Java整体知识架构详解-之微服务架构

- SpringBoot
 - 概述
 - 实现了自动配置,集成了大量常用第三方库,开箱即用
 - spring-boot-starter
 - 将代码改造成starter可插拔插件,做到引入pom后开箱即用
 - SpringBoot可直接集成的常用服务
 - 常用安全框架Security,可用于登录认证,OAuth2授权认证等
 - 常用数据库mysql, oracle等快速集成
 - 常用Nosql数据库,redis,MongoDB,Solr,Elasticsearch,InfluxDB等快速集成
 - 常用缓存架构ehcache2.x,Redis等
 - 常用消息服务,JMS(ActiveMQ等),AMQP(Rabbitmq等),Kafka相关mq支持
 - 常用定时任务Quartz Scheduler
- SpringCloud
 - Eureka
 - , 注册中心,管理协调分布式集群信息,类似的有zookeeper,Nacos等
 - Eureka1.x目前还在维护,但预计的2.x已经宣布不再开发
 - Zuul
 - 概述
 - 网关服务
 - 网关基本功能
 - 单点入口
 - 路由转发
 - 熔断限流
 - 日志监控
 - 安全认证
 - 网关高级应用
 - 4 红绿部署
 - 开发者测试分支
 - 埋点测试
 - 压力测试
 - 调试路由
 - 金丝雀测试
 - 粘性金丝雀
 - 失败注入测试
 - 降级测试

比较

同步阻塞模型,容易埋点监控,稳定成熟,容易debug

Zuul2

- 异步非阻塞模型,线程开销少,连接数易扩展,但编程模型复杂,开发调试运维复杂 SpringCloudGateway
 - spring自己研发的异步非阻塞模型的网关,和zuul2功能模型类似,也有相同的问题和优势

Fegin

RPC远程调用技术,使用的http协议,引入Feign后自带了Ribbon负载均衡的引入,默认轮询

Config

概述

配置管理中心,采用git作为文件的管理中心

优点

- 采用了git作为管理中心,通过目录分级的形式管理不同服务配置,对于功能性要求不高的系统简单使用它的基础功能也够了
- 可以对密码等敏感信息进行加密

缺点

》对于庞大复杂的系统,或对自己配置管理要求比较高的,这个服务缺点明显,第一没有管理界面,第二没有附加默认附加功能,默认不能自动更新配置,第三如果要配置自动更新不仅配置复杂,而且依赖比较多,需要依赖bus总线和mq,还需要qit上配置hook调用

Ribbon

负载均衡组件,主要有权重和轮询这两种模式

Sleuth

概述

链路调用追踪组件

优点

- 组件小巧,部署简单,配置简单
- 有简单的服务调用图展示
- 缺点

功能单一,主要注重在简单的链路及调用时间的展示,对于链路调用在<u>分布式架构</u>的讨论里 讨论了几种工具,个人比较推荐Cat,国内的东西还是更容易看懂,报表丰富也不差

Hystrix

概述

熔断降级限流组件

容错模式

超时

主动超时

限流

限制最大并发数

熔断 错误数达到阈值时,类似保险丝熔断 隔离 隔离不同的依赖调用 降级 服务降级 展示界面 Dashboard 查看单个应用的Hystrix信息 Turbine 整合多个应用的Hystrix信息整合显示 隔离模式 线程池模式 通过线程池策略来接收请求,会消耗一定的线程,自行斟酌,不要把核心线程数列队等 设置太大 线程池预估建议 Thread Size=每秒请求QPS*每个请求预计耗时s+预留空间breathing room;比如: QPS等于30,预计0.2秒也就是200毫秒一个请求,预留空间比如说4个线程,那么 Size=30*0.2+4=10,得出线程数为10;队列大小设置5~10差不多 信号量模式 使用的是Semaphore信号量隔离,本质是计数器,不会消耗大量线程 Bus 消息总线,通常配合mg,用来通知同样部署总线的服务 SpringBoot2.0 WebFlux Reactive Stream jdk9的发布订阅处理 webflux服务端开发 reactor约等于jdk8的stream+jdk9的Reactive Stream 例子 controller调用 Mono 代表0-1的返回 Flux 代表1-n的返回 例子 原理 SSE(Server-Send Events) 服务端 RouteFunction模式

- 代码示例
 - handler
 - routers
- restclient框架开发
- 优势
 - 可以有更高并发量

SpringCloud和Dubbo对比

- dubbo作为微服务治理框架专注于RPC领域,而SpringCloud则宣称自己是一整套的微服务解决方案
- dubbo在rpc协议上用的是dubbo协议已,它还支持多种协议,http, rmi, hession等,Spring Cloud目前使用的是http协议
- dubbo的集群容错模式,默认调用失败选取别的相同服务调用;它还可以设置调用失败,直接报异常;或不报异常,记录日志;或失败后定时调用;或并发调用,成功一条其它作废;SpringCloud是通过 Hystrix来实现熔断,快速失败等
- 负载均衡上,dubbo有四种模式,随机、轮询、一致性hash、最小调用;SpringCloud的负载模式则有随机、轮询、过滤故障服务轮询,过滤故障服务最小并发等等
- **速度上dubbo协议底层采用Netty长连接模式,适合少量数据的高并发调用,速度会比SpringCloud快点**

