iThome

Container Summit 2015

航向容器新世界

12/10~11

台大集思會議中心

深入理解Dockerfile、Docker映像檔以及Docker容器

孫宏亮@DaoCloud

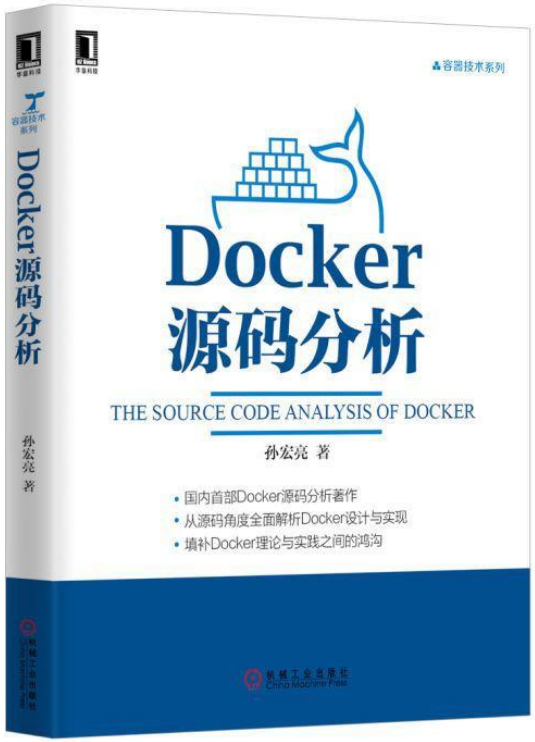
allen.sun@daocloud.io

2015/12/11

個人介紹

孫宏亮

- DaoCloud技術合夥人、軟件工程師
- 主要負責企業級容器雲平台的研發工作
- 著有 《Docker源碼分析》
  《Docker容器與容器雲》
- 數年來一直從事雲計算、PaaS領域的研究與實踐
- 中國大陸較早一批接觸Docker的先行者，同時也是Docker技術的主要推廣者。



http://www.daocloud.io

Agenda

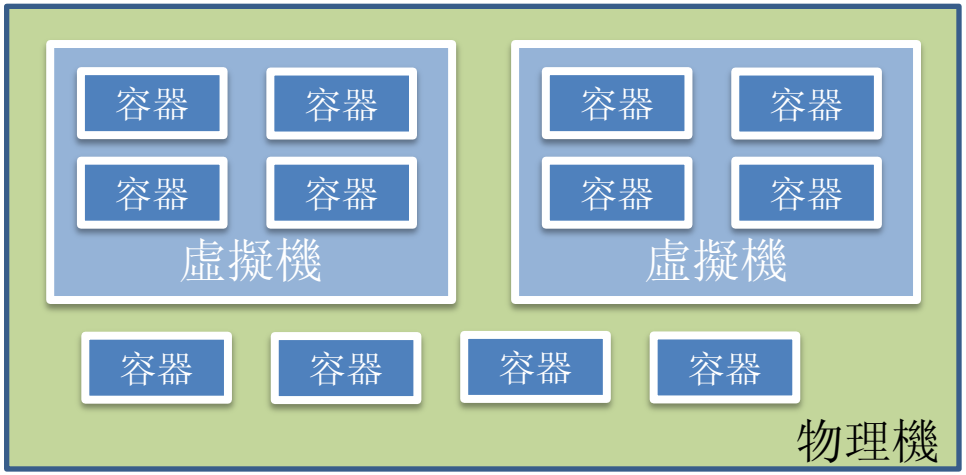
- Docker簡介
- Dockerfile
- Docker映像檔
- Docker容器

Docker簡介

Docker包含兩方面技術：

－容器技術：

有效分配與管理物理資源
實現資源隔離



容器、物理機以及虛擬機關系

Docker简介

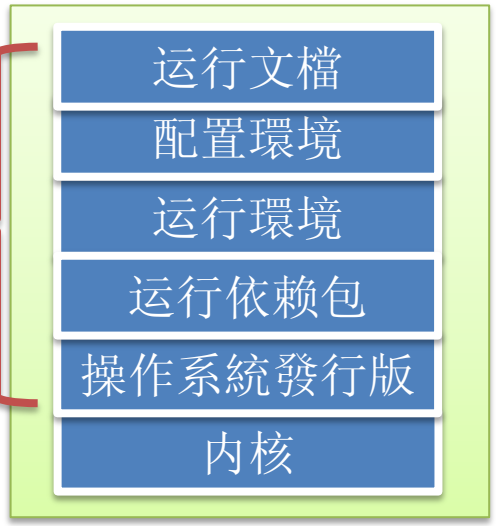
Docker包含兩方面技術：

一鏡像技術

打破“代碼即應用”的觀念

從系統環境開始，自底至上打包應用

Docker: “映像檔即應用”



