主标题

副标题

作者：BobLi Swigger
课号：0000
日期：2020年10月10日

目录

一级标题1

按照官网给出的教程一步步安装即可，官网给的很详细，不作赘述。
https://docs.docker.com/engine/install/debian/

一级标题2

一些可视化的客户端，类似基于git的github desktop或sourcetree等可视化软件。

- shipyard
- dockerUI
- maDocker

一级标题3

二级标题3-1

docker的镜像由分层的文件系统组成。

- bootfs
- rootfs
- 应用层文件系统

第一层（最低端）的文件系统是bootfs（引导文件系统），我们几乎用不到，仅需了解即可。

第二层文件系统是rootfs，类似于ubuntu等的操作系统。

第三层及往上就是各种应用层的文件系统。

值得注意的是，以上这些文件系统都是只读的，因为这些文件系统属于镜像，在docker里面对于镜像是只允许读的。

像这样多个只读的文件系统层层叠加起来的组合，我们称之为**镜像**

- 读写层

下面我们将介绍docker最为核心的内容之一，write-on-copy。

对docker容器的所有写操作都发生在读写层的文件系统中，而write-on-copy的概念是是这样定义的：

> 当发生写操作是，我们从下面的只读层复制数据到读写层进行写操作，读写层的数据副本会隐藏只读层的数据。

这样，当我们对容器进行更改时，原有的镜像作为容器的骨架并不会被改变，而是把数据额外分离出一个层次进行存储，类似于git的分布式存储，它的好处是什么？在这里留给读者自己去思考。

通过以上的介绍，我们可以很自然的导出容器的概念：

> 一个由多层文件系统支撑的，独立的运行环境

容器的文件系统=镜像的文件系统+读写层 *#个人暂时的理解，还有待商榷*

那这么多的镜像，我们从哪里拉取呢？

类似于github，docker也提供了dockerhub（一个大仓库）提供人们拉取和推送镜像，而dockerhub运行的镜像就是registry，我们也可以在自己的服务器上搭建私有的docker registry（有点类似于git）。

二级标题3-2

类似于github，docker也提供了dockerhub（一个大仓库）提供人们拉取和推送镜像，而dockerhub运行的镜像就是registry，我们也可以在自己的服务器上搭建私有的docker registry（有点类似于git）。

一级标题4

二级标题4-1

三级标题1-1-1

- docker 对docker容器进行的操作;
- **ps** 显示正在运行的容器 -a列出所有容器;
- **start** 启动一个已经结束的容器;
- **stop** 停止一个正在运行的容器;
- rm 删除一个已经停止运行的容器;
- run 从image创建一个容器并运行; -i -t以交互式shell的方式运行容器; -d以守护进程（后台）的方式运行容器; -p主机端口:容器端口映射;
- **commit**提交容器为镜像
- docker image 对docker镜像进行的操作;
- ls 列出本地镜像;
- rm 删除本地镜像（空格后加上容器ID&Name）;

二级标题4-2

```
docker run -i -t --name debian -p 80:80 debian:latest
```