Docker 入门教程：Union File System 在 Docker 的应用

我们在 Docker 入门教程：镜像分层 和 Docker 入门教程：Docker 基础技术 Union File System 已经介绍了一些前提基础知识，本文我们来介绍 Union File System 在 Docker 的应用。

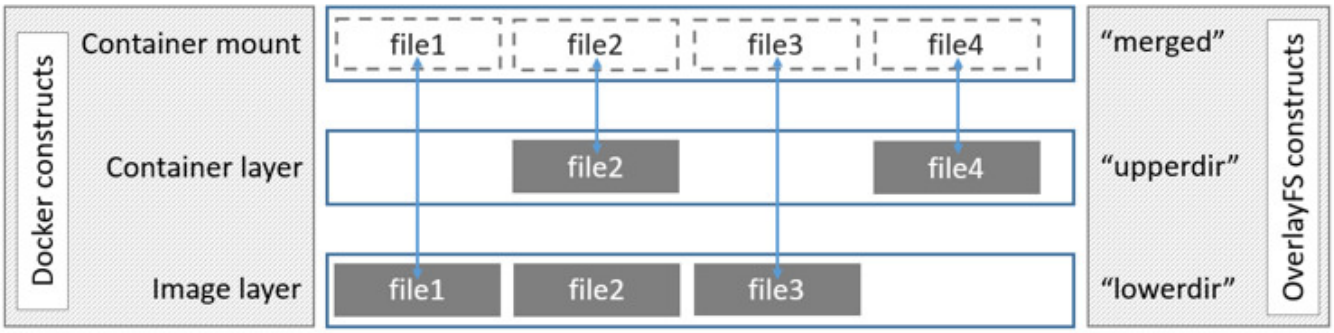
为了使 Docker 能够在 container 的 writable layer 写一些比较小的数据（如果需要写大量的数据可以通过挂载盘去写），Docker 为我们实现了存储驱动（storage drivers）。Docker 使用插件机制支持多种 storage drivers，storage driver 用来控制镜像和容器在你的电脑里面是如何存储和管理的。

截止到 Docker v19.03 版本，Docker 支持 overlay2、aufs、devicemapper、btrfs、zfs 以及 vfs 等存储驱动。有些 Linux 系统支持多种存储驱动，在默认情况下，Docker 按照 btrfs,zfs,overlay2,aufs,overlay,devicemapper,vfs 的顺序去选择存储驱动。

下面以 CentOS 系统为例介绍 Docker 是如何使用 overlay2 管理我们的镜像和容器的。注意，Linux kernel 4.0 及以上，CentOS 3.10.0-514 及以上都支持 OverlayFS。如果你系统也支持 overlay，那么请尽量使用 overlay2，因为相比 overlay 它在 inode 利用率方面更有效。

overlay2 驱动是如何工作的

OverlayFS 的相关知识我们已经在 Docker 入门教程：Docker 基础技术 Union File System 介绍过了，这里我就不再介绍了。下图很清楚的显示了 Docker 镜像和 Docker 容器是如何分层的。镜像层属于 lowerdir，容器层属于 upperdir。这两层合并（merge）之后得到的目录叫做 merged：



镜像和容器层在磁盘是如何存储的

在介绍之前，我们先来下载 ubuntu 镜像：

```
[root@iteblog.com test]$ docker pull ubuntu
```

```
Using default tag: latest
latest: Pulling from library/ubuntu
5c939e3a4d10: Pull complete
c63719cdbe7a: Pull complete
19a861ea6baf: Pull complete
651c9d2d6c4f: Pull complete
Digest: sha256:8d31dad0c58f552e890d68bbfb735588b6b820a46e459672d96e585871acc110
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

可以看出 Ubuntu 镜像一共有四层，如果你使用的是 overlay2，那么 Docker 默认把这些每层的镜像文件放在 /var/lib/docker/overlay2 目录下。我们也可以使用 docker image inspect ubuntu 命令查看镜像的详细信息：