映像檔

```
#include <stdio.h>
void main(){
    printf(“Hello World!”);
}
```

代碼＝應用？

编译

Hello.exe

执行文檔＝應用？

windows OK

Linux X

Solaris X

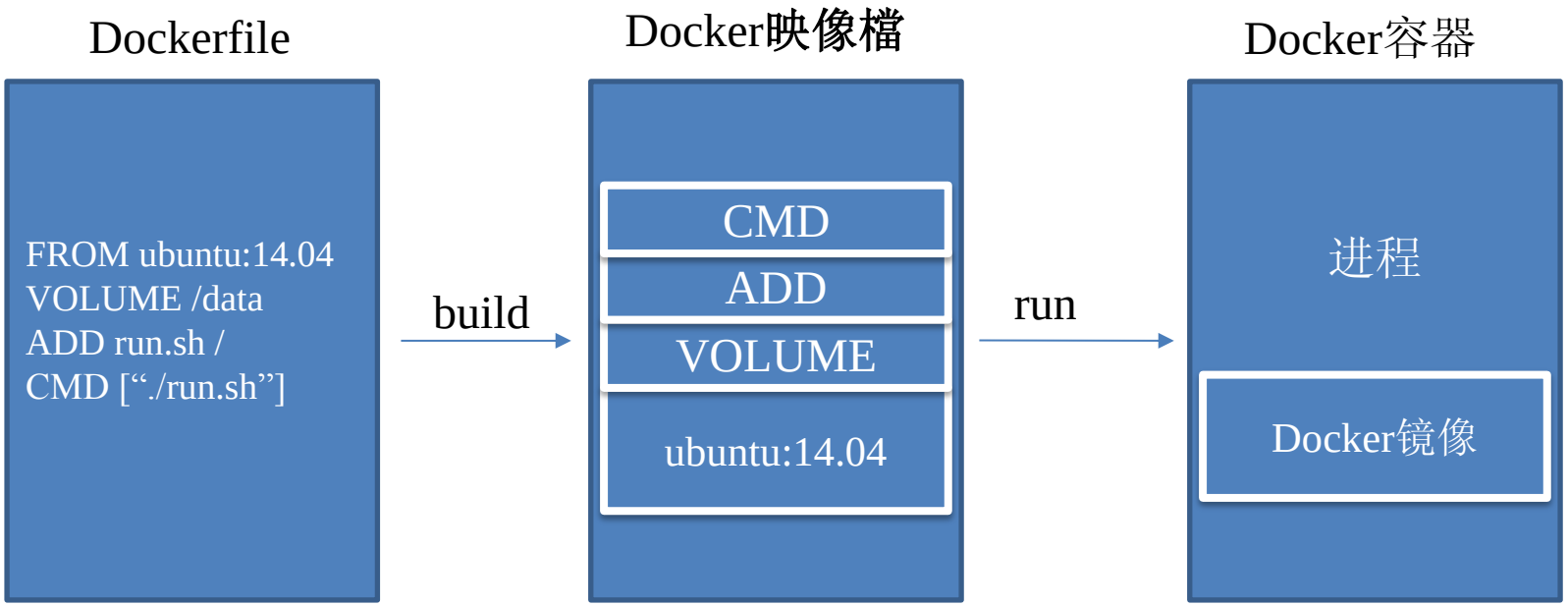
…

…

平台耦合性

Docker化實踐

Dockerfile、Docker映像檔与Docker容器



Dockerfile定義一切

Docker化实践

容器內技術棧：

1. 單進程理念
2. 不存在傳統的init進程（全局PID＝1）
   ——dockerinit與init進程的區別
3. 缺少基本的服務進程
   ——cron
   ——rsyslogd等
4. 與內核進程通信能力薄弱（ipc命名空間隔離）

導致遺留系統Docker化存在壓力。
重構？非重構下的最佳實踐？

Dockerfile

- 一种描述型文檔
- Docker映像檔的配方
- 包含多條命令

  FROM, RUN, ENTRYPOINT, CMD…
- docker build命令實現構建

  docker build –t=“demo” .

