Monday, June 6, 2016

讨论课03

谢志杰 1352975

-

一、容器技术

LXC：

LXC，其名称来自Linux软件容器（Linux Containers）的缩写，一种操作系统层虚拟化（Operating system–level virtualization）技术，为Linux内核容器功能的一个用户空间接口。它将应用软件系统打包成一个软件容器（Container），内含应用软件本身的代码，以及所需要的操作系统核心和库。通过统一的名字空间和共用API来分配不同软件容器的可用硬件资源，创造出应用程序的独立沙箱运行环境，使得Linux用户可以容易的创建和管理系统或应用容器。[1]

在Linux内核中，提供了cgroups功能，来达成资源的区隔化。它同时也提供了名称空间区隔化的功能，使应用程序看到的操作系统环境被区隔成独立区间，包括进程树，网络，用户id，以及挂载的文件系统。但是cgroups并不一定需要引导任何虚拟机。

LXC利用cgroups与名称空间的功能，提供应用软件一个独立的操作系统环境。LXC不需要Hypervisor这个软件层，软件容器（Container）本身极为轻量化，提升了创建虚拟机的速度。软件Docker被用来管理LXC的环境。

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。

- 环境管理复杂 - 从各种OS到各种中间件到各种app, 一款产品能够成功作为开发者需要关心的东西太多，且难于管理，这个问题几乎在所有现代IT相关行业都需要面对。
- 云计算时代的到来 - AWS的成功, 引导开发者将应用转移到 cloud 上, 解决了硬件管理的问题，然而中间件相关的问题依然存在 (所以openstack HEAT和 AWS cloudformation 都着力解决这个问题)。开发者思路变化提供了可能性。
- 虚拟化手段的变化 - cloud 时代采用标配硬件来降低成本，采用虚拟化手段来满足用户按需使用的需求以及保证可用性和隔离性。然而无论是KVM还是Xen在 docker 看来,都在浪费资源，因为用户需要的是高效运行环境而非OS, GuestOS既浪费资源又难于管理, 更加轻量级的LXC更加灵活和快速

- LXC的移动性 - LXC在 linux 2.6 的 kernel 里就已经存在了，但是其设计之初并非为云计算考虑的，缺少标准化的描述手段和容器的可迁移性，决定其构建出的环境难于迁移和标准化管理(相对于KVM之类image和snapshot的概念)。docker 就在这个问题上做出实质性的革新。这是docker最独特的地方。

接下来以Docker为主，讨论容器技术:

局限:

Docker并不是全能的，设计之初也不是KVM之类虚拟化手段的替代品，简单总结几点:

- Docker是基于Linux 64bit的，无法在32bit的linux/Windows/unix环境下使用
- LXC是基于cgroup等linux kernel功能的，因此container的guest系统只能是linux base的
- 隔离性相比KVM之类的虚拟化方案还是有些欠缺，所有container公用一部分的运行库
- 网络管理相对简单，主要是基于namespace隔离
- cgroup的cpu和cpuset提供的cpu功能相比KVM的等虚拟化方案相比难以度量(所以dotcloud主要是按内存收费)
- docker对disk的管理比较有限
- container随着用户进程的停止而销毁，container中的log等用户数据不便收集

优势:

1、 比虚拟机高效:

因容器复用了本地主机操作系统，仅仅是封装了容器运行所需的软件环境（从这个角度看可以参考RPM安装包），因此与主机上直接运行软件所需的资源几乎是一样的。不像虚拟机那样需要额外的内存、CPU等来支持虚拟机操作系统的运行。

2、快速交付和部署:

对开发和运维（devop）人员来说，最希望的就是一次创建或配置，可以在任意地方正常运行。而且可以保证每一个地方运行的环境都是一模一样的，不会因为开发环境与生产环境不同而导致某些问题。

docker容器的启动更是秒级的，因此可以随时快速生产、关闭。

3、轻松迁移和扩展:

docker镜像可以在任意环境中迁移，而不会出现兼容性问题，迁移过程轻松方便。

4、管理简单:

使用 Docker，只需要小小的修改，就可以替代以往大量的更新工作。所有的修改都以增量的方式被分发和更新，从而实现自动化并且高效的管理。

安全:

1、什么是云计算容器?

