文章编号：2096-1618(2021)05-0503-05

一种基于 Kubernetes 的 Web 应用部署与配置系统

程仲汉， 郑清安， 陈淑珍
(福建警察学院计算机与信息安全管理系，福建 福州 350007)

摘要：基于 Docker 和 Kubernetes 的容器云技术可以对大规模应用程序及其软硬件资源进行管理，并实现灵活的负载调度和弹性扩展。针对传统应用部署与配置方法费时费力且不够灵活的问题，基于 Kubernetes 容器云设计并实现了一个 Web 应用的自动化部署与配置系统，实现对 Web 应用的全生命周期管理。系统主要功能包括部署介质管理、部署环境管理和部署配置管理，可对应用及其版本、环境资源以及应用配置进行灵活管理，支持灰度发布、同步备份和弹性伸缩等特性。目前，系统已在甘肃省电力公司内网上线使用。

关 键 词：Docker 容器；Kubernetes 容器云；Web 服务；部署与配置
中图分类号：TP391.9 **文献标志码**：A
doi：10.16836/j.cnki.jcuit.2021.05.005

0 引言

相较传统的操作系统级虚拟化方式，如 Xen[1]、KVM[2]等，Docker 容器[3]是一种轻量级的应用虚拟化技术。它将应用程序及其依赖的组件和环境封装为彼此隔离的容器，由 Linux 为其分配资源并调度运行。Docker 可以消除各个执行环境间的差异，避免对环境所需组件的安装和配置，从而实现在开发环境、测试环境和生产环境进行跨平台持续集成和交付。

容器云是一种构建于分布式集群环境下的应用级 PaaS(platform as a service)技术[4-5]，以容器为资源分配、编排和调度的基本单位。容器云技术能够支撑大规模应用的高可靠运行，并实现资源监控、负载均衡等功能。相较基于操作系统虚拟化的 PaaS，如 OpenStack[6]，容器云具有资源利用率高、迁移速度快、调度灵活等优势。业界内的各大公司陆续推出自己的容器云产品，比较著名的有 Kubernetes[7-8]、Deis[9]等。

传统的应用上线发布需要人工代码包及其依赖的组件拷贝至现场，逐台机器进行安装和配置，这种方式耗时耗力且易出错。如果采用传统的远程部署配置工具，如 Ansible[10]、Puppet[11]等，则在应用运行时的运维管理方面不够灵活，无法做到应用高可用、灰度发布和弹性扩展。因此，基于 Kubernetes 容器云及 Docker 容器技术，将 Web 应用服务上线的各个环节进行统一管理，实现了 Web 应用的自动化部署与配置功能，并依托容器云特性，实现了灰度发布、快速启停、弹性伸缩、应用同步等功能。支持多版本应用在多个环境的灵活快速地集成、交付、上线和迭代。

1 Docker 容器与 Kubernetes 容器云技术

1.1 Docker

Docker 本质上是一种基于 LXC(Linux container)[12]的应用级虚拟化技术。LXC 在 Linux 的内核层面将上层应用程序以容器形式隔离，多个容器共享宿主的内核资源，并通过 cgroup 分配计算和存储资源。各个容器内部的程序具有对操作系统的独立视图，可看作运行在独立的系统环境中。LXC 的设计初衷是方便开发人员将程序及依赖包作为一个整体封装在虚拟容器中，使其易于跨平台迁移，降低环境搭建的复杂度。

Docker 将 LXC 进行了深化和扩展，在应用层为用户提供一套操作容器和镜像(容器的静态模板)的命令集。在内核层的容器驱动模块可以对 Docker 的计算、内存、网络等资源和环境进行定制管理。相较传统的系统虚拟化方式，Docker 容器启动和停止容器十分迅速(启停周期在秒级)，容器间共享硬件资源和系统内核，因此资源消耗很小。Docker 则为每个容器提供一个独立的系统微环境，而大部分资源均在内核层共享并由 Driver 调度，这就使其具有很高的资源利用率和接近原生的性能。