Dockerfile

DaoCloud / **dao-wordpress**

Code | Issues 0 | Pull requests 0 | Wiki | Pulse

No description or website provided. — Edit

14 commits | 2 bra

Branch: **master** | **New pull request**

GolfenGuo specify wordpress image tag

Dockerfile	specify wordpre
README.md	更新了与官方
adapter.sh	fix script
run.sh	initial commit
wordpress.conf	initial commit
wp-config.php	initial commit

Branch: **master** **dao-wordpress** / **Dockerfile**

GolfenGuo specify wordpress image tag

3 contributors

8 lines (5 sloc) | 159 Bytes

```
1  FROM wordpress:4.3.1
2
3  ADD adapter.sh /opt/adapter.sh
4  RUN chmod +x /opt/adapter.sh
5
6  ENTRYPOINT ["/opt/adapter.sh", "/entrypoint.sh"]
7  CMD ["apache2-foreground"]
```

Dockerfile

```
$ docker build -t svendowideit/ambassador .
Sending build context to Docker daemon 15.36 kB
Step 0 : FROM alpine:3.2
---> 31f630c65071

Step 1 : MAINTAINER SvenDowideit@home.org.au
---> Using cache
---> 2a1c91448f5f

Step 2 : RUN apk update && apk add socat && rm -r /var/cache/
---> Using cache
---> 21ed6e7fbb73

Step 3 : CMD env | grep _TCP= | sed 's/.*_PORT_\([0-9]*\)_TCP=tcp:\/\/\(.*\):\(.*\)/socat -t
100000000 TCP4-LISTEN:\1,fork,reuseaddr TCP4:\2:\3 \& wait/' | sh
---> Using cache
---> 7ea8aef582cc Successfully built 7ea8aef582cc
```

