**docker使用简介**

编 写 人：孙自翔

统 稿 人：

编写时间：2015-07-28

部 门 名：

审 核 人：

审核时间：

修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 修订内容简述 | 修订日期 | 版本 | 修订人 |
| 1 | 初稿 | 2015-07-28 | 0.1 | 孙自翔 |
|  |  |  |  |  |
|  |  |  |  |  |

正文目录

[1 文档说明 4](#_Toc426037473)

[1.1 本文档目的 4](#_Toc426037474)

[1.2 本文档要达成的目标 4](#_Toc426037475)

[1.3 更进一步 4](#_Toc426037476)

[1.4 一些说明 4](#_Toc426037477)

[2 一些概念 4](#_Toc426037478)

[2.1 cgroup 4](#_Toc426037479)

[2.2 linux namespace 4](#_Toc426037480)

[2.3 虚拟化 4](#_Toc426037481)

[2.4 linux 容器(lxc) 4](#_Toc426037482)

[3 docker介绍 4](#_Toc426037483)

[3.1 docker简介 5](#_Toc426037484)

[3.2 docker和虚拟机对比 5](#_Toc426037485)

[3.3 docker架构 6](#_Toc426037486)

[4 docker命令行介绍 6](#_Toc426037487)

[4.1 daemon管理 6](#_Toc426037488)

[ daemon的重启与关闭 7](#_Toc426037489)

[4.2 镜像管理 7](#_Toc426037490)

[ 镜像搜索 7](#_Toc426037491)

[ 获取镜像 7](#_Toc426037492)

[ 查看镜像 8](#_Toc426037493)

[ 删除镜像 8](#_Toc426037494)

[4.3 容器管理 8](#_Toc426037495)

[ 新建并启动 8](#_Toc426037496)

[ 启动已终止容器 9](#_Toc426037497)

[ 进入容器 9](#_Toc426037498)

[ 删除容器 9](#_Toc426037499)

[ 容器快照导出 10](#_Toc426037500)

[ 容器快照导入 10](#_Toc426037501)

[ 容器命名 10](#_Toc426037502)

[ 查看容器信息 11](#_Toc426037503)

[4.4 网络管理 11](#_Toc426037504)

[4.5 网络管理 11](#_Toc426037505)

[ 获取容器ip 11](#_Toc426037506)

[ 访问容器-直接访问 11](#_Toc426037507)

[ 外部访问容器-端口映射 11](#_Toc426037508)

[ 配置dns等 13](#_Toc426037509)

[ 配置docker网桥 13](#_Toc426037510)

[ 创建一个点到点连接 14](#_Toc426037511)

# 文档说明

## 本文档目的

本文档针对docker的使用做简要介绍，介绍docker的大部分命令的使用方式。cgroup和namespace都可以另立文档来整理。

## 本文档要达成的目标

* 理解虚拟化和容器的概念。
* 会使用大部分的docker命令
* 只介绍docker的使用，其架构和实现不介绍。

## 更进一步

* 深入研究docker涉及的各种技术（cgroup,namespace）

## 一些说明

演示平台为debian系列的宿主机，redhat系列的使用一样，有些命令(安装，服务启动)稍有不同。

# 一些概念

## cgroup

https://zh.wikipedia.org/zh-cn/Cgroups

http://coolshell.cn/articles/17049.html

## linux namespace

http://coolshell.cn/articles/17010.html

https://blog.jtlebi.fr/2013/12/22/introduction-to-linux-namespaces-part-1-uts/

http://dockerpool.com/static/books/docker\_practice/underly/namespace.html

| **分类** | **系统调用参数** | **相关内核版本** |
| --- | --- | --- |
| **Mount namespaces** | CLONE\_NEWNS | [Linux 2.4.19](http://lwn.net/2001/0301/a/namespaces.php3) |
| š**UTS namespaces** | CLONE\_NEWUTS | [Linux 2.6.19](http://lwn.net/Articles/179345/) |
| **IPC namespaces** | CLONE\_NEWIPC | [Linux 2.6.19](http://lwn.net/Articles/187274/) |
| **PID namespaces** | CLONE\_NEWPID | [Linux 2.6.24](http://lwn.net/Articles/259217/) |
| **Network namespaces** | CLONE\_NEWNET | [始于Linux 2.6.24 完成于 Linux 2.6.29](http://lwn.net/Articles/219794/) |
| **User namespaces** | CLONE\_NEWUSER | [始于 Linux 2.6.23 完成于 Linux 3.8)](http://lwn.net/Articles/528078/) |

