docker命令有很多，总的分为以下几大类：

容器生命周期管理 — docker [run|start|stop|restart|kill|rm|pause|unpause]  
容器操作运维 — docker [ps|inspect|top|attach|events|logs|wait|export|port]  
容器rootfs命令 — docker [commit|cp|diff]  
镜像仓库 — docker [login|pull|push|search]  
本地镜像管理 — docker [images|rmi|tag|build|history|save|import]  
其他命令 — docker [info|version]

以上只是日常中大部分使用的命令，更详细的可以查看官网或者命令帮助

**docker –help**

Usage: docker [OPTIONS] COMMAND [arg...]  
 docker [ --help | -v | --version ]  
A self-sufficient runtime for containers.  
Options:  
 --config=~/.docker Location of client config files  
 -D, --debug Enable debug mode  
 -H, --host=[] Daemon socket(s) to connect to  
 -h, --help Print usage  
 -l, --log-level=info Set the logging level  
 --tls Use TLS; implied by --tlsverify  
 --tlscacert=~/.docker/ca.pem Trust certs signed only by this CA  
 --tlscert=~/.docker/cert.pem Path to TLS certificate file  
 --tlskey=~/.docker/key.pem Path to TLS key file  
 --tlsverify Use TLS and verify the remote  
 -v, --version Print version information and quit  
Commands:  
 attach Attach to a running container  
 build Build an image from a Dockerfile  
 commit Create a new image from a container's changes  
 cp Copy files/folders between a container and the local filesystem  
 create Create a new container  
 diff Inspect changes on a container's filesystem  
 events Get real time events from the server  
 exec Run a command in a running container  
 export Export a container's filesystem as a tar archive  
 history Show the history of an image  
 images List images  
 import Import the contents from a tarball to create a filesystem image  
 info Display system-wide information  
 inspect Return low-level information on a container, image or task  
 kill Kill one or more running containers  
 load Load an image from a tar archive or STDIN  
 login Log in to a Docker registry.  
 logout Log out from a Docker registry.  
 logs Fetch the logs of a container  
 network Manage Docker networks  
 node Manage Docker Swarm nodes  
 pause Pause all processes within one or more containers  
 port List port mappings or a specific mapping for the container  
 ps List containers  
 pull Pull an image or a repository from a registry  
 push Push an image or a repository to a registry  
 rename Rename a container  
 restart Restart a container  
 rm Remove one or more containers  
 rmi Remove one or more images  
 run Run a command in a new container  
 save Save one or more images to a tar archive (streamed to STDOUT by default)  
 search Search the Docker Hub for images  
 service Manage Docker services  
 start Start one or more stopped containers  
 stats Display a live stream of container(s) resource usage statistics  
 stop Stop one or more running containers  
 swarm Manage Docker Swarm  
 tag Tag an image into a repository  
 top Display the running processes of a container  
 unpause Unpause all processes within one or more containers  
 update Update configuration of one or more containers  
 version Show the Docker version information  
 volume Manage Docker volumes  
 wait Block until a container stops, then print its exit code

**下面是命令的用法和简单实例：**

**attach**  Attach to a running container #将终端依附到容器上

语法：Usage: docker attach [OPTIONS] CONTAINER  
实例：进入my\_container容器  
[root@localhost ~]# docker attach my\_container   
[root@d4a75f165ce6 /]#

**build** Build an image from a Dockerfile #通过Dockerfile创建镜像

语法：Usage: docker build [OPTIONS] PATH | URL | -  
实例：使用当前目录的Dockerfile创建镜像。  
[root@localhost ~]#docker build -t runoob/ubuntu:v1 .

**commit**  Create a new image from a container's changes #通过容器创建本地镜像

语法：Usage: docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]  
实例：创建一个镜像  
[root@localhost ~]# docker commit centos\_v1 centos:v1   
68ad49c999496cff25fdda58f0521530a143d3884e61bce7ada09bdc22337638

**cp**  Copy files/folders between a container and the local filesystem #在宿主机和容器之间相互拷贝文件

语法：Usage: docker cp [OPTIONS] CONTAINER:SRC\_PATH DEST\_PATH|-  
 docker cp [OPTIONS] SRC\_PATH|- CONTAINER:DEST\_PATH  
实例：容器mysql中/usr/local/bin/存在docker-entrypoint.sh文件，可如下方式copy到宿主机  
[root@localhost ~]#docker cp mysql:/usr/local/bin/docker-entrypoint.sh /root  
修改完毕后，将该文件重新copy回容器  
[root@localhost ~]# docker cp /root/docker-entrypoint.sh mysql:/usr/local/bin/

