docker 命令官方文档

https://docs.docker.com/reference/

Docker 环境信息命令

docker info

```
docker info : 显示 Docker 系统信息，包括镜像和容器数。
```

```
docker info [OPTIONS]
```

docker version

```
docker version :显示 Docker 版本信息。
```

```
docker version [OPTIONS]

OPTIONS说明：
-f :指定返回值的模板文件。
```

系统日志信息常用命令

docker events

```
docker events : 从服务器获取实时事件
```

```
docker events [OPTIONS]

OPTIONS说明：

-f ：根据条件过滤事件；

--since ：从指定的时间戳后显示所有事件;

--until ：流水时间显示到指定的时间为止；
```

```
//第一个终端执行
docker events

//第二个终端操作容器
docker start/stop/restart

//查看第一个终端输出
```

docker logs

```
docker logs : 获取容器的日志
```

```
docker logs [OPTIONS] CONTAINER

OPTIONS说明：

-f : 跟踪日志输出

--since :显示某个开始时间的所有日志

-t : 显示时间戳

--tail :仅列出最新N条容器日志
```

```
docker logs -f mynginx
```

docker history

```
docker history : 查看指定镜像的创建历史。
```

```
docker history [OPTIONS] IMAGE
OPTIONS说明：

-H :以可读的格式打印镜像大小和日期，默认为true；

--no-trunc :显示完整的提交记录；

-q :仅列出提交记录ID
```

```
docker history mynginx:v1
```

容器的生命周期管理命令

docker create

```
docker create : 创建一个新的容器但不启动它
```

```
语法同docker run
```

docker run

```
docker run :创建一个新的容器并运行一个命令
```

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

常用选项

1. --add-host：容器中hosts文件添加 host:ip 映射记录

```
docker run --rm -it --add-host db-static:86.75.30.9 ubuntu cat /etc/hosts
```

2. -a, --attach：附加到 STDIN, STDOUT 或 STDERR

```
docker run --rm -a stdin -a stdout -i -t ubuntu /bin/bash
```

3. --cidfile：将容器id写入到指定文件

```
docker run --rm --cidfile /tmp/idfile ubuntu
```

4. -d, --detach：后台运行容器并打印容器id
5. --detach-keys：指定将容器切回后台的按键，默认：CTRL-p+CTRL-q
6. --disable-content-trust：跳过镜像验证，默认为TRUE
7. --domainname：设置容器网络服务域名
8. --entrypoint：覆盖镜像默认程序入口

```
docker run --rm -it --entrypoint /bin/bash nginx
```

9. -e, --env：设置环境变量
10. --env-file：从文件中读取环境变量
11. --expose：暴露一个或多个端口
12. --group-add：为容器用户添加更多用户组

```
docker run --rm --group-add audio --group-add nogroup --group-add 777 ubuntu id
```

13. -h, --hostname string：设置主机名
14. --init 在容器内运行init，转发信号并捕获进程

```
# ps 对比容器的元祖进程
docker run --rm -it  ubuntu bash
docker run --rm -it --init  ubuntu bash
```

15. -i, --interactive：保持标准输入设备处于打开状态
16. -l, --label list：设置容器元数据labels
17. --label-file list：从文件中读取以换行作为分隔符的label
18. --link：添加一个其他容器的链接，及修改当容器的/etc/hosts文件
19. --log-driver：容器日志驱动，none、local、json-file、syslog、journald、gelf、fluentd、awslogs、splunk、etwlogs、gcplogs、logentries
20. --log-opt：日志驱动选项
21. --mount：将文件系统挂载到容器，与 -v 类似

```
# 挂载volume
docker run -t -i --rm --mount type=volume,target=/data ubuntu bash
# 创建目录
mkdir /tmp/data
# 修改目录所有权，仅当使用userns的情况下需要
sudo chown 165536:root /tmp/data
# bind 磁盘已存在目录
docker run -t -i --rm --mount type=bind,src=/tmp/data,dst=/data ubuntu bash
```

