52ABP实战课程系列

Docker&Ubuntu从入门到实战课程

梁桐铭

2017年12月

任何的课程都逃不开理论的支持 by 角落的白板报

目录 | CONTENTS

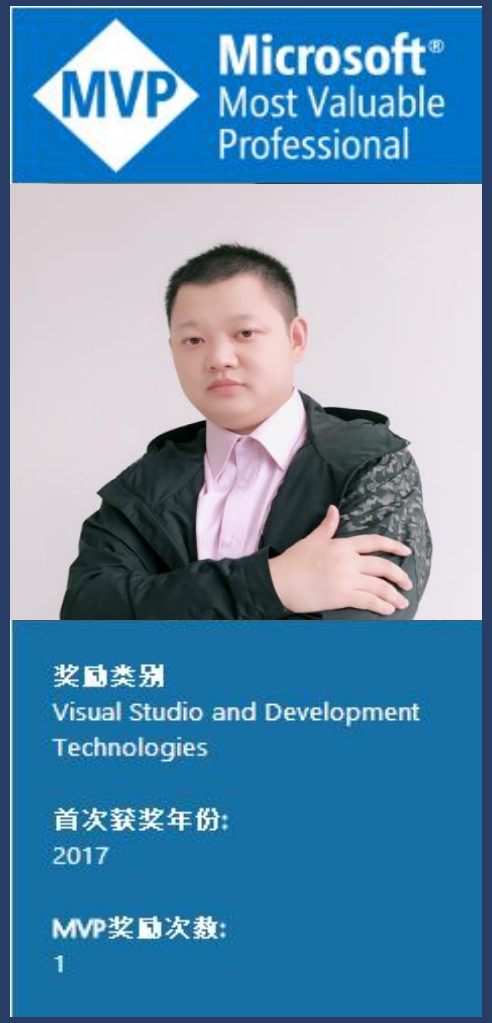
Chapter 1

工欲善其事必先利其器

适合你吗？

- 任何的课程都逃不开理论的支持
- 您如果想了解Docker，想利用Docker进行开发，那么本系列课程就比较适合你了。
- 我会讲解Docker，配合Ubuntu来使用，实践一个案例。

自我介绍

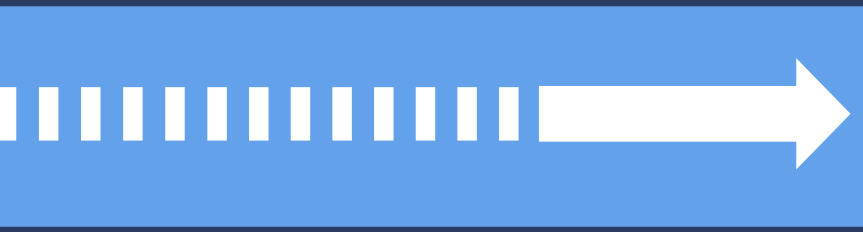


微软2017年最有价值专家（MVP）称号

2015年5月开始在国内的开源社区中进行推广ASP.NET Boilerplate Project开源框架，创建了“角落的白板报”博客及公众号，发表大量基础性文章，帮助从业人员提高开发技巧!

代码生成器(ABP Code Generator)作者，辅助ABP框架以提高开发人员的效率。

现主要关注于VSTS、Devops、区块链等方向的实践和落地。



Chapter 2

Docker的介绍

什么是容器

容器化是软件开发的一种方法，在这种方法中，程序和它所依赖的组件和集合包，以及相关的环境变量配置文件都会被完全打包成容器镜像，进行单元测试，最后将这个容器部署到服务器的操作系统中。

举个栗子：这个与现实生活中的货运集装箱类似，集装箱里面有各种货物，都分门别类的装好了，我们可以通过汽车、火车、飞机来搬运他们。

程序的集装箱是一个标准的单元，无论代码、语言、软件/框架他们的依赖关系是怎样的，它们都被统一包含在了这个单元的内部中。这使得程序员和IT运维专员不用在每个环境中配置不同的配置，它可以在部署程序的过程中不进行修改或少许修改的情况下，达到跨环境传输，从而使得每一个容器中的程序又是彼此隔离的。

