的镜像被docker run命令生成一个容器后，该容器内部有一个 writable 的区域可以对镜像的内容进行修改，但镜像本身是只读的。

本例描述的过程如下：

1. 通过 docker attach 命令附着到正在运行的容器中。
2. 在容器内做一些修改，本例中在/home 目录中新增了一个名为test的文件。
3. 先后执行“ CTRL+P” +“ CTRL +Q”返回到本机 shell。
4. 执行 docker diif 命令，可以看到容器相对于初始镜像做出的修改（ C:changed， A：added, D：deleted）。

容器的停止

```
[root@localhost ~]# docker stop 1575c6863d8b
1575c6863d8b
```

可以通过 docker stop 命令停止一个容器的运行。

docker ps 命令只能看处于运行态的容器，如果想要查看所有的容器运行记录，需要通过 docker ps –a 命令。

```
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS                          PORTS               NAMES
1575c6863d8b        busybox             "sh"                     13 minutes
ago      Exited (137) 2 minutes ago                       keen_fermat
e52b8d7d0b15        busybox             "echo 'Hello World\\!'"   2 days ago
Exited (0) 2 days ago                           dazzling_lalande
bb30ab886b21        hello-world         "/hello"                 2 days ago
Exited (0) 2 days ago                           sleepy_fermi
```

docker 还支持 docker pause,docker unpause 来暂停和恢复一个容器的执行,具体可以查看命令帮助。

监控操作

docker exec命令

docker exec命令的实质是进入到一个正在运行的容器中，并启动一个新的进程来监控容器，类似于 SSH 的功能。

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
1dfa0e308e9b        ubuntu              "/bin/bash"         4 hours ago
Up 4 hours                              determined_lalande
[root@localhost ~]# docker exec -it 1dfa0e308e9b bash
root@localhost:/#
```

本例描述的步骤如下：

1. 使用 docker run 命令在后台启动一个 server 容器。
2. 在本机 shell 中通过 docker ps 命令找到该容器对应的 ID。
3. 通过 docker exec 命令登录到容器中，从命令行可以看出，已经进入了容器命令行的 界面。

docker stats命令

通过 docker stats 命令，可以看到容器各项资源的使用情况。

```
[root@localhost hello]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
25ce06e8b42c        busybox             "sh"                8 seconds ago
Up 7 seconds                            serene_payne
[root@localhost hello]# docker stats 25ce06e8b42c
CONTAINER           CPU %               MEM USAGE / LIMIT   MEM %
NET I/O             BLOCK I/O           PIDS
25ce06e8b42c        0.00%               56KiB / 7.443GiB    0.00%
648B / 0B           0B / 0B             0
```

本例描述的步骤如下：

1. 在本机 shell 中通过 docker ps 命令找到需要监控的容器对应的 ID。
2. 运行 docker stats 命令，可以查看正在运行的进程信息。 从输出可以看到容器占用的CPU、内存、硬盘、网络的资源使用情况

docker logs命令

```
[root@192 lxb]# docker logs 25ce06e8b42c
```

通过docker logs可以捕获到容器内部STDOUT和STD_ERR输出的信息，便于定位问题。 本例描述的步骤如下：

1. 在本机 shell 中通过 docker ps 命令找到需要监控的容器对应的 ID。
2. 运行 docker logs 命令，可以查看容器内部的标准输出、标准错误信息。