Docker 目录结构

Docker的镜像和容器都存在在/var/lib/docker下面，不同的Linux系统有不同的存储方式，在Centos下面存储方式为device mapping。在Docker的使用过程中，分为三个阶段：

1）Start docker服务之后的目录结构

2）pull images后的目录结构

3）run container后的目录结构

下面对三个阶段的目录结构进行分别介绍

# 1.启动Docker daemon

通过下面的命令：

*[root@fys2 ~]# service docker*

*Usage: /etc/init.d/docker {start|stop|status|restart|condrestart|try-restart|reload|force-reload}*

启动docker daemon，在/var/lib/docker中目录如下（不启动docker daemon，该目录为NULL）：

*[root@sgserver003 docker]# tree .*

.

├── containers

├── devicemapper *//centos使用devicemapper的存储方式*

│   ├── devicemapper

│   │   ├── data *//存储池*

│   │   └── metadata *//元数据*

│   └── metadata  *//元数据的id、大小以及UUID等信息*

│   ├── base

│   └── transaction-metadata

├── graph

├── init

│   └── dockerinit-1.7.1

├── linkgraph.db

├── repositories-devicemapper

├── tmp

├── trust

└── volumes

9 directories, 7 files

目前docker的目录为一个空daemon，还没有数据

# 2. Pull images

通过docker pull命令下载一个docker images进行查看，查看目录及对比之前的目录结构及文件变化：

*[root@sgserver003 ~]# docker images*

*REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE*

*centos centos6 eb24acd8692d 12 days ago 194.6 MB*

目录及文件如下所示：

[root@sgserver003 docker]# tree ./

./

├── containers

├── devicemapper

│   ├── devicemapper

│   │   ├── data

│   │   └── metadata

│   ├── metadata

│   │   ├── 3690474eb5b4b26fdfbd89c6e159e8cc376ca76ef48032a30fa6aafd56337880

│   │   ├── base

│   │   ├── dc197f564cca177da594076ef729eb76eb4b2220cfd502b4146329b7c114b85a

│   │   ├── eb24acd8692d2d6856afb32dc0b8709c28caf1ef6182ab3975c95a275ffd8c35

│   │   ├── f6b5e489a059139ed5be5142e17d5ebb45361ce91d14550a7a68e8f20b54678d

│   │   └── transaction-metadata

│   └── mnt

│   ├── 3690474eb5b4b26fdfbd89c6e159e8cc376ca76ef48032a30fa6aafd56337880

│   ├── dc197f564cca177da594076ef729eb76eb4b2220cfd502b4146329b7c114b85a

│   ├── eb24acd8692d2d6856afb32dc0b8709c28caf1ef6182ab3975c95a275ffd8c35

│   └── f6b5e489a059139ed5be5142e17d5ebb45361ce91d14550a7a68e8f20b54678d

├── graph

│   ├── 3690474eb5b4b26fdfbd89c6e159e8cc376ca76ef48032a30fa6aafd56337880

│   │   ├── checksum

│   │   ├── json

│   │   └── layersize

│   ├── dc197f564cca177da594076ef729eb76eb4b2220cfd502b4146329b7c114b85a

│   │   ├── checksum

│   │   ├── json

│   │   └── layersize

│   ├── eb24acd8692d2d6856afb32dc0b8709c28caf1ef6182ab3975c95a275ffd8c35

│   │   ├── checksum

│   │   ├── json

│   │   └── layersize

│   ├── f6b5e489a059139ed5be5142e17d5ebb45361ce91d14550a7a68e8f20b54678d

│   │   ├── checksum

│   │   ├── json

│   │   └── layersize

│   └── \_tmp

├── init

│   └── dockerinit-1.7.1

├── linkgraph.db

├── repositories-devicemapper

├── tmp

├── trust

└── volumes

19 directories, 23 files

当pull一个镜像后，子目录中会增加了许多文件，最先看到的变化时下面三个文件夹下面

/var/lib/docker/devicemapper/metadata

/var/lib/docker/devicemapper/mnt

/var/lib/docker/graph

在4个目录中的增加了几个目录，包括隐藏的中间image，通过命令：

*[root@sgserver003 graph]# docker images -a*

*REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE*

*centos centos6 eb24acd8692d 12 days ago 194.6 MB*

*<none> <none> dc197f564cca 12 days ago 194.6 MB*

*<none> <none> f6b5e489a059 12 days ago 194.6 MB*

*<none> <none> 3690474eb5b4 2 weeks ago 0 B*

在/var/lib/docker/devicemapper/metadata的目录下的内容：

*[root@sgserver003 metadata]# cat base*