```
docker image inspect ubuntu
[
   {
      "Id": "sha256:ccc6e87d482b79dd1645affd958479139486e47191dfe7a997c862d89cd8b4c
0",
      "RepoTags": [
         "ubuntu:latest"
      ],
      ....
      "GraphDriver": {
         "Data": {
            "LowerDir": "/var/lib/docker/overlay2/4c521fd94cf4a09ab871d7c2b48817a8c3e6033f
9f80e1ce6dd2d5a16e61021c/diff:/var/lib/docker/overlay2/aaad926478d067f54935bd6f84ec8f
38dc551c98fb4d1bdae7357d4186d183ac/diff:/var/lib/docker/overlay2/4b495e45dfb3c46fd0c1
3279a55432c772d67a7eedfd47deafd8187f218b6c29/diff",
            "MergedDir": "/var/lib/docker/overlay2/7643e9e3b8404ebef6d0a801d0007f13b5fa57
646cb622d60860032e5eb0a382/merged",
            "UpperDir": "/var/lib/docker/overlay2/7643e9e3b8404ebef6d0a801d0007f13b5fa576
46cb622d60860032e5eb0a382/diff",
            "WorkDir": "/var/lib/docker/overlay2/7643e9e3b8404ebef6d0a801d0007f13b5fa5764
6cb622d60860032e5eb0a382/work"
         },
         "Name": "overlay2"
      },
      "RootFS": {
         "Type": "layers",
         "Layers": [
            "sha256:43c67172d1d182ca5460fc962f8f053f33028e0a3a1d423e05d91b532429e73d
",
```

```
            "sha256:21ec61b65b20ec53a1b7f069fd04df5acb0e75434bd3603c88467c8bfc80d9c6
",
            "sha256:1d0dfb259f6a31f95efcba61f0a3afa318448890610c7d9a64dc4e95f9add843",
            "sha256:f55aa0bd26b801374773c103bed4479865d0e37435b848cb39d164ccb2c3ba5
1"
         ]
      },
      "Metadata": {
         "LastTagTime": "0001-01-01T00:00:00Z"
      }
   }
]
```


可以看出上面命令详细地显示了 Ubuntu 一共有四层（参见 Layers ），GraphDriver.Data 信息里面显示了镜像的数据目录。里面有 LowerDir、MergedDir、UpperDir 以及 WorkDir 的信息。我们可以看下 /var/lib/docker/overlay2/ 目录下的文件：

```
[root@iteblog.com test]$ ll /var/lib/docker/overlay2/
总用量 20
drwx------ 3 root root 4096 2月   9 21:43 4b495e45dfb3c46fd0c13279a55432c772d67a7eedfd47
deafd8187f218b6c29
drwx------ 4 root root 4096 2月   9 21:43 4c521fd94cf4a09ab871d7c2b48817a8c3e6033f9f80e1c
e6dd2d5a16e61021c
drwx------ 4 root root 4096 2月   9 21:43 7643e9e3b8404ebef6d0a801d0007f13b5fa57646cb62
2d60860032e5eb0a382
drwx------ 4 root root 4096 2月   9 21:43 aaad926478d067f54935bd6f84ec8f38dc551c98fb4d1b
dae7357d4186d183ac
drwx------ 2 root root 4096 2月   9 21:43 l
[root@iteblog.com test]$
```

可以看出，输出了5个目录，除了名为 l 的目录，其他目录都是每个镜像的存放目录。名为 l 的目录是 overlay2 新加了，里面主要存放每个镜像文件里面 diff 目录的链接文件。主要是避免执行 mount 命令时参数过长而出现问题。

```
[root@iteblog.com /var/lib/docker/overlay2]$ ll l
总用量 16
lrwxrwxrwx 1 root root 72 2月   9 21:43 3XQB7H6IZZD3I7GYHEZANUBTXO -> ../4b495e45dfb3c
46fd0c13279a55432c772d67a7eedfd47deafd8187f218b6c29/diff
lrwxrwxrwx 1 root root 72 2月   9 21:43 LGEY4HPN6RJO7T7K7TTJUHRKT3 -> ../4c521fd94cf4a09
ab871d7c2b48817a8c3e6033f9f80e1ce6dd2d5a16e61021c/diff
```

lrwxrwxrwx 1 root root 72 2月 9 21:43 QPASHTGWQBPJ3B7WUCOJLGLNFK -> ../7643e9e3b8404ebef6d0a801d0007f13b5fa57646cb622d60860032e5eb0a382/diff
lrwxrwxrwx 1 root root 72 2月 9 21:43 TFOUROFQV2HLTKCYRKEJ37MXXZ -> ../aaad926478d067f54935bd6f84ec8f38dc551c98fb4d1bdae7357d4186d183ac/diff

接下来我们来看下每个镜像文件里面都有什么：

