微服务引擎

常见问题

文档版本 01

发布日期 2024-10-21

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 使用华为云 CSE 注意事项

1.1 查看 ServiceComb 引擎、Nacos 引擎和应用网关信息时为什么看不到 VPC、ELB 等云服务信息?

问题描述

用户在查看ServiceComb引擎、Nacos引擎和应用网关信息时看不到VPC、ELB等云服务信息。

问题原因

当前用户没有云服务（如VPC、ELB等）所在企业项目的查看权限，请联系租户管理员授权后再查看。

1.2 创建委托失败怎么解决?

问题描述

用户进入微服务引擎控制台，提示需要开通服务并授权，单击授权后提示授权创建失败。

原因分析

用户无权限创建委托或用户委托配额不足。

解决方法

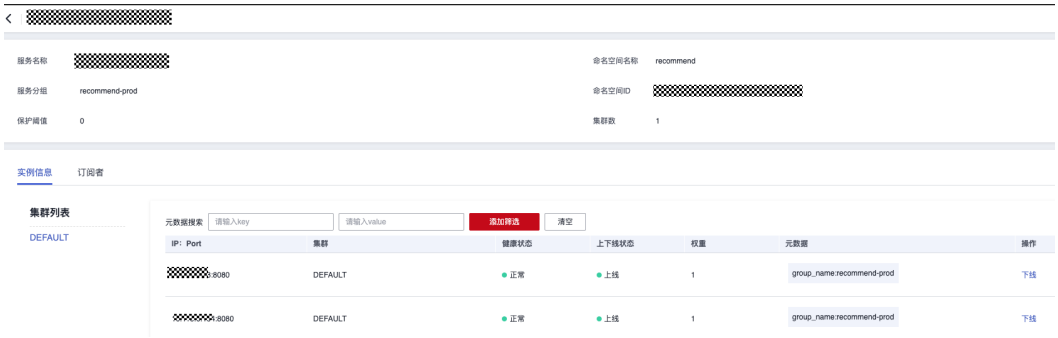
- 若用户进入IAM控制台时，系统提示权限不足，说明用户无权限创建委托，请参考**创建委托时提示权限不足怎么办**进行处理。
- 若用户有权限创建委托，则登录统一身份认证服务控制台，在统一身份认证服务的左侧导航窗中，选择“委托”，查看当前是否还可以创建委托，当可创建委托数量为0时，可删除委托或调整委托配额后进行创建。

2 Nacos 引擎

2.1 服务启动时注册了端口为 8080 和 9090 的实例，在服务列表中 9090 端口实例丢失，导致请求 grpc 的时候报错

问题描述

服务启动时分别注册了端口为8080和9090的两个实例，对应http和grpc的端口，但是服务发现列表只能看到8080的实例，导致请求grpc的时候报错。



可能原因

使用了go sdk的client连接Nacos时，在一个微服务同时注册了两个实例，当前Nacos并不支持。

解决方法

同一个微服务不要同时注册两个实例。

3 ServiceComb 引擎

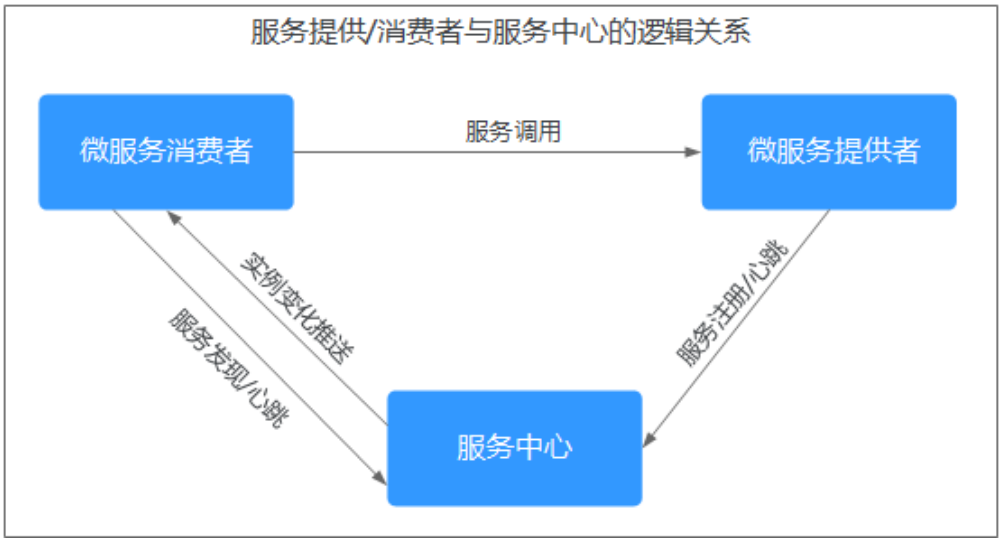
3.1 如何进行本地开发和测试?

