应用平台

开发指南

文档版本 03

发布日期 2024-04-02

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。

目 录

1 使用 NUWA 框架实现应用开发

1.1 开发概述

本文将介绍如何开发一个NUWA项目，带您体验从工程创建、代码编写、调试运行到部署上线的全过程。

本项目的目标是开发一个本地运行的HTTP服务，你将了解：

- 如何引入NUWA框架创建NUWA项目
- 如何在IDEA中启动NUWA进程
- 使用NUWA功能模块开发本地运行的HTTP服务
- NUWA项目的打包规范
- VM上运行NUWA项目

NUWA框架通过插件封装了SDK的能力，如果要使用某一种功能，需要引用对应的插件。业务基于NUWA开发必须遵循NUWA的依赖管理方式和版本， 并使用provided的模式，依赖NUWA模块。

说明

开发过程请参考**快速开始**，本章节主要是快速开始的补充增强。

NUWA 简介

NUWA提供一个完整的微服务开发框架，是一个开箱即用的应用级容器，以插件的形式汇聚云服务平台能力，让开发把时间更多的花在业务代码逻辑上。

基本概念/工作原理

NUWA是平台能力的统一入口，封装gpaas、apaas、安全、DFX等能力， 结合devops流程，让业务开箱即用，简化项目开发，同时标准化运行环境，提升运维监控的可靠性，从开发到运维提供一条龙服务。

图 1-1 NUWA 框架定位

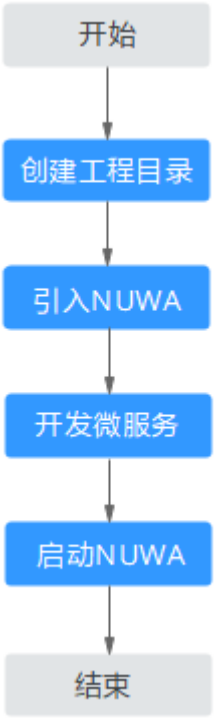
基本功能

- 基础工程能力
  - 支持自动化部署
  - 模块选择加载
  - 统一配置文件加载
  - 统一日志配置
  - 统一HCW监控&告警
  - 统一秘钥管理
  - 单元测试能力
- 服务接入能力
  - 微服务开发框架
  - 网关服务开发框架
- 平台公共能力
  - 唯一ID服务
  - 微服务流控能力
  - 灰度平台服务
  - 地理位置服务
- 基础运维能力
  - Watchdog业务监控
  - 集成调用链
  - 自动化部署

1.2 总体开发思路

使用NUWA框架开发微服务的工作流程如下：

图 1-2 NUWA 框架开发流程图



1. 创建工程目录
   创建一个工程目录，补全NUWA框架需要的目录。
2. 引入NUWA
   引入NUWA依赖，创建NUWA工程，配置NUWA，补全NUWA需要的文件。
3. 开发微服务
   基于NUWA框架开发业务代码。
4. 启动NUWA
   通过创建Application运行工程，启动NUWA。

1.3 准备工作

需要准备JDK 8、IntelliJ IDEA、Maven。

开发技能要求

- 熟悉Java语言，能够编写Java语言代码。
- 掌握IaC开发技术，熟悉YAML语言。

下载 SDK

获取SDK并进行完整性校验。

- SDK：**nuwa-open-sdk-1.1.0-20240204093135.zip**
- 完整性校验：**nuwa-open-sdk-1.1.0-20240204093135.zip.sha256**

1.4 快速开始

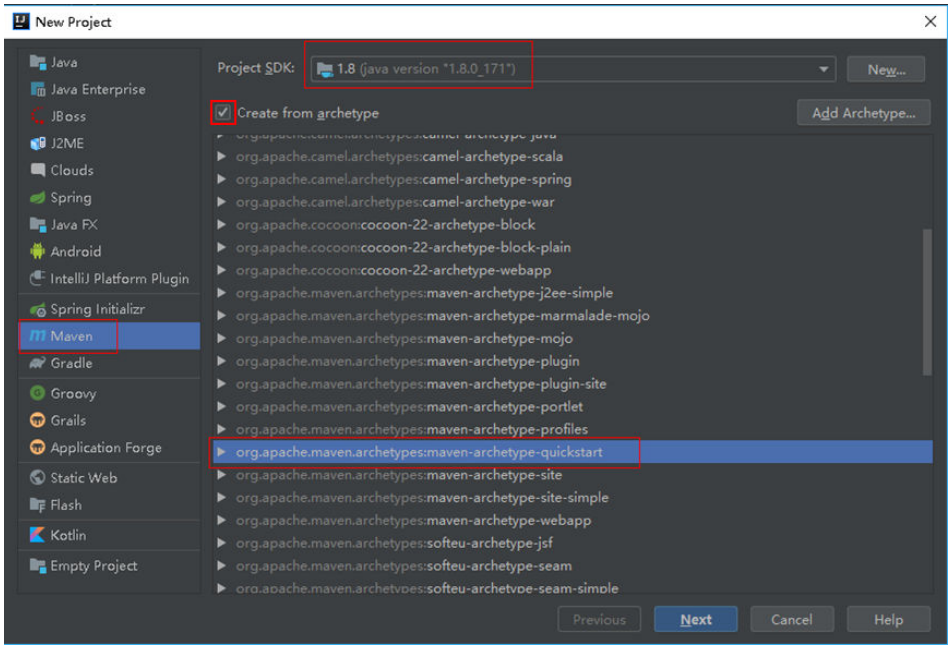
本章节以IDEA引入NUWA依赖方式为例，介绍如何开发一个NUWA项目，带您体验工程创建、代码编写、调试运行的全过程。

操作步骤

步骤1 创建工程目录

1. 创建一个Maven空工程。
   a. 打开IntelliJ IDEA，选择“File > New > Project”。

   图 1-3 创建 Maven 工程

   b. 根据界面提示，一步步单击“Next”，即可成功创建一个基础的Maven工程。

2. 改变Maven的JDK版本。
   修改pom.xml文件中的配置。

```
<properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```

3. 补全NUWA目录。
   Maven默认的资源文件目录如下：

- 业务资源文件目录：src/main/resources。
- 测试资源文件目录：src/test/resources。

默认资源文件目录中的文件也会被打包到jar包中，建议再增加一个src/main/config目录用于存放不打包为jar包的配置文件。

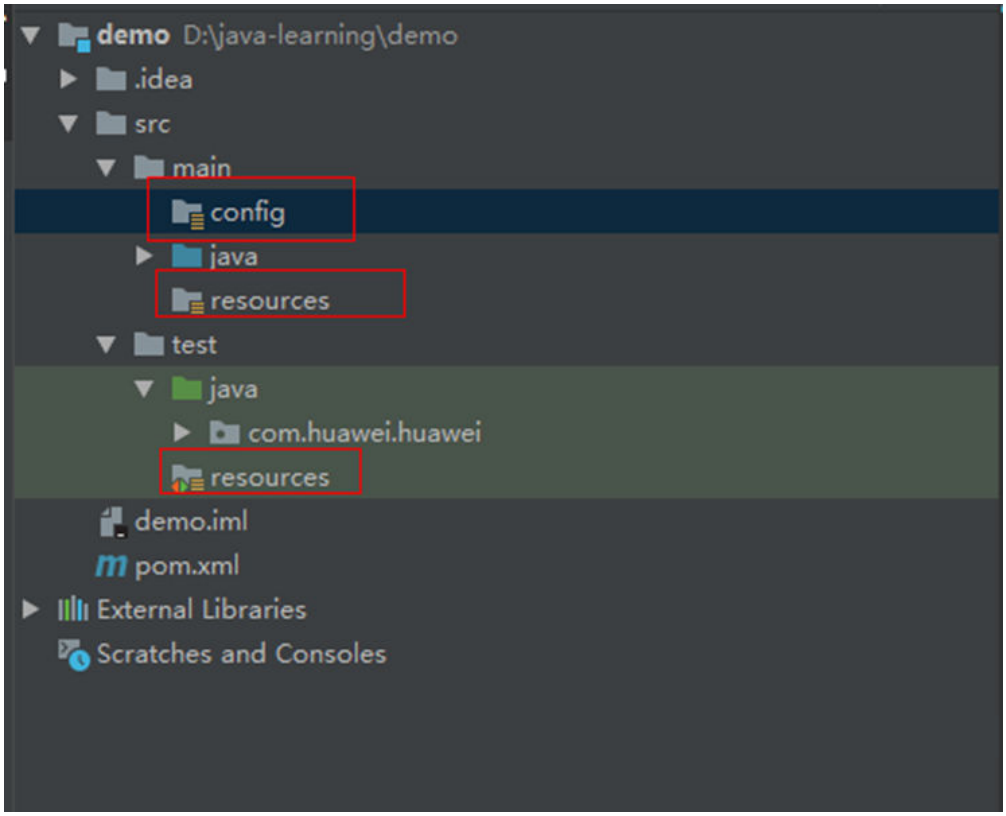
一般会将Spring的配置文件、SPI扩展的配置文件等打包到jar包中，其它的日志配置文件、业务properties、yaml配置文件等专门放在一个配置目录里。

在pom.xml里将src/main/config目录设置为资源文件目录（相当于classpath路径）。

```
<build>
  <resources>
    <resource>
      <directory>src/main/config</directory>
    </resource>
    <resource>
      <directory>src/main/resources</directory>
    </resource>
  </resources>
</build>
```

最后，工程结构如图1-4所示。

图 1-4 补全目录之后的工程目录结构



步骤2 引入NUWA

1. 手动导入SDK jar包。
   a. 在项目目录下新建一个lib目录，存放jar包。
   b. 将本地的jar包复制粘贴至lib目录下。
   c. 将jar包导入到项目中。

i. 选择“File > Project Structure > Project Settings > Module”。
ii. 单击“+”，选择“JARs or Directories...”。
iii. 选中jar包，单击“apply”。导包完成。

2. 依赖管理

在pom.xml增加dependencyManagement节点，管理依赖的软件版本。nuwa-dependencies这个pom可以管理所有依赖NUWA的版本。

```
<dependencyManagement>
   <dependencies>
      <dependency>
         <groupId>com.huawei.wisecloud.nuwa</groupId>
         <artifactId>nuwa-dependencies</artifactId>
         <version>3.0.19.101</version>
         <type>pom</type>
         <scope>import</scope>
      </dependency>
   </dependencies>
</dependencyManagement>
```

3. 依赖CSE模块

CSE是开发微服务的基础模块。编辑pom.xml，以provided的方式引入nuwa-cse-foundation模块，provided表示在打包的时候，这些jar包都不会打入到业务包，而是由NUWA安装包提供。此处无需配置版本号，上一步引入的NUWA依赖管理，已经对所有NUWA的包的版本进行了约束管理。

```
<dependency>
  <groupId>com.huawei.wisecloud.nuwa</groupId>
  <artifactId>nuwa-cse-foundation</artifactId>
  <scope>provided</scope>
</dependency>
```

另外，增加config/nuwa.boot.properties文件，并在该文件中，增加如下配置：

```
nuwa.system.module.loadingList=nuwa-cse-foundation
```

该配置有时候在IDEA启动时并不是必须的，即使删除掉，也能正常运行。但部署到服务器运行后，该配置则是必需的。

4. 选择一个日志模块

日志模块可以选择logback或log4j2，依然在pom里引入。此处以logback为例：

```
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>1.4.8</version>
  <exclusions>
    <exclusion>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-api</artifactId>
    </exclusion>
  </exclusions>
</dependency>
```

5. 配置日志文件

在src/main/config目录下创建一个logback.xml文件，示例如下：

```
<?xml version="1.0" encoding="UTF-8"?>

<configuration scan="true" scanPeriod="120 seconds" debug="false">
   <appender class="ch.qos.logback.core.rolling.RollingFileAppender" name="demo_appender">
      <encoder>
         <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level-%msg%n</pattern>
      </encoder>

      <file>${LOG_HOME}/demo.log</file>

      <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
```

```
        <FileNamePattern>${LOG_HOME}/demo-%i.log.gz</FileNamePattern>
        <minIndex>1</minIndex>
        <maxIndex>20</maxIndex>
      </rollingPolicy>

      <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
        <maxFileSize>50MB</maxFileSize>
      </triggeringPolicy>
    </appender>
    <logger name="com.huawei.demo" additivity="false" level="INFO">
      <appender-ref ref="demo_appender"/>
    </logger>

    <root level="INFO">
      <appender-ref ref="demo_appender"/>
    </root>
</configuration>
```

默认环境变量LOG_HOME=/opt/huawei/logs/${模块名称}。

业务可以修改该路径，业务自定义日志配置，可以引用该环节变量。

内置中间件绕接规则是根据日志大小50M，保留20个，建议业务至少预留日志磁盘大小20G。

步骤3 开发微服务

- 微服务原理

  图 1-5 注册中心使用流程

  Producer是服务提供方，Consumer是服务调用方。Consumer本身也可以是服务提供方，Producer也可以再去调用别的微服务。 从而形成一个微服务调用网。

  Producer会提供http接口，Producer会把自己的服务名注册到注册中心。Consumer从注册中心获取服务名对应的机器列表，然后再发http请求去调用。当然这个调用过程是CSE封装的。

  CSE微服务也可以不接入注册中心，作为一个纯粹的http服务对外提供。

- 增加微服务配置文件

  在src/main/config目录下创建一个microservice.yaml文件。

  ```
  APPLICATION_ID: demo      #'#服务名'
  service_description:
    name: demoService      # '#微服务名'
    version: 0.0.1
  cse:
    rest:
      address: 127.0.0.1:8081
  ```

  文件定义了服务名称，监听端口，以及接口版本号，没有配置“服务注册中心”，如果有多个微服务，需要相互调用，可能需要搭建一个服务注册中心，一般本地开发不需要注册中心。

- 定义一个服务类（业务自定义功能实现），示例代码如下：

```
package com.huawei.demo;

import org.apache.servicecomb.provider.rest.common.RestSchema;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import javax.ws.rs.QueryParam;

@RestSchema(schemaId = "test")
@RequestMapping(path = "/")
public class DemoService {
    @RequestMapping(path = "/hello",method = RequestMethod.GET)
    public String sayHello(@QueryParam(value = "name") String name) {
        return "hello"+ name;
    }
}
```

RestSchema是一个CSE接口的标识，CSE通过这个注解找到所有的服务类，每个服务类的schemaId不同。

说明

CSE是实现了对Spring MVC注解的支持，但是它本身并不是Spring MVC，很多Spring MVC的功能无法使用。

- 配置Spring文件

  CSE和NUWA都是基于Spring开发的，业务定义的服务类如果被CSE发现，则会交给Spring管理。

  NUWA启动原理是通过加载classpath*:META-INF/spring/*.xml以及classpath*:spring/*.xml下面的Spring文件把Spring容器拉起来。

  可以增加一个src/main/resources/META-INF/spring/demo-spring.xml文件，将服务类加进去。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context" xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd">
    <context:component-scan base-package="com.huawei.nuwa.demo" />
</beans>
```

步骤4 启动NUWA

1. 设置启动参数

   设置启动类：com.huawei.nuwa.boot.loader.NuwaClassPathLauncher

   VM启动参数：-Dlocal.registry.file=notExistJustForceLocal

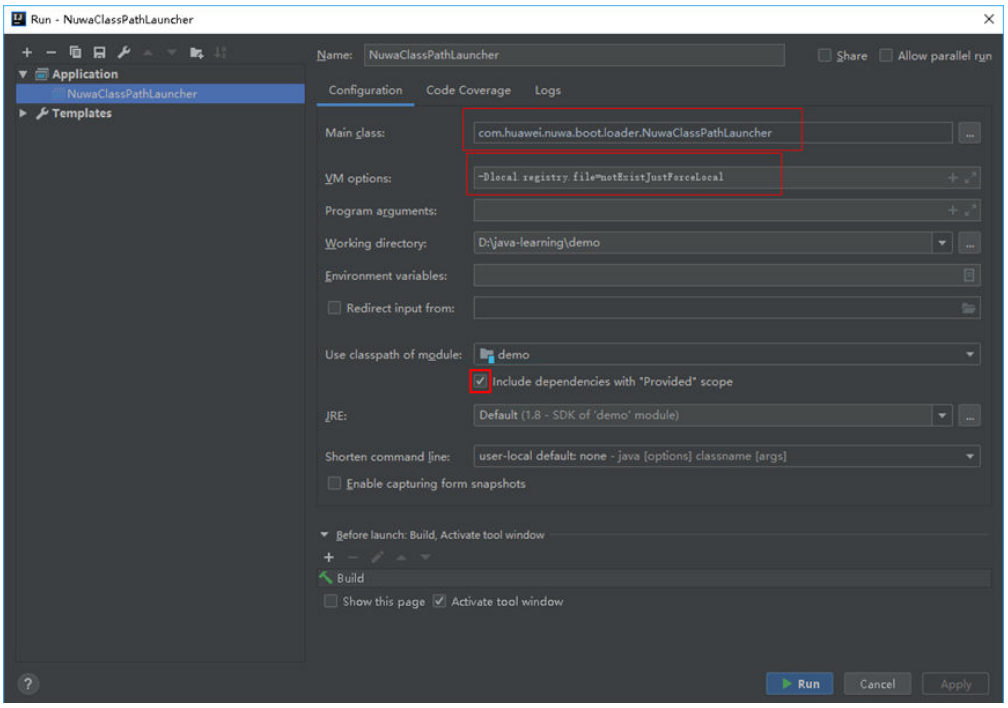
注意

本地调试或者不需要调用其它微服务，可以指定不使用注册中心。

该参数表示没有注册中心，如果不配置该参数，启动以后默认会连127.0.0.1:30100作为注册中心地址。连不上注册中心，就不会对外提供服务。

请勾选“Include dependencies with Provided scope”。默认IDEA是不加载provided领域下的jar包的。

图 1-6 本地调试启动参数设置



2. 测试服务

   执行Run，启动成功后，会输出如图1-7所示的一段内容。

图 1-7 启动成功标志

通过浏览器访问服务：

图 1-8 接口访问成功

----结束

1.5 开发指导

1.5.1 基于 CSE 开发接口

概述

CSE是ServiceComb的企业版，主要适用于开发Rest风格的接口的（微）服务。

CSE提供注册中心的功能，如果业务根据功能拆分成多个微服务，微服务之间需要相互调用，则可以部署一个CSE提供的注册中心，通过注册中心实现服务发现功能。如果只要一到两个服务，可以选择不使用注册中心，微服务对外接口使用SLB进行负载均衡：调用方 > SLB > 服务。

CSE基于Spring开发，服务端直接实现了SpringMVC的注解，以方便通过SpringMVC注解开发，但本质上并不是SpringMVC。

```
@RestSchema(schemaId = "helloService")
@RequestMapping(path = "/demo")
public class CSEDemo {

    @RequestMapping(path = "/hello", method = RequestMethod.GET)
    public String provider(@RequestParam(value = "name", required = false) String name) {
        return "hello " + name;
    }
}
```

客户端是扩展了Spring的RestTemplate：

```
RestTemplate restTemplate = RestTemplateBuilder.create();
String response = restTemplate.getForObject("cse://HelloService/demo/hello?name={name}", String.class,
    "zhangsan");
```

提供了cse://{服务名}//的协议的扩展，内部原理就是通过服务名到注册中心查询服务对应的http地址，再通过http请求去调用服务。

引入依赖

需要以provided的方式依赖NUWA。

```
<dependency>
      <groupId>com.huawei.wisecloud.nuwa</groupId>
      <artifactId>nuwa-cse-foundation</artifactId>
      <scope>provided</scope>
   </dependency>
```

另外，在config/nuwa.boot.properties中，需要增加如下配置：

```
nuwa.system.module.loadingList=nuwa-cse-foundation
```

