ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.14, No.26, September . 2018
E-mail:kfyj@dnzs.net.cn
http://www.dnzs.net.cn
Tel:+86-551-65690963 65690964

基于Docker平台的DevOps运维系统的研究与改进

凌 云

(安徽电信工程有限责任公司,安徽 合肥 230069)

摘要:我国信息产业正处于飞速发展之中,各类型的应用软件与信息系统的生存周期也日益缩减。因此,针对用户群的实际需求,及时研发对应的软件系统并将其快速部署,已经成为了IT企业竞争的主要目标。DevOps作为一种新兴方法,能够显著提高开发与运维工作之间的协同程度,从而减轻了系统运行之后的运维压力。该文提出将开源平台Docker引入到DevOps系统的管理工作当中,利用其镜像共享工程环境等功能,进一步改进了DevOps系统的可靠性与稳定性。

关键词:信息系统;快速部署;系统运维;DevOps;Docker

中图分类号:TP311 **文献标识码:A** **文章编号:1009-3044(2018)26-0209-03**

DOI:10.14004/j.cnki.ckt.2018.3221

1 概述

我国目前已经进入到了信息化时代,其最大的特征就是各类信息化应用服务技术呈现出爆炸式发展趋势,这些功能各异,类型多样的服务软件同样也导致了IT企业在运维工作方面的巨大压力。IT企业迫切希望得到一款通用性能好、可靠性强且能够统一支持开发与运维管理工作的平台,从而实现对各种软件系统的一站式集成与交付,自动化配置与部署、实时的监测与管控、自动化伸缩及恢复功能,并最终显著的优化了整个运维工作的效率与成本。在这一需求背景下,DevOps系统应运而生,并得到了快速的推广。

2 Docker与DevOps技术概述

2.1 Docker技术特征

Docker是一个分布式应用构建、迁移和运行的开放平台,由于其操作的便捷性和快速部署的高效性,一经推出就受到了广泛的好评,并在系统发布和运维领域得到了有力的推广。从技术方面分析,Docker容器引擎支持应用软件的开发团队将该应用与相关的依赖包打包成一个统一体,并将其置于一个可移植的容器中,随后就可以方便的将该容器中的内容发布至网络中的任何一台Linux机器中。另一方面,Docker还提供了注册服务器功能,使得用户可在该服务器上创建自己的镜像库来存储、管理和分享镜像,这也极大地提高了各类软件部署和运维工作的便捷程度。

在发布和管理应用服务的流程中,首先就需要将所有应用需首先部署至容器,通过编写相应的Dockerfile并执行其指令从而构建一个虚拟容器,实现快速部署。本文在构建综合性DevOps软件项目发布运维系统的过程中,选择在Jenkins平台上来实现软件项目的持续集成,并采用GitHub来完成对代码资源的统一托管。

2.2 DevOps简介

DevOps是一种全新的工作模式,围绕软件项目的开发、部署、测试与维护等一系列环节进行优化,有效地精简了软件运维工作的重复内容,并极大地提高了IT企业内各个部门之间的协同程度,实现了一站式的统一管理。目前,DevOps模式被广泛应用与各种应用服务的运维管理工作当中,其主要特点如下:

(1) 提高了运维工作对日益增多且微型化的系统调整需求的敏感性。

(2) 将开发人员与实际运行该软件项目的生产场合有机结合,提高了软件的可控性。

(3) 系统运维工作的重心从各个节点设施转移到了应用程序本身,提高了管理效率。

(4) 显著精简了软件研发部署工作的流程,控制了运维成本。

(5) 研发及部署流程实现了全部的自动化执行目标,减少了运维工作量。

2.3 Docker在DevOps系统中的作用

如上节所述,Docker最大的特点就是具有快速部署功能,以及可以通过镜像来共享工程环境,从而将软件项目的开发、测试和运维工作整合为一体,其在DevOps系统中起到的主要作用如下:

