文件编号： D000-

Spring Cloud微服务架构

版本：1.0.0

第1章 微服务架构概述

本章教学目标

- 了解微服务产生背景；
- 了解微服务技术栈；
- 理解微服务定义；
- 理解垂直架构、SOA架构；
- 理解微服务架构、微服务架构图；
- 掌握单体架构和微服务架构；
- 掌握微服务设计原则；
- 掌握微服务架构技术选型

本章教学内容

Neuedu

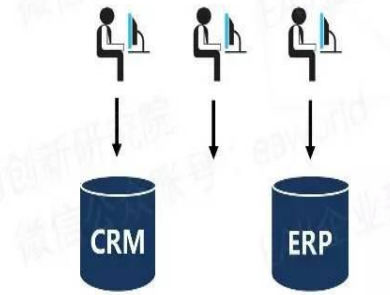
节	知识点	掌握程度	难易程度	教学形式	对应在线微课
微服务简介	微服务产生的背景	了解		线下	
	微服务定义	理解		线下	
软件架构演进	单体架构	掌握		线下	
	垂直架构	理解		线下	
	SOA架构	理解	难	线下	
	微服务架构	掌握		线下	
微服务设计原则	微服务设计原则	掌握		线下	
如何实现微服务架构	架构图及常用组件	理解	难	线下	
	微服务技术栈	了解		线下	
	微服务架构	理解		线下	
	技术选型	掌握		线下	

CONTENTS 目录

微服务产生的背景

Neuedu

- 企业转型



传统应用

- 服务内部用户为主
- 需求明确、功能全，覆盖广，大集成，中央控制，适合稳定发展阶段
- 刚性强，难以快速变化，维护成本高，快速变革的新业态无法支持

新兴互联网应用

- 服务外部客户和合作伙伴
- 需求变动快，功能简单，独立和分散，分布式进化，一切都从零开始，业务与IT无法分开，需要快速创新
- 运用规模变化大，大范围广泛的尝试，易失败（淘汰），对业务弹性、快速发布要求高

传统企业的IT建设需要转型，需要面向外部客户，需要应对外部环境的快速变化、需要快速创新，IT架构也需要向互联网企业学习作出相应的改进，来支撑企业的数字化转型。

先是单体架构，后来为了具备一定的扩展和可靠性，就有了垂直架构，也就是加了个负载均衡，接下来是SOA，解决应用系统之间如何集成和互通，微服务架构则是进一步在探讨一个应用系统该如何设计才能够更好的开发、管理更加灵活高效。

