# 第一部分 什么是微服务？为什么使用？优点？缺点？

## 1、什么是微服务？

“微服务”最初是由Martin Fowler在2014年写的一篇文章《MicroServices》中提出来的。但是，对于微服务业界没有一个严格统一的定义，Martin Fowler在文中是这样描述的，微服务架构的风格，就是将单一程序开发成一个微服务，每个微服务运行在自己的进程中，并使用轻量级机制通信，通常是HTTP RESTFUL API。这些服务围绕业务能力来划分构建的，并通过完全自动化部署机制来独立部署。这些服务可以使用不同的编程语言，以及不同数据存储技术，以保证最低限度的集中式管理。

根据上面一段话，可以总结出微服务具有如下特点：

* 按业务划分为一个独立运行的程序，即服务单元。
* 服务之间通过HTTP协议相互通信。
* 自动化部署。
* 可以用不同的编程语言。
* 可以用不同的存储技术。
* 服务集中化管理。
* 微服务是一个分布式系统。

那么微服务的“微”到底是怎样界定的呢，一般可以从三个方面界定，一是根据代码量来定义，根据代码的多少判断程序的大小；二是根据开发时间的长短来判断；三是根据业务的大小来划分。

另外微服务需要具备一些功能：

* 服务的注册和发现
* 服务的负载均衡
* 服务的容错
* 服务网关
* 服务配置的统一管理
* 链路追踪
* 实时日志

## 为什么使用？

通常一个项目的产生是以单体应用的形式存在，单体应用就是应用程序的所有功能都打包成一个独立的单元。最终以一个war包或jar包存在，没有外部的任何依赖，连包含DAO、Servie、UI等所有的逻辑。

在初始阶段，它的优点是

便于开发：只需借助IDE的开发、调试功能即可完成

易于测试：只需要通过单元测试或者浏览器即可完成测试

易于部署：打包成单一可执行包，执行即可完成部署。

但随着应用业务的发展和业务复杂度的提高，变得更加复杂，逻辑耦合严重，难以理解，团队开发人员职责不清，部署困难，回归测试成本巨大，交付效率大大降低，因此单体应用存在很多不足，主要体现在以下方面：

* **复杂性高**。业务越来越复杂，单体应用的代码量越来越大，代码的可连续性、可维护性和可扩展性下降，新人接收代码所需的时间成倍增加，业务扩展带来的代价越来越大。
* **伸缩性差**
* **交付效率差**
* **测试难度越来越大**
* **阻碍技术创新**

由此可见，在应用初期，单体应用在成本、开发时间和运维等方面都有明显的优势。但是随着业务量和用户量的增加，它所暴露出来的缺点也显而易见。为了解决这一现状，可以通过采用微服务架构。

因为微服务是一个分布式系统，在分布式系统中，服务之间相互依赖，如果一个服务出现了故障或者是网络延迟，在高并发的情况下，会导致线程阻塞，在很短的时间内该服务的线程资源会消耗殆尽，最终使得该服务不可用。由于服务的相互依赖，可能会导致整个系统的不可用，这就是“雪崩效应”。

## 微服务的优点

* 将一个复杂的业务分解成若干小的服务，每个业务拆分成一个服务，服务的边界明确，将复杂的问题简单化。
* 微服务系统是分布式系统，服务与服务之间没有任何的耦合。随着业务的增加，可以根据业务再拆分服务，具有极强的横向扩展能力。
* 服务与服务之间通过HTTP网络通信协议来通信，单个微服务内部高度耦合，服务与服务之间完全独立，无耦合，因此可以采用任何的开发语言和技术去实现。
* 重写某个服务相当于重写某一个业务的代码，非常简单，单体应用则不然。
* 微服务的每个服务单元都是独立部署的，即独立运行在某个进程里。微服务的修改和部署对其他服务没有影响。
* 微服务在CAP理论中国采用的是AP架构，既有高可用和分区容错的特点。

Consistency：指数据的强一致性。如果写入数据成功，之后读取，读到的都是新写入的数据；如果写入失败，之后读取的都不是写入失败的数据。

Availability：指服务的可用性。

Partition-tolerance：指分区容错。

## 微服务的缺点

* 微服务的复杂度
* 分布式事务
* 服务的划分
* 服务的部署

# 第二部分 微服务的功能

## 1、服务的注册和发现

服务注册是指服务注册中心注册一个服务实例，服务提供者将自己的服务信息（如服务名、IP地址等）告知服务注册中心。

服务发现是指当服务消费者需要消费另外一个服务时，服务注册中心能够告知服务消费者它所要消费服务的实例信息（如服务名、IP地址等）。

通常情况下，一个服务即是服务提供者，也是服务消费者，

一个服务实例注册后，会定时向服务注册中心提供“心跳”，以表明自己还处于可用的状态。当一个服务实例停止向服务注册中心提供心跳一段时间后，服务注册中心会认为该服务实例不可用，会将该服务实例从服务注册列表中剔除。如果这个被剔除掉的服务实例过一段时间后继续向注册中心提供心跳，那么服务注册中心会将该服务实例重新加入服务注册中心的列表中。

Eureka是Spring Cloud体系中最核心、默认的注册中心组件；

Eureka分为服务端Eureka Server和客户端Eureka Client，一个应用注册到Eureka，这个应用既可以是服务端也可以是客户端；

Eureka Server 提供了三个功能，**服务注册**、**提供注册表**、**同步状态**

