01 开篇词：欢迎来到 Docker 的世界

更新时间：2020-07-30 10:08:05

> 老骥伏枥，志在千里； 烈士暮年，壮心不已。 ——曹操

你好，我是 legendtkl，目前就职于某 BAT 大厂，任职技术专家，具有丰富的后端开发经验，包括 Docker、Kubernetes、Web 开发和大数据等。

首先感谢大家来学习这门专栏，在本专栏中，我将会为大家深入浅出地剖析 Docker 技术及其最佳实践并分享**我在使用 Docker 的时候所踩的一些坑**。

众所周知，Docker 是使用 Go 语言编写的，我从毕业之初就开始关注和使用 Go 语言，那时候 Go 语言的明星项目还比较少，Docker 算是其中一个比较耀眼的项目。

于是，我抱着学习的态度，把 Docker 的源代码撸了一遍，也算是把 Docker 技术入了门。再到后来我注意到 Kubernetes 日益火热起来，于是我又开始投入精力学习 Kubernetes。

这样的学习一直持续到了 2016 年，彼时，Docker 作为云计算的基础设施逐渐进入大众视野，随后被誉为云上操作系统的 Kubernetes 也开始逐渐崭露头角。

作为国内云计算市场的领头羊，我所在的部门也开始大量引入 Docker 和 Kubernetes 技术栈，由于我之前一直在储备这方面的技术知识，正好匹配部门当前的需求，于是开始在部门内脱颖而出。同时也印证了那句：“**机会从来都是留给有准备的人**”。

另外**我在部门容器化转型过程中通过不断的学习和总结积累了非常宝贵的一手经验，希望可以通过本专栏介绍给大家。**

1. 为什么要学习 Docker

Docker 已经变成基础设施

目前的技术水平已经真正的进入了云计算时代，各大基础设施纷纷上云。阿里已经在去年双十一将所有业务都迁移到了云上。而云的基础设施就是 Docker，准确的说是容器，而 Docker 正是当下最主流的容器技术。

之所以说 Docker 是基础设施，是因为 **Docker 的资源限制和隔离性是云上必不可少的特性**。相比于之前每个部门的应用申请一堆物理机来部署自己的应用进程，使用云计算的方式来统一管理公司所有的应用使得资源的使用率更加的高。

那么这么多的应用部署到一起就需要解决三个主要问题：**应用隔离、应用部署和资源限制**。

应用隔离

很容易理解，比如**不同的应用依赖了冲突的基础软件包**，如果都直接部署在操作系统环境下，必然会引起冲突。这个时候就需要应用之间互相隔离。Docker 使用操作系统底层的 Namespace 技术来做隔离是一种主流的技术方案。

应用部署

在容器技术诞生之前，所有的应用都是**直接部署在操作系统上面的，彼此之间共享底层的操作系统资源，比如内存、CPU、磁盘等**。

打个比方，如果我们要将应用 A 和应用 B 部署到同一台机器上，那么这两个应用需要的环境信息，我们都需要满足。如果应用 A 和 B 的环境依赖之间存在冲突，或者说不兼容，那么管理起来就会非常的困难。

而这个问题，我们通过容器的镜像技术却可以非常简单地解决掉。除此之后，Docker 技术还有很多相比于传统技术更加优势的地方，这也 Docker 技术能够迅猛发展起来的根本原因。

资源限制

不同的应用共享集群不可避免的涉及到资源使用限制的问题。比如一个公司同时有多个业务部门使用一个统一的集群，在统计研发成本时，需要统计各个部门的资源使用情况然后做划分。这样我们就可以根据资源使用限制来划分，当某些应用资源使用超限就杀掉应用。Docker 使用内核提供的 Cgroup 技术来做资源限制正好可以应用到这个场景。

Docker 技术学习方法

这几年为团队面试过很多同学，发现很多候选人对于 Docker 的技术理解都**浮于表面**，知道 Docker 如何使用，但是一旦**涉及到底层技术原理就不清楚了**。

有一些同学可能会有疑问，工作中更多的还是使用，**真的有必要了解底层原理吗**？日常使用可能没有问题，但是一旦出现疑难杂症，如果不了解 Docker 的底层知识，Docker 对于我们就是一个黑盒，对于排查问题是很不利的。

造成上面这种现象的主要原因是大部分的候选人都没有系统地进行学习，而是在日常工作中遇到技术问题去专项解决，或者网上搜索一些各种博主的文章。而网上的文章大部分都是转载来转载去，有一些知识点还是错的，我之前有一篇文章就被反复转载，而里面一个知识点是老版本中的实现，新版本中的实现早就变了。

而这篇专栏正是要解决上面这个问题的，**为大家提供一个系统地，全面地学习教程**。

2. Kuberneters

前面介绍了 Docker 技术的一些理论知识和最佳实践，但是正如前面所说，Docker 技术从来都不是一个孤立的技术。Docker 更多的还是作为一种技术基石，基于 Docker 我们还需要做一些包括容器编排管理等工作。

说到容器编排和集群管理系统，从早期的 docker-compose 到 swarm，可以说是诸侯林立。但是自从 Kubernetes 出现后慢慢的出现了大一统的趋势，凭借着 Google 内部早期积累的经验和优秀的云原生设计理念，Kubernetes 可以说是一骑绝尘，目前已经全面占领了云端统一管理的地位。

而在本专栏中，我也给大家设计了“**Kubernetes 的核心技术精讲和最佳实践**”环节，帮助大家快速掌握 Kubernetes 这一独领风骚的容器编排和集群管理系统。

