多活高可用服务

最佳实践

文档版本 05

发布日期 2024-04-12

华为云计算技术有限公司

版权所有 © 华为云计算技术有限公司 2024。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明

HUAWEI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。

目 录

1 MAS 最佳实践-商城应用改造.....1
1.1 应用场景.....1
1.2 资源准备.....1
1.3 配置 POM.xml 和 yaml 文件.....2
1.4 依赖冲突问题修复.....3
1.5 SDK 相关功能验证.....8
1.6 MySQL 监控切换验证.....12
2 同城多活改造实践.....13
2.1 方案概述.....13
2.2 资源和成本规划.....14
2.3 操作流程.....15
2.4 实施步骤.....16
2.4.1 添加数据源.....16
2.4.2 创建同步链路.....17
2.4.3 创建应用.....18
2.4.4 创建 MySQL 监控.....18
2.4.5 应用接入 SDK.....19
2.4.6 故障切换演练.....21
3 事件流编排.....22
3.1 方案概述.....22
3.2 操作流程.....22
3.3 实施步骤.....23
3.3.1 创建工作流.....23
3.3.2 编排工作流.....24
3.3.3 导入工作流.....28
3.3.4 执行工作流.....29
3.3.5 查看工作流执行详情.....31
3.3.6 复制工作流.....33
3.3.7 删除工作流.....34
3.3.8 关注工作流.....34

1 MAS 最佳实践-商城应用改造

1.1 应用场景
1.2 资源准备
1.3 配置POM.xml和yaml文件
1.4 依赖冲突问题修复
1.5 SDK相关功能验证
1.6 MySQL监控切换验证

1.1 应用场景

本次商城应用改造是基于github上50k+star的mall后台管理系统及对应前端项目进行改造来对接SDK。

后端项目地址：**https://github.com/macrozheng/mall**

前端项目地址：**https://github.com/macrozheng/mall-admin-web**

与SDK对接请参考**MAS-DB-SDK使用手册**。

1.2 资源准备

mall项目对接SDK需要提前准备的资源包括：

1. 多活实例，请参考**多活管理**，创建多活实例。
2. 两个数据库，请参考**购买RDS实例**，购买MySQL数据库。
3. 配置MySQL监控，请参考**MySQL监控管理**。

准备好上述资源后便可开始mall项目的SDK对接工作。

版权所有 © 华为云计算技术有限公司

1.3 配置 POM.xml 和 yaml 文件

配置 yaml 文件

1. 登录MAS服务控制台，在“多活管理”页面单击实例，进入实例详情控制台。
2. 在页面顶端导航栏选择“监控列表”，单击MySQL监控所在行的“更多>SDK接入配置”，获取SDK接入配置。

说明

此处获取的yaml配置不会携带MySQL用户名密码信息，在使用时，需要补充完全（与创建MySQL监控时数据中心1和数据中心2数据库用户名和密码对应），同时还需要保证ETCD地址的可用性，再添加至项目的yaml文件。

3. 获取到yaml文件配置后，将相关信息添加完毕，找到mall项目对应的yaml文件（mall-admin模块下的yaml文件），将相关信息添到mall项目的yaml文件，并且进行修改。由于mall后台项目自带springboot-dataSource，需要将其删除替换。修改后yaml文件如下图所示：

```
spring:
  redis:
    host:
    database: 0
    port: 6379
    password:
    timeout: 300ms # 连接超时时间（毫秒）
```


版权所有 © 华为云计算技术有限公司

```
devspore:
  datasource:
    props:
      version: v1
      appId:
      monitorId:
      databaseName: demo
      decipherClassName: com.huawei.devspore.mas.password.DefaultDecipher
      region: az0
    etcd:
      address:
      apiversion: v3
      username:
      password:
      https-enable: true
    sources:
      ds1:
        driverClassName: com.mysql.cj.jdbc.Driver
        jdbcUrl:
        username:
        password:
        type: com.zaxxer.hikari.HikariDataSource
      ds2:
        driverClassName: com.mysql.cj.jdbc.Driver
        jdbcUrl:
        username:
        password:
        type: com.zaxxer.hikari.HikariDataSource

    router:
      active: dc1
      routeAlgorithm: single-read-write
      retry:
```

配置 POM.xml 文件

配置POM.xml文件（最外层POM文件），需要添加以下几项依赖：

```
<mas.version>1.1.13-SNAPSHOT</mas.version>
```

```
<dependency>
    <groupId>com.huaweicloud.devspore</groupId>
    <artifactId>spring-cloud-starter-huawei-devspore-datasource</artifactId>
    <version>${mas.version}</version>
    <exclusions...>
</dependency>
```

```
<dependency>
    <groupId>com.zaxxer</groupId>
    <artifactId>HikariCP</artifactId>
    <version>4.0.3</version>
</dependency>
```

说明

由于项目自带MySQL数据库驱动，故此处无需再添加。

1.4 依赖冲突问题修复

完成相关配置后（请确保各个配置项无误且功能正常），直接运行程序会存在依赖冲突问题，下面进行逐一修复（以下问题的修复是相关联的，上一个问题修复后重新运行而产生的后续问题）。

问题一：Logback-classic及log4j-to-slf4j的包冲突问题。

版权所有 © 华为云计算技术有限公司

报错如下：

```
"C:\Program Files\Huawei\jdk1.8.0_272\bin\java.exe" ...
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/Users/wwx1122317/.m2/repository/org/apache/logging/log4j/log4j-slf4j-impl/2.13.2/log4j-slf4j-impl-2.13.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/C:/Users/wwx1122317/.m2/repository/ch/qos/logback/logback-classic/1.2.3/logback-classic-1.2.3.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Exception in thread "main" java.lang.ExceptionInInitializerError
    at com.macro.mall.MallAdminApplication.main(MallAdminApplication.java:13)
Caused by: org.apache.logging.log4j.LoggingException: log4j-slf4j-impl cannot be present with log4j-to-slf4j
    at org.apache.logging.slf4j.Log4jLoggerFactory.validateContext(Log4jLoggerFactory.java:49)
    at org.apache.logging.slf4j.Log4jLoggerFactory.newLogger(Log4jLoggerFactory.java:39)
    at org.apache.logging.slf4j.Log4jLoggerFactory.newLogger(Log4jLoggerFactory.java:30)
    at org.apache.logging.log4j.spi.AbstractLoggerAdapter.getLogger(AbstractLoggerAdapter.java:54)
    at org.apache.logging.slf4j.Log4jLoggerFactory.getLogger(Log4jLoggerFactory.java:30)
    at org.slf4j.LoggerFactory.getLogger(LoggerFactory.java:363)
    at org.apache.commons.logging.LogAdapter$Slf4jAdapter.createLocationAwareLog(LogAdapter.java:130)
    at org.apache.commons.logging.LogAdapter.createLog(LogAdapter.java:91)
    at org.apache.commons.logging.LogFactoryService.getInstance(LogFactoryService.java:46)
    at org.apache.commons.logging.LogFactoryService.getInstance(LogFactoryService.java:41)
    at org.apache.commons.logging.LogFactory.getLog(LogFactory.java:685)
    at org.springframework.boot.SpringApplication.<clinit>(SpringApplication.java:196)
    ... 1 more
```

问题原因：

spring-boot-starter-actuator与spring-cloud-starter-huawei-devspore-datasource的Logback-classic及log4j-to-slf4j包存在冲突。

解决办法：

排除相关冲突包，排除后POM文件如下：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
    <exclusions>
        <exclusion>
            <artifactId>logback-classic</artifactId>
            <groupId>ch.qos.logback</groupId>
        </exclusion>
        <exclusion>
            <artifactId>log4j-to-slf4j</artifactId>
            <groupId>org.apache.logging.log4j</groupId>
        </exclusion>
    </exclusions>
</dependency>
```

问题二：Property 'sqlSessionFactory' or 'sqlSessionTemplate' are required

报错如下：

