# 1、什么是微服务架构？

微服务架构是一种架构模式或者说是架构风格，它提倡将单一应用程序划分成一组小的服务。每个服务运行在其独立的自己的进程中服务之间相互配合、相互协调，为用户提供最终价值。服务之间采用轻量级通信。每个服务都围绕具体业务进行构建，并能够独立部署到生产环境等。

# 2、微服务的优缺点是什么？

优点：松耦合，聚焦单一业务功能，与开发语言无关，团队规模降低。在开发中，不需要了解多余的业务，只专注于当前功能，便利集中，功能小而精。微服务一个功能受损，对其他功能影响并不是太大，可以快速定位问题。微服务只专注于当前业务逻辑代码，不会和 html、css 或其他界面进行混合。可以灵活搭配技术，独立性比较好。

缺点：随着服务数量增加，管理复杂，部署复杂，服务器需要增多，服务通信和调用压力增大，运维工程师压力增大，人力资源增多，系统依赖增强，数据一致性，性能监控。

# 3、什么是Spring Cloud？

Spring Cloud是一系列框架的有序集合。它利用Spring Boot的开发便利性巧妙地简化了分布式系统基础设施的开发，如服务发现与注册、配置中心、消息总线、负载均衡、断路器、数据监控等，都可以用Spring Boot的开发风格做到一键启动和部署。Spring Cloud并没有重复制造轮子，它只是将各家公司开发的比较成熟、经得起实际考验的服务框架组合起来，通过Spring Boot风格进行再封装屏蔽掉了复杂的配置和实现原理，最终给开发者留出了一套简单易懂、易部署和易维护的分布式系统开发工具包。

# 4、Spring Boot和Spring Cloud之间关系？

Spring Boot：专注于快速方便的开发单个个体微服务（关注微观）

Spring Cloud：关注全局的微服务协调治理框架，将Spring Boot开发的一个个单体微服务组合并管理起来（关注宏观）

Spring Boot可以离开Spring Cloud独立使用，但是Spring Cloud不可以离开Spring Boot，属于依赖关系。

# 5、Spring Cloud 和 Dubbo有哪些区别?

相同点：它们都是**分布式管理框架**。

区别：

1、dubbo使用的是RPC通讯，占用带宽会少一点。Spring Cloud使用的是HTTP的Rest方式进行通讯，带宽会多一点，同时使用http协议一般会使用JSON报文，消耗会更大。

2、dubbo 开发难度较大，所依赖的jar包有很多问题大型工程无法解决。Spring Cloud 对第三方的继承可以一键式生成，天然集成。

# 6、什么是Eureka以及它的架构是什么样子？

介绍：eureka是Netflix开发的服务发现组件，本身是一个基于REST的服务。Spring Cloud将它集成在其子项目spring-cloud-netflix中， 以实现Spring Cloud的服务发现功能。

架构：

Eureka是一个C/S的架构模式，包含了两部分：

1、Eureka Server：注册中心服务端，用于维护和管理注册服务的列表

2、Eureka Client：注册中心客户端，用于向Eureka Server中注册服务和从Eureka Server中拉取服务

# 7、简述一个Eureka的自我保护机制？

心跳检查机制：Eureka Client向Eureka Server中注册完服务信息以后，Eureka Server会通过心跳检测机制来检测当前这个客户端服务是否还存活着！默认的检测机制是Eureka Client每隔30s向Eureka Server发送一个心跳检查包，如果Eureka Server在90s之内没有收到Eureka Client所发送的心跳检查包，那么此时Eureka Server将该Eureka Client从服务列表中剔除掉。

自我保护机制：自我保护模式正是一种针对网络异常波动的安全保护措施，使用自我保护模式能使Eureka集群更加的健壮、稳定的运行。

自我保护机制的工作机制是：如果在15分钟内超过85%的客户端节点都没有正常的心跳，那么Eureka就认为客户端与注册中心出现了网络故障，Eureka Server自动进入自我保护机制，

此时会出现以下几种情况：

1. Eureka Server不再从注册列表中移除因为长时间没收到心跳而应该过期的服务。
2. Eureka Server仍然能够接受新服务的注册和查询请求，但是不会被同步到其它节点上，保证当前节点依然可用。
3. 当网络稳定时，当前Eureka Server新的注册信息会被同步到其它节点中。

