1. 源码分析
   1. DiscoveryClient类
      1. 解读该类头部的注解

--向Eureka Server注册服务实例

--向Eureka Server服务租约

--当服务关闭期间，向Eureka Server取消租约

--查询Eureka Server中的服务实例列表

Eureka Client配置一个Eureka Server的URL列表

* + 1. Region、Zone

--不设置，默认为default，即设置中的defaultZone

--可以通过，eureka.client.availability-zones属性自定义Zone，getAvailabilityZones可以设置多个Zone以逗号分隔【设计出对区域性故障的容错集群】

1.1.3 DiscoveryClient的initScheduledTasks函数中两个定时任务，分别是“服务获取”和“服务续约”

Private void initScheduledTasks() {

If (clientConfig.shouldFetchRegistry()) {

// registry cache refresh timer

int registryFetchIntervalsSeconds = clientConfig.getRegistryFetchIntervalSeconds();

int expBackoffBound = clientConfig.getCacheRefreshExecutorExponentialBackOffBound();

scheduler.schedule(

new TimedSupervisorTask(

“cacheRefresh”,

scheduler,

cacheRefreshExecutor,

registryFetchIntervalSeconds,

TimeUnit.SECONDS,

expBackOffBound,

new CacheRefreshThread()

),

registryFetchIntervalSeconds, TimeUnit.SECONDS);

}

If (clientConfig.shouldRegisterWithEureka()) {

int renewalIntervalInSecs = instanceInfo.getLeaseInfo().getRenewalIntervalInSecs();

int expBackOffBound = clientConfig.getHeartbeatExecutorExponentialBackOffBound();

logger.info(“Starting heartbeat executor: ” + “renew interval is:” + renewalIntervalInSecs);

// Heartbeat Timer

Scheduler.schedule(

new TimedSupervisorTask(

“heartbeat”,

scheduler,

heartbeatExecutor,

renewlIntervalInSecs,

TimeUnit.SECONDS,

expBackOffBound,

new HeartbeatThread()

),

renewalIntervalInSecs, TimeUnit.SECONDS);

// InstanceInfo replicator

…

}

}

※服务注册与服务续约成对出现

“服务续约”：

boolean renew() {

EurekaHttpResponse<InstanceInfo> httpResponse;

try{

httpResponse = eurekaTransport.registrationClient.sendHeartBeat

(instanceInfo.getAppName(), instanceInfo.getId(),instanceInfo, null);

Logger.debug(“{} – Heartbeat status: {}”, PREFIX + appPathIdentifier,

httpResponse.getStatusCode());

if (httpResponse.getStatusCode() == 404) {

REREGISTER\_COUNTER.increment();

Logger.info(“{} – Re-registering apps/{}”, PREFIX + appPathIdentifier,

instanceInfo.getAppname());

return register();

}

return httpResponse.getStatusCode() == 200;

} catch (Throwable e) {

logger.error(“{} – was unable to send heartbeat!”, PREFIX + appPathIdentifier, e);

return false;

}

}

1. 配置详解【org.springframework.cloud.netflix.eureka.server.EurekaServerConfigBean类】

2.1Eureka客户端的配置主要分为以下两个方面。

●服务注册相关的配置信息，包括服务注册中心的地址、服务获取的间隔时间、可用区域等

●服务实例相关的配置信息，包括服务实例的名称、IP地址、端口号、健康检查路径等。

将enableSelfPreservation参数设置为false以防止关闭的实例无法被服务注册中心剔除的问题。

2.2服务注册类配置

【org.springframework.cloud.netflix.eureka.EurekaClientConfigBean类】

2.2.1 可为服务注册中心加入安全校验，例如,

[http://<username>:<password>@localhost:1111/eureka](http://%3cusername%3e:%3cpassword%3e@localhost:1111/eureka)

