Docker入门

什么是LXC？

LXC为Linux Container的简写。Linux Container容器是一种内核虚拟化技术，可以提供轻量级的虚拟化，以便隔离进程和资源，而且不需要提供指令解释机制以及全虚拟化的其他复杂性。相当于C++中的NameSpace。容器有效地将由单个操作系统管理的资源划分到孤立的组中，以更好地在孤立的组之间平衡有冲突的资源使用需求。与传统虚拟化技术相比，它的优势在于：

与宿主机使用同一个内核，性能损耗小；

不需要指令级模拟；

不需要即时(Just-in-time)编译；

容器可以在CPU核心的本地运行指令，不需要任何专门的解释机制；

避免了准虚拟化和系统调用替换中的复杂性；

轻量级隔离，在隔离的同时还提供共享机制，以实现容器与宿主机的资源共享。

总结：Linux Container是一种轻量级的虚拟化的手段。

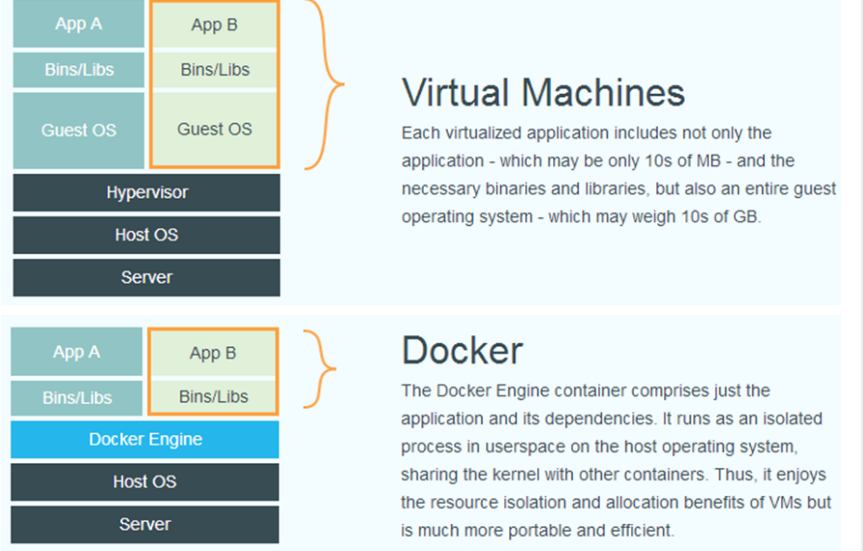
什么是docker？

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（可以商业）。

Docker项目的目标是实现轻量级的操作系统虚拟化解决方案。

Docker是通过内核虚拟化技术（namespaces及cgroups等）来提供容器的资源隔离与安全保障等。由于Docker通过操作系统层的虚拟化实现隔离，所以Docker容器在运行时，不需要类似虚拟机VM额外的操作系统开销，提高资源利用率。

下面图比较了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，直接复制本地主机的操作系统，而传统方式则是在硬件层面实现。



Docker工作模式：

Docker对使用者来讲是一个C/S模式的架构，而Docker的后端是一个非常松耦合的架构，模块各司其职，并有机组合，支撑Docker的运行。

用户是使用Docker Client与Docker Daemon建立通信，并发送请求给后者。

而Docker Daemon作为Docker架构中的主体部分，首先提供Server的功能使其可以接受Docker Client的请求；而后Engine执行Docker内部的一系列工作，每一项工作都是以一个Job的形式的存在。

Job的运行过程中，当需要容器镜像时，则从Docker Registry中下载镜像，并通过镜像管理驱动graphdriver将下载镜像以Graph的形式存储；当需要为Docker创建网络环境时，通过网络管理驱动networkdriver创建并配置Docker容器网络环境；当需要限制Docker容器运行资源或执行用户指令等操作时，则通过execdriver来完成。而libcontainer是一项独立的容器管理包，networkdriver以及execdriver都是通过libcontainer来实现具体对容器进行的操作。当执行完运行容器的命令后，一个实际的Docker容器就处于运行状态，该容器拥有独立的文件系统，独立并且安全的运行环境等。

