白皮书

构建容器化应用的 5 个检测策略

David Gordon

执行摘要

由于世界各地的企业都在采用基于容器的应用架构，开发人员必须要能根据容器平台环境对最初部署到传统基础架构的现有软件实施方案进行调整。本白皮书提出了应用容器化的五个基本问题及对应的应用检测策略，这些策略均基于热门的开源和容器原生技术。

外部化配置

一般而言，容器镜像是一种不可变且经过证明的组件，而容器环境中的持续交付正是基于这一原则。这意味着，打个比方，对于接受过 QA（质量保证）环境测试的同一镜像，无需更改持续整合/持续交付 (CI/CD) 流程序列中的下一个环境即可进行部署。然而，某些应用配置通常会因环境而异，这一点毫无疑问。常见的环境特定应用配置包括外部数据库连接字符串、公共应用编程接口 (API) 端点和功能切换标志。为了保持容器的不可变性，应从外部来源加载环境特定配置。

在传统软件架构通常采用的方法中，应用会使用文件系统（与应用主机相连接）中的环境特定配置数据。该方法基于的关键假设是，给定应用的部署对象是一台可预测的主机或一组主机。

Kubernetes 和 OpenShift 等平台会使用调度算法来评估可用的基础架构，然后将容器动态地分配给集群中的主机。容器可被分配给符合调度算法标准的任意主机节点。因此，在集群中的任意位置，应用配置信息必须随时可用。

不管是哪种为容器环境提供应用配置的现代化方案，都会遵循一个共同的基本原则，即通过服务端点提供配置数据。

Spring Cloud Config Server 是一项备受欢迎的开源应用配置提供商服务。默认情况下，该服务会开放 RESTful 端点，而该端点可返回在 Git 存储库中维护的配置属性文件。Spring 社区也提供应用端客户端，可用于检索配置信息。借助这些组件，Spring 应用可配置为定期轮询配置提供商服务，并在检测到更改时重新加载其上下文。要想实施该策略，就需要正在运行的 Spring Cloud Config Server 实例和配置数据存储，如托管的 Git 存储库。

在另一种方法中，OpenShift API 会开放 ConfigMap 对象，该对象代表的是可通过集群中的任意授权容器集访问的文件，如应用属性文件。Spring Cloud Config Kubernetes 客户端允许 Spring 应用在运行时直接使用来自 ConfigMaps 的配置数据。因此，即使没有正在运行的 Spring Cloud Config Server，部署到 OpenShift 的 Spring 应用也可以动态地使用配置。该客户端由上游 Fabric8 社区开发，并受红帽® JBoss® Fuse 支持。

日志管理

一般而言，应用日志会写入到磁盘文件上，因此，可能存在这样的假设：应用主机是预先确定的。了解应用进程的运行位置后，运营团队即会清楚日志的写入位置。因为容器在部署到 Kubernetes 等容器平台时会被动态地分配给各个节点，所以对于将日志写入本地文件系统的应用来说，其效率并不高。具有不同格式化、命名和路径约定的日志最终会分散在整个集群基础架构中。此外，容器的本地文件系统通常与容器本身的生命周期相一致。如果容器进程遭到破坏，则写入本地文件系统的所有内容也将被销毁。

对于容器原生日志管理解决方案，需要具有集群级日志目的地和格式化约定，而这可以借助容器引擎来实现。例如，默认情况下，Docker 会捕获容器主进程的标准输出和标准错误信息，同时使用基于 JSON 的标准化格式将其写入文件。这些文件位于 /etc/docker 下，其命名使用的是生成相应日志的容器 ID，因而名称唯一。在这种情况下，对应用实施的影响微乎其微，因为所有的应用日志记录都会转至标准输出。日志框架，如自称为 log4j 后续版本的 Logback，可通过简单的配置将日志记录转至标准输出。应对基于容器的应用进行专门设计，以便使用具有 Logback 等可配置框架的记录器，这样，便可对格式化和输出目的地进行有效管理。标准输出 (stdout) 和标准错误 (stderr) 应当是仅有的日志记录目的地。

除了容器日志结构，运营团队还需要采取适当的策略来收集和搜索分散在整个容器平台基础架构中的日志。可提供日志聚合、存储与显示的技术堆栈 EFK（ElasticSearch、Fluent.d、Kibana）就是一种广受欢迎的策略。容器日志和平台组件日志会被收集并传输至分布式数据存储。数据会显示在 Kibana 的高可配置性用户界面上，该用户界面还具备设计仪表板和研究个别容器集行为的功能。

分布式跟踪

通过跟踪能够了解到事务在分布式系统中传播的过程。因此，要想实现跟踪，就必须使用从系统的多个组件处收集的数据，以将有关某个事务的信息整合在一起。

基于容器的应用通常部署为多个组件，然后组成一个系统协同工作。在将应用构建为可独立部署的模块化服务套件时，通常采用的是微服务模式。虽然在开发用于部署到容器平台的应用时，完全可以选择是否严格遵循微服务模式，但务必要记住，部署到容器平台的应用往往由多个容器组成。在高度分散型和/或分解式架构中对事务进行跟踪具有一定的挑战性，因为跟踪数据源分散在整个基础架构池中。在业内，专为分布式系统跟踪开发的多种开源解决方案的应用正越来越广泛。

Zipkin 是一款开源跟踪系统，作为 Spring 框架生态系统的一部分得到了广泛普及，在过去几年内的采用率颇高。Uber 等其他公司已开发出分布式跟踪工具的全新实施方案。