```
[root@iteblog.com /var/lib/docker/overlay2]$ ll 4b495e45dfb3c46fd0c13279a55432c772d67a7eedfd47deafd8187f218b6c29/
总用量 8
-rw-------  1 root root    0 2月  9 21:43 committed
drwxr-xr-x 21 root root 4096 2月  9 21:43 diff
-rw-r--r--  1 root root   26 2月  9 21:43 link
[root@iteblog.com /var/lib/docker/overlay2]$ ll 4c521fd94cf4a09ab871d7c2b48817a8c3e6033f9f80e1ce6dd2d5a16e61021c/
总用量 16
-rw------- 1 root root    0 2月  9 21:43 committed
drwxr-xr-x 6 root root 4096 2月  9 21:43 diff
-rw-r--r-- 1 root root   26 2月  9 21:43 link
-rw-r--r-- 1 root root   57 2月  9 21:43 lower
drwx------ 2 root root 4096 2月  9 21:43 work
[root@iteblog.com /var/lib/docker/overlay2]$ ll 7643e9e3b8404ebef6d0a801d0007f13b5fa57646cb622d60860032e5eb0a382/
总用量 16
drwxr-xr-x 3 root root 4096 2月  9 21:43 diff
-rw-r--r-- 1 root root   26 2月  9 21:43 link
-rw-r--r-- 1 root root   86 2月  9 21:43 lower
drwx------ 2 root root 4096 2月  9 21:43 work
[root@iteblog.com /var/lib/docker/overlay2]$ ll aaad926478d067f54935bd6f84ec8f38dc551c98fb4d1bdae7357d4186d183ac/
总用量 16
-rw------- 1 root root    0 2月  9 21:43 committed
drwxr-xr-x 3 root root 4096 2月  9 21:43 diff
-rw-r--r-- 1 root root   26 2月  9 21:43 link
-rw-r--r-- 1 root root   28 2月  9 21:43 lower
drwx------ 2 root root 4096 2月  9 21:43 work
```

在 Docker 中，最底层的层是不包含 lower 文件的，所以，4b495e45dfb3c46fd0c13279a55432c772d67a7eedfd47deafd8187f218b6c29 文件夹里面存放的是最底层的镜像。从第二层开始，镜像文件里面包含了 link 和 lower 文件以及

work 和 diff 目录。其中：

- link：这个文件里面主要记录了当前镜像在 l 目录里面的链接文件名称；比如 4b495e45dfb3c46fd0c13279a55432c772d67a7eedfd47deafd8187f218b6c29/link 里面存储的就是 3XQB7H6IZZD3I7GYHEZANUBTXO，正好和上面 l 目录下面的 3XQB7H6IZZD3I7GYHEZANUBTXO -> ../4b495e45dfb3c46fd0c13279a55432c772d67a7eedfd47deafd8187f218b6c29/diff 对应。
- lower：这个文件里面存储了当前镜像依赖了底层镜像的 id，存储的也是 l 目录里面的链接文件名称，如果依赖了多个镜像的话使用：符号分割；比如 ubuntu 镜像的第二层依赖为 l/3XQB7H6IZZD3I7GYHEZANUBTXO。
- work：这个目录是 OverlayFS 工作需要用到的文件夹；
- diff：这个目录主要存储的是相对于依赖的镜像差异文件。

根据上面的描述，我们查看下每层镜像的 lower 文件里面的内容：

```
[root@iteblog.com /var/lib/docker/overlay2]$ cat 4c521fd94cf4a09ab871d7c2b48817a8c3e6033f9f80e1ce6dd2d5a16e61021c/lower
l/TFOUROFQV2HLTKCYRKEJ37MXXZ:l/3XQB7H6IZZD3I7GYHEZANUBTXO
[root@iteblog.com /var/lib/docker/overlay2]$
[root@iteblog.com /var/lib/docker/overlay2]$ cat 7643e9e3b8404ebef6d0a801d0007f13b5fa57646cb622d60860032e5eb0a382/lower
l/LGEY4HPN6RJO7T7K7TTJUHRKT3:l/TFOUROFQV2HLTKCYRKEJ37MXXZ:l/3XQB7H6IZZD3I7GYHEZANUBTXO
[root@iteblog.com /var/lib/docker/overlay2]$
[root@iteblog.com /var/lib/docker/overlay2]$ cat aaad926478d067f54935bd6f84ec8f38dc551c98fb4d1bdae7357d4186d183ac/lower
l/3XQB7H6IZZD3I7GYHEZANUBTXO
```

所以 ubuntu 镜像按照以下顺序组成的：