CLI交互模型

DockerC/S模型



用户通过docker客户端执行命令，如pull一个镜像，到docker的进程，然后由进程将命令的执行结果返回给客户端

RemoteAPI交互模型



Docker应用场景：



1、简化配置,统一配置,通过镜像快速启动(Simplifying)

2、代码流水线管理,开发环境->测试环境->预生产环境->灰度发布->正式发布，docker在这里实现了快速迁移(Code Oioeline Management)

3、开发效率,对开发人员,有了镜像,直接启动容器即可(Developer Productivity)

4、应用隔离,相对于虚拟机的完全隔离会占用资源,docker会比较节约资源(Applsolation)

5、服务器整合,一台服务器跑多个docker容器,提高服务器的利用率(Server Consolidation)

6、调试能力,debug调试(Debugging Capabilties)

7、多租户,一个租户多个用户,类似于阿里公有云的一个project下多个用户(Multi-tenancy)

8、快速部署,不需要启动操作系统,实现秒级部署(Rapid Deplovment)

Doceker八种开发模式：

1.共享基础容器

2.共享卷开发容器

3.开发工具容器

4.不同环境下测试容器

5.构建容器

6.安装容器

7.盒子中默认服务容器

8.基础设施/粘合剂容器

Docker基本事实（优缺点）

1.容器不同于虚拟机

2.容器不如虚拟机来得成熟

3.容器可以在几分之一秒内启动

4.容器已在大规模环境证明了自身的价值

5.IT人员称容器为轻量级

6.容器引发了安全问题

7.Docker已成为容器的代名词，但它不是唯一的提供者

8.容器能节省IT人力，加快更新

9.容器仍面临一些没有解决的问题

使用Docker理由

作为一种新兴的虚拟化方式，Docker跟传统的虚拟化方式具有众多的优势。

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低，一台主机上可以同时运行数千个Docker容器。

至于为什么要使用Docker：

1、 技术储备

相对大公司这个非常重要,如果你们都在用,他们不用就落后了,等到完全成熟以后就跟不上了。

2、 无技术栈和技术债

没有任何Openstack或者saltstack,服务down了就down了,所有的服务都是松耦合。

3、跟上潮流(提升自我,装逼)

面试的时候大家都会Docker,你不会是不是落后了。

4、符合当前业务

虽然Docker很优秀,但是，大多数处在第二种状态,很少有符合自己的业务

Docker改变了什么

面向产品：产品交付

面向开发：简化环境配置

面向测试：多版本测试

面向运维：环境一致性

面向架构：自动化扩容(微服务)

Docker更快速的交付和部署

对于开发和人员来说，最希望的就是一次创建和配置，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，并让整个过程全称可见，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。Docker容器很轻很快！容器的启动时间是秒级的，大量第节约开发、测试、部署的时间。

Docker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。

Docker更轻松的迁移和扩展

Docker容器几乎可以字啊任意的平台上运行，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。

Docker更简单的管理

使用Docker，只需要小小的修改，就可以替代往大量的更新工作。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。

Docker和虚拟化（openstack）区别



Docker3大概念

Docker镜像(image)

Docker镜像就是一个只读的模板。

例如：一个镜像可以包含一个完整的CentOS操作系统环境，里面仅安装了Apache或用户需要的其他应用程序。

镜像可以用来创建Docker容器。

Docker提供了一个很简单的机制来创建镜像或者更新现有的镜像，用户甚至可以直接从其他人那里下载一个已经做好的镜像来直接使用。

Docker容器(container)

Docker利用容器来运行应用。

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。

注意：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。

Docker仓库(repository)

仓库是集中存放镜像文件的场所。有时候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读的访问。

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，这样下载在另外一台机器上使用这个镜像时候，只需需要从仓库上pull下来就可以了。