**create**  Create a new container #仅创建一个容器

语法：Usage: docker create [OPTIONS] IMAGE [COMMAND] [ARG...]  
实例：使用docker镜像nginx:latest创建一个容器,并将容器命名为myrunoob  
[root@localhost ~]# docker create --name myrunoob nginx:latest   
09b93464c2f75b7b69f83d56a9cfc23ceb50a48a9db7652ee4c27e3e2cb1961f

**diff**  Inspect changes on a container's filesystem #查看容器内发生改变的文件

语法：Usage: docker diff CONTAINER  
实例：查看容器mymysql的文件结构更改。  
[root@localhost ~]#docker diff mymysql  
A /logs  
A /mysql\_data  
C /run  
C /run/mysqld  
A /run/mysqld/mysqld.pid  
A /run/mysqld/mysqld.sock  
C /tmp

**events** Get real time events from the server #实时输出docker服务器的事件，包括容器的创建、启动和关闭等

语法：Usage: docker events [OPTIONS]  
实例：显示docker 2016年7月1日后的所有事件。  
[root@localhost ~]# docker events --since="1467302400"  
2016-07-08T19:44:54.501277677+08:00 network connect 66f958fd13dc4314ad20034e576d5c5eba72e0849dcc38ad9e8436314a4149d4 (container=b8573233d675705df8c89796a2c2687cd8e36e03646457a15fb51022db440e64, name=bridge, type=bridge)  
2016-07-08T19:44:54.723876221+08:00 container start b8573233d675705df8c89796a2c2687cd8e36e03646457a15fb51022db440e64 (image=nginx:latest, name=elegant\_albattani)  
2016-07-08T19:44:54.726110498+08:00 container resize b8573233d675705df8c89796a2c2687cd8e36e03646457a15fb51022db440e64 (height=39, image=nginx:latest, name=elegant\_albattani, width=167)  
2016-07-08T19:46:22.137250899+08:00 container die b8573233d675705df8c89796a2c2687cd8e36e03646457a15fb51022db440e64 (exitCode=0, image=nginx:latest, name=elegant\_albattani)  
显示docker 镜像为mysql:5.6 2016年7月1日后的相关事件。  
[root@localhost ~]# docker events -f "image"="mysql:5.6" --since="1467302400"   
2016-07-11T00:38:53.975174837+08:00 container start 96f7f14e99ab9d2f60943a50be23035eda1623782cc5f930411bbea407a2bb10 (image=mysql:5.6, name=mymysql)  
2016-07-11T00:51:17.022572452+08:00 container kill 96f7f14e99ab9d2f60943a50be23035eda1623782cc5f930411bbea407a2bb10 (image=mysql:5.6, name=mymysql, signal=9)  
2016-07-11T00:51:17.132532080+08:00 container die 96f7f14e99ab9d2f60943a50be23035eda1623782cc5f930411bbea407a2bb10 (exitCode=137, image=mysql:5.6, name=mymysql)  
2016-07-11T00:51:17.514661357+08:00 container destroy 96f7f14e99ab9d2f60943a50be23035eda1623782cc5f930411bbea407a2bb10 (image=mysql:5.6, name=mymysql)  
2016-07-11T00:57:18.551984549+08:00 container create c8f0a32f12f5ec061d286af0b1285601a3e33a90a08ff1706de619ac823c345c (image=mysql:5.6, name=mymysql)  
2016-07-11T00:57:18.557405864+08:00 container attach c8f0a32f12f5ec061d286af0b1285601a3e33a90a08ff1706de619ac823c345c (image=mysql:5.6, name=mymysql)  
2016-07-11T00:57:18.844134112+08:00 container start c8f0a32f12f5ec061d286af0b1285601a3e33a90a08ff1706de619ac823c345c (image=mysql:5.6, name=mymysql)  
如果指定的时间是到秒级的，需要将时间转成时间戳。如果时间为日期的话，可以直接使用，如--since="2016-07-01"。

**exec**  Run a command in a running container #对一个启动的容器执行命令

语法：Usage: docker exec [OPTIONS] CONTAINER COMMAND [ARG...]  
实例：在容器mynginx中以交互模式执行容器内/root/runoob.sh脚本  
[root@localhost ~]# docker exec -it mynginx /bin/sh /root/runoob.sh  
http://www.runoob.com/

