# 容器生命週期管理

## Docker run 命令

**docker run ：**創建一個新的容器並運行一個命令

### 語法

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

### OPTIONS說明：

**-a stdin:** 指定標準輸入輸出內容類別型，可選 STDIN/STDOUT/STDERR 三項；

**-d:** 後臺運行容器，並返回容器ID；

**-i:** 以交互模式運行容器，通常與 -t 同時使用；

**-t:** 為容器重新分配一個偽輸入終端，通常與 -i 同時使用；

--name="nginx-lb": 為容器指定一個名稱；

**--dns 8.8.8.8:** 指定容器使用的DNS伺服器，默認和宿主一致；

--dns-search example.com: 指定容器DNS搜索功能變數名稱，默認和宿主一致；

**-h "mars":** 指定容器的hostname；

-e username="ritchie": 設置環境變數；

--env-file=[]: 從指定檔讀入環境變數；

--cpuset="0-2" or --cpuset="0,1,2": 綁定容器到指定CPU運行；

**-m :**設置容器使用記憶體最大值；

**--net="bridge":** 指定容器的網路連接類型，支援 bridge/host/none/container: 四種類型；

**--link=[]:** 添加連結到另一個容器；

**--expose=[]:** 開放一個埠或一組埠；

### 實例

使用docker鏡像nginx:latest以後臺模式啟動一個容器,並將容器命名為mynginx。

docker run --name mynginx -d nginx:latest

使用鏡像nginx:latest以後臺模式啟動一個容器,並將容器的80埠映射到主機隨機埠。

docker run -P -d nginx:latest

使用鏡像nginx:latest以後臺模式啟動一個容器,將容器的80埠映射到主機的80埠,主機的目錄/data映射到容器的/data。

docker run -p 80:80 -v /data:/data -d nginx:latest

使用鏡像nginx:latest以交互模式啟動一個容器,在容器內執行/bin/bash命令。

runoob@runoob:~$ docker run -it nginx:latest /bin/bash

root@b8573233d675:/#

## Docker start/stop/restart 命令

**docker start** :啟動一個或多少已經被停止的容器

**docker stop** :停止一個運行中的容器

docker restart :重啟容器

### 語法

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

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

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

### 實例

啟動已被停止的容器myrunoob

docker start myrunoob

停止運行中的容器myrunoob

docker stop myrunoob

重啟容器myrunoob

docker restart myrunoob

## Docker kill 命令

**docker kill** :殺掉一個運行中的容器。

### 語法

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

### OPTIONS說明：

**-s :**向容器發送一個信號

### 實例

殺掉運行中的容器mynginx

runoob@runoob:~$ docker kill -s KILL mynginx

mynginx

## Docker rm 命令

docker rm ：刪除一個或多少容器

### 語法

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

### OPTIONS說明：

**-f :**通過SIGKILL信號強制刪除一個運行中的容器

**-l :**移除容器間的網路連接，而非容器本身

**-v :**-v 刪除與容器關聯的卷

### 實例

強制刪除容器db01、db02

docker rm -f db01、db02

移除容器nginx01對容器db01的連接，連接名db

docker rm -l db

刪除容器nginx01,並刪除容器掛載的資料卷

docker rm -v nginx01

## Docker pause/unpause 命令

**docker pause** :暫停容器中所有的進程。

docker unpause :恢復容器中所有的進程。

### 語法

docker pause [OPTIONS] CONTAINER [CONTAINER...]

docker unpause [OPTIONS] CONTAINER [CONTAINER...]

### 實例

暫停資料庫容器db01提供服務。

docker pause db01

恢復資料庫容器db01提供服務。

docker unpause db01

## Docker create 命令

docker create ：創建一個新的容器但不啟動它

用法同 [docker run](http://www.runoob.com/docker/docker-run-command.html)

### 語法

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

語法同 [docker run](http://www.runoob.com/docker/docker-run-command.html)

### 實例

使用docker鏡像nginx:latest創建一個容器,並將容器命名為myrunoob

runoob@runoob:~$ docker create --name myrunoob nginx:latest

09b93464c2f75b7b69f83d56a9cfc23ceb50a48a9db7652ee4c27e3e2cb1961f

## Docker exec 命令

docker exec ：在運行的容器中執行命令

### 語法

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

### OPTIONS說明：

**-d :**分離模式: 在後臺運行

**-i :**即使沒有附加也保持STDIN 打開

**-t :**分配一個偽終端

### 實例

在容器mynginx中以交互模式執行容器內/root/runoob.sh腳本

runoob@runoob:~$ docker exec -it mynginx /bin/sh /root/runoob.sh

http://www.runoob.com/

在容器mynginx中開啟一個交互模式的終端

runoob@runoob:~$ docker exec -i -t mynginx /bin/bash

root@b1a0703e41e7:/#

# 容器操作

## Docker ps 命令

docker ps : 列出容器

### 語法

docker ps [OPTIONS]