22. --name：指定容器名称
23. --privileged：授予容器超级权限
24. -p, --publish：发布容器端口都主机端口
25. -P, --publish-all：将所有暴露的端口发布到主机随机端口
26. --pull：运行前拉取图像，（always|missing|never），默认 missing
27. --pids-limit：限制容器中pid个数
28. --read-only ：将容器跟文件系统装载为只读
29. --restart：当容器退出时的重启策略，默认为no

no：当容器退出时，不要自动重新启动
on-failure[:max-retries]：仅当容器以非零退出状态退出时才重新启动。
always：无论退出状态如何，始终重新启动容器。无论容器的当前状态如何，容器也将始终在守护程序启动时启动
unless-stopped：论退出状态如何，始终重新启动容器，包括守护程序启动时，除非容器在Docker守护程序停止之前处于停止状态。

27. --rm：当容器退出时，自动删除容器
28. --sig-proxy：将接收到的信号代理到进程（默认为true）
29. --stop-signal：停止容器的信号（默认为“SIGTERM”）
30. --stop-timeout：容器停止超时时长，单位s
31. --tmpfs：装载tmpfs目录

```
docker run -dit --rm --tmpfs /run1:rw,noexec,nosuid,size=65536k ubuntu bash
```

32. -t, --tty：分配一个伪终端设备
33. -u, --user： 用户名或用户ID (格式: <name|uid>[:<group|gid>])
34. -v, --volume ： 绑定数据卷
35. --volume-driver：容器的可选卷驱动程序
36. --volumes-from：从指定的容器装载数据卷
37. -w, --workdir：容器内的工作目录

系统

1. --cap-add：添加linux功能
2. --cap-drop：删除linux功能

```
docker run --cap-add=ALL --cap-drop=MKNOD ...
docker run --cap-add=SYS_ADMIN ...
docker run --cap-add=CAP_SYS_ADMIN ...
```

3. --isolation：指定容器的隔离技术

linux系统：只支持default，即linux命名空间隔离技术
windows系统：
default:表示使用docker守护进程配置的选项，否则默认使用process选项

```
process:命名空间隔离
hyperv:基于Hyper-V 管理程序基于分区的隔离
```

4. --platform：设置平台，如果服务器支持多个平台
5. --runtime：用于此容器使用的运行时
6. --security-opt：通过指定--security opt标志来覆盖每个容器的默认标签方案
7. --shm-size：设置/dev/shm设备的大小，/dev/shm是一个基于内存的临时文件系统
8. --storage-opt：容器的存储驱动程序选项
9. --sysctl：设置系统参数，默认 map[]
10. --ulimit：设置Ulimit选项，默认[]

网络

1. --ip：ip4地址
2. --ip6：ip6地址
3. --dns 设置自定义dns服务器，将修改容器中 /etc/resolv.conf 文件
4. --dns-option：设置DNS选项，将修改容器中 /etc/resolv.conf 文件
5. --dns-search：设置自定义DNS搜索域名，将修改容器中 /etc/resolv.conf 文件
6. --link-local-ip：设置一个或多个容器的以太网设备的链路本地IPv4/IPv6地址
7. --mac-address：容器Mac地址
8. --network：将容器加入到指定网络
9. --network-alias：为容器添加网络范围的别名

健康检查

1. --health-cmd：健康检查命令
2. --health-interval：健康检查频率（ms|s|m|h）,默认0s
3. --health-retries：健康检查重试次数，连续失败指定次数则判断为不健康
4. --health-start-period：设置启动容器的初始化时间，在此期间的健康检查不作为参考依据，（ms|s|m|h）,默认0s
5. --health-timeout：每一次检查超时时间（ms|s|m|h）,默认0s
6. --no-healthcheck：禁用健康检查

```
docker run --rm  --health-cmd "curl http://localhost" --health-interval 2s --
health-retries 5 --health-start-period 10s --health-timeout 1s nginx
```

命名空间选项

1. --cgroup-parent：容器的可选父cgroup
2. --cgroupns：要使用的Cgroup命名空间(host|private)

host：在Docker主机的cgroup命名空间中运行容器
private：在容器的私有cgroup命名空间中运行容器
""：使用配置的cgroup命名空间

3. --pid：pid 命名空间使用，默认使用命名空间隔离，指定host则使用主机pid命名空间
4. --userns：用户与组命名空间，默认为主机模式，在daemon.json中配置后，默认隔离，可通过指定host为主机模式

5. --uts：uts命名空间，默认隔离，可指定host为主机模式

cgroup资源限制选项

CPU

1. --cpu-period：限制（完全公平调度程序）CPU分配周期
2. --cpu-quota：限制（完全公平调度程序）CPU配额

```
docker run --rm -dit --cpu-period=50000 --cpu-quota=25000 ubuntu /bin/bash
```

