# Deploying Microservices: Spring Cloud vs. Kubernetes

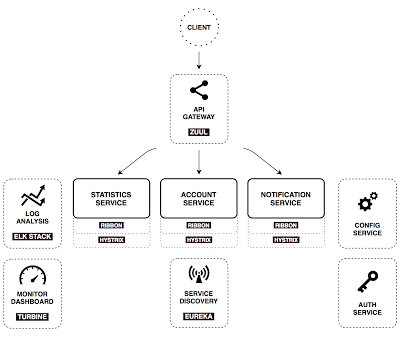
本文结合度娘译自国外大神[Bilgin Ibryam](https://dzone.com/users/1028163/bibryam.html" \o ")，原文链接：

https://dzone.com/articles/deploying-microservices-spring-cloud-vs-kubernetes

Spring Cloud 和Kubernets 都宣称自己是部署和运行微服务的最佳环境，但是他们本质上存在着巨大差别。在本文中，我们将看到两者在实施微服务的各自的优缺点。

背景

最近我读了一篇基于springcloud和docker搭建微服务的好文，作者A. Lukyanchikov。如果你还没读过，你应该读一下，这篇文章讲述如何利用springcloud来创建基于微服务架构的系统。为了构建一个可弹性扩展至成千上万微服务的系统，就必须依赖庞大的构建和运行工具集去集中治理各种服务。通过springcloud可以实现基础架构服务（日志，集群配置，服务发现治理，认证鉴权）和业务功能服务。如下图，描述了一个基于springcloud构造的MSA:

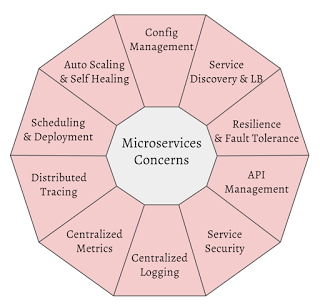


*MSA with Spring Cloud (by A. Lukyanchikov)*

上图覆盖了运行视角下的微服务，但是没有触及打包，持续集成，高可用，自身状态监测等MAS中非常重要的关注点。假设大多数java开发者都熟悉了springcloud,在本文中，我将对kubernets和springcloud涉及到的微服务相关概念进行横向对比。

微服务的概念

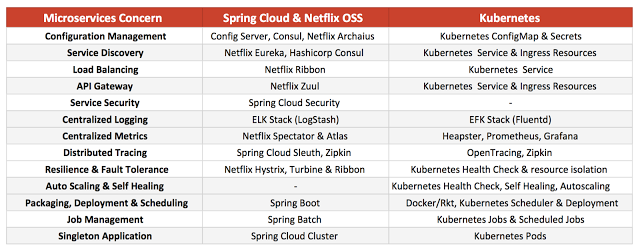
我不会逐个的对比两者之间的特性，只从更高的维度对两者的特性集合进行比较。今天有关MSA的一个好事是，它是一个有着易于理解的优缺点评估的架构风格。微服务能加强模块边界，各模块可以有独立的部署和技术差异。但是相应的也必须使系统分布式化，并且增加了巨大的维护成本（开发和维护成本增加巨大）。一个关键的成功因素是专注于被工具包围，这些工具将帮助您解决尽可能多的MSA问题。能使得启动过程迅速且容易是很重要的，但是通向产品化的旅程是很漫长的，你需要达到这样的高度才能到达那里。



*Microservices concerns*

技术栈

Spring Cloud和 Kubernetes这两个平台有着很大的不同，他们之间的特性都是不可以等价互换的。如果我们将MSA中的每个概念匹配到这两个平台实现的技术上，可得出如下表格：



Spring Cloud and Kubernetes technologies

从上表数据总结如下：

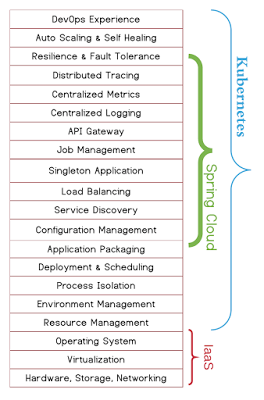
Spring Cloud有一套丰富的集成Java库，以将所有运行时关注点作为应用程序堆栈的一部分。因此，微服务本身拥有库和运行时代理来执行客户端服务发现、负载平衡、配置更新、度量跟踪等。诸如单点集群服务和批处理作业等模式也在JVM中进行管理（亲和与Java）。

Kubernetes 是多语言的，并不只正对Java平台，而是针对所有语言以通用的方式解决分布式计算挑战。它为配置管理、服务发现、负载平衡、跟踪、度量、单例、平台级和应用程序堆栈外部的计划作业提供服务。应用程序不需要任何库或代理来进行客户端逻辑，它可以用任何语言编写。

