文章编号：1005-8451（2019）12-0049-05

基于Kubernetes的容器云平台设计与实践

翁涅元，单杏花，阎志远，王雪峰
（中国铁道科学研究院集团有限公司 电子计算技术研究所，北京 100081）

摘 要：为解决铁旅App既有后台架构应用研发不够灵活，平台运维成本不断提高的问题，以提高平台研发灵活性和自动化运维水平为目的，提出了基于Kubernetes容器云平台的总体架构设计，并结合容器云平台对研发流程进行了设计。经应用实践证明，该平台架构在保证平台运行稳定性的同时，可以有效提高研发效率并降低运维成本。

关键词：Kubernetes；Docker；软件架构；自动化运维
中图分类号：U29 : TP39 **文献标识码：**A

Design and practice of container cloud platform based on Kubernetes

WENG Shengyuan, SHAN Xinghua, YAN Zhiyuan, WANG Xuefeng
(Institute of Computing Technologies, China Academy of Railway Sciences Corporation Limited, Beijing 100081, China)

Abstract: In order to solve the problems of inflexible application research and development of the existing background architecture of railway travel App, and the increasing cost of platform operation and maintenance, this article proposed the overall architecture design based on Kubernetes container cloud platform, and designed the research and development processin combination with the container cloud platform. The application practice shows that the platform architecture can effectively improve the research and development efficiency, reduce the operation and maintenance cost while ensuring the stability of the platform operation.

Keywords: Kubernetes; Docker; software architecture; automatic operation and maintenance

随着铁路信息技术的不断发展，旅客在对铁路出行满意度要求不断提高的同时，对客运延伸服务也提出了更高的要求。随着多样化旅客出行服务需求的不断提出，铁旅App服务平台也在不断添加新的业务模块，以便为旅客提供更好、更丰富的服务[1]。

不断增多的应用服务数量带来了平台的运维和升级问题。由于铁旅App服务平台采用微服务体系开发，组件众多且相互依赖，每次业务调整与升级均会涉及多个模块的同时调整，导致业务升级缓慢，无法满足快速迭代升级的要求，升级部署成功率低。平台运维方面，以手工操作结合辅助脚本的方式为主，出错率高，平台稳定性差。平台各业务功能主要通过人工巡检的方式进行日常维护，难以及时发现平台故障，且故障修复周期长，增加了运维人员的工作量和人力成本。综上，较长的升级研发周期，较高的平台故障率以及较高的人工运维成本制约了平台服务的深入开展，亟需探索提高平台研发灵活性、稳定性，降低运维成本的方法。

国内外在云平台建设方面取得了很多重要成果，现阶段平台建设领域的容器技术和容器编排工具是各大公司的研发主流。本文根据铁旅App自身业务需求和特点，结合Kubernetes等开源工具搭建私有容器云平台，将原业务系统进行容器化改造后迁移至云平台，以达到提高平台灵活性、稳定性，降低运维成本的目的。

1 容器编排工具概述

1.1 Docker

Docker是一个开源的应用容器引擎，通过将应用与相关的依赖项（环境、程序、文件）打包成镜像文件，可以将其方便地部署于任何支持Docker运行环境的主机中，最大限度地消除开发环境与部署环境的差异，结合Docker镜像仓库，研发人员可以

收稿日期：2019-03-22
基金项目：中国铁路总公司科技研究开发计划系统重大课题（P2018X002）
作者简介：翁涅元，助理研究员；单杏花，研究员。

便捷地管理镜像与交付版本[2]。同时，Docker与微服务架构有着天然的契合度，因此可做为构建容器云服务平台的基础。

1.2 Kubernetes

对于大型企业而言，随着系统平台规模的不断扩大，应用服务数量和服务器数量也逐渐增加，运维成本不断提高，单纯依靠Docker已无法满足系统运维需求。Kubernetes是Google的开源容器集群管理解决方案，在提供强大的集群管理、故障检测与自动修复能力的同时，也提供便捷的服务升级解决方案，同时支持服务的发现调度和弹性伸缩等特性。鉴于Kubernetes有Google强大的开源支撑和众多成功应用的案例，基于Kubernetes构建容器云服务平台是大型企业构建生产环境容器云服务平台的较优选择之一[3]。