收稿日期：2021-01-22
基金项目：福建省工业高校产学合作资助项目(2020H6024)；福建省中青年教师教育科研资助项目(JAT200379)；福建省中青年教师教育科研资助项目(JAT190455)

1.2 Kubernetes

Kubernetes 是 Google 提出的分布式容器 PaaS 方案。它基于 Docker 容器技术将应用微服务化[13],具有完备的服务发现、容器调度、负载均衡、弹性扩展等能力。同时,对外开放源码以及 API,便于开发者定制和扩充云平台的功能。

Kubernetes 的集群部署架构为主从模式,平台架构如图 1 所示。主节点运行服务端组件,包括 Kube-apiserver、Kube-controller-manager、Kube-scheduler、etcd,负责对集群中所有资源进行管控和调度;从节点运行 Kubelet、Kube-proxy、Docker daemon、cAdvisor,负责对节点上的容器的生命周期进行管理,以及实现服务代理功能。

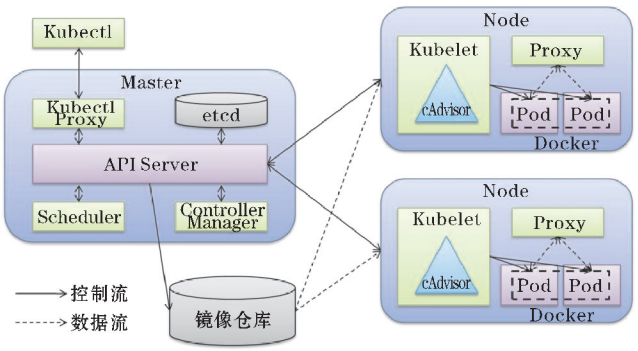

图 1 Kubernetes 架构图

Kubernetes 管理最常见的资源有:Node、Pod、ReplicaSet Controller(RC)、Service 等。Node 是集群中用于运行应用程序的从节点;Pod 是 Kubernetes 的基本操作单元,包含一个或多个相互耦合的 Docker 容器;RC 主要用于管理 Pod 的启停,确保一个应用的 Pod 副本维持在指定数量,保证高可用性;Service 为一组功能相同的 Pod 对外提供的统一的服务接口。

在主节点上的各个组件的职责:Kube-apiserver 是连接其他所有服务组件的枢纽,提供资源对象的唯一入口,通过 Kubectl proxy 监听工具中心的请求并调用其他功能模块进行工作。Kube-controller-manager 是集群内部的管理控制中心,实现容器副本管理、故障检测和恢复的自动化工作。Kube-scheduler 是集群的调度器,负责将镜像仓库中的镜像转化为容器,为其分配资源并在 Node 上的调度运行。Etcd 是高可用的 Key-Value 存储系统,用于持久化存储集群中所有的资源对象。

从节点上各个组件的职责:Kubelet 通过与本地的 Docker daemon 交互,负责 Pod 的创建、修改、监控、删除等全生命周期管理,同时定时上报本地 Node 的状态信息到 Kube-apiserver。Proxy 实现了 Service 的代理及软件模式的负载均衡器。cAdvisor 用于实时监控 Docker 上运行的容器性能指标。

2 系统设计与实现

2.1 系统架构

系统为根据国家电网甘肃省电力公司需求所研发的产品,是“信息系统自动化运维工具”的子系统。系统依照国家电网公司的信息通信系统规范,采用国家电网统一应用开发平台(Unified Applicated Platform of SGCC, SG-UAP)2.0[14]研发的 Java Web,采用Oracle 11作为业务数据库,Kubernetes 1.15.0+Docker 1.18.0作为容器云 PaaS。图 2 给出了系统的总体架构设计。系统的 Web 后端使用 RESTful API 与 Kubernetes 集群的主节点交互,并通过解析主节点返回的 JSON 结果,从而实现对控制器、服务、节点和容器等资源的管理。系统采用 SVN 存储所有 Web 应用的包文件,FTP 存储应用对应的配置文件集合,镜像仓库存储将应用整体打包好的镜像。