容器的作用是在共享的操作系统中将程序进行彼此隔离。这种方法可以使程序的交付标准化。

容器带来的好处

- 隔离性
- 可移植性
- 灵活性
- 可伸缩性和可控性。

最重要的好处是Dev(开发）和Ops（运维）之间提供了隔离。

Docker的历史

- 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，它是基于 dotCloud 公司多年云服务技术的一次革新，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。Docker 项目后来还加入了 Linux 基金会，并成立推动开放容器联盟。
- Docker 自开源后受到广泛的关注和讨论，至今其 GitHub 项目已经超过 3 万 6 千个星标和一万多个 fork。甚至由于 Docker 项目的火爆，在 2013 年底，dotCloud 公司决定改名为 Docker。Docker 最初是在 Ubuntu 12.04 上开发实现的；Red Hat 则从 RHEL 6.5 开始对 Docker 进行支持；Google 也在其 PaaS 产品中广泛应用 Docker。
- Docker 使用 Google 公司推出的 Go 语言 进行开发实现，基于 Linux 内核的 cgroup，namespace，以及 AUFS 类的 Union FS 等技术，对进程进行封装隔离，属于操作系统层面的虚拟化技术。由于隔离的进程独立于宿主和其它的隔离的进程，因此也称其为容器。最初实现是基于 LXC，从 0.7 以后开始去除 LXC，转而使用自行开发的 libcontainer，从 1.11 开始，则进一步演进为使用 runC 和 containerd。
- Docker 在容器的基础上，进行了进一步的封装，从文件系统、网络互联到进程隔离等等，极大的简化了容器的创建和维护。使得 Docker 技术比虚拟机技术更为轻便、快捷。

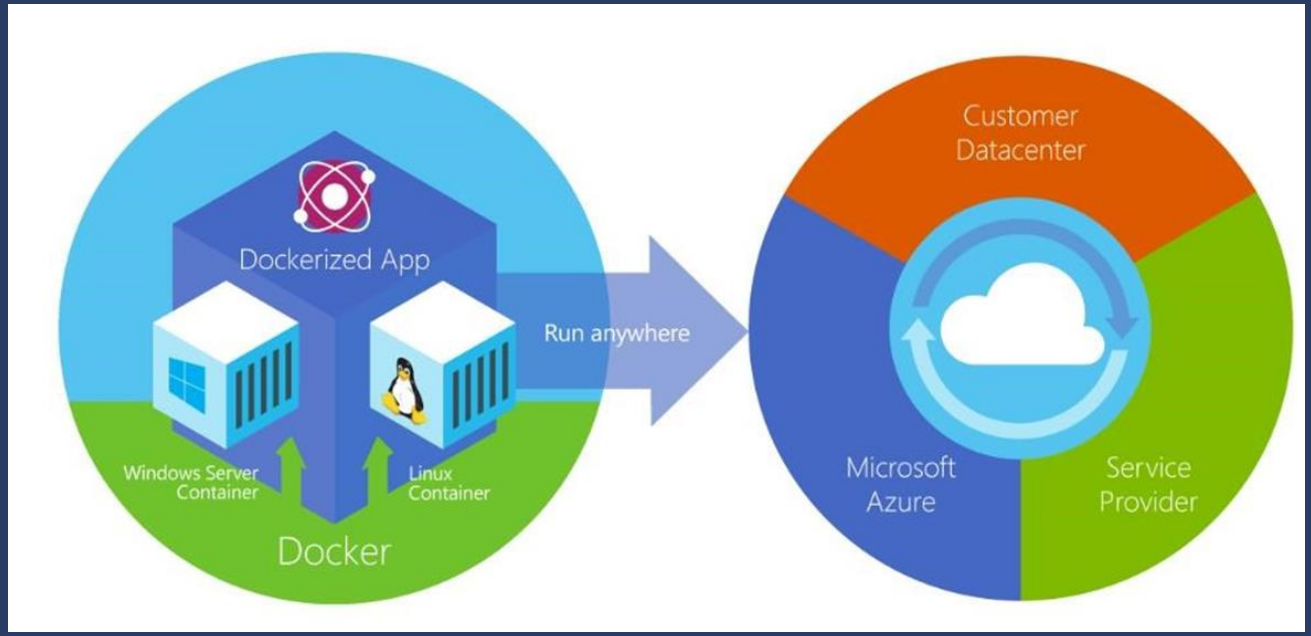
什么是Docker？

Docker 是一个 开源项目， 是一个可以将程序自动化部署到任何云或者本地的一种可移植的容器。

Docker 也是一家公司 ， 通过与(Azure)微软云，Linux和Windows等供应商的紧密合作来推动和发展这项技术。

Docker正在成为标准的部署单元， 正在被大多数软件平台和云供应商包括 (Microsoft Azure，Amazon AWS，Google等作为标准的容器。

关于支持的操作系统，Docker容器可以在Linux和Windows上本地运行。



为什么要使用Docker？

△ 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。无论是应用执行速度、内存损耗或者文件存储速度，都要比传统虚拟机技术更高效。因此，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