### OPTIONS說明：

**-a :**顯示所有的容器，包括未運行的。

**-f :**根據條件過濾顯示的內容。

**--format :**指定返回值的範本檔。

**-l :**顯示最近創建的容器。

**-n :**列出最近創建的n個容器。

--no-trunc :不截斷輸出。

**-q :**靜默模式，只顯示容器編號。

**-s :**顯示總的檔大小。

### 實例

列出所有在運行的容器資訊。

runoob@runoob:~$ 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

列出最近創建的5個容器資訊。

runoob@runoob:~$ 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 ...

列出所有創建的容器ID。

runoob@runoob:~$ docker ps -a -q

09b93464c2f7

b8573233d675

b1a0703e41e7

f46fb1dec520

a63b4a5597de

6a4aa42e947b

de7bb36e7968

43a432b73776

664a8ab1a585

ba52eb632bbd

...

## Docker inspect 命令

docker inspect : 獲取容器/鏡像的中繼資料。

### 語法

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

### OPTIONS說明：

**-f :**指定返回值的範本檔。

**-s :**顯示總的檔大小。

**--type :**為指定類型返回JSON。

### 實例

獲取鏡像mysql:5.6的元信息。

runoob@runoob:~$ 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": {}

},

...

獲取正在運行的容器mymysql的 IP。

runoob@runoob:~$ docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mymysql

172.17.0.3

## Docker top 命令

**docker top :**查看容器中運行的進程資訊，支援 ps 命令參數。

### 語法

docker top [OPTIONS] CONTAINER [ps OPTIONS]

容器運行時不一定有/bin/bash終端來交互執行top命令，而且容器還不一定有top命令，可以使用docker top來實現查看container中正在運行的進程。

### 實例

查看容器mymysql的進程資訊。

runoob@runoob:~/mysql$ docker top mymysql

UID PID PPID C STIME TTY TIME CMD

999 40347 40331 18 00:58 ? 00:00:02 mysqld

查看所有運行容器的進程資訊。

for i in `docker ps |grep Up|awk '{print $1}'`;do echo \ &&docker top $i; done

## Docker attach 命令

docker attach :連接到正在運行中的容器。

### 語法

docker attach [OPTIONS] CONTAINER

要attach上去的容器必須正在運行，可以同時連接上同一個container來共用螢幕（與screen命令的attach類似）。

官方文檔中說attach後可以通過CTRL-C來detach，但實際上經過我的測試，如果container當前在運行bash，CTRL-C自然是當前行的輸入，沒有退出；如果container當前正在前臺運行進程，如輸出nginx的access.log日誌，CTRL-C不僅會導致退出容器，而且還stop了。這不是我們想要的，detach的意思按理應該是脫離容器終端，但容器依然運行。好在attach是可以帶上--sig-proxy=false來確保CTRL-D或CTRL-C不會關閉容器。

### 實例

容器mynginx將訪問日誌指到標準輸出，連接到容器查看訪問資訊。

runoob@runoob:~$ docker attach --sig-proxy=false mynginx

192.168.239.1 - - [10/Jul/2016:16:54:26 +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" "-"

## Docker events 命令

docker events : 從伺服器獲取即時事件

### 語法

docker events [OPTIONS]

### OPTIONS說明：

**-f ：**根據條件過濾事件；

**--since ：**從指定的時間戳記後顯示所有事件;

**--until ：**流水時間顯示到指定的時間為止；

### 實例

顯示docker 2016年7月1日後的所有事件。

runoob@runoob:~/mysql$ 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)

...

