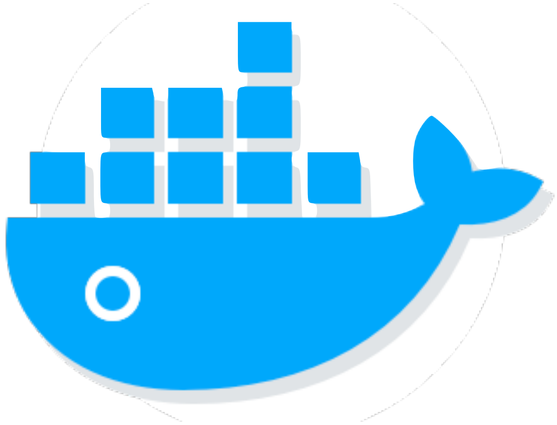


Dockerfile 实战经验分享

周悦秋@好雨科技

ABOUT ME

周悦秋　好雨科技 联合创始人

负责**云帮平台**产品运营工作

10余年工作经验

做过安全、写过C++程序

曾任澳客网 系统架构师、基础技术部总监

参与了好雨 **云帮平台** 产品的设计与研发，目前专注云计算方向的技术研究，对kubernetes、Docker、自动化构建等技术有丰富的实战与应用经验。

课程简介

近几年容器技术逐渐被用户所认可，一些企业已经在开发测试及生产环境进行了部署和应用。但随着容器技术的引进，也随之带来了**新的技术**和**新的工作方式**。

Docker镜像与镜像仓库可以说是容器技术带来的最伟大的发明，它将业务代码与环境打包在一个**分层**的镜像中，通过类似**代码仓库**的方式进行镜像的管理。

Docker强调的 Build、Ship、Run 所指的核心就是**镜像**，因此本次课程正是针对这个核心来讲解。

目标用户	开发者、系统／运维工程师
特色	理论＋经验＋实战
目的	具备独立制作和调试Docker镜像的能力

章节介绍

- 第0章：课程背景知识 3分钟
- 第1章：Docker镜像存储原理与方式 10分钟
- 第2章：Dockerfile基础 15分钟
- 第3章：Dockerfile调试技巧 15分钟
- 第4章：常见问题 3分钟

第0章：课程背景知识

1. Linux 系统基础

掌握Linux系统的基本操作命令，在CentOS、Debian/Ubuntu、Alpine 等系统中安装并配置过应用服务，有基本的Troubleshooting 能力。

2. Linux Shell 编程基础

编写过Shell脚本，了解基本的条件、循环、判断等语句以及变量的高级用法。

3. Docker 安装使用经验

了解Docker，具备基本的Docker安装和使用经验，知道如何运行、停止、维护容器。

章节介绍

- 第0章：课程背景知识
- **第1章：Docker 存储原理与方式**
- 第2章：Dockerfile基础
- 第3章：Dockerfile调试技巧
- 第4章：常见问题

章节介绍

- 第0章：课程背景知识
- **第1章：Docker 存储原理与方式**
  - **Docker 的存储原理**
  - **Docker 常用的存储方式**
- 第2章：Dockerfile基础
- 第3章：Dockerfile调试技巧
- 第4章：常见问题