**export**  Export a container's filesystem as a tar archive #将容器打包，导出为tar文件格式

语法：Usage: docker export [OPTIONS] CONTAINER  
实例：将id为a404c6c174a2的容器按日期保存为tar文件。  
[root@localhost ~]#docker export -o mysql-`date +%Y%m%d`.tar a404c6c174a2  
[root@localhost ~]#ls mysql-`date +%Y%m%d`.tar  
mysql-20160711.tar

**history** Show the history of an image#显示镜像的创建过程

语法： Usage: docker history [OPTIONS] IMAGE  
实例： 查看本地镜像runoob/ubuntu:v3的创建历史。  
[root@localhost ~]# docker history runoob/ubuntu:v3  
IMAGE CREATED CREATED BY SIZE COMMENT  
4e3b13c8a266 3 months ago /bin/sh -c #(nop) CMD ["/bin/bash"] 0 B   
<missing> 3 months ago /bin/sh -c sed -i 's/^#\s\*\(deb.\*universe\)$/ 1.863 kB   
<missing> 3 months ago /bin/sh -c set -xe && echo '#!/bin/sh' > /u 701 B   
<missing> 3 months ago /bin/sh -c #(nop) ADD file:43cb048516c6b80f22 136.3 MB

**images**  List images #列出本地所有的镜像

语法：Usage: docker images [OPTIONS] [REPOSITORY[:TAG]]  
实例：查看本地镜像列表。  
[root@localhost ~]#docker images  
REPOSITORY TAG IMAGE ID CREATED SIZE  
mymysql v1 37af1236adef 5 minutes ago 329 MB  
runoob/ubuntu v4 1c06aa18edee 2 days ago 142.1 MB  
<none> <none> 5c6e1090e771 2 days ago 165.9 MB  
httpd latest ed38aaffef30 11 days ago 195.1 MB  
alpine latest 4e38e38c8ce0 2 weeks ago 4.799 MB  
mongo 3.2 282fd552add6 3 weeks ago 336.1 MB  
redis latest 4465e4bcad80 3 weeks ago 185.7 MB  
php 5.6-fpm 025041cd3aa5 3 weeks ago 456.3 MB  
python 3.5 045767ddf24a 3 weeks ago 684.1 MB  
...  
列出本地镜像中REPOSITORY为ubuntu的镜像列表。  
[root@localhost ~]# docker images ubuntu  
REPOSITORY TAG IMAGE ID CREATED SIZE  
ubuntu 14.04 90d5884b1ee0 9 weeks ago 188 MB  
ubuntu 15.10 4e3b13c8a266 3 months ago 136.3 MB

**import**  Import the contents from a tarball to create a filesystem image #导入一个镜像，类型为tar文件

语法：Usage: docker images [OPTIONS] [REPOSITORY[:TAG]]  
实例：从镜像归档文件my\_ubuntu\_v3.tar创建镜像，命名为runoob/ubuntu:v4  
[root@localhost ~]# docker import my\_ubuntu\_v3.tar runoob/ubuntu:v4   
sha256:63ce4a6d6bc3fabb95dbd6c561404a309b7bdfc4e21c1d59fe9fe4299cbfea39  
[root@localhost ~]#docker images runoob/ubuntu:v4  
REPOSITORY TAG IMAGE ID CREATED SIZE  
runoob/ubuntu v4 63ce4a6d6bc3 20 seconds ago 142.1 MB

**info**  Display system-wide information #显示docker的系统信息

语法：Usage: docker images [OPTIONS] [REPOSITORY[:TAG]]  
实例：显示本机docker系统信息  
[root@localhost ~]# docker info  
Containers: 3 --当前有3个容器  
Images: 298   
Storage Driver: devicemapper  
Pool Name: docker-253:0-34402623-pool  
Pool Blocksize: 65.54 kB  
Backing Filesystem: xfs  
Data file: /dev/loop0  
Metadata file: /dev/loop1  
Data Space Used: 8.677 GB --对应的是下面Data loop file大小  
Data Space Total: 107.4 GB  
Data Space Available: 5.737 GB  
Metadata Space Used: 13.4 MB --对应的是下面Metadata loop file大小  
Metadata Space Total: 2.147 GB  
Metadata Space Available: 2.134 GB  
Udev Sync Supported: true  
Deferred Removal Enabled: false  
Data loop file: /var/lib/docker/devicemapper/devicemapper/data  
Metadata loop file: /var/lib/docker/devicemapper/devicemapper/metadata  
Library Version: 1.02.93-RHEL7 (2015-01-28)  
Execution Driver: native-0.2  
Logging Driver: json-file  
Kernel Version: 3.10.0-229.el7.x86\_64  
Operating System: CentOS Linux 7 (Core)  
CPUs: 2  
Total Memory: 979.7 MiB  
Name: localhost.localdomain  
ID: TFVB:BXGQ:VVOC:K2DJ:LECE:2HNK:23B2:LEVF:P3IQ:L7D5:NG2V:UKNL  
WARNING: bridge-nf-call-iptables is disabled  
WARNING: bridge-nf-call-ip6tables is disabled