(1) 通过镜像在各个工作环节的复用,提高了软件项目从研发到运维工作的标准化程度,也提高了系统管理的稳定性与统一性。

(2) 可靠的解决了底层基础环境的异构问题。通过Docker的部署容器与创建镜像等方法,可以有效的兼容底层各种异构的基础环境。无论是何种节点设施或是云计算平台,均可通过采用Docker平台提供的各种方法来实现统一的调度与管理。

(3) 提高了镜像部署的灵活性。通过Dockerfile中的各类方法(包括分层机制)来构建标准化的镜像,不仅提高了镜像的可重用性,同时使得容器仓库可方便的将镜像迁移至任意环境,在部署时只需将镜像的只读属性转变为可运行即可。

收稿日期:2018-05-18
作者简介:凌云(1981—),男,安徽合肥人,工程师。

(4) 多种工具的集成部署与快速调用。可将多种工具软件封装进Docker容器并构建镜像，随后即可通过镜像的复用实现多种工具在任意环境下的集成部署。

通过以上分析可以看出，Docker提供了一套完整的虚拟化方式来快速构建开发环境，并以发送镜像的方式快速灵活的部署在各种开发、测试与运维场合，有效的精简了软件运营过程的中间环节，同时基于容器的封装及移植模式也很好的解决了不同平台之间的兼容性问题，提高了应用环境在各个阶段的统一性。

3 系统设计与实现

3.1 系统结构设计

本文设计的系统主要部件如图1所示，首先设计了使用便捷，亲和度高的操作管理界面，将原本需要手动输入Dockerfile的大量指令通过标准化的界面按键来自动写入；其次采用了etcd数据库集群、Docker Registry、Kubernetes集群等部件，既实现了对镜像的集群调度、存储与管理，又满足了对节点镜像的快速恢复和灵活的迁移等需求；Ngnix组件负责负载均衡的任务，可确保操作界面中各项功能的高效性与可靠性；Docker容器中集成的Flannel插件可满足不同节点上容器之间的通信需求，而这一过程也得到了网络与DNS组件的有力支持；Jekins主要监控软件项目的持续集成与测试过程；托管平台Github则负责对Git代码的托管与配置。

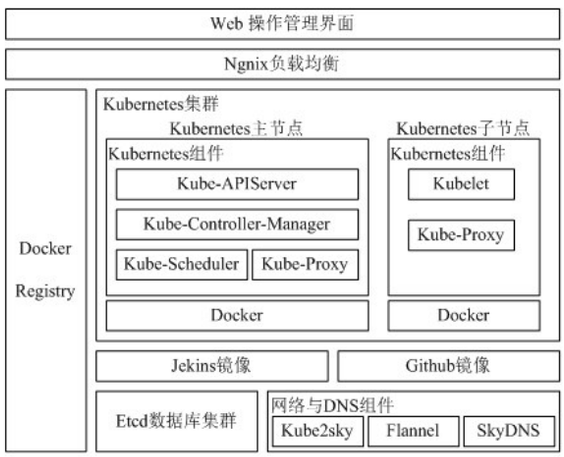

图1　系统结构示意图

本设计中各个核心模块承担的具体功能如下：

(1) Web操作管理界面提供了可视化的管理方案，简化了操作流程，同时也极大的降低了本系统的技术使用门槛。

(2) Ngnix组件提供了可靠的负载均衡服务，既可根据不同应用对应的数据格式进行分离，并针对性的分配给不同的节点予以处理，也可根据IP_Hash结果为各个用户主机分配固定的节点提供服务。

(3) Kubernetes集群：本部件分为两组子部件，分别为主节点master和子节点node以及对应的Docker构成。前者为控制节点，负责对各个容器进行监控与管理，后者为运行节点，负责各个容易的正常运行。主节点中通过Kube-Conntroller-Manager发送空盒子指令，子节点则根据指令从Docker Registry中调取相应的镜像并加以执行。