*{"device\_id":1,"size":10737418240,"transaction\_id":1,"initialized":true}*

*[root@sgserver003 metadata]#*

*cat 3690474eb5b4b26fdfbd89c6e159e8cc376ca76ef48032a30fa6aafd56337880*

*{"device\_id":2,"size":10737418240,"transaction\_id":2,"initialized":false}*

*[root@sgserver003 metadata]#*

*cat f6b5e489a059139ed5be5142e17d5ebb45361ce91d14550a7a68e8f20b54678d*

*{"device\_id":3,"size":10737418240,"transaction\_id":3,"initialized":false}*

*[root@sgserver003 metadata]#*

*cat dc197f564cca177da594076ef729eb76eb4b2220cfd502b4146329b7c114b85a*

*{"device\_id":4,"size":10737418240,"transaction\_id":4,"initialized":false}*

*[root@sgserver003 metadata]#*

*cat eb24acd8692d2d6856afb32dc0b8709c28caf1ef6182ab3975c95a275ffd8c35*

*{"device\_id":5,"size":10737418240,"transaction\_id":5,"initialized":false}*

可以看到devid\_id，按照（1,2,3,4,5）排序，除了base文件外，其他文件都是刚才添加的中间件，该目录下的文件（除了base,deviceset-metadata,tranaction-metadata），其他的文件都是images本身和images的中间信息，用来描述image的id,大小,transactin\_id以及initialized，并且大小一致

另外一个目录：/var/lib/docker/devicemapper/mnt，它主要用来挂载images和container的目录，因为devicemapping本身就是通过在存储池中挂载的方式进行运行的。

最后一个目录：/var/lib/docker/graph，目录结构如下所示：

[root@sgserver003 graph]# tree ./

./

├── 3690474eb5b4b26fdfbd89c6e159e8cc376ca76ef48032a30fa6aafd56337880

│   ├── checksum

│   ├── json

│   └── layersize

├── dc197f564cca177da594076ef729eb76eb4b2220cfd502b4146329b7c114b85a

│   ├── checksum

│   ├── json

│   └── layersize

├── eb24acd8692d2d6856afb32dc0b8709c28caf1ef6182ab3975c95a275ffd8c35

│   ├── checksum

│   ├── json

│   └── layersize

├── f6b5e489a059139ed5be5142e17d5ebb45361ce91d14550a7a68e8f20b54678d

│   ├── checksum

│   ├── json

│   └── layersize

└── \_tmp

每个images本身及中间件下面多个三个文件，分别为(json、layersize及checksum)

1) json //json文件用来描述images本身或者中间件的详细信息

{

"id": "3690474eb5b4b26fdfbd89c6e159e8cc376ca76ef48032a30fa6aafd56337880",

"created": "2016-08-30T18:18:45.109943798Z",

"container\_config": {

"Hostname": "",

"Domainname": "",

"User": "",

"AttachStdin": false,

"AttachStdout": false,

"AttachStderr": false,

"PortSpecs": null,

"ExposedPorts": null,

"Tty": false,

"OpenStdin": false,

"StdinOnce": false,

"Env": null,

"Cmd": [

"/bin/sh -c #(nop) MAINTAINER https://github.com/CentOS/sig-cloud-instance-images"],

"Image": "",

"Volumes": null,

"VolumeDriver": "",

"WorkingDir": "",

"Entrypoint": null,

"NetworkDisabled": false,

"MacAddress": "",

"OnBuild": null,

"Labels": null

},

"author": "https://github.com/CentOS/sig-cloud-instance-images",

"Size": 0

}

2）layersize，表示中间image的大小

3）checksum，校验码

在/var/lib/docker下的文件repositories-devicemapper

cat repositories-devicemapper

{

"Repositories": {

"centos": {

"centos6": "eb24acd8692d2d6856afb32dc0b8709c28caf1ef6182ab3975c95a275ffd8c35"

}

}

}

记录了image名称、tag和镜像ID等信息

上面是pull images操作后变化的目录，基本上是images的信息，下面是一些基本没有变化的目录：

/var/lib/docker/container

/var/lib/docker/tmp

/var/lib/docker/trust

/var/lib/docker/volumes

# 3. Run Container

使用docker run命令启动container

*[root@sgserver003 docker]# docker run -d centos:centos6*

*0ac1e6b833ac04e0ec59db6e5f9331b88a3f88e8fe37660059df635a2fbdbc20*

*[root@sgserver003 docker]# docker ps*

*CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES*

*a0504ca6f729 centos:centos6 "/bin/bash" 3 seconds ago Up 2 seconds trusting\_noyce*

