

27 | Spring Cloud：面向应用层的云架构解决方案

2018-02-18 赵成

赵成的运维体系管理课

[进入课程 >](#)



讲述：黄洲君

时长 09:17 大小 5.32M



上期文章我们介绍了混合云，以及在实操中我们常见的几种混合云模式。今天我们来聊一聊 Spring Cloud 如何解决应用层的云架构问题。

对于 Spring Cloud，你大概不会陌生，它跟 Spring 生态中的另一个开源项目 Spring Boot，基本上已经成为国内绝大多数公司向微服务架构转型时的首选开发框架。

Spring Boot 可以支持快速开发单个微服务应用，Spring Cloud 则提供一系列的服务治理框架，比如服务注册、服务发现、动态路由、负载均衡以及熔断等等能力，可以将一个个独立的微服务作为一个整体，进行很好的管理和维护。

从业界实际使用情况和反馈来看，由于两者完美的搭配，Spring Cloud 和 Spring Boot 确实是可以相对较低的技术成本，让开发人员方便快速地搭建起一套分布式应用系统，从

而进行高效的业务开发。

同时，优秀的服务治理能力，也为其后续在稳定性保障工作方面打下了不错的基础。

（注：因为 Spring Cloud 必须基于 Spring Boot 框架才能发挥它的治理能力，所以下面我们提到的 Spring Cloud 是默认包含了 Spring Boot 框架的。）

所以，通常我们更多地是把 Spring Cloud 作为微服务应用层面的开发框架，帮助我们提升开发效率。看起来，它貌似跟“云”这个概念没有什么直接关系。

而实际上，在将应用与云平台连接方面，Spring Cloud 也发挥着非常核心的作用。这也是为什么本期文章的标题没有直接定义为微服务治理架构，而是面向应用层的云架构。

