[PACKT] PUBLISHING

DevOps with Kubernetes

Accelerating software delivery with container orchestrators

/云计算技术实践系列丛书/

基于Kubernetes的DevOps实践

容器加速软件交付

[日] Hideto Saito [加] Hui-Chuan Chloe Lee Cheng-Yang Wu 著◎

史 天 肖 力 刘志红 译◎

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

云计算技术实践系列丛书

DevOps with Kubernetes

Accelerating software delivery with container orchestrators

基于Kubernetes的DevOps实践

容器加速软件交付

[日] Hideto Saito [加] Hui-Chuan Chloe Lee Cheng-Yang Wu 著◎

史 天 肖 力 刘志红 译◎

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

容器化被认为是实现 DevOps 的最佳方式。谷歌开发了 Kubernetes，它有效地协调容器，被认为是容器编排的领跑者。Kubernetes 是一个编排器，可以在服务集群上创建和管理容器。本书将指导管理 Kubernetes 集群，然后学习如何在 DevOps 中监控、记录日志和持续部署。

本书将介绍 DevOps 和容器的基本概念，部署和将应用程序容器化，并介绍 Kubernetes 中的网络和存储。然后，使用先进的 DevOps 技能，如通过基于属性的访问控制和基于角色的访问控制，监控、记录和连续引入 Kubernetes 资源的权限控制。本书还涵盖部署和管理 Amazon Web Services 和 Google Cloud Platform 相关内容。最后，讨论了其他编排框架，如 Docker Swarm 模式、Amazon ECS 和 Apache Mesos。

版权贸易合同登记号　图字：01-2018-3990

图书在版编目（CIP）数据

基于 Kubernetes 的 DevOps 实践：容器加速软件交付 /（日）希迪托・佐藤 (Hideto Saito) 等著；史天，肖力，刘志红译. —北京：电子工业出版社，2019.7
（云计算技术实践系列丛书）

书名原文：DevOps with Kubernetes: Accelerating software delivery with container orchestrators

ISBN 978-7-121-36570-6

Ⅰ. ①基… Ⅱ. ①希… ②史… ③肖… ④刘… Ⅲ.①云计算—研究 Ⅳ. ①TP393.027

中国版本图书馆 CIP 数据核字（2019）第 092767 号

责任编辑：刘志红　　特约编辑：李　姣
印　　刷：天津嘉恒印务有限公司
装　　订：天津嘉恒印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×980　1/16　印张：19.5　字数：437 千字
版　　次：2019 年 7 月第 1 版
印　　次：2019 年 7 月第 1 次印刷
定　　价：108.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）88254479，lzhmails@phei.com.cn。

译者序

随着技术的不断进步，软件交付方式经过三个阶段。第一阶段是以光盘为代表的实物交付方式，第二阶段是基于互联网的电子交付方式，第三阶段是以云计算为基础的在线交付。软件交付的周期也越来越短，同时应用程序的规模越来越大，随着 DevOps 解决方案应运而生，DevOps 以持续快速发布为目标，致力于通过发布流水线支持持续的软件构建和交付。

容器的出现，巧妙地解决了应用程序隔离的问题，Kubernetes 进一步释放了容器的能力。DevOps 的理念已经深入人心，Kubernetes 更是热度不减，并且走向成熟。容器的出现加速了 DevOps 的落地，Kubernetes 让容器管理更便捷。随着技术的发展，在 Kubernetes 之上构建 DevOps 无疑是正确甚至是最佳选择之一。

本书主要介绍在 Kubernetes 之上构建 DevOps 的最佳实践，先介绍了 DevOps 和 Kubernetes 的基本知识，然后介绍了在 Kubernetes 之上构建 DevOps 常见场景的最佳实践，包括存储、网络和安全、监控和日志记录、持续交付、集群管理，以及 AWS 和 GCP 上的 Kubernetes 使用。通过阅读本书，有助于提高软件交付自动化，缩短软件交付时间。

本书注重实践，是不可多得的在 Kubernetes 之上构建 DevOps 的参考手册。相信通过阅读本书，可以带给读者不少启发，节省不少实际中摸索的时间。另外，由于译者水平有限，虽然经过了反复核对，仍然难免有错误的地方，恳请各位读者指正。

