软件复用第三次讨论课

1252899 阮康乐

复用云技术

1. 什么是容器技术

容器技术，本质上是虚拟化技术的一种，属于操作系统级别的虚拟化。容器为应用程序提供了隔离的运行空间：每个容器内都包含一个独享的完整用户环境空间，并且一个容器内的变动不会影响其他容器的运行环境。为了能达 到这种效果，容器技术使用了一系列的系统级别的机制诸如利用 Linux namespaces 来进行空间隔离，通过文件系统的挂载点来决定容器可以访问哪些文件，通过 cgroups 来确定每个容器可以利用多少资源。此外容器之 间共享同一个系统内核，这样当同一个库被多个容器使用时，内存的使用效率会得到提升。

通过过容器技术，应用程序可以和环境打包在一个包中，然后运行在任何支持容器的操作系统中，容器隔离了所有的物理层。这使得人们非常轻量级地、安全可靠地发布软件包（包含了应用程序和运行时环境）。

传统意义上，人们在持续发布流水线上编译，测试，构建出一个软件包，然后再自动化配置一套环境，再将软件部署其上。现在人们可以很容易的为流水线添加一个新的 job,使其构建出一个独立的容器。这个容器可以直接运行在像 AWS 这样的云提供商的机器上。

2. 容器技术 Docker

Docker 是一个开源的应用容器引擎,让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口。

Docker 的思想来自于集装箱，集装箱解决了什么问题？在一艘大船上，可以把货物规整的摆放起来。并且各种各样的货物被集装箱标准化了，集装箱和集装箱之间不会互相影响。那么我就不需要专门运送水果的船和专门运送化学品的船了。只要这些货物在集装箱里封装的好好的，那我就可以用一艘大船把他们都运走。docker 就是类似的理念。现在都流行云计算了，云计算就好比大货轮。docker 就是集装箱。

3. Docker 的优势

Docker 带来的优势有以下几点：

1. 资源独立、隔离

资源隔离是云计算平台的最基本需求。Docker 通过 linux namespace, cgroup 限制了硬件资源与软件运行环境，与宿主机上的其他应用实现了隔离，做到了互不影响。不同应用或服务以“集装箱”（container）为单位装“船”或卸“船”，“集装箱船”（运行 container 的宿主机或集群 ）上，数千数万个“集装箱”排列整齐，不同公司、不同种类的“货物”（运行应用所需的程

序，组件，运行环境，依赖）保持独立。

2. 环境的一致性

开发工程师完成应用开发后 build 一个 docker image，基于这个 image 创建的 container 像是一个集装箱，里面打包了各种“散件货物”（运行应用所需的程序，组件，运行环境，依赖）。无论这个集装箱在哪里：开发环境、测试环境、生产环境，都可以确保集装箱里面的“货物”种类与个数完全相同，软件包不会在测试环境缺失，环境变量不会在生产环境忘记配置，开发环境与生产环境不会因为安装了不同版本的依赖导致应用运行异常。这样的一致性得益于“发货”（build docker image）时已经密封到”集装箱“中，而每一个环节都是在运输这个完整的、不需要拆分合并的”集装箱“。

3. 轻量化

相比传统的虚拟化技术（VM），使用 docker 在 cpu, memory, disk IO, network IO 上的性能损耗都有同样水平甚至更优的表现。Container 的快速创建、启动、销毁受到很多赞誉。

4. Build Once, Run Everywhere

“货物”（应用）在“汽车”，“火车”，“轮船”（私有云、公有云等服务）之间迁移交换时，只需要迁移符合标准规格和装卸方式的“集装箱”（docker container），削减了耗时费力的人工“装卸”（上线、下线应用），带来的是巨大的时间人力成本节约。这使未来仅有少数几个运维人员运维超大规模装载线上应用的容器集群成本可能，如同 60 年代后少数几个机器操作员即可在几小时内连装带卸完一艘万级集装箱船。

4. Docker 的安全性

Docker 容器的安全性，很大程度上依赖 Linux 系统自身。Docker 本身也做了很多防范。

首先 Docker 提供命名空间隔离。命名空间提供了最基础也是最直接的隔离，在容器中运行的进程不会被运行在本地主机上的进程和其他容器通过正常渠道发现和影响。

控制组是 Linux 容器机制的另一个关键组件，它可以确保各个容器可以公平地分享主机的内存、CPU、磁盘 IO 等资源。更重要的是，控制组确保了当发生在容器内的资源压力不会影响到本地主机系统和其他容器。

能力机制（Capability）是 Linux 内核一个强大的特征，可以提供细粒度的权限访问控制。通常在服务器上会运行一堆需要特权限制的进程，包括 ssh、cron、syslogd、硬件管理工具模块、网络配置工具等。几乎所有的特权进程都有容器以外的支持系统来进行管理。大部分情况下，容器并不需要具备所有的能力，只需要少数能力即可。这样就算攻击者在容器中取得了 root 权限，也不能获得本地主机较高权限，能进行的破坏也有限。

Docker 服务端的运行目前还需要 root 权限的支持，Docker 服务端安全性也是十分重要的。Docker 服务端的安全更多需要使用者注意，必须确保可信的用户才能访问到 Docker 服务。

由上述所述，Docker 的安全问题本质上就是容器技术的安全性问题，这包括共用内核问题，Namespace 还不够完善的限制以及 Docker 运行在 root

权限所致。Docker 也在致力于安全方面的改进，用户在使用 Docker 也需要防范这些安全性问题。

5. 容器技术的挑战

容器技术的生态环境发展是一个挑战。

Docker 至今仍然视“Build，Ship，Run”为宗旨，编排（Orchestration）也许正在酝酿，至少布局之中，并未有太多的显露，当然本月初对于 Conductant 的收购，可能是较为明显的一步棋。如果说完备的工具链是进入企业市场的第一步，那么强大的容器集群编排将会满足企业内部进一步的需求。三年的时间，Docker 把绝大部分的精力放在完善容器上，然而容器（Container）和应用（Application）之间总是隔着不可逾越的鸿沟。反观 Docker，细细玩味，Docker 定义的内容全部是容器的原语，只有在 Docker Compose 一层存在薄薄的应用原语。坐拥用户群与标准，高举“Docker Native”的大旗，在帮助用户转变应用模式方面，Docker 责无旁贷。应用编排领域，Docker 给外界的感受一直是不紧不慢，大有成竹在胸之态。生态之中，其他玩家最担忧的恐怕就是“Docker 的釜底抽薪”了，也就是在开源的 Docker Engine 中添加更多的商业化考量，从而使得自身的竞争处于绝对的优势。

6. 参考文献

[1]虚拟化 VS 容器化
http://www.oschina.net/news/61820/virtualization-vs-containerization
[2]Docker 全面介绍
http://weizijun.cn/2015/12/10/Docker%E5%85%A8%E9%9D%A2%E4%BB%8B%E7%BB%8D/
[3]容器云技术
http://huangtxin.github.io/2015/11/20/%E5%AE%B9%E5%99%A8%E4%BA%91%E6%8A%80%E6%9C%AF/
[4] Docker 三年回顾 http://www.infoq.com/cn/articles/docker-turns-3