docker build流程

docker build 的 cache 机制 http://docs.daocloud.io/allen-docker/docker-build-cache

Docker映像檔

Dockerfile → Docker映像檔

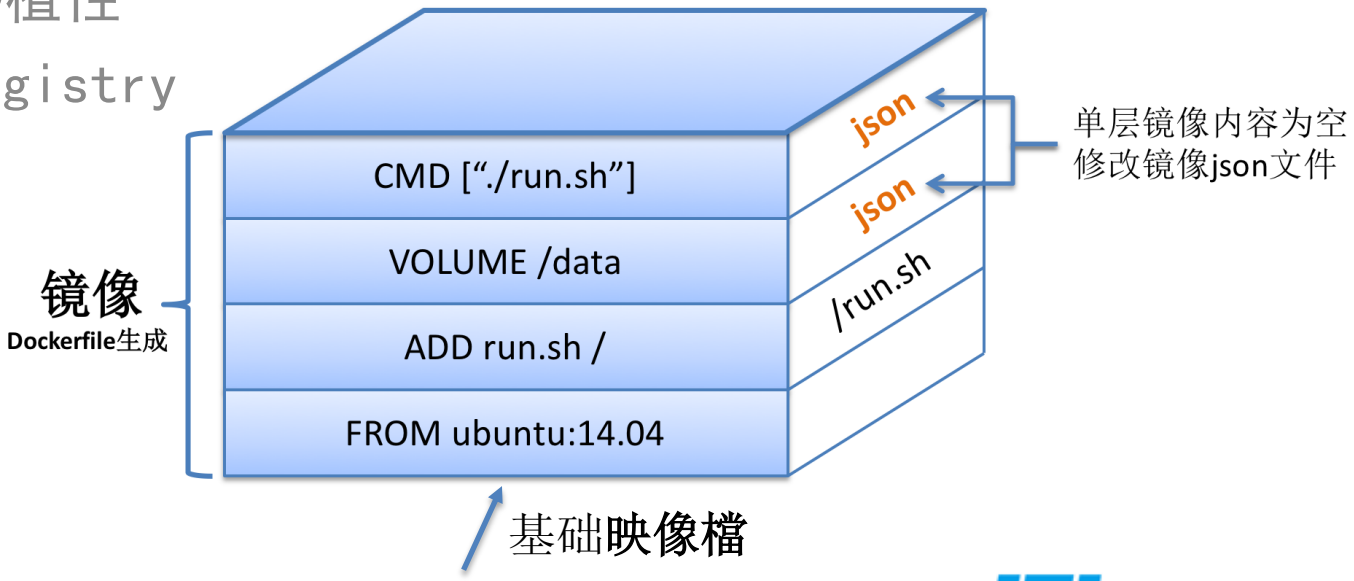
（原材料）→（交付成品）

一个映像檔由多层构成

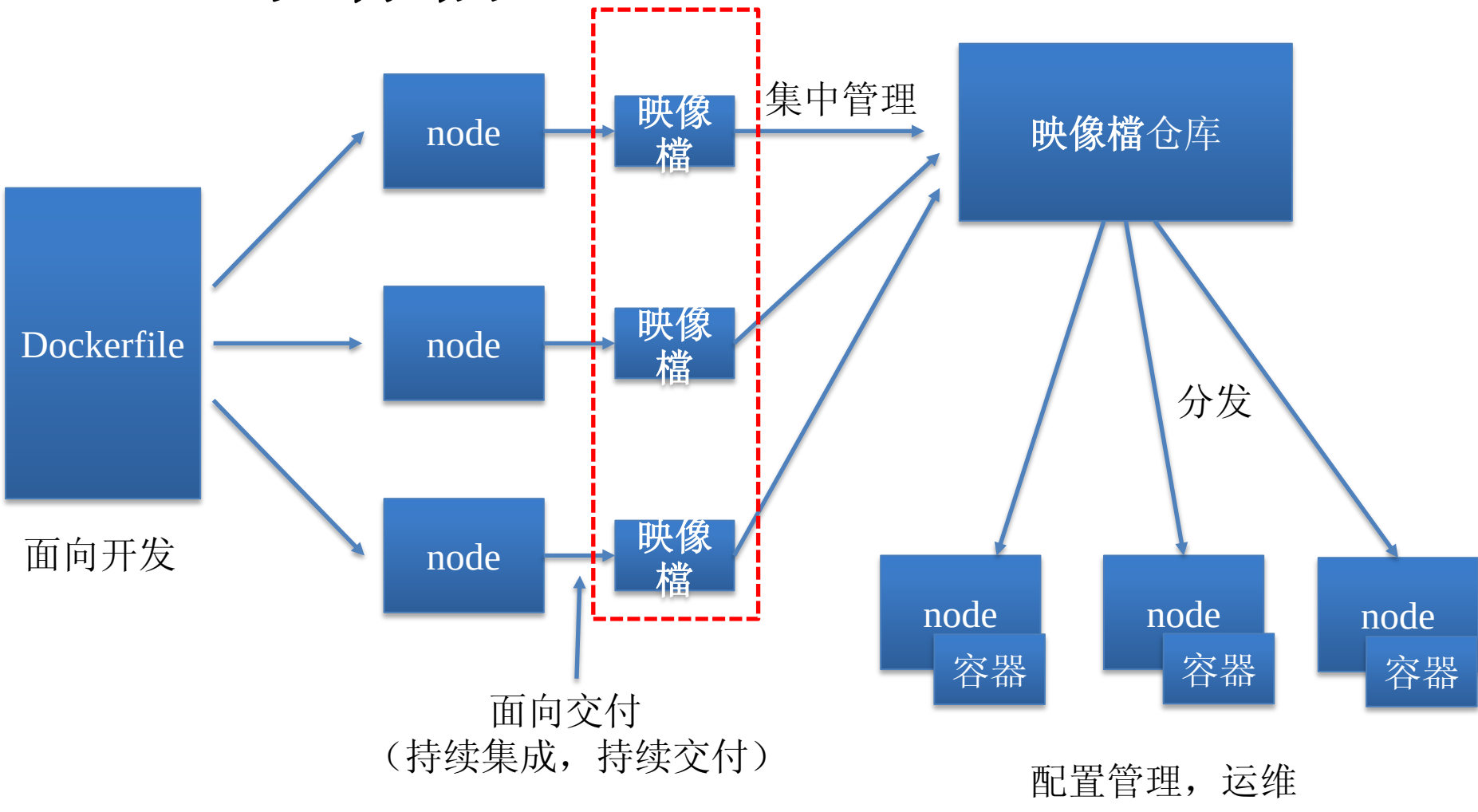
Dockerfile命令与映像檔层一一对应

映像檔拥有强移植性

汇聚于Docker Registry



Docker映像檔



Docker映像檔

1.映像檔layer
每一層映像檔中文檔內容

```
root@DaoCloud:~# docker history ubuntu:14.04
IMAGE               CREATED             CREATED BY                                      SIZE
d2a0ecffe6fa        5 weeks ago         /bin/sh -c #(nop) CMD ["/bin/bash"]             0 B
29460ac93442        5 weeks ago         /bin/sh -c sed -i 's/^#\s*\(deb.*universe\)$/   1.895 kB
b670fb0c7ecd        5 weeks ago         /bin/sh -c echo '#!/bin/sh' > /usr/sbin/polic   194.5 kB
83e4dde6b9cf        5 weeks ago         /bin/sh -c #(nop) ADD file:c8f078961a543cdefa   188.2 MB
root@DaoCloud:~#
```

