容器技术与Docker

1352961 秦博

容器技术虚拟化技术已经成为一种被大家广泛认可的容器技术服务器资源共享方式，容器技术可以在按需构建容器技术操作系统实例的过程当中为系统管理员提供极大的灵活性。

Docker 是 Docker.Inc 公司开源的一个基于LXC技术之上构建的Container容器引擎， 源代码托管在 GitHub 上, 基于Go语言并遵从Apache2.0协议开源。 Docker在2014年6月召开DockerConf 2014技术大会吸引了IBM、Google、RedHat等业界知名公司的关注和技术支持，无论是从 GitHub 上的代码活跃度，还是Redhat宣布在RHEL7中正式支持Docker, 都给业界一个信号，这是一项创新型的技术解决方案。 就连 Google 公司的 Compute Engine 也支持 docker 在其之上运行, 国内“BAT”先锋企业百度Baidu App Engine(BAE)平台也是以Docker作为其PaaS云基础。Docker产生的目的就是为了解决以下问题:

1) 环境管理复杂: 从各种OS到各种中间件再到各种App，一款产品能够成功发布，作为开发者需要关心的东西太多，且难于管理，这个问题在软件行业中普遍存在并需要直接面对。Docker可以简化部署多种应用实例工作，比如Web应用、后台应用、数据库应用、大数据应用比如Hadoop集群、消息队列等等都可以打包成一个Image部署。

2) 云计算时代的到来: AWS的成功, 引导开发者将应用转移到云上, 解决了硬件管理的问题，然而软件配置和管理相关的问题依然存在。Docker的出现正好能帮助软件开发者开阔思路，尝试新的软件管理方法来解决这个问题。

3) 虚拟化手段的变化: 云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分配的资源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。

4) LXC的便携性: 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便携性，决定其构建出的环境难于分发和标准化管理(相对于KVM之类image和snapshot的概念)。Docker就在这个问题上做出了实质性的创新方法。

Docker核心是一个操作系统级虚拟化方法, 理解起来可能并不像VM那样直观。我们从虚拟化方法的四个方面：隔离性、可配额/可度量、便携性、安全性来详细介绍Docker的技术细节。

2.1. 隔离性: Linux Namespace

每个用户实例之间相互隔离, 互不影响。 一般的硬件虚拟化方法给出的方法是VM，而LXC给出的方法是container，更细一点讲就是kernel namespace。其中

pid、net、ipc、mnt、uts、user等namespace将container的进程、网络、消息、文件系统、UTS("UNIX Time-sharing System”)和用户空间隔离开。

2.2 可配额/可度量 - Control Groups (cgroups)

cgroups 实现了对资源的配额和度量。 cgroups 的使用非常简单，提供类似文件的接口，在 /cgroup目录下新建一个文件夹即可新建一个group，在此文件夹中新建task文件，并将pid写入该文件，即可实现对该进程的资源控制。groups可以限制blkio、cpu、cpuacct、cpuset、devices、freezer、memory、net_cls、ns九大子系统的资源，

2.3 便携性: AUFS

AUFS (AnotherUnionFS) 是一种 Union FS, 简单来说就是支持将不同目录挂载到同一个虚拟文件系统下(unite several directories into a single virtual filesystem)的文件系统, 更进一步的理解, AUFS支持为每一个成员目录(类似Git Branch)设定 readonly、readwrite 和 whiteout-able 权限, 同时 AUFS 里有一个类似分层的概念, 对 readonly 权限的 branch 可以逻辑上进行修改(增量地, 不影响 readonly 部分的)。通常 Union FS 有两个用途, 一方面可以实现不借助 LVM、RAID 将多个disk 挂到同一个目录下, 另一个更常用的就是将一个 readonly 的 branch 和一个 writeable 的 branch 联合在一起，Live CD正是基于此方法可以允许在 OS image 不变的基础上允许用户在其上进行一些写操作。

2.4 安全性: AppArmor, SELinux, GRSEC

安全永远是相对的，这里有三个方面可以考虑Docker的安全特性:

1. 由kernel namespaces和cgroups实现的Linux系统固有的安全标准;
2. Docker Deamon的安全接口;
3. Linux本身的安全加固解决方案,类如AppArmor, SELinux;