CSE 使用配置

请参见**开发微服务**和**设置启动参数**。

CSE推荐配置参数，是基于一般场景给出的参考值，并非万能方案。

```
APPLICATION_ID: ${applicationID} #CSE体系中的应用ID，建议配置PBI中的服务名，方便与STS集成
service_description:
  name: ${name} #微服务名称，建议配置成PBI中的微服务名，方便与STS集成
  version: ${version} #微服务版本号，与代码工程版本无关
  properties:
    allowCrossApp: true #跨应用调用开关，没有跨应用调用需要的业务可以关闭
servicecomb:
  service:
    registry:
      address: http://127.0.0.1:30100 #服务中心地址，业务根据实际需要修改
```

```
    instance:
      healthCheck:
        interval: 30 #心跳间隔时间，默认30秒
        times: 3 #健康检查检测次数，默认3次
      watch: true #watch机制可以快速感知实例变化，提升路由准确性，建议开启
      diagnose:
        interval: 10 #服务诊断的间隔，时间单位为小时
  monitor:
    client:
      enable: false #不对接CSE提供的dashboard监控服务
  rest:
    #有需要的业务可以开启HTTPS，如果开启则还需要配置证书信息，见模板末尾的ssl.*配置项
    #默认协议为HTTP/1.1，如果要开启HTTP/2则需要配置如下示例的protocol=http2，关于HTTP/2的说明请参考README
    address: 127.0.0.1:8080?sslEnabled=false&protocol=http2
    server: #作为服务端的配置
      connection-limit: 100000 #允许服务端最大连接数，单个服务端不允许超过10W个连接，如果超过需要评审
      compression: true #支持服务端压缩
      maxHeaderSize: 8192 #HTTP消息头的最大长度推荐设置为8K，最大不要超过32K
      connection:
        idleTimeoutInSeconds: 60 # 服务端连接超时时间，一个连接在指定时间内没有接收到请求,就主动关闭该连接
      http2: #服务端HTTP2相关的配置，如果没有开启HTTP/2则不用配置
        concurrentStreams: 200 # 一条连接中，同时支持的最大的stream并发量，默认100，可以适当调大该值，详见README说明
        useAlpnEnabled: true #是否启用ALPN，默认true
    client: ##作为消费端时的配置
      maxWaitQueueSize: #请求队列大小，请根据实际情况评估值
      connection:
        maxPoolSize: 50 #请求连接池不要设置太大，不能超过50，对于异步I/O系统，瓶颈通常在后端业务处理，链路过多有风险
        idleTimeoutInSeconds: 30 #链路空闲时间，达到该阈值后主动关闭HTTP链接
        keepAlive: true #使用HTTP长连接机制，对于内部RPC通信，不能使用短连接，否则有性能问题
      http2: #客户端HTTP2相关的配置，如果没有开启HTTP/2则不用配置
        maxPoolSize: 5 # 每个连接池中，对每一个IP：Port最多建立的连接数，默认值为1，推荐5以获得更好的性能，详情请参考README文档
        multiplexingLimit: 200 #一条连接中，同时支持的最大的stream并发量，-1表示不限制，即由服务端的并发量限制
        idleTimeoutInSeconds: 30 #HTTP2 连接闲置超时时间
        useAlpnEnabled: true #是否启用ALPN，默认true
  request: #作为消费端的超时配置，支持全局配置和microservice/schema/operation三个级别的配置
    timeout: 500 #微服务消费端超时时间,默认500毫秒，如果调大到秒级，必须经过评审和评估，否则风险很大
    # 服务级别的配置
    <service>: #对某个服务端生效，实际取值为服务名
      timeout: 500
      <schema>: #对该服务的schema生效，实际取值为schema名
        timeout: 500
        <operation>: #对该服务的schema的某个operation进行限流，实际取值为方法名
          timeout: 500
  executor:
    default:
      group: 2 #线程池组数
      coreThreads-per-group: 100 #每组线程池的最小线程数，推荐和maxThreads-per-group保持一致
      maxThreads-per-group: 100 #每组线程池的最大线程数，建议根据实际情况调整大小
      maxQueueSize-per-group: 100000 #每组线程池中任务队列的最大长度。通常业务时延越小、内存越大，队列可以调整得越大，估算公式：100000/平均时延(单位:毫秒)。建议做一下压测验证。
  executors:
    Provider: # 隔离仓配置，按照schemaId和方法名配置线程隔离仓，如果需要隔离，业务自行配置。注意：如果业务自定义线程池，需要自行管理上述线程池参数、以及性能监控数据，Java-Chassis无法纳管。
      <schema>: schema-threadPool #配置到[schema]级别，value为自定义pool对应的Spring beanId
      <schema>.<operation>: schema-operation-threadPool #配置到[schema].[operation]级别
  Provider.requestWaitInPoolTimeout: <请业务按实际情况取值> # Provider端业务线程池请求排队超时时间, 单位:毫秒(全局配置, 默认30000)
  Provider.requestWaitInPoolTimeout.<schema>: <请业务按实际情况取值> # Provider端业务线程池请求排队超时时间, 单位:毫秒(契约级配置, 默认30000)
  Provider.requestWaitInPoolTimeout.<schema>.<operation>: <请业务按实际情况取值> # Provider端业务线程池请求排队超时时间, 单位:毫秒(方法级配置, 默认30000)
  uploads: # 文件上传配置，业务如果涉及文件上传流程，自行配置
    directory: /home/upload # 文件上传目录，有上传下载功能的业务需要自己规划
```

```
    maxSize: 10485760 #上传文件body大小，应用市场建议设置为10M
  handler:
    chain:
      Provider: #服务端调handler配置
        default: qps-flowcontrol-provider #默认包含服务端流控
      Consumer: #消费端调handler配置
        default: qps-flowcontrol-consumer,loadbalance #默认包含消费端流控、负载均衡功能
  datacenter: #服务隔离配置，双云环境下是必须的，保证本机房的消费端优先路由到本机房的服务端，尽量避免跨机房路由
    name: AppGallery #数据中心名称, 默认设置为AppGallery
    region: North-China #区域，对应中国/新加坡/欧洲等
    availableZone: JiuXianQiao #AZ信息，例如酒仙桥/廊坊/AZ1等
  flowcontrol: #限流配置，限流的配置需要业务根据自身情况进行调整
    Provider: #服务端全局配置
      qps:
        enabled: true #是否开启，仅支持全局开关
        global:
          limit: 10000 #全局限流，包含所有的接口流量，默认设置为1W QPS，支持microservice/schema/operation三个级别的配置，后者的优先级高于前者
        limit:
          ANY: 100 # ANY选项表示任意来源，匹配所有的调用方（包括不是Java-Chassis开发的第三方），此特性从2.5.3.B002(1.3.1.B002)版本开始提供
          ANY.<schema>: 100 # 任意来源调用本服务<schema>契约的限流
          ANY.<schema>.<operation>: 100 # 任意来源调用本服务<schema>.<operation>方法的限流
          ANY.healthEndpoint: 10 # 给健康检查接口预留10QPS的流量，防止流量高峰期Watchdog健康检查误判
          <service>: 10000 #对某个消费端生效，实际取值为【消费端】的服务名，即限定某个consumer调用本provider
            <schema>: 10000 #对该服务的schema生效，实际取值为schema名
              <operation>: 10000 #对该服务的schema的某个operation进行限流，实际取值为方法名
    Consumer: #消费端全局配置
      qps:
        enabled: true #是否开启，仅支持全局开关
        global:
          limit: 10000 # 全局限流，此特性从2.5.3.B002(1.3.1.B002)版本开始提供
        limit: #调用其他服务的限流，没有全局配置，必须要配到microservice/schema/operation三个级别中的某一个，后者的优先级高于前者
          <service>: 1000 #对某个服务端生效，实际取值为服务名
            <schema>: 10000 #对该服务的schema生效，实际取值为schema名
              <operation>: 100 #对该服务的schema的某个operation进行限流，实际取值为方法名
  loadbalance: #负载均衡/隔离配置
    retryEnabled: true #建议考虑开启重试机制, 此配置项CSE的默认值为false, 需业务手动开启
    retryOnSame: 0 #不在同一台机器重试，大部分情况下在同一台机器重试会加重目标服务实例的负载，建议切换到其它服务实例重试
    retryOnNext: 1 #在另外实例上重试次数，建议修改为1
    isolation:
      enabled: true #开启实例级故障隔离
      enableRequestThreshold: 60 #隔离门槛，该条件是隔离的必要条件，在1分钟统计周期内调用量达到该门槛才会进入熔断判断逻辑，默认为60，即1TPS，业务可修改。注意！如果值太大达不到门槛将无法触发隔离！
      singleTestTime: 5000 # 故障实例单点测试时间，隔离之后，每隔该时间会测试下被隔离的服务实例是否恢复，业务可修改，用于快速恢复
      # NuwaSDK 3.0.1.202之前的版本使用continuousFailureThreshold
      # NuwaSDK 3.0.1.202版本开始，建议删除continuousFailureThreshold配置，使用errorThresholdPercentage+recoverImmediatelyWhenSuccess
      #continuousFailureThreshold: 100 #只应该在NuwaSDK 3.0.1.202之前的版本使用这个配置！！连续发送100条都失败，触发实例级隔离。业务可根据自己的可靠性指标进行修改
      errorThresholdPercentage: 20 #基于错误率的阈值配置，注意，该配置在某些版本中会导致隔离无法恢复的问题，如果需要配置，务必先确认对应SDK的版本是否已优化
      recoverImmediatelyWhenSuccess: true # 放通实例，如果调用成功，立即清除统计状态，保证后续请求能够使用该实例。 2.5.6(1.3.2)新增
    strategy: # 负载均衡策略
      name: RoundRobin # 默认推荐用轮询，策略最简单、行为可预期，支持的策略有[RoundRobin,Random,WeightedResponse,SessionStickiness]
  references:
    version-rule: 0.0.0+ #路由时过滤服务端的版本范围，默认配置为0+，即匹配所有版本
    <service>: #调用某个<service>服务的过滤版本范围
      version-rule: 0.0.0+
  # ================================日志配置==================================================
  accesslog: #开启CSE的接口日志
```

```
    enabled: true
    pattern: "%h - %t %r %s %B %D %SCB-traceId" #accesslog格式，支持自定义
  metrics: #开启CSE的性能统计接口日志
    publisher:
      defaultLog:
        enabled: true
# ====================================SSL配置
==============================================
ssl:
  protocols: TLSv1.2 #协议版本号
  authPeer: false #是否开启双向认证
  checkCN:
    host: false #不校验host是否匹配
  keyStore: server.p12 #证书的文件名
  keyStoreType: PKCS12 #证书格式类型
  keyStoreValue: test #证书密码，需要加密存储，根据业务自身的加密算法加密
  sslCustomClass: com.huawei.appgallery.SSLCustomImpl #自定义处理类的类名，用来获取证书完整路径并解
密keyStoreValue密码
```

支持 URL 匹配的 Filter

CSE本身支持服务端和客户端的Filter SPI扩展，但是针对所有的请求。

- 客户端org.apache.servicecomb.common.rest.filter.HttpClientFilter
- 服务端org.apache.servicecomb.common.rest.filter.HttpServerFilter

NUWA对其进行了扩展，支持URL+METHOD匹配过滤。

- 服务端继承com.huawei.nuwa.cse.web.filter.MatchedHttpServerFilter
- 客户端继承com.huawei.nuwa.cse.web.filter.MatchedHttpClientFilter

实现getBeanName()接口，返回filter的名字，用于在配置文件中读取对应的配置项。假设filter的名字是{filterName}，对应的配置项如**表1-1**所示。

表 1-1 配置说明

配置项	默认值	含义
servicecomb.filter.{filterName}.enabled	true	是否启用该filter，默认是true，自定义的filter也可以重载defaultEnable() 修改默认值。
servicecomb.filter.{filterName}.urlPattern	null	URL匹配类型，可以配置多个，使用逗号分隔。只支持如下匹配方式： 后缀类型 /cse/do* **须知** 匹配规则必须在末尾加上“*”，其他位置不能有“*”。
servicecomb.filter.{filterName}.httpMethod	ALL	和urlPattern配合使用，表示进一步匹配METHOD，使用逗号分隔。

配置项	默认值	含义
servicecomb.filter.{filterName}.allowPattern	null	复合匹配类型的配置形式，结构如下： urlPattern:Method;urlPattern:Method;urlPattern:Method 例如： /test*:GET;/test2/test3*:*;/test4/test5*,/test6*:PUT,POST **说明** 这里urlPattern的配置方式参考配置项：servicecomb.filter.{filterName}.urlPattern

⚠ 注意

HttpClientFilter/HttpServerFilter扩展采用SPI加载机制，要使扩展的filter能够正常加载，需要在代码工程的resources/META-INF/services/目录下放置对应的SPI配置文件。如果业务扩展的是HttpServerFilter，则配置文件的名称为org.apache.servicecomb.common.rest.filter.HttpServerFilter；如果扩展的是HttpClientFilter，则文件名为org.apache.servicecomb.common.rest.filter.HttpClientFilter。文件内容为业务扩展的filter的类名（带包名）。例如，用户自MatchedHttpClientFilter扩展一个com.huawei.xxx.SimpleMatchedHttpClientFilter，则需要提供一份名为org.apache.servicecomb.common.rest.filter.HttpClientFilter的配置文件，内容为com.huawei.xxx.SimpleMatchedHttpClientFilter 。

微服务间认证-共享密钥鉴权

基于URL匹配Filter，实现了内置的共享秘钥鉴权filter。

- 服务端：com.huawei.nuwa.cse.auth.filter.HmacAuthServerFilter
- 客户端：com.huawei.nuwa.cse.auth.filter.HmacAuthClientFilter

```
servicecomb:
  filter:
    hmacAuthClientFilter:
      urlPattern: /compute/*
    hmacAuthServerFilter:
      urlPattern: /compute/*
```

其他共享秘钥相关的配置如下：

表 1-2 共享密钥相关配置说明

配置项	默认值	含义
nuwa.cse.hmacAuth.isMutual	false	是否开启双向认证。
nuwa.cse.hmacAuth.expireTimeMinutes	30	签名时间戳有效时间。
nuwa.cse.hmacAuth.rootKeyPath	null	如果使用cloudsoa管理的本地秘钥文件，需要指定根秘钥路径。
nuwa.cse.hmacAuth.workKeyJsonFile	null	如果使用cloudsoa管理的本地秘钥文件，需要指定工作秘钥文件路径。
nuwa.cse.hmacAuth.cryptoAlg	AES_CBC	cloudsoa加密方式。

如果不使用本地文件秘钥，比如是从数据库存储的，可以自己实现秘钥工厂类com.huawei.basecloud.cloudsoa.security.multi.KeysFactory，并配置到Spring里即可。

微服务间认证-STS 鉴权

基于URL匹配Filter，实现了内置的STS鉴权filter。

- 服务端：com.huawei.nuwa.cse.auth.filter.StsAuthServerFilter
- 客户端：com.huawei.nuwa.cse.auth.filter.StsAuthClientFilter

```
servicecomb:
  filter:
     StsAuthServerFilter:
       urlPattern: /compute/*
     StsAuthClientFilter:
       urlPattern: /compute/*
```

要支持微服务之间的认证，首先需要在STS中注册微服务之间访问的权限。通过ACL和Accessibility来录入访问权限。如，注册访问控制表（ACL）、注册访问能力表（Accessibility）等。

一致性 HASH 路由策略

一致性Hash路由策略，目的是在微服务的调用侧（客户端侧、消费端侧）实现一致性hash路由测试。保证同一个属性的请求只发送到同一个服务器上，同时增加和减少一个服务器不会造成分发策略的大的变动（大部分请求路由不发生改变）。

表 1-3 HASH 路由策略配置项说明

配置项	默认值	含义
servicecomb.handler.chain.Consumer.default	/	必选，Consumer端handler链中必须添加使用loadbalance，否则无法启用一致性HASH路由策略。
servicecomb.loadbalance.strategy.name	默认为CSE给定的默认值，RoundRobin	全局默认负载均衡策略，修改为ConsistentHash则可以启用一致性HASH策略。
servicecomb.loadbalance.{servicename}.strategy.name	默认为CSE给定的默认值，RoundRobin	修改为ConsistentHash即可启用一致性HASH策略，只对指定{servicename}的服务使用策略。
servicecomb.loadbalance.strategy.ConsistentHash.virtualNodeCount	1024	一致性hash一个节点对应的虚拟节点个数。
servicecomb.loadbalance.strategy.ConsistentHash.keyName	x-uid	用于hash的请求上下文里的字段，在invocation.context里。

使用一致性hash，需要提前把keyName写到context里。

支持@ExceptionHandler

NUWA扩展了CSE的默认异常处理类：org.apache.servicecomb.swagger.invocation.exception.DefaultExceptionToProducerResponseConverter，使CSE支持Spring的@ControllerAdvice、@ ExceptionHandler注解。

Spring的@ControllerAdvice、@ExceptionHandler注解，可以实现对请求各种异常的处理。但是由于CSE的限制，并不是@ControllerAdvice、@ExceptionHandler中所有关于异常的处理形式都能兼容，具体可使用的用法如下：

```
@ControllerAdvice
public class ExceptionHandlerService
{
    /**
     * 1、指定处理某个异常，一个异常参数，返回ResponseEntity
     */
    @ExceptionHandler(value = IllegalArgumentException.class)
    public Object processIllegalArgumentException(IllegalArgumentException e)
    {
        //最佳做法
        return new ResponseEntity(new ErrorResponse("001", e.getMessage()), HttpStatus.BAD_REQUEST);
    }

    /**
     * 2、指定处理某个异常，一个异常参数，不返回任何值
     */
    @ExceptionHandler(value = RuntimeException.class)
    public void processRuntimeException(RuntimeException e)
    {
        LOGGER.error("runtime exception intercept", e);
    }
```

```
    /**
     * 3、指定处理某个异常，一个异常参数，返回一个普通对象，HttpStatus为590
     */
    @ExceptionHandler(value = Exception.class)
    public Object processException(Exception e)
    {
        return new ErrorResponse("002", e.getMessage());
    }

    /**
     * 4、指定处理某个异常，无入参
     */
    @ExceptionHandler(value = Error.class)
    public Object processError()
    {
        return new ErrorResponse("003", "error");
    }
}
```

1.5.2 使用 Cloud Map

Cloud Map 背景说明

Cloud Map作为一个资源服务注册和发现中心，允许业务通过Cloud Map SDK来注册和查询资源信息，资源信息包括微服务实例信息（IP地址、端口、版本号、接口定义信息）、中间件集群信息（Cassandra、Redis等集群信息、业务访问需要的用户权限信息）、对外公开URL等信息以及业务自定义的依赖的资源信息。

引入 Cloud Map 模块

```
<dependency>
    <groupId>com.huawei.wisecloud.nuwa</groupId>
<artifactId>nuwa-gpaas-cloudmap</artifactId>
<scope>provided</scope>
</dependency>
```

基础配置项

NUWA中Cloud Map插件配置前缀为：nuwa.cloudmap.，可配置的配置项如表1-4所示。

表 1-4 配置项说明

配置项	默认值	含义
nuwa.cloudmap.serverAddr	nuwamap.huawei.com	可选Cloud Map服务端地址。
nuwa.cloudmap.namespaceName	normal	可选所属命名空间。
nuwa.cloudmap.clusterName	default	可选所属集群名。
nuwa.cloudmap.cacheDir	null	可选本地缓存目录，不配置时，默认地址为：{user.home}/.nuwa/map/{namespaceName}/{serviceName}/{microservice}/{clusterName}。

配置项	默认值	含义
nuwa.cloudmap.requestTimeout	10000	可选http请求超时毫秒数。
nuwa.cloudmap.keepalive	true	可选 http客户端的keepalive参数。
nuwa.cloudmap.connectTimeout	10000	可选 http连接超时毫秒数。
nuwa.cloudmap.maxConnects	Math.min(Math.max(0, processors), 4)	可选 http客户端MaxConnectionsPerHost参数。
nuwa.cloudmap.supportCompress	true	可选 http客户端CompressionEnforced参数。
nuwa.cloudmap.middlewareRefreshMilliseconds	60000	可选中间件客户端定时刷新集群信息的周期。
nuwa.cloudmap.publicUrlRefreshMilliseconds	180000	可选公共url客户端定时刷新url信息的周期。
nuwa.cloudmap.disableCachePublicUrl	false	可选禁止缓存公共url。
nuwa.cloudmap.disableDisasterCache	false	可选禁止本地容灾缓存。
nuwa.cloudmap.enableWebsocketPing	false	可选是否开启websocket心跳。
nuwa.cloudmap.websocketPingIntervalSeconds	20	可选 websocket心跳周期（秒）。
nuwa.cloudmap.websocketPingTimeOutCount	3	可选 websocket心跳最大超时次数。

如果业务使用NUWA时不配置以上任何配置项，则会读取默认配置文件/nuwa/map.properties。

使用CSE组件时，需要用到如下配置：

表 1-5 使用 CSE 组件配置说明

配置项	默认值	含义
nuwa.cloudmap.read	cse	可选，使用CSE时使用，读取时使用的默认注册中心，可配置值为cloudmap和cse。 **说明** 必须在CSE的microservice.yaml中配置。

配置项	默认值	含义
nuwa.cloudmap.write	same to read	可选，使用CSE时使用，是否开启双写注册中心，可配置值为double。 **说明** 必须在CSE的microservice.yaml中配置。

CSE的microservice.yaml中原先有个配置项service_description.environment，在Cloud Map下失去了环境隔离的作用，Cloud Map服务端不会依据该字段隔离环境，因此该配置项在Cloud Map作为注册中心时，仅剩的作用是影响CSE框架启动流程的契约校验逻辑，在development环境，CSE框架发现本地接口契约跟注册中心里已有的内容不一致的话会覆盖注册；而在其他非开发的环境，CSE碰到接口契约不一致会报错终止启动流程。

在Cloud Map场景下如要进行环境隔离，请使用clusterName。

使用 Cloud Map

业务只要引入了Cloud Map插件，NUWA都会对Cloud Map进行初始化。

业务可以直接注入NuwaMapClient到Spring的Bean中：

```
@Autowired
private NuwaMapClient nuwaMapClient;
```

也可以直接通过NuwaMapClientFactory获取：

```
NuwaMapClient mapclient = NuwaMapClientFactory.getNuwaMapClient();
```

1.5.3 DFX 能力

NUWA 系统参数修改

- 环境变量

  在bin/start.sh或者service/bin/ startup_service.sh可以设置控制NUWA行为的环境变量。

  表 1-6 配置项说明

配置项	默认值	含义
LOG_HOME	/opt/huawei/logs/业务名称	NUWA日志所在目录。
JVM_DEBUG	/	是否开启JVM 远程调试，取值范围为yes/no，也可以使用--debug参数启动NUWA。

配置项	默认值	含义
JVM_DEBUG_SUSPEND	/	是否将JVM调试参数中的suspend=n改为=y，取值范围为yes/no，也可以使用--debug-suspend参数开启。 **说明** 必须在开启JVM远程调试参数后才有作用。
CSE_LOCAL_SC	/	是否使用CSE本地文件注册中心，也可以使用--cse-sc-local参数启动NUWA。
HeapSize	自适应	堆栈大小。
JAVA_OPTS	/	可以配置额外的JVM参数。

- GC类型修改

  关于GC配置，在nuwa/bin下面有jvm.options文件，业务也可以根据实际情况覆盖此文件。

Watchdog 的集成

NUWA集成了Watchdog，业务使用start.sh和stop.sh脚本启动或关闭Watchdog。

- 使用start.sh脚本启动Watchdog

```
#!/bin/bash

BIN_PATH=`dirname $0`
APP_ROOT=$BIN_PATH/..

bash $BIN_PATH/stop.sh "$@"
bash $APP_ROOT/nuwa/bin/startup.sh
if [ $# -ne 1 -o "$1" != "watchdog" ];then
bash $APP_ROOT/nuwa/watchdog/bin/watchdog.sh start > /dev/null 2>&1
bash $APP_ROOT/nuwa/watchdog/bin/watchdog.sh mon > /dev/null 2>&1
fi
```

- 使用stop.sh脚本关闭Watchdog

```
#!/bin/bash

BIN_PATH=`dirname $0`
APP_ROOT=$BIN_PATH/..

if [ $# -ne 1 -o "$1" != "watchdog" ];then
    bash $APP_ROOT/nuwa/watchdog/bin/watchdog.sh stop >/dev/null 2>1&
fi

bash $APP_ROOT/nuwa/bin/shutdown.sh
exit 0
```

配置修改

业务包里增加${pkg}/nuwa/watchdog/cfg/common.cfg来覆盖NUWA自带的Watchdog配置文件。

开机自启动

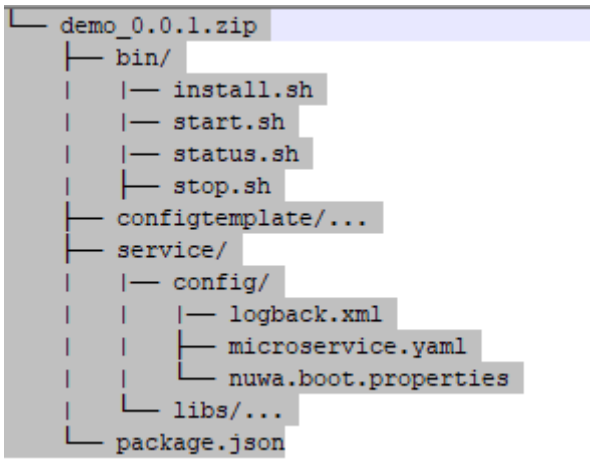
业务的start.sh脚本要求放到开机自启动里，在第一次安装NUWA项目的时候，如果申请了sudo权限，NUWA安装包会自动设置开机自启动。

1.6 打包目录

概述

- 要求业务安装包的目录结构如下：

  图 1-9 业务安装包的目录结构

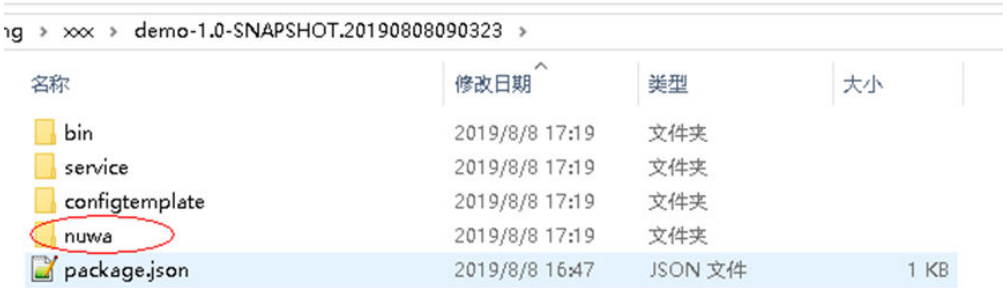
  

  - bin：要求的启停脚本
  - configtemplate：配置中心用到模板文件
  - service：业务包，包含config和libs两个目录，config下为配置文件，libs下为jar包（不包含NUWA包）。
- 自动化部署之后的目录结构

  linux下启动服务的原理：

  在自动化部署以后，NUWA包会被加载到业务包里来。

  图 1-10 自动化部署自动加载 NUWA 包

自动化部署系统会调用bin/start.sh启动服务，bin/start.sh会把NUWA启动起来。NUWA就类似一个tomcat，业务的启动脚本去调用tomcat的启动脚本。

操作步骤

步骤1 添加启动配置文件。

在IDE下面，依赖了NUWA的相关jar包，IDE会把这些jar包加载，所以运行是没有问题的。但是一旦打包以后，是不包含NUWA的jar包的，此时需要添加一个启动配置文件，说明加载NUWA的哪些模块。

添加文件src/main/config/nuwa.boot.properties，引入的插件在后面用逗号拼接。

```
nuwa.system.module.loadingList=nuwa-cse-foundation
```

步骤2 添加启动脚本实例。

start.sh要求先启动NUWA，再启动Watchog，如果参数里有Watchdog，表示启动是由Watchdog拉起的，则不需要启动Watchdog。

stop.sh先关闭Watchdog，再关闭NUWA，如果参数里有Watchdog，表示是Watchdog执行的关闭操作，则不需要关闭Watchdog。

```
#!/bin/bash

BIN_PATH=`dirname $0`
APP_ROOT=${BIN PATH}/..

bash ${BIN_PATH}/stop.sh "$@"
# bash ${APP_ROOT}/nuwa/bin/startup.sh
# 没有使用CSE注册中心
bash ${APP_ROOT}/nuwa/bin/startup.sh --cse-sc-local
if [[ $# -ne 1 || "$1" != "watchdog" ]];then
    bash ${APP_ROOT}/nuwa/watchdog/bin/watchdog.sh start > /dev/null 2>&1
fi
```

status.sh判断进程是否存在，或者其它更严谨的判断服务正常启动的方式。

```
#!/bin/bash

ps -efww |grep "java" |grep "nuwa" |grep "com.huawei.nuwa.boot.loader.NuwaClassPathLauncher" |grep -v grep
```

