**微服务简介：**

微服务架构风格的这种开发，是以开发一组小型服务的方式来开发一个独立的应用系统的，其中每个小型服务都运行在自己的进程中， 并通过轻量的REST API的进行相互通信。这些服务围绕业务功能进行构建，并能通过全自动的部署机制进行独立部署。这些微服务可以使用不同的语言开发，并且可以使用不同的数据库存储技术，对这些微服务我们仅做最低限度的集中管理。

**微服务特性：**

1. 每个微服务可独立运行在自己的进程里
2. 一系列独立运行的微服务共同构建起整个系统
3. 每个服务为独立的业务开发，一个微服务只关注某个特定的功能，如订单管理、用 户管理
4. 微服务之间通过一些轻量级的通信机制进行通信，如REST API、RPC
5. 可以使用不同的语言与数据库技术
6. 全自动的部署机制

**微服务架构的优点：**

1. 易于开发和维护

一个微服务只关注一个特定的功能，业务清晰、代码量较少，开发和维护单个微服 务相对是比较简单的，而整个应用是由若干个微服务构建而成，所以这个应用也会 维持在可控状态。

1. 单个微服务启动较快
2. 局部修改容易部署

单体应用只要有修改，就得重新部署整个应用，微服务解决了这样的问题，一般来 说，对某个微服务进行修改，只需要重新部署这个微服务即可。

1. 技术栈不受限

在微服务中，我们可以结合项目业务及团队特点，合理的选择技术栈。例如某些服 务可以使用关系型数据库mysql、某些微服务用java开发、某些微服务用NodeJS 进行开发

1. 按需伸缩

我们可以根据需求，实现细粒度的扩展。例如：系统中某个微服务遇到了瓶颈，我 们可以结合这个微服务的业务特点，增加内存、升级CUP或者增加节点。

**微服务的架构挑战：**

1. 运维要求较高

更多的微服务意味着更多的运维投入，在单体架构中，只需保证一个应用的正常运 行，而在微服务中，需要保证十几甚至几百个服务的正常运行与协作，这给项目的 运维带来了很大的挑战

1. 分布式固有的复杂性

使用微服务构建的是一个分布式系统，对于一个分布式系统，系统容错、网络延迟、 分布式事务等都给我们带来了很大的挑战

1. 接口调用成本高

微服务之间通过接口进行通信，如果修改了某一个微服务的API，可能所有的使用 了该接口的微服务都需要做调整。

1. 重复劳动

很多微服务可能都会使用到相同的功能，而这个功能并没有达到分解为一个微服务 的程度，这时可能各个服务都会开发这一功能，从而导致代码重复。

**微服务设计原则：**

1. 单一职责原则
2. 服务治理原则（每个微服务应该具有独立的业务能力、依赖与运行环境）
3. 轻量级通信原则
4. 接口明确原则（每个微服务的对外接口应该明确定义，并尽量保持不变）

**开发微服务的支持框架**：dubbo、spring-cloud...