Docker

- 概述
 - Docker是一个开源的引擎,可以为任何应用创建一个轻量级、可移植的、自给自足的容器
- 界面管理
 - 可以方便的进行查找,关闭,删除,运行等,可以找到不再使用的挂载进行删除
- 网络容器
 - 容器的bridge是可以自定义的,一个容器可以加入多个bridge网络,哪怕是一个运行中容器,可以通过`docker network connect my_bridge web`连接my_bridge网桥,当然想断开,connect换成disconnect
 - 可以同过`docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' web` 查找容器内部网络地址

容器编排管理

- . Swarm
 - Docker官方推出的管理工具
- Kubernetes (k8s)
 - google开发经验积累充足,功能更加强大完善,google大力推广社区活跃,是主流的服务编 排框架
- Dockerfile
 - 概述
 - Dockerfile是由一系列命令和参数构成的脚本,最终创建一个新的镜像,简化部署工作
 - 命令
 - FROM
 - 定义了使用哪个基础镜像启动构建流程,该命令是Dockerfile的首个命令
 - ADD

命令有两个参数,源和目标,作用是从源系统的文件系统上**复制文件**到目标容器的文件系统,如果源是个URL,则会下载内容并复制到容器中

RUN

RUN命令是Dockerfile执行命令的核心部分,它接收命令作为参数并用于创建镜像,比如 `RUN aptitude install -y riak`

CMD

和RUN命令相似,CMD可以用于执行特定的命令,和RUN不同的是,这些命令不是在构建镜像的过程中执行的,而是用在<mark>镜像构建容器后被调用</mark>,用法:CMD application "argument", "argument", .

ENTRYPOINT

配置容器启动后执行的命令,并且**不可被docker run提供的参数覆盖,每个Dockerfile 只能有一个ENTRYPOINT**,指定多个只有最后一个生效;可以从CMD中移除
`application`仅仅保留参数,参数会自动传递给ENTRYPOINT。例如:ENTRYPOINT echo
CMD "Hello world"

ENV

用于设置环境变量。例如:ENV SERVER WORKS 4

EXPOSE

- 用来指定端口,使容器内应用可以通过端口和外界交互。例如: EXPOSE 8080
- MAINTAINER
 - 一般放在Dockerfile起始部分FROM之后,用于声明作者。例如:MAINTAINER authors_name
- USER
 - 用于设置运行容器的UID。例如: USER 751
- VOLUME
 - 用于让容器访问宿主机上的目录。例如: VOLUME ["/dir_1","/dir_2" ..]
- WORKDIR
 - 用于指明CMD命令运行的目录。例如 WORKDIR /path
- Dockerfile构建镜像
 - docker build -t 镜像名称 . (最后还有个点表示以此目录中的Dockerfile构建镜像)
- Docker Compose
 - 概述
 - 用于定义和运行多容器Docker的应用程序工具,需要安装Docker Compose
 - Docker Compose常用命令
 - docker-compose ps
 - 列出所有容器
 - docker-compose logs
 - 查看服务日志输出
 - docker-compose port eureka 8761
 - 打印绑定的公共端口
 - docker-compose build

- 构建或重新构建服务
- docker-compose start eureka
 - 启动指定服务已存在的容器
- docker-compose stop eureka
 - 停止已运行的服务的容器
- docker-compse rm eureka
 - 删除指定服务的容器
- docker-compse up
 - 构建、启动容器
- docker-compose scale user=3 movie=3
 - 设置指定服务运行容器个数
- docker-compose run web bash
 - 在一个服务上执行一个命令
- docker-compose.yml 属性
 - version
 - 指定docker-compse.yml文件的写法格式
 - 指定docker-compse.yml文件的写法格式
 - 指定docker-compse.yml文件的写法格式
 - version
 - image
 - 指定服务所使用的镜像
 - services
 - 多个容器集合
 - 多个容器集合
 - 多个容器集合
 - services
 - build
 - " 用于构建镜像的参数指定,可以是一个路径,也可以是一个对象
 - 用于构建镜像的参数指定,可以是一个路径,也可以是一个对象
 - 用于构建镜像的参数指定,可以是一个路径,也可以是一个对象
 - build
 - volumes
 - 巻挂载路径
 - volumes_from
 - 从另一个服务或容器挂载它的所有卷
 - expose
 - 暴露端口给连接的服务,而不暴露给主机
 - ports
 - 对外暴露的端口定义
 - links

- 》将指定容器连接到当前连接,可以设置别名,避免ip方式导致的容器重启动态改变的无法连接情况
- external links
 - 链接到外部容器
- network_mode
 - 设置网络模式
- command
 - 覆盖容器启动后默认执行的命令
- dns
 - 配置dns服务器,可以是一个值或列表
- pid
 - 跟主机系统共享进程命名空间。打开该选项的容器可以相互通过进程 ID 来访问和操作。
- docker run支持的选项
 - working_dir, entrypoint, user, hostname, domainname, mem_limit, privileged, restart, stdin_open, tty, cpu_shares
- dns search
 - 配置DNS搜索域,可以是一个值或列表
- environment
 - 环境变量配置,可以用数组或字典两种方式
- env_file
 - 从文件中获取环境变量,可以指定一个文件路径或路径列表,**优先级低于environment** 指定的环境变量
- 最佳实践
 - 更新容器
 - 当服务的配置发生更改时,可使用docker-compse up命令更新配置,此时Compose会删除旧容器并创建新容器
 - ^{*} 利用好links
 - 服务之间相互访问用links定义别名,从而使用该别名访问其它服务