## 虚拟化

虚拟化技术[Xen、OpenVZ、KVM、Hyper-V、VMWare](http://blog.csdn.net/myweishanli/article/details/24184489)

## linux 容器(lxc)

LinuX Container

https://zh.wikipedia.org/zh-cn/LXC

# docker介绍

http://tiewei.github.io/cloud/Docker-Getting-Start/

## docker简介

March 2013. 发布

March 13, 2014 0.9 抛弃LXC的执行环境，用libcontainer库代替

## docker和虚拟机对比

docker最初是基于容器的(LinuX Container)

1. docker对比虚拟机总结

| **特性** | **容器** | **虚拟机** |
| --- | --- | --- |
| 启动 | 秒级 | 分钟级 |
| 硬盘使用 | 一般为 MB | 一般为 GB |
| 性能 | 接近原生 | 弱于 |
| 系统支持量 | 单机支持上千个容器 | 一般几十个 |
| 异构 | Linux之间 | Windows和linux之间 |

http://dockerpool.com/static/books/docker\_practice/introduction/what.html

1. docker对比虚拟机示意图

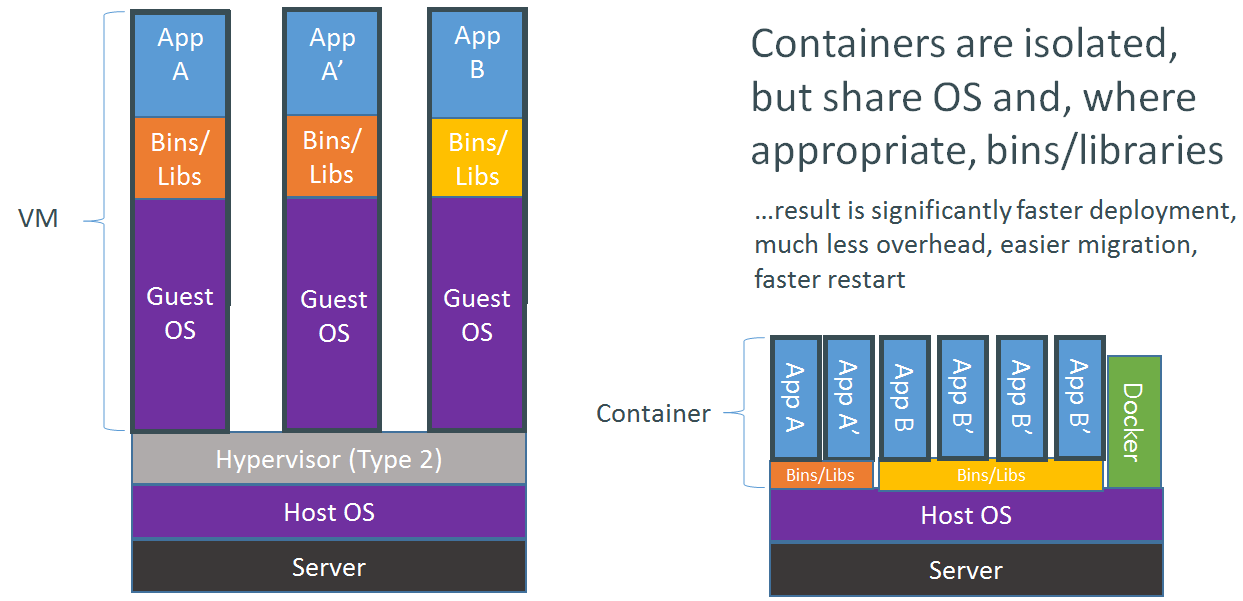
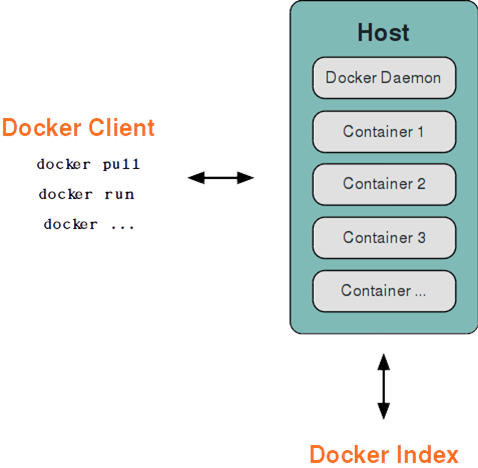