3. --cpu-rt-period：限制CPU实时周期（微秒）
4. --cpu-rt-runtime：限制CPU实时运行时间（微秒）

```
docker run --rm -dit --cpu-rt-period=50000 --cpu-rt-runtime=25000 ubuntu
/bin/bash
```

7. -c, --cpu-shares：CPU相对权重
8. --cpus：指定CPU配额，例如：--cpus=0.5，表示可以使用0.5个CPU的配额

CPUset

1. --cpuset-cpus：指定容器在哪个或哪几个CPU上运行

```
docker run --rm -dit --cpuset-cpus="1,3" ubuntu /bin/bash
docker run --rm -it --cpuset-cpus="0-2" ubuntu /bin/bash
```

2. --cpuset-mems：指定容器使用的内存节点

```
docker run --rm -dit --cpuset-mems="1,3" ubuntu /bin/bash
docker run --rm -dit --cpuset-mems="0-2" ubuntu /bin/bash
```

device

1. --device：添加主机设备到容器，默认权限：read, write, mknod，可通过:rwm 覆盖默认权限

```
docker run --rm -dit --device=/dev/sda:/dev/sda --device=/dev/snd:/dev/snd:r
ubuntu /bin/bash
```

2. --device-cgroup-rule：将规则添加到cgroup允许的设备列表
3. --device-read-bps：限制设备的读取速率（字节/秒）

```
docker run --rm -dit --device-read-bps /dev/sda:1mb ubuntu
```

4. --device-read-iops：限制设备每秒读取次数（io/秒）

```
 docker run --rm -dti --device-read-iops /dev/sda:1000 ubuntu
```

5. --device-write-bps：限制设备的写入速率（字节/秒）

```
 docker run --rm -dit --device-write-bps /dev/sda:1mb ubuntu
```

6. --device-write-iops：限制设备每秒写入次数（io/秒）

```
 docker run --rm -dti --device-write-iops /dev/sda:1000 ubuntu
```

7. --gpus：要添加到容器的GPU设备（“all”传递所有GPU）

memory

ubuntu系统开启内核支持

1. 修改系统文件/etc/default/grub

```
GRUB_CMDLINE_LINUX="cgroup_enable=memory swapaccount=1"
```

2. 更新并重启机器 update-grub; reboot;

选项

1. -m, --memory：内存限制，硬限制
2. --memory-reservation：内存限制，软限制

```
docker run --rm -dit -m 500M --memory-reservation 200M ubuntu /bin/bash
```

3. --memory-swap：交换区限制，等于内存限制加swap限制，-1表示无限制

```
 docker run --rm -dit -m 300M --memory-swap 1G ubuntu /bin/bash
```

4. --memory-swappiness：调整容器内存交换，取值 0 ~ 100 ，默认为-1。为0表示尽可能的使用物理内存，为100表示尽可能的使用交换区。产生交换的场景：1.系统需要的内存大于可用内存 2.应用启动之初需要大量内存，启动之后不再使用的内存将被交换。

```
docker run --rm -dit --memory-swappiness=0 ubuntu /bin/bash
```

5. --oom-kill-disable：禁用内存超限，杀死进程功能

```
docker run --rm -dit -m 100M --oom-kill-disable ubuntu /bin/bash
```

6. --oom-score-adj：调整容器进程超限被杀的优先级，分数越高被杀的优先级越高，取值范围-1000 ~ 1000

```
docker run --rm -dit -m 100M --oom-score-adj 100 ubuntu /bin/bash
```

docker start/stop/restart

```
docker start :启动一个或多个已经被停止的容器

docker stop :停止一个运行中的容器

docker restart :重启容器
```

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

docker stop [OPTIONS] CONTAINER [CONTAINER...]

docker restart [OPTIONS] CONTAINER [CONTAINER...]
```

```
docker restart mynginx
docker stop mynginx
docker start mynginx
```

docker kill

```
docker kill :杀掉一个运行中的容器
```

```
docker kill [OPTIONS] CONTAINER [CONTAINER...]

