|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学习报告 | | | | | |
| 报告人 | 张卓毅 | 时间 | 2020年07月29日 | 备注 |  |
| 序号 | 学习题目 | | 学习内容 | | |
| 1 | 微服务基础知识 | | 微服务(微服务架构)定义：以缩短交付周期为核心，基于DevOps的演进式架构  微服务（或微服务架构）是一种云原生架构方法，其中单个应用程序由许多松散耦合且可独立部署的较小组件或服务组成。这些服务通常   * 有自己的堆栈，包括数据库和数据模型； * 通过REST API，事件流和消息代理的组合相互通信； * 它们是按业务能力组织的，分隔服务的线通常称为有界上下文。   微服务基本特点   1. 独立部署，灵活扩展 2. 资源的有效隔离 3. 团队组织架构的调整(水平团队组织架构变为垂直团队组织架构)   系统架构演变  单体应用架构 优点：开发简单，适用于小型应用  缺点：不易拓展、维护、代码耦合    垂直应用架构：优点：解决高并发问题，针对不同模块优化，方便水平扩展，容错  缺点：系统间相互独立，重复开发工作  分布式架构：优点：分布式，可重用，扩展灵活，松耦合  缺点：服务的评估、治理、调度  出现了SOA(Service-Oriented Architecture 面向服务的架构)  流行框架 esb 、dubbo    优点：抽取公共的功能为服务，提高开发效率  对不同的服务进行集群化部署解决系统压力  基于ESB/DUBBO减少系统耦合  缺点：抽取服务粒度较大  服务提供方与调用方接口耦合度较高  微服务架构：  优点：通过服务的原子化拆分，以及微服务的独立打包、部署 和升级，小团队的交付周期将缩短，运维成本也将大幅度下降  微服务遵循单一原则。微服务之间采用Restful等轻量协议传输  缺点：微服务过多，服务治理成本高，不利于系统维护  分布式系统开发的技术成本高(容错、分布式事务等)  SOA与微服务的关系  SOA(Service-Oriented Architecture)面向服务的架构：一种设计方法，其中包含多个服务，服务之间通过相互依赖最终提供一系列功能。一个服务通常以独立的形式存在于操作系统进程中。各个服务之间通过网络调用  微服务架构：和SOA架构类似，微服务是在SOA上做的升华，微服务架构强调的是一个重点是“业务需要彻底组件化和服务化”，原有的单个业务系统会拆分为多个课以独立开发、设计、运行的小应用。这些小应用之间可以通过服务完成交互和集成    分布式中的远程调用     * 1. http相对更规范，更标准，更通用，无论哪种语言都支持http协议。现在开源中间件，基本最先支持的几个协议都包含RESTful   2. RPC框架作为架构微服务化的基础组件，它能大大降低架构微服务的成本，提高调用放于服务提供放的研发效率，屏蔽跨进程调用服务的各类复杂细节。让调用放感觉像是调用本地函数一样地调用远程函数，让服务提供方感觉像实现一个本地函数一样来实现服务 | | |
| 2 | springcloud组件 | | Nginx 同 Apache 一样都是一种 Web 服务器，一个高性能HTTP和反向代理服务器，处理请求是异步非阻塞的。基于 REST 架构风格，以统一资源描述符（Uniform Resources Identifier）URI 或者统一资源定位符（Uniform Resources Locator）URL 作为沟通依据，通过 HTTP 协议提供各种网络服务。  特点：轻量高并发  使用事件驱动架构，使得其可以支持数以百万级别地TCP连接  高度模块化和自由软件许可证使得第三方模块层出不穷  跨平台，可在linux，Linux、Windows、FreeBSD、Solaris、AIX、Mac OS 等操作系统上  极大的稳定性  代理：正向代理和反向代理      Nginx负载均衡：将客户端的请求按照一定规则分发到不同服务器处理处理  Nginx 支持的负载均衡调度算法方式如下：  ①weight 轮询（默认）：接收到的请求按照顺序逐一分配到不同的后端服务器，即使在使用过程中，某一台后端服务器宕机，Nginx 会自动将该服务器剔除出队列，请求受理情况不会受到任何影响。  这种方式下，可以给不同的后端服务器**设置一个权重值（weight）**，用于调整不同的服务器上请求的分配率。  权重数据越大，被分配到请求的几率越大；该权重值，主要是针对实际工作环境中不同的后端服务器硬件配置进行调整的。  **②ip\_hash**：每个请求按照发起客户端的 ip 的 hash 结果进行匹配，这样的算法下一个**固定 ip 地址的客户端总会访问到同一个后端服务器**，这也在一定程度上解决了**集群部署环境下 Session 共享**的问题。  ③fair：智能调整调度算法，动态的根据后端服务器的请求处理到响应的时间进行均衡分配。  响应时间短处理效率高的服务器分配到请求的概率高，响应时间长处理效率低的服务器分配到的请求少，它是结合了前两者的优点的一种调度算法。  