下面我们具体来看看。

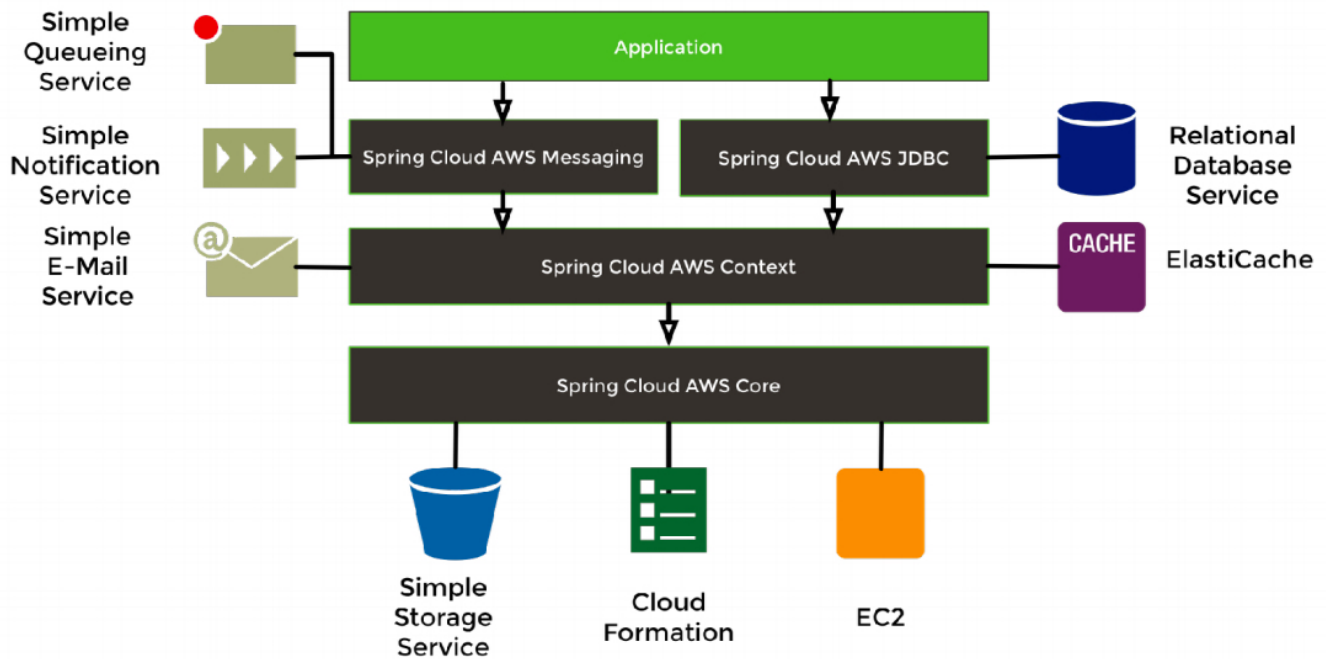
Spring Cloud 框架中云的影子

目前整个 Spring 生态是由 Pivotal 这家商业公司在主导，但是 Pivotal 更大的目标是要为客户提供云上的端到端的解决方案。

所以 Pivotal 最早提出了 Cloud-Native（云原生）的概念，或者说是一种理念，**目的是帮助企业提供云上业务端到端的技术解决方案，全面提升软件交付效率，降低运维成本**。简单来说，就是除了业务解决方案和代码，其它事情都可以交给平台处理。

基于这样的理念，Pivotal 打造了自己的云原生解决方案 PCF（Pivotal Cloud Foundry），包括多云和跨云平台的管理、监控、发布，以及基础的 DB、缓存和消息队列等等，一应俱全。

我们可以看到，在 PCF 整体解决方案中，Spring 生态是向用户的业务应用层架构拓展的非常重要的一环，帮助其进行高效的业务开发，并提供后续的稳定性保障。



所以，这个时候，**Spring Cloud** 除了提供微服务治理能力之外，还成为了微服务应用与云平台上各项基础设施和基础服务之间的纽带，并在其中起到了承上启下的关键作用。

至此，我们可以得出这样一个判断，也是本篇文章想传递的一个信息：**Spring Cloud 不仅仅是微服务治理解决方案，它同时还是面向应用层的云架构解决方案。**

虽然 Pivotal 最早提出了云原生的理念，也提供了 PCF 这样的云原生整体商业解决方案，但是从目前业界的实际应用情况来看，Spring Cloud 这个局部解决方案的应用更为广泛。

而且从图中我们看到，其与 AWS 的深度整合，也正反映出当前 Spring Cloud 在整个业界的影响力和被应用的广泛程度。

插句题外话。早期阿里开源的 Dubbo，其实是跟 Spring Cloud 类似的微服务框架，并且经过阿里大规模的应用实践，可以说是非常优秀的开源项目。早些年国内在选择微服务框架时，Dubbo 基本是首选，但是近年来因为开源维护不力，很早停止了版本更新，导致大量的用户流失，促使用户纷纷涌入 Spring Cloud 阵营。

而 Spring Cloud 经过近几年的发展，深入了解用户需求和痛点，不断完善改进，早已蜕变成我们所说的应用层的云架构，紧跟整个云计算发展趋势的大潮。

最近 Dubbo 重启开源维护，与阿里云 EDAS 产品体系整合，很大原因就是因为在用户技术架构体系里，缺少了 Spring Cloud 这样的产品，再加上 Dubbo 原有的一些用户基础，