本地开发工具给开发者提供开发、调试、测试过程中的服务发现、注册和查询功能。

概念阐述

本小节介绍如何在开发者本地进行消费者/提供者应用的开发调试。服务提供者和消费者均需要连接到在远程的服务中心，为了本地微服务的开发和调试，本小节介绍了如何**启动本地服务中心**。

服务中心是微服务框架中的重要组件，用于服务元数据以及服务实例元数据的管理和处理注册、发现。服务中心与微服务提供/消费者的逻辑关系下图所示：



前提条件

- 由于启动本地服务中心将会占用此台机器的30100、30110和30103端口，其分别表示服务中心的后台、配置中心的后台和前台服务端口。请确认以上端口未被使用。
- 使用本地开发工具前，请确认环境是否满足以下要求：

- 操作系统：Linux/Unix、Windows 64 bit
- CPU架构：x86/arm
- 浏览器：Chrome、Safari、Edge

启动本地服务中心

步骤1 进入**本地开发工具说明**，根据具体环境下的操作系统、CPU架构，下载对应版本的本地开发工具压缩包到本地并解压缩到安装目录。

步骤2 启动CSE。

- Linux/Unix系统，进入安装根目录，执行如下命令：
  ```
  nohup sh start.sh >/dev/null 2>&1 &
  ```
- Windows系统，进入安装根目录，双击cse.exe文件启动。

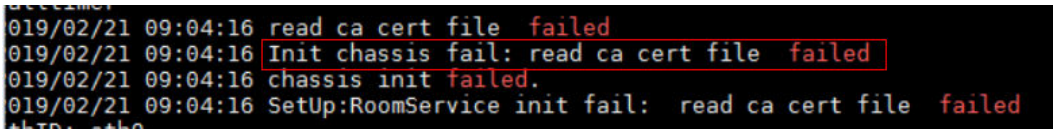
步骤3 停止CSE。

- Linux/Unix系统，进入安装根目录，执行如下命令：
  ```
  sh stop.sh
  ```
- Windows系统，关闭命令行窗口。

----结束

3.2 证书加载错误

问题描述



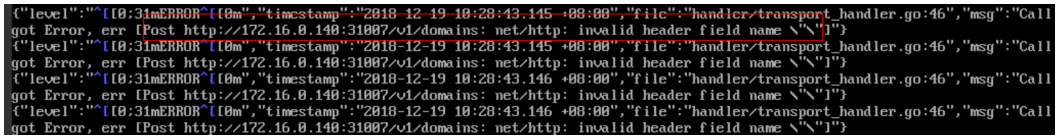
看到此错误时，说明verifyPeer选项被设置为true，却没有配置证书，go-chassis默认为false，不会自己更改配置。

解决方案

- 如果本来并不打算配置证书的，说明是开发者在开发或者持续集成的某个环节中自行更改的，请仔细排查代码从提交到部署环节中，是谁打开的此选项。
- 缺少的证书，配上即可。

3.3 无效头名称

问题描述



此问题与go-chassis无关。

解决方案

请仔细排查业务代码，尤其是定制的go chassis handler，是否有传入空header，或者不规范的header name。

3.4 mesher 性能损耗是多少?