(4) 核心镜像：包括Jenkins镜像和Github镜像两类，前者启用Jenkins服务来构建源代码的编译环境，后者则启动Github组件实现对Git代码的可靠托管；除此之外，容器中还封装了所有服务所必须使用到的各种依赖资源，显著提高了镜像的构建速度。

(5) etcd数据库集群：用来向系统提供稳定快捷的分布式存储服务，其主要管理的内容分为两部分，其一为Kubernetes集群中的大量信息，其二为网络插件Flannel的配置信息。

(6) Docker Registry：该仓库用来存放系统构建的各个镜像，并根据用户提供的调取信息从仓库内选定符合要求的镜像提交系统。

(7) 网络与DNS管理：针对网络管理需求，本系统提出采用Flannel插件来简化配置与管理任务的设计思路，同时也很好的实现了不同容器之间的网络桥接目标，在运行过程中，所有安装了镜像的主机均会创建一个虚拟网卡，同时启动Flanneld服务，按照互联网TCP协议来建立不容主机之间的可靠连接，并完成报文段的封装与转发任务，从而以各个容器为节点，建立了有效的通信网络，在此网络中，任意一台节点主机中的Docker均被分配了全网唯一的虚拟IP地址；针对DNS解析需求，本系统采用Kube2sky插件对Kubernetes集群中的各个子节点以及其正在执行中的service进行监控，并将得到的信息写入etcd数据库，而SkyDNS则负责从etcd数据库读取这些记录并加以解析，最终得到不同IP地址所对应的域名。

3.2 系统运作流程分析

DevOps极大的提高了IT企业在系统及软件项目运维方面的工作效率，本文基于Docker技术设计得DevOps运行过程如图2所示。

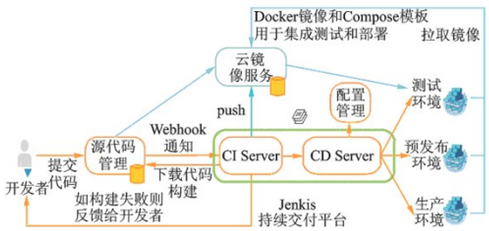

图2　基于Docker的DevOps系统运作流程

本系统具体的运作流程包括以下几个主要环节：

(1) 项目研发团队将需要托管的源代码以Git的文件格式提交给Github服务器；

(2) Github通过Webhook将代码库中新增加的源代码信息通知给Jenkins；

(3) Jenkins根据这些信息，可根据需求从Github调取某个项目的最新代码，以及对应的Dockerfile和相关的其他资源文件；

(4) Jenkins将得到的文件集中存储于云镜象服务中的Docker容器中，并根据Dockerfile中的指令集在子节点上构建镜像；

(5) 子节点将容器镜像实例化，同时启动系统测试功能对代码进行调试；

(6) 若代码运行正常，则Jekins将此镜像保存至Kocker Regaitry；

(7) 通过Kubernetes集群的持续发布功能，将软件项目发布至各种生产、测试环境。

图3给出了一个典型的使用Java语言开发，并以MySQL和Oracle为数据库构建软件项目的传统集成过程。

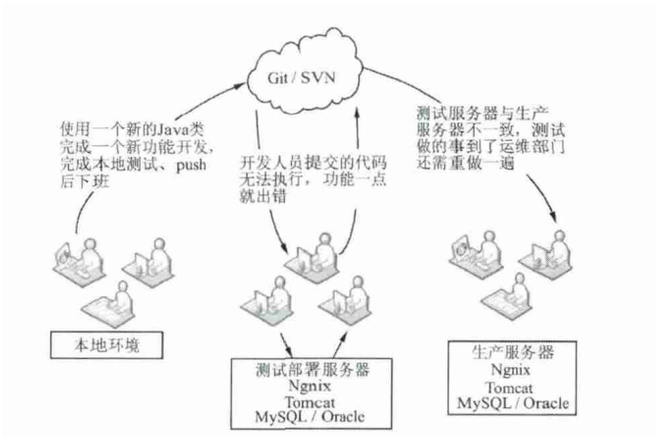

图3　传统模式下的软件项目集成

根据上图可以看出，在传统的软件项目集成过程中，由于开发环境、测试环境和运行环境之间存在多种类型的差异，使得项目在集成过程中需要被反复的说明和调整，出现了大量的重复劳动，而在本文提出的基于Docker的DevOps系统中，这一问题得到了很好的改善和优化，极大的提高了软件项目运维的工作效率，其流程如图4所示。

