云端应用开发

云如何彻底改变了我们设想和开发应用的方式

摘要：

- 平台即服务、软件定义环境的发展以及工具框架等云技术已经真正实现了微服务架构风格。微服务架构是指将单个应用作为一套小型服务来开发，每个小型服务在网络上各自的流程中运行，并通过轻量级机制（通常是利用云功能的 API）相互通信。
- 此架构风格促进了敏捷性，提高了应用的生产率、灾备能力和可扩展性。虽然微服务已经成为有待开发的互动系统应用领域的基本架构风格，但是当应用模式适合此架构风格时，微服务也是转变技术债务的整体企业战略的一部分。这些服务可独立部署和管理，利用多样化多语言编程来实现。
- 这并不适用于具有集中式数据模型、存储库和管理的应用。因此，大多数传统应用和记录系统应用都很难采用微服务架构风格。
- 应用编程接口（API）是子例程定义、协议和工具的集合，也是微服务架构的重要推动力量。它们为在微服务架构的各个组件之间构建接口提供了最合理的模型。通过 API 推动的各个微服务都能够与架构中的其他微服务、API 所支持的应用和网站以及从中提取实时信息的数据库进行通信，这是提供微服务固有能力的基础。

利用微服务在云端构建应用

将单个应用作为一套小型服务来开发，每个小型服务都在各自的流程中运行，并通过轻量级机制（通常为 HTTP 资源的 API）相互通信，这种架构模式即称为微服务，它是云端应用的主要架构。这些服务在逻辑上相互独立，由多个团队构建，专注于用户体验、业务功能，通常实现体验自动化，并且一天内支持多项部署。

传统的应用架构由 n 层组成，通常分布在展示层、业务逻辑层和数据持久层之间。这些数据层使用相同的编程语言进行编码，有时会有损用户体验。此架构是一体式，因为代码更改需要重新构建整个代码库。此外，不可通过业务功能/用户体验来选择性地扩展应用。一段时间过后，应用往往会失去模块化特性。这些失去模块化的应用最终形成了一种架构风格，即将应用作为一套服务来构建。这些服务可独立部署和管理，利用多样化多语言编程来实现。这也促进了分散式治理和团队所有权。微服务的这种性质使它们非常适合在云端开发和部署。

微服务架构风格并没有正式的定义，但可以通过某些共同特征来描述。

微服务：主要特征

- 用户体验驱动的精细化服务，用心做好一件事
- 根据定义使用多种语言
- 跨职能团队
- 在网络上运行
- 使用轻量级机制进行通信（使用 Kafka 等服务进行消息传递，使用 JSON 直接调用）
- 分散式治理
- 数据与微服务耦合
- 快速配置、基础架构自动化和集成 DevOps
- 为解决故障而设计，云作为关键推动力量
- 可扩展性和灾备能力
- 安全性

微服务组件始终被定义为可独立更换和升级的软件单元。微服务有各自的用户界面、业务逻辑和持久实现，本质上是一个独立的业务功能，并具有明确的非功能性规范。例如，进行付款、完成汽车预订等等。这并不适用于具有集中式数据模型、存储库和管理的应用。因此，大多数传统应用和记录系统应用都很难采用微服务架构风格。这有助于创建独立的无状态服务，通过轻量级 API 调用相互交互或与其他记录系统交互。例如，微服务通过呈现 Web 服务的适配器与基于大型机的应用交互。为支持更好的用户体验，微服务中的架构层可以使用不同的编程语言来实现，主要是利用 JavaScript、python 等时间脚本语言，而非阻碍敏捷性的编译语言。

微服务是用户体验驱动的服务，基于业务能力的功能组件化和多语言性质有助于创建自包含的独立可执行文件，这些可执行文件使用易集成的 DevOps 工具进行构建、部署和检测。它们是由多个独立但相互协作的小组共同开发。这些小组有自己的发布周期，通常每天发布多次。例如，一个在线购物网站有近 450 个微服务，每位开发人员约 3 个。

基于微服务的方法创建了一个连续统一体，同一团队可在生产过程中对其进行构建，并加以运行和支持。这使开发人员更接近业务，因而促进他们思考软件如何增强业务能力，这就引发了“产品”思维。

传统的集成方法将重要的智能性特质融入到通信机制本身（例如，企业服务总线等）。基于微服务的应用往往是分离的，并且尽可能地具有凝聚力。

传统的一体化应用需要集中管理组件的构建方式等。多语言方法有助于分散开发，团队可以选择适当的框架/工具来开发服务。微服务框架和 PaaS 平台服务（包括 DevOps）有助于分散治理。这也有助于分开制定数据存储决策，并简化数据模型设计，避免了服务间的交易协调，通过补偿交易的方式处理问题。这是企业管理流程的方式 - 确保致力于快速响应，能够通过快速撤销来修复错误。

