Docker基本概念

教学内容

- 第一节 什么是Docker
- 第二节 Docker与虚拟机
- 第三节 Docker的组成

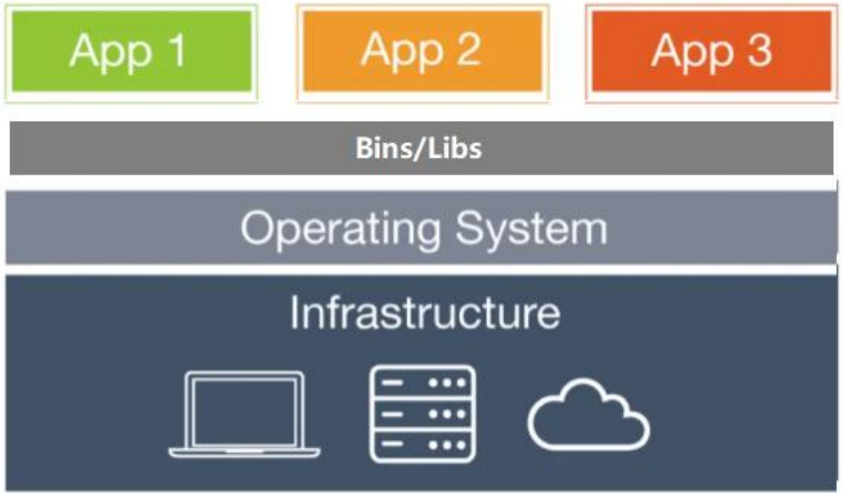
环境配置的难题

- 软件开发最大的麻烦事之一，就是环境配置，用户计算机环境各不相同，配置也要根据环境相应变化。
- 操作系统的设置，各种库和组件的安装。只有它们都正确，软件才能运行，比如开发Java应用，计算机必须配置JDK。
- 如果某些依赖过时与当前环境不兼容，那更是一场灾难。

应用部署的问题

同样在应用部署时，由于组件较多，运行环境复杂，也会遇到类似问题：

- 将不同的软件集成起来的过程中有很多不可控的风险，由于依赖关系复杂，容易出现兼容性问题
- 由于开发、测试、生产环境的差异，一旦需要重新迁移服务器或者重新部署一套环境，还将重新执行一遍



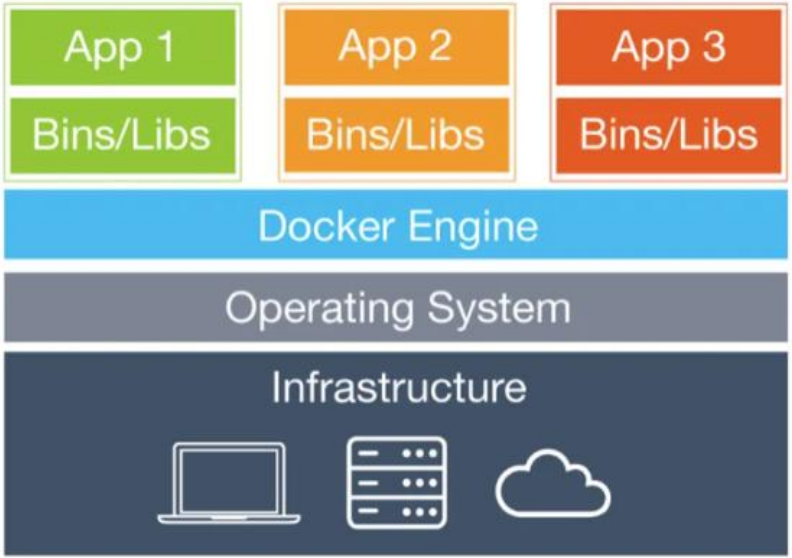
如何解决

- 环境配置如此麻烦，换一台机器，就要重来一次，能不能从根本上解决问题呢
- 如果软件连同环境一起打包，那就解决了上述问题，也就是说，使用软件时，把原始环境一模一样地复制过来。

什么是Docker

Docker如何解决依赖的兼容问题?