图3-1 虚拟机和container示意图

## docker架构

Docker 采用了 C/S架构，包括客户端和服务端。 Docker daemon 作为服务端接受来自客户的请求，并处理这些请求（创建、运行、分发容器）。 客户端和服务端既可以运行在一个机器上，也可通过 socket 或者 RESTful API 来进行通信。Docker daemon 一般在宿主主机后台运行，等待接收来自客户端的消息。 Docker 客户端则为用户提供一系列可执行命令，用户用这些命令实现跟 Docker daemon 交互。



# docker命令行介绍

主要参考资料

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

http://dockerpool.com/static/books/docker\_practice/

## daemon管理

3.3节所看到的架构中有个daemon程序。用户使用Docker Client(包括命令行)与Docker Daemon建立通信，并发送请求给后者。

### daemon的重启与关闭

查看

root@iZ25ox3a5vlZ:~# **ps aux|grep docker |grep -v grep**

root 23355 0.0 2.6 330848 13264 ? Sl Jul27 0:02 /usr/bin/docker -d -p /var/run/docker.pid

root@iZ25ox3a5vlZ:~# **service docker stop**

[ ok ] Stopping Docker: docker.

root@iZ25ox3a5vlZ:~# **ps aux|grep docker |grep -v grep**

root@iZ25ox3a5vlZ:~# **service docker start**

[ ok ] Starting Docker: docker.

root@iZ25ox3a5vlZ:~# **ps aux|grep docker |grep -v grep**

root 32458 2.5 2.6 265312 13592 ? Sl 16:44 0:00 /usr/bin/docker -d -p /var/run/docker.pid

## 镜像管理

容器启动之前，必须先有镜像(image).

Docker 镜像就是一个只读的模板。一个镜像可以包含一个完整的 ubuntu 操作系统环境，里面仅安装了 Apache 或用户需要的其它应用程序。镜像可以用来创建 Docker 容器。Docker 提供了一个很简单的机制来创建镜像或者更新现有的镜像，用户还可以直接从其他人那里下载一个已经做好的镜像来直接使用。

远程拉取镜像默认去[Docker Hub](https://hub.docker.com)(https://hub.docker.com/)上请求

### 镜像搜索

在[Docker Hub](https://hub.docker.com)上有大量的镜像可供下载，下载之前需要搜索到想要的镜像. 有时候官方仓库注册服务器下载较慢，可以从其他仓库下载.参考（http://dockerpool.com/static/books/docker\_practice/image/pull.html）

root@iZ25ox3a5vlZ:~# **docker help search**

Usage: docker search [OPTIONS] TERM

Search the Docker Hub for images

root@iZ25ox3a5vlZ:~# **docker search ubuntu**

NAME DESCRIPTION STARS OFFICIAL AUTOMATED

ubuntu Ubuntu is a Debian-based Linux operating s... 1988 [OK]

ubuntu-upstart Upstart is an event-based replacement for ... 28 [OK]

torusware/speedus-ubuntu Always updated official Ubuntu docker imag... 25 [OK]

...

root@iZ25ox3a5vlZ:~# **docker search centos**

NAME DESCRIPTION STARS OFFICIAL AUTOMATED

centos The official build of CentOS. 1149 [OK]

ansible/centos7-ansible Ansible on Centos7 45 [OK]

jdeathe/centos-ssh-apache-php CentOS-6 6.6 x86\_64 / Apache / PHP / PHP m... 11 [OK]

...

### 获取镜像

可以获取搜索到的镜像，比如ubuntu

root@iZ25ox3a5vlZ:~# docker pull ubuntu

Pulling repository ubuntu

d2a0ecffe6fa: Pulling dependent layers

83e4dde6b9cf: Downloading 5.406 MB...