服务网格技术实际利用了网络流量劫持的方式来管理服务间流量，除了mesher本身内部的逻辑处理会耗时之外，还会引起额外的用户态和内核态间转换（CPU会有额外消耗），而前者相对于后者性能影响极小，因此性能损耗基本取决于网络中传输的payload大小。以http协议举例，影响传输速度的就是header、body等内容的大小。mesher一次端到端调用中的延迟为1ms，一个典型的用户测试过自己真实的业务调用，加上mesher后，延迟高了4ms，在用户可接受范围内。

以下测试结果为加入mesher前后的性能测试对比，使用的payload很小，就是字符串helloworld，但是加入了一定的代码以增加服务端的计算时间来模仿业务代码执行耗时。

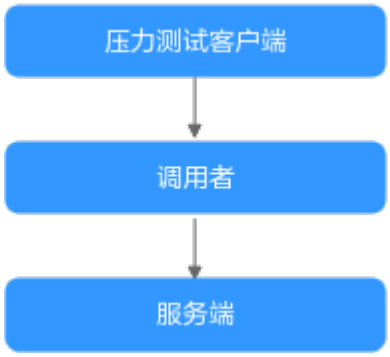


表 3-1 测试结果

指标	使用前	使用后
TPS	1749	1496
Latency	2.8ms	3.34ms
CPU	50%	100%
Concurrency	5	5

从以上结果可以看出mesher本身性能损耗很低，性能的主要瓶颈是在业务代码，如果增大payload内容，性能将会进一步降低。

建议在初期选型与POC时，使用该技术进行业务间的调用以测试真实性能损耗。

3.5 连接服务中心提示“Version validate failed”

异常消息

```
{"errorCode":"400001","errorMessage":"Invalid parameter(s)","detail":"Version validate failed, rule: {Length: 64,Length: ^[a-zA-Z0-9_\-.]*$}"}
```

问题原因

使用了新版本的SDK，却连接服务中心老版本的SDK。

排查方法

检查服务中心的版本。可以从公有云官网下载最新版本的服务中心，或者从ServiceComb官网下载最新版本的服务中心。

3.6 连接服务中心提示“Not enough quota”

异常消息

```
{"errorCode":"400100","errorMessage":"Not enough quota","detail":"no quota to create instance, ..."}
```

问题原因

没有足够的额度增加服务实例。

排查方法

登录公有云，在微服务引擎页面，可以看到实例个数的额度。如果发现页面有额度，需要检查下代码配置的服务中心地址和区域信息。

3.7 如何处理开启了安全认证的 ServiceComb 引擎专享版开启 IPv6 后服务注册失败?

问题描述

基于Java Chassis开发的微服务注册到开启了安全认证的ServiceComb引擎专享版，微服务的注册中心地址使用微服务引擎注册中心的IPv4地址，可以注册成功并正常启动。

如果修改微服务的注册中心地址为ServiceComb引擎注册中心的IPv6地址后，注册失败并报错“java.net.SocketException: Protocol family unavailable”。

可能原因

创建ServiceComb引擎专享版时，当选择开启了IPv6的VPC网络时，创建引擎支持IPv6网络。当部署服务使用IPv6网段且选择容器部署时，选择的CCE集群需要开启IPv6双栈开关。

如果选择的CCE集群资源没有开启IPv6开关，就会导致服务网络不通，报错“java.net.SocketException: Protocol family unavailable”。

解决方法

步骤1 修改部署了微服务应用的环境，添加开启了“IPv6双栈”开关的CCE集群。

修改环境，请参考**修改环境**。

步骤2 重新部署应用，请参考**创建并部署组件**。

----结束

3.8 服务名重复校验范围是什么?

问题描述

服务名重复校验范围是什么?

解答

服务名重复校验范围是微服务名称、微服务应用、微服务版本和微服务环境。

是一个微服务的主键，标识一个唯一的微服务。

请确保主键不重复。

3.9 为什么一定要定义服务契约?