在某些领域，两个平台都依赖类似的第三方工具。例如，ELK 和EFK堆栈、跟踪库等。一些库（如hystrix和spring boot）在这两个环境中同样有用。这两个平台都是互补的，可以组合在一起以创建更强大的解决方案（Kubeflix：https://github.com/fabric8io/kubeflix和SpringCloud Kubernetes：<https://github.com/fabric8io/spring-cloud-kubernetes就是这样的例子）。>

为服务的需求

为了展示每个项目的范围，这里有一个表，其中包含（几乎）端到端的MSA需求，从底层的硬件开始，到顶层的DevOps和自助服务体验，以及它与Spring Cloud和Kubernetes平台的关系。



*Microservices requirements*

在某些情况下，两个项目使用不同的方法来处理相同的需求，在某些领域，一个项目可能比另一个更强大。但也有一个优势，即两个平台相互补充，可以结合在一起以获得卓越的微服务体验。例如，SpringBoot为构建单个JAR应用程序包提供了Maven插件。这与Docker和Kubernetes的声明性部署和调度功能相结合，使得运行微服务变得轻而易举。同样，SpringCloud在应用程序库中也有使用Hystrix（带有限流和断路器模式）和Ribbon（用于负载平衡）创建弹性、容错微服务的功能。但仅仅这一点还不够，当它与Kubernetes的健康检查、流程重启和自动伸缩功能结合在一起时，微服务就变成了一个真正的反敏捷系统。

优缺点比较

由于这两个平台不能直接按特性进行比较，而且不能深入到每一个项目中，下面是（总结）每个平台的优点和缺点。

Spring Cloud

优点：

A, Spring平台本身提供的统一编程模型和Spring引导的快速应用创建能力，给开发人员提供了很好的微服务开发经验。例如，使用很少的注释，您可以创建一个配置服务器，使用很少的注释，您可以让客户机库配置您的服务(开箱即用)。

B, 有大量的库可供选择，涵盖大多数运行时问题。由于所有的库都是用Java编写的，所以它提供了多个特性、更大的控制和微调选项。

C, 不同的SpringCloud库彼此集成得很好。例如，一个外部客户机还将使用hystrix来断开电路，使用功能区来处理负载平衡请求。一切都是注释驱动的，使Java开发人员易于开发。

缺点：

A, Spring云的一个主要优点也是它的缺点——它仅限于Java。MSA的一个强大动力是在需要时能够交换技术堆栈、库甚至语言。SpringCloud是不可能的。如果您想要使用Spring Cloud/Netflix OSS基础设施服务，例如配置管理、服务发现或负载平衡，那么这个解决方案就不太好。Netflix PrANA项目实现了SIDACAR模式，以通过HTTP公开基于Java的客户端库，以使以非JVM语言编写的应用程序能够存在于NetflixOS生态系统中，但它并不十分优雅。

B, 对于Java开发人员来说，更多关心的是用Java应用程序来处理。每个微服务都需要运行不同的客户机来进行配置检索、服务发现和负载平衡。设置这些是很容易的，但这并不隐藏对环境的构建时间和运行时依赖性。例如，开发人员可以使用@EnableConfigServer创建一个配置服务器，但这只是一条快速简单的路径。每当开发人员想要运行单个微服务时，他们都需要启动并运行配置服务器。对于受控环境，开发人员必须考虑使配置服务器高度可用，并且由于它可以由Git或SVN支持，因此他们需要一个共享的文件系统来支持它。同样，对于服务发现，开发人员需要首先启动Eureka服务器。对于受控环境，他们需要在每个AZ上集群多个实例。这就好像Java开发人员除了实现所有功能服务之外，还要构建和管理一个非平凡的微服务平台（对服务水平伸缩，动态部署支持有限）。

C, Spring Cloud在微服务的发展历程中的作用范围比较狭窄，开发人员还需要考虑自动化部署、调度、资源管理、流程隔离、自我修复、构建管道等，以获得完整的微服务体验。对于这一点，我认为单拿SpringCloud和kubernetes比较是不公平的，更公平的比较应该是春天的云+Cloud Foundry（或docker swarm）和kubernetes。但这也意味着，要获得完整的端到端微服务体验，必须使用类似kubernetes本身的应用程序平台来补充SpringCloud。

Kubernetes

Kubernetes是一个开源系统，用于自动化容器化应用程序的部署、扩展和管理。它是语言平台无关的，为分布式系统的供应、运行、扩展和管理提供了原语。

优点：

A, Kubernetes是一个多语言和语言无关的容器管理平台，能够运行云本地和传统的容器化应用程序。它提供的服务，如配置管理、服务发现、负载平衡、度量收集和日志聚合，可以被各种语言使用。这允许组织中的一个平台可以由多个团队使用（包括使用Spring的Java开发人员），并有多种用途：应用程序开发、测试环境、构建环境（运行源代码管理系统、构建服务器、私有仓库）等。

