**微服务间调用关系、依赖关系**

**如何规划微服务，用到哪些微服务，架构设计**

**springcloud有哪些基础组件（注册中心、配置中心、网关，自己做裁剪，根据自己的系统设计）**

**微服务有哪些api（Apifox）**

**微服务拆分的原则：**

单一职责原则：每个微服务只负责一个特定的业务功能，避免将多个业务功能耦合在一起，以便于微服务的独立开发、部署、测试和维护。

高内聚低耦合原则：每个微服务内部具有高内聚性，相关的功能应该集中在同一个服务中，避免将不相关的功能耦合在一起，以便于微服务的可复用性和扩展性。

可组合性原则：每个微服务应该具有可组合性，其他服务可以使用该服务的功能，而不需要了解其内部实现，以便于微服务的灵活组合和复用。

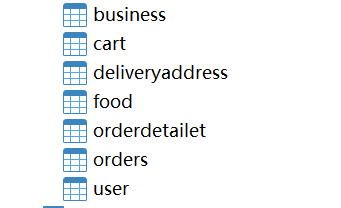
服务粒度适中原则：每个微服务应该具有适中的粒度，即一个微服务应该对应一个特定的业务场景或事务，避免服务过大或过小，以便于微服务的可维护性和可扩展性。

**微服务规划：**

**1、微服务拆分**

根据拆分原则对进行拆分，从数据库入手，进行微服务的规划。

数据库表：



Business（商家表）：在商家表中能够独立地完成新增商家，修改商家信息，查询商家信息等功能，商家可以作为一个微服务。

Cart（购物车表）：购物车表能够实现商品的增添删除，可以作为一个微服务。

DeliveryAddress（送货地址表）：拥有自己独立的功能，也能够作为一个微服务。

Food（食品表）：食品和商家有一定的关系，但与商家不是强依赖关系，如商家的变更不影响食品表的变化，二者相对独立，因此食品也能作为一个微服务。

Orders（订单表）：订单表拥有自己独立的功能，也可以作为一个微服务，而orderedDeatilet表（订单明细表）强依赖于订单表，若没有订单，明细不能够独立存在并且二者是出现在同一个事务中，当产生一个订单时，也会有与之对应的订单明细。因此这二者作为一个微服务，订单服务。

User（用户表）：有自己独立的功能，如登录注册等，作为一个微服务。

最后得到6**个微服务**：商家服务，食品服务，用户服务，订单服务，购物车服务，送货地址服务

1. **集群规划**

*需要构建集群的微服务应该考虑以下几个方面：*

1. 预计的并发量：对于高并发的微服务，需要构建集群来分担负载，提高服务的性能和可用性。
2. 可用性要求：对于需要高可用性的微服务，需要构建集群来保证服务的可用性和容错能力。
3. 数据库读写分离：对于需要读写分离的微服务，需要构建集群来分离读写请求，提高服务的性能和可扩展性。

*基于以上考虑，构建集群的微服务为：*

（1）商家服务：如果商家服务的访问量较大，需要构建集群来分担负载和提高可用性。

（2）食品服务：若商家服务的访问量较大，对应的食品的访问量也会增长，也需要构件集群

（3）订单服务：在饭点时，会有较高的订单服务并发，订单服务需要保证高可用性和容错能力，需要构建集群来实现服务的高可用性和容错能力。

1. **微服务调用关系**

首先要实现相互调用，所有的微服务都需要进行服务发现和注册，以便能够相互调用和协同工作。在本项目中服务发现和注册使用 Eureka工具来实现。

商家服务和食品服务：商家服务需要查询自己的食品信息，因此商家服务需要调用食品服务。

用户服务和购物车服务：购物车服务需要查询用户信息，因此购物车服务需要调用用户服务的。

订单服务和商家服务、购物车服务、用户服务：订单服务需要查询商家信息和用户信息，同时需要更新购物车服务的数据和向用户服务发送通知。

1. **微服务的架构设计**

***各个微服务的架构设计***

商家服务：商家服务提供增删改查商家信息的功能。商家服务需要与食品服务进行交互获取食品信息。商家服务还需要与订单服务进行交互，以便获取订单信息和更新订单状态。

食品服务：食品服务提供增删改查食品信息的功能。

用户服务：用户服务提供注册、登录、查询用户信息等功能。用户服务需要与购物车服务进行交互，以便获取购物车信息和更新购物车状态。

订单服务：订单服务提供下单、查询订单、更新订单状态等功能。订单服务需要与商家服务、购物车服务、用户服务进行交互，以便获取商家信息、购物车信息、用户信息。

购物车服务：购物车服务提供增删改查购物车信息的功能。购物车服务需要与用户服务进行交互，以便获取用户信息和当前用户的购物车状态。

送货地址服务：送货地址服务提供增删改查送货地址信息的功能。送货地址服务可以作为订单服务的一部分，用来保存送货地址信息。

***整体架构设计***

在该项目中，将用到以下组件完成系统的设计

Eureka：Eureka是一个服务注册和发现的组件，它可以让微服务自动进行注册和发现，从而方便微服务之间的通信。

Feign服务调用及负载均衡：Feign是一个基于接口注解的声明式Web服务客户端，它可以帮助我们更方便地进行服务调用和负载均衡。通过Feign，我们可以将需要调用的服务接口定义为一个Java接口，并使用注解来描述服务调用的细节，从而让服务调用变得更加简单和方便。

Hystrix熔断降级：Hystrix是一个用于处理分布式系统的故障和延迟的库，它可以实现熔断、降级、限流等功能。通过在微服务之间添加Hystrix，我们可以增强系统的稳定性和可靠性，在微服务之间进行熔断和降级，从而避免故障的扩散。

Gateway微服务网关：Gateway是一个基于Spring Cloud的API网关，它可以实现对微服务的统一访问和路由、负载均衡、认证和授权、熔断和限流等功能。Gateway可以作为微服务的入口，为微服务提供一层保护和管理，从而简化微服务的开发和部署。

Config集中配置管理集群：Config是一个用于管理微服务配置信息的组件，它可以将配置信息集中存储并动态刷新到微服务实例中。通过在多个实例上部署Config Server，我们可以形成一个集中配置管理的集群，从而提高Config Server的可用性和负载均衡能力。

Bus配置刷新：Bus是一个用于刷新配置信息的组件，它可以实现配置信息的动态刷新和发布。通过在微服务之间添加Bus，我们可以实现配置信息的统一刷新和发布，从而避免手动重启微服务，提高配置信息的可靠性和一致性。