传统的虚拟机技术启动应用服务往往需要数分钟，而 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。大大的节约了开发、测试、部署的时间。

△ 一致的运行环境

开发过程中一个常见的问题是环境一致性问题。由于开发环境、测试环境、生产环境不一致，导致有些 bug 并未在开发过程中被发现。而 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。

为什么要使用Docker？

△ 持续交付和部署

对开发和运维（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(ContinuousDelivery/Deployment）系统进行自动部署。而且使用 Dockerfile 使镜像构建透明化，不仅仅开发团队可以理解应用运行环境，也方便运维团队理解应用运行所需条件，帮助更好的生产环境中部署该镜像。

△ 更轻松的迁移

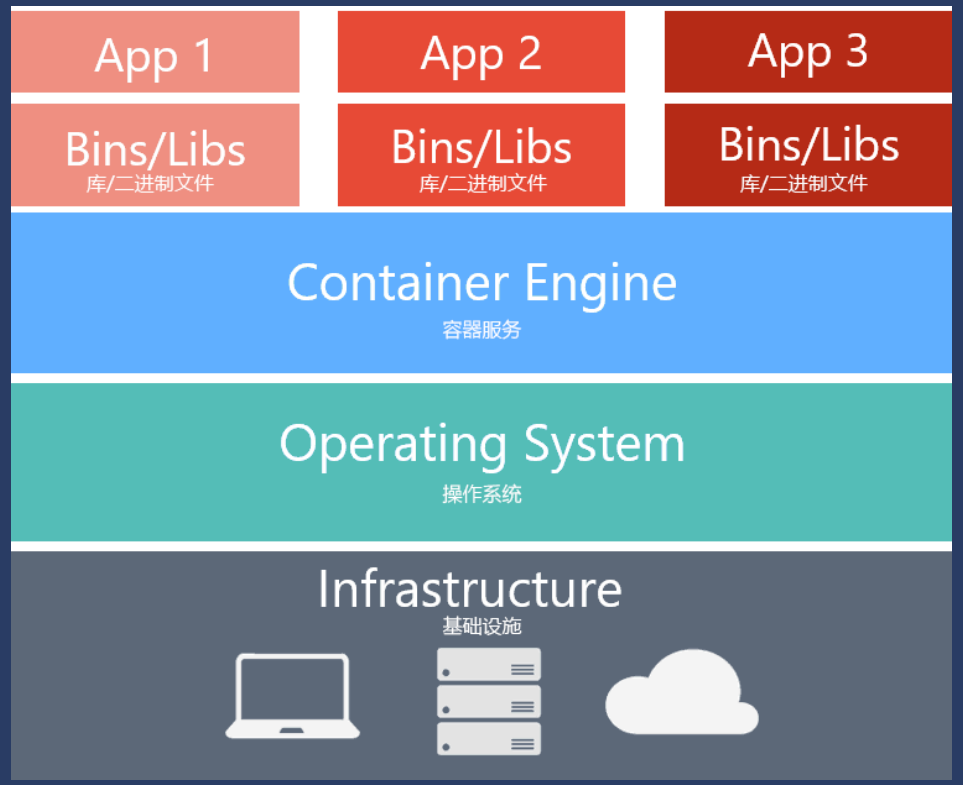
为什么要用 Docker16由于 Docker 确保了执行环境的一致性，使得应用的迁移更加容易。Docker 可以在很多平台上运行，无论是物理机、虚拟机、公有云、私有云，甚至是笔记本，其运行结果是一致的。因此用户可以很轻易的将在一个平台上运行的应用，迁移到另一个平台上，而不用担心运行环境的变化导致应用无法正常运行的情况。

