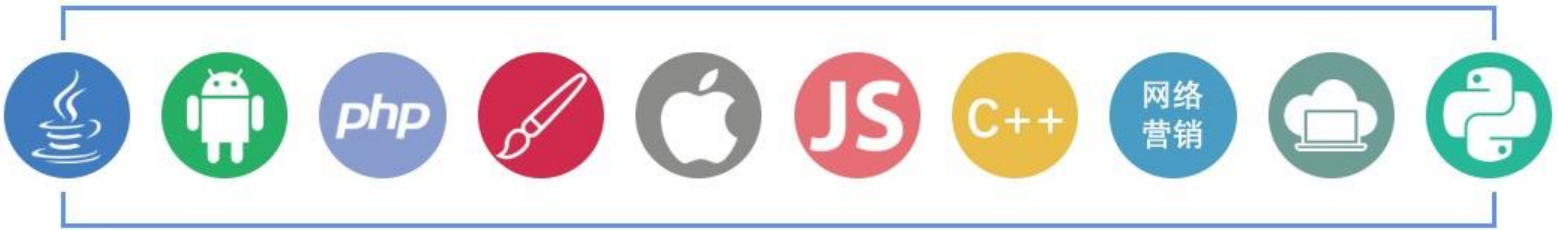
Docker概览



Docker概览 - 课程概要

- Docker 简介
- Docker 整体结构了解
- Docker 底层技术了解
- 总结

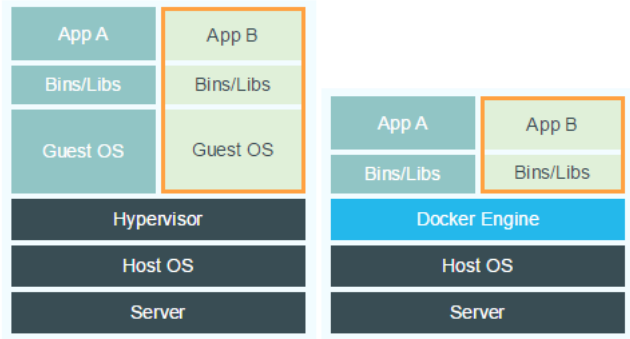
Docker概览

Docker简介

什么是Docker

Docker是开发，运行和部署应用程序的开放管理平台。

- 开发人员能利用docker 开发和运行应用程序
- 运维人员能利用docker 部署和管理应用程序



VM（Virtual Machine）　　Docker

Docker平台介绍 (The Docker platform)

- Docker提供了在一个完全隔离的环境中打包和运行应用程序的能力，这个隔离的环境被称为容器。
- 由于容器的隔离性和安全性，因此可以在一个主机(宿主机)上同时运行多个相互隔离的容器，互不干预。
- Docker已经提供工具和组件(Docker Client、Docker Daemon等)来管理容器的生命周期：
  - 使用容器来开发应用程序及其支持组件。
  - 容器成为分发和测试你的应用程序的单元。
  - 准备好后，将您的应用程序部署到生产环境中，作为容器或协调服务。无论您的生产环境是本地数据中心，云提供商还是两者的混合，这都是一样的。

为什么使用Docker?

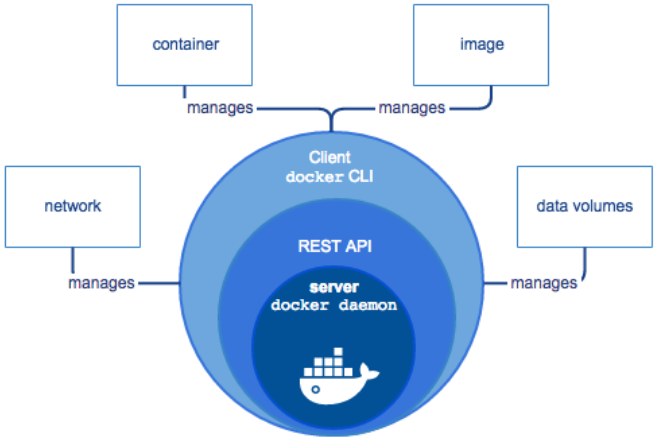
- Docker使您能够将应用程序与基础架构分开，以便您可以快速交付软件。
- 借助Docker，您可以像管理应用程序一样管理基础架构。
- 通过利用Docker的方法快速进行运输，测试和部署代码，您可以显着缩短编写代码和在生产环境中运行代码之间的延迟。
- 如：
  - 开发人员在本地编写代码，可以使用Docker同事进行共享，实现协同工作。
  - 使用Docker开发完成程序，可以直接对应用程序执行自动和手动测试。
  - 当开发人员发现错误或BUG时，可以直接在开发环境中修复后，并迅速将它们重新部署到测试环境进行测试和验证。
  - 利用Docker开发完成后，交付时，直接交付Docker，也就意味着交付完成。后续如果有提供修补程序或更新，需要推送到生成环境运行起来，也是一样的简单。
- Docker主要解决的问题：
  - **保证程序运行环境的一致性；**
  - **降低配置开发环境、生产环境的复杂度和成本；**
  - **实现程序的快速部署和分发。**