企业级系统规模普遍较大，微服务组件众多，所以对服务间接口进行统一管理是企业的关键需求。微服务引擎通过契约管理满足这一需求。

管理角度：通过契约管理，企业中的接口管理者可以统一定义微服务的契约文件（符合接口描述标准的接口定义文件），从而做到规范并协调多个开发团队的接口开发，降低沟通成本且避免后期的混乱。

开发角度：在微服务开发的时候，不同团队甚至不同ISV间，可以基于统一的契约文件开发同一应用或系统，从而方便整体系统一致性的维护。具体表现在，单体应用中模块间是代码级调用，在编译期就可以解决API不兼容问题，修复成本也极低。微服务解耦后，服务间变为了远程调用，接口不一致通常发现时间较晚，会造成更大的修复成本。有了契约可以使架构师根据契约文件严格审查变更，并反向生成代码，保证兼容性。

另外，对于规模较小、统一管理要求不高的系统，产品支持从接口代码自动生成契约文件。

3.10 如何解决微服务应用开发过程中微服务开发框架同 netty 版本不匹配的问题?

问题描述

开发微服务应用时，运行日志提示如下错误：

```
"Caused by: java.lang.NoSuchMethodError:
io.netty.handler.codec.http.websocketx.WebSocketClientHandshakerFactory.newHandshaker(Ljava/net/
URI;Lio/netty/handler/codec/http/websocketx/WebSocketVersion;Ljava/lang/String;ZLio/netty/handler/codec/
http/HttpHeaders;IZZJ)Lio/netty/handler/codec/http/websocketx/WebSocketClientHandshaker;"
```

原因分析

通常是由于某个第三方软件引入了不匹配的版本依赖。

解决方法

可在开发环境下使用**mvn dependency:tree**命令查看依赖树，排查微服务开发框架同netty版本是否匹配。

例如，ServiceComb 2.0.1开发框架所匹配的netty依赖版本为4.1.45.Final。

使用maven管理复杂依赖关系，请参考：**https://servicecomb.apache.org/cn/docs/maven_dependency_management/**。

3.11 ServiceComb 专享版引擎版本从 1.x 升级到 2.x 时有哪些注意事项?

ServiceComb引擎专享版从1.x升级到2.x的过程中及升级完以后可能会出现的现象及解决方法如下：

- **现象1**：在ServiceComb引擎专享版从1.x版本升级至2.x版本的过程中，使用接口获取配置或更新配置失败，报connection refused或Connection was closed，出现错误信息示例如下：

  ```
  [ERROR] Config update from xxx.xxx.xxx.xx failed. Error message is [Connection refused: xxx.xxx.xxx.xx]. org.apache.servicecomb.config.client.ConfigCenterClient$ConfigRefresh.lambda$null $13(ConfigCenterClient.java:428)
  ```

  或

  ```
  [ERROR]Config update from xxx.xxx.xxx.xx failed. Error message is [Connection was closed]. org.apache.servicecomb.config.client.ConfigCenterClient$ConfigRefresh.lambda$null $13(ConfigCenterClient.java:428)
  ```

  解决方法：ServiceComb引擎专享版1.x版本升级至2.x版本时配置中心会有短暂的重启，重启期间获取配置或更新配置会报错断连。因此引擎升级过程中避免更新配置，升级完成后该问题即可解决。

- **现象2**：使用引擎版本为1.x配置中心接入的用户，无法使用“业务场景治理”功能。

  解决办法：由于引擎版本为2.x的配置中心换成了kie，需要将配置中心接入方式切换为kie，具体切换方式详见**Spring Cloud使用配置中心**中相关内容。

- **现象3**：在使用版本为2.x的ServiceComb引擎时，使用导入配置文件功能，存在原配置中心格式的文件无法导入，提示文件为空或者格式错误。

  解决办法：将配置文件的配置项格式修改为2.x引擎要求的配置文件格式，新的配置文件为json文件，内容格式如下：

  ```
  {
    "data":[
      {
        "key":"xxx",
        "labels":{
          "environment":"xxx",
      "service": "xxx",
      "app": "xxx",
      "version": "xxx"
        },
        "value":"xxx",
        "value_type":"text",
        "status":"enabled"
      },
      {
        "key":"xxx",
        "labels":{
  ```

```
            "environment":"xxx"
         },
         "value":"xxx",
         "value_type":"text",
         "status":"enabled"
      },
      {
         "key":"xxx",
         "labels":{
            "environment":"xxx",
      "service": "xxx"
         },
         "value":"xxx",
         "value_type":"text",
         "status":"enabled"
      },
      {
         "key":"xxx",
         "labels":{
            "environment":"xxx",
      "service": "xxx",
      "app": "xxx"
         },
         "value":"xxx",
         "value_type":"text",
         "status":"enabled"
      }
   ]
}
```