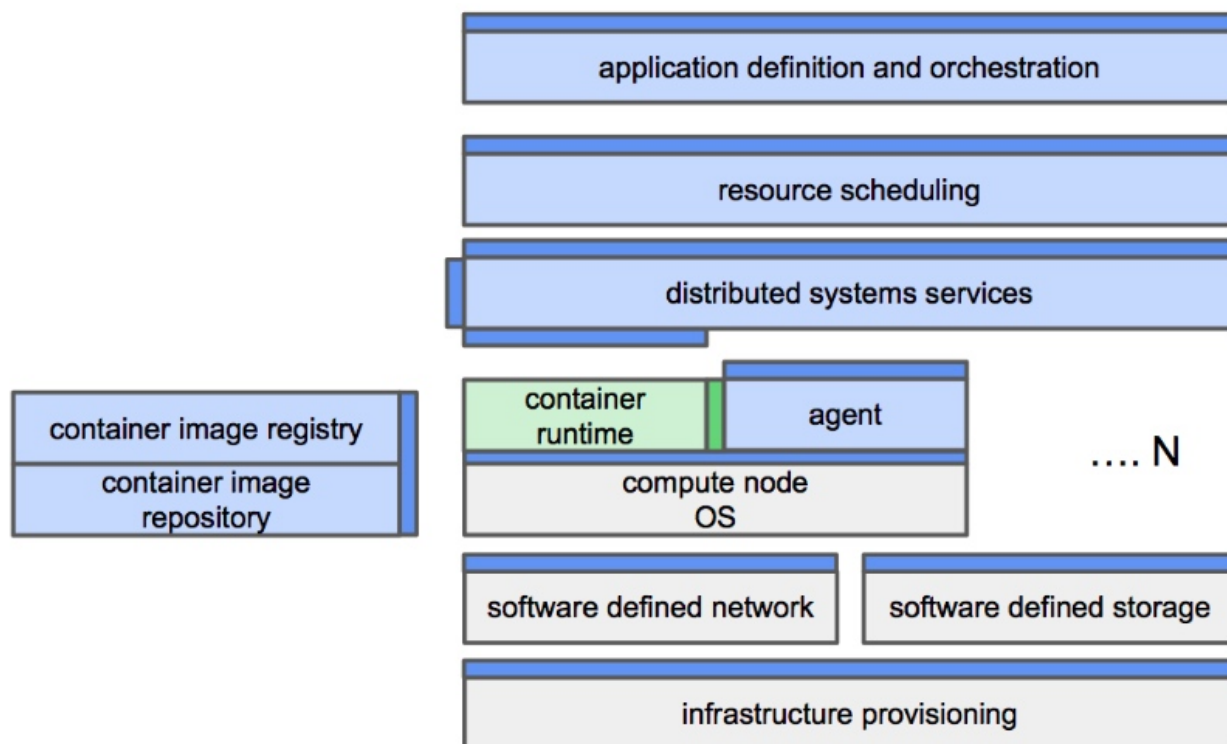
重启维护无论从哪方面看都是值得的。但是需要多久才能重拾用户的信心，就要看 Dubbo 的后续表现了。

以 Spring Cloud 为代表的云原生模式也是当前业界的主流模式。虽然它可能以解决应用层面的问题为主，尚未与云平台全面对接整合，不过它所带动起来的云原生的理念却被业界越来越广泛地接受。

同时，随着容器及编排技术的发展和成熟，就出现了另外一个云原生的体系，且活跃程度非常高：它就是以 Google 为首的 CNCF（Cloud Native Computing Foundation）。下面我们一起看一下。

CNCF

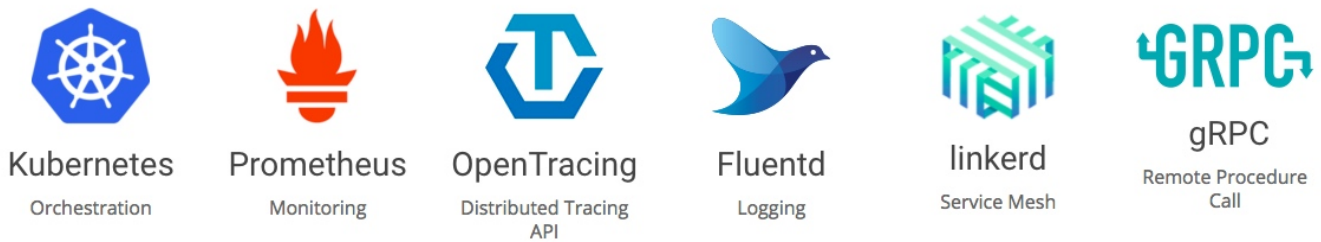
CNCF 设想中的云原生分层架构示意图：



CNCF 组织成立后，圈中大佬们纷纷加入，比如 AWS、微软、思科、Pivotal 等等，国内的腾讯云、阿里云和华为也参与其中，可见其影响力有多大。

CNCF 的核心项目除了 K8S 外，还有 Goggle 的 gRPC，Docker 的 ContainerD，CoreOS 的 Rkt 等重量级开源项目。同时也有与 Spring Cloud 类似，但更加通用的微服务治理框架，如 Linkerd 和 Envoy，它们被称为 Service Mesh（服务网格）。