目录(分目录)的总体变化为：

/var/lib/docker/containers/

├── 0ac1e6b833ac04e0ec59db6e5f9331b88a3f88e8fe37660059df635a2fbdbc20

│   ├── 0ac1e6b833ac04e0ec59db6e5f9331b88a3f88e8fe37660059df635a2fbdbc20-json.log

│   ├── config.json

│   ├── hostconfig.json

│   ├── hostname

│   ├── hosts

│   ├── resolv.conf

│   └── resolv.conf.hash

└── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805

├── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805-json.log

├── config.json

├── hostconfig.json

├── hostname

├── hosts

├── resolv.conf

└── resolv.conf.hash

/var/lib/docker/devicemapper/

├── devicemapper

│   ├── data

│   └── metadata

├── metadata

│   ├── 0ac1e6b833ac04e0ec59db6e5f9331b88a3f88e8fe37660059df635a2fbdbc20

│   ├── 0ac1e6b833ac04e0ec59db6e5f9331b88a3f88e8fe37660059df635a2fbdbc20-init

│   ├── 3690474eb5b4b26fdfbd89c6e159e8cc376ca76ef48032a30fa6aafd56337880

│   ├── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805

│   ├── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805-init

│   ├── base

│   ├── c6ba6252263bc1d5f23709f9c30755ff3d0ce0532a32581b9c60c35d9756e5fc

│   ├── dc197f564cca177da594076ef729eb76eb4b2220cfd502b4146329b7c114b85a

│   ├── eb24acd8692d2d6856afb32dc0b8709c28caf1ef6182ab3975c95a275ffd8c35

│   ├── f6b5e489a059139ed5be5142e17d5ebb45361ce91d14550a7a68e8f20b54678d

│   └── transaction-metadata

└── mnt

├── 0ac1e6b833ac04e0ec59db6e5f9331b88a3f88e8fe37660059df635a2fbdbc20

├── 0ac1e6b833ac04e0ec59db6e5f9331b88a3f88e8fe37660059df635a2fbdbc20-init

├── 3690474eb5b4b26fdfbd89c6e159e8cc376ca76ef48032a30fa6aafd56337880

├── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805

│   ├── id

│   ├── lost+found

│   └── rootfs

│   ├── bin

│   │   ├── arch

│   │   ├── awk -> gawk

│   │   ├── basename

│   │   ├── bash

│   │   ├── cat

│   │   ├── chgrp

│   │   ├── chmod

│   │   ├── chown

│   │   ├── cp

....................................................

.....................................................