其中：

- key和value是配置项对应的键和值，其为必填。
- labels是配置范围，其为必填，通过填写environment，service，app，version等字段来确定配置范围。
- value_type是配置项类型，其为必填，可以选择ini、json、text、yaml、properties、xml，默认为text。
- status是配置是否启用，其为选填，可以选择enabled（开启），disabled（关闭），默认关闭。

- **现象4**：若在ServiceComb引擎1.x版本的配置中心设置了全局配置，当升级到2.x之后，全局配置根据配置中心升级后的范围会相应的自动调整作用范围environment=${environmentName}，environmentName取值可以为空、development、testing、acceptance或production。此时如果SDK调整以kie作为配置中心时，需要在项目配置文件中增加自定义标签以获取该部分配置，以下以environment=production为例展示：

  spring-cloud-huawei框架：

  ```
  spring:
    cloud:
      servicecomb:
        config:
          serverType: kie
          kie:
            customLabel: environment
            customLabelValue: production
  ```

  servicecomb-java-chassis框架：

  ```
  servicecomb:
    kie:
      customLabel: environment
      customLabelValue: production
  ```


3.12 用户业务从微服务引擎专业版迁移到微服务引擎专享版checklist

迁移背景

微服务引擎专业版为逻辑多租引擎，所有用户共用一个引擎，一旦引擎故障，会造成所有注册业务的中断。为防止此类故障发生和保证业务的连续性，请将业务从微服务引擎专业版切换至商用微服务引擎专享版。

用户业务从微服务引擎专业版迁移至专享版后，将会有以下优势：

1. 物理隔离。微服务引擎专享版采用物理隔离的方式部署，租户独占微服务引擎。一个引擎故障不会影响其他引擎。
2. 多AZ。微服务引擎专享版支持多AZ部署，提升可靠性。
3. 容量大。单个微服务引擎专享版可支持2000实例，且可创建多个引擎，大大超过微服务引擎专业版（单租户500实例）。

用户从微服务引擎专业版（Cloud Service Engine）引擎切换至专享版（微服务引擎名称可自定义）后，引擎功能保持一致，对使用无影响，用户的配置、服务数据也将在切换过程中迁移到新引擎。

注意事项

迁移注意事项如下：

1. 首先需要确认是否有使用微服务引擎专业版，如果实例注册在名称为Cloud Service Engine的引擎上，说明使用了专业版。
2. 若用户业务要迁移至微服务引擎专享版引擎，则必须先创建微服务引擎专享版引擎。高可用引擎当前规格为100，500，2000，可以根据客户自身业务实例规模进行选择。
3. 新建微服务引擎所在VPC必须与待升级组件部署环境的VPC一致。
4. 迁移本质上是注册中心、配置中心地址的切换，切换前，所有服务注册到旧的微服务引擎；切换后，所有服务注册到新的微服务引擎；切换过程中，部分微服务注册到新引擎，部分微服务注册到旧引擎，这两部分的微服务无法进行服务发现和调用，可能导致业务不可用。
5. 确认部署方式，如果使用ServiceStage应用托管进行部署，可以联系运维人员获取快速迁移方案；如果未使用应用托管需要用户自己去修改配置中心以及注册中心的地址为新的专享版引擎的配置地址以及注册发现地址。若未使用ServiceStage应用托管，因为部署方式的多样性，建议迁移之前拉运维人员进行风险评估，并确认可靠的迁移方案。
6. 迁移不仅仅是实例的迁移，也包括配置的迁移，所以需要提前备份配置中心数据，可以联系运维人员进行协助。配置迁移包括动态配置迁移以及全局配置迁移。全局配置：在专业版控制台，逐个切环境，查看是否有全局配置，若存在，需要导出进行备份。动态配置：若一个微服务没有动态配置，则可忽略该服务。若一个微服务下的某个作用域没有动态配置，则可忽略该作用域。若存在配置则可以导出动态配置并保存。
7. 进行迁移前建议对各个微服务进行原地升级，确保无平台外因素导致升级失败。
8. 排查是否涉及JAVA_ARGS参数，若存在需要检查是否存在以下内容：

```
spring.cloud.servicecomb.discovery.address
spring.cloud.servicecomb.credentials.enabled
```

```
spring.cloud.servicecomb.credentials.accessKey
spring.cloud.servicecomb.credentials.secretKey
spring.cloud.servicecomb.credentials.akskCustomCipher
spring.cloud.servicecomb.credentials.project
```