OPTIONS说明：
-s :向容器发送一个信号
```

```
docker kill -s KILL  mynginx
docker kill -s TERM  mynginx
```

如果容器终止时的状态对用户而言很重要，用户可能会想要了解docker kill 和docker stop之间的区别。
docker kill 的行为和标准的kill 命令行程序并不相同。kill 程序的默认工作方式是向指定的进程发送TERM信号（即信号值为15）。这个信号表示程序应该被终止，但是不要强迫程序终止。当这个信号被处理时，大多数程序将执行某种清理工作，但是该程序也可以执行其他操作，包括忽略该信号。
而docker kill 对正在运行的程序使用的是KILL信号，这使得该进程没办法处理终止过程。这就意味着一些诸如包含运行进程ID之类的文件可能会残留在文件系统中。根据应该用程序管理状态的能力，如果再次启动容器，这可能会也可能不会造成问题。
docker stop 命令则像kill 命令那样工作，发送的是一个TERM型号。

命令	默认信号量	默认信号量的值
kill	TERM	15
docker kill	KILL	9
docker stop	TERM	15

docker rm

```
docker rm ： 删除一个或多个容器。
```

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

OPTIONS说明：

-f :通过 SIGKILL 信号强制删除一个运行中的容器。

-l :移除容器间的网络连接，而非容器本身。

-v :删除与容器关联的卷。
```

docker pause/unpause

```
docker pause :暂停容器中所有的进程。
docker unpause :恢复容器中所有的进程。
```

```
docker pause CONTAINER [CONTAINER...]
docker unpause CONTAINER [CONTAINER...]
```

容器运维操作命令

docker exec

```
docker exec ： 在运行的容器中执行命令
```

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

OPTIONS说明：

-d :分离模式: 在后台运行

-i :即使没有附加也保持STDIN 打开

-t :分配一个伪终端
```

```
docker exec -i -t  mynginx /bin/bash
// 通过exec 执行命令
docker exec mynginx ls
```

docker ps

```
docker ps : 列出容器
```

```
docker ps [OPTIONS]

OPTIONS说明：

-a :显示所有的容器，包括未运行的。

-f :根据条件过滤显示的内容。
```

```
--format :指定返回值的模板文件。

-l :显示最近创建的容器。

-n :列出最近创建的n个容器。

--no-trunc :不截断输出。

-q :静默模式，只显示容器编号。

-s :显示总的文件大小。
```

```
docker ps
```

docker inspect

```
docker inspect : 获取容器/镜像的元数据。
```

```
docker inspect [OPTIONS] NAME|ID [NAME|ID...]

OPTIONS说明：

-f :指定返回值的模板文件。

-s :显示总的文件大小。

--type :为指定类型返回JSON
```

```
// 查看容器的示例id
docker inspect -f  '{{.Id}}' <id>
// 检查镜像或者容器的参数，默认返回 JSON 格式
docker inspect <id>
```

docker top

```
docker top :查看容器中运行的进程信息，支持 ps 命令参数
```

```
docker top [OPTIONS] CONTAINER [ps OPTIONS]
```

```
# 查看所有运行容器的进程信息。
for i in  `docker ps |grep Up|awk '{print $1}'`;do echo \ &&docker top $i; done
```

docker attach

```
docker attach :连接到正在运行中的容器
```

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

```
docker attach mynginx
```

docker exec 与 docker attach 的区别：

1. exec 在容器中执行命令，并且可以通过-i -t 创建虚拟终端的方式与容器交互
2. attach 进入容器某个正在执行的命令终端，不能交互操作。但是如果该容器的命令终端是一个可以交互的终端，那么也可以交互

退出attach模式：

1. 运行容器时指定 -i -t 参数，那么attach容器后可通过Ctrl + P + Ctrl + Q 退出attach状态

```
docker run -d -it -P nginx
docker attach <containerID>
```

2. 仅指定-t参数，可以通过Ctrl + C 退出attach状态

```
docker run -d -t -P nginx
docker attach <containerID>
```

3. 不指定-t参数，在attach的时候指定 --sig-proxy=false ，可以通过 Ctrl + C 退出attach状态。--sig-proxy 仅在没有指定-t参数的情况下生效。于--sig-proxy 默认值为true，默认将所有接收信号代理给了容器中的进程，所以Ctrl + C 的时候会将sigint (signal interrupt)终止信号发送到attach进程，从而转发到被attach的进程，导致容器中进程中断，从而退出容器。

```
docker run -d -i -P nginx
docker run -d -P nginx
docker attach --sig-proxy=false  <containerID>
```

docker wait