1.1：Docker 的存储原理

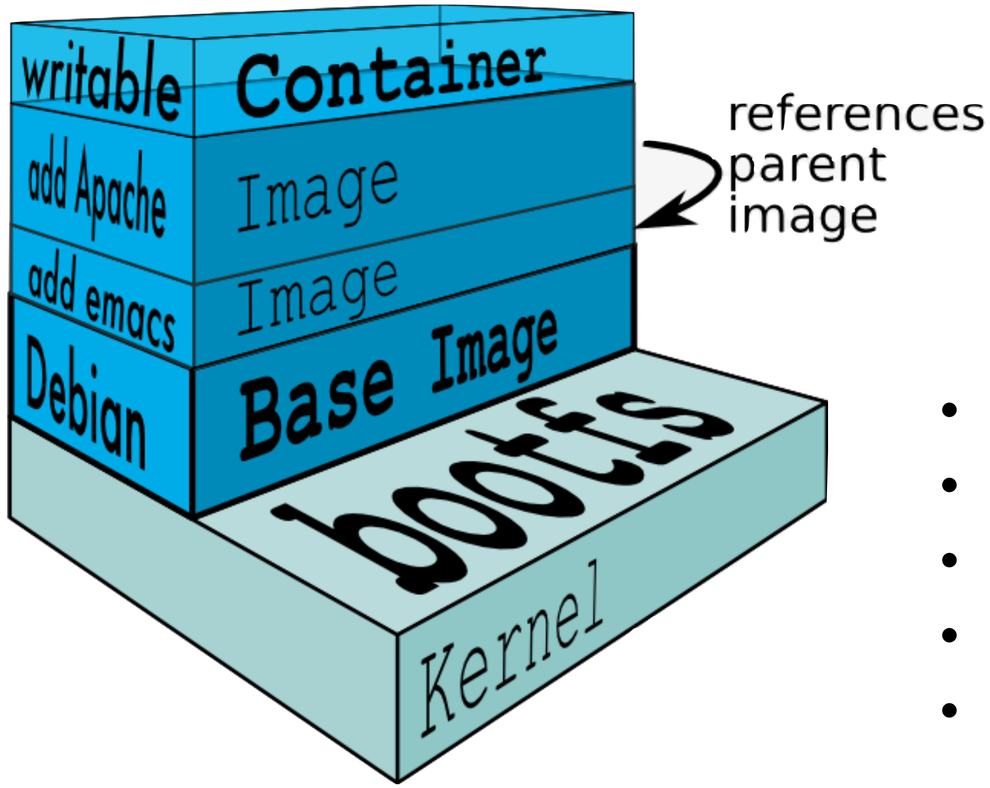
写时复制（CoW: Copy-On-Write）

表示只在需要写时才去复制。比如基于一个image启动多个Container，如果为每个Container都去分配一个image一样的文件系统，将会占用大量的磁盘空间。**CoW技术可以让所有的容器共享image的文件系统，所有数据都从image中读取，只有当要对文件进行写操作时，才从image里把要写的文件复制到自己的文件系统进行修改。**所以无论有多少个容器共享同一个image，所做的写操作都是对从image中复制到自己的文件系统中的复本上进行，并不会修改image的源文件，且多个容器操作同一个文件，会在每个容器的文件系统里生成一个复本，每个容器修改的都是自己的复本，相互隔离，相互不影响。

用时分配（allocate-on-demand）

用在原本没有这个文件的场景，只有在要新写入一个文件时才分配空间，这样可以提高存储资源的利用率。比如启动一个容器，并不会为这个容器预分配一些磁盘空间，而是当有新文件写入时，才按需分配。

1.1：Docker 的存储原理



- 镜像由基础的Linux bootfs 引导
- 一个镜像可以由多层父镜像组成
- 多层镜像文件系统都是只读的
- 实例化后的镜像（容器）的空间是可读写的。
- docker history <image_name> 查看镜像分层信息

章节介绍

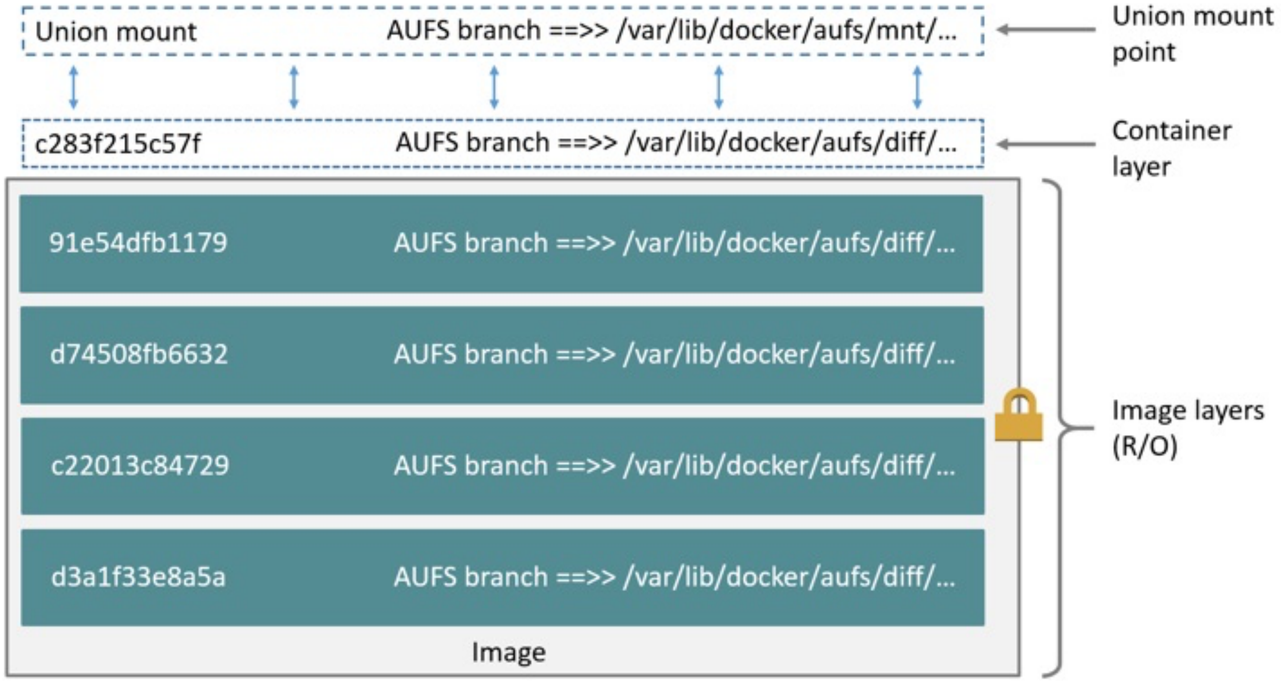
- 第0章：课程背景知识
- **第1章：Docker 存储原理与方式**
  - Docker 的存储原理
  - **Docker 常用的存储方式**
- 第2章：Dockerfile基础
- 第3章：Dockerfile调试技巧
- 第4章：常见问题