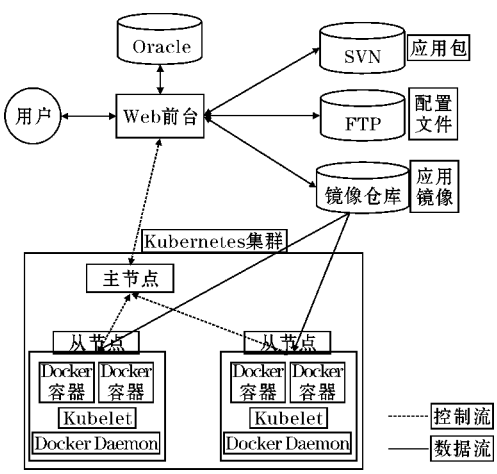

图 2 系统总体架构图

2.2 主要功能

系统在功能上主要分为 3 个部分:部署介质管理,对应用的元数据、包文件、配置信息进行维护;部署环境管理,对应用部署所需集群中的节点软硬件信息进行维护;部署配置管理,以部署任务的形式管理应用部署、升级、同步等过程。Web 应用以容器集群方式部署,并支持推送和更新配置。

部署介质主要包括 Web 应用包文件、应用包含的配置文件及应用打包生成的镜像文件。SVN 服务器对不同厂商和不同项目的应用包进行分版本存储。系统在应用包入库后会扫描包中所有配置文件,并将其上传至 FTP 服务器中,用于将来对应用进行配置管理。镜像仓库是采用 Docker Registry 搭建的私有仓库,以容器形式运行,并通过文件映射实现存储持久化。在应用包入库后,系统会提前将应用与 Web 中间件(目前支持 Tomcat 7、8 和 Weblogic 11)打包为 Docker 镜像,以减少发布所需的时间。

应用以集群方式部署,部署环境管理就是将整个 Kubernetes 资源池中的节点进行分组管理。集群按照类型分为私有集群和共享集群。对于私有集群,每个分组集群仅为一个 Web 应用所用,即某个应用仅会部署至指定集群范围内的节点中。这是通过对节点指定标签为集群 ID,并在创建部署任务时,使用节点标签选择器 NodeSelector 完成,这样就能实现不同应用间的资源隔离。至于共享集群可理解为一个大的资源池,支持多个应用共享节点资源,由 Kubernetes scheduler 的负载均衡策略来调度容器在节点上的分布。

部署任务的建立是通过在 Kubernetes 中建立一个 RC 容器集群和一个 Service 完成,两者通过设置部署 ID 号为资源标签绑定。RC 对应用应该具有容器副本进行健康检查,如果当前处于 running 状态的副本不足,它就启动新的容器,直至副本数达到用户要求,从而确保应用的高可用性。用户还可以调整 RC 的副本数,以达到容器的弹性扩展。Service 将 Web 应用包含的容器集群以一个统一的地址和端口对外提供服务,并支持 Kubernetes 内置的软件负载均衡方式。

为清晰说明系统各个功能之间的关系,图 3 给出应用部署过程的整体流程。首先,用户使用“部署介质管理”将应用包上传至镜像仓库,再使用“部署环境管理”创建部署集群并加入主机节点,最后使用“部署配置管理”创建部署任务,完成应用的配置与上线。应用运行期间,系统支持版本更新、版本回滚、不同环境间的应用同步、配置修改、扩容缩容等功能。