**inspect**  Return low-level information on a container, image or task #用于查看容器的配置信息，包含容器名、环境变量、运行命令、主机配置、网络配置和数据卷配置等。

语法：Usage: docker inspect [OPTIONS] CONTAINER|IMAGE|TASK [CONTAINER|IMAGE|TASK...]  
实例：获取镜像mysql:5.6的元信息。  
[root@localhost ~]# docker inspect mysql:5.6  
[  
 {  
 "Id": "sha256:2c0964ec182ae9a045f866bbc2553087f6e42bfc16074a74fb820af235f070ec",  
 "RepoTags": [  
 "mysql:5.6"  
 ],  
 "RepoDigests": [],  
 "Parent": "",  
 "Comment": "",  
 "Created": "2016-05-24T04:01:41.168371815Z",  
 "Container": "e0924bc460ff97787f34610115e9363e6363b30b8efa406e28eb495ab199ca54",  
 "ContainerConfig": {  
 "Hostname": "b0cf605c7757",  
 "Domainname": "",  
 "User": "",  
 "AttachStdin": false,  
 "AttachStdout": false,  
 "AttachStderr": false,  
 "ExposedPorts": {  
 "3306/tcp": {}  
 },  
...  
获取正在运行的容器mymysql的 IP。  
[root@localhost ~]# docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mymysql  
172.17.0.3

**kill**  Kill one or more running containers #强制终止容器

语法：Usage: docker kill [OPTIONS] CONTAINER [CONTAINER...]  
实例：杀掉运行中的容器mynginx  
[root@localhost ~]# docker kill -s KILL mynginx  
mynginx

**load**  Load an image from a tar archive or STDIN #将打包的镜像导入

语法：Usage: docker load [OPTIONS]  
实例：导入一个打包好的的busybox镜像  
[root@localhost ~]#docker load < busybox.tar  
[root@localhost ~]#docker images  
REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE  
busybox latest 769b9341d9377 weeks ago 2.489 MB

**login**  Log in to a Docker registry. #登录到docker 镜像仓库，默认为dockerhub

语法：Usage: docker login [OPTIONS] [SERVER]  
实例：登陆到Docker Hub  
[root@localhost ~]# docker login -u 用户名 -p 密码

**logout**  Log out from a Docker registry. #退出登录

语法：Usage: docker logout [SERVER]  
实例：退出docker hub  
[root@localhost ~]# docker logout  
Remove login credentials for https://index.docker.io/v1/

**logs**  Fetch the logs of a container #用于查看容器的日志，它将输出到标准输出的数据作为日志输出到docker logs命令的终端上。常用于后台型容器

语法：Usage: docker logs [OPTIONS] CONTAINER  
实例：跟踪查看容器mynginx的日志输出。  
[root@localhost ~]# docker logs -f mynginx  
192.168.239.1 - - [10/Jul/2016:16:53:33 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.93 Safari/537.36" "-"  
2016/07/10 16:53:33 [error] 5#5: \*1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 192.168.239.1, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "192.168.239.130", referrer: "http://192.168.239.130/"  
192.168.239.1 - - [10/Jul/2016:16:53:33 +0000] "GET /favicon.ico HTTP/1.1" 404 571 "http://192.168.239.130/" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.93 Safari/537.36" "-"  
192.168.239.1 - - [10/Jul/2016:16:53:59 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.93 Safari/537.36" "-"  
...  
查看容器mynginx从2016年7月1日后的最新10条日志。  
[root@localhost ~]# docker logs --since="2016-07-01" --tail=10 mynginx

**network** Manage Docker networks #管理docker网络

语法：Usage: docker network COMMAND  
实例：用于动态的将容器添加进一个已有网络/将容器从网络中移除。  
[root@localhost ~]#docker network connect/docker network disconnect  
创建docker网卡  
[root@localhost ~]#docker network create -d overlay --ip-range=192.168.2.0/24 --gateway=192.168.2.1 --subnet=192.168.2.0/24 multihost2  
显示网卡  
[root@localhost ~]#docker network ls  
删除docker网卡  
[root@localhost ~]#docker network rm  
查看docker网卡的相关详细信息  
[root@localhost ~]#docker network inspect