```
docker wait : 阻塞运行直到容器停止，然后打印出它的退出代码。
```

```
docker wait [OPTIONS] CONTAINER [CONTAINER...]
```

docker export

```
docker export :将文件系统作为一个tar归档文件导出到STDOUT。
```

```
docker export [OPTIONS] CONTAINER
OPTIONS说明：

-o :将输入内容写到文件。
```

```
docker export mynginx -o mynginx.tar
```

```
# 解压归档文件，查看内容
tar vxf mynginx.tar -c mynginx
```

docker port

```
docker port :列出指定的容器的端口映射，或者查找将PRIVATE_PORT NAT到面向公众的端口。
```

```
docker port [OPTIONS] CONTAINER [PRIVATE_PORT[/PROTO]]
```

```
docker port mynginx
```

docker cp

```
docker cp :用于容器与主机之间的数据拷贝
```

```
docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-

docker cp [OPTIONS] SRC_PATH|- CONTAINER:DEST_PATH

OPTIONS说明：

-L :保持源目标中的链接
```

```
# 主机/www/html目录拷贝到容器1881f0cba5bc的/www目录下
docker cp /www/html 1881f0cba5bc:/www/

# 将主机/www/html目录拷贝到容器1881f0cba5bc中，目录重命名为www。
docker cp /www/html 1881f0cba5bc:/www

# 将容器1881f0cba5bc的/www目录拷贝到主机的/tmp目录中
docker cp  1881f0cba5bc:/www /tmp/
```

docker diff

```
docker diff : 检查容器里文件结构的更改。
```

docker diff命令会列出 3 种容器内文件状态变化（A - Add, D - Delete, C - Change ）的列表清单

```
docker diff [OPTIONS] CONTAINER
```

```
docker diff mynginx
```

docker rename

```
docker rename CONTAINER NEW_NAME
```

docker stats

```
docker stats [容器]
```

docker update

```
# 查看帮助，可以看出docker update 是用来修改docker run 指定的运行参数
docker update -h
```

```
docker update --restart always myubuntu3
docker update --restart no myubuntu3
```

镜像管理命令

docker build

```
docker build 命令用于使用 Dockerfile 创建镜像。
```

```
docker build [OPTIONS] PATH | URL | -
OPTIONS说明：

--add-host :向hosts文件中添加自定义 host:ip 映射

--build-arg=[] :设置镜像创建时的变量；

--cache-from :指定镜像用作当前构建的缓存镜像

--cgroup-parent :容器的可选父cgroup

--compress : 使用gzip压缩构建上下文

--cpu-period :限制 CPU CFS周期；

--cpu-quota :限制 CPU CFS配额；

-c,--cpu-shares :设置 cpu 使用权重；

--cpuset-cpus :指定使用的CPU id；

--cpuset-mems :指定使用的内存 id；

--disable-content-trust :跳过镜像校验，默认为true;

-f,--file :指定要使用的Dockerfile路径；

--force-rm :不论构建是否成功，总是删除中间容器，默认false。注意：中间容器；

--iidfile :将镜像ID写入到文件

--isolation :使用容器隔离技术；
    linux系统：只支持default，即linux命名空间隔离技术
    windows系统：
        default:表示使用docker守护进程配置的选项，否则默认使用process选项
    process:命名空间隔离
    hyperv:基于Hyper-V 管理程序基于分区的隔离
```

```
--label=[] :设置镜像使用的元数据；

-m :设置内存最大值；

--memory-swap :设置Swap的最大值为内存+swap，"-1"表示不限swap；

--network: 在构建期间为RUN指令设置网络模式（默认“default”）与docker run指令无关

--no-cache :创建镜像的过程不使用缓存；

--pull :总是尝试去更新镜像的新版本；

--quiet, -q :安静模式，成功后只输出镜像 ID；

--rm :构建成功后，删除中间容器，默认true。注意：中间容器，不是镜像；

--security-opt :安全选项

--shm-size :设置/dev/shm的大小，默认值是64M；/dev/shm 是基于内存的tmpfs文件系统。

--ulimit :Ulimit配置。

--tag, -t : 镜像的名字及标签，通常 name:tag 或者 name 格式；可以在一次构建中为一个镜像设置多个标签。

--target : 设置要生成的目标生成阶段

--ulimit : Ulimit 选项
```

```
docker build  -t nodetodo:v1.0.0 -f Dockerfile .
```

docker images

```
docker images : 列出本地镜像。
```

```
docker images [OPTIONS] [REPOSITORY[:TAG]]

