

Docker入门-概念

- Docker入门-概念
 - Docker简介
 - 什么是Docker
 - Docker特点
 - 为什么要使用Docker
 - 生命周期

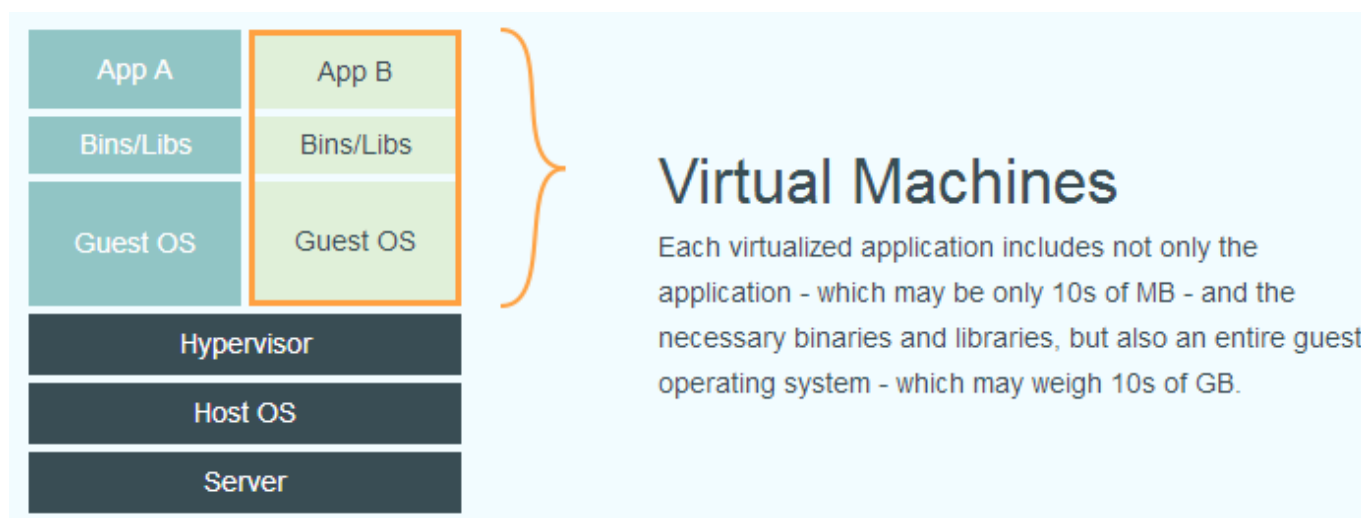
Docker简介

什么是Docker

Docker 最初是 dotCloud 公司创始人 Solomon Hykes 在法国期间发起的一个公司内部项目，并于2013 年 3 月以 Apache 2.0 授权协议开源，主要项目代码在 [GitHub](#) 上进行维护，后来还加入了 Linux 基金会，并成立推动 [开放容器联盟（OCI）](#)。

Docker 最初是在 Ubuntu 12.04 上以[Go 语言](#) 进行开发实现的, Red Hat 则从 RHEL 6.5 开始对 Docker 进行支持 (换句话说不支持CentOS6.5以下)。

Docker 是一种 容器化技术，类似虚拟机的概念，但不同的是传统虚拟机是在虚拟硬件的基础上，完整模拟一整个操作系统，而Docker是以单个应用（容器）为单位进行虚拟。



Virtual Machines

Each virtualized application includes not only the application - which may be only 10s of MB - and the necessary binaries and libraries, but also an entire guest operating system - which may weigh 10s of GB.

Docker

The Docker Engine container comprises just the application and its dependencies. It runs as an isolated process in userspace on the host operating system, sharing the kernel with other containers. Thus, it enjoys the resource isolation and allocation benefits of VMs but is much more portable and efficient.

Docker特点

Docker具有以下特点：

- **文件系统隔离**：每个进程容器运行在完全独立的根文件系统里。
- **资源隔离**：可以使用cgroup为每个进程容器分配不同的系统资源，例如CPU和内存。
- **网络隔离**：每个进程容器运行在自己的网络命名空间里，拥有自己的虚拟接口和IP地址。
- **写时复制**：采用写时复制方式创建根文件系统，这让部署变得极其快捷，并且节省内存和硬盘空间。
- **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。
- **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。
- **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，例如运行一个一次性交互shell。

为什么要使用Docker

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	弱于原生
系统支持量	单机支持上千个容器	一般几十个

- **更高效的利用系统资源**：由于容器不需要进行硬件虚拟以及运行完整操作系统等额外开销，Docker 对系统资源的利用率更高。相比虚拟机技术，一个相同配置的主机，往往可以运行更多数量的应用。
- **更快速的启动时间**：Docker 容器应用，由于直接运行于宿主内核，无需启动完整的操作系统，因此可以做到秒级、甚至毫秒级的启动时间。
- **一致的运行环境**：Docker 的镜像提供了除内核外完整的运行时环境，确保了应用运行环境一致性，从而不会再出现「这段代码在我机器上没问题啊」这类问题。
- **持续交付和部署**：对DevOps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。使用Docker 可以通过定制应用镜像来实现持续集成、持续交付、部署。开发人员可以通过 Dockerfile 来进行镜像构建，并结合 持续集成(Continuous Integration) 系统进行集成测试，而运维人员则可以直接在生产环境中快速部署该镜像，甚至结合 持续部署(Continuous Delivery/Deployment)系统进行自动部署。
- **更轻松的迁移**：由于 Docker 确保了执行环境的一致性，使得应用的迁移更加容易。Docker 可以在很多平台上运行，无论是物理机、虚拟机、公有云、私有云，甚至是笔记本，其运行结果是一致的。因此用户可以很轻易的将在一个平台上运行的应用，迁移到另一个平台上，而不用担心运行环境的变化导致应用无法正常运行的情况。

生命周期

我们主要从三个层面去理解docker,分别为运行层，资源利用层，以及仓库

1. 运行层：镜像和容器是docker里面很核心的两个概念，是我们使用docker运行的基础。
2. 资源利用层：使用存储卷和网络可以帮助我们更好的利用资源，以及更加规范化。

3. 仓库：镜像资源库，包括共有仓库和私有仓库，使用仓库可以帮我们查找、管理、拉取和推送镜像。

结合上面的概念，这里有一张图比较好的概括了整个Docker工作的生命周期（以及主要命令）。