Docker概览

Docker整体结构了解

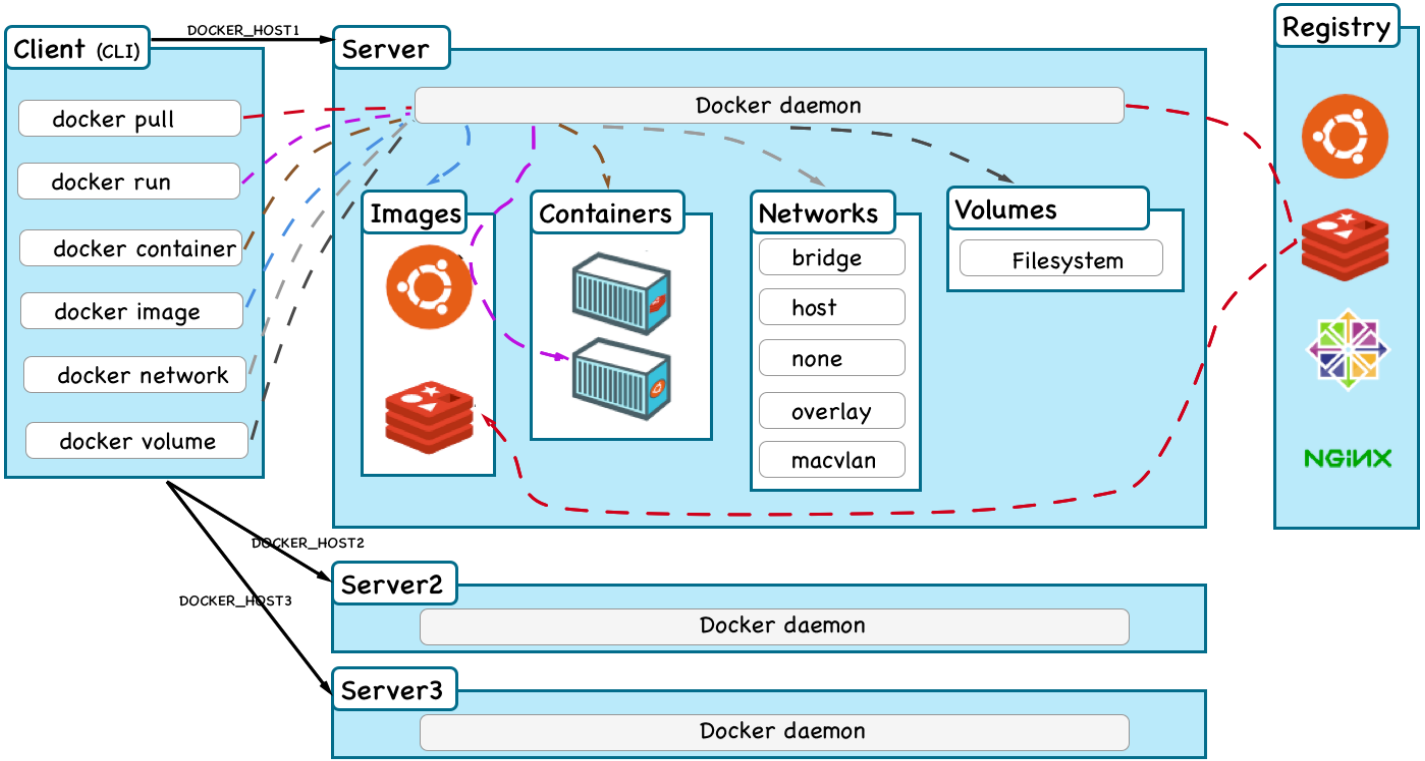
Docker引擎介绍 (Docker Engine)

- Docker Engine是一个包含以下组件的客户端-服务端(C/S)应用程序
  - 服务端 --- 一个长时间运行的守护进程(Docker Daemon)
  - REST API --- 一套用于与Docker Daemon通信并指示其执行操作的接口
  - 客户端 --- 命令行接口CLI(Command Line Interface)



- CLI利用Docker命令通过REST API直接操控Docker Daemon执行操作
- Docker Daemon负责创建并管理Docker的对象(镜像、容器、网络、数据卷)