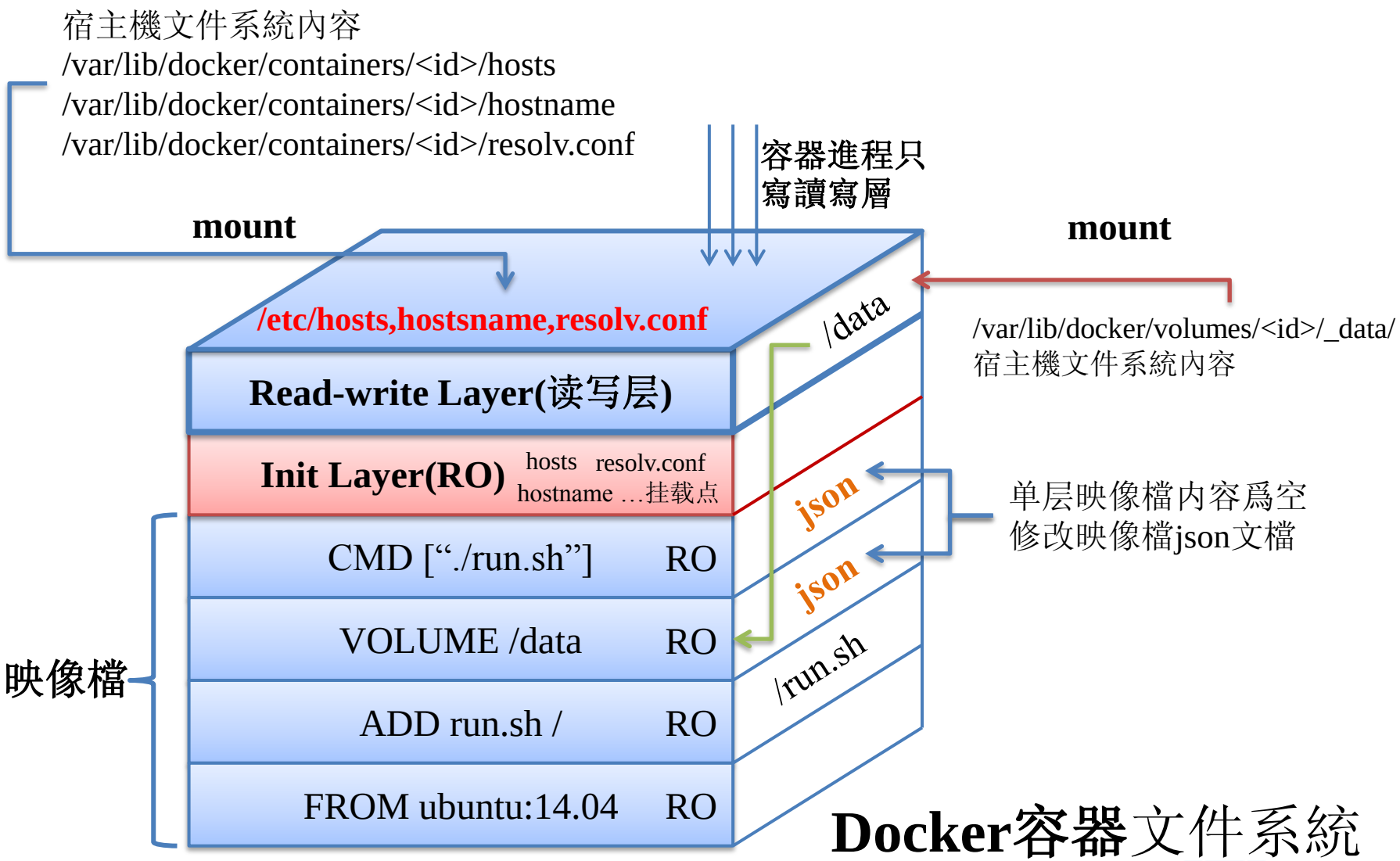
```
root@DaoCloud:/var/lib/docker/aufs/diff# ls | xargs ls
29460ac934423a55802fcad24856827050697b4a9f33550bd93c82762fb6db8f:   1
etc

83e4dde6b9cfddf46b75a07ec8d65ad87a748b98cf27de7d5b3298c1f3455ae4:                2
bin   boot   dev   etc   home   lib   lib64   media   mnt   opt   proc   root   run   sbin   srv   sys   tmp   usr   var

b670fb0c7ecd3d2c401fbfd1fa4d7a872fbada0a4b8c2516d0be18911c6b25d6:   3
etc   sbin   usr   var

d2a0ecffe6fa4ef3de9646a75cc629bbd9da7eead7f767cb810f9808d6b3ecb6:   4
root@DaoCloud:/var/lib/docker/aufs/diff#
```