顯示docker 鏡像為mysql:5.6 2016年7月1日後的相關事件。

runoob@runoob:~/mysql$ 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)

2016-07-11T00:57:19.140141428+08:00 container die c8f0a32f12f5ec061d286af0b1285601a3e33a90a08ff1706de619ac823c345c (exitCode=1, image=mysql:5.6, name=mymysql)

2016-07-11T00:58:05.941019136+08:00 container destroy c8f0a32f12f5ec061d286af0b1285601a3e33a90a08ff1706de619ac823c345c (image=mysql:5.6, name=mymysql)

2016-07-11T00:58:07.965128417+08:00 container create a404c6c174a21c52f199cfce476e041074ab020453c7df2a13a7869b48f2f37e (image=mysql:5.6, name=mymysql)

2016-07-11T00:58:08.188734598+08:00 container start a404c6c174a21c52f199cfce476e041074ab020453c7df2a13a7869b48f2f37e (image=mysql:5.6, name=mymysql)

2016-07-11T00:58:20.010876777+08:00 container top a404c6c174a21c52f199cfce476e041074ab020453c7df2a13a7869b48f2f37e (image=mysql:5.6, name=mymysql)

2016-07-11T01:06:01.395365098+08:00 container top a404c6c174a21c52f199cfce476e041074ab020453c7df2a13a7869b48f2f37e (image=mysql:5.6, name=mymysql)

## Docker logs 命令

docker logs : 獲取容器的日誌

### 語法

docker logs [OPTIONS] CONTAINER

### OPTIONS說明：

**-f :**跟蹤日誌輸出

**--since :**顯示某個開始時間的所有日誌

**-t :**顯示時間戳記

**--tail :**僅列出最新N條容器日誌

### 實例

跟蹤查看容器mynginx的日誌輸出。

runoob@runoob:~$ 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從2016年7月1日後的最新10條日誌。

docker logs --since="2016-07-01" --tail=10 mynginx

## Docker wait 命令

**docker wait :** 阻塞運行直到容器停止，然後列印出它的退出代碼。

### 語法

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

### 實例

docker wait CONTAINER

## Docker export 命令

**docker export :**將檔案系統作為一個tar歸檔檔匯出到STDOUT。

### 語法

docker export [OPTIONS] CONTAINER

### OPTIONS說明：

**-o :**將輸入內容寫到檔。

### 實例

將id為a404c6c174a2的容器按日期保存為tar檔。

runoob@runoob:~$ docker export -o mysql-`date +%Y%m%d`.tar a404c6c174a2

runoob@runoob:~$ ls mysql-`date +%Y%m%d`.tar

mysql-20160711.tar

## Docker port 命令

**docker port :**列出指定的容器的埠映射，或者查找將PRIVATE\_PORT NAT到面向公眾的埠。

### 語法

docker port [OPTIONS] CONTAINER [PRIVATE\_PORT[/PROTO]]

### 實例

查看容器mynginx的埠映射情況。

runoob@runoob:~$ docker port mymysql

3306/tcp -> 0.0.0.0:3306

## 容器rootfs命令Docker commit 命令

docker commit :從容器創建一個新的鏡像。

### 語法

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

### OPTIONS說明：

**-a :**提交的鏡像作者；

**-c :**使用Dockerfile指令來創建鏡像；

**-m :**提交時的說明文字；

**-p :**在commit時，將容器暫停。

### 實例

將容器a404c6c174a2 保存為新的鏡像,並添加提交人資訊和說明資訊。

runoob@runoob:~$ docker commit -a "runoob.com" -m "my apache" a404c6c174a2 mymysql:v1

sha256:37af1236adef1544e8886be23010b66577647a40bc02c0885a6600b33ee28057

runoob@runoob:~$ docker images mymysql:v1

REPOSITORY TAG IMAGE ID CREATED SIZE

mymysql v1 37af1236adef 15 seconds ago 329 MB

## Docker cp 命令

**docker cp :**用於容器與主機之間的資料拷貝。

### 語法

docker cp [OPTIONS] CONTAINER:SRC\_PATH DEST\_PATH|-

docker cp [OPTIONS] SRC\_PATH|- CONTAINER:DEST\_PATH

### OPTIONS說明：

**-L :**保持源目標中的連結

### 實例

將主機/www/runoob目錄拷貝到容器96f7f14e99ab的/www目錄下。

docker cp /www/runoob 96f7f14e99ab:/www/

將主機/www/runoob目錄拷貝到容器96f7f14e99ab中，目錄重命名為www。

docker cp /www/runoob 96f7f14e99ab:/www

將容器96f7f14e99ab的/www目錄拷貝到主機的/tmp目錄中。

docker cp 96f7f14e99ab:/www /tmp/

## Docker diff 命令

docker diff : 檢查容器裡檔結構的更改。

### 語法

docker diff [OPTIONS] CONTAINER

### 實例

查看容器mymysql的檔結構更改。

runoob@runoob:~$ 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

## 鏡像倉庫Docker login/logout 命令

**docker login :** 登陸到一個Docker鏡像倉庫，如果未指定鏡像倉庫地址，默認為官方倉庫 Docker Hub

**docker logout :** 登出一個Docker鏡像倉庫，如果未指定鏡像倉庫地址，默認為官方倉庫 Docker Hub

### 語法

docker login [OPTIONS] [SERVER]

docker logout [OPTIONS] [SERVER]