步骤3 配置Maven打包插件并执行打包命令

在微服务的pom.xml文件，使用Maven的maven-assembly-plugin作为打包插件。

图 1-11 引入 Maven 打包插件

```
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
    <version>2.3.2</version>
    <configuration>
      <excludes>
        <exclude>logback.xml</exclude>
        <exclude>*.yaml</exclude>
        <exclude>*.json</exclude>
        <exclude>alpha/**</exclude>
        <exclude>log4j2.xml</exclude>
        <exclude>*.properties</exclude>
      </excludes>
    </configuration>
  </plugin>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-assembly-plugin</artifactId>
    <version>3.1.1</version>
    <configuration>
      <appendAssemblyId>false</appendAssemblyId>
    </configuration>
    <executions>
      <execution>
        <id>make-auto-deploy</id>
        <phase>package</phase>
        <goals>
          <goal>single</goal>
        </goals>
        <configuration>
          <descriptors>deploy/zip_file.xml</descriptors>
          <finalName>${project.name}-${package_version}</finalName>
          <skipAssembly>false</skipAssembly>
          <appendAssemblyId>false</appendAssemblyId>
          <outputDirectory>${project.build.directory}</outputDirectory>
        </configuration>
      </execution>
    </executions>
  </plugin>
```

一般会将部署有关的文件放在deploy目录里。

```
<?xml version="1.0" encoding="UTF-8"?>
<assembly xmlns="http://maven.apache.org/ASSEMBLY/2.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/ASSEMBLY/2.0.0 http://maven.apache.org/xsd/
assembly-2.0.0.xsd">
    <id>make-auto-deploy-vm</id>

    <includeBaseDirectory>false</includeBaseDirectory>

    <formats>
        <format>zip</format>
    </formats>

    <dependencySets>
        <dependencySet>
            <outputDirectory>service/libs</outputDirectory>
            <scope>runtime</scope>
            <useTransitiveFiltering>true</useTransitiveFiltering>
        </dependencySet>
    </dependencySets>
    <fileSets>
        <fileSet>
            <directory>src/main/config</directory>
            <outputDirectory>service/config</outputDirectory>
            <fileMode>0600</fileMode>
            <directoryMode>0700</directoryMode>
            <lineEnding>unix</lineEnding>
```

```
    </fileSet>
    <fileSet>
      <directory>deploy/bin</directory>
      <outputDirectory>bin</outputDirectory>
      <fileMode>0700</fileMode>
      <directoryMode>0700</directoryMode>
      <lineEnding>unix</lineEnding>
    </fileSet>
    <fileSet>
      <directory>deploy_vm</directory>
      <outputDirectory>./</outputDirectory>
      <includes>
        <include>package.json</include>
      </includes>
      <fileMode>0600</fileMode>
      <directoryMode>0700</directoryMode>
      <filtered>true</filtered>
    </fileSet>
    <fileSet>
      <directory>deploy/configtemplate</directory>
      <outputDirectory>configtemplate</outputDirectory>
      <fileMode>0600</fileMode>
      <directoryMode>0700</directoryMode>
      <lineEnding>unix</lineEnding>
    </fileSet>
  </fileSets>
</assembly>
```

在项目根目录直接执行mvn clean package或者使用IDEA的打包方式，完成后会在target目录下，生成一个可以自动化部署的包，至此打包完成。

----结束

1.7 注意事项

没有以provided方式依赖NUWA，或者忘记在nuwa.boot.properties中添加所需的插件，是业务常犯的一个错误，在使用过程中请注意这两点。

2 使用 Spring Cloud 框架实现应用开发

2.1 Spring Cloud 概述

Spring Cloud为开发人员提供了一些工具来快速构建分布式系统中的一些常见模式（例如配置管理，服务发现，断路器，智能路由，微代理，控制总线，短期微服务和契约测试）。分布式系统的协调导致了样板模式，使用Spring Cloud，开发人员可以快速构建实现这些模式的服务和应用程序。它们可以在任何分布式环境中工作，包括开发人员自己的笔记本电脑、裸机数据中心和Cloud Foundry等托管平台。

约束与注意事项

AppStage提供的SDK是基于Java1.8版本开发的，如果Spring Cloud项目使用Java11及以上版本，则不支持使用AppStage提供的SDK进行应用开发。

2.2 准备工作

开发技能要求

- 熟悉Java语言，能够编写Java语言代码。
- 掌握IaC开发技术，熟悉YAML语言。
- 了解Spring Cloud框架。

环境准备

- 已下载并安装Maven，根据以下步骤配置Maven。
  a. 在<localRepository>标签内添加自己的本地仓库位置路径，这个本地仓库位置是自己创建的。

     D:\apache-maven-3.8.6-bin\repository

     ```
     <localRepository>D:\apache-maven-3.8.6-bin\repository</localRepository>
     ```

  b. 修改Maven默认的JDK版本。

     在<profiles>标签下添加一个<profile>标签，修改Maven默认的JDK版本。

     ```
     <profile>
       <id>JDK-1.8</id>
       <activation>
     ```



```
        <activeByDefault>true</activeByDefault>
        <jdk>1.8</jdk>
    </activation>
    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>
    </properties>
</profile>
```

- 安装并配置IntelliJ IDEA开发工具。
  a. 在IntelliJ IDEA中选择File > Settings > Bulid,Execution,Deployment > Bulid Tools > Maven。
  b. 在User settings file中配置setting.xml。
  c. 在Local repository中配置自定义的Maven仓库地址。
- JAVA开发环境的配置。

  AppStage提供的SDK是基于Java1.8版本开发的，如果Spring Cloud项目使用Java11及以上版本，则不支持使用AppStage提供的SDK进行应用开发。以下步骤以win7环境配置JDK8 64位为例，如果已经下载JDK并配置好环境请跳过本步骤。

  a. 下载**JDK文件**。
  b. 下载完成后按照提示安装，位置自选，比如安装到本地C:\Program Files\Java\jdk1.8.0_131。
  c. 配置Java环境变量：右键“计算机>属性>高级系统设置>环境变量”，进行如下操作。
     i. 新建系统变量JAVA_HOME，变量值为实际JDK安装位置。
     ii. 在Path中添加%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin（注意用英文分号分隔）。
     iii. 新建系统变量CLASSPATH，变量值为%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar。
     iv. 打开命令行窗口，输入“java -version”，显示如**图1**表示配置成功。

     图 2-1 配置成功示例

     ```
     C:\>java -version
     java version "1.8.0_131"
     Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
     Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
     ```

SDK 下载与安装

1. 获取SDK并进行完整性校验。
   - SDK：**nuwa-open-sdk-1.1.0-20240204093135.zip**
   - 完整性校验：**nuwa-open-sdk-1.1.0-20240204093135.zip.sha256**
2. 打开本地Spring Cloud项目。
3. 手动导入jar包。
   a. 在项目目录下新建一个lib目录，存放jar包。
   b. 将本地的jar包复制粘贴至lib目录下。
   c. 将jar包导入到项目中。
      i. 选择“File > Project Structure > Project Settings > Module”。

ii. 单击“+”，选择“JARs or Directories...”。
iii. 选中jar包，单击“apply”。导包完成。

下载 Demo

下载Spring Cloud项目的Demo，参考本文档对Demo源码进行理解，您可以基于Demo进行二次开发，节省开发成本。

Demo下载链接：**huaweicloud-appstage-demo-java-codeHub**。

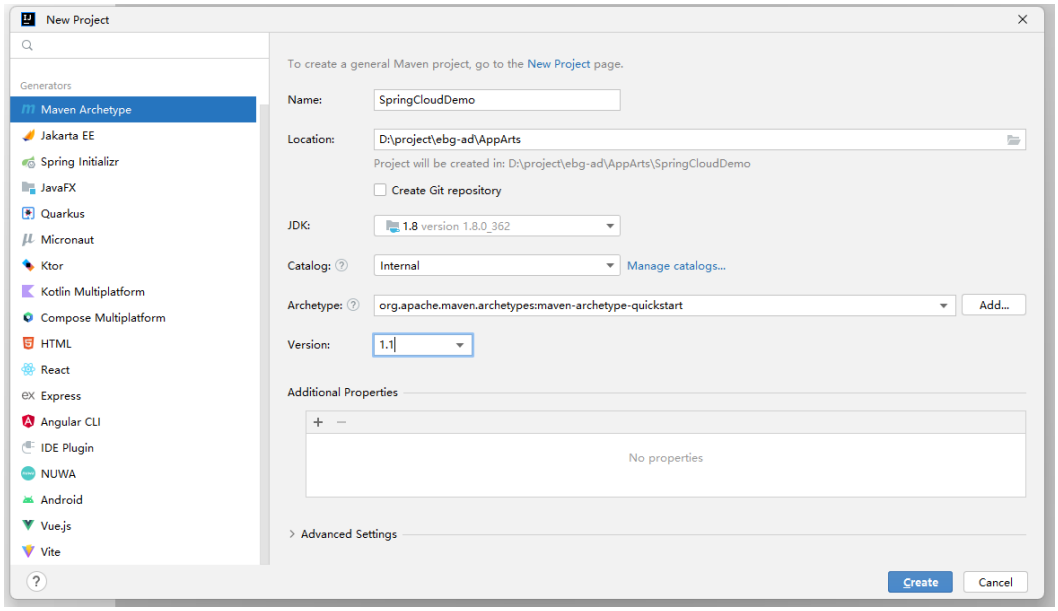
2.3 开发指导

2.3.1 构建 Spring Cloud 工程

创建父工程

步骤1 创建Maven工程。

图 2-2 创建 Maven 工程



步骤2 父pom添加依赖。

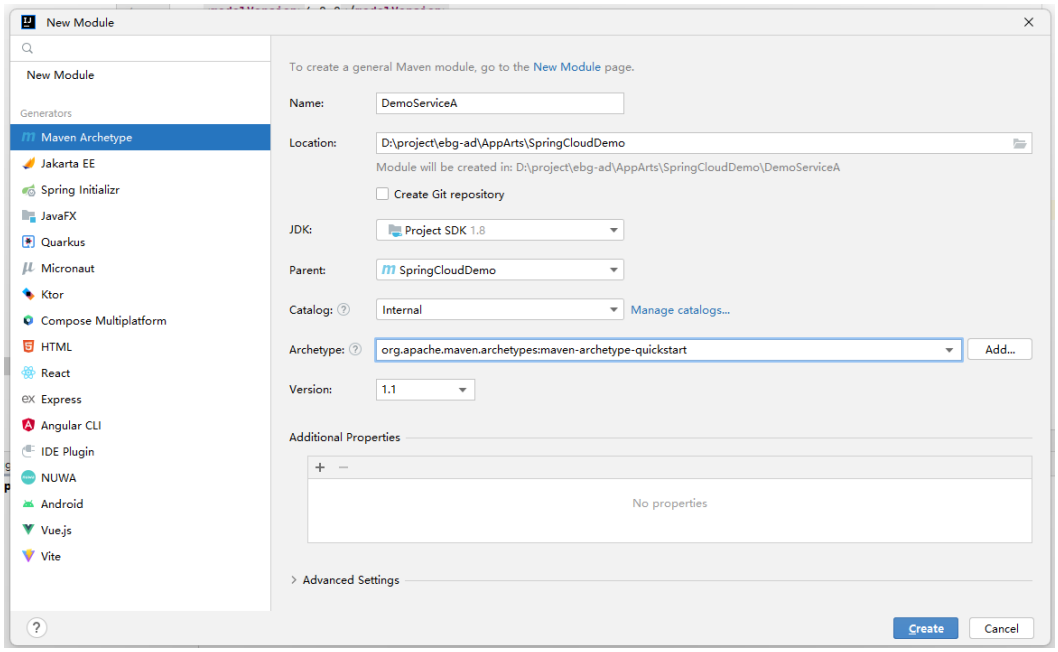
```
<dependencyManagement>
   <dependencies>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-dependencies</artifactId>
         <version>2.2.8.RELEASE</version>
         <type>pom</type>
         <scope>import</scope>
      </dependency>
   </dependencies>
</dependencyManagement>
```

----结束

创建子工程 ServiceA

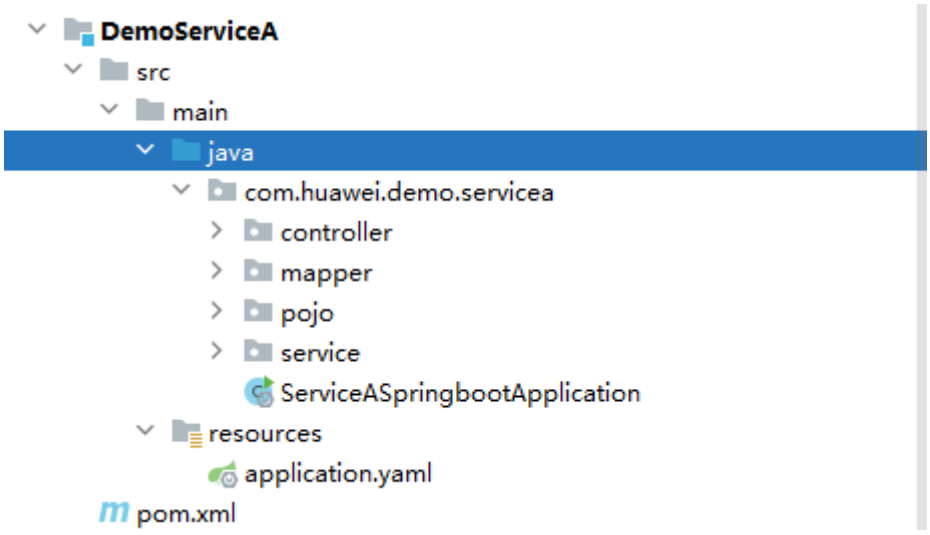
步骤1 创建Maven工程。

图 2-3 创建 Maven 工程



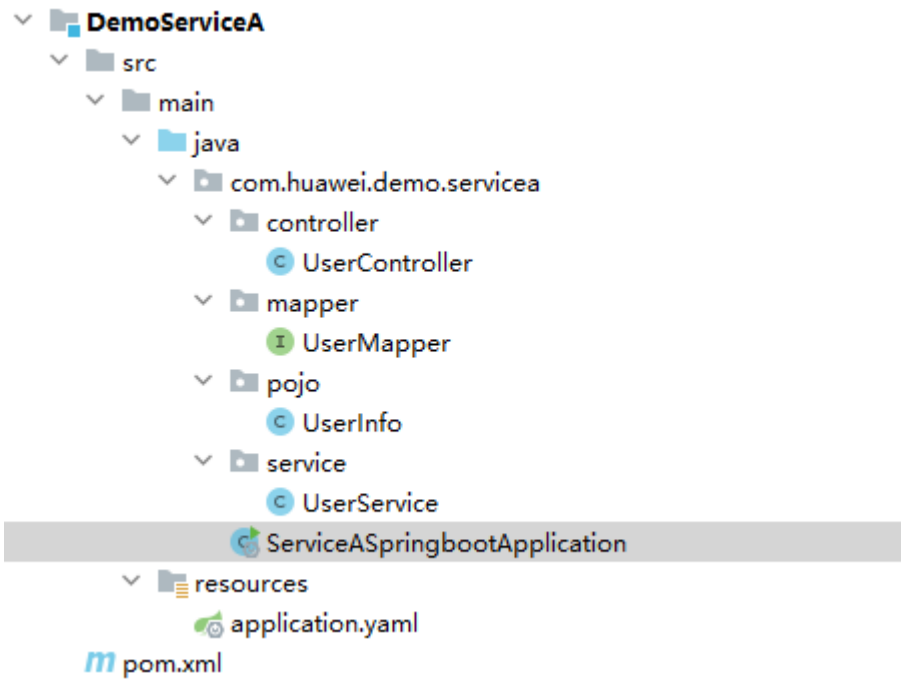
步骤2 新建src目录。

图 2-4 新建 src 目录



步骤3 编写业务代码。

图 2-5 业务代码文件



1. 编写启动类

```
package com.huawei.demo.servicea;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

/**
 * 启动类
 *
 * @author XXX
 * @since 2023-12-05
 */
@SpringBootApplication
@EnableEurekaClient
public class ServiceASpringbootApplication {
    public static void main(String[] args) {
        SpringApplication.run(ServiceASpringbootApplication.class, args);
    }
}
```

2. 编写Controller类

```
package com.huawei.demo.servicea.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.huawei.demo.servicea.pojo.UserInfo;
import com.huawei.demo.servicea.service.UserService;

/**
 * 用户对外接口
 *
 * @author XXX
 * @since 2023-12-06
 */
```

```
@RestController
@RequestMapping("/user")
public class UserController {
    @Autowired
    private UserService userService;

    @GetMapping("/{userId}")
    public UserInfo getUserByName(@PathVariable String userId) {
        return userService.getUserById(userId);
    }
}
```

3. 编写Mapper类

```
package com.huawei.demo.servicea.mapper;

import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Select;

import com.huawei.demo.servicea.pojo.UserInfo;

/**
 * 用户查询
 *
 * @author XXX
 * @since 2023-12-06
 */
@Mapper
public interface UserMapper {
    @Select("select * from demo_user_info where user_id = #{userId}")
    UserInfo getUserById(String userId);
}
```

4. 编写Pojo类

```
package com.huawei.demo.servicea.pojo;

import lombok.Data;

/**
 * user信息
 *
 * @author XXX
 * @since 2023-12-06
 */
@Data
public class UserInfo {
    private String userId;

    private String userName;

    private String phone;

    private String address;
}
```

5. 编写Service类

```
package com.huawei.demo.servicea.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.huawei.demo.servicea.mapper.UserMapper;
import com.huawei.demo.servicea.pojo.UserInfo;

/**
 * userService
 *
 * @author XXX
 * @since 2023-12-06
 */
@Service
public class UserService {
```



```
    @Autowired
    private UserMapper userMapper;

    public UserInfo getUserById(String userId) {
        return userMapper.getUserById(userId);
    }
}
```

6. 配置微服务

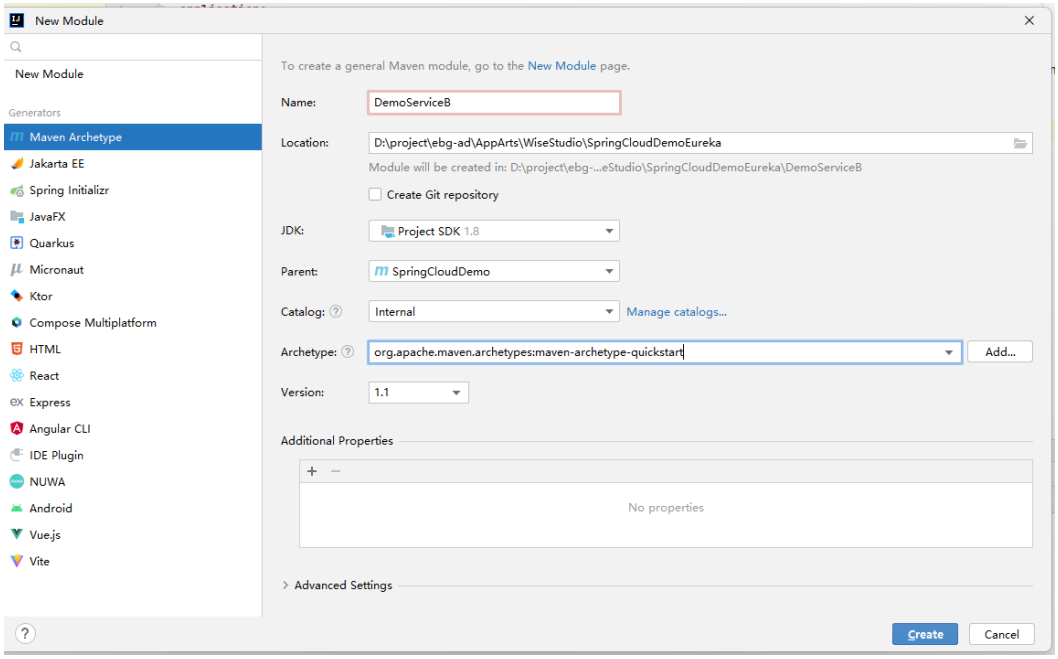
```
server:
  port: 8081
spring:
  application:
    name: demoServiceA
  datasource:
    url: jdbc:mysql://127.0.0.1:3306/spring_cloud_demo?
useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai
    username: root
    password: ***
    driver-class-name: com.mysql.jdbc.Driver

mybatis:
  configuration:
    map-underscore-to-camel-case: true
  type-aliases-package: com.huawei.dmo.servicea
```

----结束

创建子工程 ServiceB

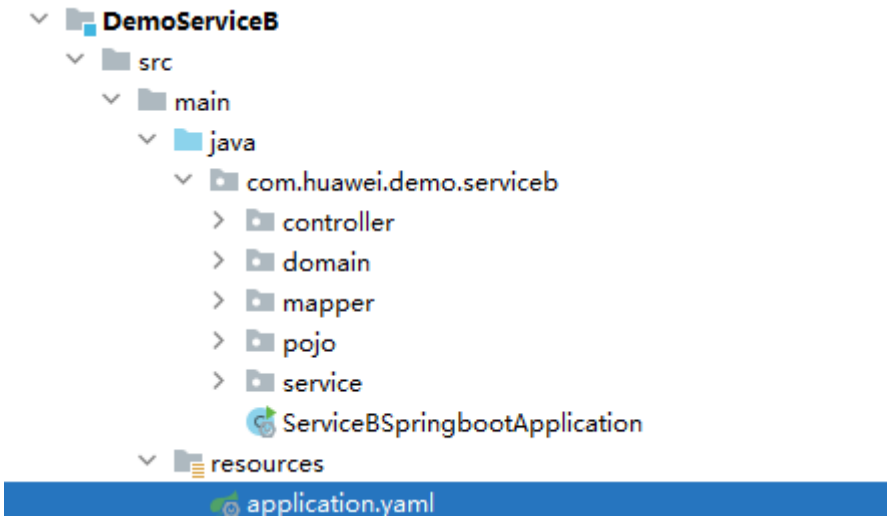
步骤1 创建Maven工程。

图 2-6 创建 Maven 工程



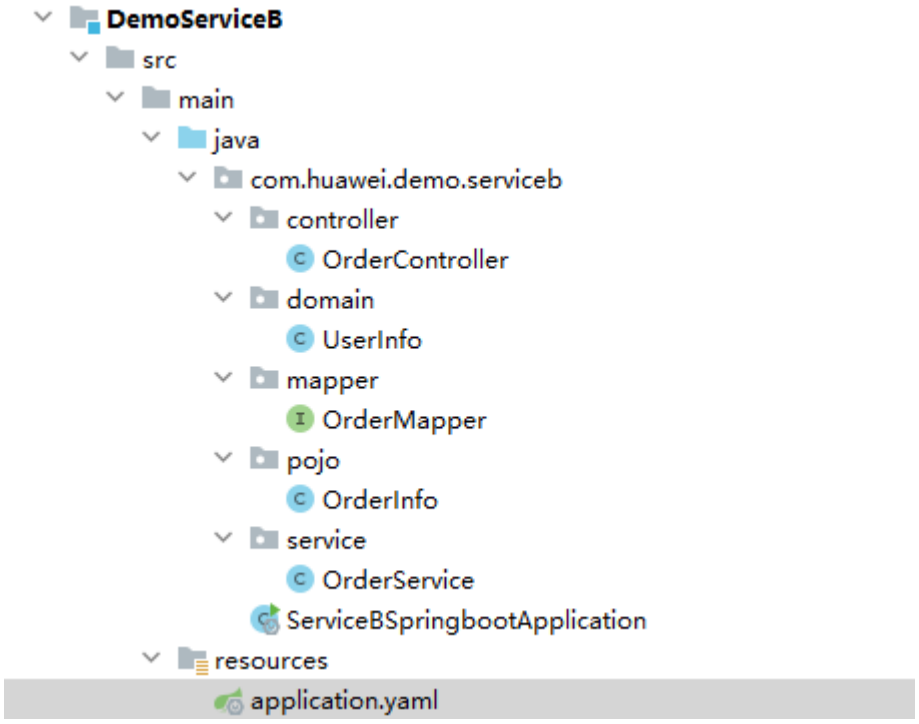
步骤2 新建src目录。

图 2-7 新建 src 目录



步骤3 编写业务代码。

图 2-8 业务代码文件



1. 编写启动类

```
package com.huawei.demo.serviceb;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

/**
 * 功能描述
```



```
 *
 * @author XXX
 * @since 2023-12-05
 */
@SpringBootApplication
@EnableEurekaClient
public class ServiceBSpringbootApplication {
    public static void main(String[] args) {
        SpringApplication.run(ServiceBSpringbootApplication.class, args);
    }

    @Bean
    @LoadBalanced
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
}
```