1.2：Docker 常用的存储方式

AUFS

AUFS（AnotherUnionFS）是一种Union FS，是**文件级的存储驱动**。AUFS能透明覆盖一或多个现有文件系统的层状文件系统，把多层合并成文件系统的单层表示。这种文件系统可以一层一层地叠加修改文件。无论底下有多少层都是只读的，只有最上层的文件系统是可写的。

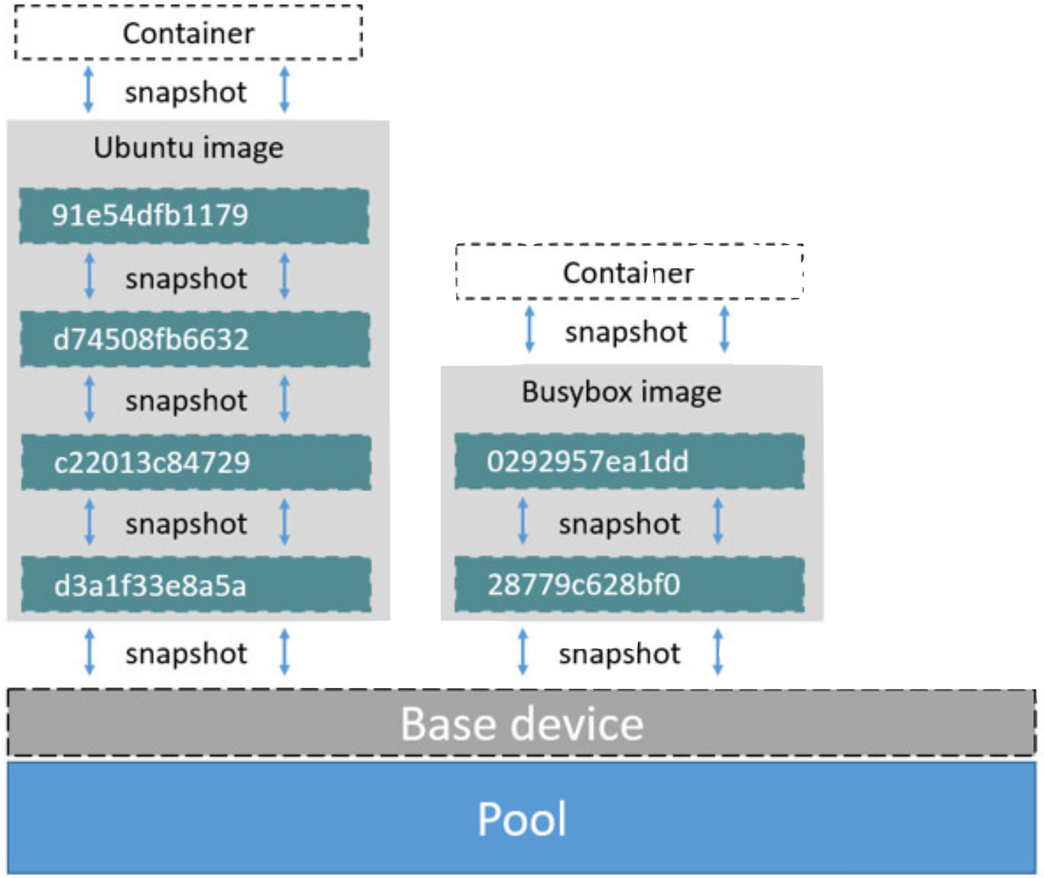
在Docker中，底下的只读层就是image，可写层就是Container。结构如下图所示：