Docker映像檔

2.映像檔json文件

每一層映像檔的描述信息,記錄映像檔的變化

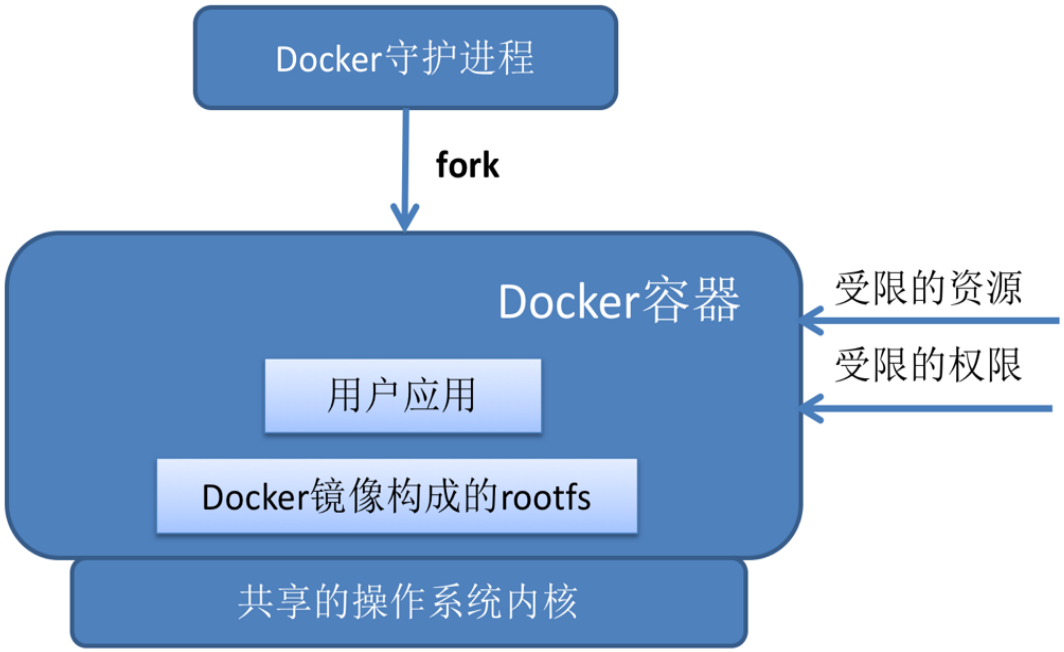
```
root@DaoCloud:/var/lib/docker/graph# ls | xargs ls
29460ac934423a55802fcad24856827050697b4a9f33550bd93c82762fb6db8f:
json  layersize
root@DaoCloud:/var/lib/docker/graph/d2a0ecffe6fa4ef3de9646a75cc629bbd9da7eead7f767cb810f9808d6b3ecb6# cat json
{"id":"d2a0ecffe6fa4ef3de9646a75cc629bbd9da7eead7f767cb810f9808d6b3ecb6","parent":"29460ac934423a55802fcad2485682705069
7b4a9f33550bd93c82762fb6db8f","created":"2015-07-09T19:28:34.439309646Z","container":"1d6ee792a30e4dda034f8b72fb1a7f8e9
aebe4f6e6feeaa6dcc88ac4b66ee744","container_config":{"Hostname":"dd360632d03c","Domainname":"","User":"","AttachStdin":
false,"AttachStdout":false,"AttachStderr":false,"PortSpecs":null,"ExposedPorts":null,"Tty":false,"OpenStdin":false,"Std
inOnce":false,"Env":null,"Cmd":["/bin/sh","-c","#(nop) CMD [\"/bin/bash\"]"],"Image":"29460ac934423a55802fcad2485682705
0697b4a9f33550bd93c82762fb6db8f","Volumes":null,"VolumeDriver":"","WorkingDir":"","Entrypoint":null,"NetworkDisabled":f
alse,"MacAddress":"","OnBuild":null,"Labels":{}},"docker_version":"1.6.2","config":{"Hostname":"dd360632d03c","Domainna
me":"","User":"","AttachStdin":false,"AttachStdout":false,"AttachStderr":false,"PortSpecs":null,"ExposedPorts":null,"Tt
y":false,"OpenStdin":false,"StdinOnce":false,"Env":null,"Cmd":["/bin/bash"],"Image":"29460ac934423a55802fcad24856827050
697b4a9f33550bd93c82762fb6db8f","Volumes":null,"VolumeDriver":"","WorkingDir":"","Entrypoint":null,"NetworkDisabled":fa
lse,"MacAddress":"","OnBuild":null,"Labels":{}},"architecture":"amd64","os":"linux","Size":0}
root@DaoCloud:/var/lib/docker/graph/d2a0ecffe6fa4ef3de9646a75cc629bbd9da7eead7f767cb810f9808d6b3ecb6#
json  layersize
```



