容器系列二：容器的视角-设计交付和架构

作者：代豪

前言：

我们要使用容器，自然就不能回避容器的设计生成过程，而生成之后就涉及到加载运行和管理，那么今天我们就从设计交付和架构这两个角度来说一下容器。

注：本期分享由代豪原创，有容云整理发布。

了解最新云计算资讯，关注有容云官方微信

上篇的《容器起源》说了容器是什么，那么这次我们从两个视角谈谈容器。我们要使用容器，自然就不能回避容器的设计生成过程，而生成之后就涉及到加载运行和管理，那么今天我们就从设计交付和架构这两个角度来说一下容器。（注：由于目前容器中 Docker 应用比较广泛，以下内容中的容器默认为 Docker。）

一. 设计交付运行角度

简述

Docker Image 的设计者按 Dockerfile 的语法规则进行 Docker Image 的设计描述，描述完成后通过 build 命令生成 Docker Image； 交付测试者时，测试者只需基于 Docker Image 进行加载测试即可。

使用过程

我们看容器的好处很多，那么使用容器的过程是怎样的呢?

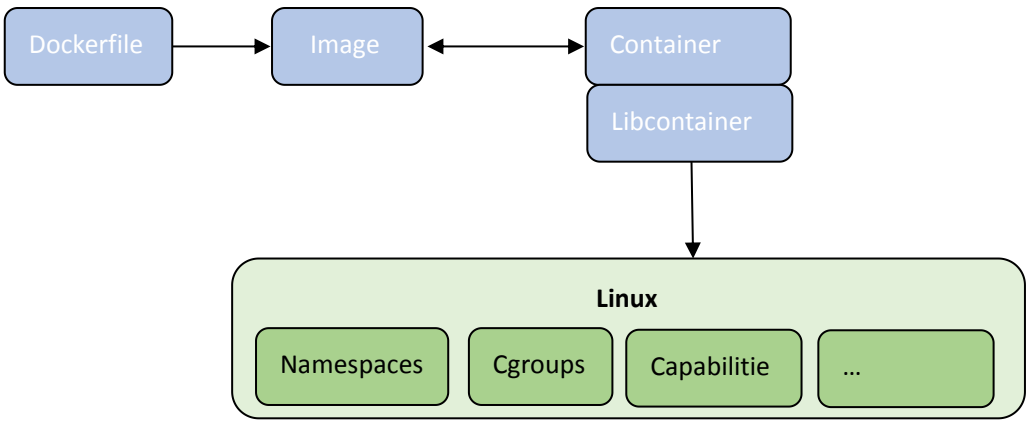
首先，不能回避的是容器的设计，也就是用户用自己的需要设计容器。这里我们先看看传统的编译型语言的使用过程，一个程序员先写好代码，然后通过编译器把代码编译成机器指令，并按一定的系统规则组织起来形成执行文件(如 Windows 里的 exe 文件)，然后交付给测试者，测试者测试完成通过后交付给使用者，使用者直接执行这个文件，而不用关心代码的设计编译和测试过程。

对应容器而言，也有个设计编译和测试使用过程；在容器设计过程有一个叫 Dockerfile

的东西，这个文件就相当于用户 Docker 的设计，在 Dockerfile 完成后 build 命令可以理解为编译过程，实际上就是通过 Dockerfile 的描述生成 Docker 镜像 Image，然后把镜像交付给测试者，测试者测试通过后把镜像提交给使用者。其对应的阶段表格为:

阶段	开发	交付	运行
容器的形式	Dockerfile 描述文件	Docker Image 镜像文件	Container 动态进程
执行的命令	build	run，commit	stop，start，restart

上述表格可以理解为通过描述文件 Dockerfile 生成了镜像文件 Docker Image，加载 Docker Image 运行就形成了动态进程容器 Container。 一旦加载执行起来，容器进程的行为就需要跟 Linux 内核进行交互了。容器进程基于 Linux 内核的支持，那么把这个因素考虑进去后我们可以把图画成这样。



从图上看到 Container 进程通过 Libcontainer 来进行跟 Linux 内核的交互，由于新版

本的 Docker 不支持 LXC 了，因此只列了 Libcontainer。Container 执行需要 Linux 内核的支持，这里为简单起见只说一下两个基础的功能，空间的隔离和资源的限制。

1.）空间隔离

空间的隔离就是用了 Linux 的 Namespaces，也就是名字空间，这个功能的基本作用就是隔离，比如两个用户 u1 和 u2，在不同的 Namespace 下运行是互相不可见的，一个具体的例子是 u1 的进程 PID 和 u2 的进程 PID 可以是相同的，如果是常规进程，那么同一个 OS 下两个进程 PID 是不相同的。

2.）资源限制

资源的限制用了 Linux 内核的 Cgroups，这个功能是用来限制每个容器进程对资源使用量的，比如 CPU，内存，IO 等，这样避免一个容器进程占用资源过大而使其它资源进程受到影响。

有了 Namespaces 和 Cgroups 这两个非常基础的重要功能，空间的隔离使各个容器进程有了接近虚拟机的独立性，资源的限制使各个容器进程有了运行资源的”平等性”，这样基于容器的应用就具备了接近虚拟机的一些节本特性，同时看到容器实际上是强依赖 Linux 内核功能的，这一点上跟虚拟机大家各自的 OS 或内核独立又很大的不同，比如说内核的 Bug 和漏洞都会影响到各个容器进程，这些需要篇幅，这里就先不展开了。