### 查看镜像

镜像下载到本地之后，可以查看.也可以运行镜像。

root@iZ25ox3a5vlZ:~# **docker images**

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

ubuntu 14.04 d2a0ecffe6fa 2 weeks ago 188.4 MB

ubuntu 14.04.2 d2a0ecffe6fa 2 weeks ago 188.4 MB

ubuntu latest d2a0ecffe6fa 2 weeks ago 188.4 MB

ubuntu trusty d2a0ecffe6fa 2 weeks ago 188.4 MB

root@iZ25ox3a5vlZ:~# **docker images --tree**

# 运行

root@iZ25ox3a5vlZ:~# **sudo docker run -t -i ubuntu:12.04 /bin/bash**

root@709de5c48001:/# 此时已经在容器中了

### 删除镜像

不需要的镜像可以删除

root@iZ25ox3a5vlZ:~# **docker images**

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

test/ubuntu\_lamp v1.0 4585b33df7be 7 minutes ago 188.2 MB

ubuntu 14.04 d2a0ecffe6fa 2 weeks ago 188.4 MB

ubuntu 12.04 56bd9b2fadf9 6 months ago 149.6 MB

root@iZ25ox3a5vlZ:~# **docker help rmi**

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

Remove one or more images

...

root@iZ25ox3a5vlZ:~# **docker rmi 4585b33df7be**

Untagged: test/ubuntu\_lamp:v1.0

Deleted: 4585b33df7beedab2b7cf20dfc5c9ff8054e156748409406d12a3d8d2594d8d0

root@iZ25ox3a5vlZ:~# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

ubuntu 14.04 d2a0ecffe6fa 2 weeks ago 188.4 MB

ubuntu 14.04.2 d2a0ecffe6fa 2 weeks ago 188.4 MB

### 镜像存放位置

http://liubin.org/2014/03/10/about-docker-images-storage/

root@iZ25ox3a5vlZ:~# **cat /var/lib/docker/repositories-aufs |python -mjson.tool**

{

"Repositories": {

"centos": {

"latest": "34943839435dfb2ee646b692eebb06af13823a680ace00c0adc232c437c4f90c"

},

"debian": {

"latest": "479215127fa7b852902ed734f3a7ac69177c0d4d9446ad3a1648938230c3c8ab"

},

"haichengsun123/debian-nginx": {

"latest": "42a7628febd3527380db30f7aba7eb3430cd5acf278fea15574a96eb0b9a1aeb"

},

"haichengsun123/unbuntu-update": {

"v2": "9edf33b007554188123a0107083c68f1401024061c837665ebcaae7f06ca2883"

},

"maxexcloo/debian": {

"latest": "f9e18dbdce6a6463279b3d2d863ee1332b89ef958d98d7a3b479e7ce9e32da61"

},

"ouruser/sinatra": {

"v2": "ea27f4536d15e9870385fb3543a29c8f2a10f89488d609fb2ed25036ae5eeceb"

},

"sshd-init": {

"latest": "ce624f3e2fdf294a1a33932148064d17b7d8bfc02358b766b9d83560165df65d"

},

"sshd-init-after": {

"latest": "47593df61425e2628f350a9c354fda982d52c826dc527f6e803f67a0df60d0ee"

},

"ubuntu": {

"12.04": "56bd9b2fadf9971f29e7e0ce773b93abdf0da02357e8c7034f9ae222011c716d",

"14.04": "d2a0ecffe6fa4ef3de9646a75cc629bbd9da7eead7f767cb810f9808d6b3ecb6",

"14.04.2": "d2a0ecffe6fa4ef3de9646a75cc629bbd9da7eead7f767cb810f9808d6b3ecb6",

"latest": "d2a0ecffe6fa4ef3de9646a75cc629bbd9da7eead7f767cb810f9808d6b3ecb6",

"trusty": "d2a0ecffe6fa4ef3de9646a75cc629bbd9da7eead7f767cb810f9808d6b3ecb6",

"trusty-20150630": "d2a0ecffe6fa4ef3de9646a75cc629bbd9da7eead7f767cb810f9808d6b3ecb6"

},

"ubuntu-ssh": {

"latest": "10c8e71d82c99dc34f96391515e0efbf0f1bae3f7d00a552dcc1006a10a49d8d"