有人担心跟踪框架的实施可能会导致系统与该特定实施出现耦合，而 OpenTracing 计划的建立就是为了制定供应商中立的跟踪标准，从而减轻这种担忧。OpenTracing 计划所规定的规格和语义约定在很大程度上受到了 Zipkin 的启发。Uber 开发的 Jaeger 代替了多种 Zipkin 组件，同时遵循 OpenTracing 约定，有效保证了在兼容 OpenTracing 的不同实施方案之间进行切换的能力。

符合 OpenTracing 标准的解决方案允许架构在未来进行灵活扩展。其中一些最有效且被广泛采用的解决方案（如 Zipkin 和 Jaeger）还具备随时可用的功能。

指标

在容器环境中进行有效的应用指标收集时会面临诸多挑战，其中许多与之前所述的日志记录和跟踪所遇到的挑战相同。考虑到容器生命周期的短暂性，指标端点并不是静态的；容器实例最终会被升级后的实例替换，以便在集群中的不同节点上启动。Kubernetes 和 OpenShift 等平台采用的是被称为服务的网络抽象，这种服务可定义逻辑容器集以及访问这些容器集的策略。然而， Kubernetes 服务会提供无效的监控端点，因为其需要的是有关单个容器的更加精细的统计数据，而非一组容器的全部统计数据。

Prometheus 是一种开源监控和警报工具包，在微服务环境中格外受欢迎。Prometheus 包括用于收集和显示指标的组件以及检测库的综合菜单，如 Java 管理扩展 (JMX) 导出器。

JMX 是监控 Java™ 应用的标准。借助 Jolokia 和 Prometheus JMX Exporter 等指标代理，指标用户可使用 Kubernetes API 以聚合视图观察一组容器的 JMX 指标。

对于容器环境中的每个应用组件，建议开放应用指标。同时，还可考虑使用备受欢迎的检测库来发布符合 Prometheus 阐释格式的指标端点。与分布式跟踪域相似，应用指标和监控似乎倾向于普遍接受的规范，而 Prometheus 已有朝领先标准发展之势。

健康检查

对于自我修复型架构而言，使用健康检查端点对容器化应用进行检测至关重要。部署到传统基础架构的应用往往具备静态网络地址，因此，传统应用监控通常会利用服务或主机端点的可预测性。由于容器会动态地在节点上运行，健康监控服务必须记录所有容器实例。

Kubernetes 能够自动监控健康检查端点，并对处于不健康状态的容器作出响应。集群中的所有容器都会受到监控，而 Kubernetes 还能通过部署、删除或重新启动容器集来响应健康检查。应用必须开放一个 API 以用于观察其健康状态，因此需要进行一些细微检测。

健康检查检测的有效性会因情况而异。例如，应用在被调用时，可开放返回 200 HTTP 响应代码的独立控制器。在许多情况下，这种健康检查都能起到作用，但只能发现特定类型的问题。如果应用与数据库之间的连接不正常，则对端点的粗略健康检查可能检测不出什么问题。

对所有组件和连接的状态进行全面清查是最有效的健康检查方式，这对于应用本身来说至关重要。Spring Boot Actuator 是一种备受欢迎的健康检查检测库，可对 Spring 应用上下文进行扫描，并询问其发现的每个组件的运行时状态。至于容器化应用，强烈推荐采用受 Spring Boot Actuator 支持的深度健康检查类型。

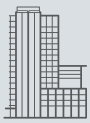
综述

随着容器技术采用率不断上升，传统应用的迁移工作也在努力与之保持同步。如果拥有一整套可以解决常见容器化问题的工具，将有助于简化新应用开发和传统应用迁移工作，可确保实施方案灵活有效。

在许多情况下，无论是部署到容器平台，还是部署到传统基础架构，无需进行更改即可实现有效的应用实施方案。然而，通过采用相关技术，开放有关应用运行时的信息，同时确保应用能够观察和响应环境条件，可以高效地管理由许多容器组成且部署到基础架构池的应用。

有关本白皮书中所述检测技术的更多信息，请参阅以下资源：

- 用于外部化应用配置的 **Spring Cloud Config Kubernetes** (https://github.com/spring-cloud-incubator/spring-cloud-kubernetes)
- 用于应用日志管理的 **Logback** (https://logback.qos.ch)
- 用于基于 **Open Tracing** 标准 (http://opentracing.io) 进行分布式跟踪的 **Zipkin** (https://zipkin.io) 和 **Jaeger** (http://jaegertracing.io)
- 用作指标工具包的 **Prometheus** (https://prometheus.io)，包括用于指标发布的 **Prometheus JMX Exporter** (https://github.com/prometheus/jmx_exporter)
- 用于开放深度健康检查端点的 **Spring Boot Actuator** (https://spring.io/guides/gs/actuator-service)



关于红帽

红帽是世界领先的开源解决方案供应商，依托社区力量为客户提供稳定可靠及高性能的云技术、Linux、中间件、存储和虚拟化产品。红帽还提供屡获殊荣的支持、培训和咨询服务。作为紧密连接全球企业、合作伙伴和开源社区的中心，红帽致力于通过为广大客户提供实用、创新型技术产品，有效释放其宝贵资源以推动业务增长，并为未来 IT 发展奠定坚实基础。

*查看更多**红帽产品组合**信息，请访问 redhat.com/zh*

销售及技术支持

800 810 2100
400 890 2100

红帽软件（北京）有限公司

北京市朝阳区东大桥路 9 号
侨福芳草地大厦 A 座 8 层
邮编：100020
86 10 6533 9300

红帽官方微博　红帽官方微信

cn.redhat.com
#f11943_0418