**node** Manage Docker Swarm nodes #管理docker swarm节点

语法：Usage: docker node COMMAND

**pause**  Pause all processes within one or more containers #暂停容器中所有的进程。

语法：Usage: docker pause CONTAINER [CONTAINER...]  
实例：暂停数据库容器db01提供服务。  
[root@localhost ~]#docker pause db01

**port**  List port mappings or a specific mapping for the container #输出容器与宿主机端口映射的信息

语法：Usage: docker port CONTAINER [PRIVATE\_PORT[/PROTO]]  
实例：查看容器mynginx的端口映射情况。  
[root@localhost ~]# docker port mymysql  
3306/tcp -> 0.0.0.0:3306

**ps**  List containers #列出所有容器，默认只列出当前正在运行的容器，-a参数可以查看所有状态的容器

语法：Usage: docker ps [OPTIONS]  
实例：列出所有在运行的容器信息。  
[root@localhost ~]# docker ps  
CONTAINER ID IMAGE COMMAND ... PORTS NAMES  
09b93464c2f7 nginx:latest "nginx -g 'daemon off" ... 80/tcp, 443/tcp myrunoob  
96f7f14e99ab mysql:5.6 "docker-entrypoint.sh" ... 0.0.0.0:3306->3306/tcp mymysql  
列出最近创建的5个容器信息。  
[root@localhost ~]# docker ps -n 5  
CONTAINER ID IMAGE COMMAND CREATED   
09b93464c2f7 nginx:latest "nginx -g 'daemon off" 2 days ago ...   
b8573233d675 nginx:latest "/bin/bash" 2 days ago ...   
b1a0703e41e7 nginx:latest "nginx -g 'daemon off" 2 days ago ...   
f46fb1dec520 5c6e1090e771 "/bin/sh -c 'set -x \t" 2 days ago ...   
a63b4a5597de 860c279d2fec "bash" 2 days ago ...  
列出所有创建的容器ID。  
[root@localhost ~]# docker ps -a -q  
09b93464c2f7  
b8573233d675  
b1a0703e41e7  
f46fb1dec520  
a63b4a5597de  
6a4aa42e947b  
de7bb36e7968  
43a432b73776  
664a8ab1a585  
ba52eb632bbd  
...

**pull**  Pull an image or a repository from a registry #从仓库中下载一个镜像，默认为docker hub上

语法：Usage: docker pull [OPTIONS] NAME[:TAG|@DIGEST]  
实例：从Docker Hub下载java最新版镜像。  
[root@localhost ~]# docker pull java  
从Docker Hub下载REPOSITORY为java的所有镜像。  
[root@localhost ~]# docker pull -a java

**push**  Push an image or a repository to a registry #上传镜像到仓库，默认为docker hub上，需要用户名和密码

语法：Usage: docker push [OPTIONS] NAME[:TAG]  
实例：上传本地镜像myapache:v1到镜像仓库中。  
要先用docker login登录上，不然会报以下错误  
[root@localhost ~]# docker push myapache:v1  
The push refers to a repository [docker.io/ivictor/centos] (len: 1)  
unauthorized: access to the requested resource is not authorized  
[root@localhost ~]# docker push myapache:v1

**rename**  Rename a container #更改容器的名称

语法：Usage: docker rename CONTAINER NEW\_NAME

**start/stop/restart  #启动、停止、重启一个容器**

语法：Usage:docker start [OPTIONS] CONTAINER [CONTAINER...]  
 Usage:docker stop [OPTIONS] CONTAINER [CONTAINER...]  
 Usage:docker restart [OPTIONS] CONTAINER [CONTAINER...]  
实例：启动已被停止的容器myrunoob  
[root@localhost ~]#docker start myrunoob  
停止运行中的容器myrunoob  
[root@localhost ~]#docker stop myrunoob  
重启容器myrunoob  
[root@localhost ~]#docker restart myrunoob

**rm**  Remove one or more containers #删除一个或多个容器，默认只能删除非运行状态的容器，-f参数可以强制删除

语法：Usage: docker rm [OPTIONS] CONTAINER [CONTAINER...]  
实例：强制删除容器db01、db02  
[root@localhost ~]# docker rm -f db01、db02  
移除容器nginx01对容器db01的连接，连接名db  
[root@localhost ~]# docker rm -l db   
删除容器nginx01,并删除容器挂载的数据卷  
[root@localhost ~]# docker rm -v nginx01

**rmi**  Remove one or more images #删除一个或多个镜像

Usage: docker rmi [OPTIONS] IMAGE [IMAGE...]