2. 编写Controller类

```
package com.huawei.demo.serviceb.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.huawei.demo.serviceb.pojo.OrderInfo;
import com.huawei.demo.serviceb.service.OrderService;

/**
 * 用户对外接口
 *
 * @author XXX
 * @since 2023-12-06
 */
@RestController
@RequestMapping("/order")
public class OrderController {
    @Autowired
    private OrderService orderService;

    @GetMapping("/{orderId}")
    public OrderInfo findOrder(@PathVariable String orderId) {
        return orderService.findOrder(orderId);
    }
}
```

3. 编写Domain类

```
package com.huawei.demo.serviceb.domain;

import lombok.Data;

/**
 * user信息
 *
 * @author XXX
 * @since 2023-12-06
 */
@Data
public class UserInfo {
    private String userId;

    private String userName;

    private String phone;

    private String address;
}
```

4. 编写Mapper类

```
package com.huawei.demo.serviceb.mapper;

import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Select;

import com.huawei.demo.serviceb.pojo.OrderInfo;

/**
 * 用户查询
 *
 * @author XXX
 * @since 2023-12-06
 */
@Mapper
public interface OrderMapper {
    @Select("select * from demo_order_info where order_id = #{orderId}")
    OrderInfo getOrderById(String orderId);
}
```

5. 编写Pojo类

```
package com.huawei.demo.serviceb.pojo;

import com.huawei.demo.serviceb.domain.UserInfo;

import lombok.Data;

/**
 * order信息
 *
 * @author XXX
 * @since 2023-12-06
 */
@Data
public class OrderInfo {
    private String orderId;

    private String orderName;

    private String price;

    private String userId;

    private UserInfo userInfo;
}
```

6. 编写Service类

```
package com.huawei.demo.serviceb.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

import com.huawei.demo.serviceb.domain.UserInfo;
import com.huawei.demo.serviceb.mapper.OrderMapper;
import com.huawei.demo.serviceb.pojo.OrderInfo;

/**
 * 功能描述
 *
 * @author XXX
 * @since 2023-12-06
 */
@Service
public class OrderService {

    @Autowired
    private RestTemplate restTemplate;

    @Autowired
```



```
    private OrderMapper orderMapper;

    public OrderInfo findOrder(String orderId) {
        OrderInfo orderInfo=orderMapper.getOrderById(orderId);
        String url = "http://demoServiceA/user/" + orderInfo.getUserId();
        UserInfo userInfo = restTemplate.getForObject(url, UserInfo.class);
        orderInfo.setUserInfo(userInfo);
        return orderInfo;
    }
}
```

7. 配置微服务

```
server:
  port: 8082
spring:
  application:
    name: demoServiceB
  datasource:
    url: jdbc:mysql://127.0.0.1:3306/spring_cloud_demo?
useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai
    username: root
    password: ***
    driver-class-name: com.mysql.jdbc.Driver

eureka:
  client:
    service-url:
      defaultZone: http://localhost:11011/eureka/

mybatis:
  configuration:
    map-underscore-to-camel-case: true
  type-aliases-package: com.huawei.dmo.servicea
```

----结束

创建 Eureka 注册中心

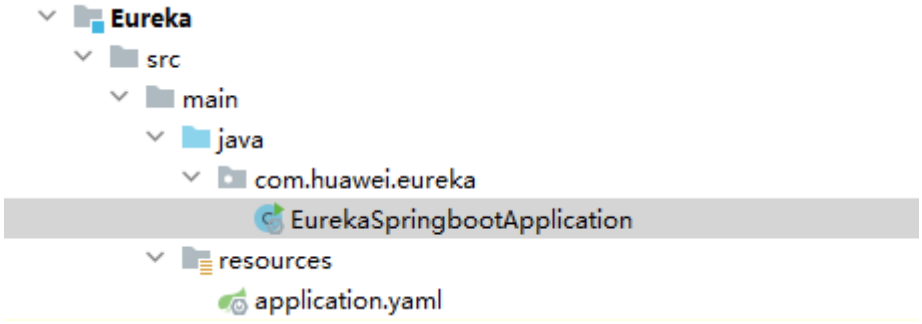
步骤1 创建Eureka微服务。

步骤2 添加依赖。

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-eureka-server</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

步骤3 新建启动类。

图 2-9 新建启动类

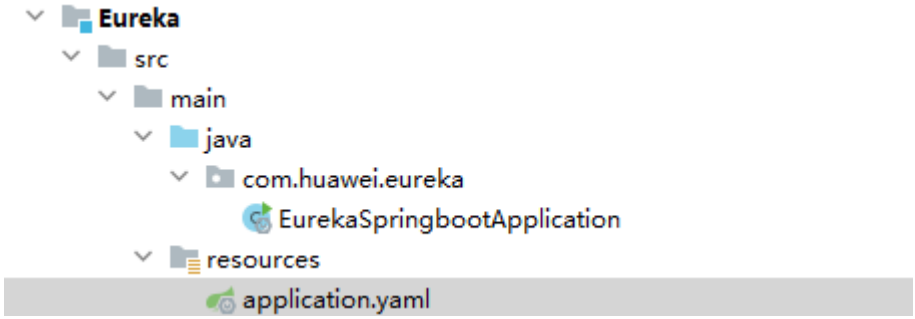


步骤4 启动类中增加注解@EnableEurekaServer，标明注册中心微服务。

```
@SpringBootApplication
@EnableEurekaServer
public class EurekaSpringbootApplication {
    public static void main(String[] args) {
        SpringApplication.run(EurekaSpringbootApplication.class, args);
    }
}
```

步骤5 在Eureka微服务中增加配置文件。

图 2-10 增加配置文件



步骤6 配置注册中心微服务端口等。

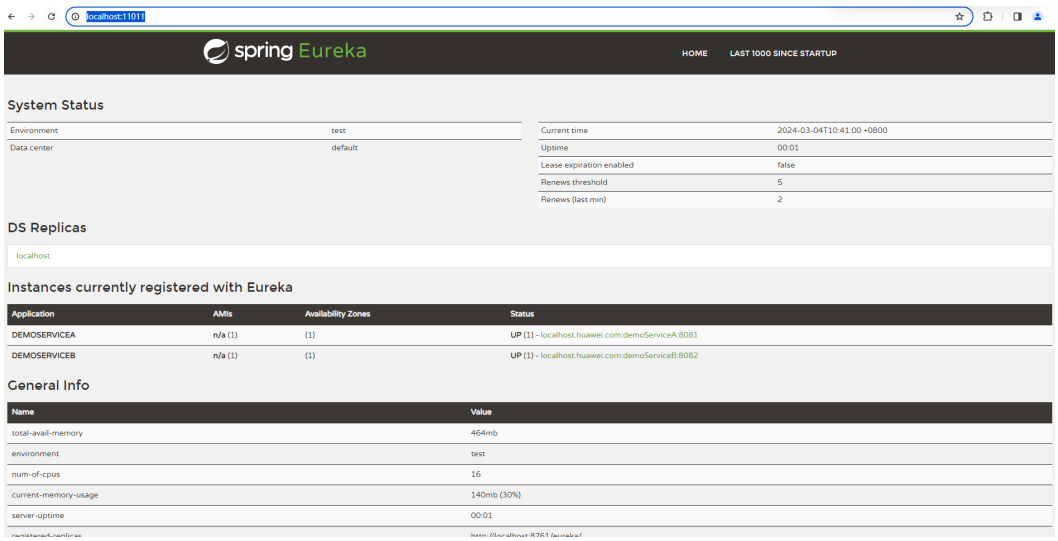
```
server:
  port: 11011
spring:
  application:
    name: eureka

eureka:
  client:
    register-with-eureka: false
    fetch-registry: false
```

步骤7 其他微服务中增加注册中心配置。

```
eureka:
  client:
    service-url:
      defaultZone: http://localhost:11011/eureka/
```

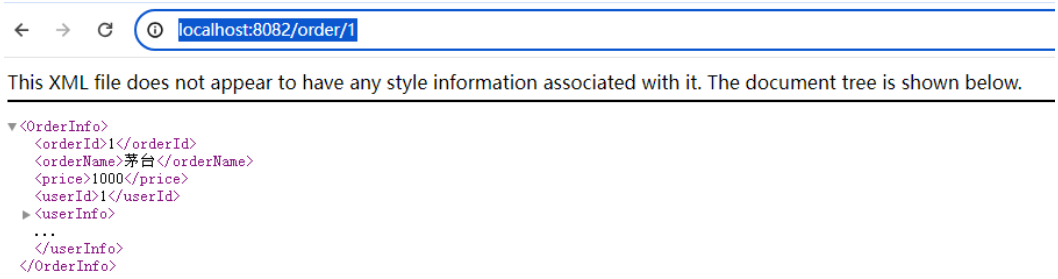
步骤8 依次启动Eureka、DemoServiceA、DemoServiceB，访问http://localhost:11011/，可以看到DemoServiceA、DemoServiceB的信息。

图 2-11 查看微服务注册信息



步骤9 调用接口：http://localhost:8082/order/1。

图 2-12 调用接口



----结束

2.3.2 接入 STS（ACMS）

STS提供了微服务注册以及敏感配置项管理的功能，STS是接入Cloud Map的前提条件，Cloud Map依赖STS认证能力。

操作步骤

接入STS的具体操作请参见**使用STS SDK（Spring Cloud框架）**。

2.3.3 敏感配置项托管

由于业务的敏感配置不能明文地存放在版本包、配置中心、IaC代码中，因此业务可以借助STS敏感配置项的功能，存放业务的敏感配置。

操作步骤

步骤1 使用STS的敏感配置项管理功能，需要在ACMS中录入敏感配置项，具体请参见**录入敏感配置**。

步骤2 在IaC脚本中的业务配置项配置文件中指定敏感配置项取值路径。

此处以增加一个名为spring.redis.password的敏感配置项为例，这个敏感配置项是访问Redis的密码。

```
spring.redis.password: MicroService/{service}/{microservice}/spring.redis.password/default   # 配置路径为：MicroService/服务名/微服务名/敏感配置项名称/敏感配置项标签
```

步骤3 在IaC脚本中的业务配置项属性定义文件中，声明该配置项为敏感配置项。

```
type: object
properties:
  spring.redis.password:
    format: sensitive
```

步骤4 在application.yml配置文件中增加敏感配置项名称的配置。

```
nuwa:
  security:
    config:
      sensitiveWords: spring.redis.password,org.app.protocol-login.oauth.clientSecret,org.app.jwt-key
```

步骤5 启动敏感配置项自动解密。

在启动类中添加@EnableStsEncryptableProperties注解。

----结束

2.3.4 接入 Cloud Map

Spring Cloud通常是使用其自带的Eureka注册中心，使用应用平台可以将Eureka注册中心替换为Cloud Map，Cloud Map除了能够提供服务发现的功能，还可以提供数据库、敏感信息等的纳管功能。

前提条件

Cloud Map依赖STS认证能力，接入Cloud Map前需要先接入STS。

操作步骤

接入Cloud Map的具体操作请参见**使用Cloud Map SDK（Spring Cloud框架）**。

2.3.5 使用 WiseDBA 进行数据库纳管

前提条件

- WiseDBA需要依赖Cloud Map，接入WiseDBA前需要先接入Cloud Map。
- 在AppStage运维中心的WiseDBA中申请数据库并创建Schema，具体请参见创建数据库实例及**创建Schema**。

操作步骤

使用WiseDBA进行数据库纳管的具体操作请参见**使用Rainbow SDK（Spring Cloud框架）**。

2.3.6 集成 OrgID 登录功能

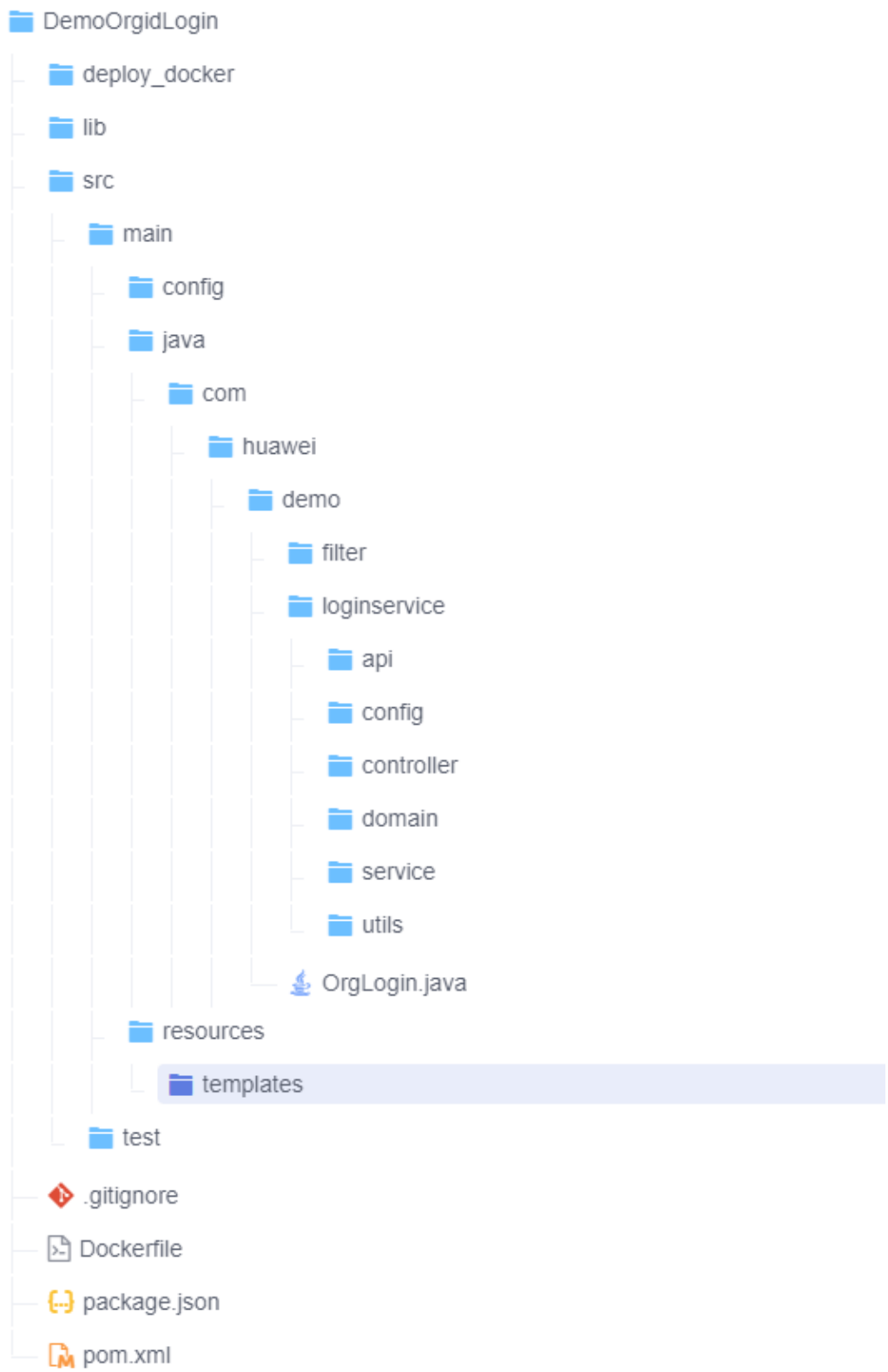
2.3.6.1 概述

支持对接OrgID组织成员账号服务，对接后，通过标准Oauth2.0协议登录到OrgID的应用，从而实现使用OrgID服务对自身应用的组织、部门、成员账号进行管理。

2.3.6.2 了解代码结构

OrgID登录功能Demo的代码结构如**图2-13**所示。

图 2-13 代码结构



- deploy_docker为docker部署配置信息。

```
├─deploy_docker
│  ├─bin
│  │      configparser
```

```
|   |   start.sh                          ----服务在服务器上的启动脚本
|   |
|   ├─configtemplate
|   |       application.properties.tmpl      ----STS域名及微服务STS证书路径
|   |       application.yml.tmpl             ----服务器上环境变量配置信息
|   |       logback-spring.xml.tmpl          ----logback日志配置
|   |       meta.txt
|   |
|   └─resources
|           index.html                    ----demo应用的静态页面（登录后的首页欢迎页）
|           login.html                    ----demo应用的静态页面（登录页，单击按钮跳转登录）
```

- lib目录为工程依赖的jar包。

```
├─lib
|       cloudsoa-security-1.1.14.101.jar
|       gpaas-middleware-common-2.2.6.100.jar
|       nuwa-cloudmap-core-1.0.12.100.jar
|       nuwa-cloudmap-spring-boot-starter-1.0.12.100.jar
|       nuwa-core-3.1.5.100.jar
|       rainbow-core-drds-1.2.18.201.jar
|       rainbow-proxy-1.2.18.102.jar
|       spring-cloud-starter-cloudmap-discovery-1.0.12.100.jar
|       sts-key-sdk-1.1.19.100.jar
|       sts-sdk-base-1.1.19.100.jar
|       sts-spring-boot-1.1.19.100.jar
```

- src目录为项目工程源码目录。

```
└─src
    ├─main
    |   ├─config
    |   |       application.properties                    ----sts域名及微服务sts证书路径（本地
启动时配置）
    |   |       application.yml                          ----工程的环境变量配置（本地启动时配
置）
    |   |       logback-spring.xml                        ----logback日志配置
    |   |
    |   ├─java
    |   |   └─com
    |   |       └─huawei
    |   |           └─demo
    |   |               |   OrgLogin.java                 ----项目启动类
    |   |               |
    |   |               ├─filter
    |   |               |       LogRecordPostFilter.java      ----接口日志过滤器（打印所有请求的
异常信息及请求信息）
    |   |               |
    |   |               └─loginservice
    |   |                   ├─api
    |   |                   |       UserLoginApi.java         ----接口api定义
    |   |                   |
    |   |                   ├─config
    |   |                   |       LoginConfig.java          ----登录拦截器配置（决定对哪些路径放
行）
    |   |                   |       RedisConfig.java          ----Redis数据库连接工具参数配置
    |   |                   |       RestTemplateConfig.java       ----RestTemplate，http请求参数配
置
    |   |                   |       UserLoginInterceptor.java     ----拦截器拦截逻辑，对未登录用户进
行拦截
    |   |                   |
    |   |                   ├─controller
    |   |                   |       UserLoginController.java      ----接口实现controller层
    |   |                   |
    |   |                   ├─domain
    |   |                   |   |   Constance.java          ----常量定义
    |   |                   |   |
    |   |                   |   └─entity
    |   |                   |           BaseResponse.java       ----基础响应体定义
    |   |                   |           UserInfo.java           ----Oauth协议登录后，认证源（OrgID）
返回的用户信息
    |   |                   |
```

```
        ├─service
        │      LoginService.java                    ----接口实现service层
        │
        └─utils
               ApplicationContextUtil.java          ----上下文获取工具类
               JsonUtils.java                       ----JsonUtils工具类，json串与对象之间的相互转换
               JwtUtil.java                         ----Jwt工具类，用于生成token、解析token信息
               RedisUtil.java                       ----Redis工具类，对Redis数据库做数据操作
```

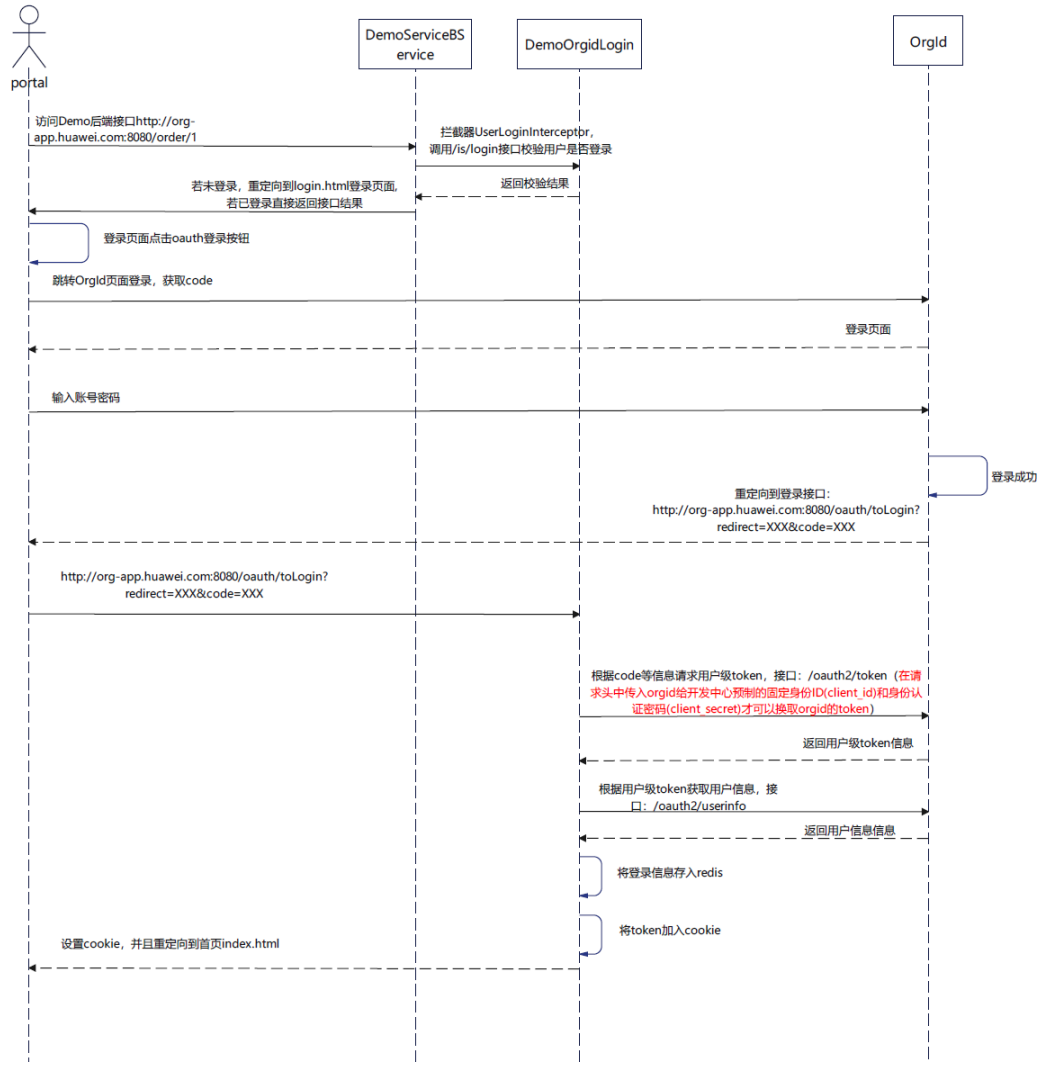
- resources目录，用于存放本地启动时静态资源（欢迎页与登录页html文件），同docker部署时resources目录。

```
    └─resources
        └─templates
               index.html
               login.html
```

2.3.6.3 接口详解

了解 Oauth2.0 协议登录流程（与 OrgID 的交互流程）

图 2-14 Demo 登录流程图



1. 登录获取code：应用A首先需要在OrgID平台上进行注册，并进行相应的配置，比如，首页登录url，退出地址url等，然后用户通过浏览器在OrgID界面单击应用或者直接访问应用服务地址，浏览器向应用A发起登录请求，应用A接收后，会向OrgID发起认证，认证的请求信息为：

```
method: get,
url:    https://orgin-dev.huawei.com/oauth2/authorize,
param:  response_type: code,
        client_id: 应用A的clientID（需要在OrgID页面上获取）
        scope: phone+profile+email（授予应用A的用户信息范围）
        redirect_uri: https://myApp.com/login (在orgID上配置的应用A的首页登录url)
```

如果OrgID监测到用户未登录，则会跳转到OrgID的登录页，进行登录。登录成功后，OrgID会返回一个带有code参数的重定向请求（重定向的地址为应用A在OrgID上配置的首页登录url）。重定向的url实例为：

```
url: https://myApp.com/login?code=lwooqdwgqfCqQkZ-kiZF7zwuiYyYg8MH4gZ0EE1NFk-
L4rnl9v9Nqjw1cW1awpbw5RFhCLYdi-zPa2e-Qru8qBma6KIN7f-
HBrwRLdAm4kNx24B0D0gUXOEs2eezH5UE
```