B, 与SpringCloud相比，Kubernetes解决了更广泛的MSA问题。除了提供运行时服务，Kubernetes还允许您提供环境、设置资源约束、RBAC、管理应用程序生命周期、启用自动缩放和自我修复（其行为几乎类似于反敏捷平台）。

C, Kubernetes技术基于谷歌15年的研发和管理的经验。此外，它拥有近1000名提交者，是GitHub上最活跃的开源社区之一。

缺点:

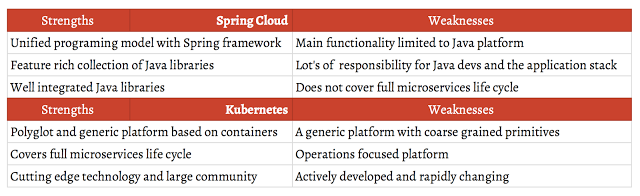
A, kubernetes是多语言无关平台，因此，它的服务和原语是通用的，不针对不同的平台（如用于JVM的SpringCloud）进行优化。例如，配置作为环境变量或装入的文件系统传递给应用程序。它没有SpringCloud配置提供的高级配置更新功能。

B, Kubernetes不是一个以开发人员为中心的平台。它旨在供具有开发人员意识的IT人员使用。因此，Java开发人员需要学习一些新的概念，并需要学习解决问题的新方法。尽管使用minikube启动kubernetes的开发实例非常容易，但是手动安装高可用性kubernetes集群还是有很大的操作开销（学习成本高昂）。

C, Kubernetes仍然是一个相对较新的平台（已有2年历史），它仍在积极开发和成长。因此，在每一个版本中都添加了许多新特性，这可能很难跟上。好消息是，已经设想了这一点，并且API是可扩展的和向后兼容的（太新，资料少，问题多）。

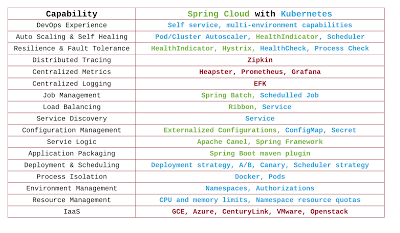
两者结合使用

正如您所看到的，两个平台在某些领域都有优势，在其他领域也有需要改进的地方。SpringCloud是一个快速入门的、开发人员友好的平台，而Kubernetes是DevOps友好的，具有更陡的学习曲线，但涵盖了更广泛的微服务关注。以下是这些要点的摘要。



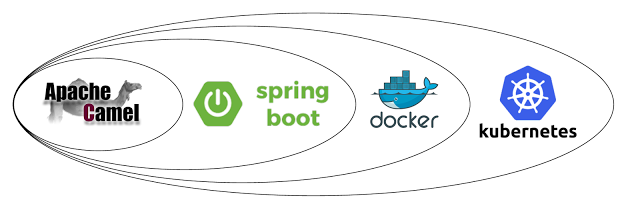
*Strengths and weaknesses*

这两个框架都解决了不同范围的MSA问题，并且它们以一种根本不同的方式来解决问题。SpringCloud方法试图解决JVM中的每一个MSA挑战，而Kubernetes方法则试图通过在平台级别解决问题，使开发人员的问题消失。SpringCloud在JVM内部非常强大，Kubernetes在管理这些JVM方面非常强大。因此，将它们结合起来并从这两个项目的最佳部分中受益，感觉就像是一个自然的过程。



*Spring Cloud backed by Kubernetes*

通过这种组合，Spring提供了应用程序打包，Docker和Kubernetes提供了部署和调度。Spring通过hystrix线程池提供应用程序内的Bulkheading，Kubernetes通过资源、进程和命名空间隔离提供Bulkheading。Spring为每个微服务提供健康端点，Kubernetes执行健康检查和到健康服务的流量路由。Spring将配置外部化并更新，Kubernetes将配置分发给每个微服务。这张单子一直在列。



*My favorite microservices stack*

我最喜欢的微服务平台呢？我都喜欢。我喜欢Spring框架提供的开发人员体验。它都是注释驱动的，并且有涵盖各种功能需求的库。我还喜欢ApacheCamel（而不是Spring集成），它与应用程序级别的集成、连接器、消息传递、路由、弹性和容错性有关。然后，对于与集群和管理多个应用程序实例有关的任何事情，我更喜欢神奇的Kubernetes功能。每当功能重叠时，比如服务发现、负载平衡、配置管理，我都会尝试使用kubernetes提供的polyglot原语。