│   ├── home

│   ├── lib

│   │   ├── firmware

│   │   ├── modules

│   │   ├── security

│   │   ├── terminfo

│   │   │   ├── a

│   │   │   │   └── ansi

│   │   │   ├── d

│   │   │   │   └── dumb

│   │   │   ├── l

│   │   │   │   └── linux

│   │   │   └── v

│   │   │   ├── vt100

│   │   │   ├── vt100-am

│   │   │   ├── vt100-nav

│   │   │   ├── vt102

│   │   │   ├── vt200

│   │   │   ├── vt220

│   │   │   └── vt52

├── dc197f564cca177da594076ef729eb76eb4b2220cfd502b4146329b7c114b85a

├── eb24acd8692d2d6856afb32dc0b8709c28caf1ef6182ab3975c95a275ffd8c35

└── f6b5e489a059139ed5be5142e17d5ebb45361ce91d14550a7a68e8f20b54678d

/var/lib/docker/graph

├── 3690474eb5b4b26fdfbd89c6e159e8cc376ca76ef48032a30fa6aafd56337880

│   ├── checksum

│   ├── json

│   └── layersize

├── dc197f564cca177da594076ef729eb76eb4b2220cfd502b4146329b7c114b85a

│   ├── checksum

│   ├── json

│   └── layersize

├── eb24acd8692d2d6856afb32dc0b8709c28caf1ef6182ab3975c95a275ffd8c35

│   ├── checksum

│   ├── json

│   └── layersize

├── f6b5e489a059139ed5be5142e17d5ebb45361ce91d14550a7a68e8f20b54678d

│   ├── checksum

│   ├── json

│   └── layersize

└── \_tmp

└── a4d22e38c2dfacb0473e019659122377856a61dff2293b7b4f028f67d6db7551

/var/lib/docker/tmp

├── GetV2ImageBlob338961026

├── GetV2ImageBlob755489488

└── GetV2ImageBlob923887599

和pull images的目录对比，有变化主要有3个文件：

/var/lib/docker/devicemapper/medata

/var/lib/docker/devicemapper/mnt

/var/lib/docker/container

1）/var/lib/docker/devicemapper/metadata

metadata/

├── 3690474eb5b4b26fdfbd89c6e159e8cc376ca76ef48032a30fa6aafd56337880

├── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805

├── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805-init

├── base

├── c6ba6252263bc1d5f23709f9c30755ff3d0ce0532a32581b9c60c35d9756e5fc

├── dc197f564cca177da594076ef729eb76eb4b2220cfd502b4146329b7c114b85a

├── eb24acd8692d2d6856afb32dc0b8709c28caf1ef6182ab3975c95a275ffd8c35

├── f6b5e489a059139ed5be5142e17d5ebb45361ce91d14550a7a68e8f20b54678d

└── transaction-metadata

和pull images相比多了下面两个文件：

├── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805

├── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805-init

当docker运行一个从镜像建立的容器，会在镜像顶部添加一个可读写的层，应用程序可以在这里运行

2）/var/lib/docker/devicemapper/mnt

和metadata中的文件一一，也新增加目录

├── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805

│   ├── id

│   ├── lost+found

│   └── rootfs

│   ├── bin

│   │   ├── arch

│   │   ├── awk -> gawk

│   │   ├── basename

│   │   ├── bash

│   │   ├── cat

│   │   ├── chgrp

│   │   ├── chmod

│   │   ├── chown

.........................................

├── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805-init

其中第一个目录为启动的container的运行目录，包括操作系统数据及写入数据等

3）/var/lib/docker/containers

containers/

└── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805

├── a0504ca6f729fb8167e5f2285533862e63b04ff401e324940f992785bfc33805-json.log

├── config.json //配置文件

├── hostconfig.json

├── hostname //主机名

├── hosts

├── resolv.conf //dns解析

└── resolv.conf.hash

增加的文件均是用来描述该容器的信息以及数据，当stop该container后，该目录被删除

# 4. 总结如下

/var/lib/docker/devicemapper/devicemapper/data #用来存储相关的存储池数据

/var/lib/docker/devicemapper/devicemapper/metadata #用来存储相关的元数据。

/var/lib/docker/devicemapper/metadata/

#用来存储 device\_id、大小、以及传输\_id、初始化信息

/var/lib/docker/devicemapper/mnt #用来存储挂载信息

/var/lib/docker/container/ #用来存储容器信息

/var/lib/docker/graph/

#用来存储镜像中间件及本身详细信息和大小 、以及依赖信息

/var/lib/docker/repositores-devicemapper #用来存储镜像基本信息

/var/lib/docker/tmp #docker临时目录

/var/lib/docker/trust #docker信任目录

/var/lib/docker/volumes #docker卷目录

# 5. 实例

启动了一个Docker container，其中运行了Ambari，并启动了zookeeper和hdfs，在docker中相关目录如下：

**1）/var/lib/docker/containers**

*[root@fys1 containers]#*

cd b3d48a60bbfb5c1f45254a445f7bc9699729d098a75d297ed97cbfcd3d1d5359/

[root@fys1 b3d48a60bbfb5c1f45254a445f7bc9699729d098a75d297ed97cbfcd3d1d5359]# ls

b3d48a60bbfb5c1f45254a445f7bc9699729d098a75d297ed97cbfcd3d1d5359-json.log config.json hostconfig.json hostname hosts resolv.conf resolv.conf.hash

*[root@fys1 b3d48a60bbfb5c1f45254a445f7bc9699729d098a75d297ed97cbfcd3d1d5359]# ll*

total 40

-rw------- 1 root root 13135 Sep 18 21:20

b3d48a60bbfb5c1f45254a445f7bc9699729d098a75d297ed97cbfcd3d1d5359-json.log

-rw-r--r-- 1 root root 3902 Sep 18 21:19 config.json

-rw-r--r-- 1 root root 591 Sep 18 21:19 hostconfig.json

-rw-r--r-- 1 root root 13 Sep 18 21:19 hostname

-rw-r--r-- 1 root root 175 Sep 18 21:19 hosts

-rw-r--r-- 1 root root 90 Sep 18 21:19 resolv.conf

-rw------- 1 root root 71 Sep 18 21:19 resolv.conf.hash

**2）/var/lib/docker/aufs/mnt/**

*[root@fys1 b3d48a60bbfb5c1f45254a445f7bc9699729d098a75d297ed97cbfcd3d1d5359]# pwd*