}

}

}Deleted: 4585b33df7beedab2b7cf20dfc5c9ff8054e156748409406d12a3d8d2594d8d0

root@iZ25ox3a5vlZ:~# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

ubuntu 14.04 d2a0ecffe6fa 2 weeks ago 188.4 MB

ubuntu 14.04.2 d2a0ecffe6fa 2 weeks ago 188.4 MB

## 容器管理

Docker 利用容器来运行应用。容器是从镜像创建的运行实例。它可以被启动、开始、停止、删除。每个容器都是相互隔离的、保证安全的平台。

可以把容器看做是一个简易版的 Linux 环境（包括root用户权限、进程空间、用户空间和网络空间等）和运行在其中的应用程序。

\*注：镜像是只读的，容器在启动的时候创建一层可写层作为最上层。

### 新建并启动

启动容器有两种方式，一种是基于镜像新建一个容器并启动，另外一个是将在终止状态（stopped）的容器重新启动。

Docker 的容器非常轻量级，很多时候用户都是随时删除和新创建容器。

指定--rm 标记，则容器在终止后会立刻删除。--rm 和 -d 参数不能同时使用。

--rm=false Automatically remove the container when it exits (incompatible with -d)

root@iZ25ox3a5vlZ:~# **docker exec 8975a4aba2b9 ps**

PID TTY TIME CMD

22 ? 00:00:00 ps

root@iZ25ox3a5vlZ:~# **docker run ubuntu:14.04 /bin/echo 'Hello world'**

Hello world

root@iZ25ox3a5vlZ:~# **docker run ubuntu:14.04 ip a**

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

17: eth0: <NO-CARRIER,BROADCAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state DOWN group default

link/ether 02:42:ac:11:00:08 brd ff:ff:ff:ff:ff:ff

inet 172.17.0.8/16 scope global eth0

valid\_lft forever preferred\_lft forever

inet6 fe80::42:acff:fe11:8/64 scope link tentative

valid\_lft forever preferred\_lft forever

root@iZ25ox3a5vlZ:~# **docker run -t -i ubuntu:14.04 /bin/bash #运行之后就在容器中了**

root@2fef13770bef:/# ps aux

利用 docker run 来创建容器时，Docker 在后台运行的标准操作包括：

1. 检查本地是否存在指定的镜像，不存在就从公有仓库下载
2. 利用镜像创建并启动一个容器
3. 分配一个文件系统，并在只读的镜像层外面挂载一层可读写层(mount)
4. 从宿主主机配置的网桥接口中桥接一个虚拟接口到容器中去(brctl)
5. 从地址池配置一个 ip 地址给容器(ip l)
6. 执行用户指定的应用程序
7. 执行完毕后容器被终止

### 启动已终止容器

上面运行的命令退出之后，可以查看已经终止的容器。终止的容器可以重新启动

root@iZ25ox3a5vlZ:~# **docker ps -a**  #输出有删减

CONTAINER ID IMAGE COMMAND CREATED 2fef13770bef ubuntu:14.04 "/bin/bash" 8 minutes ago

root@iZ25ox3a5vlZ:~# **docker start -a i 2fef13770bef**

root@2fef13770bef:/# exit

exit

### 进入容器

在使用 -d 参数时，容器启动后会进入后台。 某些时候需要进入容器进行操作，有很多种方法，包括使用 docker attach 命令或 nsenter 工具等。

只介绍docker attach，可以以id或名字attach

root@iZ25ox3a5vlZ:~# **docker ps**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

5bbee832656f sshd-init-after:latest "/bin/bash" 26 minutes ago Up 26 minutes pensive\_torvalds

root@iZ25ox3a5vlZ:~# **docker attach 5bbee832656f**

root@5bbee832656f:/#

### 删除容器

可以使用 docker rm 来删除一个处于终止状态的容器。

如果要删除一个运行中的容器，可以添加 -f 参数。Docker 会发送 SIGKILL 信号给容器。

root@iZ25ox3a5vlZ:~# **docker ps -a |grep 5c051733f8d1**

5c051733f8d1 ubuntu:14.04 .. admiring\_bardeen

root@iZ25ox3a5vlZ:~# **docker rm 5c051733f8d1**

5c051733f8d1

root@iZ25ox3a5vlZ:~# **docker ps -a |grep 5c051733f8d1**

root@iZ25ox3a5vlZ:~# **docker run -tdi ubuntu:14.04 /bin/bash**

bed4b4754bc661a9c82076fa5f534c57d0d95077d185050fae60be2edb8266ad

root@iZ25ox3a5vlZ:~# **docker ps**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

bed4b4754bc6 ubuntu:14.04 "/bin/bash" 5 seconds ago Up 5 seconds berserk\_colden

root@iZ25ox3a5vlZ:~# **docker rm berserk\_colden**

Error response from daemon: You cannot remove a running container. Stop the container before attempting removal or use -f