**run**  Run a command in a new container #启动一个容器，等同于docker create && docker start命令

语法：Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]  
实例：使用docker镜像nginx:latest以后台模式启动一个容器,并将容器命名为mynginx。  
[root@localhost ~]# docker run --name mynginx -d nginx:latest  
使用镜像nginx:latest以后台模式启动一个容器,并将容器的80端口映射到主机随机端口。  
[root@localhost ~]#docker run -P -d nginx:latest  
使用镜像nginx:latest以后台模式启动一个容器,将容器的80端口映射到主机的80端口,主机的目录/data映射到容器的/data。  
[root@localhost ~]#docker run -p 80:80 -v /data:/data -d nginx:latest  
使用镜像nginx:latest以交互模式启动一个容器,在容器内执行/bin/bash命令。  
[root@localhost ~]# docker run -it nginx:latest /bin/bash  
root@b8573233d675:/#

**save**  Save one or more images to a tar archive (streamed to STDOUT by default) #将指定镜像保存成 tar 归档文件

语法：Usage: docker save [OPTIONS] IMAGE [IMAGE...]  
实例：将镜像runoob/ubuntu:v3 生成my\_ubuntu\_v3.tar文档  
[root@localhost ~]# docker save -o my\_ubuntu\_v3.tar runoob/ubuntu:v3  
[root@localhost ~]# ll my\_ubuntu\_v3.tar  
-rw------- 1 runoob runoob 142102016 Jul 11 01:37 my\_ubuntu\_v3.ta

**search**  Search the Docker Hub for images #在docker hub上搜索镜像

语法：Usage: docker search [OPTIONS] TERM  
实例：从Docker Hub查找所有镜像名包含java，并且收藏数大于10的镜像  
[root@localhost ~]# docker search -s 10 java  
NAME DESCRIPTION STARS OFFICIAL AUTOMATED  
java Java is a concurrent, class-based... 1037 [OK]   
anapsix/alpine-java Oracle Java 8 (and 7) with GLIBC ... 115 [OK]  
develar/java 46 [OK]  
isuper/java-oracle This repository contains all java... 38 [OK]  
lwieske/java-8 Oracle Java 8 Container - Full + ... 27 [OK]  
nimmis/java-centos This is docker images of CentOS 7... 13 [OK]

**service**  Manage Docker services #管理docker service。Docker Service是一种声明式的、可扩展的、负载均衡的应用。Docker Service是面向用户的应用

语法：Usage: docker service COMMAND  
实例：创建一个拥有5个副本的Docker Service  
[root@localhost ~]# docker service create --name myService -p 8080:8080 --replicas 5 an\_image

**stats**  Display a live stream of container(s) resource usage statistics #动态显示容器的资源消耗情况，包括：CPU、内存、网络I/O

语法：Usage: docker stats [OPTIONS] [CONTAINER...]  
实例：显示ID为af93e92b036a的状态信息  
[root@localhost ~]# docker stats af93e92b036a  
CONTAINER CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS  
af93e92b036a 0.02% 3.535 MiB / 1.781 GiB 0.19% 44.24 kB / 42.14 kB 17.43 MB / 1.683 MB 6  
CONTAINER CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS  
af93e92b036a 0.02% 3.535 MiB / 1.781 GiB 0.19% 44.24 kB / 42.14 kB 17.43 MB / 1.683 MB 6  
CONTAINER CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS  
af93e92b036a 0.00% 3.535 MiB / 1.781 GiB 0.19% 44.24 kB / 42.14 kB 17.43 MB / 1.683 MB 6