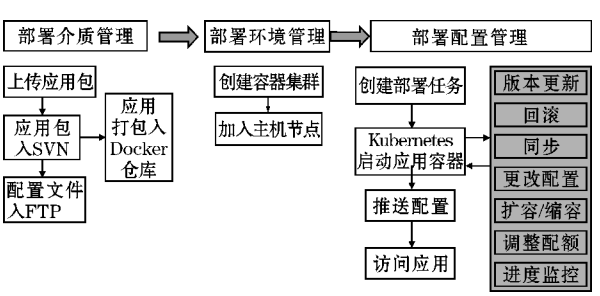

图 3　部署与配置流程图

3　关键子功能

系统的 3 个关键子功能为:灰度发布、应用同步和配置推送的实现细节。

3.1　灰度发布

在传统的 Web 版本升级时,旧版本应用服务会先被终止,然后待新版本应用部署启动后才可以使用,这会造成服务在升级期间不可用。因此,系统利用 Kubernetes 的 rolling-update(滚动升级)接口,实现灰度发布功能。在滚动升级期间,副本控制器会将旧版本的 Pod 逐个销毁,并启动新版本的 Pod,同时保证总副本数维持不变。新版本 Pod 会逐步替换,且期间服务不会中断,直至旧版本 Pod 全部被更新。在滚动升级期间,用户访问到哪个版本由服务的负载均衡器决定。

3.2　应用同步

传统的主备集群服务器方式需要为每个重要的 Web 应用准备一套灾备服务,用于在主服务宕机时实现热切换。然而,灾备服务在大部分情况下并没有被使用,这就造成了硬件资源的浪费。如果使用冷切换的容灾方式,灾备应用就无法及时提供服务。同步功能为以上问题提供了解决方案:在灾备环境下准备一个共享容器池,为每个主应用各创建包含少量容器副本的灾备服务(只占用少量资源),当某个主应用宕机时,再动态扩容出足够多的容器副本。这样,通过容器云的弹性扩展功能和 Docker 的快速启停容器机制,既实现了无缝切换的容灾,又高效地利用了资源。

同步的目标有两个方面:将 Web 应用从开发测试到上线进行快速集成;为服务创建备份应用,以便当主应用宕机时,可实现应用的持续可用性。系统将部署集群按照用途分为测试环境、生产环境和灾备环境。同步可以将部署在测试环境下的应用克隆至生产环境,或将生产环境下的应用克隆至灾备环境。同步前会检查目标集群是否已部署应用,如果没有则直接部署,否则会检查应用版本并进行更新。同步的结果是目标和源集群中的 Web 应用及其版本一致,但具体的容器规模和配置信息可根据用户需求和实际情况修改。同步的功能界面如图 4 所示。