若存在，迁移后将不再需要，可以删除。

9. 如果专享版引擎版本为2.x以上版本，使用导入配置文件功能，发现原配置中心格式的文件无法导入，提示文件为空或者格式错误，请参考**ServiceComb专享版引擎版本从1.x升级到2.x时有哪些注意事项?** 中相应操作进行处理。

3.13 创建引擎提示“Duplicate cluster name”

异常消息

创建引擎时，报错提示：

```
{"error_code":"SVCSTG.00500500","error_message":"{\"kind\":\"Status\":\"Failure\"...\"message\":\"duplicate cluster name\",\"reason\":\"Conflict\"}"}
```

问题原因

CSE引擎依赖的CCE服务出现问题，此时引擎的创建、删除、配置等功能都将不可用。

解决方案

可联系运维人员进行解决。

3.14 创建引擎过程中处理接入地址步骤失败，提示 the subnet could not be found

问题描述

创建引擎过程中，处理接入地址步骤失败，报错提示：

```
{"error_code":"SVCSTG.00500404","error_message":"{"code":"VPC.0202","message":"Query resource by id xxx  fail.the subnet could not be found."}"}
```

问题原因

用户的项目未对CSE云服务进行委托授权。

解决方案

- 当您使用从ServiceStage发放的微服务引擎实例时，如想在CSE中发放新实例，需要对CSE云服务进行授权，具体操作可参考**授权使用微服务引擎**。
- 当您没有授予任何权限时，由于CSE使用依赖VPC云服务，因此需要先参考**创建委托**创建云服务委托cse_admin_trust，将操作权限委托给CSE。

3.15 本地轻量化注册服务报错：does not match rule: {Max: 100, Regexp: ^[a-zA-Z0-9]{1,160}$|^[a-zA-Z0-9][a-zA-Z0-9_\-.]{0,158}[a-zA-Z0-9]$}"}

问题描述

本地轻量化注册服务报错：does not match rule: {Max: 100, Regexp: ^[a-zA-Z0-9]{1,160}$|^[a-zA-Z0-9][a-zA-Z0-9_\-.]{0,158}[a-zA-Z0-9]$}"}

本地注册服务报错日志：

register failed, will retry. please check config file. message=read response failed. status:400; message:Bad Request; content:{"errorCode":"400001","errorMessage":"Invalid parameter(s)","detail":"field 'MicroService.Schemas' invalid value '[MogoManagerController ResExpandFollowController PurPurchaseController DevSupplierClassController BilTransferConfirmController RptRentAccountController ApprovalProjectParaController RptOperateInfoController ResThemeController ExportController SysBannerController SysBillingMethodController ReportSourceController DevMaintainController ReportInvoiceController SysPushMsgController ResHouseTypeController TestCotroller UnionPayController WebMvcEndpointHandlerMapping_WebMvcLinksHandler ResSourceConfigureController BizCheckOutUserController BilInterfaceMsgController BizSettleController SysInterfaceMsgController RptBigDataController ResRepairController ContCheckInHouseController SysBranchCodeController ResProjectHotController ContRentLadderController ContContractSourceRelController ReportDepositController RptWeeklyController GlobalExceptionController PurPayDetailController JobController ContShortContractController ResCostConfigureDetailControl

does not match rule: {Max: 100, Regexp: ^[a-zA-Z0-9]{1,160}$|^[a-zA-Z0-9][a-zA-Z0-9_\-.]{0,158}[a-zA-Z0-9]$}"}