**swarm**  Manage Docker Swarm #管理docker swarm

语法：Usage: docker swarm COMMAND  
实例：初始化swarm manager并制定网卡地址  
docker swarm init --advertise-addr 192.168.10.117  
强制删除集群，如果是manager，需要加–force  
docker swarm leave --force  
docker node rm docker-118  
查看swarm worker的连接令牌  
docker swarm join-token worker  
查看swarm manager的连接令牌  
docker swarm join-token manager  
使旧令牌无效并生成新令牌  
docker swarm join-token --rotate  
加入docker swarm集群  
docker swarm join --token SWMTKN-1-5d2ipwo8jqdsiesv6ixze20w2toclys76gyu4zdoiaf038voxj-8sbxe79rx5qt14ol14gxxa3wf 192.168.10.117:2377  
查看集群中的节点  
docker node ls  
查看集群中节点信息  
docker node inspect docker-117 --pretty  
调度程序可以将任务分配给节点  
docker node update --availability active docker-118  
调度程序不向节点分配新任务，但是现有任务仍然保持运行  
docker node update --availability pause docker-118  
调度程序不会将新任务分配给节点。调度程序关闭任何现有任务并在可用节点上安排它们  
docker node update --availability drain docker-118  
添加节点标签  
docker node update --label-add label1 --label-add bar=label2 docker-117  
删除节点标签  
docker node update --label-rm label1 docker-117  
将节点升级为manager  
docker node promote docker-118  
将节点降级为worker  
docker node demote docker-118  
查看服务列表  
docker service ls  
查看服务的具体信息  
docker service ps redis  
创建一个不定义name，不定义replicas的服务  
docker service create nginx  
创建一个指定name的服务  
docker service create --name my\_web nginx  
创建一个指定name、run cmd的服务  
docker service create --name helloworld alping ping docker.com  
创建一个指定name、version、run cmd的服务  
docker service create --name helloworld alping:3.6 ping docker.com  
创建一个指定name、port、replicas的服务  
docker service create --name my\_web --replicas 3 -p 80:80 nginx  
为指定的服务更新一个端口  
docker service update --publish-add 80:80 my\_web  
为指定的服务删除一个端口  
docker service update --publish-rm 80:80 my\_web  
将redis:3.0.6更新至redis:3.0.7  
docker service update --image redis:3.0.7 redis  
配置运行环境，指定工作目录及环境变量  
docker service create --name helloworld --env MYVAR=myvalue --workdir /tmp --user my\_user alping ping docker.com  
创建一个helloworld的服务  
docker service create --name helloworld alpine ping docker.com  
更新helloworld服务的运行命令  
docker service update --args “ping www.baidu.com” helloworld  
删除一个服务  
docker service rm my\_web  
在每个群组节点上运行web服务  
docker service create --name tomcat --mode global --publish mode=host,target=8080,published=8080 tomcat:latest  
创建一个overlay网络  
docker network create --driver overlay my\_network  
docker network create --driver overlay --subnet 10.10.10.0/24 --gateway 10.10.10.1 my-network  
创建服务并将网络添加至该服务  
docker service create --name test --replicas 3 --network my-network redis  
删除群组网络  
docker service update --network-rm my-network test  
更新群组网络  
docker service update --network-add my\_network test  
创建群组并配置cpu和内存  
docker service create --name my\_nginx --reserve-cpu 2 --reserve-memory 512m --replicas 3 nginx  
更改所分配的cpu和内存  
docker service update --reserve-cpu 1 --reserve-memory 256m my\_nginx  
指定每次更新的容器数量  
--update-parallelism  
指定容器更新的间隔  
--update-delay  
定义容器启动后监控失败的持续时间  
--update-monitor   
定义容器失败的百分比  
--update-max-failure-ratio  
定义容器启动失败之后所执行的动作  
--update-failure-action  
创建一个服务并运行3个副本，同步延迟10秒，10%任务失败则暂停  
docker service create --name mysql\_5\_6\_36 --replicas 3 --update-delay 10s --update-parallelism 1 --update-monitor 30s --update-failure-action pause --update-max-failure-ratio 0.1 -e MYSQL\_ROOT\_PASSWORD=123456 mysql:5.6.36  
回滚至之前版本  
docker service update --rollback mysql  
自动回滚   
如果服务部署失败，则每次回滚2个任务，监控20秒，回滚可接受失败率20%  
docker service create --name redis --replicas 6 --rollback-parallelism 2 --rollback-monitor 20s --rollback-max-failure-ratio .2 redis:latest  
创建服务并将目录挂在至container中  
docker service create --name mysql --publish 3306:3306 --mount type=bind,src=/data/mysql,dst=/var/lib/mysql --replicas 3 -e MYSQL\_ROOT\_PASSWORD=123456 mysql:5.6.36  
Bind带来的风险   
1、绑定的主机路径必须存在于每个集群节点上，否则会有问题   
2、调度程序可能会在任何时候重新安排运行服务容器，如果目标节点主机变得不健康或无法访问   
3、主机绑定数据不可移植，当你绑定安装时，不能保证你的应用程序开发方式与生产中的运行方式相同  
添加swarm配置  
echo "this is a mysql config" | docker config create mysql -  
查看配置  
docker config ls  
查看配置详细信息  
docker config inspect mysql  
删除配置  
docker config rm mysql  
添加配置  
docker service update --config-add mysql mysql  
删除配置  
docker service update --config-rm mysql mysql  
添加配置  
docker config create homepage index.html  
启动容器的同时添加配置  
docker service create --name nginx --publish 80:80 --replicas 3 --config src=homepage,target=/usr/share/nginx/html/index.html nginx