PaaS 平台、容器化和软件定义环境是自动化和适当级别仪表化的核心要素。通过自动化整个流程，减少了构建、测试（已自动化）、部署及运作微服务的复杂性。仪表化包括故障检测、通过复杂监控恢复服务、事件管理，以及提供固有能力的云技术。

单个功能可扩展性是微服务架构的核心。这反过来需要最好以水平方式上下扩展的弹性基础设施，以满足应用要求。微服务通常跨越混合 IT 环境的边界，涵盖本地云、私有云、公共云和平台/软件即服务。这就要求支持较新的安全机制，如 OAuth、SAML 和基于云的身份验证/授权。

API 和微服务

微服务以业务为中心，而 API 则体现了向消费者展示功能的方式。在云和现代 Web 架构的大环境下，API 通常是以 REST (JSON/HTTP) 形式实现的接口。API 通常是互动系统应用所用微服务的实现。API 较少关注复用，而更多关注消费和货币化。API 也可作为网络服务或其他机制来实现。API 既可在企业内（私有）也可在企业间（公共或合作伙伴）使用。

基于微服务的应用通过 API 网关/结构访问各个服务，该网关/结构是所有客户端的单一入口点。API 网关通过以下两种方式之一处理请求。有些请求只是代理/路由到适当的服务。其他请求则通过扇出到多个服务来处理。

例如，Amazon API Gateway 是一种完全托管的服务，可以帮助开发者轻松创建、发布、维护、监控和保护任意规模的 API。只需在 AWS 管理控制台中点击几下，您便可以创建可充当应用“前门”的 API，从后端服务访问数据、业务逻辑或功能，例如基于 Amazon Elastic Compute Cloud (Amazon EC2) 运行的工作负载、基于 AWS Lambda 运行的代码或任意 Web 应用。Amazon API Gateway 负责管理所有任务，涉及接受和处理成千上万个并发 API 调用，包括流量管理、授权和访问控制、监控，以及 API 版本管理。

微服务开发技能

团队秉承跨职能、独立执行、非常敏捷和面向 DevOps 的运营文化及组织方式，对于有效的概念化以及微服务的设计、实现和高效利用至关重要。微服务开发的核心是 DevOps 和基础设施自动化，而服务（组件）可以内置于任何流行的技术（java、.net、python 以及 node.js 等）中。因此，每位微服务开发人员都应熟悉微服务框架（以及相关技术）。仅拥有核心开发语言技能没有多大用处，因为

关键范例

- 某大型银行正在根据 Kubernetes 和 Docker 利用基于 Spring Cloud 的框架在私有云上构建微服务。这通过基于企业服务总线的中间件与记录系统应用相集成。
- 某大型电信公司正在建立一个基于内部 Cloud Foundry 的结构，力求标准化和简化基于微服务的开发。
- 某大型电梯制造商正在 Bluemix 容器中构建微服务，这些微服务与他们的 ERP 系统以及 IBM 的物联网解决方案相连接，该方案为大型房地产公司提供 API，为他们的乡镇/校园构建集成控制解决方案。

构建团队本质上不仅仅是编写代码，而且还将故障恢复、监控、日志记录以及其他所有功能内置到他们开发的微服务中。测试是自动化的，并通过正确的自动化工具集（chai/mocha/selenium 都是相关的工具集示例）内置到开发框架中。

微服务开发与 DevOps 和微服务框架以及底层基础架构自动化框架紧密相关。对于从事微服务的团队来说，了解云概念、PaaS 框架、12 要素设计原则、核心微服务框架相关技术和服务等至关重要。微服务小组通常由解决方案架构师、云平台架构师、微服务专家、混合云平台/PaaS 平台专家、DevOps 顾问，以及最重要的敏捷顾问组成。

总结

微服务是云端应用开发的基础，也是实现全套云功能的核心。

微服务架构风格有助于独立开发和部署服务，而底层工具框架 (DevOps) 则可实现敏捷性、组件化、分散部署并促进故障解决方法的设计。微服务开发需要一套能够实现架构风格的技术框架。帮助构建微服务的框架随附必要的库和工具。

这些框架的底层是基于 Cloud Foundry 的环境、基于 Docker 或容器的环境，或者具有重要软件定义环境级别的虚拟化环境。

因此，利用云概念和框架不仅对于微服务的开发至关重要，而且也是必不可少的。

如有任何疑问，请通过以下方式联系 IBM，我们会为您提供更专业的咨询：

1. 免费咨询电话：**400-810-1818 转 2396（服务时间：9:00-17:00）**
2. 填写需求，提交至 IBM，我们会尽快与您取得联系。

IBM Corporation
Global Process Services
Route 100
Somers, NY 10589

美国出品
2017 年 6 月

XXX00000-CNZH-00