FATA[0000] Error: failed to remove one or more containers

root@iZ25ox3a5vlZ:~# **docker rm -f berserk\_colden**

berserk\_colden

### 容器快照导出

如果在容器中做了改动(比如安装了lamp环境等)，需要导出本地某个容器到本地

root@iZ25ox3a5vlZ:~# **docker run -tdi ubuntu:14.04 /bin/bash**

8975a4aba2b99707276a3f858f61dde1f9179d252d11cd223bbbe6770e57200a

root@iZ25ox3a5vlZ:~# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

8975a4aba2b9 ubuntu:14.04 "/bin/bash" 12 seconds ago Up 11 seconds prickly\_franklin

root@iZ25ox3a5vlZ:~# **docker export 8975a4aba2b9 > ubuntu\_lamp.tar**

root@iZ25ox3a5vlZ:~# **ls ubuntu\_lamp.tar**

ubuntu\_lamp.tar

### 容器快照导入

可以使用从容器快照文件中再导入为镜像

root@iZ25ox3a5vlZ:~# **docker run -tdi ubuntu:14.04 /bin/bash**

8975a4aba2b99707276a3f858f61dde1f9179d252d11cd223bbbe6770e57200a

root@iZ25ox3a5vlZ:~# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

8975a4aba2b9 ubuntu:14.04 "/bin/bash" 12 seconds ago Up 11 seconds prickly\_franklin

root@iZ25ox3a5vlZ:~# **docker export 8975a4aba2b9 > ubuntu\_lamp.tar**

root@iZ25ox3a5vlZ:~# **ls ubuntu\_lamp.tar**

ubuntu\_lamp.tar

用户既可以使用 docker load 来导入镜像存储文件到本地镜像库，也可以使用 docker import 来导入一个容器快照到本地镜像库。这两者的区别在于容器快照文件将丢弃所有的历史记录和元数据信息（即仅保存容器当时的快照状态），而镜像存储文件将保存完整记录，体积也要大。此外，从容器快照文件导入时可以重新指定标签等元数据信息。

### 容器命名

当创建容器的时候，系统默认会分配一个名字。还可以自定义命名容器

1. 自定义的命名，比较好记，比如一个web应用容器我们可以给它起名叫web
2. 当连接其他容器时，可以作为一个有用的参考点，比如连接web容器到db容器

root@iZ25ox3a5vlZ:~# **docker run -idt --name centos\_host centos /bin/bash**

1237f023112f01319845a514a198348b167986cb947ec222982fea60c44eaffc

root@iZ25ox3a5vlZ:~# **docker ps**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

1237f023112f centos:latest "/bin/bash" 6 seconds ago Up 5 seconds centos\_host

### 查看容器信息

每个容器的详细信息可以导出，并过滤某些字段

root@iZ25ox3a5vlZ:~# **docker inspect e13ce2504430**

root@iZ25ox3a5vlZ:~# **docker inspect -f '{{.State.Pid}}' e13ce2504430**

4310

root@iZ25ox3a5vlZ:~# **docker inspect -f '{{.NetworkSettings.IPAddress}}' e13ce2504430**

172.17.0.18

root@iZ25ox3a5vlZ:~# **docker inspect -f '{{.Name}}' b9703e84e26e**

/ssh-to-container

## 网络管理

Docker 允许通过外部访问容器或容器互联的方式来提供网络服务。

## 网络管理

### 获取容器ip

\* ip地址可以通过[查看容器信息](#_查看容器信息)获取,.