```
Caused by: java.lang.IllegalArgumentException: Property 'sqlSessionFactory' or 'sqlSessionTemplate' are required
    at org.springframework.util.Assert.notNull(Assert.java:198) ~[spring-core-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.mybatis.spring.support.SqlSessionDaoSupport.checkDaoConfig(SqlSessionDaoSupport.java:122) ~[mybatis-spring-2.0.5.jar:2.0.5]
    at org.mybatis.spring.mapper.MapperFactoryBean.checkDaoConfig(MapperFactoryBean.java:73) ~[mybatis-spring-2.0.5.jar:2.0.5]
    at org.springframework.dao.support.DaoSupport.afterPropertiesSet(DaoSupport.java:44) ~[spring-tx-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.invokeInitMethods(AbstractAutowireCapableBeanFactory.java:1855) ~[spring-beans-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.initializeBean(AbstractAutowireCapableBeanFactory.java:1792) ~[spring-beans-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:595) ~[spring-beans-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:517) ~[spring-beans-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:323) ~[spring-beans-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:226) ~[spring-beans-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:321) ~[spring-beans-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:202) ~[spring-beans-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.springframework.beans.factory.config.DependencyDescriptor.resolveCandidate(DependencyDescriptor.java:276) ~[spring-beans-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.springframework.beans.factory.support.DefaultListableBeanFactory.doResolveDependency(DefaultListableBeanFactory.java:1306) ~[spring-beans-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.springframework.beans.factory.support.DefaultListableBeanFactory.resolveDependency(DefaultListableBeanFactory.java:1226) ~[spring-beans-5.2.6.RELEASE.jar:5.2.6.RELEASE]
    at org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor$AutowiredFieldElement.inject(AutowiredAnnotationBeanPostProcessor.java:640) ~[spring-beans-5.2.6.RELEASE.jar:5.2.6.RELEASE]
```

问题原因：

缺少相关依赖包：mybatis-spring-boot-starter。

解决办法：

添加相应依赖包，并去除新加入的包的冲突项，完成后POM如下图所示：

版权所有 © 华为云计算技术有限公司

```
<dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>1.3.2</version>
    <exclusions>
        <exclusion>
            <artifactId>logback-classic</artifactId>
            <groupId>ch.qos.logback</groupId>
        </exclusion>
        <exclusion>
            <artifactId>log4j-to-slf4j</artifactId>
            <groupId>org.apache.logging.log4j</groupId>
        </exclusion>
    </exclusions>
</dependency>
```

问题三：Failed to configure a DataSource: 'url' attribute is not specified and no embedded datasource could be configured

报错如下：

```
2022-03-30 10:47:44.447 ERROR 22952 --- [          main] o.s.b.w.e.t.TomcatStarter                : Error starting Tomcat context. Exception: org.springframework.beans.factory.UnsatisfiedDependencyException. Message: Error c
2022-03-30 10:47:44.685  INFO 22952 --- [          main] o.a.c.c.StandardService                  : Stopping service [Tomcat]
2022-03-30 10:47:44.701  WARN 22952 --- [          main] ConfigServletWebServerApplicationContext : Exception encountered during context initialization - cancelling refresh attempt: org.springframework.context.ApplicationCon
2022-03-30 10:47:44.722  INFO 22952 --- [          main] ConditionEvaluationReportLoggingListener :

Error starting ApplicationContext. To display the conditions report re-run your application with 'debug' enabled.
2022-03-30 10:47:44.726 ERROR 22952 --- [          main] o.s.b.d.LoggingFailureAnalysisReporter   :

***************************
APPLICATION FAILED TO START
***************************

Description:

Failed to configure a DataSource: 'url' attribute is not specified and no embedded datasource could be configured.

Reason: Failed to determine a suitable driver class


Action:

Consider the following:
    If you want an embedded database (H2, HSQL or Derby), please put it on the classpath.
    If you have database settings to be loaded from a particular profile you may need to activate it (no profiles are currently active).


Process finished with exit code 1
```

问题原因：

原生的springboot-dataSource被替换，一些依赖包未找到相应的yaml对应的数据源配置而报错。

解决办法：

删除相关依赖包，使其能对应SDK提供的dataSource，过程如下：

- 删除或注释最外层的POM文件的druid-spring-boot-starter依赖项，如下图所示：

```
        <!--集成druid连接池-->
<!--        <dependency>-->
<!--            <groupId>com.alibaba</groupId>-->
<!--            <artifactId>druid-spring-boot-starter</artifactId>-->
<!--            <version>${druid.version}</version>-->
<!--        </dependency>-->
```

- 删除或注释mall-mbg模块POM文件的druid-spring-boot-starter依赖项，如下图所示：

版权所有 © 华为云计算技术有限公司

```
<!--        <dependency>-->
<!--            <groupId>com.alibaba</groupId>-->
<!--            <artifactId>druid-spring-boot-starter</artifactId>-->
<!--        </dependency>-->
```

问题四：io.netty.buffer.PooledByteBufAllocator.<init>(ZIIIIIIZ)V

报错如下：

```
2022-03-30 11:24:00.765 ERROR 18476 --- [ault-executor-0] i.g.i.ManagedChannelImpl                : [Channel<1>: (ip:///100.85.116.241:2379,100.85.126.7:2379,100.95.144

java.lang.NoSuchMethodError Create breakpoint : io.netty.buffer.PooledByteBufAllocator.<init>(ZIIIIIIZ)V
    at io.grpc.netty.Utils.createByteBufAllocator(Utils.java:172) ~[grpc-netty-1.37.0.jar:1.37.0]
    at io.grpc.netty.Utils.access$000(Utils.java:71) ~[grpc-netty-1.37.0.jar:1.37.0]
    at io.grpc.netty.Utils$ByteBufAllocatorPreferDirectHolder.<clinit>(Utils.java:93) ~[grpc-netty-1.37.0.jar:1.37.0]
    at io.grpc.netty.Utils.getByteBufAllocator(Utils.java:140) ~[grpc-netty-1.37.0.jar:1.37.0]
    at io.grpc.netty.NettyClientTransport.start(NettyClientTransport.java:231) ~[grpc-netty-1.37.0.jar:1.37.0]
    at io.grpc.internal.ForwardingConnectionClientTransport.start(ForwardingConnectionClientTransport.java:33) ~[grpc-core-1.37.0.jar:1.37.0]
    at io.grpc.internal.ForwardingConnectionClientTransport.start(ForwardingConnectionClientTransport.java:33) ~[grpc-core-1.37.0.jar:1.37.0]
    at io.grpc.internal.InternalSubchannel.startNewTransport(InternalSubchannel.java:258) ~[grpc-core-1.37.0.jar:1.37.0]
    at io.grpc.internal.InternalSubchannel.access$400(InternalSubchannel.java:65) ~[grpc-core-1.37.0.jar:1.37.0]
    at io.grpc.internal.InternalSubchannel$2.run(InternalSubchannel.java:200) ~[grpc-core-1.37.0.jar:1.37.0]
    at io.grpc.SynchronizationContext.drain(SynchronizationContext.java:95) ~[grpc-api-1.37.0.jar:1.37.0]
    at io.grpc.SynchronizationContext.execute(SynchronizationContext.java:127) ~[grpc-api-1.37.0.jar:1.37.0]
    at io.grpc.internal.ManagedChannelImpl$NameResolverListener.onResult(ManagedChannelImpl.java:1827) ~[grpc-core-1.37.0.jar:1.37.0]
    at io.grpc.NameResolver$Listener2.onAddresses(NameResolver.java:208) ~[grpc-api-1.37.0.jar:1.37.0]
    at io.etcd.jetcd.resolver.IPNameResolver.doResolve(IPNameResolver.java:150) ~[jetcd-core-0.5.7.jar:?] <3 internal lines>
```

并且每一段时间会跳出报错信息：