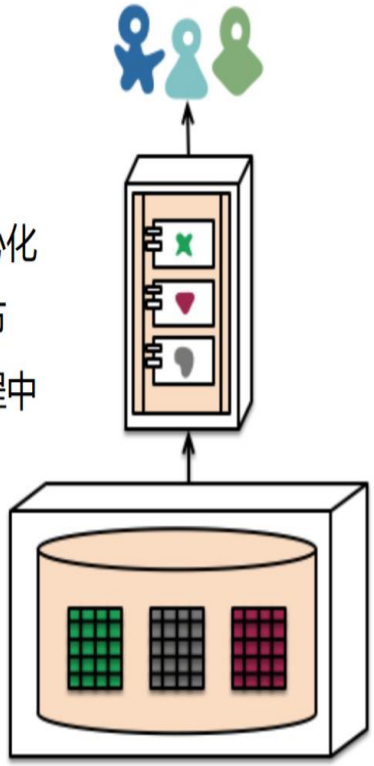
微服务产生的背景

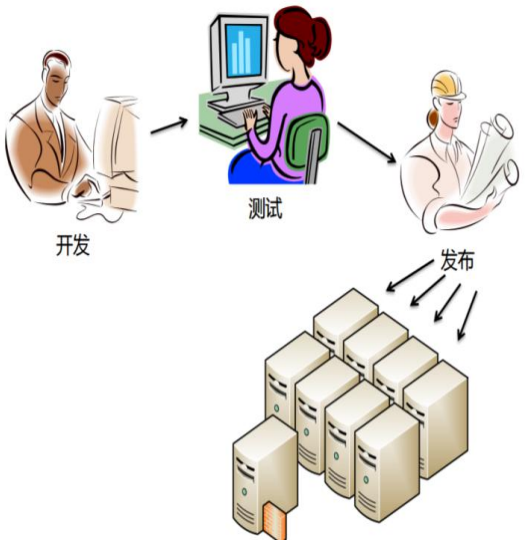
Neuedu

- 问题

单体应用

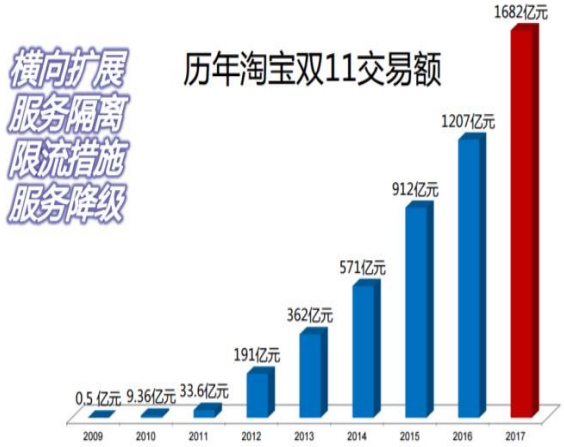
- 功能集中
- 代码与数据中心化
- 在一个包中发布
- 运行在一个进程中





带来的问题

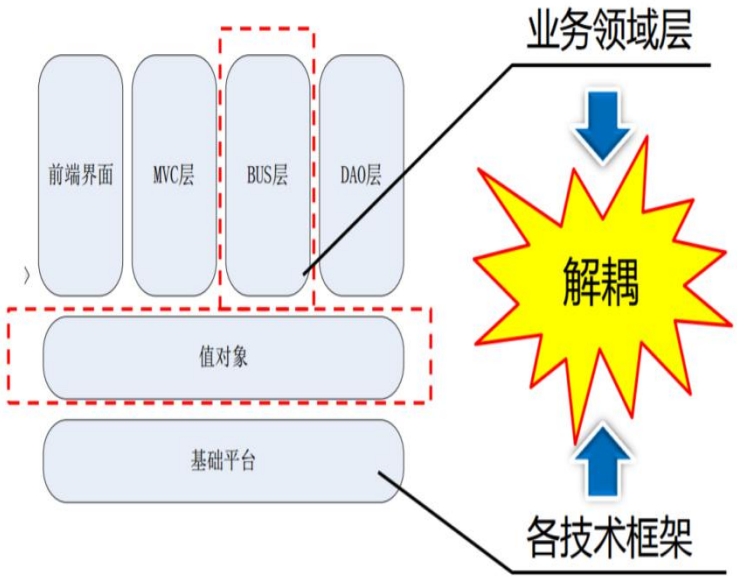
- 开发效率低
- 交付周期长
- 技术转型难
- 新人培养周期长



微服务产生的背景

- 愿景

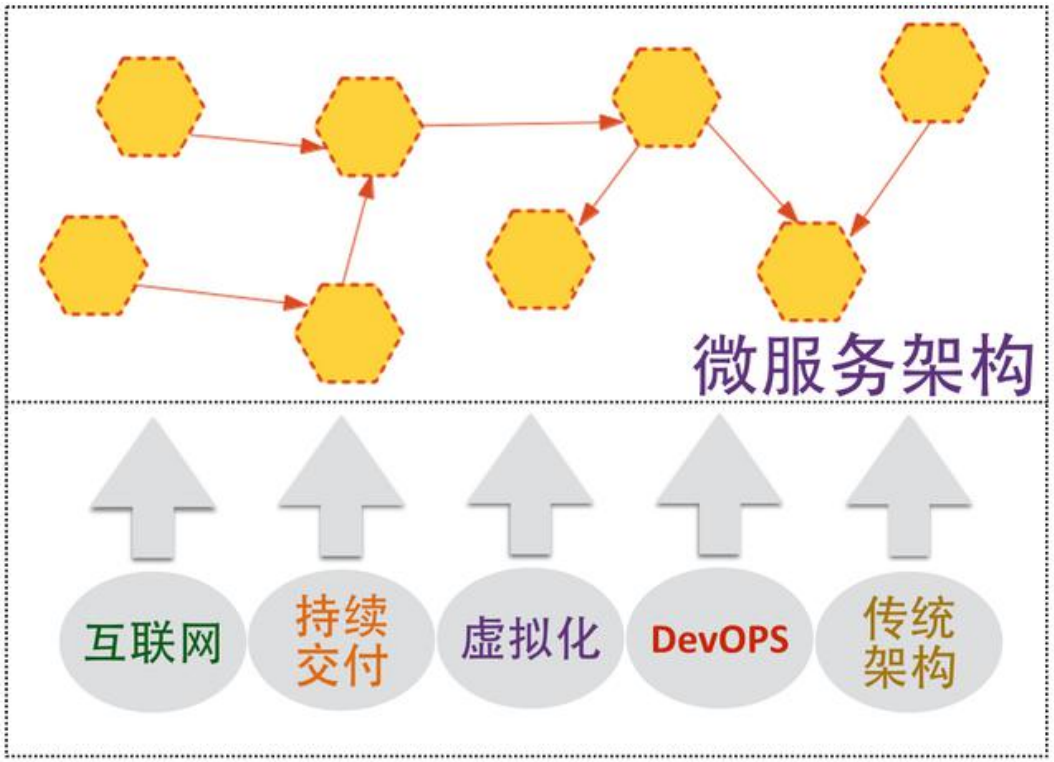
象更换零件一样更换软件



微服务产生的背景

Neuedu

- 微服务的诞生并非偶然。它是互联网高速发展，敏捷、精益、持续交付方法论的深入人心，虚拟化技术与DevOps文化的快速发展以及传统单体架构无法适应快速变化等多重因素的推动下所诞生的产物。