1.2：Docker 常用的存储方式

Device mapper

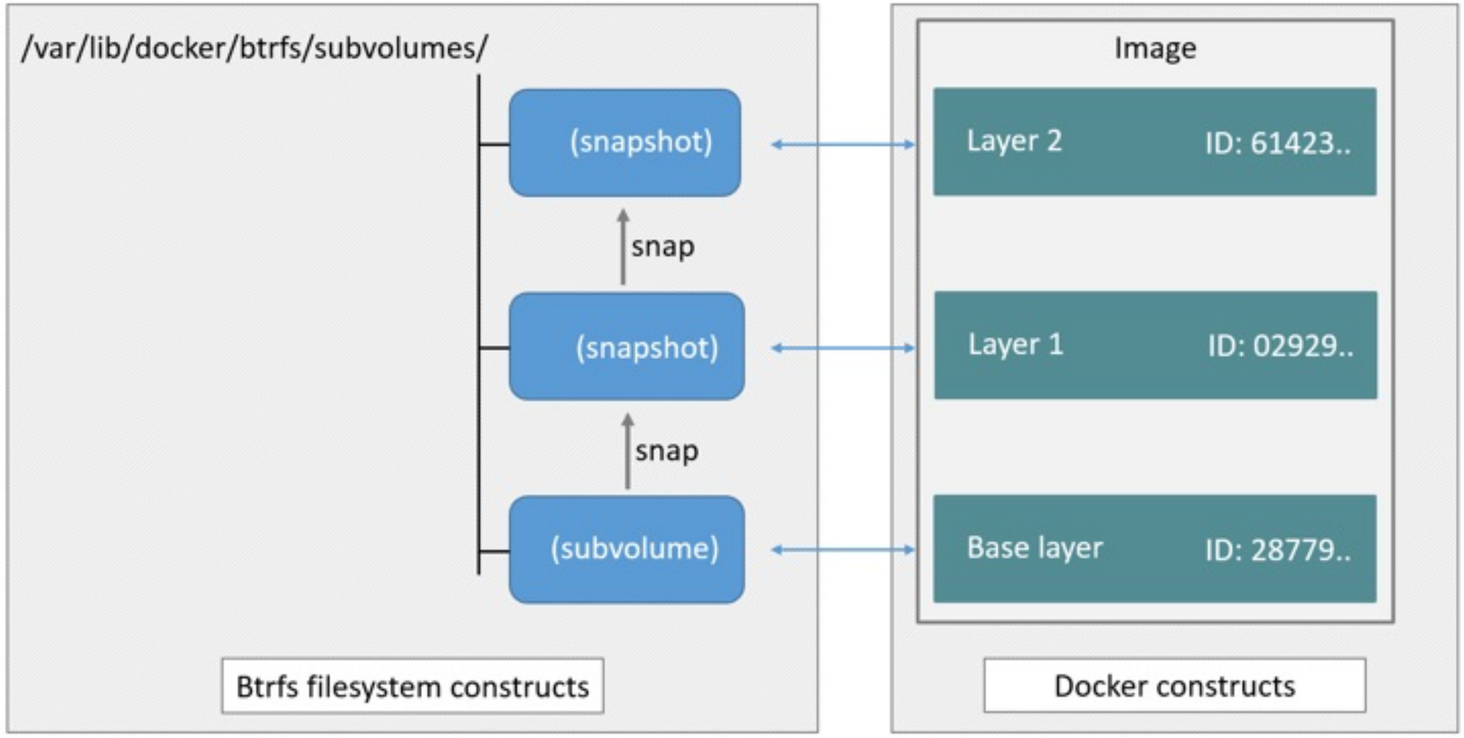
Device mapper是Linux内核2.6.9后支持的，提供的一种从逻辑设备到物理设备的映射框架机制。AUFS文件级存储，而Device mapper是块级存储，所有的操作都是直接对块进行操作，而不是文件。Device mapper驱动会先在块设备上创建一个资源池，然后在资源池上创建一个带有文件系统的基本设备，所有镜像都是这个基本设备的快照，而容器则是镜像的快照。结构如下图所示：