/var/lib/docker/aufs/mnt/b3d48a60bbfb5c1f45254a445f7bc9699729d098a75d297ed97cbfcd3d1d5359

*[root@fys1 b3d48a60bbfb5c1f45254a445f7bc9699729d098a75d297ed97cbfcd3d1d5359]# ll*

total 88

dr-xr-xr-x 2 root root 4096 Aug 27 06:03 bin

drwxr-xr-x 3 root root 4096 Aug 27 06:03 boot

drwxr-xr-x 3 root root 4096 Sep 18 19:59 cmss

drwxr-xr-x 4 root root 4096 Sep 18 19:48 dev

drwxr-xr-x 108 root root 4096 Sep 18 20:15 etc

drwxr-xr-x 4 root root 4096 Sep 18 20:01 hadoop

drwxr-xr-x 5 root root 4096 Sep 18 19:56 home

dr-xr-xr-x 8 root root 4096 Aug 27 06:03 lib

dr-xr-xr-x 6 root root 4096 Sep 18 19:57 lib64

drwx------ 2 root root 4096 Jun 2 21:42 lost+found

drwxr-xr-x 2 root root 4096 Sep 23 2011 media

drwxr-xr-x 2 root root 4096 Sep 23 2011 mnt

drwxr-xr-x 4 root root 4096 Sep 15 08:58 opt

drwxr-xr-x 2 root root 4096 Sep 18 19:48 proc

dr-xr-x--- 5 root root 4096 Sep 19 11:07 root

dr-xr-xr-x 2 root root 4096 Aug 27 06:04 sbin

drwxr-xr-x 2 root root 4096 Jun 2 21:42 selinux

drwxr-xr-x 2 root root 4096 Sep 23 2011 srv

drwxr-xr-x 2 root root 4096 Sep 18 19:48 sys

drwxrwxrwt 13 root root 4096 Sep 18 21:25 tmp

drwxr-xr-x 49 root root 4096 Sep 18 20:01 usr

drwxr-xr-x 52 root root 4096 Sep 18 19:56 var

**3）/var/lib/docker/aufs/layers**

*[root@fys1 layers]# //每个image都会有多层images*

cat 3b78d9d5891b4fbec68a2d0e6557f61295bf55ed4ffc27d4e53910b62cef4c12

d362ac3544123243ae675ebf66d8e8f9c7bd9a45bde9ea77c7cebc0f832bc488

34c9f33be456155c9e62396ec23c70091413d02251d9500c26f3ac29bb216aee

0659986da3424060b5ce82a6125b3cb3b94a2e587ae370cf65088ad3e538686e

0c5f61621f373785bd90a755c9176564179eed1f37678181c43e901e598dff5f

5345f8a145ae42aa87c15a9e6b77c2a1d14f7f11357e177beaae9bbeece4f2b7

ba140e8bd5e6e0a7d635435fd9a8fa2ed2ff7e2fa6dde3fb719164315a321620

e5207061dc82a4032d1d669dd94e014dd26bab8c7f502644e43aea62ff415d06

e636d1b431b6912567760d7f7cf3b306fc17555759deae90248570e8944215de

c9f50820ffcfab60479bded8681ed34499c67c883b950a61720902e71b572262

be4dd2049f36d4ee685f171b160413d74e2aabbba901a70ab169f0ece4d9280e

..................

*[root@fys1 local]# pwd*

/var/lib/docker/aufs/diff/3b78d9d5891b4fbec68a2d0e6557f61295bf55ed4ffc27d4e53910b62cef4c12/usr/local

*[root@fys1 local]# ll*

total 4

drwxr-xr-x 3 10021 10021 4096 Dec 14 2015 hadoop-2.7.1

**4) /var/lib/docker/graph/**

[root@fys1 3b78d9d5891b4fbec68a2d0e6557f61295bf55ed4ffc27d4e53910b62cef4c12]# ls

json layersize

[root@fys1 3b78d9d5891b4fbec68a2d0e6557f61295bf55ed4ffc27d4e53910b62cef4c12]# ll

total 8

-rw------- 1 root root 2198 Jul 6 22:00 json

-rw------- 1 root root 4 Jul 6 22:00 layersize

*总结：运行的docker相关的数据包括两个方面：image，基本image和在image上进行的操作；container的存储映射，保存在mnt中，例如centos image中包括操作系统的基本结构以及写到系统中的数据*