微服务定义

Neuedu

- 实际上，从业界的讨论来看,微服务本身并没有一个严格的定义。不过，ThoughtWorks的首席科学家，martinfowler（马丁-福勒）先生对微服务的这段描述，似乎更加具体、贴切，通俗易懂：
- Microservice
  - The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

微服务定义

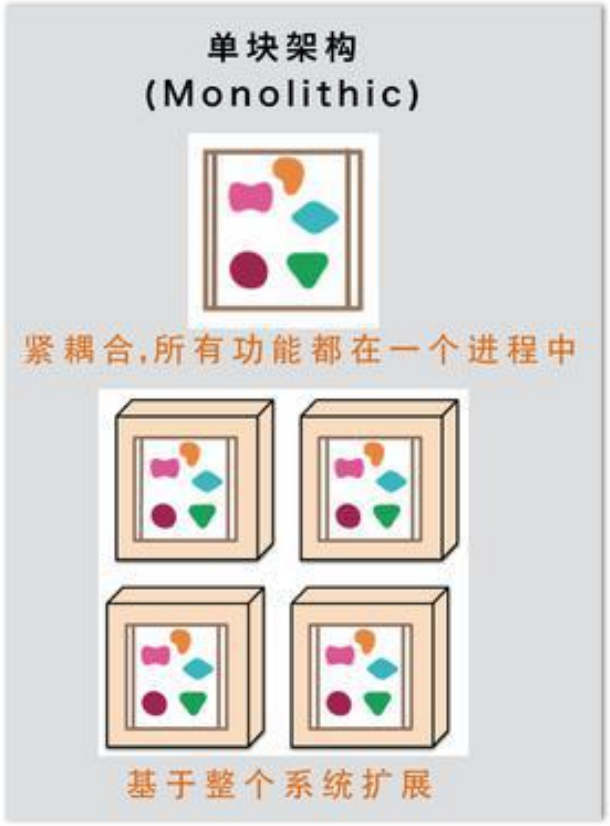
- 英文翻译后如下：
- 微服务架构
  - 微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间互相协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务与服务间采用轻量级的通信机制互相沟通（通常是基于HTTP协议的RESTful API）。每个服务都围绕着具体业务进行构建，并且能够被独立的部署到生产环境、类生产环境等。另外，应当尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构建。

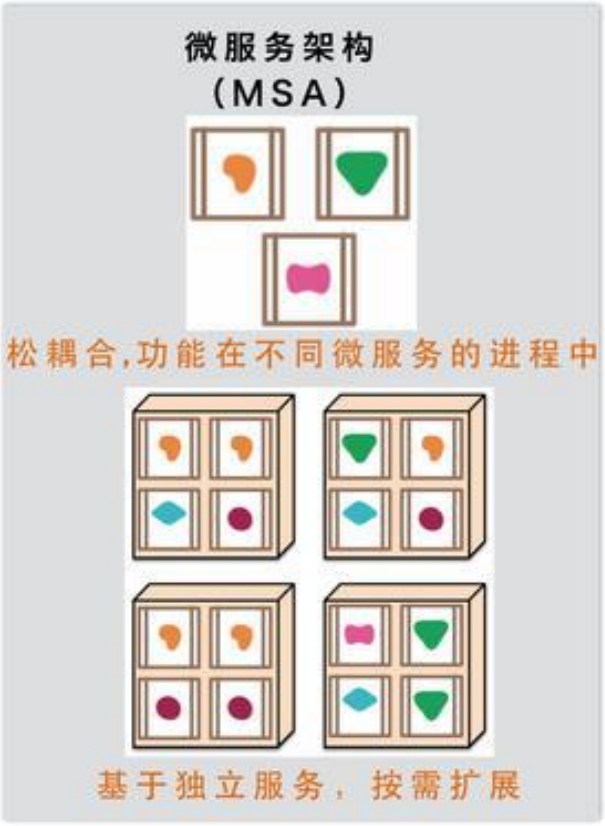
微服务定义

- suite of small services：由一系列小服务组成
- running in its own process： 每个服务运行于自己的独立进
- built around business capabilities：围绕着业务功能进行建模
- independently deployable：每个服务可进行独立部署
- bare minimum of centralized management：最低限度集中管理

