微服务：

单体架构：

将业务的所有功能集中在一个项目中开发，打成一个包部署

优点：

架构简单

部署成本低

缺点：

耦合度高

分布式架构：

根据业务功能对系统进行拆分，每个业务模块作为独立项目开发，成为一个服务

优点：

降低服务耦合

有利于服务升级拓展

服务是架构要考虑的问题：

服务如何拆分？

服务集群地址如何维护？

服务之间如何实现远程调用？

服务健康状态如何感知？

微服务：

微服务其实就是一种经过良好架构设计的分布式架构方案。

微服务架构特征：

单一职责：微服务拆分粒度更小，每一个服务都对应唯一的业务能力，做到单一职责，避免重复开发

面向服务：微服务对外暴露业务接口

自治：团队独立、技术独立、数据独立、部署独立

隔离性强：服务调用做好隔离、容错、讲机、避免出现级联问题

单体架构的特点：

简单方便、高度耦合、扩展性差，适合小型项目。

分部署架构特点：

松耦合，扩展性好，但架构复杂、难度大，适合大型项目

微服务：一种良好的分布式架构方案

优点：拆分粒度更小，服务更独立、耦合度更低

缺点：架构更复杂、运维、监控、部署难度提高

微服务架构：微服务方案需要技术框架落地：如SpringCloud、Dubbo

服务拆分注意事项：

不同微服务，不要重复开发相同业务；

微服务数据独立，不要访问其他微服务的数据库；

微服务可以将自己的业务暴露为接口，供其他微服务调用。

Eureka注册中心的作用：

消费者如何获取服务提供者具体信息：

服务提供者启动的时候向eureka注册自己的信息

Eureka保存这些信息

消费者根据服务名称向eureka拉取提供者信息

如果有多个服务提供者，消费者该如何选择？

服务消费者利用负载均衡算法，从服务列表中挑选一个

消费者如何感知服务提供者健康状态？

服务提供者会每隔30秒向eurekaServer发送心跳请求，报告健康状态；

Eureka会更新记录服务列表信息，心跳不正常会被剔除；

消费者就可以拉取到最新的信息

Eureka架构中，微服务角色有两类：

EurekaServer：服务端，注册中心

记录服务信息；

心跳监控

EurekaClient：客户端

Provider：服务提供者：

注册自己的信息到EurekaServer;

每隔30秒向EurekaServer发送心跳

Consumer：服务消费者，

根据服务名称从EurekaServer拉取服务列表；

基于服务列表做负载均衡，选中一个微服务后发起远程调用

搭建EurekaServer：

引入eureka-server依赖；

添加@EnableEurekaServer注解；

在application.yml中配置eureka地址

服务注册：

引入eureka-client依赖；

在application.yml中配置eureka地址

服务发现：

引入eureka-client依赖；

在application.yml中配置eureka地址；

给RestTemplate添加@LoadBalanced注解；

用服务提供者的服务名称远程调用

Ribbon负载均衡规则：

规则接口是IRule

默认实现是ZoneAvoidanceRule，根据Zone选择服务列表，然后轮询

负载均衡自定义方式：

代码方式：配置灵活，但修改时需要重新打包发布，

配置方式：直观、方便、无需重新打包发布，到那时无法做全局配置

饥饿加载:

开启饥饿加载

指定饥饿加载的微服务名称