服务架构演进史

- ## 原始分布式时代
  - **背景：**早期计算机资源相对较小，为了突破硬件能力的限制，寻找多台计算机共同协作来支撑同一套软件系统的可行方案。
  - **成果：**网络运算架构NCA（远程服务调用雏形），
    AFS文件系统（分布式文件系统最早实现），
    Kerberos协议（服务认证和访问控制基础性协议）
  - **后续：**制定了分布式技术体系DCE。包含基于NCA的远程服务调用规范DCE/RPC，基于AFS的分布式文件系统规范DCE/DFS等。
  - **缺陷：**强求分布式操作的透明性，即开发人员不必关注
    访问的方法和资源是否远程或本地，使得技术难度极高。
- ## 单体系统时代
  - **优点：**因为更容易开发、部署、测试而获得更好便捷性
  - **缺点：**由于隔离能力缺失，难以阻断错误传播、不便于动态更新、难以技术异构。关键在于当规模变大，交付一个可靠的单体系统变得极具挑战性和困难。
- ## SOA时代
  - **拆分单体的历史架构：**烟囱式架构（信息孤岛）、微内核架构（公共服务、数据资源等集中为核心，其他服务作为插件）、事件驱动架构（通过事件管道进行服务间通信）
  - **特征：**有清晰的软件设计指导原则，采用SOAP作为远程调用协议、使用企业服务总线ESB进行交互等。
  - **缺陷：**过于严格的规范定义带来过度复杂性，过于
    精密的流程和理论需要理解深刻才能驾驭
  - **贡献：**提出服务间的松散耦合、注册发现、治理、隔离、编排等微服务的概念。解决分布式环境中提出的主要技术问题。
- ## 微服务时代
  - **概念：**微服务是一种通过多个小型服务组合来构建
    单个应用的架构风格，服务围绕业务能力而非技术
    标准来构建，服务可以采用不同语言和技术实现。
    服务采用轻量级通信机制和自动化部署。
  - **业务与技术特征**
    - **围绕业务能力构建**
    - **分散治理**
    - **通过服务来实现独立自治的组件**
    - **产品化思维**
    - **数据去中心化**
    - **强终端弱管道**
    - **容错性设计**
    - **演进式设计**
    - **基础设施自动化**
- ## 后服务时代（云原生）
  - **意义：**用虚拟化技术和容器技术来解决分布式架构问题。未来，K8S成为服务端标准运行环境，服务网格成为微服务之间通信交互的主流模式，将通信协议、认证授权等技术问题隔离于程序代码，微服务只关注业务本身的逻辑。
- ## 无服务时代
  - **愿景**
    - 不需要考虑技术组件，后端技术组件现成
    - 不需要考虑如何部署，完全托管至云端
    - 不需要考虑计算能力，由数据中心支撑
    - 不需要考虑运维操作，由云服务商负责
  - **内容**
    - **后端设施：**数据库、消息队列、日志、存储等。运行在云中。
    - **函数：**指业务逻辑代码，无服务中函数即服务
  - **特点：**无服务架构适合于短链接、无状态、适合事件驱动的交互形式。对于具有复杂业务逻辑、依赖服务端状态、响应速度要求较高、需要长链接等特征应用不太合适（无服务按使用量计费，因此函数不会一直以活动状态常驻服务器，不便依赖服务端状态）。