2 平台设计

2.1 总体设计

容器云平台部署于物理服务器之上，总体结构由资源层、平台层、应用层3部分组成，如图1所示。

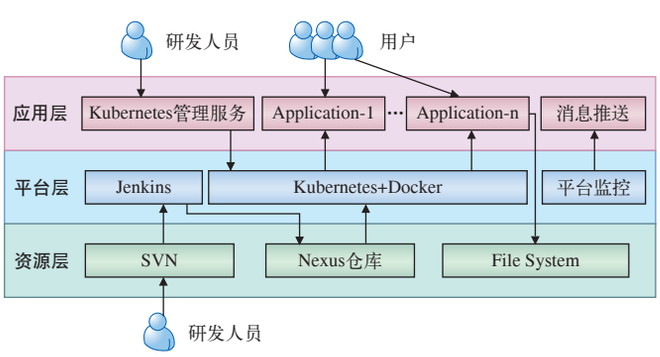

图1 容器云平台总体架构图

资源层位于平台底层，为平台提供基本存储和资源管理能力，包含用于代码版本管理的SVN，用于构建Maven和Docker镜像仓库的Nexus，以及文件存储服务。

平台层包括容器云平台的核心组件，如：集群监控程序，用于提供容器镜像基本运行能力的Docker工具，为Docker统一编排与调度能力的Kubernetes，用于将项目构建为容器镜像并上传至Nexus仓库中的Jenkins。

应用层最接近用户，用于部署为用户开发的业务应用，以及定制化的集群控制软件等。

2.2 平台功能设计

容器云平台从项目全生命周期支持和自动化运维支持2个方面为开发人员提供服务。项目全生命周期支持包括代码管理、持续构建[4]、环境隔离、统一部署等多个功能，覆盖项目启动研发至部署上线全生命周期；自动化运维支持包括平台监控工具、故障自动恢复和水平扩容支持等功能，为运维人员提供了高效便捷的自动化工具。

2.2.1 项目全生命周期支持

容器云平台围绕项目全生命周期为研发人员提供如下服务。

（1）代码管理

云平台内使用SVN作为代码版本管理与协同开发的基础，项目研发人员统一将代码提交至SVN进行代码存储。该功能是项目构建的基础。

（2）持续构建

云平台内使用Jenkins作为持续构建工具，通过定制的定时触发和手动触发机制，从SVN代码仓库中提取最新的项目源码进行构建、打包并推送至镜像仓库进行存储，以便后续使用。

（3）环境隔离

云平台利用Kubernetes命名空间资源隔离的特性为研发人员提供研发、预发布和生产环境，3个环境彼此隔离部署，使研发人员可以更安全地测试新业务而不影响线上正在运行的既有服务。

（4）统一部署工具

云平台提供统一的项目发布工具，保持了平台内项目结构的一致性，同时也将研发人员从Kubernetes工具繁琐的操作流程中解放出来，降低了对研发人员的技术要求。统一的项目发布工具记录了用户的每一步操作，使得平台应用更新有迹可循，为潜在的业务升级故障风险提供了修复参考，统一部署界面如图2所示。

2.2.2 自动化运维支持

（1）平台监控工具

云平台监控工具提供平台运行状态、物理节点、应用状态、服务调用负载等全方位监控指标，为运维人员提供统一的监控视图，同时对平台产生的异

图2　统一部署管理界面截图

常提供钉钉自动报警功能，显著缩短了故障的发现时间，降低了人工巡检成本。平台监控的物理节点及应用层监控截图如图 3 所示。

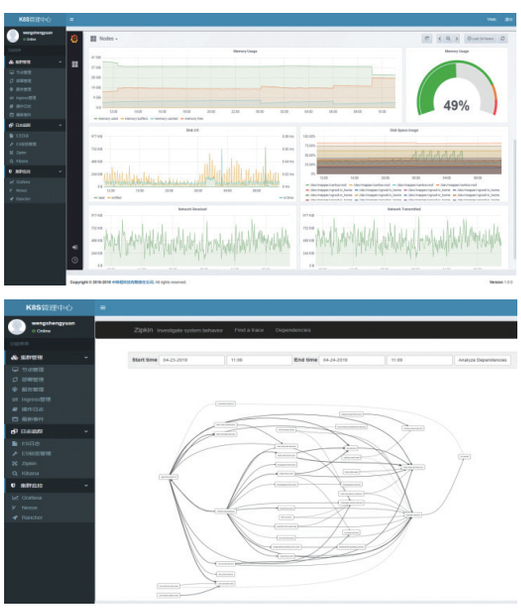

图3　物理节点及应用层监控截图

（2）故障自动恢复

云平台依赖镜像仓库保存了应用的所有历史版本，在出现应用升级故障时，可以及时回滚为旧版应用。

（3）水平扩容支持

通过容器封装的应用服务可以很方便地以增加副本的形式进行水平扩容，运维人员仅需根据负载需求设置副本数量即可。

2.3　平台可靠性设计

在云平台的日常运行中，不可避免会碰到节点失效的情况，比如，由于内核安全漏洞、基础软件缺陷或硬件驱动 Bug，甚至潜在的硬件故障和计划性停机维护等导致需要在宿主机上进行重启操作。容器云平台需要采取必要的技术方案来降低节点重启带来的影响，保证容器云平台的高可靠性。以下对部分关键技术方案进行说明。