问题原因

本地轻量化默认Schemas：100个配额，达到配额上限。

解决方案

修改本地轻量化工具包解压后的文件中的“/conf/app.conf”文件中的quota_plugin="unlinit"。

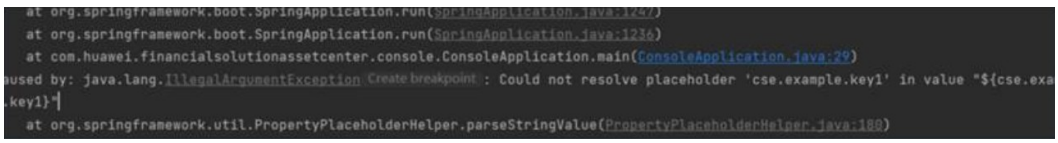
说明

其他配额不足的时候也是在这个文件中修改。

3.16 SpringCloud 应用连接 ServiceComb 引擎 2.x 版本配置中心失败

问题描述

代码中配置了配置中心但是无法获取到配置项。



问题原因

配置文件中配置中心的类型和地址不正确。

排查过程

1. 引擎使用的2.x版本但是配置中心使用的是1.x版本的key。
2. 配置文件指定了环境，但是代码中获取的时候配置环境的key不正确。

解决方案

1. 将配置文件中配置中心的类型改为kie，即 spring.cloud.servicecomb.config.serverType: kie。
2. 在配置文件中修改环境的key，即参数server.env。

3.17 在全局配置中配置相关的配置项修改后，服务获取的配置内容未修改成功

问题现象

全局配置修改后未生效，日志中未打印关键字changed。

问题原因

没有配置config-cc的依赖。

排查过程

1. 在application.yaml中查看是否有该配置中心的地址。
2. 在pom.xml文件中是否有config-cc的依赖。

解决方案

在pom.xml文件中添加config-cc的依赖。

```
<dependency>
 <groupId>org.apache.servicecomb</groupId>
 <artifactId>config-cc</artifactId>
</dependency>
```

3.18 获取配置失败

问题现象

微服务在接入相应的微服务开发框架（如spring-cloud-huawei、java-chassis）后，微服务通过SDK调用查询配置接口到ServiceComb引擎获取配置项失败。

问题原因

微服务与注册中心间的连接因网络、CPU等其他因素发生抖动时，可能会导致请求异常。

解决方案

微服务框架具有自愈能力，拉取配置失败后会自动进行重试，一般情况下不会导致业务异常。您可以查看下次获取配置是否成功，若否，请联系技术支持人员。

4 应用网关

4.1 使用应用网关时出现 503 报错

问题现象

应用网关平稳运行过程中偶现503报错，且此过程中未变更网关配置。

错误码为503的具体报错信息为：upstream connect error or disconnect/reset before headers. reset reason: connection termination。

问题原因

应用网关侧的空闲连接保持时间与服务侧的不匹配。一般来说是服务侧的空闲连接保持时间比应用网关侧的时间长导致的503报错。

解决方案

1. 在连接池中配置idleTimeout，确保网关的idleTimeout数值略低于服务侧的空闲连接保持时间。比如：服务的idletimeout是20s，网关设置15s即可。具体操作请参考**连接池配置**。

   以服务侧的网络代理为Tomcat为例：

   Tomcat的两个参数keepAliveTimeout和connectionTimeout配置的值都不能小于应用网关中配置的idleTimeout值，配置说明请参考**Apache Tomcat 9 Configuration Reference**。

   说明

   配置Tomcat两个参数值，有两种场景：

   - 当为独立的springboot.jar包时，在“application.properties”文件中配置。
   - 当为springboot.war包时，在“server.xml”文件中配置。

2. 在路由中配置重试策略，具体操作请参考**配置重试策略**。

   说明

   重试只对幂等请求生效。比如：POST请求无法触发重试策略。