```
2022-03-30 11:24:11.108  WARN 18476 --- [pool-5-thread-1] c.h.d.d.c.RemoteConfigurationWatcher     : etcd error: Panic! This is a bug!
2022-03-30 11:24:12.111 ERROR 18476 --- [ault-executor-0] i.g.i.ManagedChannelImpl                : [Channel<45>: (ip:///100.85.116.241:2379,100.85.126.7:2379,100.95.144.201:2379)] Uncaught exception in the SynchronizationCon

java.lang.NoClassDefFoundError Create breakpoint : Could not initialize class io.grpc.netty.Utils$ByteBufAllocatorPreferDirectHolder
    at io.grpc.netty.Utils.getByteBufAllocator(Utils.java:140) ~[grpc-netty-1.37.0.jar:1.37.0]
    at io.grpc.netty.NettyClientTransport.start(NettyClientTransport.java:231) ~[grpc-netty-1.37.0.jar:1.37.0]
    at io.grpc.internal.ForwardingConnectionClientTransport.start(ForwardingConnectionClientTransport.java:33) ~[grpc-core-1.37.0.jar:1.37.0]
    at io.grpc.internal.ForwardingConnectionClientTransport.start(ForwardingConnectionClientTransport.java:33) ~[grpc-core-1.37.0.jar:1.37.0]
    at io.grpc.internal.InternalSubchannel.startNewTransport(InternalSubchannel.java:258) ~[grpc-core-1.37.0.jar:1.37.0]
    at io.grpc.internal.InternalSubchannel.access$400(InternalSubchannel.java:65) ~[grpc-core-1.37.0.jar:1.37.0]
    at io.grpc.internal.InternalSubchannel$2.run(InternalSubchannel.java:200) ~[grpc-core-1.37.0.jar:1.37.0]
    at io.grpc.SynchronizationContext.drain(SynchronizationContext.java:95) ~[grpc-api-1.37.0.jar:1.37.0]
    at io.grpc.SynchronizationContext.execute(SynchronizationContext.java:127) ~[grpc-api-1.37.0.jar:1.37.0]
    at io.grpc.internal.ManagedChannelImpl$NameResolverListener.onResult(ManagedChannelImpl.java:1827) ~[grpc-core-1.37.0.jar:1.37.0]
    at io.grpc.NameResolver$Listener2.onAddresses(NameResolver.java:208) ~[grpc-api-1.37.0.jar:1.37.0]
    at io.etcd.jetcd.resolver.IPNameResolver.doResolve(IPNameResolver.java:150) ~[jetcd-core-0.5.7.jar:?] <3 internal lines>
```

问题原因：

Io.netty相关依赖包版本冲突问题。SDK的依赖包spring-cloud-starter-huawei-devspore-datasource所需的io.netty版本相对本项目的版本更高。项目的io.netty相关依赖包为4.1.49.Final，需升级为4.1.69.Final。

解决办法：

将io.netty的相关依赖包先进行排除，后统一引入4.1.69Final版本。

1. 排除4.1.49.Final版本的io.netty相关依赖：

版权所有 © 华为云计算技术有限公司

```
<dependency>
    <groupId>com.huaweicloud.devspore</groupId>
    <artifactId>spring-cloud-starter-huawei-devspore-datasource</artifactId>
    <version>${mas.version}</version>
    <exclusions>
        <exclusion>
            <artifactId>netty-buffer</artifactId>
            <groupId>io.netty</groupId>
        </exclusion>
        <exclusion>
            <artifactId>netty-common</artifactId>
            <groupId>io.netty</groupId>
        </exclusion>
        <exclusion>
            <artifactId>netty-transport</artifactId>
            <groupId>io.netty</groupId>
        </exclusion>
        <exclusion>
            <artifactId>netty-tcnative-boringssl-static</artifactId>
            <groupId>io.netty</groupId>
        </exclusion>
        <exclusion>
            <artifactId>netty-codec</artifactId>
            <groupId>io.netty</groupId>
        </exclusion>
    </exclusions>
</dependency>
```

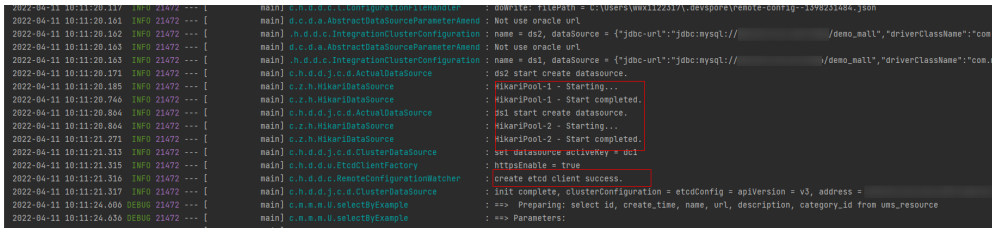
2. 添加4.1.69.Final版本：

```
<dependency>
    <groupId>io.netty</groupId>
    <artifactId>netty-tcnative-boringssl-static</artifactId>
    <version>2.0.39.Final</version>
</dependency>
<dependency>
    <groupId>io.netty</groupId>
    <artifactId>netty-common</artifactId>
    <version>4.1.69.Final</version>
</dependency>
<dependency>
    <groupId>io.netty</groupId>
    <artifactId>netty-buffer</artifactId>
    <version>4.1.69.Final</version>
</dependency>
<dependency>
    <groupId>io.netty</groupId>
    <artifactId>netty-transport</artifactId>
    <version>4.1.69.Final</version>
</dependency>
```

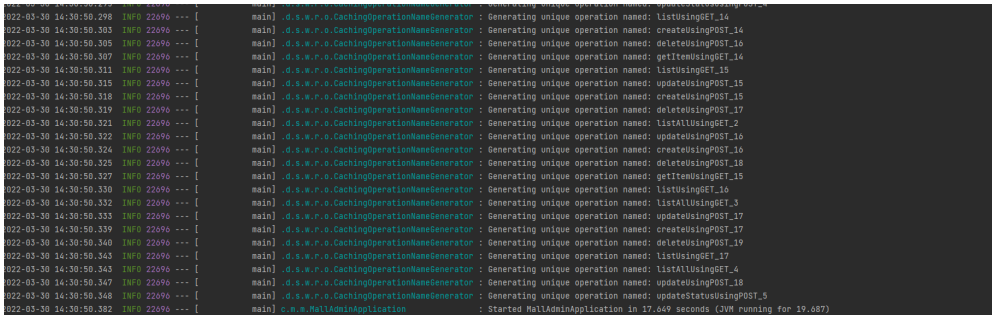
按照如上的流程解决完上述四个问题，重新运行项目便可成功对接SDK。

创建ETCD client success且数据源初始化成功：

版权所有 © 华为云计算技术有限公司



程序运行成功：



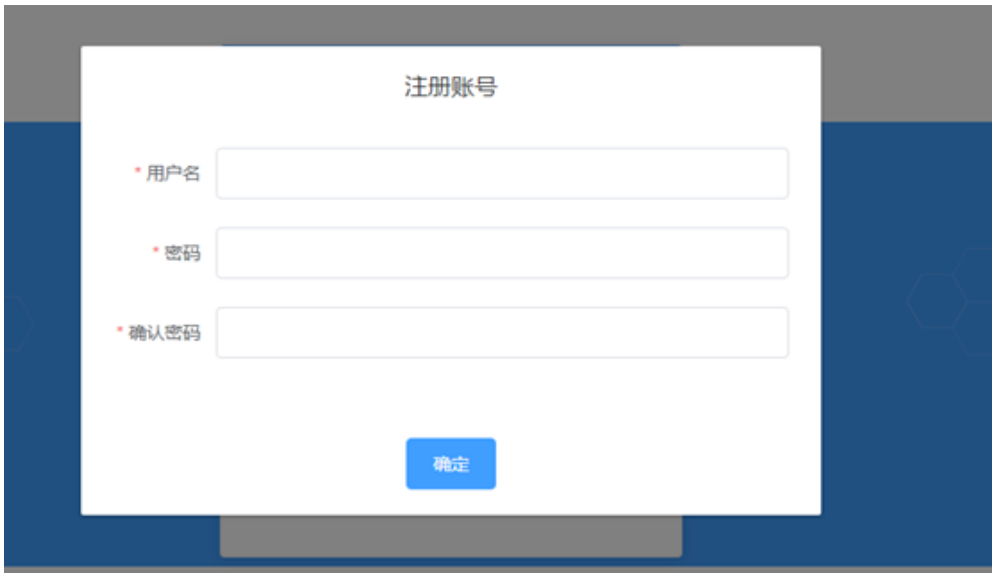
1.5 SDK 相关功能验证

完成SDK对接，并成功运行后，需要测试SDK功能是否正常实现（由于在本地环境，暂时连接不了ETCD实现自动切换，切换操作由手动实现，同时由于测试需要，本演示项目在前端添加了一个“注册管理员”按钮用于注册管理员）。

数据库查询：数据库查询的表为管理员表 ，位于demo-mall数据库下的ums-admim（需要创建一个名为demo-mall的数据库，可使用github后端项目中的document文件夹下的sql文件直接创建相关表。）

管理员注册：单击注册管理员按钮，填写用户名和密码即可。

版权所有 © 华为云计算技术有限公司



SDK 连接数据库测试

默认连接数据库为ds1，可在yaml文件中配置。此处调用注册管理员接口，查看是否成功修改SDK。

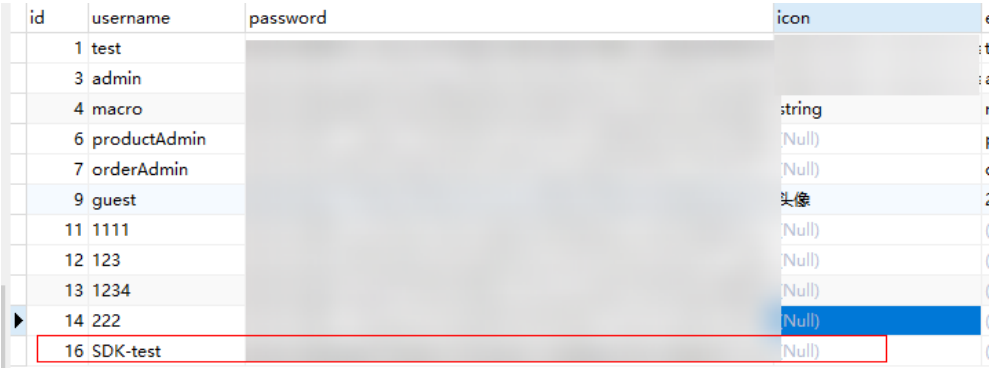
1. 通过Navicat查看当前管理员表的用户数据（当前为ds1对应数据库）。

id	username	password	icon
1	test		
3	admin		
4	macro		string
6	productAdmin		(Null)
7	orderAdmin		(Null)
9	guest		头像
11	1111		(Null)
12	123		(Null)
13	1234		(Null)
14	222		(Null)

2. 通过mall前端页面进行注册，注册用户名为“SDK-test”。

版权所有 © 华为云计算技术有限公司

3. 查看注册后数据库ds1管理员表的用户数据。



id	username	password	icon
1	test		
3	admin		
4	macro		string
6	productAdmin		(Null)
7	orderAdmin		(Null)
9	guest		头像
11	1111		(Null)
12	123		(Null)
13	1234		(Null)
14	222		(Null)
16	SDK-test		(Null)

结论：数据库ds1新增“SDK-test”用户名，SDK提供的服务正常运行。

SDK 切换数据源测试

验证SDK提供的切换数据源功能，可在yaml直接配置当前活跃的数据源，使用之前已注册的“SDK-test”用户名进行登录验证。

1. 切换数据源为ds2，切换配置如下：

版权所有 © 华为云计算技术有限公司

