# 初始docker技术

原创 Mr大表哥 2017-02-27 14:33:05 评论(0) 231人阅读

## 一、Docker简介

Docker是什么?
Docker的英文本意是“搬运工”，在程序员的世界里，Docker搬运的是集装箱（Container），集装箱里装的是任意类型的App，开发者通过Docker可以将App变成一种标准化的、可移植的、自管理的组件，可以在任何主流系统中开发、调试和运行。

说白了,docker是一种用了新颖方式实现的轻量级虚拟机,类似于VM,但是在原理和应用上和VM的差别还是很大的.并且docker的专业叫法是应用容器(ApplicationContainer)。

为啥要用容器?
应用容器是个啥样子呢,一个做好的应用容器长得就像一个装好了一组特定应用的虚拟机一样,比如我现在想用mysql,那我就找个装好了mysql的容器就可以了,然后运行起来,我就能使用mysql了。

为啥不能直接安装一个mysql?安装一个SQLServer也可以啊,可是有的时候根据每个人电脑的不同,在安装的时候可能会报出各种各样的错误,万一你的机器中毒了,你的电脑挂了,你所有的服务都需要重新安装.但是有了docker,或者说有了容器就不同了,你就相当于有了一个可以运行起来的虚拟机,只要你能运行容器,mysql的配置就省了.而且如果你想换个电脑,直接把容器”端过来”就可以使用容器里面的服务.

Docker 基于 Go 语言开发，代码托管在Github上，并遵循Apache 2.0 开源协议。Docker 容器可以封装任何有效负载，几乎可以在任何服务器之间进行一致性运行。换句话说，开发者构建的应用只需一次构建即可多平台运行。运营人员只需配置他们的服务，即可运行所有的应用。

若是利用容器的话,那么开发直接在容器里开发,测试的时候把整个容器给测试,测好了把测试后容器再上线就好了.通过容器,整个开发,测试和生产环境可以保持高度一致。

此外容器也VM一样具有一定得隔离性,各个容器之间的数据和内存空间相互隔离,可以保证一定的安全性。

Hyper-V、KVM和Xen等虚拟机管理程序都“基于虚拟化硬件仿真机制。这意味着，它们对系统要求很高.然而，容器却使用共享的操作系统。这意味着它们在使用系统资源方面比虚拟机管理程序要高效得多。容器不是对硬件进行虚拟化处理，而是驻留在一个Linux实例上。
Docker可以解决虚拟机能够解决的问题，同时也能够解决虚拟机由于资源要求过高而无法解决的问题。

为什么要使用docker?
1 、快速交付应用程序
开发者使用一个标准的 image 来构建开发容器，开发完成之后，系统管理员就可以使用这个容器来部署代码
docker可以快速创建容器，快速迭代应用程序，并让整个过程可见，使团队中的其他成员更容易理解应用程序是如何创建和工作的。
docker容器很轻！很快！容器的启动时间是次秒级的，节约开发、测试、部署的时间
2 、更容易部署和扩展
docker容器可以在几乎所有的环境中运行，物理机、虚拟机、公有云、私有云、个人电脑、服务器等等。
docker容器兼容很多平台，这样就可以把一个应用程序从一个平台迁移到另外一个。
3 、效率更高
docker容器不需要 hypervisor ，他是内核级的虚拟化。
4 、快速部署也意味着更简单的管理
通常只需要小小的改变就可以替代以往巨型和大量的更新工作。

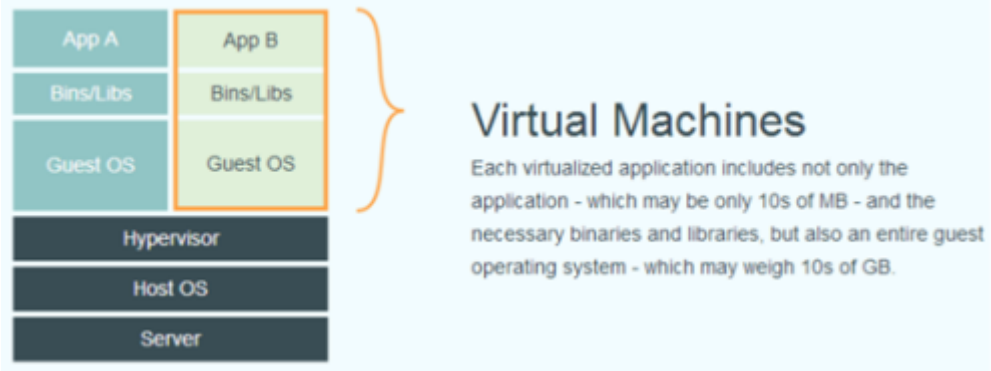
Docker 的常用案例包括：
自动打包和部署应用
创建轻量、私有的 PaaS 环境
自动化测试和持续集成/部署
部署并扩展 Web 应用、数据库和后端服务器