<username>为安全校验信息的用户名，<password>为该用户密码

表1-1 以eureka.client为前缀的配置

|  |  |  |
| --- | --- | --- |
| 参数名 | 说明 | 默认值 |
| enabled | 启用Eureka客户端 | True |
| registryFetchIntervalSeconds | 从Eureka服务端获取注册信息的间隔时间，单位为秒 | 30 |
| instanceInfoReplicationIntervalSeconds | 更新实例信息的变化到Eureka服务端的间隔时间，单位为秒 | 30 |
| initialInstanceInfoReplicationIntervalSeconds | 初始化实例信息到Eureka服务端的间隔时间，单位为秒 | 40 |
| eurekaServiceUrlPollIntervalSeconds | 轮询Eureka服务端地址更改的间隔时间，单位为秒。当我们与Spring Cloud Config配合，动态刷新Eureka的serviceURL地址时需要关注该参数 | 300 |
| eurekaServerReadTimeoutSeconds | 读取Eureka Server信息的超时时间，单位秒 | 8 |
| eurekaServerConnectTimeoutSeconds | 连接Eureka Server的超时时间，单位为秒 | 5 |
| eurekaServerTotalConnections | 从Eureka客户端到所有Eureka服务端的链接总数 | 200 |
| eurekaServerTotalConnectionsPerHost | 从Eureka客户端到每个Eureka服务端主机的连接总数 | 50 |
| eurekaConnectionIdleTimeoutSeconds | Eureka服务端连接的空闲关闭时间，单位为秒 | 30 |
| heartbeatExecutorThreadPoolSize | 心跳连接池的初始化线程数 | 2 |
| heartbeatExecutorExponentialBackOffBound | 心跳超时重试延迟时间的最大乘数值 | 10 |
| cacheRefreshExecutorThreadPoolSize | 缓存刷新线程池的初始化线程数 | 2 |
| cacheRefreshExecutorExponentialBackOffBound | 缓存刷新重试延迟时间的最大乘数值 | 10 |
| useDnsForFetchingServiceUrls | 使用DNS来获取Eureka服务端的serviceUrl | false |
| registerWithEureka | 是否将自身的实例信息注册到Eureka服务端 | true |
| preferSameZoneEureka | 是否偏好使用处于相同Zone的Eureka服务端 | true |
| filterOnlyUpInstances | 获取实例时是否过滤，仅保留UP状态的实例 | true |
| fetchRegistry | 是否从Eureka服务端获取注册信息 | true |

2.3服务实例类配置

【org.springframework.cloud.netflix.eureka.EurekaInstanceConfigBean】

表1-2 以eureka.instance为前缀

|  |  |  |
| --- | --- | --- |
| 参数名 | 说明 | 默认值 |
| preferIpAddress | 是否优先使用IP地址作为主机名的标识 | False |
| leaseRenewalIntervalInSeconds  【最好使用默认值，因为服务器内部有一些计算可以续约周期进行假设】 | Eureka客户端向服务端发送心跳的时间间隔，单位为秒 | 30 |
| leaseExpirationDurationInSeconds | Eureka服务端在收到最后一次心跳之后等待的时间上限，单位为秒。  **超过该时间之后服务端会将该服务实例从服务清单中剔除，从而禁止服务调用请求被发送到该实例上** | 90 |
| nonSecurePort | 非安全的通信端口号 | 80 |
| securePort | 安全的通信端口号 | 443 |
| nonSecurePortEnabled | 是否启用非安全的通信端口号 | True |
| securePortEnabled | 是否启用安全的通信端口号 |  |
| appname | 服务名，默认取spring.application.name的配置值，如果没有则为unknown |  |
| hostname | 主机名，不配置的时候将根据操作系统的主机名来获取 |  |
| statusPageUrlPath | 查看状态页 | /info |
| healthCheckUrlPath | 查看健康检查标准 | /health |
| instanceId |  |  |
| metadataMap.zone | 告知Eureka服务所在的区域 |  |

2.3.1 查看com.netflix.appinfo.InstanceInfo类中的详细定义来了解原生Eureka对元数据的定义。

2.3.2 实例名配置

1、默认规则：**【用在Eureka信息面板上，Status一列】**

eureka.instance.instanceId=

${spring.cloud.client.hostname}:${spring.application.name}:${spring.application.instance\_id}:${serves.port}

2、使用eureka.instance.instanceId参数来进行配置

3、端口号配置可使用随机数：server.port=${random.int[10000,19999]}

4、实例名规则：

eureka.instance.instanceId=${spring.application.name}:${random.int}

2.3.3 spring-boot-actuator模块的配置见P69