链滴

微服务架构（一）- 概述

作者：guobing
原文链接：https://ld246.com/article/1496467642622
来源网站：链滴

<blockquote>
微服务架构是近些年来在软件架构领域出现的一个新名词，虽然其诞生时间不长，但是其在各种术资料，演讲、文章中出现的频率让大家意识到了它对软件架构领域带来的影响。
</blockquote>
<h2 id="1--什么是微服务架构">1. 什么是微服务架构</h2>
对于微服务很难有一个确切的定义，就像对于 <code>NoSql</code> 和 <code>函数式编程<code> 的概念一样，大家都有一个大致的认识，但是下个准确的定义比较难。
摘录一段 <code>马丁- 福勒</code> 先生的话：
<blockquote>
微服务架构是一种架构模式，它提倡将单一应用划分成一组小的服务。服务之间互相协调，互相合，为用户提供最终价值。每个服务运行在独立的进程中。服务于服务之间才用轻量级的通信机制互沟通。每个服务都围绕着具体业务构建，能够被独立的部署。
</blockquote>
<h2 id="1-1-多微才算微">1.1 多微才算微</h2>
两种错误的认识：
<blockquote>

按代码行数判断。 实现相同的功能，不同的语言代码量不同
按重写时间判断。成员工作经验、技术背景、熟悉技术栈的不同，需要的开发时间也不同

</blockquote>
正确的认识：
<code>团队觉得好才是真的好</code>
那一般也遵循两个原则
<blockquote>

业务独立性。要保证微服务是一个具有业务独立性的单元
团队自主性。考虑到沟通成本，统一服务开发人数不建议超过 10 人。当人数过多时，考虑再拆服务。

</blockquote>
<h2 id="1-2-单一职责">1.2 单一职责</h2>
每个微服务，应该处理单一的业务逻辑，在软件架构层面遵循单一职责原则。符合高内聚，低耦原则。
<h2 id="1-3-轻量级通信">1.3 轻量级通信</h2>
对于微服务而言，应该使用与平台无关，与语言无关的通信轻量级通信机制，使服务与服务之间协作更加标准化。
<h2 id="1-4-独立性">1.4 独立性</h2>
指在应用交付的过程中，开发测试部署的独立性。每个服务都是一个独立的业务单元。有独立的码库，有独立的测试机制。
<h2 id="1-5-进程隔离">1.5 进程隔离</h2>
通常情况下，每个服务都运行在一个独立的操作系统进程中
<h2 id="2--微服务诞生背景">2. 微服务诞生背景</h2>
<blockquote>

互联网行业的快速发展。随着互联网的高速发展，对互联网软件的架构提出了新的挑战，如何构高可用，可伸缩，可复用，可扩展的系统架构成为了新的主题

</blockquote>
<blockquote>

敏捷、精益方法论的深入人心

</blockquote>

<blockquote>

单体架构面临的挑战。单体应用架构存在明显的短板，不能适应互联网告诉发展的时代需求

</blockquote>
<blockquote>

容器虚拟化技术。Docker 的出现，有效的解决了微服务的环境搭建，部署，以及运维成本高的题，为微服务朝大规模应用起到了推波助澜的作用。使用 Docker，能更快速的交付和部署，能更轻的迁移和扩展。能更简单的管理。

</blockquote>
<h2 id="3--微服务架构与SOA">3. 微服务架构与 SOA</h2>
早在 1996 年，Gartner 就提出了面向服务架构（SOA），SOA 阐述了“对于复杂的企业 IT 系，应按照不同，可重用的粒度划分，将功能相关的一组功能提供者组织在一起为消费者提供服务，其的是为了解决企业内部 IT 资源之间不能互联的信息孤岛问题”
微服务并不是一个全新的概念，其实和 SOA 的思想是一致的。
微服务有这么几个体现
<blockquote>

团队级，自底向上开展实施
一个系统被拆分成多个服务，粒度细
无集中式总线，架构松散
继承方式简单，HTTP/JSON/REST
服务能独立部署

</blockquote>
<h2 id="4--微服务的本质">4. 微服务的本质</h2>
微服务通常包括以下几个部分
<blockquote>

服务作为组件。将应用模块化，并为其构建相对独立的单元。传统实现组件的方式是隔离独立的分或抽取公用的部分，构建共享库，从而达到解耦和共享的目的。不过对于共享库而言，其实语言相的。并且与应用程序在同一个进程中。实际上，微服务也可以看成一种组件，与传统方式组件的最大别是组件可以被独立部署。因此，微服务一个明显的优势就是，能以松散的服务方式，构建可以独立署的模块化应用。组件与组件之间定义了清晰的，与平台无关的、语言无关的接口
围绕业务组织团队。提倡以业务为核心，按照业务能力来组织团队。
关注产品而非项目。提倡采用产品模式构建，让团队负责整个服务的生命周期，从服务的分析、发、测试、部署、运维，所有成员的个人目标和团队目标是一致的。
技术多样性。更容易尝试新的技术和解决方案，快速尝试，对整个项目带来的风险小
业务数据独立。提倡服务自助管理其业务数据
基础设施自动化。对持续交付和部署流水线要求较高，要部署的业务单元也更多，需要更稳定的础设置自动化机制。
演进式架构。总的来说，是要根据业务不断演变

</blockquote>
<h2 id="5--微服务不是银弹">5. 微服务不是银弹</h2>
微服务存在很多挑战，主要有：
<blockquote>

分布式系统的复杂度。主要包括性能、可靠性、异步、数据一致性等方面
运维成本。主要包括配置、部署、监控与告警、日志收集
部署自动化。
DevOps 与组织架构

</blockquote>

服务间的依赖测试
服务间的依赖管理

</blockquote>