2. 通过code获取token：通过在第一步中的重定向url路径中获取code。应用A拿着code，向orgID发送一个post请求来获取token，请求信息如下：

```
method:   post,
url:      https://orgid-dev.huawei.com/oauth2/token
formdata: grant_type: authorization_code,
          code: 第一步中获取到的code,
          redirect_url: https://myApp.com/login,
          client_id: 应用A的clientID
          client_secret: 应用A的client_secret
```

获取的响应为：

```
{
   "access_token": "J2Bp_wnA_EJDxCqEdaw2VrE_xol60YeGG1_zKbKzGLBX0l407dkjfe9O-
iOxl7dn87nGekGcxZCwlSzyGH9yp71QRtO36R18qki6foIQYyIN58o1mi8c4-0dFGCwmHII",
   "refresh_token": "pTEUCnloqhCRuJpkueXAeqTNXo2gPRja90Ba9nAe6l1si-d9hK-
njhB3W_3rbDIvH4rIs_59FrJ1Bb6iVkDkKj81NOV7uiNtRbSois8Eweh2QTfBR9Dx9aZ6PcJdJg0-",
   "scope": "phone profile email",
   "token_type": "Bearer",
   "expires_in": 7200
}
```

3. 通过token获取登录的用户信息：得到token后，应用A拿着token，向orgID发送一个get请求来获取用户信息，请求信息如下：

```
method:    get
url:       https://orgid-dev.huawei.com/oauth2/userinfo
headers:
           Authorization: Bearer access_token (其中access_token是第二步请求中的响应体)
```

获取的响应为：

```
{
   "tenant_name": "zzxTest",
   "role": "admin",
   "user_id": "1008600000020011612",
   "user_name": "hid_prsll6r9d8w784c",
   "name": "150******53",
   "mobile": "150******53",
   "tenant": "919008600000119****",
   "employee_code": "00001"
}
```

demo 工程中的接口介绍

Demo工程中的接口如图2-15所示。

图 2-15 Demo 工程中的接口

```
1 implementation
@RequestMapping(value = "/login")
String login();

1 implementation
@RequestMapping(value = "/is/login")
BaseResponse<String> isLogin(HttpServletRequest httpServletRequest) throws Exception;

1 implementation
@RequestMapping(value = "/index")
String index();

1 implementation
@RequestMapping(value = "/oauth2/toLogin")
void toOauth2Login(HttpServletResponse httpServletResponse,
    @RequestParam(value = "code", required = true)
    String code) throws IOException;

1 implementation
@RequestMapping(value = "/app/logout")
void logout(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws Exception;

1 implementation
@RequestMapping(value = "/callback/logout")
void callBackLogout(HttpServletRequest request, HttpServletResponse httpServletResponse) throws Exception;
}
```

- /login接口

  本Demo应用引入了thymeleaf，在配置文件中配置thymeleaf基本参数后，该接口返回为登录页的静态资源login.html，即登录页的界面。

  图 2-16 配置 thymeleaf

  ```
  spring:
    application:
      name: DemoOrgidLogin
    thymeleaf:
      prefix: classpath:/dist
      encoding: UTF-8
      suffix: .html
      mode: HTML5
      cache: false
      servlet:
        content-type: text/html
  ```

- /is/login接口

  该接口为查询当前访问的用户是否登录，如果登录，则返回当前登录的用户信息，否则返回消息为空。与OrgID的登录流程在本Demo中进行，当其他微服务需要判断当前用户登录状态时，内部调用此接口来获取当前登录的用户信息。

- /index接口

  DemoOrgidLogin登录成功后的访问地址首页，同/login接口，返回为登录页的静态资源，即index.html。

- /oauth2/toLogin接口

该接口的接口地址为OrgID侧配置的回调地址，即上述Oauth2.0登录流程中的：回调应用，提供授权码code。

当用户输入账号密码登录后，OrgID会携带授权码code回调此接口，该接口需要依次完成Oauth2.0登录流程中的1~3步，最终拿到用户信息后，根据用户信息生成cookie，建立demo应用与用户浏览器的会话。

- /app/logout接口

  该接口为demo应用的退出接口，当用户需要退出应用时，需完成以下两件事：

  - 需要清除自身会话，完成自身应用退出逻辑。
  - 重定向到OrgID的退出页完成OrgID侧的退出逻辑。

- /callback/logout接口

  该接口为OrgID的回调接口，当用户从OrgID侧发起退出登录时，会通知到应用侧。此时会回调该接口，不同于/app/logout退出接口，该接口只需清理demo应用自身会话，完成自身退出登录。

2.3.6.4 开发者使用 demo 应用配置详细说明

```
spring:
  application:
    name: DemoOrgidLogin                                  # 应用名称，用户可自行决定自身应用名
  thymeleaf:                                              # thymeleaf默认配置
    prefix: classpath:/dist
    encoding: UTF-8
    suffix: .html
    mode: HTML5
    cache: false
    servlet:
      content-type: text/html
  redis:                                                  # Redis连接信息，该信息开发者需配置自己的Redis连接信息
    host: {{spring.redis.host}}                           # Redis连接信息
    port: {{spring.redis.port}}                           # Redis端口号
    password: {{spring.redis.password}}                   # Redis连接密码
  web:
    resources:
      add-mappings: false
      static-locations:
        - classpath:/dist/
server:
  port: 8083                                              # 服务端口号，开发者可根据需求配置自身服务端口号
org:
  url: {{org.url}}                                        # orgid访问域名，为OrgID固定域名不可更改，用于访问
OrgID接口
  web-url: {{org.web-url}}                                # orgid访问域名，为OrgID固定域名不可更改，用于访
问OrgID接口
  app:
    cookie-domain: {{org.app.cookie-domain}}              # 为demo应用与用户建立的会话的cookie域，
需配置为开发者自身应用域名或ip
    jwt-key: {{org.app.jwt-key}}                          # jwt加密秘钥，用于创建和解析jwt串。该值开发者可
自行配置，妥善保管长度需为4的倍数
    ent-point-url: {{org.app.ent-point-url}}              # demoApp自身应用的域名或ip，开发者自行配置
    protocol-login:
      oauth:
        clientId: {{org.app.protocol-login.oauth.clientId}}         # Oauth协议在OrgID侧生成的应用凭证，开发者
需自行配置
        clientSecret: {{org.app.protocol-login.oauth.clientSecret}} # Oauth协议在OrgID侧生成的应用凭证秘钥，
开发者需自行配置
demo:
  login:
    url: {{demo.login.url}}                               # demo应用自身登录url，用于拦截未登录后，跳转重定
向的登录地址，开发者需自行配置
nuwa:                                                     # Cloud Map配置信息
  security:
    config:
```

```
    sensitiveWords: spring.redis.password,org.app.protocol-login.oauth.clientSecret,org.app.jwt-key
cloudmap:
  read: cloudmap
  clusterName: {{nuwa.cloudmap.clusterName}}
  provider:
    cluster: {{nuwa.cloudmap.provider.cluster}}
  serverAddr: {{nuwa.cloudmap.serverAddr}}
  version: {{nuwa.cloudmap.version}}
  namespaceName: {{nuwa.cloudmap.namespaceName}}
logging:
 config: /opt/huawei/app/service/config/logback-spring.xml          # logback日志配置路径
```

2.3.6.5 应用对接的整体流程

操作步骤

步骤1 申请注册HuaweiID账号（已有华为ID账号跳过此步）。

注册链接：**用户登录—华为云 (huaweicloud.com)**，打开后单击“注册”，填写完整信息。

图 2-17 注册 HuaweiID 账号

步骤2 **访问OrgID服务**，创建OrgID组织。

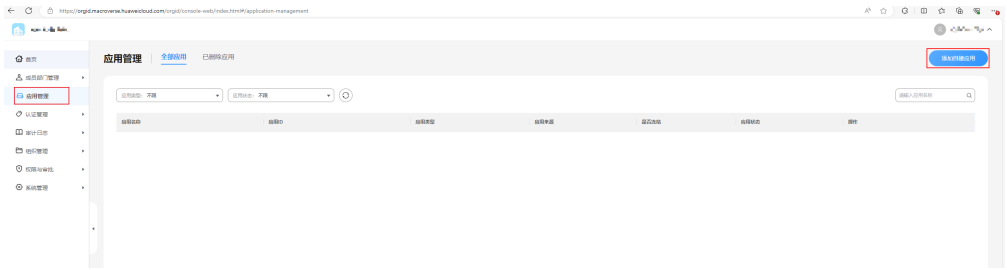
1. 单击“创建组织”，填写组织信息。
2. 单击“进入控制台”，进入控制台。

步骤3 在OrgID管理中心创建自身对接Demo应用信息。

进入控制台单击刚创建好的组织即可进入业务侧管理中心，或直接访问**管理中心**。

1. 选择“应用管理”，单击右上角的“添加自建应用”。

图 2-18 添加自建应用



2. 填写应用信息，上传应用图标，选择Oauth协议。

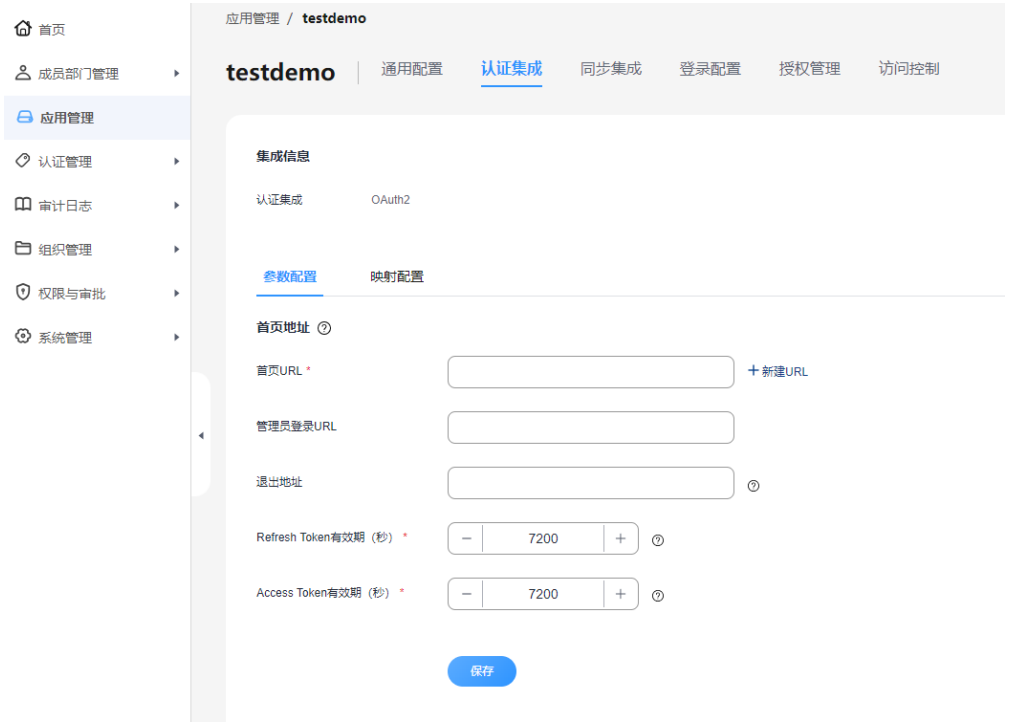
图 2-19 添加自建应用

3. 配置Demo应用信息。

表 2-1 应用配置参数说明

参数	说明
首页URL	Oauth登录交互流程的回调地址。
管理员登录URL	如果Demo应用区分管理员角色，且登录后为管理后台，可自行配置。
退出地址	OrgID退出登录时，通知应用的回调接口。
Token有效期	OrgID侧会话最长时间，默认两小时。

图 2-20 应用参数配置



4. 授权登录用户。
   a. 在应用管理页面选择“授权管理”，单击“授权设置”。

图 2-21 授权设置

   b. 自行选择授权范围。

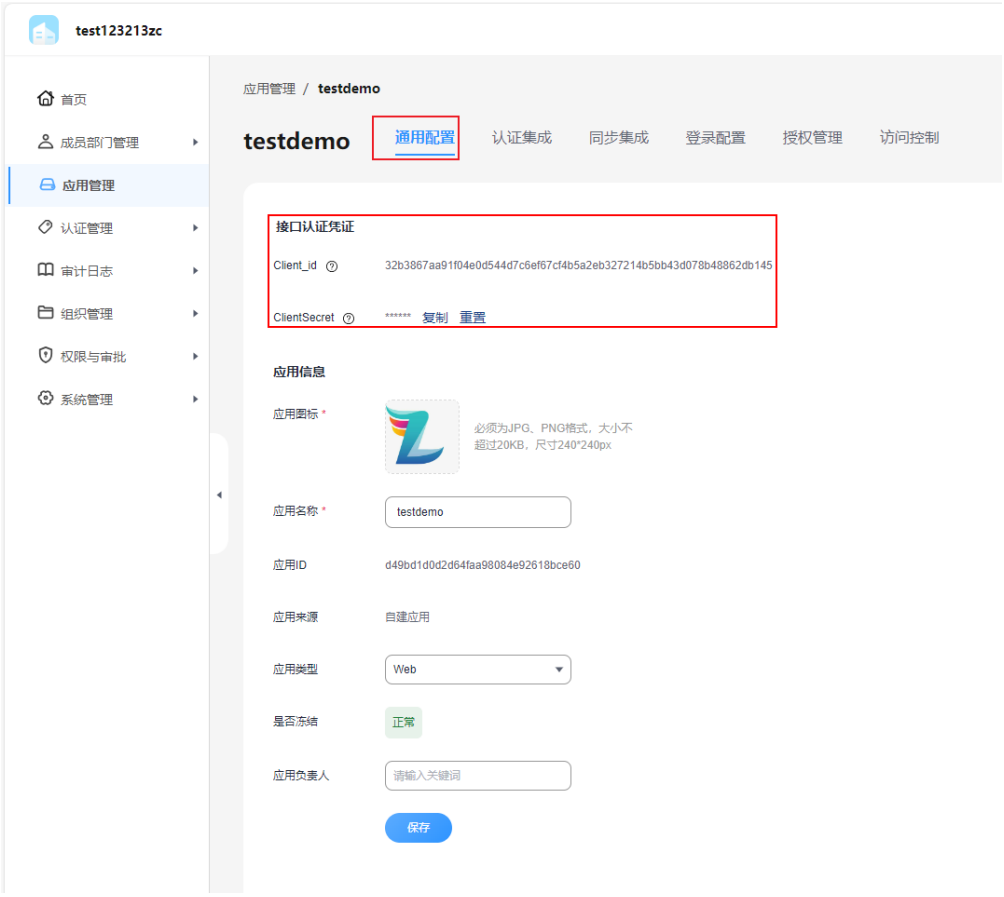
图 2-22 选择授权范围

5. 查看创建好的Demo应用配置秘钥。
   在应用管理页面，选择“通用配置”，复制接口认证凭证。

   说明

   接口认证凭证为OrgID侧生成的应用秘钥，请开发者妥善保管。

图 2-23 查看应用秘钥



6. 修改Demo工程的应用配置信息。

   将application.yml.tmpl中的clientSecret配置为创建好的应用凭证信息，同时将login.html中的按钮登录链接改为已创建好的应用的登录链接，链接地址如图2-24所示。

图 2-24 查看应用的登录链接

7. 修改完成后，启动工程，访问首页登录地址。

通过“域名+端口号+/login”访问首页地址，单击“登录”跳转OrgID登录页，输入账号后，可成功跳转登录后的首页地址。

----结束

2.4 实践案例

本实践以Spring Cloud Demo项目为例，带您体验使用AppStage的开发中心、运维中心及运行时引擎进行工程创建、代码开发、打包发布，部署上线的全过程。具体请参考**基于Spring Cloud框架进行应用上云**。

3 应用平台 IaC 部署代码开发

3.1 IaC 概述

IaC 简介

基础设施即代码（Infrastructure as Code，简称IaC）是一种以YAML作为输入，经由云原生环境管理服务、IaC执行引擎、Operator平台解析和执行，实现环境自动部署以及管理动态基础设施的方法。它强调一致，可重复的供给和变更系统及其配置。当代码发生变更后，可以进行自动化测试，测试完成后可自动化的应用变更到运行系统中。使用基础设施即代码的方法，可以使用敏捷工程的优秀实践（如测试驱动开发，持续集成，持续发布）来更加快速安全的变更基础设施。

IaC 优势

- 变更风险可控
  - 根据变更内容自动识别风险等级，不同等级不同的管控策略。
  - 变更执行过程引入变更前准入评估，变更中灰度评估，变更后结果评估。
  - 声明式IaC代码引入可测试能力，通过软件工程实践保证代码质量。
- 架构治理安全合规可视
  - IaC代码即架构部署视图，可全面了解服务部署涉及的资源及依赖关系。
  - 依赖关系显示声明，在IaC代码中明确服务间共享数据库等公共资源，有助于识别设计中的“BadSmell”。
  - 版本发布和变更端到端可追溯，部署信息精确关联发布版本，来源可端到端溯源。
- 接入效率提升
  - 复用面向资源的CRUD接口，对接IaC3.0不额外增加特定接口。
  - 资源接入基于语言无关的契约定义，接口实现可基于任何语言。
  - 统一的资源接入联调验收环境和验收用例，方便开发调测。
- 开发效率提升
  - 标准化声明式资源定义模板。
  - 全球一份IaC代码，减少重复开发。

  - 内置单云双云等部署模式，开发代码不感知差异。
  - 部署上下文信息（region、az、公有云租户、Cloud Map地址，AIOps地址等）内置到平台，开发人员不感知。
- 重复性变更全自动化
  - 服务依赖的中间件、数据库、一二三方服务全IaC自动化变更。
  - 微服务、函数等应用代码运行时全IaC自动化变更。
  - 网络资源，网络互通策略IaC自动化变更。
  - 例行OS补丁升级，漏洞修复，安全加固等安全变更IaC自动化。

3.2 准备工作

开发技能要求

熟悉YAML语法。

下载 Demo

下载Spring Cloud项目的Demo，参考本文档对Demo源码进行理解，您可以基于Demo进行二次开发，节省开发成本。

Demo下载链接：**huaweicloud-appstage-demo-java-codeHub**。

3.3 了解代码仓结构

3.3.1 概述

IaC3.0支持如下两种场景的部署模式。

- IaC Spec包

  同一个服务下所有微服务的IaC代码在一个仓中管理，打包生成IaC Spec的包，可以实现服务下所有的微服务在同一个服务环境下一键部署，该场景下的IaC代码放置组装服务的代码仓。
- IaC Patch包

  微服务的IaC代码单独管理，通过IaC Spec包创建了服务环境之后，可以通过微服务级别的IaC Patch包进行微服务的独立部署。该场景下的IaC3.0代码放置微服务代码仓与微服务软件代码共同管理。

3.3.2 IaC Spec 包

IaC Spec包典型目录结构如下：

表 3-1 IaC Spec 包结构说明

位置	类型	个数	描述
iacspec_{service}_{version}.zip	文件	1	IaC压缩包。

位置	类型	个数	描述
└── package.json	文件	1	包描述文件。
└── global/	文件夹	1	全局默认的IaC描述，包含完整文件结构。
│ └── meta.yaml	文件	1	变更策略描述，相关说明请参见**component间的编排**。
│ └── environment/	文件夹	1	公共资源。
│ └── resources.yaml	文件	1	公共资源列表，相关说明请参见**在IaC代码中声明资源**。
│ └── values.yaml	文件	1	公共资源参数值。
│ └── {microservice}/	文件夹	0-N	微服务资源。
│ └── resources.yaml	文件	1	微服务资源列表，相关说明请参见**在IaC代码中声明资源**。
│ └── values.yaml	文件	1	微服务资源参数值，通过$ref的方式来引用。
│ └── configs/	文件夹	1	微服务配置目录。
│ └── config_schema.yaml	文件	1	微服务配置字段定义。
│ └── {cluster}_config_records.yaml	文件	0-N	微服务集群配置项。
└── specs/	文件夹	1	环境特定的IaC描述，结构与global相同，但仅包含与global有差异的文件。
│ └── cn_product_cbu/	文件夹	1	中国区生产环境，命名采用站点级Cloud Map的名称，可以在环境管理界面查看可选的站点级Cloud Map名称列表。
│ └── environment/	文件夹	0-1	环境公共资源。
│ └── values.yaml	文件	0-1	公共资源参数值。
│ └── {microservice}/	文件夹	0-N	微服务资源。
│ └── values.yaml	文件	0-1	微服务资源参数值。
│ └── configs/	文件夹	0-1	微服务配置目录。
│ └── {cluster}_config_records.yaml	文件	0-N	微服务集群配置项。
│ └── aaa_product_cbu/	文件夹	1	亚非拉生产环境。

位置	类型	个数	描述
│ └── eu_product_cbu/	文件夹	1	欧洲生产环境。

IaC Spec包样例：

```
└── iacspec_WiseEyeDeployService_1.1.2.zip
    ├── global/                                  # global目录：放置所有规格目录所复用的配置文件
    │   ├── meta.yaml                            # IaC主体描述文件，内容不包含components
    │   ├── environment/...                      # 公共资源，环境
    │   ├── WiseEyeDeployConductorService/...    # 微服务目录，名称需要为WiseEyeDeployService服务下的微服务名称
    │   ├── WiseEyeDeployRepoService/...         # 微服务目录，名称需要为WiseEyeDeployService服务下的微服务名称
    │   └── WiseEyeDeployConsoleService/         # 微服务目录，名称需要为WiseEyeDeployService服务下的微服务名称
    │       ├── config/                          # 微服务配置目录
    │       │   ├── config_records.yaml          # 使用NUWA框架开发的变量定义框架，使用spring框架不生效
    │       │   └── config_schema.yaml
    │       ├── resources.yaml                   # 微服务的资源列表
    │       ├── packages.yaml
    │       └── values.yaml                      # 变量定义文件，被resources.yaml引用
    ├── specs/                                   # 环境特定的IaC描述，结构与global相同，但仅包含与global有差异的文件。
    │   ├── eu_product_cbu/...                   # 欧洲生产环境
    │   ├── aaa_product_cbu/...                  # 亚非拉生产环境
    │   └── cn_product_cbu/                      # 中国区生产环境，命名采用站点级Cloud Map的名称，可以在环境管理界面查看可选的站点级Cloud Map名称列表
    │       ├── WiseEyeDeployConductorService/...  # 微服务目录，名称需要为WiseEyeDeployService服务下的微服务名称
    │       ├── WiseEyeDeployRepoService/...     # 微服务目录，名称需要为WiseEyeDeployService服务下的微服务名称
    │       └── WiseEyeDeployConsoleService/     # 微服务目录，名称需要为WiseEyeDeployService服务下的微服务名称
    │           ├── values.yaml                  # 变量定义文件，被resources.yaml引用
    │           └── config/                      # 微服务配置目录
    │               └── config_records.yaml      # 使用NUWA框架开发的变量定义框架，使用spring框架不生效
    └── package.json                             # 包描述文件（必须）
```

上述目录结构相关文件和目录说明如下：

- package.json
  package.json为包描述文件，字段如表3-2所示。

  表 3-2 package.json 字段说明

位置	类型	必填	描述
type	string	是	包类型，常量：iacspec。
name	string	是	包名称，格式：service/{service-id}，其中service-id为服务ID。
version	string	是	版本号。

  package.json样例：


```
{
  "type": "iacspec",
  "name": "service/com.huawei.wiseeyedeployservice",
  "version": "1.0.0"
}
```

- global文件夹和spec文件夹

  IaC Spec包通过不同规格目录来描述同一个服务在不同用途环境下所需的基础设施。但是，同一服务的不同的规格仍然存在大量相同的配置，需要一种机制来完成不同规格间配置的复用。因此，IaC支持放置一个global目录，其与specs目录同级，用于放置被所有规格目录所复用的配置文件。而各具体规格目录，只需包含与global目录的增量差异文件即可。

  - global文件夹：放置被所有规格目录所复用的配置文件。

    global文件夹里面的微服务都可以被规格文件夹specs中的代码复用（可根据meta.yaml指定复用哪些微服务，取决于你在相应环境的部署规划）。

    global文件夹的作用类似于Java中的父类，spec类似于继承了global的子类，实际部署时还是使用的specs中的文件，但specs中的文件可以继承和复用global文件。