关于作者

Hideto Saito 在计算机行业拥有约 20 年的经验。1998 年，在日本 SUN Microsystems 工作期间，他对 Solaris OS、OPENSTEP 和 Sun Ultra Enterprise 10000（即 StarFire）有深入研究，是 UNIX 和 MacOS X 操作系统拥护者。

2006 年，他搬到南加州，作为一名软件工程师开发在 Linux 和 MacOS X 上运行的产品和服务。以快速地 Objective-C 编码而闻名。

Hui-Chuan Chloe Lee 是 DevOps 拥护者和软件开发者。她在软件行业从业已超过 5 年。作为技术爱好者，她喜欢学习新技术，这让她的生活更快乐和充实。她喜欢阅读、旅行，与爱人共度时光。

Cheng-Yang Wu 自从获得中国台湾大学计算机科学硕士学位以来，一直致力于解决基础设施和系统可靠性问题。他的懒惰促使他掌握 DevOps 技能，最大限度地提高了工作效率，然后挤出时间以编写代码为乐趣。他喜欢烹饪，因为就像使用软件一样——完美的菜肴总是需要精心的调制。

目　录

1 DevOps 简介

软件交付周期越来越短，与此同时，应用程序的规模越来越大，软件开发工程师和运维工程师面临寻找解决方案的压力。然后，一个叫 DevOps 的新角色诞生了，它致力于支持软件的持续构建和交付。

本章包含以下内容：

- 软件交付方法是如何演进的？
- 什么是微服务？为什么要采取这个架构？
- DevOps 如何支持应用程序的构建，并将其交付给用户？

软件交付的挑战

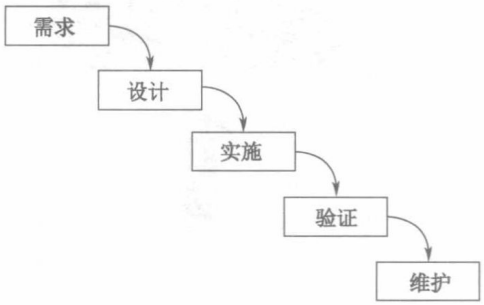
如何构建计算机应用程序并且交付给用户，一直以来都是被不断讨论的话题，并且随着时间的推移不断发展。它和软件开发周期（Software Development Life Cycle，SDLC）息息相关，这里将介绍几种常见类型的流程、方法和历史。在本章中，我们将介绍这个发展过程。

瀑布模型和实物交付

退回到 20 世纪 90 年代，软件交付采用**实物**方式，比如软盘、硬盘或者光盘。因此，SDLC 是一个非常长的周期，因为整个过程并不简单。

在这个阶段，主要的软件开发模型是**瀑布模型**，该模型包含需求、设计、实施、验证、维护阶段，如下图所示。

这种情况下，我们不能从一个阶段返回到上一个阶段。例如，在开始或者结束**实施**阶段后，返回**设计**阶段（例如查找技术可扩展性问题）是不可接受的。这是因为它



会影响到整体进度和成本。项目趋于继续向前并完成发布，然后进入下一个发布周期，包括新的设计。

这种方式和实物交付完美匹配，因为它需要与制作与交付软盘/CD-ROM 给用户的物流管理协调一致。瀑布模型和实物交付通常需要很长时间，一般是一年到几年。

敏捷模型和电子交付

几年之后，互联网被广泛接受，随之软件交付从实物转变成**电子交付**，比如在线下载。因此，许多软件公司（也称为 dot-com 网络公司）试图找出如何缩短 SDLC 流程，以交付能够击败竞争对手的软件。

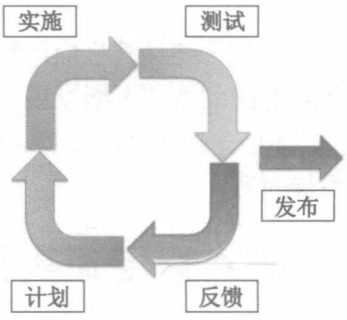
许多开发者开始采用新的方法，比如增量、迭代和**敏捷**模型，以便更快地交付给用户。即使发现了新的错误，现在也可以更容易地通过电子交付给用户更新和打补丁。微软也从 Windows 98 开始引入 Windows update 更新。