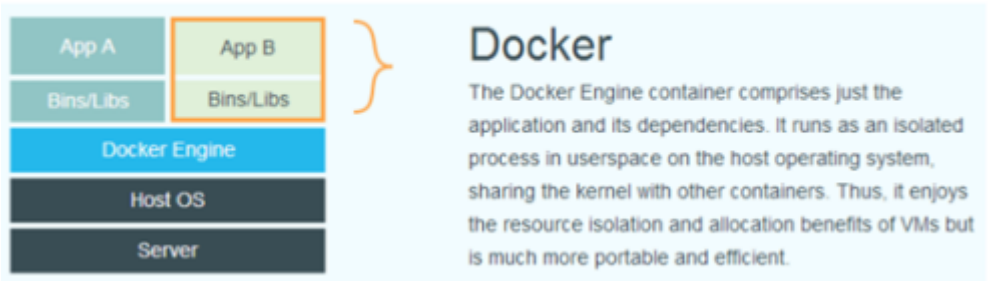
那么为啥不用VM?
那么既然容器和VM这么类似为啥不用VM?docker容器相对于VM还是有很多优点的:
1.启动速度快,容器通常在一秒内可以启动.而VM要很久.
2.资源利用率高,一台普通服务器可以跑上千个容器，而跑VM就。。。。。。

3. 性能开销小，VM需要额外的CPU和内存来完成OS的功能，这一部分占据了额外的资源.

为啥相似的功能在性能上会有如此巨大的差距呢?看一下他们的设计图,先看VM的:

下面的图片比较了 Docker 和传统虚拟化方式的不同之处，




可见容器是在操作系统层面上实现虚拟化，直接复用本地主机的操作系统，而传统方式则是在硬件层面实现。

Docker优势和劣势

作为一种新兴的虚拟化方式，Docker 跟传统的虚拟化方式相比具有众多的优势。

首先，Docker 容器的启动可以在秒级实现，这相比传统的虚拟机方式要快得多。

其次，Docker 对系统资源的利用率很高，一台主机上可以同时运行数千个 Docker 容器。

容器除了运行其中应用外，基本不消耗额外的系统资源，使得应用的性能很高，同时系统的开销尽量小。传统虚拟机方式运行 10 个不同的应用就要起 10 个虚拟机，而 Docker 只需要启动 10 个隔离的应用即可。

具体说来，Docker 在如下几个方面具有较大的优势。

更快速的交付和部署

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。开发者可以使用一个标准的镜像来构建一套开发容器，开发完成

之后，运维人员可以直接使用这个容器来部署代码。 Docker 可以快速创建容器，快速迭代应用程序，并让整个过程全程可见，使团队中的其他成员更容易理解应用程序是如何创建和工作的。 Docker 容器很轻很快！容器的启动时间是秒级的，大量地节约开发、测试、部署的时间。

更高效的虚拟化

Docker 容器的运行不需要额外的 hypervisor 支持，它是内核级的虚拟化，因此可以实现更高的性能和效率。

更轻松的迁移和扩展

Docker 容器几乎可以在任意的平台上运行，包括物理机、虚拟机、公有云、私有云、个人电脑、服务器等。这种兼容性可以让用户把一个应用程序从一个平台直接迁移到另外一个。