1.2：Docker 常用的存储方式

Btrfs

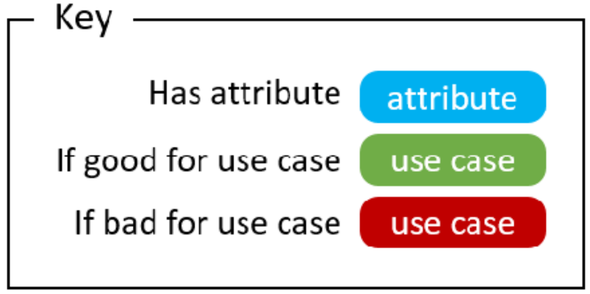
Btrfs被称为下一代写时复制文件系统，并入Linux内核，也是文件级的存储，但可以像Device mapper一直接操作底层设备。**为了灵活利用设备空间，Btrfs 将磁盘空间划分为多个chunk 。每个chunk可以使用不同的磁盘空间分配策略。**



1.2：Docker 常用的存储方式

下图列出Docker各种存储方式的优点缺点：

AUFS	stable	production-ready	good memory use	smooth Docker experience	high write activity	PaaS-type work
Devicemapper (loop)	stable	in mainline kernel	smooth Docker experience	production	performance	lab testing
Devicemapper (direct-lvm)	stable	production-ready	in mainline kernel	smooth Docker experience	PaaS-type work	
Btrfs	in mainline kernel	high write activity	container churn	build pools		
Overlay	stable	good memory use	in mainline kernel	container churn	lab testing	
ZFS native (ZoL)	PaaS-type work					
ZFS FUSE	stable	lab testing	production			



就如Docker官网上说的，没有单一的驱动适合所有的应用场景，要根据不同的场景选择合适的存储驱动，才能有效的提高Docker的性能。

章节介绍

- 第0章：课程背景知识
- 第1章：Docker镜像存储原理与方式
- **第2章：Dockerfile基础**
- 第3章：Dockerfile调试技巧
- 第4章：常见问题

章节介绍

- 第0章：课程背景知识
- 第1章：Docker镜像存储原理与方式
- **第2章：Dockerfile基础**

  build构建

  Dockerfile 主要命令辨析

  Dockerfile书写规范
- 第3章：Dockerfile调试技巧
- 第4章：常见问题

2.1：build构建

理解 “context”

context，上下文（环境），可以理解为build的工作目录，是 docker build 中很重要的一个概念。构建镜像必须指定 context：

```
docker build -t <name:tag> <context路径>
```

不要和构建镜像的Dockerfile 中的 WORKDIR 混淆。

docker build 命令将 context 的目录上传给 docker daemon，由它负责制作镜像。

示例Demo，理解context

2.2：Dockerfile 主要命令解析

- **COPY vs ADD**

他们都是将build目录中的文件拷贝到container来制作image。

ADD 多了2个功能，下载URL和解压。如果不想把压缩文件拷贝到container后会被解压的话，那么使用COPY。如果你的文件通过URL下载，或者是需要将压缩包解压到container的话，请使用ADD。

示例2-2-1，理解COPY与ADD

示例结论：

1、ADD 支持gz，bz2的压缩格式

2、使用ADD，COPY时，如果源是文件夹，目标文件夹如果不存在会自动创建

3、使用ADD，COPY时，如果源是文件，想放到目标不存在的文件夹中，需要再最后加个 /

作业：

1、ADD 除了支持gz，bz2压缩格式之外是否还支持其它压缩格式?

2、ADD 可以从URL下载文件，还支持文件解压，是否支持从URL下载一个压缩包并解压呢?

2.2：Dockerfile 主要命令解析

· **RUN、CMD 与 ENTRYPOINT**

RUN 是在制作镜像时执行的命令，它会影响到镜像的制作。

CMD和 ENTRYPOINT 是用来标记容器运行时的进程及参数是什么，但它们的区别是什么呢?