### OPTIONS說明：

**-u :**登陸的用戶名

**-p :**登陸的密碼

### 實例

登陸到Docker Hub

docker login -u 用戶名 -p 密碼

登出Docker Hub

docker logout

## Docker pull 命令

**docker pull :**從鏡像倉庫中拉取或者更新指定鏡像

### 語法

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

### OPTIONS說明：

**-a :**拉取所有 tagged 鏡像

--disable-content-trust :忽略鏡像的校驗,默認開啟

### 實例

從Docker Hub下載java最新版鏡像。

docker pull java

從Docker Hub下載REPOSITORY為java的所有鏡像。

docker pull -a java

## Docker push 命令

**docker push :**將本地的鏡像上傳到鏡像倉庫,要先登陸到鏡像倉庫

### 語法

docker push [OPTIONS] NAME[:TAG]

### OPTIONS說明：

--disable-content-trust :忽略鏡像的校驗,默認開啟

### 實例

上傳本地鏡像myapache:v1到鏡像倉庫中。

docker push myapache:v1

## Docker search 命令

**docker search :** 從Docker Hub查找鏡像

### 語法

docker search [OPTIONS] TERM

### OPTIONS說明：

**--automated :**只列出 automated build類型的鏡像；

--no-trunc :顯示完整的鏡像描述；

**-s :**列出收藏數不小於指定值的鏡像。

### 實例

從Docker Hub查找所有鏡像名包含java，並且收藏數大於10的鏡像

runoob@runoob:~$ 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]

# 本地鏡像管理

## Docker images 命令

docker images : 列出本地鏡像。

### 語法

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

### OPTIONS說明：

**-a :**列出本地所有的鏡像（含中間映射層，預設情況下，過濾掉中間映射層）；

--digests :顯示鏡像的摘要資訊；

**-f :**顯示滿足條件的鏡像；

**--format :**指定返回值的範本檔；

--no-trunc :顯示完整的鏡像資訊；

**-q :**只顯示鏡像ID。

### 實例

查看本地鏡像列表。

runoob@runoob:~$ 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

...