更简单的管理

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。

对比传统虚拟机总结

特性容器虚拟机

启动秒级分钟级

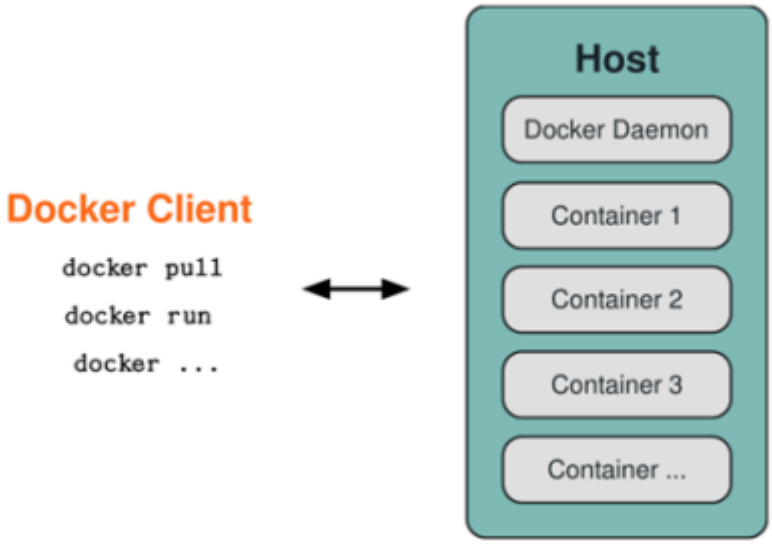
硬盘使用一般为MB 一般为GB

性能接近原生弱于

系统支持量 单机支持上千个容器

## 二、Docker 的体系结构

docker使用C/S 架构，dockerdaemon 作为 server 端接受 client 的请求，并处理（创建、运行、分发容器），他们可以运行在一个机器上，也通过socket或者 RESTful API 通信




Docker daemon 一般在宿主主机后台运行。

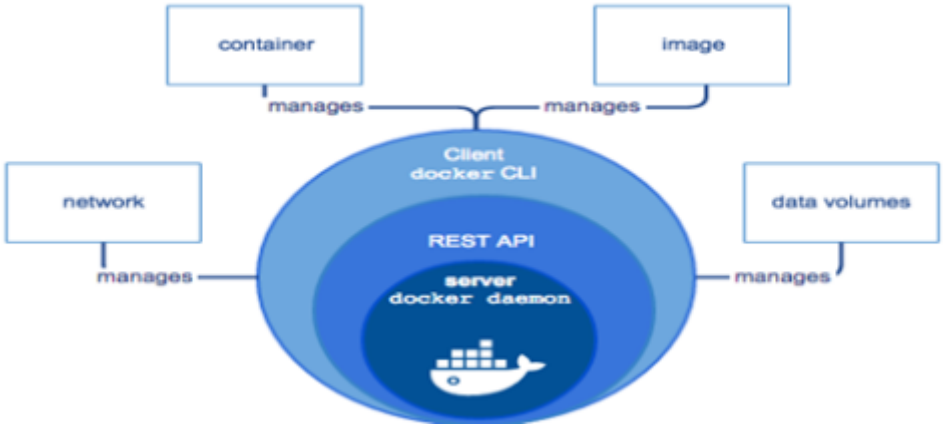
Docker client以系统命令的形式存在，用户用docker命令来跟docker daemon交互。

Docker 守护进程（Docker daemon）

如上图所示，Docker 守护进程运行在一台主机上。用户并不直接和守护进程进行交互，而是通过 Docker 客户端间接和其通信。

Docker 客户端（Docker client）

Docker 客户端，实际上是docker的二进制程序，是用户与 Docker 交互方式。它接收用户指令并且与背后的 Docker 守护进程通信。




Docker 内部：

要理解 Docker 内部构建，需要理解以下三种部件：

Docker 镜像 - Docker images

Docker 仓库 - Docker registeries

Docker 容器 - Docker containers

Docker 镜像

Docker 镜像是 Docker 容器运行时的只读模板，镜像可以用来创建 Docker 容器。每一个镜像由一系列的层（layers）组成。Docker 使用UnionFS（联合文件系统）来将这些层联合到单独的镜像中。UnionFS允许独立文件系统中的文件和文件夹(称之为分支)被透明覆盖，形成一个单独连贯的文件系统。正因为有了这些层的存在，Docker 是如此的轻量。当你改变了一个 Docker 镜像，比如升级到某个程序到新的版本，一个新的层会被创建。因此，不用替换整个原先的镜像或者重新建立(在使用虚拟机的时候你可能会这么做)，只是一个新的层被添加或升级了。现在你不用重新发布整个镜像，只需要升级，层使得分发 Docker 镜像变得简单和快速。

每个docker都有很多层次构成，docker使用 union file systems 将这些不同的层结合到一个image 中去。

例如：centos镜像中安装MySQL5.6，就成了nginx镜像”，其实在此时Docker镜像的层级概念就体现出来了。底层一个centos操作系统镜像，上面叠加一个ngnx层，就完成了一个nginx镜像的构建。层级概念就不难理解，此时我们一般centos操作系统镜像称为nginx镜像层的父镜像。