这些项目的优势在于，它们是与 K8S 集成和配套的，可以很便捷地应用于 K8S 生态中。虽然 K8S 自身也是支持服务发现、负载均衡这些基本的微服务治理的，但是在 CNCF 中，它显得更加包容与开放，不断吸引业界最佳实践的开源产品加入，共同打造更加开放的生态。下图为 CNCF 当前的项目。



同时，因为目前 K8S 已实际上成为业界容器编排方面的标准，且被广泛应用，所以各大云厂商，无论公有云和私有云，都会主动支持 K8S 在云计算体系中的落地。

因此，我们根本不用担心 K8S 与云平台上 的资源和各种服务的对接问题，而且它最终也会将应用与云平台很好地连接起来，让开发者能够更加专注于业务开发。至于剩下的工作，则都交由平台去做。

当前，CNCF 的各个项目社区非常活跃，以至于我们一提到云原生，就会联想到基于 CNCF 和 K8S 的生态体系。虽然 Google 和 CNCF 都不是云原生的提出者，但目前看来，它们都是云原生的最佳实践者。

可以预见的技术发展趋势

我们可以看到，无论是 Spring Cloud、CNCF、云原生、还是 K8S 等等新技术或理念，究其根本，都是为了能够更快更好地支持业务需求的快速实现。

从云原生的理念中，我们可以看到，跟业务无直接关系且相对通用的技术在不断地被标准化，而且标准化层面越来越高。

从最底层的硬件和网络设备，到上层数据库、缓存、文件存储以及消息队列等等基础组件服务，再到 Spring Cloud 和 Service Mesh 这样的应用层面的服务管理和治理能力，都正变得越来越标准和通用。

技术每被标准化一层，原来繁琐低效的工作就少一些，技术标准化的层面越高，技术门槛就会变得越低。我们可以作个大胆的预想：或许未来真的只会有业务解决方案和业务代码。

对于我们技术人员来说，未来更多更迫切的能力需求将会是：如何利用好业界已有的丰富的技术产品和平台，在面对更加丰富多样且复杂的业务领域需求时，能够更加专注于寻求业界解决方案，以更好地将业务和技术连接起来。找到适合业务解决方案的技术并落地实现，而不再只是专注于技术层面的造轮子。

对于运维来说，我们同样要了解技术发展趋势。虽然我们不会直接参与具体的业务解决方案和代码的开发，但是，如果架构师是业务架构的设计者，那么我们应该成为技术架构的管理者，从效率、成本、稳定性这几个方面来检验架构是否合理，并为架构朝着更加健康的方向发展保驾护航。这也是运维职能转型和思路转变的一个重要方向。

欢迎你留言与我讨论。

如果今天的内容对你有帮助，也欢迎你分享给身边的朋友，我们下期见！




赵成的运维体系管理课

带你直击运维的本质

赵成

美丽联合集团技术
服务经理



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 26 | 为什么混合云是未来云计算的主流形态？

下一篇 28 | 以绝对优势立足：从CDN和云存储来聊聊云生态的崛起

精选留言 (2)

写留言



刘圣威

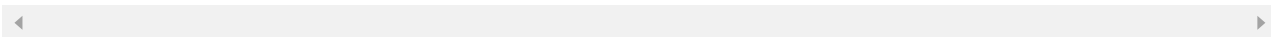
2018-03-05



非常认同您的假设，甚至以后都可以是ai写代码

展开 ▾

作者回复: 这个还要假以时日



casper2dd

2018-02-28



“那么运维应该成为技术架构的管理者，从效率、成本、稳定性这几个方面来检验架构是否合理”这句话能具体举个例子么 因为感觉成熟的解决方案 对运维关心的 成本 效率 稳定性都包括了 比如弹性扩容 故障定位 感觉以后能做的越来越少了

作者回复: 专栏里写的持续交付，和后面的稳定性建设都是实际的案例。

从技术实现角度，解决方案和思路都是很多成熟的东西可借鉴的，但是落地具体业务时是需要做大量适配的，包括后期的技术运营，这个工作只会越来越多。