微服务定义

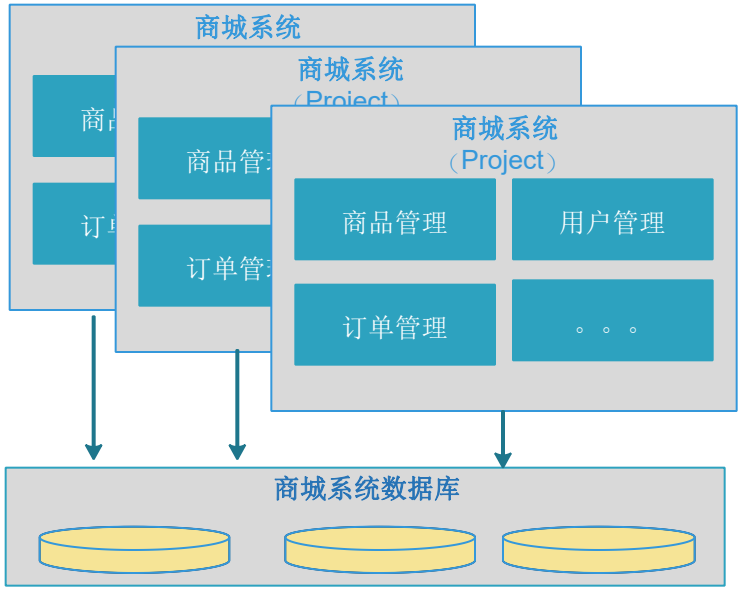
- 将功能分散到各个离散的服务中然后实现对方案的解耦。服务更原子，自治更小，然后高密度部署服务





CONTENTS 目录

单体架构



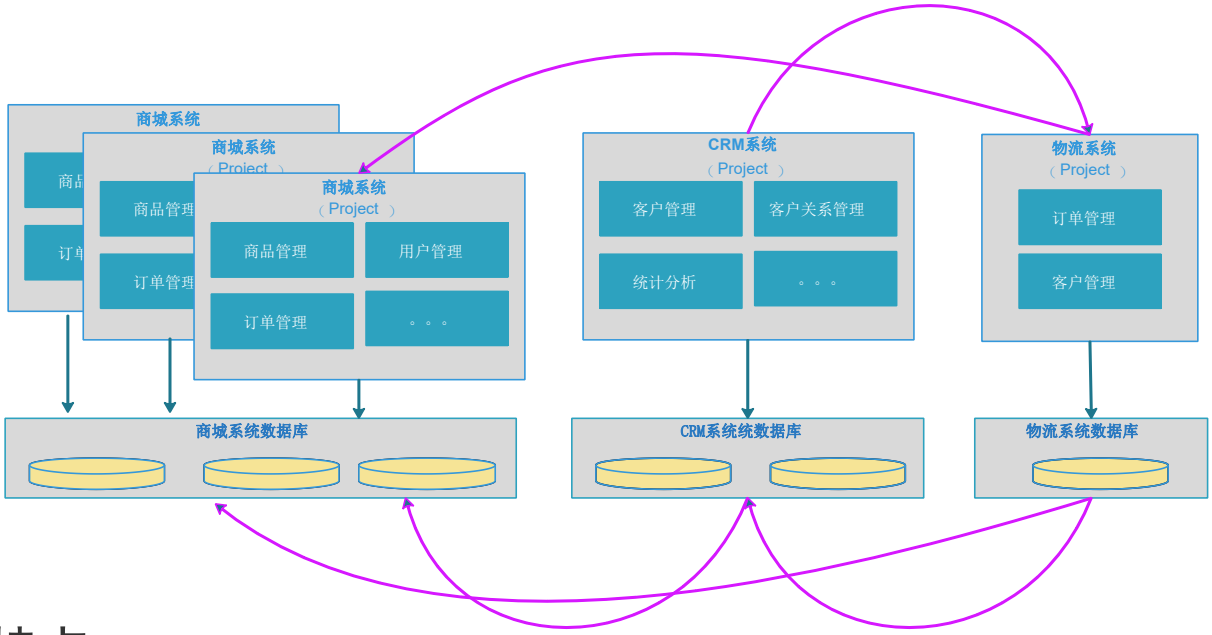
- 特点：
  - 1、所有的功能集成在一个项目工程中。
  - 2、所有的功能打一个war包部署到服务器。
  - 3、应用与数据库分开部署。
  - 4、通过部署应用集群和数据库集群来提高系统的性能。

单体架构

Neuedu

- 优点：
  - 1、项目架构简单，前期开发成本低，周期短，小型项目的首选。
- 缺点：
  - 1、全部功能集成在一个工程中，对于大型项目不易开发、扩展及维护。
  - 2、系统性能扩展只能通过扩展集群结点，成本高、有瓶颈。
  - 3、技术栈受限。