因此Eureka Server可以很好的应对因网络故障导致部分节点失联的情况，而不会像ZK那样如果有一半不可用的情况会导致整个集群不可用而变成瘫痪。

自我保护的开关配置：

eureka.server.enable-self-preservation = true # 开启自我保护机制，值设置为false关闭自我保护机制

# 8、什么是Ribbon以及它的工作流程？

概述：Ribbon是一个客户端的负载均衡工具

工作流程：

基本流程如下：

* 拦截我们的RestTemplate请求http://userservice/user/1
* RibbonLoadBalancerClient会从请求url中获取服务名称，也就是user-service
* DynamicServerListLoadBalancer根据user-service到eureka拉取服务列表
* eureka返回列表，localhost:8081、localhost:8082
* IRule利用内置负载均衡规则，从列表中选择一个，例如localhost:8081
* RibbonLoadBalancerClient修改请求地址，用localhost:8081替代userservice，得到http://localhost:8081/user/1，发起真实请求

# 9、在Ribbon中定义了哪些常用的负载均衡算法以及默认的负载均衡算法是哪一个？

常用的负载均衡算法：

1、RoundRobinRule：简单轮询服务列表来选择服务器

2、AvailabilityFilteringRule：对以下两种服务器进行忽略：

（1）在默认情况下，这台服务器如果3次连接失败，这台服务器就会被设置为“短路”状态。短路状态将持续30秒，如果再次连接失败，短路的持续时间就会几何级地增加。

（2）并发数过高的服务器、如果一个服务器的并发连接数过高，配置了AvailabilityFilteringRule规则的客户端也会将其忽略。并发连接数的上限，可以由客户端的<clientName>.<clientConfigNameSpace>.ActiveConnectionsLimit属性进行配置。

3、WeightedResponseTimeRule：  为每一个服务器赋予一个权重值。服务器响应时间越长，这个服务器的权重就越小。这个规则会随机选择服务器，这个权重值会影响服务器的选择。

4、ZoneAvoidanceRule：以区域可用的服务器为基础进行服务器的选择。使用Zone对服务器进行分类，这个Zone可以理解为一个机房、一个机架等。而后再对Zone内的多个服务做轮询。它是Ribbon默认的负载均衡规则。

5、BestAvailableRule：忽略那些短路的服务器，并选择并发数较低的服务器。

6、RandomRule： 随机选择一个可用的服务器。

7、RetryRule：重试机制的选择逻辑。

默认的负载均衡算法：ZoneAvoidanceRule

# 10、什么是fegin？以及如何去使用？

概述：

1、fegin一个声明式的http的客户端工具用来简化远程调用，基于接口的注解的方式来声明一个http的客户端2、feign整合了ribbon，具有负载均衡的能力

3、整合了Hystrix，具有熔断的能力

使用：

1、添加pom依赖

<dependency> <groupId>org.springframework.cloud</groupId> <artifactId>spring-cloud-starter-openfeign</artifactId></dependency>

2、在启动类上添加@EnableFeignClients注解

3、定义一个接口，通过@FeignClient(value = "leadnews-article")指定调用的哪个服务

# 11、你们项目中的配置信息是如何进行管理的？

项目中的配置信息使用的是统一配置中心，常见的统一配置中心有：Spring Cloud Config和nacos。我们项目中使用的是nacos，使用nacos可以很轻松的实现配置信息的热更新。

具体的使用方式如下所示：

1、在nacos中创建一个配置信息，指定配置信息的dataId。dataId的组成：${application.name}-${profile}.yml

2、在项目中添加依赖

<dependency> <groupId>com.alibaba.cloud</groupId> <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId></dependency>

3、在项目的classpath路径下创建一个bootstrap.yml文件，文件中的内容如下所示：

server: port: 51803spring: application: name: user-service # 指定服务名称 cloud: nacos: config: server-addr: 192.168.200.130:8848 # 指定配置中心的ip地址和端口号 file-extension: yml # 指定配置中心中文件的扩展名

# 12、什么是Spring Cloud Gateway以及在你们的项目中如何去应用该组件的？

Spring Cloud Gateway：是Spring Cloud中所提供的一个服务网关组件，是整个微服务的统一入口，在服务网关中可以实现请求路由、统一的日志记录，流量监控、权限校验等一系列的相关功能！

项目应用：权限的校验

具体实现思路：使用Spring Cloud Gateway中的全局过滤器拦截请求(GlobalFilter、Order)，从请求头中获取token，然后解析token。如果可以进行正常解析，此时进行放行；如果解析不到直接返回。