3. 你可以收获什么

通过本课程你可以学习到 Docker 和 Kubernetes 的基础知识和核心原理以及最佳实践。

最重要的是**授人以渔**。本专栏的文章组织形式都是一种学习思路的再现，比如提出问题，尝试解决问题，理解问题背后的原理。

我觉得这也是一种比较好的学习方式，**问题驱动式的学习方式能够帮助我们不仅知其然、而且知其所以然。更重要的是通过这种方式习得的知识要比填鸭式的照本宣科显得更加的牢靠**。

4. 课程组织

为了能让你更快的掌握 Docker 和 Kubernetes 的基础知识和核心原理，同时也照顾一下那些没有基础的同学，我将专栏内容作了如下安排“

本专栏围绕 **Docker 和 Kuberneters** 核心技术展开，共划分为 **5** 部分 **47** 个小节。

第一章： Docker 基础

本部分内容会介绍 Docker 的一些背景和基础知识，简单概括，包括如下部分：

- Docker 容器的发展之路。介绍 Docker 技术如何从一个不起眼的技术逐步发展壮大并最终引领了一场 PaaS 革命的；
- Docker 安装与运行。介绍 Docker 的基本操作，包括不同平台的安装和运行；
- Docker 技术概览。这篇文章会介绍 Docker 的基本使用情况，让大家对 Docker 技术有一个总体的概览；
- Docker 镜像技术。介绍 Docker 的镜像技术。Docker 的镜像技术是 Docker 技术中非常重要的一环，Docker 正是靠着镜像技术改变了当年如火如荼的 PaaS 格局；
- 其他：包括 Docker 常用的镜像等；
- 动手实践：在这个环节，我们将动手实践构建出第一个属于我们自己的 Docker 应用。有句话说的好，“纸上得来终觉浅，绝知此事要躬行”。只有真正动手实践才能更加深刻的理解内部原理。

第二章： Docker 核心技术

在第一部分，我们主要介绍一些 Docker 的基础技术，在这个部分我们将带领大家更加深入地理解 Docker 的核心技术。包括如下部分：

- Docker 隔离技术的本质 Namespace 深入解析：从内核层面介绍 Docker 的隔离的本质 Namespace 技术；
- Docker 资源限制技术的幕后主使 Cgroup 剖析：从操作系统角度介绍 Cgroup 技术；
- Docker 镜像深入理解，主要介绍 Docker 镜像的组织形式，包括联合文件系统挂载等。
- Docker 的本质，Docker 的本质其实就是进程；
- Docker 镜像构建，介绍 Dockerfile 的使用；
- Docker 网络相关：介绍 Docker 的网络的基本原理和几种模式，以及 Docker 中 link 的实现原理；
- Docker 数据存储相关：主要包括 Docker 的数据卷和数据共享相关。

第三章： Docker 最佳实践

前面两部分介绍完了理论部分，第三章主要介绍 Docker 最佳实践：

- Dockerfile 最佳编程实践：详细介绍 Dockerfile 编写中的各种指令的使用场景和最佳实践；
- 如何构建最小镜像最佳实践：镜像构建作为应用打包部署中非常关键的一环，这篇文章将介绍如何使得构建出来的镜像尽量的小；
- 其他 Docker 使用中的最佳实践，比如如何使用 tag 等；
- 我们容器化的时候为什么要保持 Docker 是个单进程模型：相信大家都听过这句话 “Docker 是个单进程模型”，那么为什么这么说呢？本篇文章将从操作系统层面进行分析；
- 容器设计模式：介绍当前基于容器构建分布式应用程序几种典型的设计模式；
- Docker 监控方案最佳实践：这篇文章将介绍 Docker 的监控最佳实践；
- 从 0 到 1 构建一个分布式高可用的 Web 应用。

第四章： 云原生容器技术Kubernetes

前面介绍了 Docker 技术的一些理论知识和最佳实践，但是正如前面所说，Docker 技术从来都不是一个孤立的技术。Docker 更多的还是作为一种技术基石，基于 Docker 我们还需要做一些包括容器编排管理等工作。

说到容器编排和集群管理系统，从早期的 docker-compose 到 swarm，再到最近两年的 Kubernetes，终于出现了大一统的趋势，也就是 Kubernetes。Kubernetes 凭借着 Google 内部早期积累的经验和优秀的云原生设计理念，一骑绝尘。在 2018 年全面推广开来，目前已经全面占领了云端统一管理的地位。我们这一章就是要介绍 Kubernetes 相关的技术知识，包括：

- 云原生和 Kubernetes 简介；
- Kubernetes 集群资源隔离介绍：Namespace；
- Kubernetes 核心设计 Pod 的解析；
- 配置管理 ConfigMap 和 Secret；
- 容器化守护进程 DaemonSet；
- Kubernetes 的控制器模式介绍；
- Kubernetes 的典型控制器 ReplicaSet、ReplicaController 和 Deployment 介绍；
- 批处理：Job 和 CronJob 介绍；
- 实践部分：通过 Kubernetes 来构建我们的容器化应用。

第五章： 云原生监控方案

目前 Prometheus 已经成为云原生监控方案的事实标准，最后一章我们将介绍 Prometheus 的一些知识，包括：

- Prometheus 简介：包括定位等；
- Prometheus 架构解析；
- 通过 Prometheus 监控容器。

最后，大家在学习过程中**有任何问题都可以在专栏的评论区进行留言**，我会统一进行回复。

}

02 成王败寇：容器的发展之路