垂直架构



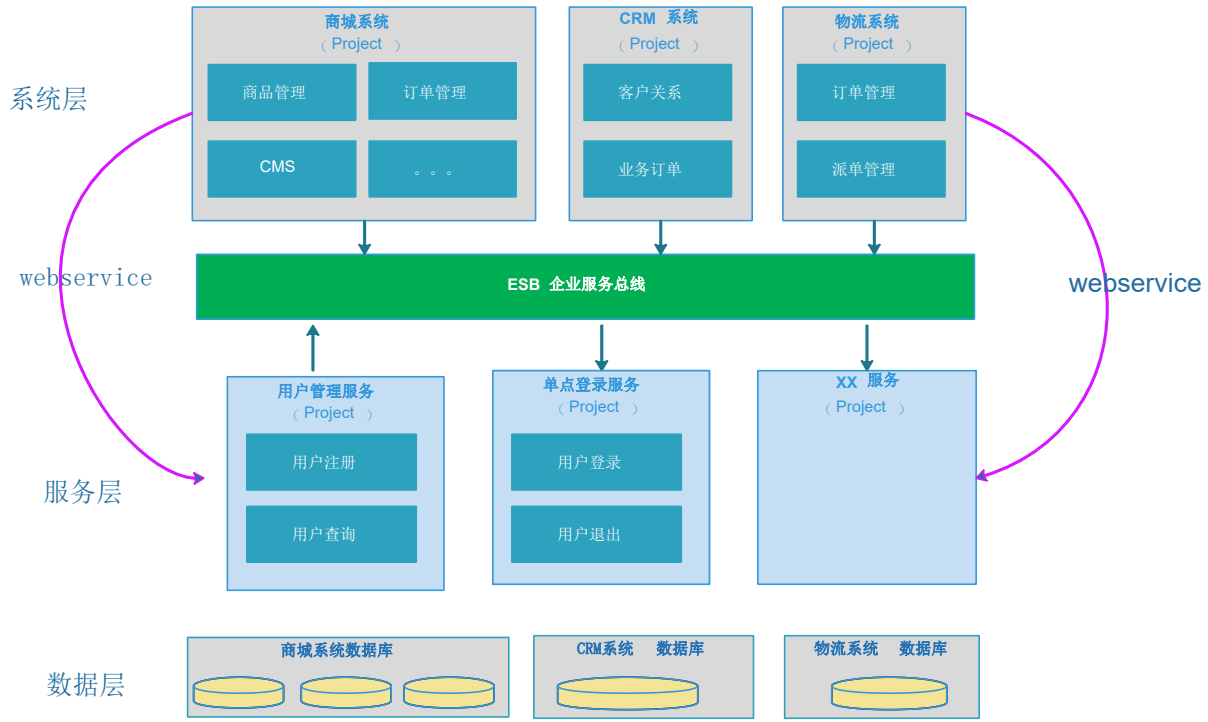
- 特点：
  - 1、以单体结构规模的项目为单位进行垂直划分项目即将一个大项目拆分成一个一个单体结构项目。
  - 2、项目与项目之间的存在数据冗余，耦合性较大，比如上图中三个项目都存在客户信息。
  - 3、项目之间的接口多为数据同步功能，如：数据库之间的数据，通过网络接口进行数据库同步。

垂直架构

Neuedu

- 优点：
  - 1、项目架构简单，前期开发成本低，周期短，小型项目的首选。
  - 2、通过垂直拆分，原来的单体项目不至于无限扩大。
  - 3、不同的项目可采用不同的技术。
- 缺点：
  - 1、全部功能集成在一个工程中，对于大型项目不易开发、扩展及维护。
  - 2、系统性能扩展只能通过扩展集群结点，成本高、有瓶颈。