在这种情况下，软件开发者只需要写很少的逻辑和模块，而不是一次性编写整个应用程序。然后交付给质量保证工程师（QA），接着开发者可以继续添加新模块，最后再将其交付给质量保证工程师（QA）。

当所需的模块或功能准备就绪时，它将会被发布，如下图所示。

该模型使 SDLC 循环和软件交付更快，并且在此过程中也易于调整，因为循环一般持续几周到几个月，这个周期足以进行快速调整。

尽管敏捷模型目前受到大多数人的喜爱，但在当时应用软件交付意味着软件二进制文件，例如 EXE 程序，被设计用于在客户的 PC 上安装和运行。另一方面，基础设施（例如服务器和网络）是静态的，并且需要事先设置好。因此，SDLC 的范围不倾向于涉及这些基础设施。



云端的软件交付

几年之后，智能手机（例如 iPhone）和无线技术（例如 Wi-Fi 和 4G 网络）被广泛接受，软件应用程序也从二进制转变为在线服务，网络浏览器是应用程序软件的交互界面，客户不再需要安装应用程序。另外一方面，基础设施转变为动态的，因为应用程序需求不断变化，并且容量也需要持续增长。

虚拟化技术和**软件定义网络（SDN）**使服务器更加动态化。现在，像 **Amazon Web**

Services（AWS）和 **Google Cloud Platform（GCP）**这样的云服务，可以轻松创建和管理动态基础设施。

现在，基础设施被纳入软件开发交付周期的范围，并且是重要的组件之一，因为应用程序被安装并且运行在服务器上，而不再是客户的 PC 机上。因此，软件和服务交付周期缩短为几天到几周。

持续集成

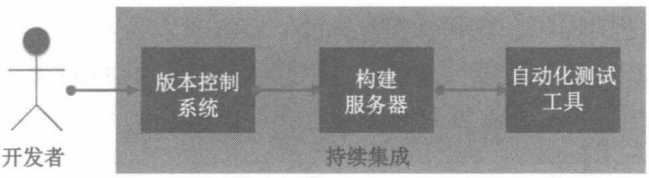
如上文所述，软件交付环境不断变化，交付周期也越来越短。为了实现更高质量的快速交付，开发者和 QA 开始采用一些自动化技术。其中一项流行的自动化技术就是**持续集成（Continuous Integration，CI）**，CI 包含一系列工具的组合，例如**版本控制系统（Version Control Systems，VCS）**、**构建服务器和自动化测试工具**。

VCS 帮助开发者在中心服务器上维护程序源代码。它阻止开发者的代码被覆盖或者和其他开发者的代码冲突，并且可以保留历史记录。因此，它可以更容易地保持源代码的一致性，直到下一个交付周期。

与 VCS 相同，持续集成中有一个集中的构建服务器，它连接到 VCS，定期或者在开发者更新代码到 VCS 时自动检索源代码，然后触发新构建。如果构建失败，它会及时通知开发者。因此，当有人将损坏的代码提交到 VCS 时，构建服务器可以帮助开发者及时调整。

自动化测试工具也与构建服务器集成。当构建成功后，它会调用单元测试程序，然后将结果通知给开发者和 QA。这有助于识别何时有人编写错误的代码，并存储到 VCS。

整个 CI 的流程如下图所示：



CI 不仅可以帮助开发者和 QA 提升代码质量，并且缩短了程序归档或者模块打包周期。在电子交付的时代，CI 足够应付。然而，交付到用户即意味着需要将应用程序部署到服务器。