```
7643e9e3b8404ebef6d0a801d0007f13b5fa57646cb622d60860032e5eb0a382
4c521fd94cf4a09ab871d7c2b48817a8c3e6033f9f80e1ce6dd2d5a16e61021c
aaad926478d067f54935bd6f84ec8f38dc551c98fb4d1bdae7357d4186d183ac
4b495e45dfb3c46fd0c13279a55432c772d67a7eedfd47deafd8187f218b6c29
```

也就是说 7643e9e3b8404ebef6d0a801d0007f13b5fa57646cb622d60860032e5eb0a382 是最顶层，4b495e45dfb3c46fd0c13279a55432c772d67a7eedfd47deafd8187f218b6c29 是最底层。

有人可能会问，你不是说 OverlayFS 有 merged 目录吗？我咋没看到？这是因为我们的 ubuntu 镜像并没有运行，我们得运行这个镜像，得到容器这时候才会有 merged 目录：

```
[root@iteblog.com /var/lib/docker/overlay2]$ docker run -it ubuntu

root@adf203eba47d:/# cd home/
root@adf203eba47d:/home# ll
root@adf203eba47d:/home# touch iteblog
root@adf203eba47d:/home#
```

这时候会生成容器层：

```
[root@iteblog.com /var/lib/docker/overlay2]$ ll
总用量 36
drwx------ 3 root root 4096 2月   9 21:43 4b495e45dfb3c46fd0c13279a55432c772d67a7eedfd47deafd8187f218b6c29
drwx------ 4 root root 4096 2月   9 21:43 4c521fd94cf4a09ab871d7c2b48817a8c3e6033f9f80e1ce6dd2d5a16e61021c
drwx------ 4 root root 4096 2月   9 22:20 7643e9e3b8404ebef6d0a801d0007f13b5fa57646cb622d60860032e5eb0a382
drwx------ 5 root root 4096 2月   9 22:22 a72f1c7ff372d657bb084d1a4109cf9f9beaf768825e6644c8a534cc01e574c5
drwx------ 4 root root 4096 2月   9 22:22 a72f1c7ff372d657bb084d1a4109cf9f9beaf768825e6644c8a534cc01e574c5-init
drwx------ 4 root root 4096 2月   9 21:43 aaad926478d067f54935bd6f84ec8f38dc551c98fb4d1bdae7357d4186d183ac
drwx------ 2 root root 4096 2月   9 22:22 l
```

其中 a72f1c7ff372d657bb084d1a4109cf9f9beaf768825e6644c8a534cc01e574c5-init 里面就是容器层未修改的内容，a72f1c7ff372d657bb084d1a4109cf9f9beaf768825e6644c8a534cc01e574c5 里面就是容器的可写层，我们的修改就是写到这里面的。比如我们上面进入 ubuntu 里面的 bash 并在 /home 目录新建了一个名为 iteblog 的文件，我们可以看出，在 a72f1c7ff372d657bb084d1a4109cf9f9beaf768825e6644c8a534cc01e574c5 里面已经有 merge 目录，并且写的文件就存储在这里面：

```
[root@iteblog.com /var/lib/docker/overlay2]$ ll a72f1c7ff372d657bb084d1a4109cf9f9beaf768825e6644c8a534cc01e574c5/merged/home/
总用量 0
-rw-r--r-- 1 root root 0 2月   9 22:33 iteblog
```

我们可以使用 mount | grep overlay 命令查看已经挂载的 OverlayFS：

```
[root@iteblog.com /var/lib/docker/overlay2]$ mount | grep overlay
overlay on /var/lib/docker/overlay2/a72f1c7ff372d657bb084d1a4109cf9f9beaf768825e6644c8a534cc01e574c5/merged type overlay (rw,relatime,lowerdir=/var/lib/docker/overlay2/l/LSY6T4HKDZHDH5H56KIBPY2LS2:/var/lib/docker/overlay2/l/QPASHTGWQBPJ3B7WUCOJLGLNFK:/var/lib/docker/overlay2/l/LGEY4HPN6RJO7T7K7TTJUHRKT3:/var/lib/docker/overlay2/l/TFOUROFQV2HLTKCYRKEJ37MXXZ:/var/lib/docker/overlay2/l/3XQB7H6IZZD3I7GYHEZANUBTXO,upperdir=/var/lib/docker/overlay2/a72f1c7ff372d657bb084d1a4109cf9f9beaf768825e6644c8a534cc01e574c5/diff,workdir=/var/lib/docker/overlay2/a72f1c7ff372d657bb084d1a4109cf9f9beaf768825e6644c8a534cc01e574c5/work)
```

可以看到，这个不就是我们在 Docker 入门教程：Docker 基础技术 Union File System 文章中介绍的 OverlayFS 挂载的使用嘛。