\* 也可以通过执行命令(exec)的方式运行ip命令

### 访问容器-直接访问

可以在宿主机上直接访问容器，前提是开启ssh服务。

root@iZ25ox3a5vlZ:~# **docker run -idt 47593df61425**

b983d3cf4428106e431147eae76d663027c65174b9dd9d1947e5c1645e313729

root@iZ25ox3a5vlZ:~# **docker ps**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

b983d3cf4428 sshd-init-after:latest "/bin/bash" 2 seconds ago Up 2 seconds drunk\_yonath

root@iZ25ox3a5vlZ:~# **docker exec b983d3cf4428 service ssh start**

\* Starting OpenBSD Secure Shell server sshd

...done.

root@iZ25ox3a5vlZ:~# **docker exec b983d3cf4428 ip a s eth0**

45: eth0: <BROADCAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UP group default

link/ether 02:42:ac:11:00:16 brd ff:ff:ff:ff:ff:ff

inet 172.17.0.22/16 scope global eth0

valid\_lft forever preferred\_lft forever

inet6 fe80::42:acff:fe11:16/64 scope link

valid\_lft forever preferred\_lft forever

root@iZ25ox3a5vlZ:~# **ssh root@172.17.0.22**

The authenticity of host '172.17.0.22 (172.17.0.22)' can't be established.

ECDSA key fingerprint is 9c:2b:21:a7:19:fb:5f:9f:a5:fe:15:ee:cd:a7:73:22.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '172.17.0.22' (ECDSA) to the list of known hosts.

root@172.17.0.22's password:

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.16.0-0.bpo.4-amd64 x86\_64)

\* Documentation: https://help.ubuntu.com/

Last login: Mon Dec 29 09:16:01 2014 from 172.17.42.1

root@b983d3cf4428:~# # in the container

### 外部访问容器-端口映射

注意观察iptables规则

1. 可以映射指定端口

root@iZ25ox3a5vlZ:~# **docker run -idt -p 5000:22 47593df61425**

0f9fedc73c1e96b7b6981520c092005fafcd2820423861c3242f710769f5de91

root@iZ25ox3a5vlZ:~# **docker ps**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

0f9fedc73c1e sshd-init-after:latest "/bin/bash" 2 seconds ago Up 1 seconds 0.0.0.0:5000->22/tcp focused\_jang

root@iZ25ox3a5vlZ:~# **netstat -anpt|grep 5000**

tcp6 0 0 :::5000 :::\* LISTEN 20371/docker-proxy

root@iZ25ox3a5vlZ:~# **docker exec 0f9fedc73c1e service ssh start**

\* Starting OpenBSD Secure Shell server sshd

...done.

root@iZ25ox3a5vlZ:~# **ssh -p5000 root@123.56.106.58** # 用宿主机的任何一个ip访问

root@123.56.106.58's password:

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.16.0-0.bpo.4-amd64 x86\_64)

\* Documentation: https://help.ubuntu.com/

Last login: Mon Dec 29 09:16:01 2014 from 172.17.42.1

root@0f9fedc73c1e:~# # in the container

root@0f9fedc73c1e:~#

1. 可以映射指定的ip和端口

root@iZ25ox3a5vlZ:~# **docker run -idt -p 10.170.217.1:5000:22 47593df6142**

5c17728dcb929eff4beedbc8d647ff58cf390140251990e31e021bb5f8965d95

root@iZ25ox3a5vlZ:~# **docker ps**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

5c17728dcb92 sshd-init-after:latest "/bin/bash" 3 seconds ago Up 3 seconds 10.170.217.1:5000->22/tcp modest\_brown

root@iZ25ox3a5vlZ:~# **docker exec 5c17728dcb92 service ssh start**

\* Starting OpenBSD Secure Shell server sshd

...done.

root@iZ25ox3a5vlZ:~# **ssh -p 5000 root@123.56.106.58**

ssh: connect to host 123.56.106.58 port 5000: Connection refused

root@iZ25ox3a5vlZ:~# **netstat -anpt |grep 5000**

tcp 0 0 10.170.217.1:5000 0.0.0.0:\* LISTEN 20572/docker-proxy

root@iZ25ox3a5vlZ:~# **ssh -p 5000 root@10.170.217.1**

The authenticity of host '[10.170.217.1]:5000 ([10.170.217.1]:5000)' can't be established.