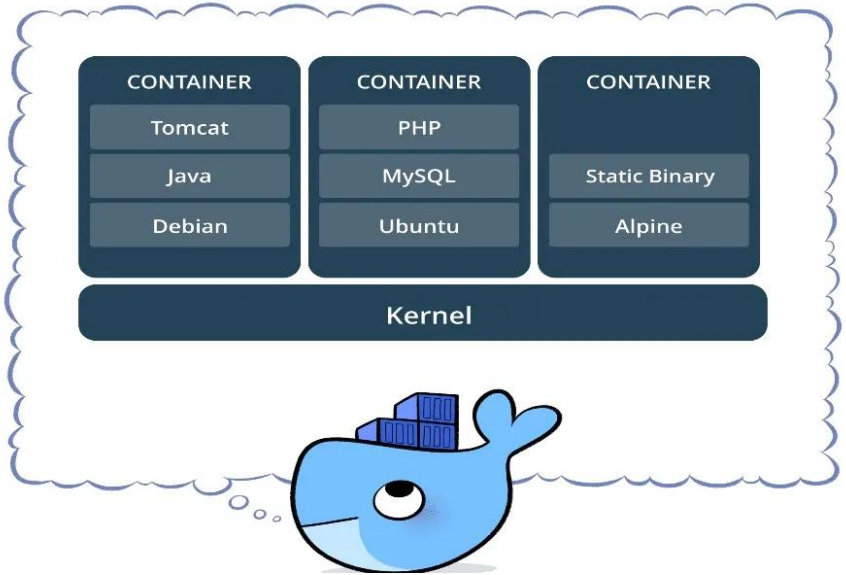
- 将应用的Bins/Libs与应用一起打包
- 将每个应用放到一个隔离容器去运行，避免互相干扰



什么是Docker

Docker如何解决不同系统环境的问题?

- Docker将用户程序与所需要调用的系统函数库一起打包
- Docker在运行到不同操作系统时，直接通过内核创建虚拟的操作系统实例（内核和库），来隔离不同的进程和资源



什么是Docker

- Docker允许开发中将应用、依赖、函数库、配置一起打包，形成可移植镜像
- Docker应用运行在容器中，使用沙箱机制，相互隔离
- Docker镜像中包含完整运行环境，包括系统函数库，仅依赖系统的Linux内核，因此可以在任意Linux操作系统上运行

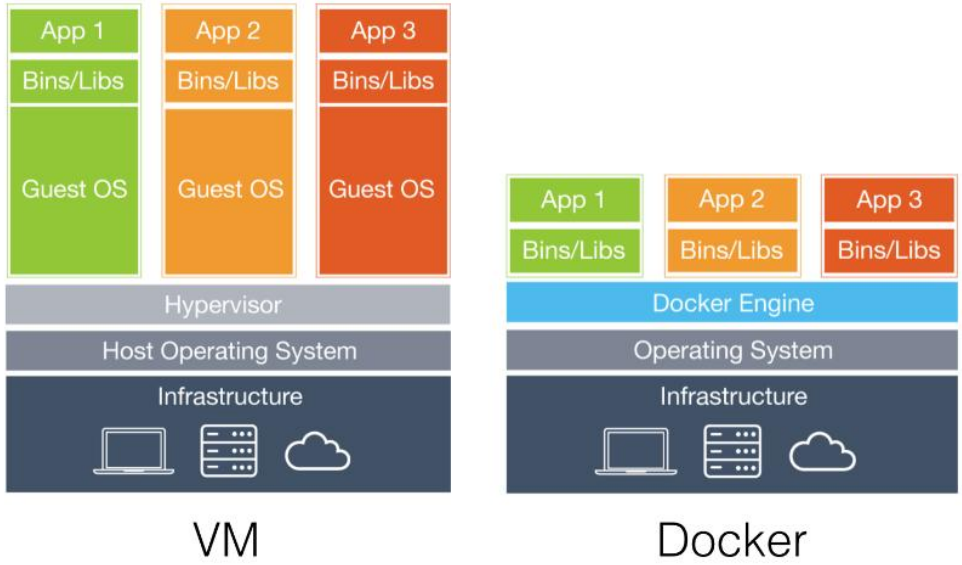
什么是Docker

Docker 的主要用途，目前有三大类。

- 提供一次性的环境。比如，本地测试他人的软件、持续集成的时候提供单元测试和构建的环境。
- 提供弹性的云服务。因为 Docker 容器可以随开随关，很适合动态扩容和缩容。
- 组建微服务架构。通过多个容器，一台机器可以跑多个服务，因此在本机就可以模拟出微服务架构。

Docker与虚拟机

- 虚拟机（virtual machine）是带环境安装的另一种解决方案，它可以在一种操作系统里面运行另一种操作系统，比如在 Windows 系统里面运行 Linux 系统。
- 应用程序对此毫无感知，因为虚拟机看上去跟真实系统一模一样，而对于底层系统来说，虚拟机就是一个普通文件，不需要了就删掉，对其他部分毫无影响。