2.3.1　可靠的文件、数据存储系统

容器云平台中对重要文件采用 CEPH 分布式文件系统进行存储，以提供更高的可靠性和扩展性；对用户数据采用主备方式建立数据库集群；同时，采用 ETCD 集群为平台提供高性能、高可靠性、高一致性的配置数据管理服务 [5]。

2.3.2　Kubernetes集群高可用部署

容器云平台中所有的用户应用均运行于 Kubernetes 环境之下，因此 Kubernetes 是容器云平台的核心组件，对其可靠性有极高要求。鉴于 Kubernetes 中 Master 和 ETCD 存储组件的重要性 [6]，在设计云平台部署架构时，通过 Active–Active 的 Master 多节点模式和 ETCD 服务集群方式来保障 Kubernetes 集群的稳定性，使得仅当所有 Master 节点同时故障的情况下，才会影响平台运行，任意单节点宕机均不会影响集群的正常运行。Kubernetes 高可用集群结构如图 4 所示。

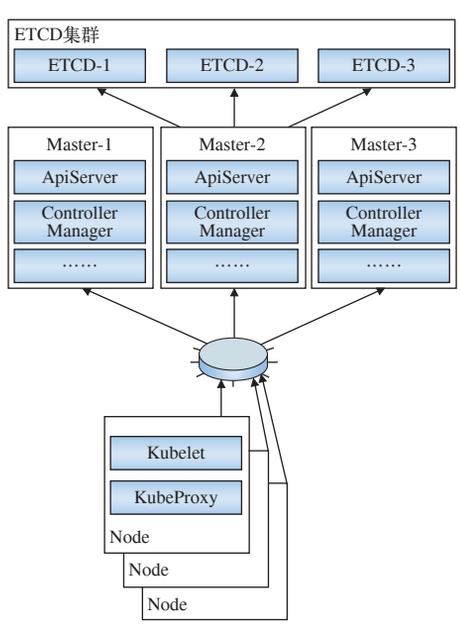

图4　Kubernetes高可用集群结构图

2.3.3 应用高可用部署

平台用户所部署的应用均以容器的形式运行于 Kubernetes 环境下，Kubernetes 通过 Replication Controller/Replica Set[7] 服务提供应用副本的冗余性。用户将需要部署的应用打包为容器镜像，以 Deployment 的形式部署于 Kubernetes 环境之下，通过合理设置 Kubernetes 应用健康检测探针和应用部署的副本数量即可保证应用的可靠运行。在出现节点故障或由于应用本身的原因导致程序奔溃时，Kubernetes 可自动重启应用或新建应用副本，最终确保可用副本数量符合用户设置。

2.3.4 多级监控与自动报警

容器云平台分别从系统、平台、应用 3 个层面对平台健康状况进行全方位监控 [8]。

（1）系统层面，在所有机器上部署 Zabbix 监控客户端，实时采集并上报宿主机和系统的关键指标，一旦发现故障则通过 Zabbix 触发器将报警信息发送给运维人员，以便及时排除故障。

（2）平台层面，通过自主研发的 Kubernetes 管理监控服务，调用 Kubernetes 的 ApiServer 定时查询 Kubernetes 集群的关键指标，如节点运行状态、网络通信状态、Kubernetes 关键组件运行状态等 [9]。一旦发现 Kubernetes 集群内部组件出现异常，立即通知平台运维人员。

（3）应用层面，该层面的监控利用 Kubernetes 的健康状态检测探针机制，定期检测应用运行状态，当应用运行状态发生异常时，自动重启故障副本并通知运维人员。

3 操作标准和流程设计

影响系统稳定性的诸多原因中，人为因素也是很重要的一部分，因此需要尽可能避免人为因素造成的系统异常。结合容器云平台特性，本文制定了一套开发、测试与部署的操作标准和流程 [10]。

3.1 操作标准

（1）研发人员应以小步增量的方式进行研发，保持较短的构建和测试过程，维持较高的测试版本更新频率；

（2）所有项目代码、配置、说明文档均上传 SVN 进行版本管理，避免线下沟通和口头沟通；

（3）设定版本里程碑，保证生产环境升级的有序性和计划性，新增服务或升级服务在生产上线前必须通过开发环境测试和预发布环境灰度测试；

（4）通过权限设置禁止研发人员绕过统一部署平台进行越权操作；

（5）研发人员必须依赖平台内代码仓库和组件私服以保证平台范围内的关键代码和组件的统一版本管理；

（6）编写编码规范手册、集群操作手册等相关文件，对平台使用人员进行引导。

3.2 流程设计

（1）利用 Kubernetes 的命名空间机制将平台计算资源划分为开发环境、预发布环境和生产环境 3 个相互独立的子空间，实现不同环境的资源隔离，确保生产环境的安全与稳定；