△ 更轻松的维护和扩展

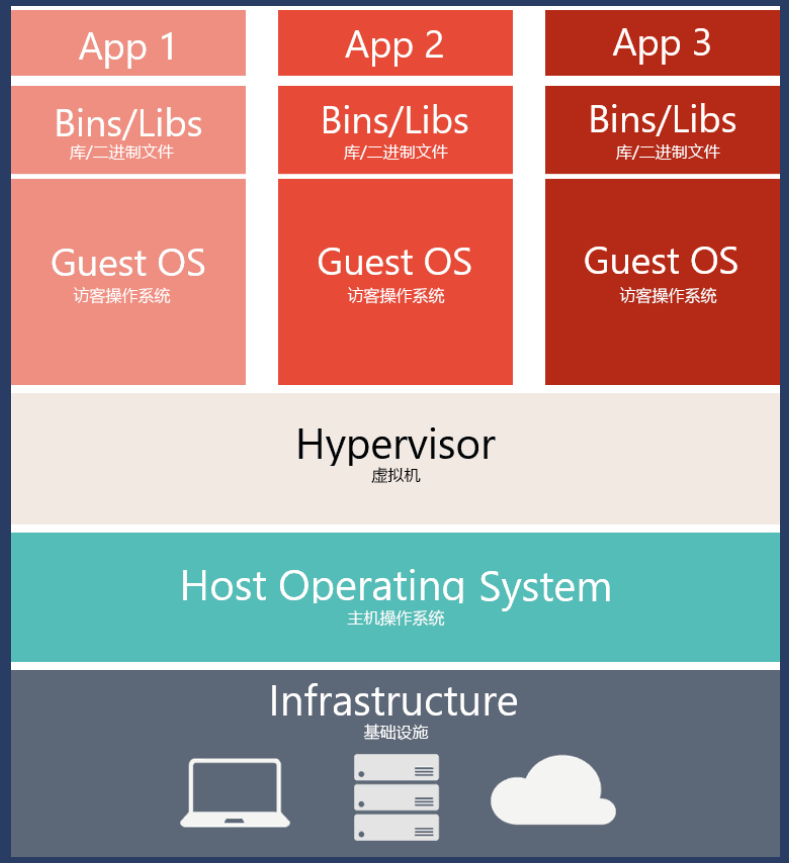
Docker 使用的分层存储以及镜像的技术，使得应用重复部分的复用更为容易，也使得应用的维护更新更加简单，基于基础镜像进一步扩展镜像也变得非常简单。此外，Docker 团队同各个开源项目团队一起维护了一大批高质量的 官方镜像，既可以直接在生产环境使用，又可以作为基础进一步定制，大大的降低了应用服务的镜像制作成本。

传统虚拟机与Docker容器的比较

Windows Server Containers – 通过进程和名称空间隔离技术提供应用程序隔离。Windows Server容器与容器主机以及主机上运行的所有容器共享一个内核。

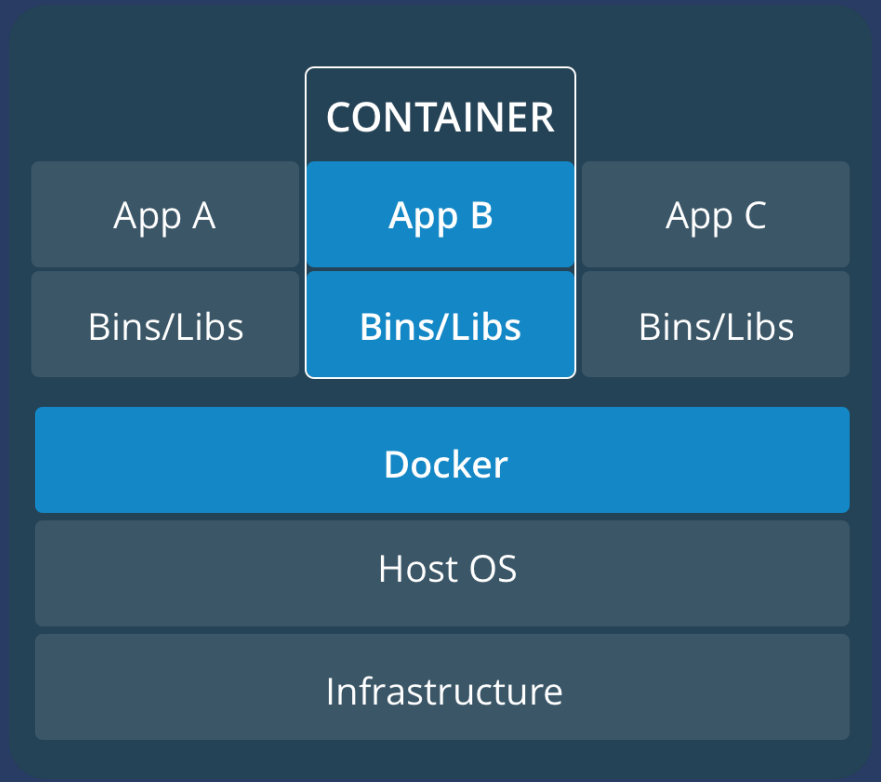


Hyper-V Containers – 通过在高度优化的虚拟机中运行每个容器来扩展Windows Server Containers提供的隔离。在此配置中，容器主机的内核不与Hyper-V容器共享，从而提供更好的隔离。