Docker容器文件系統

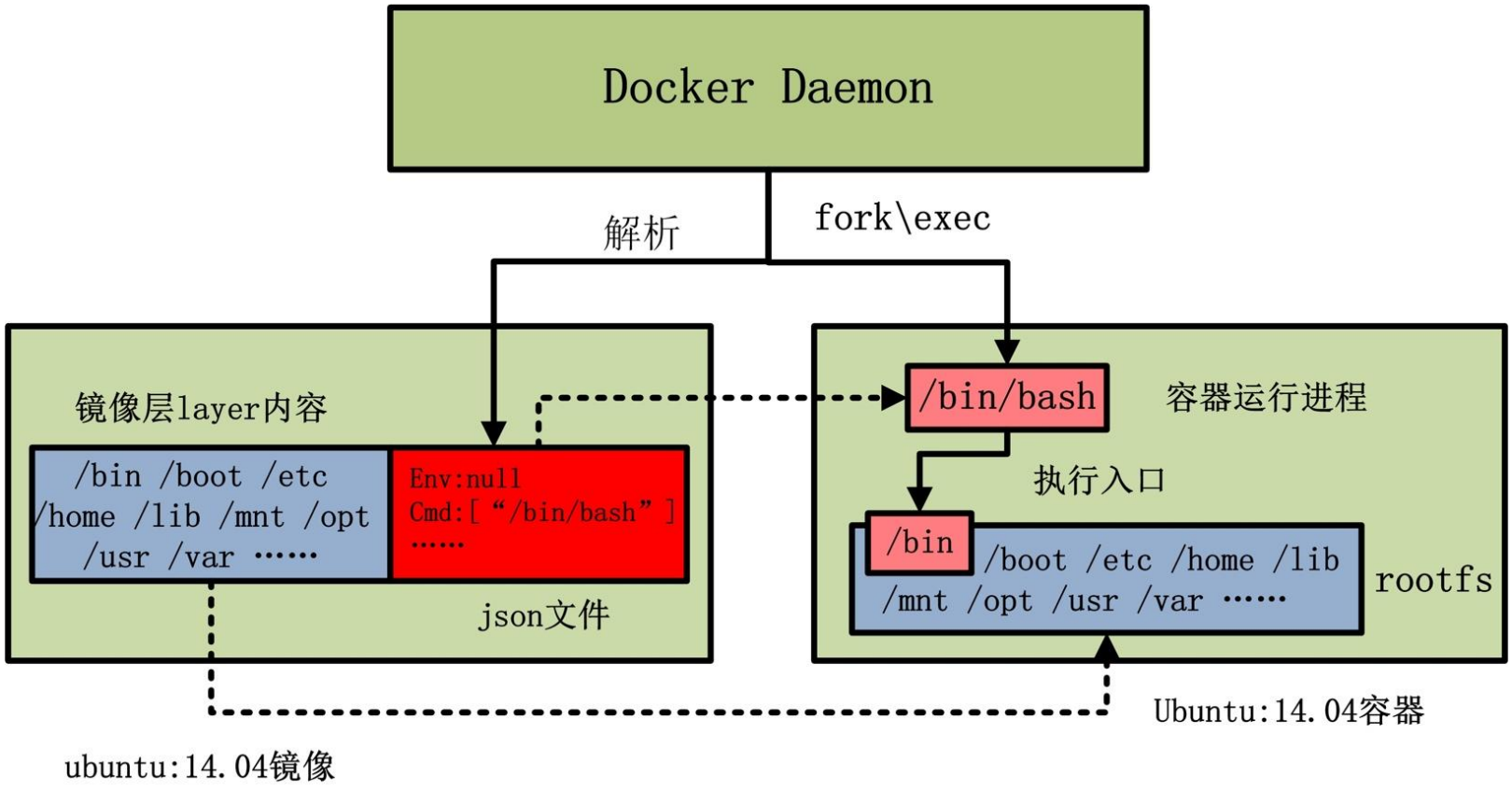
Docker容器

- Docker服務的最終交付件
- 受到資源的隔離與控制
- 受到權限的控制
- 支撐用戶應用的運行
- 本質是Linux進程
- Docker守護進程的子進程