```yaml
router:
  active: dc2
  routeAlgorithm: single-read-write
  retry:
    times: 3  # 重试次数，推荐次数3次
    delay: 5000  # 重试时延，推荐时延5000毫秒
  nodes:
    dc1:
      master: ds1
    dc2:
      master: ds2
```

切换后连接数据库为ds2，管理员表其对应用户数据如下：

id	username	password	icon
1	test		
3	admin		
4	macro		string
6	productAdmin		(Null)
7	orderAdmin		(Null)
9	123		(Null)

2. 使用“SDK-test”用户名和密码通过mall前端页面进行登录操作。

登录时提示用户名错误，无法登录，因为数据库ds2对应表中无“SDK-test”用户信息。

3. 使用数据库ds2中其他已有用户名和密码进行登录。

   可以登录成功。

结论：SDK切换数据库成功。

版权所有 © 华为云计算技术有限公司

1.6 MySQL 监控切换验证

根据**1.2 资源准备**所准备的实例和MySQL监控，使用之前已注册的“SDK-test”用户名登录mall商城，验证MySQL监控切换数据库功能。

1. 使用“SDK-test”用户名和密码登录mall商城。

   根据**SDK切换数据源测试**，将数据源切换为ds1，“SDK-test”用户登录成功。通过管理员账号登录mall商城后台，可以发现用户列表表单中有“SDK-test”这个用户。
2. 登录MAS控制台，在“多活管理”页面单击实例，进入实例控制台。
3. 在页面顶端导航栏选择“监控列表”，进入“监控列表”页面，单击MySQL监控所在行右侧的“切换”。
4. 在弹窗中单击“确认”。活跃数据库由数据中心1变为数据中心2。
5. 再次使用“SDK-test”用户名和密码登录mall商城。

   此时“SDK-test”用户登录失败。再次通过管理员账号登录mall商城后台，可以发现用户列表表单中已没有“SDK-test”这个用户。
6. 单击监控所在行右侧的“回切”按钮。
7. 在弹窗中单击“确认”。活跃数据库由数据中心2变为数据中心1。
8. 再次使用“SDK-test”用户名和密码登录mall商城，可以发现“SDK-test”用户名再次登录成功。

结论：MySQL监控器切换数据中心成功。

版权所有 © 华为云计算技术有限公司

2 同城多活改造实践

2.1 方案概述
2.2 资源和成本规划
2.3 操作流程
2.4 实施步骤

2.1 方案概述

应用场景

企业业务快速成长，面临的挑战越来越多，对业务稳定可靠运行的需求越来越高。但设备故障、电力故障、人为损坏、火灾、自然灾害等突发事故会对业务的稳定运行造成巨大威胁，影响企业的正常运营，制约企业的发展，甚至可能给企业利益造成无法预估的损失。为了满足企业高可用的业务运行需求，企业应用多活部署已成为行业趋势。

多活高可用服务（Multi-Site High Availability Service，简称MAS）源自华为消费者多活应用高可用方案，提供从流量入口、数据到应用层的端到端的业务故障切换及容灾演练能力，保障故障场景下的业务快速恢复，提升业务连续性。

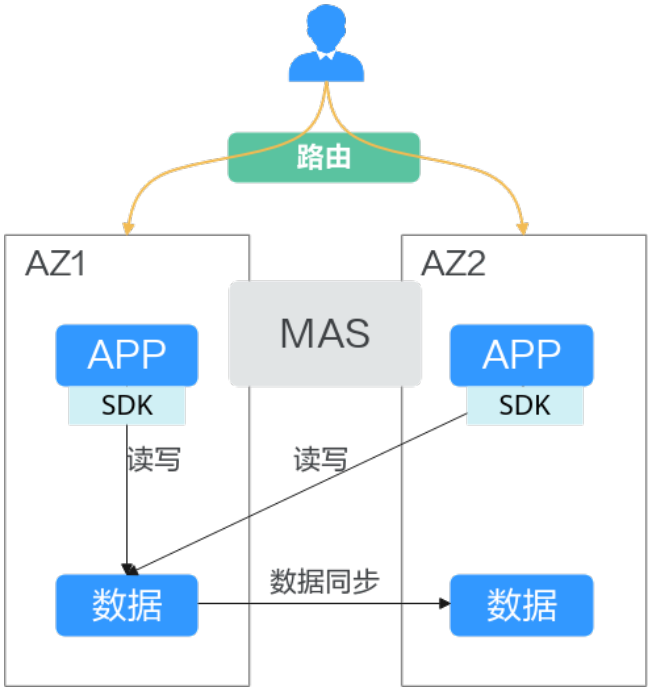
本手册基于一个**GitHub**上的商城应用改造实践，介绍如何基于MAS将应用改造为同城多活架构的过程。

方案架构

本实践对Demo商城的web应用和Mysql数据库进行同城多活改造。

通过MAS改造后的应用架构图如下所示：

版权所有 © 华为云计算技术有限公司



同城多活应用场景：

- 适用于容灾可用性要求达到99.95%，要求业务双活的容灾场景。
- 同城跨AZ的应用数据多活，并分担部分业务流量。

方案优势

同城多活架构优势如下：

1. 应用同城可用区多活，业务双边负载分担灵活。
2. 数据实时同步，服务故障跨AZ快速切换。
3. 秒级RTO、RPO，保证业务连续性。
4. 低成本容灾演练能力。
5. 业务改造工作量相对较低。

2.2 资源和成本规划

购买MAS多活管理中心及RDS for MySQL涉及到的资源和成本规划，如下表所示。

表 2-1 资源和成本规划

资源	资源说明	数量	费用（元）
虚拟私有云VPC	创建一个虚拟私有云VPC	1	00.00
虚拟私有云子网	创建一个虚拟私有云子网	1	00.00
安全组	创建一个安全组	1	00.00

版权所有 © 华为云计算技术有限公司

资源	资源说明	数量	费用（元）
多活管理中心	创建MAS铂金版实例	1	20.00/小时
模块数量	• MAS功能模块版本开通“同城多活” • 功能点开通“MySQL”	1	100.00/天
云数据库RDS for MySQL	• 2核 4GB • 数据库版本5.7 • 选择按需计费	2	2.04/小时

须知

本文提供的成本预估费用仅供参考，资源的实际费用以华为云管理控制台显示为准。

2.3 操作流程

前提条件

进行应用改造需要提前完成的事项：

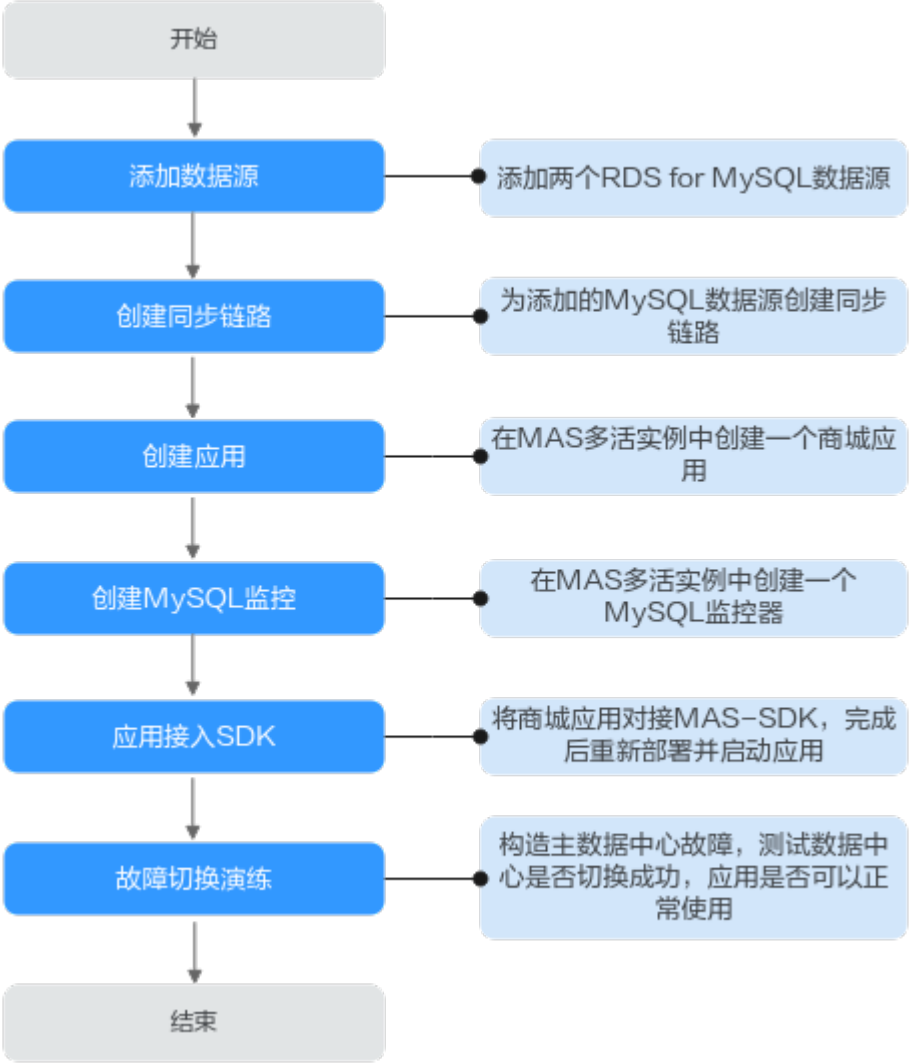
- 已开通MAS“同城多活”版本功能模块，功能点包括“MySQL”。具体详情请参见**开通功能模块**。
- 创建好“同城多活”类型的命名空间，分区类型选择“华为云”。具体详情请参见**创建命名空间**。
- 在已创建的命名空间下购买MAS多活管理中心，即创建好多活实例。具体详情请参见**购买多活管理中心**。
- 已在华为云**购买RDS for MySQL**两个数据库实例，且与MAS多活实例配置为相同VPC、子网、安全组，与多活实例网络互通；数据库实例需配置为相同的用户名和密码。