SOA架构

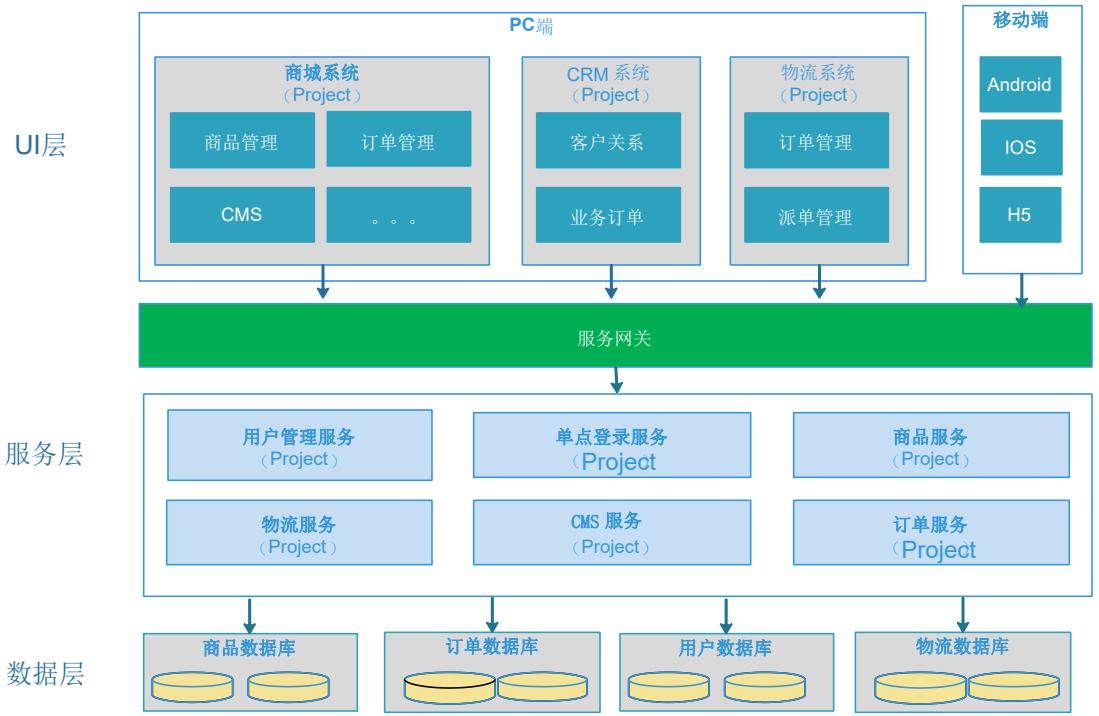


- 特点：
  - 1、基于SOA的架构思想将重复公用的功能抽取为**组件**，以服务的方式给各各系统提供服务。
  - 2、各各项目（系统）与服务之间采用webservice、rpc等方式进行通信。
  - 3、ESB企业服务总线作为项目与服务之间通信的桥梁。

SOA架构

- 优点：
  - 1、将重复的功能抽取为服务，提高开发效率，提高系统的可重用性、可维护性。
  - 2、可以针对不同服务的特点制定集群及优化方案。
  - 3、采用ESB减少系统中的接口耦合。
- 缺点：
  - 1、系统与服务的界限模糊，不利于开发及维护。
  - 2、虽然使用了ESB，但是服务的接口协议不固定，种类繁多，不利于系统维护。
  - 3、抽取的服务的粒度过大，系统与服务之间耦合性高。

微服务架构



- 特点：
  - 1、将系统服务层完全独立出来，并将服务层抽取为一个一个的微服务。
  - 2、微服务遵循单一原则。
  - 3、微服务之间采用RESTful等轻量协议传输。

微服务架构

Neuedu

- 优点：
  - 1、服务拆分粒度更细，有利于资源重复利用，提高开发效率。
  - 2、可以更加精准的制定每个服务的优化方案，提高系统可维护性。
  - 3、微服务架构采用去中心化思想，服务之间采用RESTful等轻量协议通信，相比ESB更轻量。
  - 4、适用于互联网时代，产品迭代周期更短。
- 缺点：
  - 1、微服务的复杂度远超单体架构，开发人员需要学习更多的架构知识和框架知识，微服务测试成本高。
  - 2、分布式系统开发的技术成本高（容错、分布式事务等），对团队挑战大。
  - 服务的划分，根据具体业务场景来拆分服务，需要依靠团队人员都业务的熟悉程度和理解程度，还要考虑架构冲突、业务扩展、开发风险等因素。
  - 服务的部署，比单体架构复杂得多，涉及底层组件，还需要对服务治理、监控和管理等，治理成本高，不利于系统维护。