但是需要注意的是 Nginx 默认不支持 fair 算法，如果要使用这种调度算法，请安装 upstream\_fair 模块。  ④url\_hash：按照访问的 **URL 的 hash 结果分配请求**，每个请求的 URL 会指向后端固定的某个服务器，可以在 **Nginx 作为静态服务器的情况下提高缓存效率。**  同样要注意 Nginx 默认不支持这种调度算法，要使用的话需要安装 Nginx 的 hash 软件包。  微服务网关SpringCloudGateway  Spring cloud gateway是spring官方基于Spring 5.0、Spring Boot2.0和Project Reactor等技术开发的网关，Spring Cloud Gateway旨在为微服务架构提供简单、有效和统一的API路由管理方式，Spring Cloud Gateway作为Spring Cloud生态系统中的网关，目标是替代Netflix Zuul，其不仅提供统一的路由方式，并且还基于Filer链的方式提供了网关基本的功能，例如：安全、监控/埋点、限流等。  网关提供API全托管服务，丰富的API管理功能，辅助企业管理大规模的API，以降低管理成本和安全风险，包括协议适配、协议转发、安全策略、防刷、流量、监控日志等贡呢。一般来说网关对外暴露的URL或者接口信息，我们统称为路由信息。   * 路由。路由是网关最基础的部分，路由信息有一个ID、一个目的URL、一组断言和一组Filter组成。如果断言路由为真，则说明请求的URL和配置匹配 * 断言。Java8中的断言函数。Spring Cloud Gateway中的断言函数输入类型是Spring5.0框架中的ServerWebExchange。Spring Cloud Gateway中的断言函数允许开发者去定义匹配来自于http request中的任何信息，比如请求头和参数等。 * 过滤器。一个标准的Spring webFilter。Spring cloud gateway中的filter分为两种类型的Filter，分别是Gateway Filter和Global Filter。过滤器Filter将会对请求和响应进行修改处理   网关是介于客户端和服务器端之间的中间层，所有的外部请求都会先经过 网关这一层。也就是说，API 的实现方面更多的考虑业务逻辑，而安全、性能、监控可以交由 网关来做，这样既提高业务灵活性又不缺安全性  优点：安全，只有网关系统对外进行暴露，微服务可以隐藏在内网，通过防火墙保护  易于监控。可以在网关收集监控数据并将其推送到外部系统进行分析。  易于认证。可以在网关上进行认证，然后再将请求转发到后端的微服务，而无需要每个微服务中进行认证  减少客户端于各个微服务之间的交互次数  易于统一鉴权  Netflix Zuul – 服务网关  作用：api网关，路由，负载均衡等多种作用  简介：类似nginx，反向代理的功能，不过netflix自己增加了一些配合其他组件的特性。  在微服务架构中，后端服务往往不直接开放给调用端，而是通过一个API网关根据请求的url，路由到相应的服务。当添加API网关后，在第三方调用端和服务提供方之间就创建了一面墙，这面墙直接与调用方通信进行权限控制，后将请求均衡分发给后台服务端。  Netflix Eureka-服务发现  作用：实现服务治理（服务注册与发现）  简介：Spring Cloud Eureka是Spring Cloud Netflix项目下的服务治理模块。  由两个组件组成：Eureka服务端和Eureka客户端。  Eureka服务端用作服务注册中心。支持集群部署。  Eureka客户端是一个java客户端，用来处理服务注册与发现。  在应用启动时，Eureka客户端向服务端注册自己的服务信息，同时将服务端的服务信息缓存到本地。客户端会和服务端周期性的进行心跳交互，以更新服务租约和服务信息。  Netflix Ribbon – 客服端负载均衡  作用：Ribbon，主要提供客户侧的软件负载均衡算法。  简介：Spring Cloud Ribbon是一个基于HTTP和TCP的客户端负载均衡工具，它基于Netflix Ribbon实现。通过Spring Cloud的封装，可以让我们轻松地将面向服务的REST模版请求自动转换成客户端负载均衡的服务调用。  Netflix Hystrix – 断路器  作用：断路器，保护系统，控制故障范围。  简介：为了保证其高可用，单个服务通常会集群部署。由于网络原因或者自身的原因，服务并不能保证100%可用，如果单个服务出现问题，调用这个服务就会出现线程阻塞，此时若有大量的请求涌入，Servlet容器的线程资源会被消耗完毕，导致服务瘫痪。服务与服务之间的依赖性，故障会传播，会对整个微服务系统造成灾难性的严重后果，这就是服务故障的“雪崩”效应。  Spring Cloud Config – 分布式配置  作用：配置管理  简介：SpringCloud Config提供服务器端和客户端。服务器存储后端的默认实现使用git，因此它轻松支持标签版本的配置环境，以及可以访问用于管理内容的各种工具。这个还是静态的，得配合Spring Cloud Bus实现动态的配置更新。 | | |