Docker与虚拟机

虽然用户可以通过虚拟机还原软件的原始环境。但是，这个方案有几个缺点。

- 资源占用多,虚拟机会独占一部分内存和硬盘空间
- 冗余步骤多,虚拟机是完整的操作系统，需要经过繁琐的配置
- 启动慢，启动操作系统需要多久，启动虚拟机就需要多久。可能要等几分钟，应用程序才能真正运行。

Docker与虚拟机

Docker和虚拟机都是基于软件的平台虚拟化技术，其中：

- 虚拟机属于完全虚拟化，即模拟完整的底层硬件环境特权指令的执行，客户操作系统无需进行修改。比如我们常用的VirtualBox，VMWare Workstation和Parallels Desktop等虚拟化软件。
- Docker容器技术是操作系统级虚拟化，即直接通过内核创建虚拟的操作系统实例（内核和库），来隔离不同的进程和资源， 多个容器可以在同一台机器上运行，共享操作系统内核，但各自作为独立的进程在用户空间中运行。
- Docker容器不需要额外的虚拟机管理软件和虚拟机操作系统层，直接在宿主机操作系统层面上实现虚拟化，从而达到轻量级，高效的目的。

Docker的组成

Docker 包括三个基本概念

- 镜像（Image）
- 容器（Container）
- 仓库（Repository）
- 理解了这三个概念，就理解了 Docker 的整个生命周期。

镜像

- Docker 把应用程序及其依赖，打包在 image 文件里面。只有通过这个文件，才能生成 Docker 容器。
- image 文件可以看作是容器的模板，除了提供容器运行时所需的程序、库、资源、配置等文件外，还包含了一些为运行时准备的一些配置参数（如匿名卷、环境变量、用户等）。
- Docker 根据 image 文件生成容器的实例。同一个 image 文件，可以生成多个同时运行的容器实例。
- 镜像不包含任何动态数据，其内容在构建之后也不会被改变。

镜像

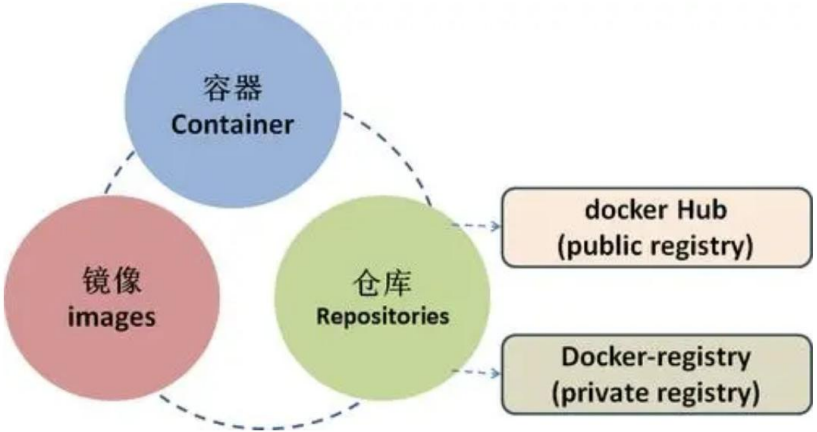
- image 是二进制文件。实际开发中，一个 image 文件往往通过继承另一个 image 文件，加上一些个性化设置而生成。举例来说，你可以在 Ubuntu 的 image 基础上，往里面加入 Apache 服务器，形成你的 image。
- image 文件是通用的，一台机器的 image 文件拷贝到另一台机器，照样可以使用。一般来说，为了节省时间，我们应该尽量使用别人制作好的 image 文件，而不是自己制作。即使要定制，也应该基于别人的 image 文件进行加工，而不是从零开始制作。
- 为了方便共享，image 文件制作完成后，可以上传到网上的仓库。Docker 的官方仓库 Docker Hub 是最重要、最常用的 image 仓库。此外，出售自己制作的 image 文件也是可以的。