- 商城应用部署在已创建的命名空间下的可用区，且业务部署架构需要和MAS多活分区保持一致；与多活实例配置为相同VPC、子网、安全组，与多活实例网络互通。

改造流程

通过MAS多活实例进行商城应用同城多活改造流程如下图所示：

版权所有 © 华为云计算技术有限公司

图 2-1 商城应用改造流程



2.4 实施步骤

2.4.1 添加数据源

1. 登录MAS控制台，进入“数据管理>数据源”页面，选择前提条件中已创建好的命名空间，单击右上角的“添加数据源”。
2. 在“添加数据源”页面，填写配置信息，然后单击“提交”。

表 2-2 数据源配置参数

参数	配置说明
所属分区	选择数据源所属分区，配置为命名空间下的主多活分区。
数据源类型	选择数据源类型，本实践选择“MySQL”。
数据源名称	默认自动生成，或者根据规划自定义填写。

版权所有 © 华为云计算技术有限公司

参数	配置说明
描述	填写数据源的描述信息。
链接模式	选择链接模式，本实践选择“RDS”。
RDS列表	在列表中选择已有的RDS。此RDS为将配置为创建同步链路中的源数据源。
用户名	填写数据库用户名。
密码	填写数据库用户密码。
确认密码	再次填写数据库用户密码。

3. 重复步骤2，“所属分区”配置为命名空间下的备多活分区，将另一个RDS实例也添加至数据源。

2.4.2 创建同步链路

1. 登录MAS控制台，进入“数据管理>数据同步”页面，选择前提条件中已创建好的命名空间，单击右上角的“创建同步链路”。
2. 在“创建同步链路”页面，填写配置信息，然后单击“提交”。

表 2-3 同步链路配置参数

参数	配置说明
命名空间	选择已创建好的命名空间。
任务异常通知设置	本实践配置为“关”。
任务异常自动结束时间	配置任务异常自动结束时间，任务处于异常状态一段时间后，将会自动结束。默认值14，单位为天。
添加链路	新增一条同步链路。
名称	同步链路名称，默认自动生成，或者根据规划自定义填写。
源数据源	选择源数据源。源数据源的数据库实例将配置为创建DB监控数据中心1的数据库。
目标数据源	选择目标数据源。目标数据源中RDS实例中不能有新创建的数据库，否则会导致链路创建异常。
同步方案	选择数据同步方案。
同步方向	选择同步方向，本实践配置“单向”。
限速配置	选择限速配置，本实践配置“限速”。
限速时段	选择限速时段，本实践配置“全天限流”。
流速大小	配置流速大小值，单位为MB/S，最大值为9999。

版权所有 © 华为云计算技术有限公司

2.4.3 创建应用

1. 登录MAS控制台，在“多活管理”页面单击前提条件中已创建的多活实例，进入实例控制台。
2. 在页面顶端导航栏选择“应用列表”，单击页面左上角的“新增应用”。
3. 在新增应用弹窗填写应用信息，完成后单击“确认”。

表 2-4 应用配置信息

参数	配置说明
应用名称	自定义应用名称。本实践配置为“litemall-mas-demo”。
描述	填写对该应用的描述内容，选填项。

2.4.4 创建 MySQL 监控

1. 登录MAS控制台，在“多活管理”页面单击前提条件中已创建的多活实例，进入实例控制台。
2. 在页面顶端导航栏选择“监控列表”，单击页面左上角的“新增监控”。
3. 在新增监控页面配置MySQL监控基础配置，完成后单击“下一步：数据中心配置”。

表 2-5 基础配置信息

参数	配置说明
监控	此处配置为“MySQL监控”。
应用名称	选择创建应用中已创建的应用。
监控器名称	自定义监控器名称。本实践配置为“litemall-db”。
异常通知设置	默认为“关”。若设置为“开”，当监控器或被监控的数据库异常时会及时给用户发送异常通知或告警通知。支持华为云消息通知服务，请先配置密钥。
通知主题	若异常通知设置选择“开”，则需要在通知主题列表选择主题，如无主题列表，单击新增，新增通知主题之后进行后续操作。
是否监控	默认“是”，选“否”则不会探测数据库异常情况。
是否自动切换	默认“是”，选“否”则不会自动切换数据库。
监控用户名	被监控数据库的用户名。
监控用户密码	被监控数据库的用户密码。
确认密码	再次填写监控用户密码。
DRS联动	本实践配置为“开”，关联DRS。请先配置密钥，再开启DRS联动。

版权所有 © 华为云计算技术有限公司

参数	配置说明
多活分区	选择多活分区。多活分区为多活实例所归属的命名空间所创建的分区名称。
DRS任务	选择创建同步链路中已创建的DRS任务。

4. 填写数据中心配置，分别配置数据中心1和数据中心2的信息，完成后单击“下一步：数据库配置”。

表 2-6 数据中心配置信息

参数	配置说明
云选择	被监控的数据库部署的环境，本实践配置为“华为云”。
局点	被监控数据库所在区域，选择本实践命名空间所选择的区域。
IPv4地址	配置为前提条件中的两个RDS的访问地址和端口，并将创建同步链路中选择的源数据源地址配置为数据中心1的地址。

5. 填写数据库配置，完成后单击“下一步：高级配置”。

表 2-7 数据库配置信息

参数	配置说明
监控数据库	填写监控数据库名称。
连接数据库	填写连接数据库名称。 **说明** “连接数据库”需要和“监控数据库”的值保持一致，且是被监控数据源中存在的数据库名。

6. 填写高级配置，本实践使用默认参数即可，完成后单击“下一步：确认配置”。
7. 确认配置无误后单击“立即创建”，完成创建MySQL监控。
8. 单击刚创建好的MySQL监控器所在行的“连接池配置”，“路由算法”配置为“单边读写”。
9. 单击“确定”，完成MySQL监控器连接池配置。

2.4.5 应用接入 SDK

1. 下载SDK至本地。
   a. 登录MAS控制台，进入“帮助中心”页面。
   b. 在“SDK下载>MySQL/Oracle/PostgreSQL”页签，选择Java语言最新版本SDK，单击“下载”。
2. 引用依赖。

   商城项目配置POM.xml文件（最外层POM文件），需要添加以下依赖：