课堂练习（5分钟）

- 1、单体架构优缺点？
- 2、SOA架构优缺点？
- 3、微服务架构优缺点？

CONTENTS 目录

微服务设计原则

- 1. 单一职责原则
- 2. 服务自治原则
- 3. 轻量级通信机制
- 4. 微服务粒度

微服务设计原则

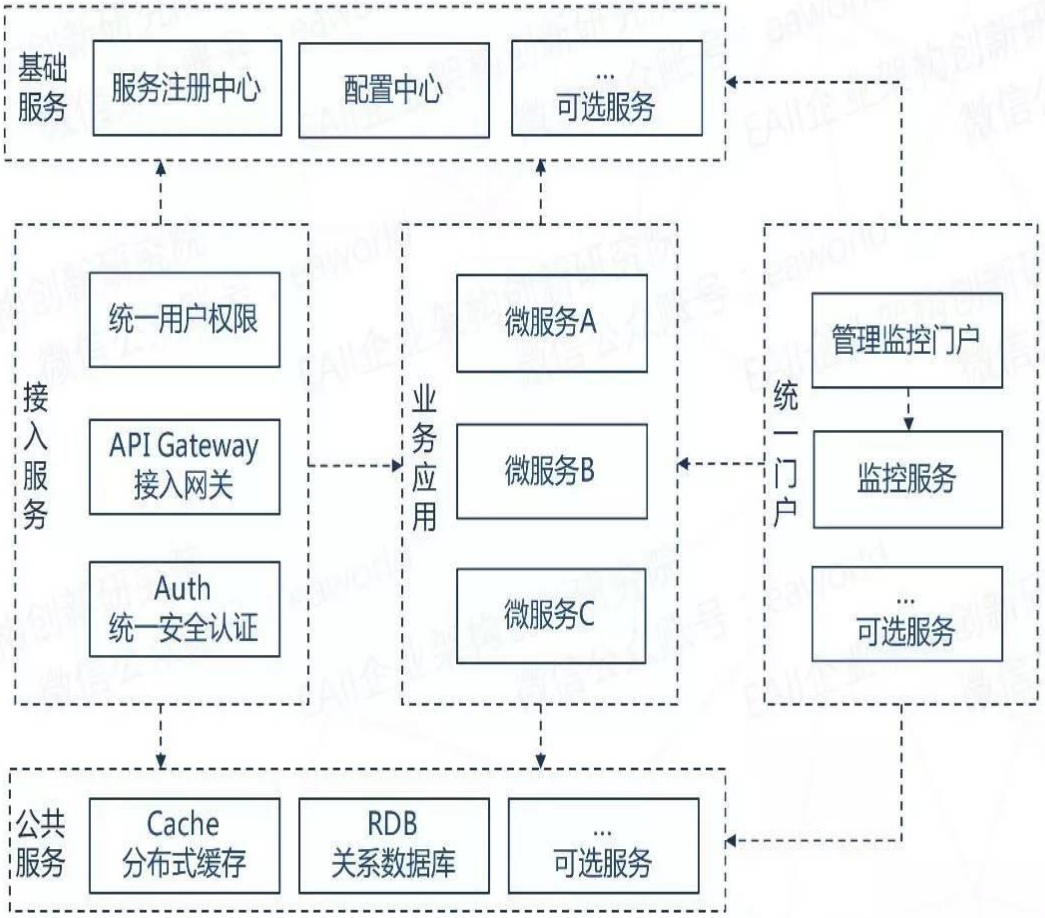
Neuedu

- 1. 单一职责原则
  - 指一个单元只应关注整个系统功能中单独、有界限的一部分。
  - 单一职责原则可以帮助我们更优雅地开发、更敏捷地交付。
- 2. 服务自治原则
  - 指每个微服务应具备独立的业务能力、依赖与运行环境。
  - 在微服务架构中，服务是独立的业务单元，应该与其他服务高度解耦。
  - 每个微服务从开发、测试、构建、部署，都应可以独立运行，而不依赖其他的服务。
- 3. 轻量级通信机制
  - 微服务之间应该通过轻量级通信机制进行交互。
  - 轻量级通信机制应具备两点：一是体量较轻，二是跨语言、跨平台。
- 4. 微服务粒度
  - 应使用合理的粒度划分微服务，而不是一味的把服务做小。
  - 每一个架构师心中的粒度标准是不一样的，所以这块争论最多。

CONTENTS 目录

架构图和常用组件

- 微服务应用平台运行架构



架构图和常用组件

Neuedu

- 常用组件
  - Service A组件
  - 服务发现组件
  - 服务网关
  - 配置服务器
  - 负载均衡
  - 服务接口调用
  - 消息队列
  - …

微服务技术栈

Neuedu

- 微服务技术栈：多种技术的集合体

微服务条目	落地技术
服务开发	SpringBoot,Spring,SpringMVC
服务配置与管理	Netflix公司的Archaius、阿里的Diamond等
服务注册与发现	Eureka、Consul、Zookeeper等
服务调用	Rest、RPC、gRPC
服务熔断器	Hystrix、Envoy等
负载均衡	Ribbon、Nginx等
服务接口调用（客户端调用服务的简化工具）	Feign等
消息队列	Kafka、RabbitMQ、ActiveMQ等

微服务技术栈

Neuedu

微服务条目	落地技术
服务配置中心管理	SpringCloudConfig、Chef等
服务路由（API网关）	Zuul等
服务监控	Zabbix、Nagios、Metrics、Specatator等
全链路追踪	Zipkin、Brave、Dapper等
服务部署	Docker、OpenStack、Kubernetes等
数据流操作开发包	SpringCloud Stream(封装与Redis，Rabbit，Kafka等发送接收消息)
事件消息总线	SpringCloud Bus

微服务架构

- 微服务架构有哪些
  - Spring Cloud
  - 京东JSF
  - 新浪微博Motan
  - 当当网DubboX
  - ......