Docker 仓库

Docker 仓库用来保存镜像，可以理解为代码控制中的代码仓库。同样的，Docker 仓库也有公有和私有的概念。公有的 Docker 仓库名字是 Docker Hub。Docker Hub 提供了庞大的镜像集合供使用。这些镜像可以是自己创建，或者在别人的镜像基础上创建。

仓库是集中存放镜像文件的场所。有时候会把仓库和仓库注册服务器（Registry）混为一谈，并不严格区分。实际上，仓库注册服务器上往往存放着多个仓库，每个仓库中又包含了多个镜像，每个镜像有不同的标签（tag）。

仓库分为公开仓库（Public）和私有仓库（Private）两种形式。

最大的公开仓库是 Docker Hub，存放了数量庞大的镜像供用户下载。国内的公开仓库包括 Docker Pool等，可以提供大陆用户更稳定快速的访问。

当然，用户也可以在本地网络内创建一个私有仓库。

当用户创建了自己的镜像之后就可以使用push命令将它上传到公有或者私有仓库，这样下次在另外一台机器上使用这个镜像时候，只需要从仓库上pull下来就可以了。

*注：Docker 仓库的概念跟Git类似，注册服务器可以理解为 GitHub 这样的托管服务。

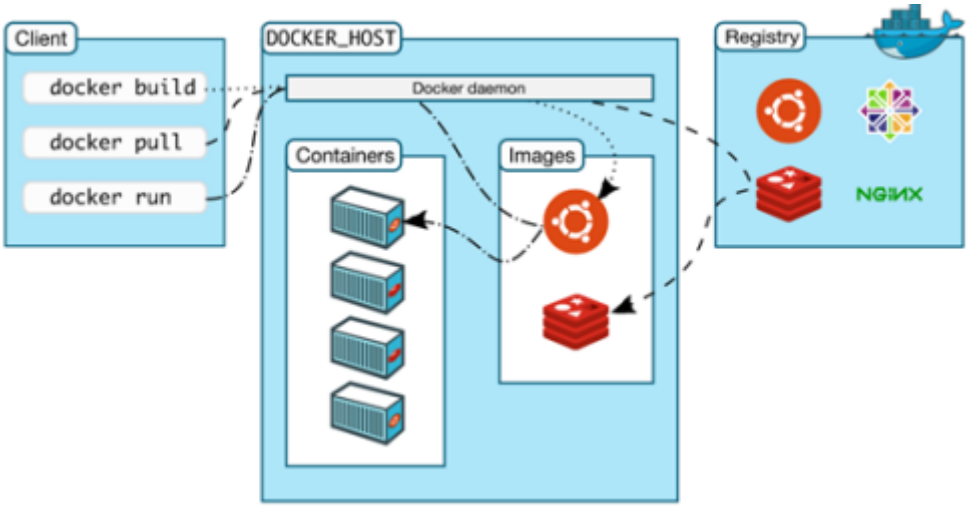
Docker 容器

Docker 利用容器来运行应用，一个Docker容器包含了所有的某个应用运行所需要的环境。每一个 Docker 容器都是从 Docker 镜像创建的。Docker 容器可以运行、开始、停止、移动和删除。每一个 Docker 容器都是独立和安全的应用平台。