示例2-2-2，理解CMD与ENTRYPOINT

2.2：Dockerfile 主要命令解析

示例结论

1、CMD 指定的命令和参数都可以在容器运行时改变，但ENTRYPOINT却不能

2、CMD 可以为ENTRYPOINT提供默认参数，但可以通过启动容器时更改

3、请不要使用CMD和ENTRYPOINT指令的 shell形式，它会使用”sh -c” 的形式通过运行一个shell进程来运行命令，并且会忽略CMD和docker run命令行指定的参数。

4、CMD和ENTRYPOINT指令的 shell形式在docker stop的时候不会立即退出，而是等待关闭指令超时后才关闭。

2.3：Dockerfile最佳实践

节省构建时间

- 将系统包软件源地址更改为国内镜像
- 不要安装无用的包（apt-install --no-install-recommends）
- 借助缓存机制

减少镜像大小

- 尽量用较小的基础镜像，父镜像选择优先级 Alpine > Debina > Ubuntu > CentOS
- 能在一条RUN指令中执行的命令就不要执行多次（最小化镜像层数）
- 需要编译的软件在临时系统编译出来，然后将编译好的文件和依赖ADD/COPY到最终镜像中
- 不要升级系统组件（apt-get upgrade）
- 使用. dockerignore过滤无用的文件

其它

- 使用明确的父镜像标签，如 ubuntu:14.04.5
- 每个容器运行一个进程

示例2-3，Dockerfile最佳实践

章节介绍

- 第0章：课程背景知识
- 第1章：Docker镜像存储原理与方式
- 第2章：Dockerfile基础
- **第3章：Dockerfile调试技巧**
- 第4章：常见问题

章节介绍

- 第0章：课程背景知识
- 第1章：Docker镜像存储原理与方式
- 第2章：Dockerfile基础
- **第3章：Dockerfile调试技巧**

  录像机方式

  调试 ENTRYPOINT 脚本的技巧

  实战
- 第4章：常见问题

3.1：录像机方式

跑一个基础镜像，一边操作，一边根据操作的结果写Dockerfile，操作完了之后Dockerfile也就写完了，然后根据这个Dockerfile 在build一个镜像。

以 apline操作系统为基础，创建一个php＋nginx的镜像

示例Demo，录像机方式写dockerfile

3.2：调试 ENTRYPOINT 脚本的技巧

大多数情况下ENTRYPOINT 入口文件都是一个shell脚本，实际上这里所说的调试技巧就是shell脚本的调试技巧，主要时和编写Dockerfile和调试容器结合起来一起用而已。

- **利用变量设置Debug标识符**
- **设置断点**
- **依赖服务的检查**
- **环境变量的使用**
- **设置时区**

示例Demo，演示这些调技巧

3.3：实战——从0开始制作一个镜像

示例：apline系统配置

php+nginx+mysql+禅道pms

章节介绍

- 第0章：课程背景知识
- 第1章：Docker镜像存储原理与方式
- 第2章：Dockerfile基础
- 第3章：Dockerfile调试技巧
- **第4章：常见问题**

4：常见问题

- **系统包，Docker镜像下载慢：** 使用国内镜像加速
- **基础镜像系统选择：** 建议 alpine > debian > ubuntu > centos
- **修改容器后commit镜像大**：避免通过commit的方式制作镜像，这种方式属于“黑箱镜像”，时间一长你自己都不知道做过哪些操作，更别提以后的镜像升级了，而且镜像使用分层存储，commit一次就会产生一层数据，这也是为什么commit出来的镜像很大。建议去写Dockerfile，这才是正道。
- **如何深入学习Dockerfile**：Docker Hub上去看看其他人写的Dockerfile，吸取好的经验，或者参考好雨云市的应用Dockerfile，这些都是生产环境的Dockerfile文件。
- **有镜像，如何获取Dockerfile：**
  - 看镜像作者的项目源码
  - Docker history <镜像id> 看每一层的操作，
  - docker inspect <镜像id> 查看镜像元数据

参考资料

- Understand images, containers, and storage drivers
- Create a base image
- Best practices for writing Dockerfiles
- Understand images, containers, and storage drivers
- Docker and AUFS in practice
- Docker and Btrfs in practice
- Select a storage driver
- Docker and the Device Mapper storage driver
- Docker存储方式选型建议
- Docker五种存储驱动原理及应用场景和性能测试对比
- Docker 问答录
- 好雨云市Dockerfile 示例（https://github.com/goodrain-apps）

好雨 goodrain

好雨招聘

如果你坚信**容器技术**是云计算的**未来**，

如果你愿意**接受挑战**

如果你坚信通过自己的**努力**可以**改变命运**，

加入好雨，我们都是**合伙人**

加入好雨，我们**一起**为云计算**下一场好雨**！

StuQ

谢 谢！

我的微信

Dockerfile公众号

本次课程demo地址：https://github.com/zhouyq/stuq_dockerfile

StuQ

学习自检&课后反馈

请同学们打开问卷

对自己今晚的收获进行自检

对讲师的分享进行反馈，以便讲师下次准备更棒的学习内容