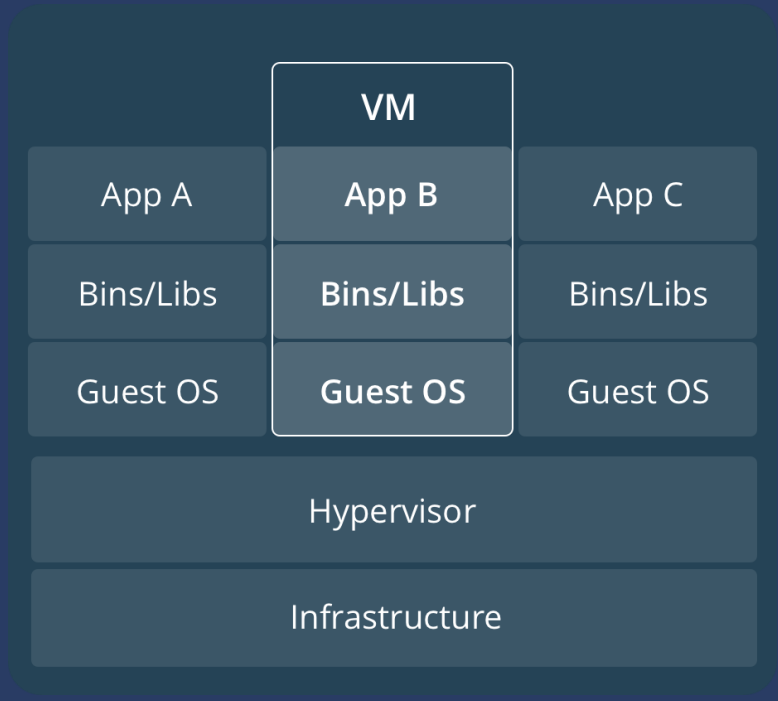


传统虚拟机与Docker容器的比较

Windows Server Containers - 通过进程和名称空间隔离技术提供应用程序隔离。Windows Server容器与容器主机以及主机上运行的所有容器共享一个内核。



Hyper-V Containers - 通过在高度优化的虚拟机中运行每个容器来扩展Windows Server Containers提供的隔离。在此配置中，容器主机的内核不与Hyper-V容器共享，从而提供更好的隔离。



传统虚拟机与Docker容器的比较

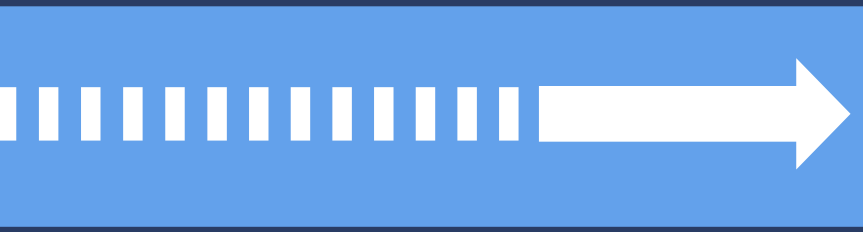
虚拟机

- 虚拟机包括了应用程序，所需的库/二进制文件和一个完整的操作系统。它的虚拟化比集装箱要重要得多。

容器化技术

- 容器也同样保护了应用程序，以及其所有的依赖项，但是他与其他容器可以共享操作系统的内核，在主机操作系统的用户空间中作为独立的进程运行。ps:除了在“Hyper-V容器”中，每个容器在每个容器的特定虚拟机内运行
- 从程序体系结构的角度来看，每个Docker容器都是单个进程，可以是整个应用程序（单一应用程序）或单个服务或微服务。当您的应用程序或服务进程在Docker容器内运行时，您获得的好处是它还包含了所有的依赖关系，因此，在任何支持Docker的环境中进行部署通常都是可以保证的。
- 由于Docker容器是在相同的共享OS内核上运行的沙箱，因此它提供了非常重要的好处。他们很容易部署，启动速度很快。作为在同一个内核上运行的一个副作用，你得到的隔离度比虚拟机少，而且使用的资源也少得多。正因为如此，容器的启动速度很快。