4 结束语

目前在IT运维领域，对用户需求的敏感性要求越来越高，对于项目的持续集成与快捷部署的要求也日益增强，在这一背景下，DevOps系统的执行效率也得到了越来越多企业的认可。本文通过将Docker容器技术、Kubernetes持续集成技术和

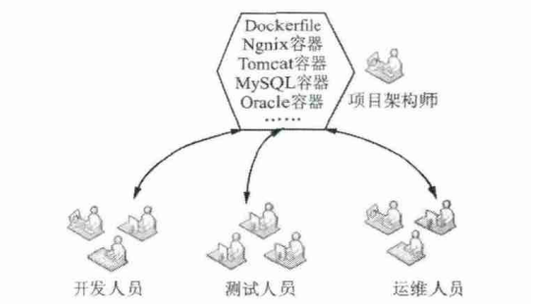

图4　本系统实现的软件项目集成

Jenkins云镜像技术等前沿技术的有机结合，设计实现了一款新型的DevOps系统，显著提高了软件项目的持续集成与发布能力。相信随着数据挖掘技术、人工智能技术和信息通信技术的不断改进，该类系统对大规模数据信息的处理能力将会不断，势必会引领下一场云计算技术发展的浪潮。

参考文献：

[1] Garber L.News briefs[J].IEEE Security and Privacy,2011,9(6):9-11.
[2] GERDIS R.DevOps: collaboration is guarantee of success[J]. Software and Integrated Circuit,2016(6):16-17.
[3] 汪恺,张功萱,周秀敏.基于容器虚拟化技术研究[J].计算机技术与发展,2015,25(8):138-141.
[4] 余双波,江泓,陈宇雷.综合网络管理系统性能管理功能的CIM 建模[J].信息安全与通信保密,2009(1):84-88.

[通联编辑：代影]

（上接第208页）

通过主机管理界面可查看主机日志及运行情况，点击表格中数字链接部分，可打开日志显示窗口，在窗口中可将日志存储为CSV文件格式，便于作数据审计和数据分析。以防火墙的日志显示窗口(如图4所示)为例，可以看到某些IP地址因连续输入错误密码被添加到黑名单，这一行为在同一段时间出现数次，而且是不同的IP地址在进行。可以推断有非法入侵者设法破解防火墙登录密码，非法进入防火墙，在IP被阻止后，采取改变IP的方法重复操作。通过这些日志，可以锁定某些入侵频率较高的IP地址。

4 结束语

通过日志服务器，我校信息中心管理人员能全面掌握网络设备和服务器的运行情况和授权访问情况，及时调整管理方案，减轻管理人员工作负担，极大地提高了网络安全的技防力量。下一步我们将进行日志审计和数据分析方向的研究，进一步提高网络安全防范能力。

参考文献：

[1] 刘东远,程东.日志服务器的研究和应用[J].电脑知识与技术,2006(33):54.
[2] 李甜.基于Syslog的日志审计系统的研究和实现[J].中国新通信,2008(12):47.
[3] 温炜,郭玲. Syslog在企业网络管理中的应用[J].宁夏电力,2009(5):42.
[4] 李晨光原创IT博客. 利用Eventlog Analyzer分析日志[EB/OL].[2014-5-19]. https://www.cnblogs.com/chenguang/p/3742234.html.
[5] 百度百科. Event Log [EB/OL].[2018-4-21]. https://baike.baidu.com/item/Event%20Log/9562319.

[通联编辑：代影]