仓库

- 镜像构建完成后，可以很容易的在当前宿主机上运行，但是，如果需要在其它服务器上使用这个镜像，我们就需要一个集中的存储、分发镜像的服务，Docker Registry 就是这样的服务。
- 一个 Docker Registry 中可以包含多个仓库（Repository）；每个仓库可以包含多个 标签（Tag）；每个标签对应一个镜像。
- 通常，一个仓库会包含同一个软件不同版本的镜像，而标签就常用于对应该软件的各个版本。我们可以通过 <仓库名>:<标签> 的格式来指定具体是这个软件哪个版本的镜像。如果不给出标签，将以 latest 作为默认标签。
- 以 Ubuntu 镜像 为例，ubuntu 是仓库的名字，其内包含有不同的版本标签，如，16.04, 18.04。我们可以通过 ubuntu:16.04，或者 ubuntu:18.04 来具体指定所需哪个版本的镜像。如果忽略了标签，比如 ubuntu，那将视为 ubuntu:latest。

仓库

- DockerHub：DockerHub是一个Docker镜像的托管平台。这样的平台称为Docker Registry。
- 国内也有类似于DockerHub 的公开服务，比如 网易云镜像服务、阿里云镜像库等。

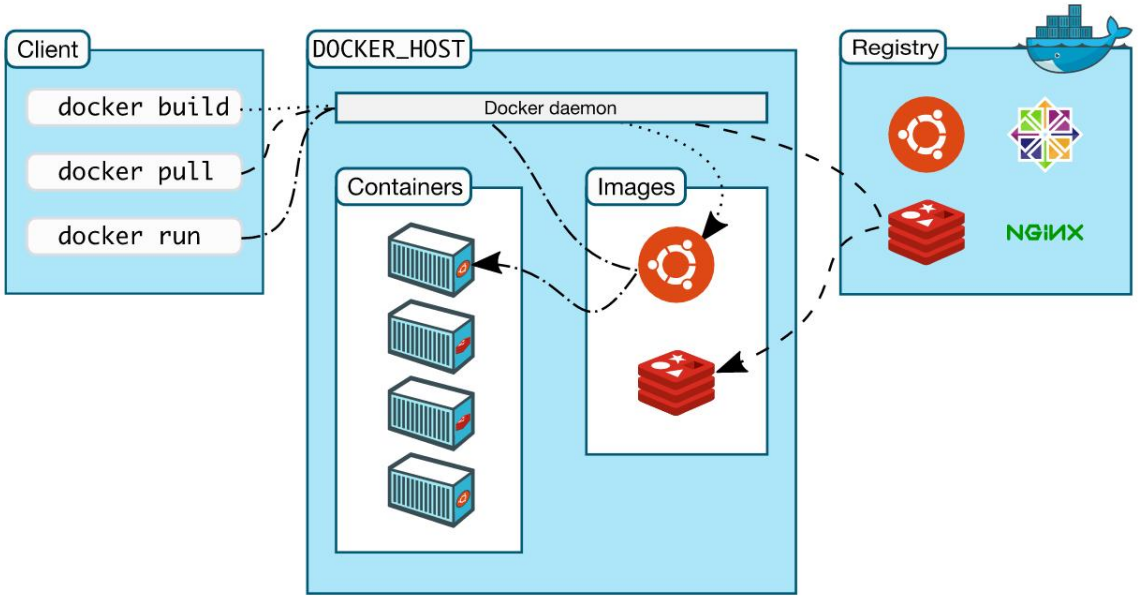


容器

- 镜像（Image）和容器（Container）的关系，就像是面向对象程序设计中的 类 和 实例 一样，镜像是静态的定义，容器是镜像运行时的实体。
- 简单的说，容器是独立运行的一个或一组应用，以及它们的运行态环境，容器可以被创建、启动、停止、删除、暂停等。
- 容器的实质是进程，但与直接在宿主执行的进程不同，容器进程运行于属于自己的独立的命名空间。因此容器可以拥有自己的 root 文件系统、自己的网络配置、自己的进程空间，甚至自己的用户 ID 空间。
- 容器内的进程是运行在一个隔离的环境里，使用起来，就好像是在一个独立于宿主的系统下操作一样。这种特性使得容器封装的应用比直接在宿主运行更加安全。

Docker架构

- 客户端(client)：通过命令或RestAPI向Docker服务端发送指令。可以在本地或远程向服务端发送指令。
- 服务端(server)：Docker守护进程，负责处理Docker指令，管理镜像、容器等



安装Docker

- Docker 分为 CE 和 EE 两大版本。CE 即社区版（免费，支持周期 7 个月），EE 即企业版，强调安全，付费使用，支持周期 24 个月。
- Docker CE 支持 64 位版本 CentOS 7，并且要求内核版本不低于 3.10，CentOS 7 满足最低内核的要求。

安装Docker

- 参考安装文档