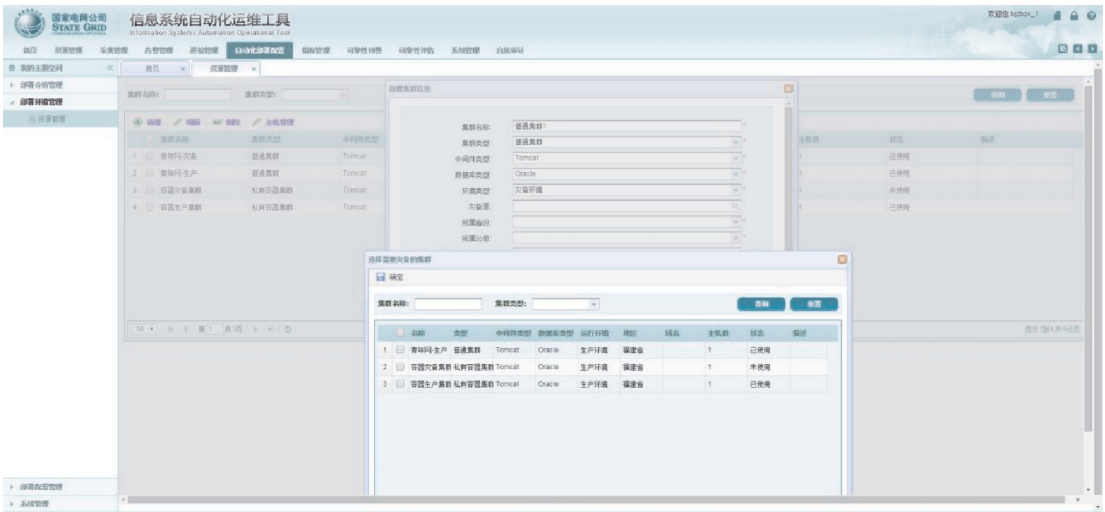

图 4 同步功能界面

3.3 配置推送

容器的运行状态是与外界隔离的,因此修改容器内部的配置文件很困难。配置推送功能的目标是对一个 Web 的容器集群进行统一配置,将需要修改的配置信息动态推送到容器内部。一般 Web 项目的配置文件有 ini、properties 和 xml 3 种格式,在 FTP 上存储了每个应用包版本的全量配置文件,并创建了包含配置默认值的模板。对于 ini 和 properties 这 2 种格式,系统会提前抽取配置的键值对信息并以 Key-Value 格式存储。当创建应用容器时,系统会将用户修改过的配置(即相对配置模板变动的部分)以环境变量的方式写入 Docker 内部,信息包括配置文件路径、配置项名称和配置项值,如图 5 所示。

```
DEPLOY_5=/WEB-INF/osproperty.properties#<>#APPLICATION_PORT1#<>#8090
DEPLOY_6=/WEB-INF/osproperty.properties#<>#RT_INTERVAL#<>#300000
DEPLOY_ID=a59ff7ba15d264cfa9bf6582
DEPLOY_7=/WEB-INF/osproperty.properties#<>#RT_FETCH_INTERVAL#<>#1
DEPLOY_0=/WEB-INF/osproperty.properties#<>#RT_SPAN_TIME#<>#1800000
DEPLOY_1=/WEB-INF/osproperty.properties#<>#LT_INTERVAL#<>#300000
DEPLOY_2=/WEB-INF/osproperty.properties#<>#APPLICATION_IP1#<>#192.168.253.223
DEPLOY_3=/WEB-INF/osproperty.properties#<>#RT_FETCH_NUM#<>#1
```

图 5 使用 Docker 环境变量存储修改的配置信息

在容器启动时,通过修改中间件启动脚本 catalina.sh 来获取环境变量并动态修改配置。Xml 文件为树型结构,难以用 Key-Value 格式存储增量配置。因此,只能采用全量推送的方式:将修改好的 Xml 文件以文本方式存储在 FTP 中,然后再由 catalina.sh 动态下载并覆盖配置文件。系统的配置推送功能界面如图 6 所示。