```
<dependency>
        <groupId>com.huaweicloud.devspore</groupId>
```


版权所有 © 华为云计算技术有限公司

```
        <artifactId>spring-cloud-starter-huawei-devspore-datasource</artifactId>
        <version>1.2.9-RELEASE</version>
</dependency>
```

3. 获取SDK接入配置。
   a. 登录MAS控制台，在“多活管理”页面单击多活实例，进入多活实例控制台。
   b. 在“监控列表”页面，单击MySQL监控器左侧操作栏“更多”>“SDK接入配置”，复制配置参数。

图 2-2 复制配置参数

说明

此处获取的yaml配置不会携带数据库用户名密码信息，在使用时，需要补充完全（与创建MySQL监控时数据中心1和数据中心2数据库用户名和密码对应），同时还需要保证ETCD地址的可用性，再添加至项目的yaml文件。

4. 创建多活数据源配置。

获取SDK接入配置后，将相关信息添加完毕，找到项目对应的yaml文件，将相关信息添到商城项目的yaml文件，并且进行修改。由于商城后台项目自带springboot-dataSource，需要将其删除替换。修改后yaml文件如下所示：

```
devspore:
  datasource:
    # 如果配置了etcd，props则必须配置
    props:
      version:    v1
      appId:    670ddad4-f831-46a6-96de-8f1af0a347ed
      monitorId:    d715fcc1-894e-4f3e-aaa9-bbe0835efdfe
      databaseName:    litemall
      decipherClassName:    com.huawei.devspore.mas.password.DefaultDecipher  # 加解密类，需要实现基类 com.huawei.devspore.mas.password.Decipher，默认值为com.huawei.devspore.mas.password.DefaultDecipher【如果MAS服务上配置了用户名和密码，且密码是加密的，需要自己实现该类解密密文】
      region:    az0
    # etcd配置，对接MAS服务关键配置，本地模式则无需配置
    etcd:
      address:    192.168.1.174:2379,192.168.1.60:2379,192.168.1.16:2379
      apiVersion:    v3
      username:    etcduser
      password:    **********    # 请去基本信息页的ETCD链接地址自行查询，并务必进行加密处理
      httpsEnable:    true
      certificatePath: D:/2.cer/mas_cer/    # etcd证书路径，私钥请务必加密保存
    # 数据源配置 - 必选
    sources:
      ds1:
        driverClassName:    org.mariadb.jdbc.Driver  # 驱动名称，自定义
        jdbcUrl:    jdbc:mariadb://localhost:3306/litemall
        username:    litemall
        password:    **********    # 请务必进行加密处理
        type:    com.zaxxer.hikari.HikariDataSource  # 数据源类型，自定义，目前只支持，com.zaxxer.hikari.HikariDataSource 和 org.apache.commons.dbcp2.BasicDataSource
```


版权所有 © 华为云计算技术有限公司