容器是从镜像创建的运行实例。它可以被启动、开始、停止、删除。每个容器都是相互隔离的、保证安全的平台。

可以把容器看做是一个简易版的 Linux 环境（包括root用户权限、进程空间、用户空间和网络空间等）和运行在其中的应用程序。

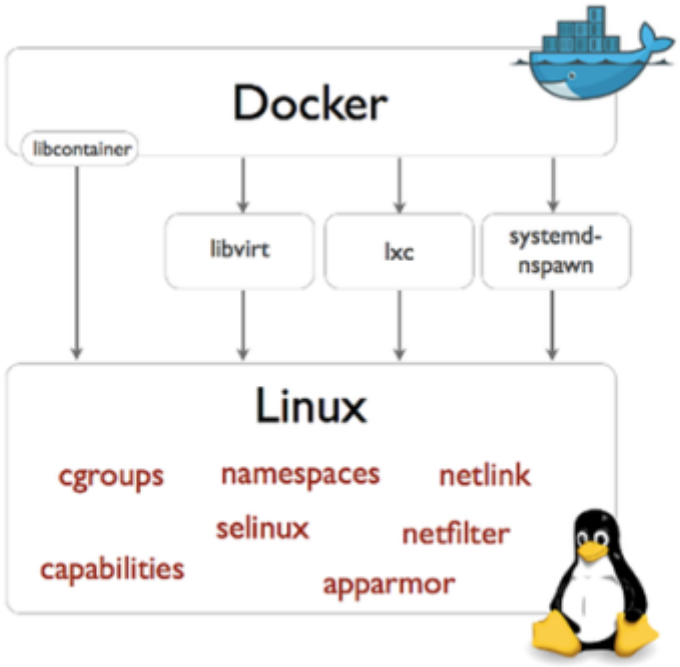
*注：镜像是只读的，容器在启动的时候创建一层可写层作为最上层。




与虚拟机相比，容器有一个很大的差异，它们被设计用来运行"单进程"，无法很好地模拟一个完整的环境。Docker设计者极力推崇“一个容器一个进程的方式”，如果你要选择在一个容器中运行多个进程，那唯一情况是：出于调试目的。

容器是设计来运行一个应用的，而非一台机器。你可能会把容器当虚拟机用，但你将失去很多的灵活性，因为Docker提供了用于分离应用与数据的工具，使得你可以快捷地更新运行中的代码/系统，而不影响数据。

Docker 从 0.9 版本开始使用libcontainer 替代 lxc，libcontainer和 Linux 系统的交互图如下：




Docker 底层技术

docker底层的 2 个核心技术分别是Namespaces 和 Control groups

Namespaces用来隔离各个容器

1)pid namespace

不同用户的进程就是通过pid namespace 隔离开的，且不同 namespace 中可以有相同pid。所有的LXC进程在docker中的父进程为docker进程，每个lxc进

程具有不同的 namespace 。

2) pid namespace
有了pid namespace，每个 namespace 中的pid能够相互隔离，但是网络端口还是共享 host 的端口。网络隔离是通过 net namespace 实现的，每个 net namespace 有独立的 network devices, IPaddresses, IP routing tables, /proc/net 目录。这样每个 container 的网络就能隔离开来。docker默认采用veth的方式将 container 中的虚拟网卡同 host 上的一个docker bridge: docker0 连接在一起。

3) ipc namespace
container 中进程交互还是采用linux常见的进程间交互方法（interprocess communication - IPC),包括常见的信号量、消息队列和共享内存。container 的进程间交互实际上还是host 上具有相同pid namespace 中的进程间交互。

4) mnt namespace
类似chroot，将一个进程放到一个特定的目录执行。mnt namespace 允许不同 namespace 的进程看到的文件结构不同，这样每个 namespace 中的进程所看到的文件目录就被隔离开了。在container里头，看到的文件系统，就是一个完整的linux系统，有/etc、/lib 等，通过chroot实现。

5) uts namespace
UTS("UNIX Time-sharing System") namespace 允许每个 container 拥有独立的 hostname 和 domain name, 使其在网络上可以被视作一个独立的节点而非 Host 上的一个进程。

6) user namespace
每个 container 可以有不同的 user 和 group id，也就是说可以在 container 内部用 container 内部的用户执行程序而非 Host 上的用户。

有了以上 6 种 namespace 从进程、网络、IPC、文件系统、UTS和用户角度的隔离，一个 container 就可以对外展现出一个独立计算机的能力，并且不同 container从 OS 层面实现了隔离。然而不同 namespace 之间资源还是相互竞争的，仍然需要类似ulimit来管理每个 container 所能使用的资源 - -cgroup。
cgroups（Control groups）实现了对资源的配额和度量。

三、Docker 安装
docker官网：https://docs.docker.com
Docker值得关注的特性：
o 文件系统隔离：每个进程容器运行在一个完全独立的根文件系统里。

o 资源隔离：系统资源，像CPU和内存等可以分配到不同的容器中，使用cgroup。
o 网络隔离：每个进程容器运行在自己的网络空间，虚拟接口和IP地址。
o 日志记录：Docker将会收集和记录每个进程容器的标准流（stdout/stderr/stdin），用于实时检索或批量检索。
o 变更管理：容器文件系统的变更可以提交到新的映像中，并可重复使用以创建更多的容器。无需使用模板或手动配置。
o 交互式shell：Docker可以分配一个虚拟终端并关联到任何容器的标准输入上，