（2）研发人员将代码托管至 SVN 服务，在有效的版本控制机制下实现合作开发；

（3）Jenkins 组件负责定期提取 SVN 上的代码，并按照项目配置构建应用镜像，上传至 Nexus 镜像仓库并打上相应的版本标签；

（4）在开发与测试阶段，研发人员与测试人员均在开发环境下对正处于开发过程中的程序进行部署、升级、测试和调整，研发人员通过自主研发的 Kubernetes 管理服务统一进行应用的部署与升级操作。

（5）新版发布阶段，运维人员首先在预发布环境下部署最新版本的生产程序，并适当引入少量生产环境的流量进行灰度测试。

（6）当灰度测试通过后将新版本程序正式部署于生产环境，并最终完成程序的研发迭代过程。

系统研发的迭代流程如图 5 所示。

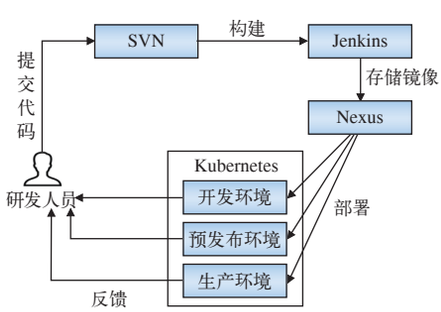

图5 研发流程示意图

4　云平台实施效果

容器云平台设计与搭建完毕后，将铁旅 App 服务的业务系统进行迁移测试，得到如下实施效果。

（1）应用部署效率提高

不同于原来手动拷贝副本并重启应用的升级方式，在容器云平台内部署应用仅需在管理界面选择自动构建好的应用镜像并填入所需的副本数量即可，其余工作全部由 Kubernetes 自动完成。显著降低部署难度，提升成功率，使原本 3 ～ 5 天 1 次的升级频率提升为 1 天多次，且不影响服务的稳定性。更高的部署效率令后台可以更迅速地响应产品需求，更快地推出更新、更完善的服务。

（2）服务故障率下降

借助容器云平台的高可用架构设计，可以实现有效冗余和故障自动恢复，不再发生由于机器故障导致后台无可用服务实例的情况，后台服务整体稳定性得到显著提升。

（3）系统监控能力提升

得益于容器云平台的多级监控机制，运维人员可以实现对系统、平台、应用的全方位监控，监控可以细化至每一个节点和应用实例。监控指标更加丰富，同时借助消息推送渠道，使软硬件故障通知更为及时，响应更加迅速。

（4）人员成本下降

容器云平台将运维人员从大量繁琐的手动操作步骤中解放出来，减少了大量原本需要手动记录的部署信息，有效减少了运维人员的工作量。

5　结束语

容器云平台基于 Kubernetes 技术将研发与运维人员从繁琐的手工操作中解放出来，同时，为系统后台提供了高可靠性、失败冗余和容灾恢复等性能。容器云平台在铁旅 App 业务系统服务后台的改造实践中取得了良好效果。对于在大规模并发访问和业务负载波动剧烈的情况下，如何应用 Kubernetes 技术做到弹性自动扩容和动态分配系统资源仍待进一步研究。

参考文献

[1] 刘卓华，李聚宝，王丽华，等．关于 12306 网站旅客服务的研究与设计 [J]. 铁路计算机应用，2016，25（10）：38-40.

[2] 丁海斌，崔　隽，陆　凯．基于 Docker 的 DevOps 系统设计与实现 [J]. 指挥信息系统与技术，2017（3）：87-92.

[3] 宗序梅，任彦辉．基于 Kubernetes 的 PaaS 平台研究与实践 [J]. 江苏通信，2018（2）：76-79.

[4] 张文林．持续交付及其在大型项目中的应用 [J]. 软件导刊，2017，16（10）：159-161.

[5] 盛乐标，周庆林，游伟倩，等．Kubernetes 高可用集群的部署实践 [J]. 电脑知识与技术，2018，14（26）：46-49.

[6] 陈卫平．高可靠、可用数据库系统平台的建立 [J]. 铁路计算机应用，2005，14（z1）：239-242.

[7] 王骏翔，郭　磊．基于 Kubernetes 和 Docker 技术的企业级容器云平台解决方案 [J]. 上海船舶运输科学研究所学报，2018，41（3）：54-60.

[8] 仇　臣．Docker 容器的性能监控和日志服务的设计与实现 [D]. 杭州：浙江大学，2016.

[9] 樊　炼，廖振松．一种云计算时代的 DevOps 自动化运维平台 [J]. 电信工程技术与标准化，2018，31（11）：68-71.

[10] 董　昕，郭　勇，王　杰．基于 DevOps 能力模型的持续集成方法 [J]. 计算机工程与设计，2018，39（7）：1930-1937.

责任编辑　李依诺