容器技术在很多方面都不同于诸如VMware和VirtualBox这样的技术。

首先，容器通常有一个目的，那就是要托管一个网络服务器或一个数据库。从技术上讲，容器与虚拟机之间的根本区别在于虚拟机模拟虚拟硬件，它需要一个系统管理程序，因此它需要比云计算容器技术更多的磁盘空间和更强大的处理能力。这样一来，云计算容器就成为了受云计算供应商安全性程序垂青的组件。例如，Docker就与所有的主要云计算服务供应商有着一个合作伙伴关系，如亚马逊、微软以及谷歌等。

毫无疑问，容器技术是非常有用的，它能够提供更好的便携性和定制化，同时减少资源消耗和成本支出，但是与其他众多"流行的"技术类似，安全性是阻碍其进一步发展的因素。

2、云计算容器技术的安全性问题

容器技术的最大问题在于，它们缺少一个像虚拟机所拥有的安全边界。从理论上来说，如果一名黑客能够在底层操作系统中找到一个漏洞，那么他就同样可以利用这个漏洞来获得访问容器的权限。反之的可能性也是理论上存在的，黑客可以找到容器的漏洞，进而利用它来获得访问底层服务器的权限。

更糟糕的是，Docker和其他的容器技术采用了一些可作为“root”超级用户运行的功能(Docker表示，在上个月发布的1.8版本中已经解决了root权限问题)。这个问题可能会对云计算供应商环境带来更大的影响，我们可以想象：所有的云计算服务都通过容器进行部署，而一名黑客能够突破一个容器，并访问相同硬件上的其他容器。对于云计算供应商和云计算用户来说，这个问题都有可能是灾难性的。所以，容器技术的部署需要深思熟虑。

容器的另一个问题是实际的创建过程。例如，如果某一家企业创建了它自己的容器，那么其安全性水平将起决于企业本身的能力;如果工作人员没有很好地开发、保护和管理它，那么容器可能就无法实现其预期效益——也许使用预制的容器可能会更好。但是，需要引起注意的是，如果企业需要从一个存储库中获得一个容器，它可能并不能确切地知道正在下载什么内容;例如，如果容器有一个记录按键操作的技术可将用户名和密码上传至远程服务器，那么会怎么样?

这些安全问题都是较为普遍存在的，因为业界对于容器安全方面的研究还投入不多。此外，对于如何确保其安全性也没有一个明确的指导意见。

简单而言，在业内把容器技术和虚拟机的安全性划上等号之前，还是有很多工作要做的。但是，这项工作已经开始。Docker在2015年八月发布了一个重大的安全更新，其中就包括了名为Docker 内容信任的新功能，这个新功能主要是通过为容器库提供一个基于公共密钥的签名机制来实现容器部署的安全性，从而在一定程度上缓解这一问题。

3、确保云计算容器安全性的最佳实践

如果某一家企业是从公共库中获取Docker容器的，那么它应当寻找那些由新的Docker内容信任系统签名的Docker容器，以便于确保它下载的是一个合法的容器。其他需要注意的关键点包括：确保禁用不需要的功能、确保只有受信任的用户能够操作控制容器的守护进程。此外，还应启用容器间的防火墙以限制不同容器之间的交互。

当容器技术变得越来越安全时，它们将在大多数企业中占据一席之地。标准部署匹配容器化优势将为业内用户带来便于部署、较低的资源要求以及成本降低等诸多好处。例如，想要部署数据库系统的IT团队能够很容易地获得一个MySQL容器，而这个容器中已经准备好了所有的必备组件，这样就大大缩短了系统部署所需的时间。对于容器技术来说， 最可能的应用场景将是成为虚拟机的配合角色而不是取代它们。在企业部署应用中，这两种技术都有其立足之所。

总之，容器是一个很好的技术，应考虑将其纳入商业应用。配合虚拟机技术，它们能够节省时间和金钱，但是它们的部署却是需要费一番思量的，当然其来源也应是可信的。

复用:

Volumes

卷是容器的数据部分。当容器被创建时候被初始化。卷允许你持久化和分享容器的数据。数据卷和默认的联合文件系统是独立的，作为宿主系统中的正常文件和目录的形式存在。所以，如果你删除，更新或重新构建你的容器，你的数据卷不会被影响。当你想要更新容器时，你可以直接去更改他（作为奖励，数据卷可以在多个容器见分享和复用，这一点很赞~）

组件复用

Docker容器可以基于“基础镜像”创建许多个独立的、不同功能的组件，这意味着一次制作的镜像可以再将来的许多项目中不断复用。

Reference:

[1] Docker http://baike.baidu.com/item/Docker

[2] Linux容器虚拟技术及GIS应用前景 http://www.oschina.net/question/2535385_2143148

[3] docker容器技术系列一：基本概念 http://www.toxingwang.com/cloud/docker/2994.html