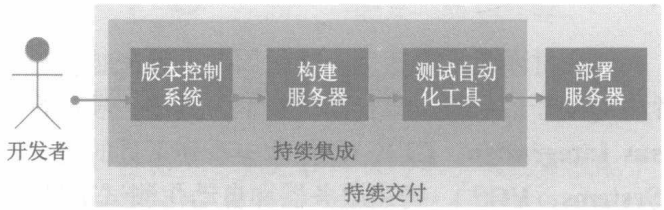
持续交付

CI 加上自动化部署是服务器上的应用程序向用户提供服务的理想过程。但是，还有一些技术挑战需要解决。如何将软件交付给服务器？如何优雅地关闭已有的应用程序？

如何替换或者回滚应用程序？如果系统库也要更新，如何升级或者替换？如果需要，如何修改操作系统中的用户和组设置？

因为基础设施包含服务器和网络，所以这些都取决于开发、QA、预发布、生产等环境。每个环境都有不同的服务器配置和 IP 地址。

持续交付（Continuous Delivery，CD）是一种可以实现的最佳实践，它是 CI 工具、配置管理工具和编排工具的组合：



配置管理

配置管理工具帮助配置操作系统，包括用户、用户组和系统库，并且可以管理多台服务器，以便我们更换服务器时与所需的状态或者配置保持一致。

它不是脚本语言，因为脚本语言是逐行执行的命令。如果我们两次运行脚本，则可能导致一些错误，举个例子，如尝试两次创建同一个用户。另外一方面，配置管理会检查状态，所以，如果用户已经创建，配置管理工具不会执行任何操作。但是，如果我们意外或者故意删除用户，配置管理工具将再次创建用户。

它还支持将应用程序部署或安装到服务器中。因为如果你告诉配置管理工具下载应用程序，然后设置并运行应用程序，它就会尝试这样做。

此外，如果你告诉配置管理工具关闭应用程序，然后下载并替换为新的可用的软件包，然后重新启动应用程序，它将保持新版本。

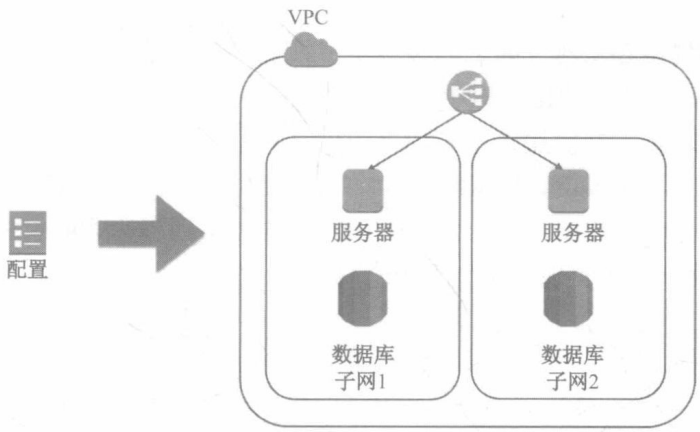
当然，一些用户希望仅在需要时更新应用程序，例如蓝绿部署。配置管理工具将允许你手工触发执行。

蓝绿部署是一种准备两组应用程序堆栈的技术，只有一个环境（例如蓝色）为生产服务。当你需要部署新版本的应用程序时，部署到另一个环境（例如绿色），然后执行最终测试。如果一切工作正常，更改负载均衡器或路由器设置，将网络流量从蓝色切换为绿色。然后绿色变为生产服务，而蓝色变为休眠状态，并等待下一个版本部署。

基础设施即代码

配置管理工具不仅支持操作系统或者虚拟机，还支持云基础设施。如果你需要在云端创建并且配置网络、存储和虚拟机，则需要一些云端操作。

配置管理工具可以通过配置文件自动化设置云基础设施，如下图所示：



配置管理相对于**标准操作过程（Standard Operation Procedure，SOP）**有一些优势。例如，使用 **VCS**（比如 Git）维护配置文件，可以跟踪环境设置的更改历史记录。

复制环境也变得容易。例如，你需要在云上部署一套额外的环境。如果你遵循传统方法（即读取 SOP 文档来操作云环境），它总是存在潜在的人为和操作失误。另一方面，我们可以通过配置管理工具执行操作，快速自动地在云上创建环境。

基础设施即代码可能包含也可能不包含在持续交付（CD）过程中，因为基础设施更换或者更新成本高于仅替换服务器上的应用程序二进制文件。

编排

编排工具也被归为配置管理工具的一种。但是，在配置和分配云资源时，它更加智能和动态。例如，编排工具可以管理多个服务器和网络资源，当管理员想要增加应用程序实例时，编排工具可以确定可用的服务器，然后自动部署和配置应用程序和网络。

虽然编排工具超出了 SDLC 范围，但它在需要扩展应用程序和重构基础设施资源时有助于持续交付。

总而言之，SDLC 已经发展为通过若干流程、工具和方法以实现快速交付。最终，软件（服务）交付只需要几小时。与此同时，软件架构和设计也在不断地发展，以实现大型应用程序。

微服务趋势

基于目标环境和应用程序的大小，软件架构和设计也在不断发展。