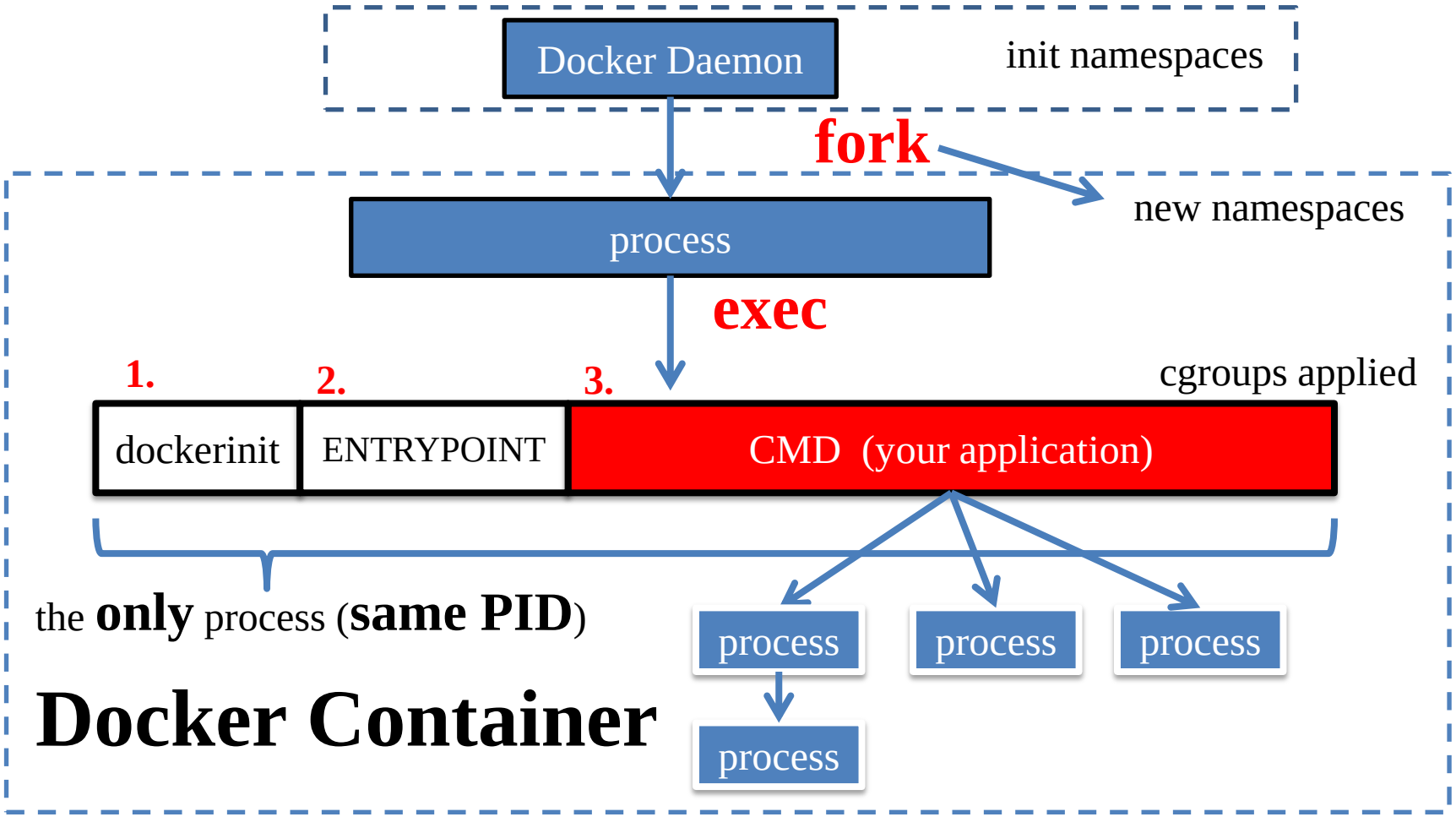


Docker容器

运行Docker映像檔



Docker容器


Docker Daemon
init namespaces
fork
new namespaces
process
exec
1.
2.
3.
cgroups applied
dockerinit
ENTRYPOINT
CMD (your application)
the only process (same PID)
process
process
process
process
Docker Container

iThome

Container Summit 2015

航向容器新世界

Thank you