```
    ds2:
      driverClassName:    org.mariadb.jdbc.Driver
      jdbcUrl:    jdbc:mariadb://localhost:3306/litemall
      username:    litemall
      password:    **********    # 请务必进行加密处理
      type:    com.zaxxer.hikari.HikariDataSource
  # 路由配置 - 必选
  router:
    active:    dc1
    routeAlgorithm:    single-read-write
    nodes:
      dc1:
        master:    ds1
        azs:
        loadBalance: round_robin
        slaves :
      dc2:
        master:    ds2
        azs:
        loadBalance: round_robin
        slaves :
```

5. 重新部署商城应用，验证商城应用能否读取数据源数据。
   a. 使用管理员账号登录电商应用。
   b. 测试电商应用的相关数据是否能够正常创建和查询。

2.4.6 故障切换演练

1. 查看监控器活跃的数据中心。
   a. 登录MAS控制台，在“多活管理”页面单击多活实例，进入实例控制台。
   b. 在页面顶端导航栏选择“监控列表”，查看创建的“litemall-db”MySQL监控器。

      在未出现故障前，监控器的活跃数据中心为数据中心1，即主数据中心，此时商城应用是连接数据中心1。
2. 构造数据中心1故障。

   通过**修改数据库端口**的方式，使“litemall-db”监控器的数据中心1故障。
3. 查看监控器的活跃的数据中心是否已自动切换。

   在多活实例控制台“监控列表”页面，可以发现“litemall-db”监控器数据中心1状态异常，“litemall-db”监控器活跃的数据中心已切换至数据中心2，此时商城应用是连接数据中心2。
4. 验证电商应用是否正常运行。
   a. 使用管理员账号登录电商应用。
   b. 测试电商应用的相关数据是否能够正常创建和查询。
5. 恢复数据中心1的故障。

   恢复数据中心1之前的端口，当数据中心1故障恢复时，此时监控器不会自动回切。
6. 手动回切数据中心。

   在多活实例控制台“监控列表”页面，单击“litemall-db”监控器左侧操作栏的“回切”，并在弹窗中单击“确认”，活跃的数据中心切换至数据中心1。
7. 重复步骤**4**，再次验证电商应用是否正常运行。

版权所有 © 华为云计算技术有限公司

3 事件流编排

3.1 方案概述
3.2 操作流程
3.3 实施步骤

3.1 方案概述

应用场景

MAS通过工作流编排和执行，提供业务进行跨区域容灾双活切换的完整流程。MAS工作流提供如下能力：

- 提供切换流程编排功能，可以一键式流程切换。
- 提供丰富插件，可实现各层级的切换/操作能力。
- 支持工作流模板，通过模板可快速完成复杂的容灾切换工作流的创建。

3.2 操作流程

前提条件

进行应用改造需要提前完成的事项：

- 已开通MAS“同城多活”版本功能模块，功能点包括“MySQL”。具体详情请参见**开通功能模块**。
- 创建好“同城多活”类型的命名空间，分区类型选择“华为云”。具体详情请参见**创建命名空间**。
- 在已创建的命名空间下购买MAS多活管理中心，即创建好多活实例。具体详情请参见**购买多活管理中心**。
- 已在华为云**购买RDS for MySQL**两个数据库实例，且与MAS多活实例配置为相同VPC、子网、安全组，与多活实例网络互通；数据库实例需配置为相同的用户名和密码。

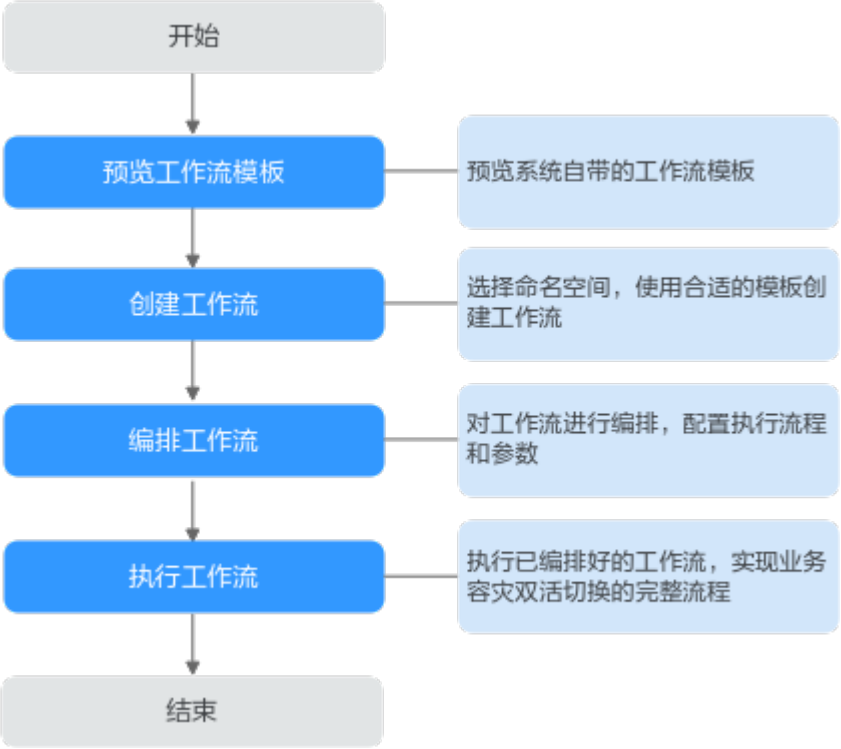
版权所有 © 华为云计算技术有限公司

- 商城应用部署在已创建的命名空间下的可用区，且业务部署架构需要和MAS多活分区保持一致；与多活实例配置为相同VPC、子网、安全组，与多活实例网络互通。

使用流程

工作流的使用流程如下图所示。

图 3-1 工作流使用流程



3.3 实施步骤

3.3.1 创建工作流

前提条件

已创建“异地容灾”类型的命名空间，如何创建命名空间请参考创建命名空间。

操作步骤

步骤1 登录MAS控制台，进入“工作流管理>工作流列表”页面。

步骤2 在左上角命名空间下拉框选择所需命名空间。

步骤3 单击“创建”，或者单击工作流模板“使用”。

版权所有 © 华为云计算技术有限公司

图 3-2 创建工作流

步骤4 在弹窗中填写工作流信息，完成后单击“确定”。

表 3-1 工作流配置信息

参数	配置说明
名称	自定义工作流名称。
描述	填写对应工作流的描述信息。

----结束

3.3.2 编排工作流

前提条件

已创建“异地容灾”类型的命名空间，如何创建命名空间请参考创建命名空间。

已通过模板创建工作流。

操作步骤

步骤1 登录MAS控制台，进入“工作流管理>工作流列表”页面。

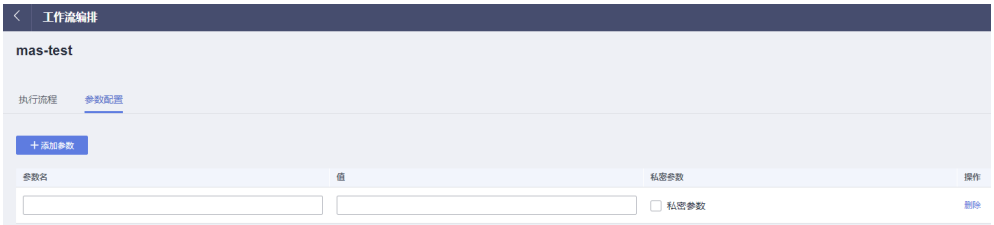
步骤2 在左上角命名空间下拉框选择所需命名空间。

步骤3 单击待编辑工作流所在行的“编辑”，进入“工作流编排”页面。

步骤4 在“工作流编排>参数配置”页面，配置参数信息。

1. 添加参数。单击“添加参数”，补全参数名和参数值。

图 3-3 参数配置



版权所有 © 华为云计算技术有限公司

表 3-2 参数配置

参数	配置说明
参数名	输入参数名称。
值	输入参数值。
私密参数	是否配置为私密参数。 **说明** 当配置为私密参数时，参数值不可见。在配置插件参数时，私密参数在插件不需要加密的参数下拉框选项中是不可见的，只有当插件中的某个参数需要加密时，私密参数才可见且可选。

2. 编辑参数。单击已有参数，对参数进行编辑。
3. 删除参数。单击待删除参数所在行的“删除”，在弹窗中单击“确认”。

步骤5 在“工作流编排>执行流程”页面，对工作流的阶段进行编排。

图 3-4 编排阶段

1. 新增阶段。单击新增阶段按键，在新增阶段弹窗中输入阶段信息，完成后单击“确定”。

版权所有 © 华为云计算技术有限公司

图 3-5 新增阶段

表 3-3 阶段配置信息

参数	配置说明
名称	自定义阶段名称。
执行方式	选择阶段串行/并行执行方式。 – 串行：表示当前阶段下的作业都将串行执行，阶段前图标表示串行。 – 并行：表示当前阶段下的作业都将并行执行，阶段前图标表示并行。

2. 编辑阶段。单击已有阶段后的“⋮”>“编辑”，在编辑阶段弹窗中编辑阶段信息，完成后单击“确定”。
3. 删除阶段。单击已有阶段后的“⋮”>“删除”，在弹窗中单击“确认”。

步骤6 在“工作流编排>执行流程”页面，对工作流的作业进行编排。

1. 添加作业。单击阶段下的“添加作业”，在添加作业弹窗中输入作业信息，完成后单击“确定”

版权所有 © 华为云计算技术有限公司

图 3-6 添加作业

表 3-4 作业配置信息

参数	配置说明
名称	自定义作业名称。