注意：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实战

环境

|  |  |
| --- | --- |
| 192．168．1．156 | Centos7 |

因为Centos7环境直接有docker软件，所以这里采用的是Centos7的环境

首先，安装Docker

[root@xiaohu2 ~]# yum -y install docker

查看docker版本

[root@xiaohu2 ~]# docker version

Client:

Version: 1.10.3

API version: 1.22

Package version: docker-common-1.10.3-44.el7.centos.x86\_64

Go version: go1.4.2

Git commit: 9419b24-unsupported

Built: Fri Jun 24 12:09:49 2016

OS/Arch: linux/amd64

通过docker获取centos镜像

[root@xiaohu2 ~]# docker search centos

INDEX NAME DESCRIPTION STARS OFFICIAL AUTOMATED

docker.io docker.io/centos The official build of CentOS. 2410 [OK]

docker.io docker.io/ansible/centos7-ansible Ansible on Centos7 78 [OK]

docker.io docker.io/jdeathe/centos-ssh CentOS-6 6.8 x86\_64 / CentOS-7 7.2.1511 x8... 26 [OK]

获取docker镜像

[root@xiaohu2 ~]# docker pull centos

为该镜像创建容器，这个稍等时间，时间比较长

[root@xiaohu2 ~]# docker run -it centos /bin/bash

查看docker镜像

[root@xiaohu2 ~]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

docker.io/centos latest 05188b417f30 5 days ago 196.8 MB

导出Docker镜像到本地

[root@xiaohu2 ~]# docker save centos > /opt/centos.tar.gz

[root@xiaohu2 ~]# cd /opt/

[root@xiaohu2 opt]# ls

centos.tar.gz

导入docker镜像

[root@xiaohu2 ~]# docker load < /opt/centos.tar.gz

[root@xiaohu2 ~]# docker images 查看镜像

REPOSITORY TAG IMAGE ID CREATED SIZE

docker.io/centos latest 05188b417f30 5 days ago 196.8 MB

Docker容器管理

启动Docker容器：

启动容器有两种方式，一种是基于镜像新建一个容器并启动，另外一个是将在终止状态(stopped)的容器重新启动。

因为Docker的容器实在太轻量级了，很多时候用户都是随时删除和新创建容器

新建容器并启动

所需要的命令docker run

[root@xiaohu2 ~]# docker run centos /bin/echo "xiaohu" 在本地直接执行

xiaohu

[root@xiaohu2 ~]# docker run --name mydocker -it centos /bin/bash 启动一个bash终端，允许用户交替

[root@d09df36333ac /]# pwd

/

[root@d09df36333ac /]# ls

anaconda-post.log dev home lib64 media opt root sbin sys usr

bin etc lib lost+found mnt proc run srv tmp var

--name:给容器定义一个名称

-i:则让容器的标准输入保持打开。

-t:让Docker分配一个伪终端,并绑定到容器的标准输入上

/bin/bash:执行一个命令

启动容器

[root@xiaohu2 ~]# docker start d09df36333ac

d09df36333ac

查看是否启动

[root@xiaohu2 ~]# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

d09df36333ac centos "/bin/bash" 2 minutes ago Up 20 seconds mydocker

a84ac2c9f14e centos "/bin/echo xiaohu" 3 minutes ago Exited (0) 3 minutes ago jovial\_hypatia

f1461cb73478 centos "/bin/bash" 12 minutes ago Exited (0) 11 minutes ago pedantic\_lamarr

停止容器

[root@xiaohu2 ~]# docker stop d09df36333ac

删除容器

[root@xiaohu2 ~]# docker rm d09df36333ac 删除命令

d09df36333ac

[root@xiaohu2 ~]# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

a84ac2c9f14e centos "/bin/echo xiaohu" 5 minutes ago Exited (0) 5 minutes ago jovial\_hypatia

f1461cb73478 centos "/bin/bash" 15 minutes ago Exited (0) 13 minutes ago pedantic\_lamarr