Docker也是打包应用程序/服务的一种方式，并以可靠和可重现的方式将其推出。所以，可以说Docker是一门技术，也是一门哲学和一个过程。



Chapter 3

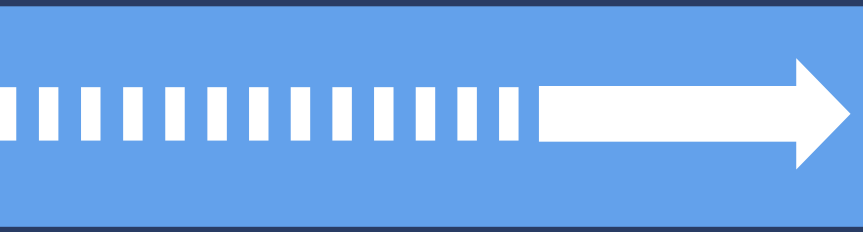
关于Docker的概念

基本概念

- Docker镜像： Docker镜像是容器的基础。图像是根文件系统更改的有序集合，以及用于容器运行时的相应执行参数。映像通常包含堆叠在一起的分层文件系统的联合。图像没有状态，并且在部署到各种环境中时永远不会改变
- 容器(Container)：容器是Docker镜像的运行时实例。一个Docker容器包含：一个Docker镜像，一个执行环境和一套标准的指令。缩放服务时，您将从同一图像中实例化多个容器。或者，在批处理作业中，从同一图像中实例化多个容器，将不同的参数传递给每个实例。一个容器“包含”一些单一的过程，比如服务或者web应用。这是1：1的关系。
- 储存库（Repository）：一个相关图像的集合，通过一个标签来区分，该标签将区分特定图像的历史版本。某些回购包含特定图像的多种版本，如SDK，运行时/胖，瘦标签。随着Windows容器变得越来越流行，单个回购站可以包含平台变体，例如Linux和Windows映像。

基本概念

- 标签(TAG)：标签是一个应用于存储库中的Docker镜像的标签。标签是一个存储库中的各种图像是如何相互区分的。它们通常用于区分同一图像的多个版本。
- Dockerfile： Dockerfile是一个文本文档，包含构建Docker镜像的指令。
- 构建(Build)：构建是使用Dockerfile构建Docker镜像的过程。构建使用Dockerfile和“上下文”。上下文是构建映像的目录中的一组文件。构建可以通过像“docker build”或“docker-compose”这样的命令来完成，其中包含了诸如图像名称和标记之类的附加信息。

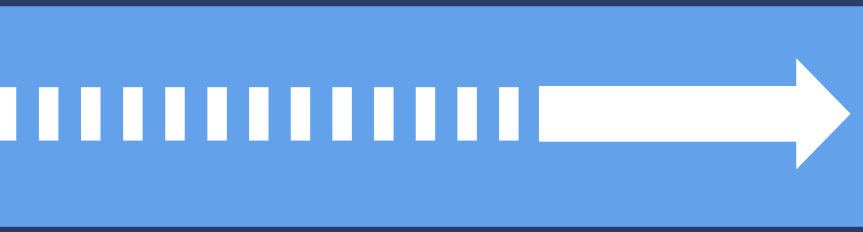


Chapter 4

Docker EE & Docker CE

Docker EE & Docker CE

- Docker EE
- Docker EE由公司支持，可在经过认证的操作系统和云提供商中使用，并可运行来自Docker Store的、经过认证的容器和插件。
- Docker CE
- Docker CE是免费的Docker产品的新名称，Docker CE包含了完整的Docker平台，非常适合开发人员和运维团队构建容器APP。事实上，Docker CE 17.03，可理解为Docker 1.13.1的Bug修复版本。因此，从Docker 1.13升级到Docker CE 17.03风险相对是较小的。
- 大家可前往Docker的RELEASE log查看详情 https://github.com/docker/docker/releases 。
- Docker公司认为，Docker CE和EE版本的推出为Docker的生命周期、可维护性以及可升级性带来了巨大的改进。



Chapter 5

Let's Do It

环境准备

- Xshell Xftp（安全终端模拟软件）
- Ubuntu 16.04 64位 服务器一台（可自建虚拟机）
- Visual Studio Code
- 下载地址：http://onpmq0amb.bkt.clouddn.com/xshell+xftp.rar

Ubuntu 安装 Docker CE

- 安装地址:
- https://github.com/52ABP/52ABP.Docker_Course/blob/master/src/1.0%E5%AE%89%E8%A3%85Docker.Md