执行方式	选择作业串行/并行执行方式。 – 串行：表示当前作业下的任务都将串行执行，作业前图标表示串行。 – 并行：表示当前作业下的任务都将并行执行，作业前图标表示并行。

2. 编辑作业。单击已有作业后的“⋮”>“编辑”，在编辑作业信息弹窗中编辑作业信息，完成后单击“确定”。
3. 删除作业。单击已有作业后的“⋮”>“删除”，在弹窗中单击“确认”。

步骤7 在“工作流编排>执行流程”页面，对工作流的任务进行编排。

1. 添加任务。
   a. 单击作业中的“添加任务”。
   b. 单击所需插件。
   c. 完成参数配置。

表 3-5 任务配置信息

参数	配置说明
任务名称	自定义任务名称。

版权所有 © 华为云计算技术有限公司

参数	配置说明
插件配置	插件功能与参数配置说明，请参考对应的插件说明。 – DRS灾备切换 – DRS检查RTO&RPO – RDS实例读写设置 – RDS实例重启 – RDS实例创建备份 – DWS检查容灾任务 – DWS容灾操作 – DCS开启/关闭白名单 – OBS桶存量比较 – OBS跨区域复制任务设置 – OBS桶策略操作授权用户 – DNS添加记录集 – DNS删除记录集 – DNS公网域名记录集权重设置 – 人工卡点 – 等待 – AOM作业执行 – 混沌实验执行 – PerfTest测试任务启动/停止 – HTTP请求 – MAS多活管理切换/回切

2. 编辑任务。单击已有任务名称，在编辑任务信息弹框中编辑任务信息，完成后单击“确定”。
3. 复制任务。单击已有任务后的，完成任务复制。
4. 删除任务。单击已有任务后的，在弹窗中单击“确认”。

步骤8 如果在编排过程中需要对已有的阶段、作业、任务进行顺序调整，单击不放，拖拽到所需位置即可。

步骤9 单击页面右上角“保存草稿”，可以将工作流保存草稿，保存草稿时不需要补全任务信息。

步骤10 单击页面右上角“保存”，完成工作流的编排。

----结束

3.3.3 导入工作流

前提条件

已创建“异地容灾”类型的命名空间，如何创建命名空间请参考**创建命名空间**。

版权所有 © 华为云计算技术有限公司

操作步骤

步骤1 登录MAS控制台，进入“工作流管理>工作流列表”页面。

步骤2 在左上角命名空间下拉框选择所需命名空间。

步骤3 单击“导入工作流”，在弹窗中下载“工作流导入模板”。

图 3-7 导入工作流

步骤4 上传文件后，单击确定。

图 3-8 上传文件

步骤5 上传成功后，导入工作流可在列表中显示。

----结束

3.3.4 执行工作流

前提条件

1. 已创建“异地容灾”类型的命名空间，如何创建命名空间请参考创建命名空间。
2. 在命名空间下已购买执行工作流的多活管理中心，创建多活实例请参考购买多活管理中心。
3. 在命名空间下已创建并编排好工作流。

版权所有 © 华为云计算技术有限公司

操作步骤

步骤1 登录MAS控制台，进入“工作流管理>工作流列表”页面。

步骤2 在左上角命名空间下拉框选择所需命名空间。

步骤3 单击待执行工作流名称。

步骤4 在工作流详情页面，单击右上角“执行”。

步骤5 在“任务执行”弹窗，编辑参数配置。

说明

在“任务执行”弹窗修改参数值，只针对本次执行任务生效，下次执行此任务时，参数值依然为修改前的参数值。如需长期修改参数值，需要在“工作流编排>配置详情”页面编辑参数。

步骤6 在“任务执行”弹窗，单击“阶段选择”，勾选需要执行的阶段，单击“确定”，开始执行任务。

图 3-9 任务执行

步骤7 如果工作流中配置了“人工卡点”任务，在执行任务中会弹出“人工卡点”弹窗，在弹窗中输入审核意见，单击“继续执行”。

版权所有 © 华为云计算技术有限公司

图 3-10 人工卡点

步骤8 重试单个失败任务。

如某个任务执行失败，单击任务卡片中的重试图标“ ”，弹窗中单击“确认”，可以重新执行此任务。

步骤9 跳过单个失败任务。

如某个任务执行失败，单击任务卡片中的跳过图标“ ”，弹窗中单击“确认”，可以跳过此任务继续执行工作流。

步骤10 重试所有失败任务。

如果工作流执行后执行状态为“失败”，单击右上角“重试”，可以将所有失败的任务重新执行。

----结束

3.3.5 查看工作流执行详情

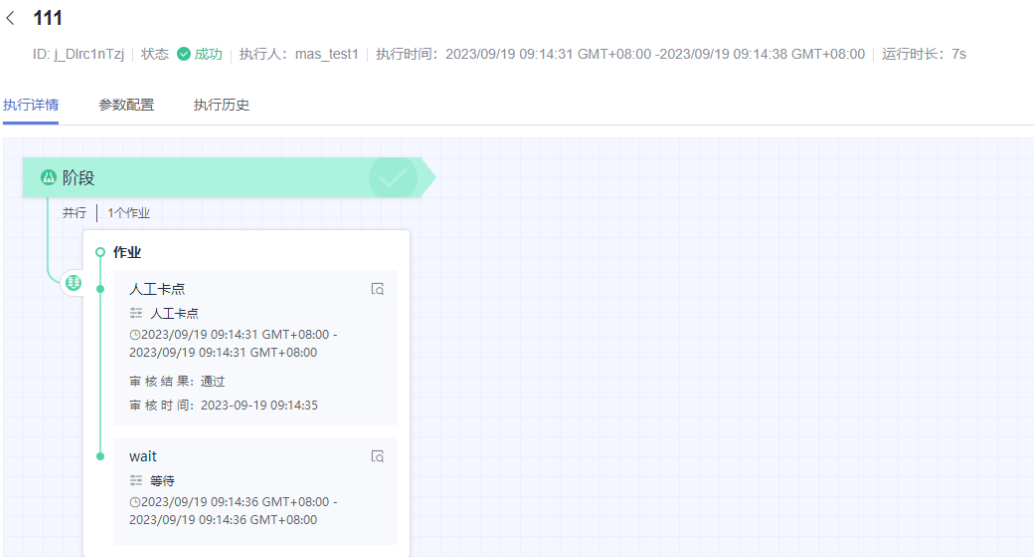
查看工作流详情

步骤1 登录MAS控制台，进入“工作流管理>工作流列表”页面。

步骤2 单击待查看的工作流名称，进入“执行详情”页面。

版权所有 © 华为云计算技术有限公司

图 3-11 执行详情



步骤3 单击任务的 ，查看任务执行的日志信息。

说明

查看日志只显示7天内的日志信息。

步骤4 单击任务的 ，查看任务更多执行信息。

步骤5 如果工作流执行失败，可以重试单个失败任务，跳过单个失败任务，重试所有失败任务。

步骤6 单击“执行”，可以重新执行工作流。

步骤7 单击“编辑”，可以重新编排工作流。

步骤8 单击“参数配置”，可以查看参数配置。

步骤9 单击“执行历史”，单击工作流的ID，可以查看该工作流的历史执行详情。

----结束

查看工作流执行历史

步骤1 登录MAS控制台，进入“工作流管理>执行历史”页面。

步骤2 在“执行历史”页面查看工作流的执行历史记录，支持输入工作流名称进行查询。

版权所有 © 华为云计算技术有限公司

图 3-12 执行历史

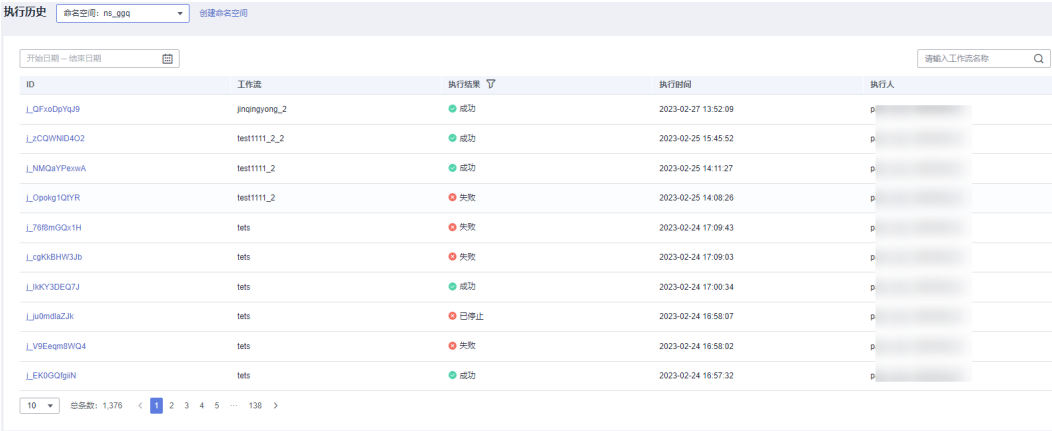


表 3-6 历史记录说明

内容	内容说明
ID	工作流ID。
工作流	工作流名称。
执行结果	工作流执行结果。 ● 成功 ● 失败 ● 已停止 ● 执行中 ● 停止中
执行时间	工作流的执行时间。
执行人	执行工作流的账号名称。

步骤3 单击某个工作流ID，查看该工作流ID的执行详情。

----结束

3.3.6 复制工作流

通过复制已有工作流，您可以快速创建一个新的工作流。通过选择命名空间，可以将工作流复制到另一个命名空间。

说明

复制后的新工作流无法直接执行，需要编辑保存后才可以执行工作流。

操作步骤

步骤1 登录MAS控制台，进入“工作流管理>工作流列表”页面。

步骤2 单击待复制工作流所在行的“复制”。

版权所有 © 华为云计算技术有限公司

步骤3 在弹窗中填写工作流名称和描述信息。

步骤4 选择所需复制到的命名空间，默认为当前命名空间。

步骤5 单击“确认”，完成复制工作流。

----结束

3.3.7 删除工作流

删除单个工作流

步骤1 登录MAS控制台，进入“工作流管理>工作流列表”页面。

步骤2 单击待删除工作流所在行的“删除”。

步骤3 在弹窗中单击“确定”，完成删除工作流。

----结束

批量删除工作流

步骤1 登录MAS控制台，进入“工作流管理>工作流列表”页面。

步骤2 勾选待删除的工作流。

步骤3 单击列表上方的“删除”。

步骤4 在弹窗中单击“确定”，完成批量删除工作流。

----结束

3.3.8 关注工作流

操作步骤

步骤1 登录MAS控制台，进入“工作流管理>工作流列表”页面。

步骤2 单击待关注工作流所在行的“关注”，完成关注工作流。

步骤3 单击待关注工作流所在行的“取消关注”，完成取消关注工作流。

步骤4 在工作流列表右侧下拉框选中“我的关注”，可以筛选出列表中所有已关注的工作流。

步骤5 在“工作流管理>我关注的”页面查看关注中最新执行的工作流的执行状态。

----结束

版权所有 © 华为云计算技术有限公司