微服务架构

Neuedu

- 微服务架构对比

功能点/服务框架	备选方案 Netflix/Spring cloud	Motan	gRPC	Thrift	Dubbo/DubboX
功能定位	完整的微服务框架	RPC框架,但整合了ZK或Consul,实现集群环境的基本的服务注册/发现	RPC框架	RPC框架	服务框架
支持Rest	是 Ribbon支持多种可插拔的序列化选择	否	否	否	否
支持RPC	否	是(Hession2)	是	是	是
支持多语言	是(Rest形式)?	否	是	是	否
服务注册/发现	是(Eureka) Eureka服务注册表，Karyon服务端框架支持服务自注册和健康检查	是(zookeeper/consul)	否	否	是
负载均衡	是(服务端zuul+客户端Ribbon) Zuul-服务,动态路由 云端负载均衡Eureka(针对中间层服务器)	是(客户端)	否	否	是(客户端)
配置服务	Netflix Archaius Spring cloud Config Server 集中配置	是(zookeeper提供)	否	否	否
服务调用链监控	是(zuul) Zuul提供边缘服务,API网关	否	否	否	否
高可用/容错	是(服务端Hystrix+客户端Ribbon)	是(客户端)	否	否	是(客户端)
典型应用案例	Netflix	Sina	Google	Facebook	
社区活跃程度	高	一般	高	一般	已经不维护了
学习难度	中等	低	高	高	低
文档丰富度	高	一般	一般	一般	高
其他	Spring Cloud Bus为我们的应用程序带来了更多管理端点	支持降级	Netflix内部在开发集成gRPC	IDL定义	实践的公司比较多

微服务架构

Neuedu

- 微服务中SpringCloud与Dubbo的区别

	Dubbo	Spring
服务注册中心	Zookeeper	Spring Cloud Netflix Eureka
服务调用方式	RPC	REST API
服务监控	Dubbo-monitor	Spring Boot Admin
断路器	不完善	Spring Cloud Netflix Hystrix
服务网关	无	Spring Cloud Netflix Zuul
分布式配置	无	Spring Cloud Config
服务跟踪	无	Spring Cloud Sleuth
消息总线	无	Spring Cloud Bus
数据流	无	Spring Cloud Stream
批量任务	无	Spring Cloud Task

微服务架构

Neuedu

- 微服务中SpringCloud与Dubbo最大区别
  - Spring Cloud抛弃了RPC通讯，采用基于HTTP的REST方式。
  - Spring Cloud牺牲了服务调用的性能，但是同时也避免了原生RPC带来的问题。
  - REST比RPC更为灵活，不存在代码级别的强依赖，在强调快速演化的微服务环境下，显然更合适。
- 社区的支持与力度：Dubbo曾经停运了5年，虽然重启了，但是对于技术发展的新需求，还是需要开发者自行去拓展，对于中小型公司，显然显得比较费时费力，也不一定有强大的实力去修改源码
- 解决的问题域不一样：Dubbo的定位是一款RPC框架，Spring Cloud的目标是微服务架构下的一站式解决方案 ；

技术选型

- 从开发和运行平台两个维度考虑技术选型。
- 开发框架的选择
  - 首选Spring Cloud作为微服务开发框架

- 运行平台
  - 微服务不绑定运行平台
  - 部署在PC Server、阿里云、AWS等云计算平台
  - 部署在Docker

技术选型

Neuedu

- 为什么使用Spring Cloud
  - 微服务架构的优点表明它可以提高我们的生产力，但是分布式系统本身的技术成本问题给互联网那些创业型公司不少的挑战，阿里、百度等巨头所提供的微服务技术只是解决其中某个问题，而整合封装这些优秀的技术恐怕是Spring最擅长的领域了，Spring Cloud也正因为此而诞生。
  - 使用Spring Cloud来构建微服务架构可以省去你整合各家技术的成本，Spring Cloud为我们构建微服务架构提供了一站式的解决方案，就好比当初Spring诞生是为解决EJB企业应用开发的众多问题而提供的一站式轻量级企业应用开发解决方案一样，随着使用Spring Cloud公司数量的增加，相信微服务将被Spring Cloud统领。

本章重点总结

- 了解微服务产生背景；
- 了解微服务技术栈；
- 理解微服务定义；
- 理解垂直架构、SOA架构；
- 理解微服务架构、微服务架构图；
- 掌握单体架构和微服务架构；
- 掌握微服务设计原则；
- 掌握微服务架构技术选型

课后作业【必做任务】

- 1、微服务定义
- 2、微服务技术栈
- 3、微服务架构技术选型

课后作业【线上任务】

- 线上任务
  - 安排学员线上学习任务（安排学员到睿道实训平台进行复习和预习的任务，主要是进行微课的学习）