    - meta.yaml：描述变更的组件与过程。
    - WiseEyeDeployConsoleService：描述要变更的微服务。
      - resources.yaml：微服务变更的主体文件，其他所有的values.yaml、config文件夹中的yaml等文件都围绕此文件展开。文件名必须为resources.yaml。
      - 其他文件：为变量配置文件，其定义的内容都会被resources.yaml引用，文件名称可自定义。
    - 其他微服务文件夹：结构同WiseEyeDeployConsoleService。
  - spec文件夹：同一个服务在不同用途环境下所需配置文件（基础设施）。这个文件目录是必须的。

    specs是在环境上部署服务时，最终使用的配置文件，当部署服务时，第一关注点和入口就是specs。

    specs目录下的规格文件夹，命名采用站点级Cloud Map的名称（cn_product_cbu、eu_product_cbu）。可以在环境管理界面查看可选的站点级Cloud Map名称列表。

    当某个规格被选用于部署时，会先将该规格目录下所有文件与global目录进行合并，得到该规格目录最终的所有配置文件，再进行部署动作。

合并策略：如果文件的相对路径相同，则规格目录下的文件保留，global目录下的文件被覆盖，其他文件则共存。

说明

global目录应包含完整的YAML内容，即：其下meta.yaml通过$ref引用的YAML内容都存在于global目录中。这样能确保即使规格目录为空，也能引用到global目录的默认参数，从而完成部署。

3.3.3 IaC Patch 包

IaC Patch包典型目录结构如下：

表 3-3 IaC Patch 包结构说明

位置	类型	个数	描述
iacpatch_{microservice}_{version}.zip	文件	1	IaC压缩包。
└── package.json	文件	1	包描述文件。
└── global/	文件夹	1	全局默认的IaC描述，包含完整文件结构。
│ └── resources.yaml	文件	1	微服务资源列表，相关说明请参见**在IaC代码中声明资源**。
│ └── values.yaml	文件	1	微服务资源参数值。
│ └── configs/	文件夹	1	微服务配置目录。
│ └── config_schema.yaml	文件	1	微服务配置字段定义。
│ └── {cluster}_config_records.yaml	文件	0-N	微服务集群配置项。
└── specs/	文件夹	1	环境特定的IaC描述，结构与global相同，但仅包含与global有差异的文件。
│ └── cn_product_cbu/	文件夹	1	中国区生产环境，命名采用站点级Cloud Map的名称，可以在环境管理界面查看可选的站点级Cloud Map名称列表。
│ └── values.yaml	文件	0-1	微服务资源参数值。
│ └── configs/	文件夹	0-1	微服务配置目录。
│ └── {cluster}_config_records.yaml	文件	0-N	微服务集群配置项。
│ └── aaa_product_cbu/	文件夹	1	亚非拉生产环境。
│ └── eu_product_cbu/	文件夹	1	欧洲生产环境。

IaC Patch包样例：

```
└── iacpatch_WiseEyeDeployConsoleService_1.1.2.zip
    ├── global/                          # global目录：放置所有规格目录所复用的配置文件
    │   ├── config/                      # 微服务配置目录
    │   │   ├── config_records.yaml      # 使用NUWA框架开发的变量定义框架，使用spring框架不生效
    │   │   └── config_schema.yaml
    │   ├── resources.yaml               # 微服务的资源列表
    │   ├── packages.yaml
    │   └── values.yaml                  # 变量定义文件，被resources.yaml引用
    ├── specs/                           # 环境特定的IaC描述，结构与global相同，但仅包含与global有差异的文件
```



```
    ├── eu_product_cbu/...              # 欧洲生产环境
    ├── aaa_product_cbu/...              # 亚非拉生产环境
    └── cn_product_cbu/                  # 中国区生产环境，命名采用站点级Cloud Map的名称，可以在环境管理界面查看可选的站点级Cloud Map名称列表
        ├── values.yaml                 # 变量定义文件，被resources.yaml引用
        └── config/                     # 微服务配置目录
            └── config_records.yaml     # 使用NUWA框架开发的变量定义框架，使用spring框架不生效
└── package.json                        # 包描述文件（必须）
```

上述目录结构相关文件和目录说明如下：

- package.json为包描述文件，字段如表3-4所示。

  表 3-4 package.json 字段说明

位置	类型	必填	描述
type	string	是	包类型，常量：iacpatch。
name	string	是	包名称，格式：service/{service-id}/{component-name}，其中{service-id}为服务ID，{component-name}为组件名称。
version	string	是	版本号。

  package.json样例：

  ```
  {
    "type": "iacpatch",
    "name": "service/com.huawei.wiseeyedeployservice/WiseEyeDeployConsoleService",
    "version": "1.0.0"
  }
  ```

spec包通过不同规格目录来描述同一个服务在不同用途环境下所需的基础设施。但是，同一服务的不同的规格仍然存在大量相同的配置，需要一种机制来完成不同规格间配置的复用。因此，IaC支持放置一个global目录，其与specs目录同级，用于放置被所有规格目录所复用的配置文件。而各具体规格目录，只需包含与global目录的增量差异文件即可。

- global文件夹：放置被所有规格目录所复用的配置文件。

  global文件夹里面的微服务都可以被规格文件夹specs中的代码复用（可根据meta.yaml指定复用哪些微服务，取决于你在相应环境的部署规划）。

  global文件夹的作用类似于Java中的父类，spec类似于继承了global的子类，实际部署时还是使用的specs中的文件，但specs中的文件可以继承和复用global文件。

- spec文件夹：同一个服务在不同用途环境下所需配置文件（基础设施）。这个文件目录是必须的。

  specs是在环境上部署服务时，最终使用的配置文件，当部署服务时，第一关注点和入口就是specs。

  specs目录下的规格文件夹，命名采用站点级Cloud Map的名称（cn_product_cbu、eu_product_cbu）。可以在环境管理界面查看可选的站点级Cloud Map名称列表。

当某个规格被选用于部署时，会先将该规格目录下所有文件与global目录进行合并，得到该规格目录最终的所有配置文件，再进行部署动作。

合并策略：如果文件的相对路径相同，则规格目录下的文件保留，global目录下的文件被覆盖，其他文件则共存。

说明

global目录应包含完整的YAML内容，即：其下meta.yaml通过$ref引用的YAML内容都存在于global目录中。这样能确保即使规格目录为空，也能引用到global目录的默认参数，从而完成部署。

3.3.4 global 与 specs 的协同关系

本节以WiseChaos的IaC代码为例，详细描述代码运行原理及涉及的各个IaC文件的作用，微服务WiseChaos的整体IaC代码结构如下：

```
iac3.0                                  # IaC3.0代码根目录：目录名字可自定义
└─ WiseEyeChaosMonkeyService             # 此级目录为服务级目录，名字与所部署的服务名称相同
   ├─ global                          # global目录：放置所有规格目录所复用的配置文件
├─ WiseEyeChaosIssueMgrService
├─ WiseEyeChaosManageService
├─ WiseEyeChaosMonkeyExecutor
│  └─ config
│        ├─ business_config.yaml
│        ├─ env.yaml
│        ├─ envs_dynamic.yaml
│        ├─ hosts.yaml
│        └─ sidecar_aiops_param.json
│     └─ resources.yaml
│  └─ values.yaml
├─ WiseEyeChaosMonkeyPortal
      │  └─ config
      │        └─ sidecar_aiops_param.json
      │  └─ resources.yaml
      │  └─ values.yaml
└─ WiseEyeChaosPortal
      └─meta.yaml
   ├─specs             # 规格文件，描述了每个规格的定制化需求，最终部署的时候以specs文件为准
      ├─cn_product_cbu
         ├─WiseEyeChaosMonkeyExecutor
      │  └─config
      │        ├─ envs_dynamic.yaml
      │        ├─ hosts.yaml
      │        └─ sidecar_aiops_param.json
      │  └─values.yaml
      ├─WiseEyeChaosMonkeyPortal
            │  └─config
            │        └─ sidecar_aiops_param.json
            │  └─values.yaml
            └─ meta.yaml
      ├─aaa_product_cbu
      └─eu_product_cbu
   └─package.json
```

当对此微服务的IaC代码打包合并时：

- specs中存在而global中不存在的文件，使用specs中的文件。
- specs中不存在而global中存在的文件，使用global中的文件。
- specs和global中都存在的，则使用specs中的文件。

以/iac3.0/WiseEyeChaosMonkeyService/specs/cn_product_cbu为例，在specs中的cn_product_cbu目录和global中都有WiseEyeChaosMonkeyExecutor和WiseEyeChaosMonkeyPortal微服务文件夹。

合并后最终呈现的文件目录如下：

说明

“覆盖”指的是完全取代，不是内容合并。

```
cn_product_cbu
  ├─ meta.yaml                      # 来自于specs目录，覆盖global中的文件
  ├─ WiseEyeChaosMonkeyExecutor
  │   ├─ values.yaml                # 来自于specs目录，覆盖global中的文件
  │   ├─ resources.yaml              # 来自于global中的文件
  │   └─ config
  │       ├─ business_config.yaml    # 来自于global中的文件，因为specs中不存在
  │       ├─ env.yaml               # 来自于global中的文件，因为specs中不存在
  │       ├─ envs_dynamic.yaml       # 来自于specs目录，覆盖global中的文件
  │       ├─ hosts.yaml             # 来自于specs目录，覆盖global中的文件
  │       └─ sidecar_aiops_param.json # 来自于specs目录，覆盖global中的文件
  ├─ WiseEyeChaosMonkeyPortal
  │   ├─ values.yaml                # 来自于specs目录，覆盖global中的文件
  │   ├─ resources.yaml              # 来自于global中的文件，因为specs中不存在
  │   └─ config
  │       └─ sidecar_aiops_param.json # 来自于specs目录，覆盖global中的文件
  ├─ WiseEyeChaosIssueMgrService      # 来自于global中的文件，因为specs中不存在
  ├─ WiseEyeChaosManageService        # 来自于global中的文件，因为specs中不存在
  └─ WiseEyeChaosPortal              # 来自于global中的文件，因为specs中不存在
```

3.4 开发微服务部署代码

3.4.1 变更流程编排开发

3.4.1.1 变更流程编排概述

在一次完整的业务变更中，往往会涵盖多种类型、多个模块的变更，如集群扩容、申请ELB、创建数据库、软件升级等等。在IaC的语境下，每一个变更本质上都是IaC资源的变更。在一次完整的业务变更中，部分资源的变更依赖于其他资源的变更，如为一个微服务创建nuwa实例之前往往需要先创建该微服务的数据库。

变更流程编排就是使用IaC代码对各资源在具体变更过程中的依赖关系、先后顺序进行代码化描述。变更流程编排本质上就是描述各模块、各资源之间的依赖关系。在变更过程中，IaC将根据由依赖关系生成的有向无环图顺序执行各资源的变更过程。

变更流程编排包含两部分：

- component内部各资源的编排
- 各component间的编排

3.4.1.2 component 内部编排

组件内部编排在spec包各组件的resources.yaml文件中描述，通过为资源指定dependsOn属性表达依赖关系。

component内部允许同一个资源出现多次，这表示同一个资源的不同变更阶段，这一场景下该资源的所有节点必须声明alias字段并且alias取值必须在component内全局唯一，同一个资源的所有alias之间必须显式地在dependsOn字段中声明串行依赖。

dependsOn是列表类型，每个元素使用type、name、alias等字段描述对其他资源的引用，其中name字段是必填字段，type、alias是可选字段；component解析某资源的

dependsOn时，会根据元素属性从相同component中搜索满足条件的资源作为当前资源的依赖。

部分资源之间已经有隐式引用关系，系统自动添加dependsOn，不需要再显式声明。每种类型的哪些属性隐含引用关系，可以参考其文档，具体请参见资源介绍。

说明

资源不能有循环依赖（A dependsOn B,B dependsOn C,C dependsOn A），不能依赖自己。

以下示例定义了两个资源，一个名为chaosmonkey-elb的ELB，一个名为chaosmonkey-slb的SLB；chaosmonkey-slb依赖于chaosmonkey-elb。在变更时，先变更chaosmonkey-elb，变更成功后再变更chaosmonkey-slb，如果chaosmonkey-elb变更失败，则不会变更chaosmonkey-slb。

```
- name: chaosmonkey-elb                 # 资源名称
  type: WiseCloud::LoadBalancer::ELBV2   # 资源类型
  properties:
    listeners:
      - name: listener
        protocol: HTTP
        protocolPort: 80
        poolName: pool_a
    pools:
      - name: pool_a
        protocol: HTTP
- name: chaosmonkey-slb                 # 资源名称
  type: WiseCloud::LoadBalancer::SLB     # 资源类型
  dependsOn:                            # 定义对其他资源的依赖
    - name: chaosmonkey-elb             # 依赖的资源名称
  properties:
    elbName: chaosmonkey-elb
    elbPoolNames: ["pool_a"]
    deployVersion: 1.4.12
    slbConfigs:
      targets:
        - clusterName: mgr
      routes:
        - location: /
          target: mgr
```

3.4.1.3 component 间的编排

3.4.1.3.1 component 间的编排概述

IaC Spec类型的IaC代码可以在global目录下的meta.yaml中描述资源的变更流程以及变更策略，component间的编排在spec包中的meta.yaml文件中描述，涉及applyPipeline/pipelines两个字段。pipelines中支持定义多个流程，applyPipeline描述本次变更要使用的流程。

```
type: WiseCloud::Environment          # 描述当前环境类型，当前为固定值WiseCloud::Environment
applyPipeline: environment-deploy     # 定义默认选用的组件编排流水线名称，当前默认使用environment-deploy
pipelines: ...
```

表 3-5 meta.yaml 字段说明

字段	说明
type	描述当前环境类型，当前为固定值WiseCloud::Environment。

字段	说明
applyPipeline	定义默认选用的组件编排流水线名称，当前默认使用environment-deploy。
pipelines	pipelines中支持定义多个流程。

3.4.1.3.2 定义 pipeline

pipeline能力丰富，通过设计pipeline可以实现精巧的多阶段部署、部分变更、主动暂停等复杂场景的变更编排。在以下样例中，定义了一个pipeline。

```
applyPipeline: environment-deploy          # 定义默认选用的组件编排流水线名称，当前默认使用environment-deploy
pipelines:
- name: environment-deploy                 # 流水线的名称，用户可自定义
  action: Serial                           # 串行编排
  tasks:
  - action: Apply                          # 表示进行任务变更
    component:
      name: environment                    # 变更的组件名称，对应微服务目录，根据具体情况修改
  - action: Parallel                       # 表示微服务是并行编排
    tasks:
    - action: Apply                        # 表示进行任务变更
      component:
        name: WiseEyeChaosMonkeyMgrService  # 变更的组件名称，对应微服务目录，根据具体情况修改
    - action: Apply                        # 表示进行任务变更
      component:
        name: WiseEyeChaosMonkeyPortal     # 变更的组件名称，对应微服务目录，根据具体情况修改
```

以上示例定义了一个名为default的流程，该流程中编排了三个component。在变更时，先变更名为environment的component，该component变更完成后再并行变更名为WiseEyeChaosMonkeyMgrService和WiseEyeChaosMonkeyPortal的component。

name/action都是一个流水线的基本信息，任务的编排是需要声明不同类型的task实现；当前支持的任务类型包括：Serial（串行）/Parallel（并行）/Apply（声明式任务执行），其中，Serial/Parallel不涉及资源部署，我们仅通过此类任务将需要部署的资源串联起来，如果把pipeline比作一棵树的声明，那么所有tasks都是这棵树的子节点，Apply是这棵树的叶子节点，只有叶子节点被执行时才会对环境产生影响。

图 3-1 pipeline 任务声明

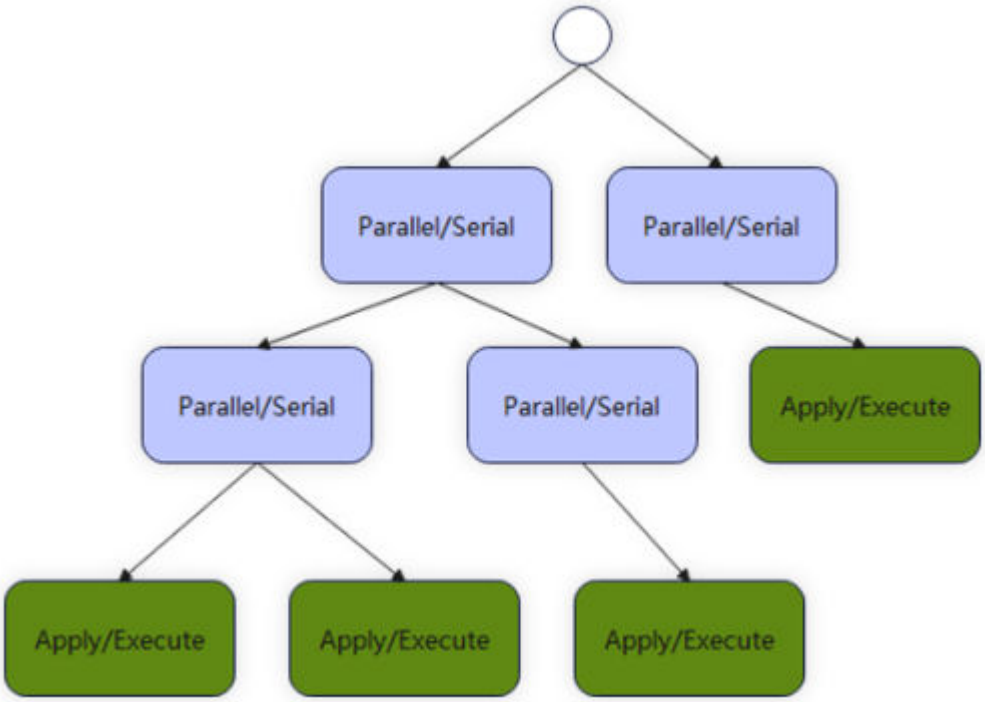


表 3-6 组件编排流水线（pipeline）对象定义

字段名称	字段类型	必选	描述
name	String	是	流水线的名称，用户可自定义。
action	String	是	编排方式，可选值Serial/Parallel。 • Serial，串行执行，其参数tasks下的子任务会根据定义的顺序依次执行。 • Parallel，并行执行，其参数tasks下的子任务会同时执行。
tasks	List<PipelineTask>	是	声明流水线的子任务，通过声明pipeline的任务，实现对资源部署流程的编排。

表 3-7 PipelineTasks 字段说明

字段名称	字段类型	必选	描述
name	String	否	子任务的名称，用户可自定义。
action	String	是	编排方式，可选值。Serial/Parallel/Apply。

字段名称	字段类型	必选	描述
tasks	List<PipelineTask>	否	编排方式为Serial/Parallel时必选。
component	ApplyComponent	否	需要执行变更的component，编排方式为Apply时必选。

任务的数据结构在设计上支持了无限层级嵌套，配合action属性可以实现任意的任务串并行编排。

表 3-8 component 字段说明

字段名称	必选	描述
name	是	组件名称，填写的值必须为已声明的组件，否则在校验阶段会报错。
resources	否	选择变更的资源，选填，该参数可用于部分部署。 如果不填写则会变更此component下所有资源，使用name+type+alias确认唯一资源（alias是资源的别名，如果component下不存在同名资源则不需要填写）。 如果填写component中不存在的资源，系统会在校验阶段报错；resources下可以定义properties，这样在执行此任务时，这些properties的值就会去覆盖你在resources.yaml下定义的资源参数。

3.4.1.3.3 部分变更

一般情况下，一个服务级的环境中会包含ECS/CCE集群、中间件资源、微服务配置和部署等资源，原则上一般将ECS/CCE资源、中间件资源集中放在一个component中，各微服务的资源分别放在一个component中。默认情况下，一个component被变更时，其内部的所有资源都会以由IaC代码描述的状态为目标状态进行变更。

在部分场景下，我们希望某次变更时只变更一部分具体资源，而跳过其他资源的变更，这一功能可以通过在pipeline中指定component的resources属性实现。常见的应用场景如：

component中定义了一个微服务的多个集群（集群A、集群B）的资源，由于种种原因跳过IaC直接在管理台修改了集群A的replica，本次变更只希望变更集群B，不希望对集群A带来任何影响。component的部署过程可划分为几个步骤，先变更灰度集群，再变更现网集群。

结合多阶段变更和部分变更的能力，可以很方便地实现。

样例：

```
type: WiseCloud::Environment
applyPipeline: chaos_gray_upgrade
```

```
components:
  - name: WiseEyeChaosPortal
    values:
      replicas: 3
      grayStage:
        grayInstances: 100
        grayProcess: FINISHED
        grayStatus: 2
    resources:
    - name: chaos-portal-grey
      type: WiseCloud::MicroService::NuwaContainer
      properties:
        microserviceName: WiseEyeChaosPortal
        clusterName: portal_grey
        replicas:
          $jq: '.values.replicas'
        terminationGracePeriodSeconds: 30
        network:
          slbEnable: true
        grayStage:
          $jq: '.values.grayStage'
    - name: chaos-portal
      type: WiseCloud::MicroService::NuwaContainer
      properties:
        microserviceName: WiseEyeChaosPortal
        clusterName: portal_live
        replicas:
          $jq: '.values.replicas'
        terminationGracePeriodSeconds: 30
        network:
          slbEnable: true
        grayStage:
          $jq: '.values.grayStage'
pipelines:
# 先变更灰度集群，再变更现网集群
  - name: chaos_gray_upgrade
    action: Serial
    tasks:
      - name: portal_grey
        action: Apply
        component:
          name: WiseEyeChaosPortal
          resources:
          - type: "WiseCloud::MicroService::NuwaContainer"
            name: "chaos-portal-grey"
      - name: portal_live
        action: Apply
        component:
          name: WiseEyeChaosPortal
          resources:
          - type: "WiseCloud::MicroService::NuwaContainer"
            name: "chaos-portal-live"
```

3.4.2 在 IaC 代码中声明资源

定义 component

定义component是IaC将一个环境的资源组织起来的方式，我们可以把同一类资源组织起来成为一个component。所有被IaC定义的资源必须属于一个component。在IaC3.0的代码中，component体现为global文件夹下的目录，在以下样例中，IaC代码定义了WiseEyeChaosMonkeyMgrService和WiseEyeChaosMonkeyPortal两个component。

```
├── global/
    │   ├── meta.yaml
    │   ├── WiseEyeChaosMonkeyMgrService/...
    │   └── WiseEyeChaosMonkeyPortal/...
```

定义资源

一个component下可以定义多个资源，所有的资源描述都存放于resources.yaml中，在以下样例中，WiseEyeChaosMonkeyMgrService下的resources.yaml中定义了该组件下的全量资源。

```
├── global/
│   │   ├── meta.yaml
│   │   ├── WiseEyeChaosMonkeyMgrService/
│   │       ├── config/
│   │       │   ├── business_config.yaml
│   │       │   └── config_schema.yaml
│   │       ├── resources.yaml
```

在resources.yaml中定义资源，资源的type和name构成资源的唯一标记，以列表的形式存在。

```
- name: chaos-mgr-p1                                    # 微服务平台显示的资源名称，最大长度为16字符，如果超过部署会报错
  type: WiseCloud::Microservice::NuwaContainer                  # 类型，NUWA容器集群的IaC代码固定为此值
  properties: ...
- name: chaos-mgr-p2                                    # 微服务平台显示的资源名称，最大长度为16字符
  type: WiseCloud::Microservice::NuwaContainer                  # 类型，NUWA容器集群的IaC代码固定为此值
  properties:
    ...
```

表 3-9 resources.yaml 字段说明

字段	含义
name	微服务平台显示的资源名称，最大长度为16字符。
type	资源类型。支持配置管理、Cloud Map、NUWA Container、GaussDB(for MySQL)及SLB。
properties	属性值，包含资源的详细参数。详细参数介绍请参见**资源介绍**。

$ref 语法

如果资源文件过大可以通过引用的方式对文件进行拆分及复用。

- yaml文档整体引用

  resources.yaml原始内容：

  ```
  # resources.yaml
  - name: virtualapp
    type: Wisecloud::Nuwa::Runtime::MicroService
    properties:
      configs:
        - name: common.java.http.ssl_protocols
          value: TLSv1.2
          sensitive: false
        - name: common.java.http.enabled_cipher_suites
          value: aes_128_gcm|aes_256_gcm
          sensitive: false
  ```