OPTIONS说明：

-a :列出本地所有的镜像（含中间映像层，默认情况下，过滤掉中间映像层）；

--digests :显示镜像的摘要信息；

-f :显示满足条件的镜像；

--format :指定返回值的模板文件；

--no-trunc :显示完整的镜像信息；

-q :只显示镜像ID
```

```
# 查看所有悬空镜像
docker images --filter dangling=true
```

```
# 清除所有悬空镜像
docker image prune
```

docker rmi

```
docker rmi : 删除本地一个或多个镜像。
```

```
docker rmi [OPTIONS] IMAGE [IMAGE...]

OPTIONS说明：

-f :强制删除；

--no-prune :不移除该镜像的过程镜像，默认移除；
```

docker tag

```
docker tag : 标记本地镜像，将其归入某一仓库。
```

```
docker tag [OPTIONS] IMAGE[:TAG] [REGISTRYHOST/][USERNAME/]NAME[:TAG]
```

```
docker tag mynginx:v1 mynginx:1.0
```

docker save

```
docker save : 将指定镜像保存成 tar 归档文件。
```

```
docker save [OPTIONS] IMAGE [IMAGE...]
OPTIONS 说明：

-o :输出到的文件。
```

```
docker save -o mynginx.tar mynginx:v1
```

```
# 解压文件，查看内容
tar vxf mynginx.tar -C mynginx
```

docker load

```
docker load : 导入使用 docker save 命令导出的镜像。
```

```
docker load [OPTIONS]
OPTIONS 说明：

--input , -i : 指定导入的文件，代替 STDIN。

--quiet , -q : 精简输出信息。
```

```
docker load -i mynginx.tar
```

docker import

```
docker import : 从归档文件中创建镜像。
```

```
docker import [OPTIONS] file|URL|- [REPOSITORY[:TAG]]
OPTIONS说明：

-c :应用docker 指令创建镜像；

-m :提交时的说明文字
```

```
docker import  mynginx.tar mynginx:v3
```

docker commit

```
docker commit :从容器创建一个新的镜像。
```

```
docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]
OPTIONS说明：

-a :提交的镜像作者；

-c :使用Dockerfile指令来创建镜像；

-m :提交时的说明文字；

-p :在commit时，将容器暂停
```

```
docker commit -a "nick" -m "mynginx commit" mynginx mynginxtest:1.0
```

镜像仓库命令

docker login/logout

```
docker login : 登陆到一个Docker镜像仓库，如果未指定镜像仓库地址，默认为官方仓库 Docker Hub

docker logout : 登出一个Docker镜像仓库，如果未指定镜像仓库地址，默认为官方仓库 Docker Hub
```

```
docker login [OPTIONS] [SERVER]

docker logout [OPTIONS] [SERVER]
OPTIONS说明：

-u :登陆的用户名

-p :登陆的密码
```

```
docker login -u 用户名 -p 密码

docker logout
```

docker pull

```
docker pull : 从镜像仓库中拉取或者更新指定镜像
```

```
docker pull [OPTIONS] NAME[:TAG|@DIGEST]
OPTIONS说明：

-a :拉取所有 tagged 镜像

--disable-content-trust :忽略镜像的校验,默认开启
```

```
docker pull nginx
```

docker push

```
docker push : 将本地的镜像上传到镜像仓库,要先登陆到镜像仓库
```

```
docker push [OPTIONS] NAME[:TAG]
OPTIONS说明：

--disable-content-trust :忽略镜像的校验,默认开启
```

docker search

```
docker search : 从Docker Hub查找镜像
```

```
docker search [OPTIONS] TERM
OPTIONS说明：

--automated :只列出 automated build类型的镜像；

--no-trunc :显示完整的镜像描述；

-f <过滤条件>:列出收藏数不小于指定值的镜像
```

```
//从 Docker Hub 查找所有镜像名包含 nginx，并且收藏数大于 10 的镜像
docker search -f stars=10 nginx
```

参数说明：
NAME: 镜像仓库源的名称
DESCRIPTION: 镜像的描述
OFFICIAL: 是否 docker 官方发布
stars: 类似 Github 里面的 star，表示点赞、喜欢的意思。
AUTOMATED: 自动构建。