**tag**  Tag an image into a repository #标记本地镜像，将其归入某一个仓库

语法：Usage: docker tag IMAGE[:TAG] IMAGE[:TAG]  
实例：将镜像ubuntu:15.10标记为 runoob/ubuntu:v3 镜像。  
[root@localhost ~]# docker tag ubuntu:15.10 runoob/ubuntu:v3  
[root@localhost ~]#  docker images runoob/ubuntu:v3  
REPOSITORY TAG IMAGE ID CREATED SIZE  
runoob/ubuntu v3 4e3b13c8a266 3 months ago 136.3 MB

**top**  Display the running processes of a container #查看容器中运行的进程信息，支持 ps 命令参数

语法：Usage: docker top CONTAINER [ps OPTIONS]  
实例：查看容器mymysql的进程信息。  
[root@localhost ~]# docker top mymysql  
UID PID PPID C STIME TTY TIME CMD  
999 40347 40331 18 00:58 ? 00:00:02 mysqld  
查看所有运行容器的进程信息。  
[root@localhost ~]#for i in `docker ps |grep Up|awk '{print $1}'`;do echo \ &&docker top $i; done

**unpause**  Unpause all processes within one or more containers #恢复容器内暂停的进程

语法：Usage: docker unpause CONTAINER [CONTAINER...]  
实例：恢复数据库容器db01提供服务。  
[root@localhost ~]#docker unpause db01

**update**  Update configuration of one or more containers #动态的更新一个或多个容器的配置

语法：Usage: docker update CONTAINER [CONTAINER...]  
实例：更新一个容器为cpu-shares=512  
[root@localhost ~]# docker update --cpu-shares 512 abebf7571666  
更新容器的cpu-shares和memory  
[root@localhost ~]# docker update --cpu-shares 512 -m 300M abebf7571666 hopeful\_morse  
更新容器restart策略  
[root@localhost ~]#docker update --restart=on-failure:3 abebf7571666 hopeful\_morse

**version**  Show the Docker version information #显示docker的版本信息

语法：Usage: docker version [OPTIONS]  
实例：显示本机docker 版本信息  
[root@localhost ~]# docker version  
Client:  
 Version: 1.12.6  
 API version: 1.24  
 Package version: docker-1.12.6-61.git85d7426.el7.centos.x86\_64  
 Go version: go1.8.3  
 Git commit: 85d7426/1.12.6  
 Built: Tue Oct 24 15:40:21 2017  
 OS/Arch: linux/amd64  
Server:  
 Version: 1.12.6  
 API version: 1.24  
 Package version: docker-1.12.6-61.git85d7426.el7.centos.x86\_64  
 Go version: go1.8.3  
 Git commit: 85d7426/1.12.6  
 Built: Tue Oct 24 15:40:21 2017  
 OS/Arch: linux/amd64

**volume** Manage Docker volumes #管理docker数据卷

语法：Usage: docker volume COMMAND  
实例：启动一个Volume\_Container容器，包含两个数据卷  
[root@localhost ~]# docker run -v /var/volume1 -v /var/volume2 -name Volume\_Container ubuntu14.04 linux\_command  
创建App\_Container容器，挂载Volume\_Container容器中的数据卷  
[root@localhost ~]# docker run -t -i -rm -volumes-from Volume\_Container -name App\_Container ubuntu14.04 linux\_command  
或者再创建一个容器，挂载App\_Container中从[root@localhost ~]# Volume\_Container挂载的数据卷  
docker run -t -i -rm -volumes-from App\_Container -name LastApp\_Container ubuntu14.04 linux\_command

**wait**  Block until a container stops, then print its exit code #捕捉容器停止时的退出码,执行此命令后，该命令会“hang”在当前终端，直到容器停止，此时，会打印出容器的退出码。

语法：Usage: docker wait CONTAINER [CONTAINER...]