Docker结构概览图



Docker结构简介

- Docker客户端(Docker Client)
  - Docker客户端(Docker Client)是用户与Docker进行交互的最主要方式。当在终端输入docker命令时，对应的就会在服务端产生对应的作用，并把结果返回给客户端。Docker Client除了连接本地服务端，通过更改或指定DOCKER_HOST连接远程服务端。
- Docker服务端(Docker Server)
  - Docker Daemon其实就是Docker 的服务端。它负责监听Docker API请求(如Docker Client)并管理Docker对象(Docker Objects)，如镜像、容器、网络、数据卷等
- Docker Registries
  - 俗称Docker仓库，专门用于存储镜像的云服务环境.
  - Docker Hub就是一个公有的存放镜像的地方，类似Github存储代码文件。同样的也可以类似Github那样搭建私有的仓库。
- Docker 对象(Docker Objects)
  - 镜像：一个Docker的可执行文件，其中包括运行应用程序所需的所有代码内容、依赖库、环境变量和配置文件等。
  - 容器：镜像被运行起来后的实例。
  - 网络：外部或者容器间如何互相访问的网络方式，如host模式、bridge模式。
  - 数据卷：容器与宿主机之间、容器与容器之间共享存储方式，类似虚拟机与主机之间的共享文件目录。

Docker概览

Docker 底层技术了解

Docker底层使用的技术介绍（一）

- Docker使用Go语言实现。
- Docker利用linux内核的几个特性来实现功能:
  - 利用linux的命名空间(Namespaces)
  - 利用linux控制组(Control Groups)
  - 利用linux的联合文件系统(Union File Systems)

  这也就意味着Docker只能在linux上运行。

  在windows、MacOS上运行Docker，其实本质上是借助了虚拟化技术，然后在linux虚拟机上运行的Docker程序。
- 容器格式（ Container Format ）：
  - Docker Engine将namespace、cgroups、UnionFS进行组合后的一个package，就是一个容器格式(Container Format)。Docker通过对这个package中的namespace、cgroups、UnionFS进行管理控制实现容器的创建和生命周期管理。
  - 容器格式(Container Format)有多种，其中Docker目前使用的容器格式被称为libcontainer

Docker底层使用的技术介绍（二）

- Namespaces（命名空间）：为Docker容器提供操作系统层面的隔离
  - 进程号隔离：每一个容器内运行的第一个进程，进程号总是从1开始起算
  - 网络隔离：容器的网络与宿主机或其他容器的网络是隔离的、分开的，也就是相当于两个网络
  - 进程间通隔离：容器中的进程与宿主机或其他容器中的进程是互相不可见的，通信需要借助网络
  - 文件系统挂载隔离: 容器拥有自己单独的工作目录
  - 内核以及系统版本号隔离：容器查看内核版本号或者系统版本号时，查看的是容器的，而非宿主机的
- Control Groups（控制组-cgroups）：为Docker容器提供硬件层面的隔离
  - 控制组能控制应用程序所使用的硬件资源。
  - 基于该性质，控制组帮助docker引擎将硬件资源共享给容器使用，并且加以约束和限制。如控制容器所使用的内存大小。
- Union File Systems（联合文件系统--UnionFS）：利用分层(layer)思想管理镜像和容器

Docker概览

总结

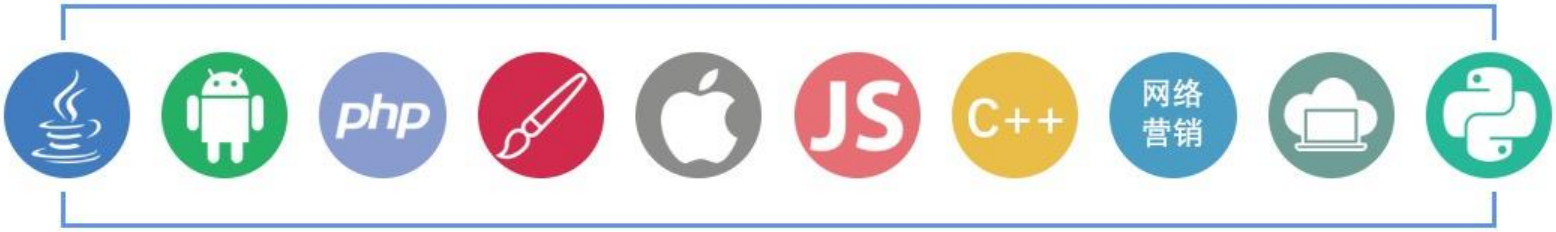
Docker概览-总结

重点掌握Docker以下组件或对象直接的关系

- Docker Client CLI
- Docker Daemon（Docker server）
- Docker Objects
- Registry

了解以下列出的Docker使用的底层技术

- Namespaces
- Control Groups
- Union File Systems
- Container Format



Thank you!

改变中国IT教育，我们正在行动