ECDSA key fingerprint is 9c:2b:21:a7:19:fb:5f:9f:a5:fe:15:ee:cd:a7:73:22.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[10.170.217.1]:5000' (ECDSA) to the list of known hosts.

root@10.170.217.1's password:

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.16.0-0.bpo.4-amd64 x86\_64)

\* Documentation: https://help.ubuntu.com/

Last login: Mon Dec 29 09:16:01 2014 from 172.17.42.1

1. 可以映射指定的ip任意端口

root@iZ25ox3a5vlZ:~# **docker run -idt -p 10.170.217.1::22 47593df6142**

ae24270bca4aa3a653c36e38a7853062e02ca8e5936c3459971fbfb97e6a4f06

root@iZ25ox3a5vlZ:~# **docker ps**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

ae24270bca4a sshd-init-after:latest "/bin/bash" 3 seconds ago Up 2 seconds 10.170.217.1:49154->22/tcp romantic\_morse

root@iZ25ox3a5vlZ:~# **docker port ae24270bca4a 22**

10.170.217.1:49154

root@iZ25ox3a5vlZ:~# **docker exec ae24270bca4a service ssh start**

\* Starting OpenBSD Secure Shell server sshd

...done.

root@iZ25ox3a5vlZ:~# **ssh -p 49154 root@10.170.217.1**

The authenticity of host '[10.170.217.1]:49154 ([10.170.217.1]:49154)' can't be established.

ECDSA key fingerprint is 9c:2b:21:a7:19:fb:5f:9f:a5:fe:15:ee:cd:a7:73:22.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[10.170.217.1]:49154' (ECDSA) to the list of known hosts.

root@10.170.217.1's password:

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.16.0-0.bpo.4-amd64 x86\_64)

\* Documentation: https://help.ubuntu.com/

Last login: Mon Dec 29 09:16:01 2014 from 172.17.42.1

root@ae24270bca4a:~# # in the container

root@ae24270bca4a:~#

### 配置dns等

-h HOSTNAME or --hostname=HOSTNAME 设定容器的主机名，它会被写到容器内的 /etc/hostname 和 /etc/hosts

--dns=IP\_ADDRESS 添加 DNS 服务器到容器的 /etc/resolv.conf 中， --dns-search=DOMAIN 设定容器的搜索域，

root@iZ25ox3a5vlZ:~# **docker run -idt --hostname=jd\_test --dns=8.8.8.8 --dns-search=jd.com 47593df6142 /usr/sbin/sshd -D**

677b5f14bc00e87cdcc99d53a3d813f71410dd34c8dd45641b17a377a6090de1

root@iZ25ox3a5vlZ:~# **docker ps**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

677b5f14bc00 sshd-init-after:latest "/usr/sbin/sshd -D" 5 seconds ago Up 5 seconds adoring\_davinci

root@iZ25ox3a5vlZ:~# **docker exec 677b5f14bc00 ip a s eth0**

71: eth0: <BROADCAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UP group default

link/ether 02:42:ac:11:00:23 brd ff:ff:ff:ff:ff:ff

inet 172.17.0.35/16 scope global eth0

valid\_lft forever preferred\_lft forever

inet6 fe80::42:acff:fe11:23/64 scope link

valid\_lft forever preferred\_lft forever

root@iZ25ox3a5vlZ:~# **ssh root@172.17.0.35**

The authenticity of host '172.17.0.35 (172.17.0.35)' can't be established.

ECDSA key fingerprint is 9c:2b:21:a7:19:fb:5f:9f:a5:fe:15:ee:cd:a7:73:22.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '172.17.0.35' (ECDSA) to the list of known hosts.

root@172.17.0.35's password:

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.16.0-0.bpo.4-amd64 x86\_64)

\* Documentation: https://help.ubuntu.com/

Last login: Mon Dec 29 09:16:01 2014 from 172.17.42.1

root@jd\_test:~# **cat /etc/hostname**

jd\_test

root@jd\_test:~# **cat /etc/resolv.conf**

nameserver 8.8.8.8

search jd.com

root@jd\_test:~# **hostname**

jd\_test

root@jd\_test:~# **cat /etc/hosts**

172.17.0.35 jd\_test

127.0.0.1 localhost

### 配置docker网桥

默认情况下，所有容器都会被连接到 docker0 网桥上。观察

### 创建一个点到点连接

TODO

