# 1.单体应用

## 什么是单体应用

单体应⽤(Monolithic)

功能集中 、 代码中⼼化 、 一个发布包、部署后运⾏在同一进程的应用程序，我们通常称之为单体应用。

单体应用的优势

## 单体应用的优势

1. 易于开发

-现有的大部分工具、应用服务器、框架都是这类单块架构应用程序，容易理解⽽而且为人所熟知

-IDE（集成开发环境，是一种编程软件如：IntelliJ IDEA，eclipse等）能够有效加载并配置单块架构应用程序的依赖，方便开发人员开发、运行、调试

2、易于测试

- 将发布包部署到某一环境，启动该进程，就可以立即开始系统测试或者功能测试

3、易于部署

- 一个部署包，复制其到服务器相应的位置即可

4、易于伸缩

- 新建一个服务器节点，配置好该节点的运行环境，复制软件包到相应的位置，运行

应用程序。负载均衡器能采取分发策略，有效的将请求分发到新创建的节点

## 单体应用面临的挑战

1. 维护成本增加

-随着功能和代码量的增加，单块架构的维护成本在明显增加。

-自动化测试机制不完善的情况下，易导致“修复越多，缺陷越多”的恶性循环

1. 新人培养周期长

-对于新加入团队的成员，需要花更多时间了解熟悉业务、配置环境、熟悉代码

1. 交付周期长

-代码编译、检查，运行测试、构建、功能验证等，反馈周期变长

1. 可伸缩性差

-无法做到按需伸缩，资源利用率低

# 2.微服务架构

微服务——新的建筑风格 --马丁·福勒，2014年3月

## 2.1.什么是微服务架构

微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间互相协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务与服务间采⽤轻量级的通信机制互相协作（通常是基于HTTP协议的RESTful API）。每个服务都围绕着具体业务进行构建，并且能够被独立的部署到生产环境、类生产环境等。另外，应当尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构建。

## 2.2.微服务架构的不是

1、不是传统的SOA（面向服务的架构）

SOA是集成企业级应用/信息孤岛的架构风格

微服务是将应用构建成不同业务的独立服务

2、不是传说的银弹

微服务架构是典型的分布式系统，对开发/测试/部署/运维带来挑战

3、不是新的语言或技术

微服务并不是新发明的语言或者技术，而是基于敏捷/持续交付/DevOPS（开发运维一体化）等演进的一种架构风格

## 2.3.微服务架构的认识误区

1、微服务让开发变得简单

微服务架构并不解决开发的问题，更多强调是一种架构风格，和现有的CD（持续交付）/DevOps（开发运维一体化）实践相辅相成。

2、微服务应该是按 \*\*\* 拆分的

难以找到量化的标准，业务复杂度，团队能力，流程实践，技术选型都是拆分微服务的相关因素

3、微服务与 SOA（面向服务的架构）只差了⼀一个 ESB（企业服务总线）

这只是从服务间通信集中化的方式考虑，没有从扩展性/容错性/持续交付等维度考虑

4、微服务架构是 \*\*\* 部⻔门的事

微服务架构的演进，涉及业务/开发/测试/部署运维，要求不同角色协作，并及时获取反馈。

## 2.4.微服务架构的本质

一组独立交付的业务单元构成的分布式系统

## 2.5.微服务的优势

1、边界性

业务上下文独立、功能耦合度低

1. 独立性

独立部署、按需伸缩

1. 技术多样性

使用合适的语言或工具、使用合适的数据存储

1. 组织优化

康威定律、康威逆定律

## 2.6.微服务技术的挑战

1、分布式系统的复杂度

网络因素（带宽、超时）、数据一致性

2、运维成本

环境配置、部署/监控/告警

3、测试与治理成本

测试策略与集成、服务治理

4、团队文化与协作成本

团队协作、服务全景图