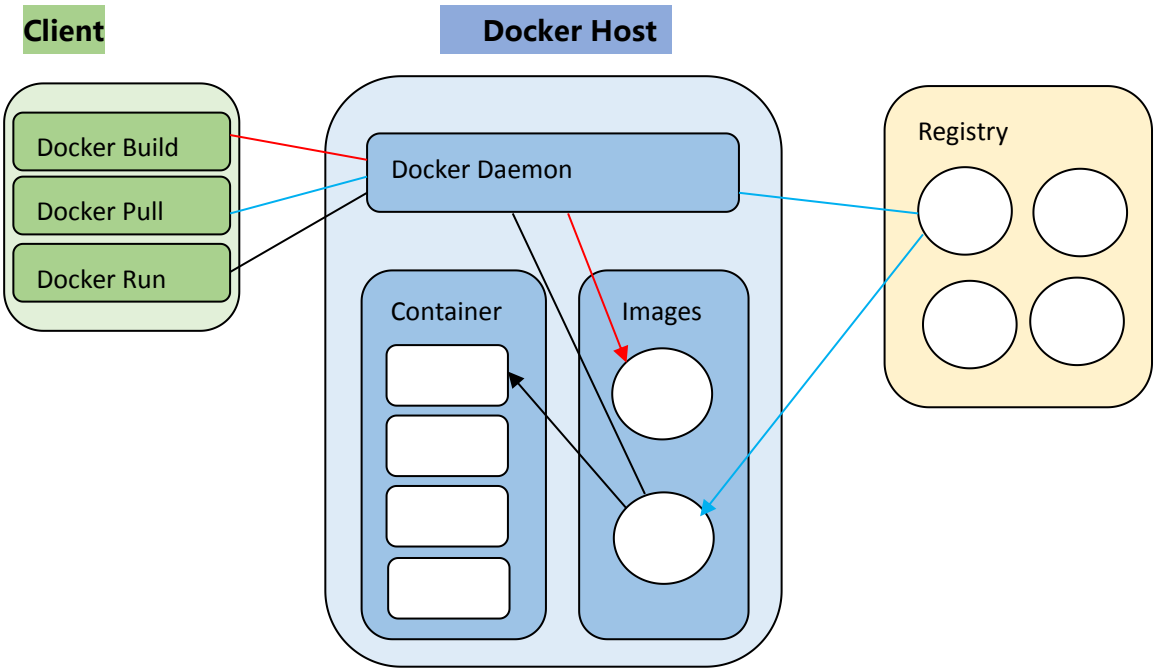
这里看到容器进程的运行是依赖 Linux 内核的，那么这里就不能回避操作系统了，那么问题就来了：用什么样的操作系统来支持容器的运行？CentOS，Ubuntu 是现在大家常用的，这里提一下 CoreOS 和 Rancher OS，这两系统非常合适容器的支撑。为什么这么说呢?

我们已经知道容器共享内核的运行环境，这个操作系统有了基本的内核功能就可以了，

实际上相对操作系统而言可以做得非常小，当然也可以自己拿通用的操作系统进行裁减，但并不是最优的方案，之所以提到 CoreOS 和 Rancher OS，这两个 OS 不但体积小(Rancher OS 29 兆左右，CoreOS 200 多兆)，而且针对容器的支持做了很多调整。比如 Rancher OS，里面进程皆容器，System Docker 起了系统运行的各种必备功能，然后起了 User Docker(Container)。 所以，CoreOS 和 Rancher OS 可以认为是为容器而生的。

二. Docker 架构

现在我们从另一个角度看一下容器：对于 Docker 的运行，是一个典型的客户机服务器 C/S 模型或架构，如下面的图：



图里列了三个常用的基本命令 :build(建镜像)，pull(从仓库拉镜像)，run(运行 Docker)，

不同的颜色代表了不同的命令执行的流程。我们看到所有的 client 命令都提交给了 Docker Daemon， Docker Daemon 负责镜像(Image)， 容器进程(Container)，仓库(Registry)的协调和管理，Docker Daemon 里有 Docker Engine 和 Docker Job 的概念。 对于深入理解 Docker Daemon，建议看一下 Docker 的 Golang 源代码，想省时间可以从 MainDaemon 或 NewDaemon 看起。

上述角度都是基于单个节点上的 Docker(container)而言的，当有多个主机上运行 Docker 时，多节点上的 Docker 管理和协调就需要新的工具了，也可以理解为 Docker 集群的管理，有 Swarm，Rancher，Kunbernets 等，这里先不展开了。好的，今天的分享就暂告一个段落了，后续内容请继续关注!

温馨提示：

有容云携手 Rancher Labs 推出【Rancher | 实战群】，在线为您分享 Docker 技术干货，更有往期回顾精选期刊等你拿！

本群汇集了 Rancher 中国最强技术精英团队及业内技术派高人，宗旨是为了大家拥有更专业的平台交流 Rancher 实战技术，实时与 Rancher 创始团队面对面！同时欢迎各位分享自己的经验、疑难问题，我们将定期邀请分享嘉宾做各类话题分享及回顾，共同实践研究 Docker 容器生态圈。

对 Rancher 和 Docker 技术感兴趣、或对本文中细节需继续探讨的朋友，欢迎加入本群参与讨论！

加群方法：

1.关注【有容云】公众号

2.留言”我要加群”

QQ 群号：216521218