  可以将配置的定义（properties > configs 下的内容）放置在config/business_config.yaml文件下，通过$ref的方式来引用，则resources.yaml可修改为：

```
# resources.yaml
- name: virtualapp
  type: Wisecloud::Nuwa::Runtime::MicroService
  properties:
    configs:
      $ref: 'config/business_config.yaml#' # 运行时，会将config/business_config.yaml的整个YAML文档对象，替换掉整个$ref键值对
```

- yaml文档片段引用

  如果values.yaml存在以下内容：

  ```
  values:
    weixin:
      peak_tps: 200000
  ```

  则在resources.yaml，可按照如下方式进行引用200000。

  ```
  $ref: 'values.yaml#/values/weixin/peak_tps'
  ```

- 文本内容字符串引用

  ```
  $ref: 'db/schema.sql'    # 引用的是schema.sql这个文本文件内容所构成的字符串
  ```

3.5 资源介绍

3.5.1 NUWA Container

3.5.1.1 参数配置说明

3.5.1.1.1 基础参数

表 3-10 基础参数

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
name	string	必选	-	IaC3.0资源名	只能包括数字字母、'-'，'_'，'.'，必须以字母开头，字母或数字结尾。长度：2-64 **说明** 如果没有配置clusterName，则资源名会被当做微服务集群名，参数规范则会以微服务集群名的为准。	WiseCloudFGCEventBuilderService_cluster1

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
type	string	必选	-	BaaS服务类型，operator要求必填，固定为WiseCloud::MicroService::NuwaContainer	固定为WiseCloud::MicroService::NuwaContainer	WiseCloud::MicroService::NuwaContainer
microserviceName	string	必选	-	微服务名称	微服务名称	WiseCloudFGCEventBuilderService
clusterName	string	非必选（建议填写）	默认与name值相同	微服务集群名	只能包括数字字母、'-'，'_'，'.'，必须以字母开头，字母或数字结尾。长度：2-56 **说明** 不填与name值相同，由于两字段限制不同，超过限制会报错。	cluster1
replicas	int	必选	-	Pod副本数	整数类型 **说明** • 多AZ要配置AZ的倍数，如果部署了双AZ，那么此处要配置为2的倍数。 • 如果使用了evs盘或者elb，为保证滚动升级每个AZ至少保留一个节点，那么单AZ至少要配置为2，双AZ至少配置为4。	1
pdbMaxUnavailable	string	非必选	-	Pod干扰预算	整数百分比，整数范围为[1, 50]	-

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
terminationGracePeriodSeconds	integer	非必选	-	优雅下线宽限期	1-65535	-

示例：

```
- name: WiseCloudFGCEventBuilderService_cluster1
  type: WiseCloud::MicroService::NuwaContainer
  properties:
    microserviceName: WiseEyeChaosMonkeyExecutor
    clusterName: cluster1
    replicas: 5
```

3.5.1.1.2 挂载信息

表 3-11 挂载信息

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
name	string	必选	-	• log_volume：日志卷 • data_volume：data卷 • sfs_volume：sfs卷 **说明** sfs的卷命名并非固定，此处的名称主要还是供container的volumeMounts引用。	• log_volume 每个POD的日志卷大小，log_volume_type不配置或者配置为local时，代表使用集群节点的本地盘，有如下注意事项：容器场景下对/opt/huawei/logs的日志卷大小限制为100G，当日志磁盘写满之后，k8s会驱逐该POD并重新拉起一个新的POD，在此过程中，会影响业务。请在log配置文件中配置绕接策略对总的日志大小不超过此限制，由于NUWA及中间件也会记录一部分日志，建议业务配置保存的log总大小不要超过90G。另外业务需要将所有的日志配置到AIOps sidecar日志服务中，以通过AIOps日志服务进行日志采集，否则POD销毁时会造成日志丢失。 如果log_volume_type配置为evs，则根据实际的大小进行配置。 • data_volume data卷大小，默认挂载了/opt/huawei/data的路径，最大支持配置为5Gi。data_volume_type配置为evs则不受此限制。	volumes: - name: log_volume size: 12Gi type: local - name: data_volume size: 2Gi type: local
size	string	必选	-	定义存储空间		

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
type	string	可选	默认是local，代表使用集群节点本地存储。	• evs • sfs 参见右侧样例	sfs: volumes: - name: sfs_volume id: 11a701c9-529f-4992-9291-bc47bbf5b4c5 # 如果是已有的存储，该值代表存储的资源id type: sfs shareLocation: xxx # 共享路径 mountOptions: ["vers=3","hard","nolock"]	local

示例：

```
- name: WiseCloudFGCEventBuilderService_cluster1
  type: WiseCloud::MicroService::NuwaContainer
  properties:
    volumes:
      - name: log_volume
        size: 12Gi
        type: local
      - name: data_volume
        size: 2Gi
        type: local
```

3.5.1.1.3 容器配置

表 3-12 容器配置

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
image	string	必选	-	镜像地址 Runtime已预置如下镜像仓库地址，如果业务是从如下镜像仓库地址下载镜像，则只需要从组织名开始填写。 北京四：swr.cn-north-4.myhuaweicloud.com 乌兰察布一：swr.cn-north-9.myhuaweicloud.com 华南广州：swr.cn-south-1.myhuaweicloud.com	长度：1-256 **说明** 镜像仓库（:前的一串）只能是小写。	-

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
ports	int[]	可选	0	如果使用Cloud Map，将此端口填写为在Cloud Map注册中心注册的端口 如果不使用Cloud Map，可将此项注释掉或者填写为0。 **说明** 此参数只是起一个标识作用，并不是代表配置了此端口就代表端口一定打开。实际打开的端口以业务使用的为准，目前仅支持一个端口配置，数组类型是为后面可扩展。	端口范围：0-65535	ports: - 8080
flavor	string	-	"2C4Gi"	主容器CPU&memory的规格	长度范围：1-32。需要满足CPU和memory的规格，比如1C2G，2C4G，4C8G，8C16G等。	flavor: 2C4G

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
armFlavor	string	可选	"2C4Gi"	参考flavor 双AZ arm混部时独立控制arm配置 arm配置建议: 按照业内通用的指导，arm cpu算力相对于x86下降，具体下降指标和使用场景密切相关。 对于计算密集型的业务，可以考虑增加配置，IO密集型可以同规格，具体性能还是以各自业务的实际性能测试为准。	同上	同上
gpu	int	可选	null	主容器使用的GPU规格（GPU即显卡）；当前ERS管理的资源池中尚未提供GPU，所以此配置项在使用前请先和runtime开发人员联系	取值为整数，配1代表占用一块显卡，显卡不可分割。	-
stsEnable	bool	可选	TRUE	是否启用sts	-	TRUE
commandArgs	list(string)	可选	[]	启动参数	长度为0-256	-

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
volumeMounts	list(object)	可选	[]	此参数与volumes下的sfs挂载卷搭配使用，volumes中定义挂载卷的信息，此处引用并挂载在容器中运行	name要在volumes中有对应的值。	volumeMounts: - name: sfs_volume1 mountPath: /opt/huawei/sfs1 - name: sfs_volume2 mountPath: /opt/huawei/sfs2
envs	type = list(object({ name = string value = string }))	可选	[]	配置环境变量	name和value的长度为不超过5000。	envs: - name: "EVS_TEST" value: "test_ENV"
hostAliases	type = list(object({ hostName = string ip = string }))	可选	[]	配置hosts	-	hostAliases: - hostName: "a.b.com" ip:"1:1:1:1:1:1:1:1" - hostName: "c.d.com" ip:"2.2.2.2"

示例1（双AZ同构，或者不同构但同配置）：

```
- name: WiseCloudFGCEventBuilderService_cluster1
  type: WiseCloud::MicroService::NuwaContainer
  properties:
    containers:
      - image: swr.cn-north-4.myhuaweicloud.com/wiseeye/wiseeyechaosmonkeyservicewebsite:3.0.9.204.SP2
        flavor: 2C8G
        stsEnable: true
        ports:
          - 8080
```

示例2（arm x86混部时，独立配置flavor）：

```
- name: WiseCloudFGCEventBuilderService_cluster1
  type: WiseCloud::MicroService::NuwaContainer
  properties:
    containers:
```



```
- image: swr.cn-north-4.myhuaweicloud.com/wiseeye/wiseeyechaosmonkeyservicewebsite:3.0.9.204.SP2
  flavor: 2C8G
  armFlavor: 4C16G
  stsEnable: true
  ports:
    - 8080
```

3.5.1.1.4 容器健康检查

表 3-13 容器健康检查

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
readinessProbe: exec: command:	list(string)	必须配置其中一种，且只能配置一种	[]	命令行检查方式	命令行检查方式\http请求检查方式\tcp端口检查三种方式只能选择一种。	readinessProbe: exec: command: ["echo", "hello"]
readinessProbe: httpGet:	object		-	http请求检查方式	命令行检查方式\http请求检查方式\tcp端口检查三种方式只能选择一种。	readinessProbe: httpGet: path: /health port: 8080 scheme: HTTP
readinessProbe: tcpSocket: port:	int		-	tcp端口检查	命令行检查方式\http请求检查方式\tcp端口检查三种方式只能选择一种。	readinessProbe: tcpSocket: port:8080
livenessProbe: exec: command:	list(string)	必须配置其中一种，且只能配置一种	[]	命令行检查方式	命令行检查方式\http请求检查方式\tcp端口检查三种方式只能选择一种。	livenessProbe: exec: command: ["echo", "hello"]
livenessProbe: httpGet:	object		-	http请求检查方式	命令行检查方式\http请求检查方式\tcp端口检查三种方式只能选择一种。	livenessProbe: httpGet: path: /health.html port: 8080 scheme: HTTP

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
livenessProbe: tcpSocket: port:	int		-	tcp端口检查	命令行检查方式\http请求检查方式\tcp端口检查三种方式只能选择一种。	livenessProbe: tcpSocket: port:8080
initialDelaySeconds	int	可选	10s	表示容器启动多少秒之后开始探测，单位秒。默认值为10s。	不小于5	readinessProbe: exec: command: ["echo", "hello"] initialDelaySeconds: 10 periodSeconds: 20 successThreshold:1 failureThreshold: 3 timeoutSeconds: 5
periodSeconds	int	可选	20s	间隔周期，表示每多少秒探测一次容器，单位秒，默认值为20s。	不大于180	
successThreshold	int	可选	1	表示连续检测多少次成功后则记作成功。默认值为1。	不大于10，liveness探针只能为1。	
failureThreshold	int	可选	3	表示连续检测多少次失败当做是失败处理，并会重启容器。默认值为3。	不大于10	

示例：


```
livenessProbe:
  httpGet:
    path: /health
    port: 8080
    scheme: HTTP
  initialDelaySeconds: 20
  timeoutSeconds: 3
  periodSeconds: 10
  successThreshold: 1
  failureThreshold: 3
readinessProbe:
  httpGet:
    path: /health
    port: 8080
    scheme: HTTP
  initialDelaySeconds: 20
  timeoutSeconds: 3
  periodSeconds: 10
  successThreshold: 1
  failureThreshold: 10
      - name: APIGateway
        type: WiseCloud::Agent::APIGateway
        version: x.x.x.x
        flavor: 1C2G
        param:
          $ref: 'config/sidecar_apigw_param.json
```

3.5.1.1.5 sidecar 配置

表 3-14 sidecar 配置

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
sidecars	type = list(object)	-	[]	由于POD销毁或被驱逐时，日志文件会丢失，因此AIOps Log Sidecar为必选参数，需要业务配置此sidecar及时将日志进行采集。	• name：sidecar的名称，比如AIOps Log。 • type：sidecar的类型，详见代码示例。 • version：sidecar的版本。 • cpu：sidecar容器的cpu，配置举例：0.5 或者 2 或者 500m，配置为null或者""“时，代表使用系统默认值。 • memory：sidecar的内存，单位为M Mi G	详见代码示例，按照业务需求配置所需的sidecar。

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
					Gi，配置举例：100M 0.5G 2G，配置为null或者”"时，代表使用系统默认值。注意M和Mi的区别，M是1000的倍比，Mi是1024的倍比。 • param：sidecar的配置参数，不同sidecar的配置参数不一样，具体可以参考各个sidecar的配置方式。	

示例：

```
sidecars:
    - name: AIOpsLog
      type: WiseCloud::Agent::AIOpsLog
```



```
    version: 3.3.0.100
    flavor: 0.4C500M
    param:
      $ref: 'config/sidecar_aiops_param.json'
  - name: RASP
    type: WiseCloud::Agent::RASP
    version: 2.2.0.102.SP5
    flavor: 0.3C500M
    param: ''
  - name: BIFlume
    type: WiseCloud::Agent::BIFlume
    version: x.x.x.x
    flavor: 0.2C500M
    param:
      $ref: 'config/sidecar_biflume_param.json'
  - name: APIGateway
    type: WiseCloud::Agent::APIGateway
    version: x.x.x.x
    flavor: 1C2G
    param:
      $ref: 'config/sidecar_apigw_param.json
```

3.5.1.1.6 网络配置

表 3-15 网络配置

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
isolatedDomain	string	可选	-	隔离域	只有CCE_TURBO集群支持，老CCE集群不支持。如果不配置，则使用ENS隔离域规划中对应的隔离域，见上面隔离域说明链接中的说明。	-

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
slbEnable	bool	可选	FALSE	是否启用slb	“否”仅针对第一次添加SLB信息时有效 **说明** 如果之前有部署过，并且配置为“是”，此时再修改为“否”是无效的。	-

示例：

```
- name: WiseCloudFGCEventBuilderService_cluster1
  type: WiseCloud::MicroService::NuwaContainer
  properties:
    network:
      slbEnable: true
      isolatedDomain: $ {创建的隔离域名称}
```

3.5.1.1.7 SLB 配置（可选）

表 3-16 SLB 配置

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
port	int32	必选	null	微服务在SLB上暴露的端口	-	8080
weight	int32	必选	null	微服务在SLB上负载的权重	-	20
timeout	int32	必选	null	微服务调用转发的超时时间	-	20
maxFails	int32	必选	null	微服务调用的失败次数	-	3

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例

targetGroup	object{
	name,
	loadBalancer
	}
		必选	""	微服务注册到SLB上的后端服务器组名	• 用于创建RouteRule，其中的name为必选，对应NuwaRuntime的backendClusterName，其它为可选。
					• 在routes配置的情况下，loadBalancer也属于必选，不配置routes的话，loadBalancer则是可选。
					• loadBalancer当前仅支持配置roundRobin（加权轮询）且不支持修改为其它策略，如需修
						name: chaosmonekey_portal_static
						loadBalancer:
						strategy: roundRobin

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
					改可登录SLB管理台修改。	
listenerGroupName	string	必选	""	微服务前端关联的SLB ListenserGroup实例IaC名称	对应NuwaRuntime中的slbServiceName，长度为[1, 64] SLB下需要存在这个监听，为空时不会创建RouteRule。	chaos_slb_listener_lhq
routes	array[location: string]	可选	[]	微服务的路由规则	不推荐使用，请直接使用slb的iac3.0创建。	/fgc/v1

示例：

```
- name: WiseCloudFGCEventBuilderService_cluster1
  type: WiseCloud::MicroService::NuwaContainer
  properties:
    bindSlb:
      - port: 8080
        grayStatus: 1
        weight: 20
        maxFails: 3
        timeout: 20
        listenerGroupName: chaos_slb_listener_lhq
        targetGroup:
          name: chaosmonekey_portal_static
          loadBalancer:
            strategy: roundRobin
```

3.5.1.1.8 证书配置

表 3-17 证书配置

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
certConfigs	type = list(object({ name = string tag = string }))	可选	[]	证书配置	• name：证书名称 • tag：证书tag	# 业务证书配置 certConfigs: name: "server" # 证书名称 tag: "default" # 证书tag

3.5.1.1.9 daemonSet

表 3-18 daemonSet

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
daemonSet	type = list(object)	可选	[]	目前仅支持AIOps DaemonSet和BI DaemonSet，挂载hostpath提供存储持久化到node的能力。	• name：AIOps or BI • type：DaemonSet的类型，详见代码示例 • enable：true|false，是否启用 • logPath：推送日志路径（非文件名），选填。 • AIOps固定为'/opt/huawei/logs' • limitSize：日志存储限制，选填，默认值为100G。仅AIOps生效。 • groups：日	详见代码示例

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
					志配置组名，必填。当前BI场景支持1个或多个分组，AIOps场景有且只能填写一个分组并且提前在AIOps管理面创建好。	

示例：

```
daemonSet:
    - name: AIOps
      type: WiseCloud::Agent::AIOps
      enable: true
      logPath: '/opt/huawei/logs'
      limitSize: 100G
      groups: ["logConfigGroupName"]
    - name: BI
      type: WiseCloud::Agent::BI
      enable: false
      logPath: '/opt/huawei/logs/bi'
      groups: ["ODS_V001_DM_service1", "ODS_V001_DM_service2"]
      paramJson: "{\"dataGroups\":[{\"dataGroup\":\"ODSName_BatchFileExampleDS\",\"agentType
\":\"batch\",\"batchConfig\":{\"datapushInputs\":{\"jobType\":\"file\",\"dayPeriod\":{\"startTime
\":\"10:00:00\",\"offset\":\"1\"},\"file\":{\"sources\":[{\"pattern\":\"/opt/huawei/hcy/*.txt\",\"filename
\":\"test.txt\"}]}},\"datapushOutput\":{\"postfix\":\"txt\",\"permitEmptyFile\":true},\"advanced\":
{\"extendFields\":{\"datapushInput.isUtc\":false,\"datapushInput.file.sourcePolicy
\":3,\"datapushInput.file.countThreshold\":0,\"datapushInput.file.sizeThreshold\":0,\"sendThreadCount
\":3,\"datapushInput.file.fileRetryTimes\":3,\"datapushInput.file.fileWaitTimes\":3}}}},{\"dataGroup
\":\"ODSName_StreamFileExampleDS\",\"agentType\":\"stream\",\"streamConfig\":{\"filebeatInputs\":{\"type
\":\"log\",\"enabled\":true,\"paths\":[\"/opt/huawei/logs/*.log\",\"/opt/huawei/logs/*.txt\"]},\"advanced\":
{\"extendFields\":{\"filebeatInputs.harvester_limit\":5,\"queue.mem.events
\":4096,\"queue.mem.flush.min_events\":2048}}}}]}"
      # 配置json格式化
```

3.5.1.1.10 业务配置

表 3-19 业务配置

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
configs	object(private = object(name = string prefix = string version = string schema = object records = object) public = object(name = string prefix = string))	否	{}	配置项的根字段，包含两个属性，分别是private和public，分别为业务配置项和公共配置项，其下各个字段的描述如下所示。	-	见下文样例
prefix	string	否	""	配置项的归属路径	仅限于publicConfig	/public/cloudeye/wiseEyeConfigService

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
schema	object(type = string properties = object(key1 = object(type = string description = string format = string)))	否	{}	配置项属性，properties属性为key-object格式，key是配置名称，object是配置项各项描述，其中format指配置项类型，默认为notype，如果是敏感配置项为sensitive。 **说明** 如果是非敏感配置项，可以不在schema中声明，以减少维护工作量。	仅限于privateConfig	见下文样例

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
records	object(key1 = string key2 = string)	否	{}	描述配置，key-value格式，key为配置名称，value为配置值。 value仅支持字符串类型 如果是数字、布尔值、对象和数组，需要加单引号，例：'10'、'true'、'{"test": 1}'、'[1,2]'。	仅限于privateConfig	records: test: '{"a":"a", "b":"b"}' timeout: '10' enableSa: 'true'
name	string	否	""	配置项名称，对应NuwaRuntime的Container[0].configTag，对应PublicConfig和PrivateConfig的name。 **说明** 配合动态配置生效，需要nuwa基础镜像版本要保持在3.0.11版本以上，否则报错。	publicConfig/privateConfig均有 限制：8位以内的小写字母和数字	见下文样例

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
version	string	否	""	配置版本，对应NuwaRuntime的Container[0].configVeersion，对应PrivateConfig的version。 **说明** 配合动态配置生效，需要nuwa基础镜像版本要保持在3.0.11版本以上，否则报错。	仅限于privateConfig	见下文样例

示例：

```
# resources.yaml
- name: WiseCloudFGCEventBuilderService_cluster1
  type: WiseCloud::MicroService::NuwaContainer
  properties:
    configs:
      $ref: 'config/business_config.yaml#'
# business_config.yaml
public:
  prefix: /com.huawei.wiseeye
  name: public5
private:
  version: 1.0.0      #（配合动态配置生效，需要nuwa基础镜像版本要保持在3.0.11版本以上，否则报错）
  name: fgcva         #（配合动态配置生效，需要nuwa基础镜像版本要保持在3.0.11版本以上，否则报错）
  schema:
    type: object
    properties:
      AIOPSCONFIG_AIOPSSERVER:      # 默认为format: notype，如果非敏感项，可以不填
        format: notype              # 默认为notype，如果非敏感项，可以不填
      AIOPSCONFIG_AIOPSTOKENURI:    # 敏感项必须填
        format: sensitive           # 敏感项必须填
  records:                          # 必填
    AIOPSCONFIG_AIOPSSERVER: https://XX.XX.XX.XX:XXXX/
    test: '{"a":"a","b":"b"}'   # 仅支持字符串类型，[]、{}的值，yaml会识别为对象和数组，必须加单引号
    timeout: '10'               # 仅支持字符串类型，数字和布尔值也要加引号
    enableSa: 'true'
```

3.5.1.1.11 滚动升级策略

表 3-20 滚动升级策略

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
rollingUpdateStrategy	type = object({ maxSurge = string maxUnavailable = string })	可选	maxSurge : "25%" maxUnavailable: 0	滚动升级策略配置 • maxSurge：滚动升级时最多可以多启动多少个pod。 • maxUnavailable：滚动升级时最大可以删除多少个pod。	整数：最小值为0 百分比：0% ~ 100% **说明** • maxSurge和maxUnavailable不能同时为0 • maxUnavailable不能设置为100或者100%，避免升级时集群没有可用的pod。	rollingUpdateStrategy: maxSurge = "50%" maxUnavailable = "25%"

3.5.1.1.12 优雅下线

表 3-21 优雅下线

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
preStopConfig	type = object({ execCommand = list })	必选	[]	优雅退出处理	-	preStopConfig: execCommand: ["/bin/bash", "-c", "sleep 20"]
terminationGracePeriodSeconds	int	可选	30	优雅退出宽限时间，此时间为整个POD的最大退出时间。	-	terminationGracePeriodSeconds: 30

3.5.1.1.13 灰度策略

表 3-22 灰度策略

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
grayStage	type = object({grayInstances = number grayProcess= stringgrayStatus = number })	可选	null	灰度升级策略配置。 • grayInstances：灰度升级的实例数比例，范围为1~100。 • grayStatus：灰度实例对接SLB时的状态，1：生产，2：灰度。 • grayProcess：灰度状态，只能填"INGRAY"和"FINISHED"。	-	grayStage: grayInstances: 50 grayProcess: "INGRAY" grayStatus: 2

示例：

```
- name: WiseCloudFGCEventBuilderService
  type: WiseCloud::MicroService::NuwaContainer
  properties:
    grayStage:
      grayInstances: 50
      grayProcess: "INGRAY"
      grayStatus: 2
```

3.5.1.1.14 水平自动伸缩（HPA）

如果业务不需要使用hpa，请不要配置hpa相关参数。hpa开关配置关闭，也会创建hpa资源，只是不会生效扩缩容。

表 3-23 水平自动伸缩（HPA）

参数	说明	是否必填	备注
hpa_scale_disabled	是否禁用hpa，false表示不禁用；true表示禁用。	是	取值为false是开启hpa。
polling_interval	负载检测周期，单位秒。	否	-
origin_instances	初始副本数，不配置则使用min_instances值。	否	-
min_instances	最小副本数	是	不配置默认为1。
max_instances	最大副本数	是	不配置默认为1。
hpa_scale_triggers	扩缩容指标配置，目前仅支持cpu/memory。	是	仅统计主容器资源。
hpa_scale_up_rules	扩容规则，定义稳定时间窗，减少扩容毛刺。	否	-
hpa_scale_down_rules	缩容规则，定义稳定时间窗，减少缩容毛刺。	否	-
hpa_scale_up_policies	扩容策略，定义扩容步长。	否	-
hpa_scale_down_policies	缩容策略，定义缩容步长。	否	-