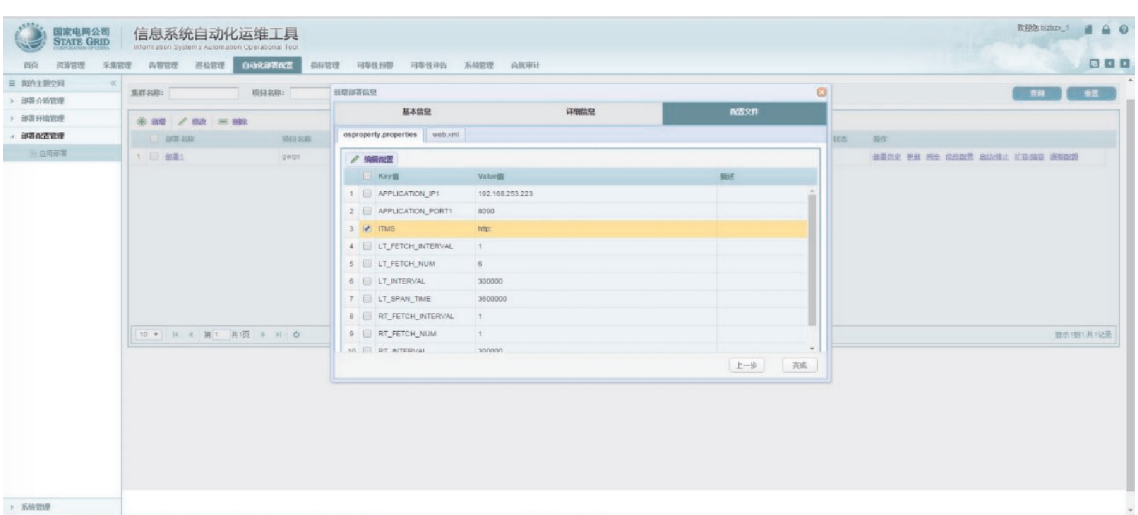

图 6 配置推送功能界面

4　结束语

Docker 容器具有轻、薄、快的特性,可为应用程序的持续集成和快速创建提供便利。容器云技术的大规模应用管理能力则支撑了容器集群的高可用、负载均衡、动态伸缩等特性。基于当前业界最为主流的开源容器云平台 Kubernetes,提出一种 Web 应用自动化部署与配置系统。系统主要功能包括部署介质管理、部署环境管理、部署配置管理,贯穿了应用上线发布的全生命周期,还包括应用的灰度发布、应用同步和配置推送等关键子功能。系统能够改变传统低效且复杂的应用部署配置和运维管理方式,已在甘肃省电力公司内网上线使用。

参考文献:

[1] Barham P,Dragovic B,Fraser K,et al. Xen and the art of virtualization[J]. Proc of Sosp,2003,37(5):164-177.

[2] Kivity A,Kamay Y,Laor D,et al. KVM:the Linux virtual machine monitor[C]. Linux Symposium,2007.

[3] Boettiger C. An introduction to Docker for reproducible research, with examples from the R environment[J]. ACM SIGOPS Operating Systems Review,2014,49(1):71-79.

[4] 杨清波,陈振宇,刘东,等. 基于容器的调控云 PaaS 平台的设计与实现[J]. 电网技术,2020,44(6):27-34.

[5] 杨凯琪,赵玉龙,陈林. 异构容器云间应用迁移模型研究[J]. 计算机应用研究,2020,37(4):143-147.

[6] Sefraoui O,Aissaoui M,Eleuldj M. OpenStack:Toward an open-source solution for cloud computing[J]. International Journal of Computer Applications,2012,55(3):38-42.

[7] Bernstein D. Containers and cloud:from LXC to Docker to Kubernetes[J]. IEEE Cloud Computing,2014,1(3):81-84.

[8] 龚正. Kubernetes 权威指南:从 Docker 到 Kubernetes 实践全接触[M]. 北京:电子工业出版社,2016.

[9] 浙江大学 SEL 实验室. Docker:容器与容器云[M]. 北京:人民邮电出版社,2015:257-260.

[10] 黄巨涛,杨永娇,刘梓健,等. 基于 Ansible 的电力云平台自动部署系统设计[J]. 电子设计工程,2020,28(3):43-46.

[11] 李新虎,刘正伟,刘俊朋. 基于 puppet 工具的软件批量部署的实现[J]. 信息技术与标准化,2014(6):70-73.

[12] Bernstein D. Containers and Cloud:From LXC to Docker to Kubernetes[J]. Cloud Computing,IEEE,2014,1(3):81-84.

[13] 张丽敏,高晶,李务斌,等. 微服务环境下容器编排可视化实践研究[J]. 计算机工程与科学,2019,41(8):1366-1373.

[14] 游龙勇,崔金杰,贺健博,等. 国家电网公司"五位一体"管理平台设计[J]. 电力信息与通信技术,2015,13(3):78-82.

A Web Application Deployment and Configuration System based on Kubernetes

CHENG Zhonghan,　ZHENG Qingan,　CHEN Shuzhen

(Department of Computer and Information Security Management, Fujian Police College, Fuzhou 350007, China)

Abstract: Container cloud technology based on Docker and Kubernetes can manage large-scale applications and their software and hardware resources, and achieve load scheduling and elastic expansion. Aiming at the problem that traditional application deployment and configuration methods are time-consuming, laborious and inflexible, a web application automatic deployment and configuration system based on Kubernetes container cloud is designed and implemented, which realizes the whole life cycle management of web application. The main functions of the system contain deployment media management, deployment environment management and deployment configuration management. It can flexibly manage the application and its version, environment resources and application configuration, and support gray publishing, synchronous backup and elastic scaling. At present, the system has been used in the intranet of Gansu Electric Power Company.

Keywords: Docker container; Kubernetes container cloud; Web service; deployment and configuration