列出本地鏡像中REPOSITORY為ubuntu的鏡像列表。

root@runoob:~# 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

## Docker rmi 命令

docker rmi : 刪除本地一個或多少鏡像。

### 語法

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

### OPTIONS說明：

**-f :**強制刪除；

**--no-prune :**不移除該鏡像的過程鏡像，默認移除；

### 實例

強制刪除本地鏡像runoob/ubuntu:v4。

root@runoob:~# docker rmi -f runoob/ubuntu:v4

Untagged: runoob/ubuntu:v4

Deleted: sha256:1c06aa18edee44230f93a90a7d88139235de12cd4c089d41eed8419b503072be

Deleted: sha256:85feb446e89a28d58ee7d80ea5ce367eebb7cec70f0ec18aa4faa874cbd97c73

## Docker tag 命令

**docker tag :**標記本地鏡像，將其歸入某一倉庫。

### 語法

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

### 實例

將鏡像ubuntu:15.10標記為 runoob/ubuntu:v3 鏡像。

root@runoob:~# docker tag ubuntu:15.10 runoob/ubuntu:v3

root@runoob:~# docker images runoob/ubuntu:v3

REPOSITORY TAG IMAGE ID CREATED SIZE

runoob/ubuntu v3 4e3b13c8a266 3 months ago 136.3 MB

## Docker build 命令

**docker build :**使用Dockerfile創建鏡像。

### 語法

docker build [OPTIONS] PATH | URL | -

### OPTIONS說明：

--build-arg=[] :設置鏡像創建時的變數；

--cpu-shares :設置 cpu 使用權重；

--cpu-period :限制 CPU CFS週期；

**--cpu-quota :**限制 CPU CFS配額；

--cpuset-cpus :指定使用的CPU id；

--cpuset-mems :指定使用的記憶體 id；

--disable-content-trust :忽略校驗，默認開啟；

**-f :**指定要使用的Dockerfile路徑；

**--force-rm :**設置鏡像過程中刪除中間容器；

--isolation :使用容器隔離技術；

--label=[] :設置鏡像使用的中繼資料；

**-m :**設置記憶體最大值；

**--memory-swap :**設置Swap的最大值為記憶體+swap，"-1"表示不限swap；

**--no-cache :**創建鏡像的過程不使用緩存；

**--pull :**嘗試去更新鏡像的新版本；

**-q :**安靜模式，成功後只輸出鏡像ID；

**--rm :**設置鏡像成功後刪除中間容器；

**--shm-size :**設置/dev/shm的大小，預設值是64M；

--ulimit :Ulimit配置。

### 實例

使用目前的目錄的Dockerfile創建鏡像。

docker build -t runoob/ubuntu:v1 .

使用URL github.com/creack/docker-firefox 的 Dockerfile 創建鏡像。

docker build github.com/creack/docker-firefox

## Docker history 命令

docker history : 查看指定鏡像的創建歷史。

### 語法

docker history [OPTIONS] IMAGE

### OPTIONS說明：

**-H :**以可讀的格式列印鏡像大小和日期，默認為true；

--no-trunc :顯示完整的提交記錄；

**-q :**僅列出提交記錄ID。

### 實例

查看本地鏡像runoob/ubuntu:v3的創建歷史。

root@runoob:~# 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

## Docker save 命令

**docker save :**將指定鏡像保存成 tar 歸檔檔。

### 語法

docker save [OPTIONS] IMAGE [IMAGE...]

### OPTIONS說明：

**-o :**輸出到的檔。

### 實例

將鏡像runoob/ubuntu:v3 生成my\_ubuntu\_v3.tar文檔

runoob@runoob:~$ docker save -o my\_ubuntu\_v3.tar runoob/ubuntu:v3

runoob@runoob:~$ ll my\_ubuntu\_v3.tar

-rw------- 1 runoob runoob 142102016 Jul 11 01:37 my\_ubuntu\_v3.ta

## Docker save 命令

**docker save :**將指定鏡像保存成 tar 歸檔檔。

### 語法

docker save [OPTIONS] IMAGE [IMAGE...]

### OPTIONS說明：

**-o :**輸出到的檔。

### 實例

將鏡像runoob/ubuntu:v3 生成my\_ubuntu\_v3.tar文檔

runoob@runoob:~$ docker save -o my\_ubuntu\_v3.tar runoob/ubuntu:v3

runoob@runoob:~$ ll my\_ubuntu\_v3.tar

-rw------- 1 runoob runoob 142102016 Jul 11 01:37 my\_ubuntu\_v3.ta

## Docker import 命令

docker import : 從歸檔檔中創建鏡像。

### 語法

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

### OPTIONS說明：

**-c :**應用docker 指令創建鏡像；

**-m :**提交時的說明文字；

### 實例

從鏡像歸檔檔my\_ubuntu\_v3.tar創建鏡像，命名為runoob/ubuntu:v4

runoob@runoob:~$ docker import my\_ubuntu\_v3.tar runoob/ubuntu:v4

sha256:63ce4a6d6bc3fabb95dbd6c561404a309b7bdfc4e21c1d59fe9fe4299cbfea39

runoob@runoob:~$ docker images runoob/ubuntu:v4

REPOSITORY TAG IMAGE ID CREATED SIZE

runoob/ubuntu v4 63ce4a6d6bc3 20 seconds ago 142.1 MB

# Info/version

Docker info 命令

docker info : 顯示 Docker 系統資訊，包括鏡像和容器數。。

### 語法

docker info [OPTIONS]

### 實例

查看docker系統資訊。

## Docker version 命令

docker version :顯示 Docker 版本資訊。

### 語法

docker version [OPTIONS]

### OPTIONS說明：

**-f :**指定返回值的範本檔。

### 實例

顯示 Docker 版本資訊。

$ docker version

Client:

Version: 1.8.2

API version: 1.20

Go version: go1.4.2

Git commit: 0a8c2e3

Built: Thu Sep 10 19:19:00 UTC 2015

OS/Arch: linux/amd64

Server:

Version: 1.8.2

API version: 1.20

Go version: go1.4.2

Git commit: 0a8c2e3

Built: Thu Sep 10 19:19:00 UTC 2015

OS/Arch: linux/amd64