示例：

在resources.yaml中添加hpa参数如下：

```
hpa:    #弹性伸缩配置
  $ref: 'config/hpa.yaml#/recommend'
```

在config目录，增加一个hpa.yaml文件，存放hpa的相关配置项。

```
recommend:
  disabled: true       #true表示关闭hpa，false表示开启hpa
  pollingInterval: 5   #负载检测周期，单位秒
  minReplicas: 2       #最小副本数
  maxReplicas: 4       #最大副本数
  triggers:            #业务根据时间情况选择弹性伸缩策略
    -type: CPU         #业务容器的CPU利用率大于40%则触发扩容条件
     metadata:
        averageUtilization: 40%
    -type: Memory                  #业务容器的内存利用率大于60%则触发扩容条件
     metadata:
        averageUtilization: 60%
```


3.5.1.1.15 指定分组和资源标签

表 3-24 指定分组和资源标签

参数名称	参数类型	是否必选	默认值	说明	参数规范	举例
resourceTag	type = object ({group = stringfeatures = string})	可选	无	• group：指定集群部署的分组。 • features：指定集群部署的标签信息。	此处配置的信息要再Runtime页面提前预置好。	resourceTag: group: "common" features: "dev"

3.5.1.2 配置 demo

```
# resources.yaml
- name: sdkCluster
  type: WiseCloud::MicroService::NuwaContainer
  properties:
    microserviceName: WiseCloudNuwaSDK    # STS证书中服务的名称
    replicas:
      $ref: 'config/values.yaml#/values/replicas'
    terminationGracePeriodSeconds: 30
    network:
      slbEnable:
        $ref: 'config/values.yaml#/values/slbEnable'
      isolatedDomain: XXX
    volumes:
      - name: log_volume
        size: 10Gi
        type: local
      - name: data_volume
        size: 2Gi
        type: local
    containers:                          # 微服务基于容器化的部署
      - image:
          $ref: 'config/values.yaml#/values/image'
        flavor: 2C4G
        stsEnable: true
        ports:
          - 8080
        livenessProbe:
          httpGet:
            path: /health.html
            port: 8080
            scheme: HTTP
          initialDelaySeconds: 20
          timeoutSeconds: 3
          periodSeconds: 10
          successThreshold: 1
          failureThreshold: 3
        readinessProbe:
          httpGet:
            path: /health.html
            port: 8080
            scheme: HTTP
```

```
        initialDelaySeconds: 20
        timeoutSeconds: 3
        periodSeconds: 10
        successThreshold: 1
        failureThreshold: 10
      preStopConfig:
        execCommand: ["/bin/bash", "-c", "sleep 20"]
      envs:
        $ref: 'config/envs.yaml#'
      hostAliases:
        - hostName: "a.b.com"
          ip: "1:1:1:1:1:1:1:1"
        - hostName: "c.d.com"
          ip: "2.2.2.2"
  certConfigs:                    # 可选
        - name: console
          tag: default
  configs:
    $ref: 'config/business_config.yaml#'
  daemonSet:
    - name: AIOps
      enable: true
      logPath: '/opt/huawei/logs'
      limitSize: 100G
      groups: ["logConfigGroupName"]
    - name: BI
      enable: false
      logPath: '/opt/huawei/logs/bi'
      groups: ["ODS_V001_DM_service1", "ODS_V001_DM_service2"]
  sidecars:
    - name: AIOpsLog
      version: 1.11.3.100
      flavor: 0.4C500M
      param:
        $ref: 'config/sidecar_aiops_param.json'
     - name: RASP
      version: 2.2.0.102.SP5
      flavor: 0.3C500M
  bindSlb:
    - port: 8080
      grayStatus: 1
      weight: 20
      maxFails: 3
      timeout: 20
      instanceName: chaos_slb_listener_lhq
      listenerGroupName: chaos_slb_listener_lhq
      targetGroup:
        name: chaosmonekey_portal_static
```

```
# business_config.yaml
public:
  prefix: /com.huawei.wiseeye
  name: public5
private:
  schema:
    type: object
    properties:
      AIOPSCONFIG_AIOPSSERVER:
        format: notype
      AIOPSCONFIG_AIOPSTOKENURI:
        format: sensitive
  records:
    AIOPSCONFIG_AIOPSSERVER: 'https://XX.XX.XX.XX:XXXX/'
    AIOPSCONFIG_AIOPSTOKENURI: openapi/XXX/XXX/
```

3.5.1.3 错误码说明

表 3-25 错误码说明

错误码	说明
NotFound	实例不存在
NuwaRuntime.Microservice.CreateError	实例创建失败
NuwaRuntime.Microservice.ReadError	实例读取失败
NuwaRuntime.Microservice.DeleteError	实例删除错误
NuwaRuntime.Microservice.UpdateError	实例更新错误
NuwaRuntime.Microservice.UnknownError	未知异常

3.6 部署调测

开发完成IaC代码之后，可以通过环境管理的页面进行环境的创建和变更操作，具体请参见**环境管理概述**。

如果代码在本地调测无问题，可以提交到代码仓，请参考**变更环境**。

容器部署的详情，可在微服务平台（NUWA）查看POD的部署情况，具体请参见**实例管理**。

4 打包规范

4.1 软件包

软件包一般用于虚拟机部署使用，其中包括有软件包（虚拟机部署使用），测试用例包，函数包（函数部署使用）。

文件名

- 文件名后缀只支持zip。
- 文件名只允许包含英文、数字、“-”、“_”、“（）”、“.”、空格，最大长度不超过200。

  说明

  以上符号是英文符号，不支持中文符号。文件名不合规时，会导致发布电子流失败，并且只能重走电子流。

zip 包大小限制

组合包解压后不超过50G，单个子包解压前不超过30G，解压后不能超过50G（可配置），超出限制将导致电子流发布失败。

文件结构

图 4-1 软件包结构

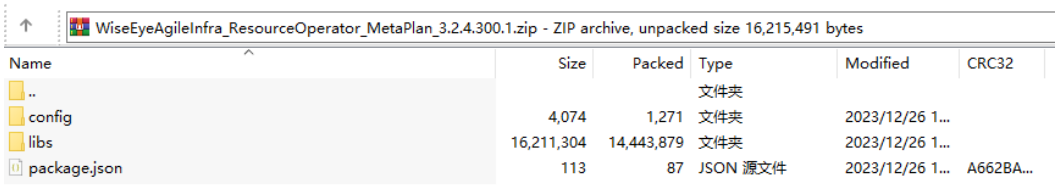
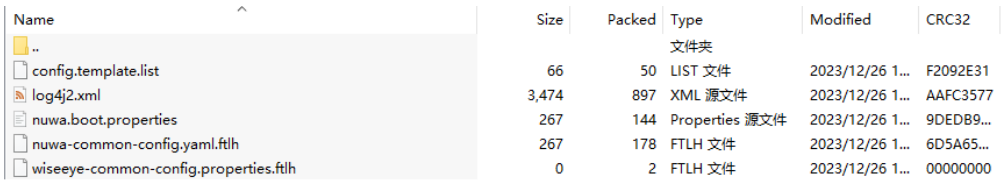


表 4-1 软件包结构说明

位置	类型	描述
config/	文件夹	配置文件所在的目录。
libs/	文件夹	打成的依赖包所在的目录。
package.json	文件	包描述文件。 **说明** • 无论是否使用自动部署，都必须包含package.json文件。 • package.json文件必须放在zip包的根目录中。

- config目录

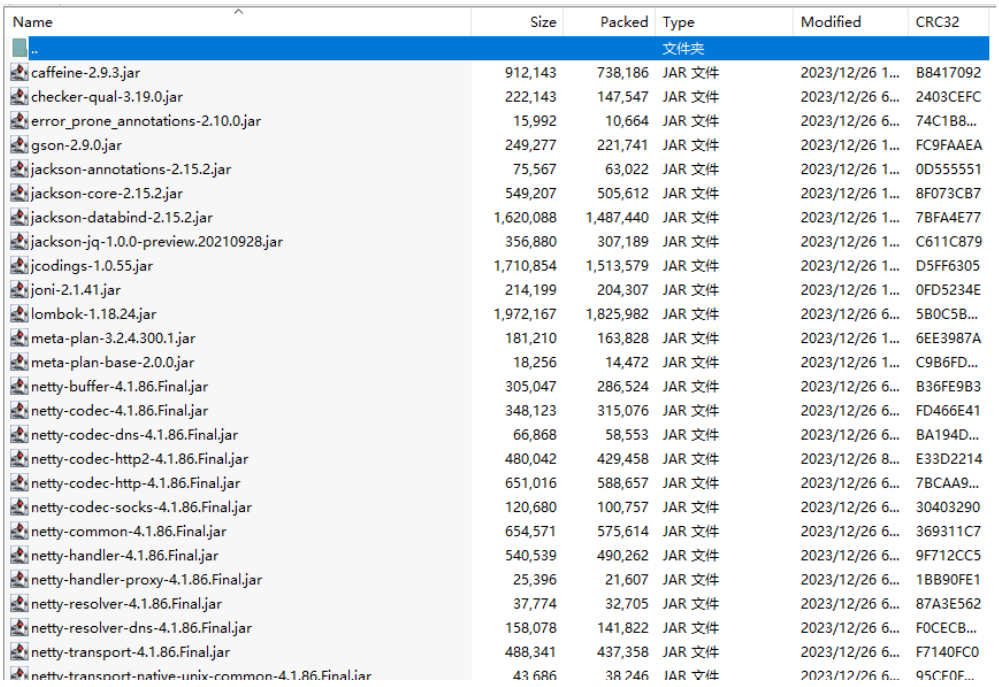
  图 4-2 config 目录

Name	Size	Packed	Type	Modified	CRC32
..			文件夹		
config.template.list	66	50	LIST 文件	2023/12/26 1...	F2092E31
log4j2.xml	3,474	897	XML 源文件	2023/12/26 1...	AAFC3577
nuwa.boot.properties	267	144	Properties 源文件	2023/12/26 1...	9DEDB9...
nuwa-common-config.yaml.ftlh	267	178	FTLH 文件	2023/12/26 1...	6D5A65...
wiseeye-common-config.properties.ftlh	0	2	FTLH 文件	2023/12/26 1...	00000000

- libs目录

  图 4-3 libs 目录

Name	Size	Packed	Type	Modified	CRC32
..			文件夹		
caffeine-2.9.3.jar	912,143	738,186	JAR 文件	2023/12/26 1...	B8417092
checker-qual-3.19.0.jar	222,143	147,547	JAR 文件	2023/12/26 6...	2403CEFC
error_prone_annotations-2.10.0.jar	15,992	10,664	JAR 文件	2023/12/26 6...	74C1B8...
gson-2.9.0.jar	249,277	221,741	JAR 文件	2023/12/26 1...	FC9FAAEA
jackson-annotations-2.15.2.jar	75,567	63,022	JAR 文件	2023/12/26 1...	0D555551
jackson-core-2.15.2.jar	549,207	505,612	JAR 文件	2023/12/26 1...	8F073CB7
jackson-databind-2.15.2.jar	1,620,088	1,487,440	JAR 文件	2023/12/26 1...	7BFA4E77
jackson-jq-1.0.0-preview.20210928.jar	356,880	307,189	JAR 文件	2023/12/26 1...	C611C879
jcodings-1.0.55.jar	1,710,854	1,513,579	JAR 文件	2023/12/26 1...	D5FF6305
joni-2.1.41.jar	214,199	204,307	JAR 文件	2023/12/26 1...	0FD5234E
lombok-1.18.24.jar	1,972,167	1,825,982	JAR 文件	2023/12/26 6...	5B0C5B...
meta-plan-3.2.4.300.1.jar	181,210	163,828	JAR 文件	2023/12/26 1...	6EE3987A
meta-plan-base-2.0.0.jar	18,256	14,472	JAR 文件	2023/12/26 1...	C9B6FD...
netty-buffer-4.1.86.Final.jar	305,047	286,524	JAR 文件	2023/12/26 6...	B36FE9B3
netty-codec-4.1.86.Final.jar	348,123	315,076	JAR 文件	2023/12/26 6...	FD466E41
netty-codec-dns-4.1.86.Final.jar	66,868	58,553	JAR 文件	2023/12/26 6...	BA194D...
netty-codec-http2-4.1.86.Final.jar	480,042	429,458	JAR 文件	2023/12/26 8...	E33D2214
netty-codec-http-4.1.86.Final.jar	651,016	588,657	JAR 文件	2023/12/26 6...	7BCAA9...
netty-codec-socks-4.1.86.Final.jar	120,680	100,757	JAR 文件	2023/12/26 6...	30403290
netty-common-4.1.86.Final.jar	654,571	575,614	JAR 文件	2023/12/26 6...	369311C7
netty-handler-4.1.86.Final.jar	540,539	490,262	JAR 文件	2023/12/26 6...	9F712CC5
netty-handler-proxy-4.1.86.Final.jar	25,396	21,607	JAR 文件	2023/12/26 6...	1BB90FE1
netty-resolver-4.1.86.Final.jar	37,774	32,705	JAR 文件	2023/12/26 6...	87A3E562
netty-resolver-dns-4.1.86.Final.jar	158,078	141,822	JAR 文件	2023/12/26 6...	F0CECB...
netty-transport-4.1.86.Final.jar	488,341	437,358	JAR 文件	2023/12/26 6...	F7140FC0
netty-transport-native-unix-common-4.1.86.Final.jar	43,686	38,246	JAR 文件	2023/12/26 6...	95CF0F...

- package.json

  软件包的package.json内容一般如下：

```
{
  "type": "software",                    //软件包类型标识，固定写法，不能随便填写，否则导致电子流异常
```

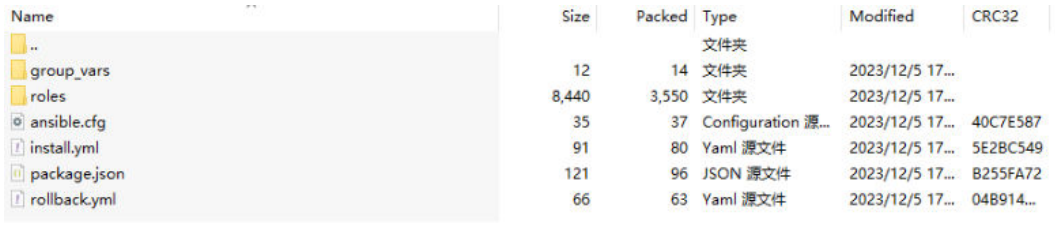


```
 "scope":"1180196813870297011",          //服务ID，在AppStage运维中心工作台右上角的个人账号信息管理中，选择“租户管理”，查看服务ID
 "name": "WiseEyeDeployReleaseCenter",    //可以自定义，建议使用微服务名称
 "version": "{{VERSION_ID}}",             //版本号，版本号如果重复，在后面加时间戳，不会覆盖原来的版本，支持英文、数字、“-”、“_”、“（）”、“.”、空格，包含其他字符会导致电子流异常
}
```

4.2 部署包

部署包就是ansible-playbook，作为业务虚拟机部署的脚本使用。

图 4-4 部署包结构



Name	Size	Packed	Type	Modified	CRC32
..			文件夹		
group_vars	12	14	文件夹	2023/12/5 17...	
roles	8,440	3,550	文件夹	2023/12/5 17...	
ansible.cfg	35	37	Configuration 源...	2023/12/5 17...	40C7E587
install.yml	91	80	Yaml 源文件	2023/12/5 17...	5E2BC549
package.json	121	96	JSON 源文件	2023/12/5 17...	B255FA72
rollback.yml	66	63	Yaml 源文件	2023/12/5 17...	04B914...

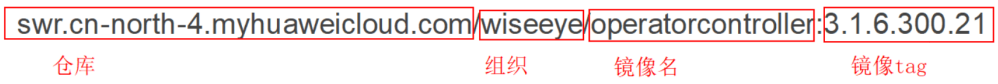
其中package.json里面的内容为playbook。

4.3 镜像包

镜像包，package.json里面的type为docker，一般是以docker镜像的形式存储在镜像仓中。也可以用docker的镜像包打出tar包的形式，同时在tar包同层级目录放置package.json内容，和tar包一起，再打出zip包格式。

容器镜像地址分为四个部分：镜像仓库、组织、镜像名、镜像tag，一个完整的容器镜像地址如图4-5所示。

图 4-5 镜像地址

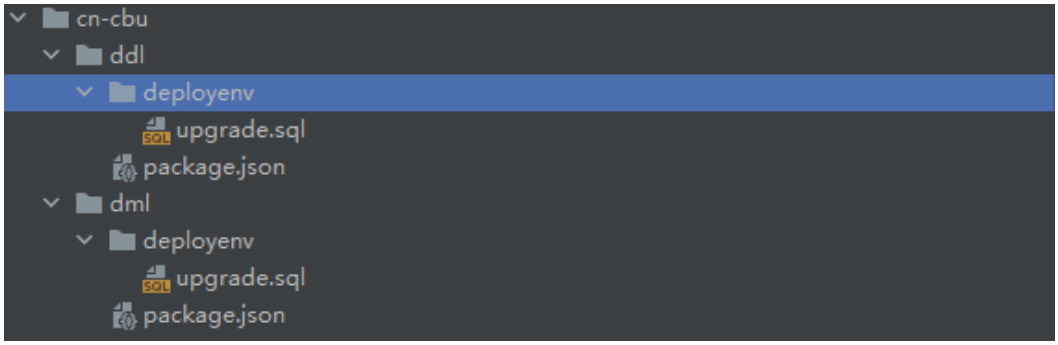


4.4 SQL 包

SQL包是指包含了通过WiseDBA执行的SQL脚本的zip包，分为DDL和DML两种类型。

包结构

图 4-6 SQL 包结构



- DDL的package.json如下所示，主要是写ddl语句。

```
{
  "name": "${service_name}-ddl-sqlchange-cn-cbu",//数据库包的包名，包括站点、业务、服务、实例类型、实例名和包名等信息
  "site_name": "cbu",                    //站点名，中国区为cbu，欧洲区为eu-cbu，亚非拉为aaa-cbu
  "business_name": "${business_name}",        //AppStage业务控制台中业务定义的产品英文名称，查看方式请参考产品管理
  "service_name": "${service_name}",          //AppStage业务控制台中业务定义的服务英文名称，查看方式请参考服务管理
  "instance_name": "${mysql_instance_cn_cbu}",  //WiseDBA中纳管的数据库实例名称
  "instance_type": "GaussDB4MySQL",           //数据库实例类型，支持GaussDB4MySQL/RDS4MySQL/GaussDB4Cassandra/GaussDB4OpenGauss，分别对应WiseDBA中的GaussDB(for MySQL)/RDS for MySQL/GeminiDB(for Cassandra)/GaussDB
  "type": "dbscript_ddl",                 //包类型，ddl语句固定为dbscript_ddl
  "version": "${package_version}"             //数据库包的版本，即包坐标中的version字段，例如：1.0.1
}
```

- DML的package.json如下所示，主要是写dml语句。

```
{
  "name": "${service_name}-dml-sqlchange-cn-cbu",//数据库包的包名，包括站点、业务、服务、实例类型、实例名和包名等信息
  "site_name": "cbu",                    //站点名，中国区为cbu，欧洲区为eu-cbu，亚非拉为aaa-cbu
  "business_name": "${business_name}",        //AppStage业务控制台中业务定义的产品英文名称，查看方式请参考产品管理
  "service_name": "${service_name}",          //AppStage业务控制台中业务定义的服务英文名称，查看方式请参考服务管理
  "instance_name": "${mysql_instance_cn_cbu}",  //WiseDBA中纳管的数据库实例名称
  "instance_type": "GaussDB4MySQL",           //数据库实例类型，支持GaussDB4MySQL/RDS4MySQL/GaussDB4Cassandra/GaussDB4OpenGauss，分别对应WiseDBA中的GaussDB(for MySQL)/RDS for MySQL/GaussDB(for Cassandra)/GaussDB
  "type": "dbscript_dml",                 //包类型，dml语句固定为dbscript_dml
  "version": "${package_version}"             //数据库包的版本，即包坐标中的version字段，例如：1.0.1
}
```

4.5 IaC 3.0 包

文件名

- 文件名后缀只支持zip。
- 文件名只允许包含英文、数字、“-”、“_”、“（）”、“.”、空格，最大长度不超过200。

说明

以上符号是英文符号，不支持中文符号。文件名不合规时，会导致发布电子流失败，并且只能重走电子流。

包结构

IaC3.0 包有IaC Spec和IaC Patch两种类型，具体包结构介绍请参见**了解代码仓结构**。

4.6 Terraform 包

Terraform包的规范请参考**Terraform**。

5 附录

5.1 使用 configparser 工具优化代码

configparser为自定义参数解析工具，通过NUWA部署时，解析参数模板，将模板中的参数变量，替换为实际的配置项值。

准备工作

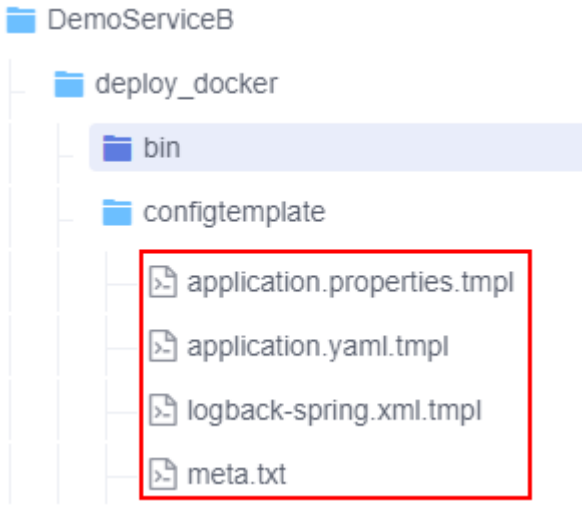
1. 下载configparser工具，在tools文件夹中获取configparser工具。
2. 将本地的configparser工具复制粘贴至项目的bin目录下。

使用 configparser 工具

步骤1 在Dockerfile文件中，为configparser文件添加可执行权限。

步骤2 定义业务配置项模板文件和meta.txt，样例如图1 业务配置项模板所示。

图 5-1 业务配置项模板



步骤3 在业务配置项模板文件（.tmpl文件）中，使用 {{参数名称}} 格式定义需要动态替换的参数，样例如图5-2所示。

图 5-2 模板参数定义

```
sts.server.domain={{sts.server.domain}}
sts.config.path={{sts.config.path}}
```

步骤4 在meta.txt文件中定义需要替换的业务配置项模板文件。如图5-3所示。

图 5-3 替换业务配置项模板文件

```
./application.properties.tmpl|../service/config/application.properties
./application.yaml.tmpl|../service/config/application.yaml
./logback-spring.xml.tmpl|../service/config/logback-spring.xml
```

- application.properties.tmpl为配置文件模板，application.properties为目标配置文件。
- 配置中指定的文件路径是相对于meta.txt文件的路径。

步骤5 启动业务进程之前，在启动脚本中调用configparser工具，进行参数替换。使用方式如下：

```
/opt/huawei/app/bin/configparser -meta /opt/huawei/app/configtemplate/meta.txt -log configparser.log -mode front -tempPath /opt/huawei/app/configtemplate/config-temp
```

- 使用绝对路径的方式调用configparser工具，/opt/huawei/app/bin/为容器启动时的绝对路径。
- -meta：指定meta.txt文件，/opt/huawei/app/configtemplate/为容器启动时meta.txt文件的绝对路径。
- -log：存放configparser工具的运行日志。
- -mode front：固定使用此参数值。
- -tempPath：工具运行过程中生成临时文件的路径。

  /opt/huawei/app/configtemplate/为容器启动时的绝对路径，必须保证此目录路径存在。config-temp文件夹可以不存在，会自动创建，运行结束后会清理此路径。 如果不配置，默认使用/opt/huawei/app/nuwa/config-temp。

----结束

6 修订记录

发布日期	修订记录
2024-04-02	第三次正式发布。 新增**使用configparser工具优化代码**章节。
2024-03-08	第二次正式发布。 新增**使用Spring Cloud框架实现应用开发**章节。 刷新**应用平台IaC部署代码开发**章节。 新增**部署包**章节。 新增**镜像包**章节。 新增**SQL包**章节。 新增**Terraform包**章节。
2023-11-25	第一次正式发布。