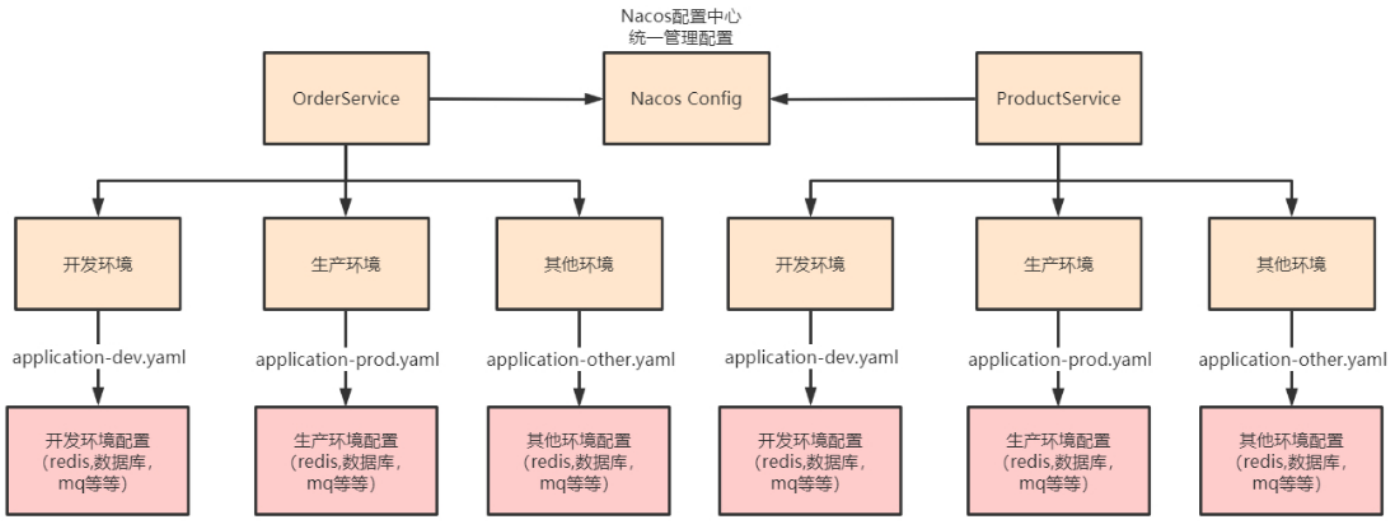
1. Nacos配置中心使用

官方文档： https://github.com/alibaba/spring-cloud-alibaba/wiki/Nacos-config

Nacos 提供用于存储配置和其他元数据的 key/value 存储，为分布式系统中的外部化配置提供服务器端和客户端支持。使用 Spring Cloud Alibaba Nacos Config，您可以在 Nacos Server 集中管理你 Spring Cloud 应用的外部属性配置。

1.维护性 2.时效性 3.安全性



springcloud config 对比

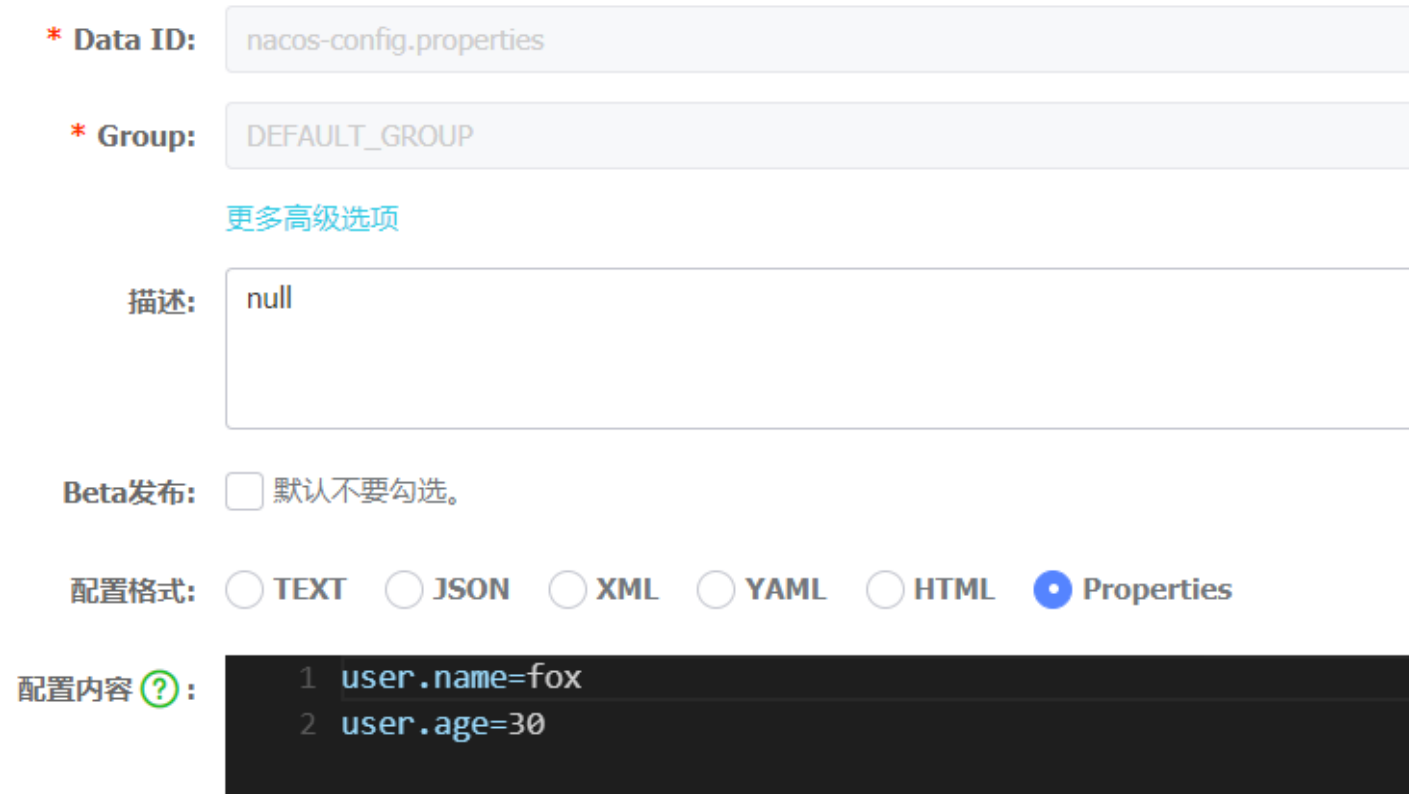
三大优势：

- springcloud config大部分场景结合git 使用, 动态变更还需要依赖Spring Cloud Bus 消息总线来通过所有的客户端变化.
- springcloud config不提供可视化界面
- nacos config使用长轮询更新配置, 一旦配置有变动后，通知Provider的过程非常的迅速, 从速度上秒杀springcloud原来的config几条街,

对比项目/配置中心	spring cloud config	apollo	nacos
开源时间	2014.9	2016.5	2018.6
配置实时推送	支持（Spring Cloud Bus）	支持（HTTP长轮询1s内）	支持（HTTP长轮询1s内）
版本管理	支持（Git）	自动管理	自动管理
配置回滚	支持（Git）	支持	支持
灰度发布	支持	支持	待支持
权限管理	支持	支持	待支持
多集群多环境	支持	支持	支持
监听查询	支持	支持	支持
多语言	只支持Java	Go,C++,Python,Java,.net,OpenAPI	Python,Java,Nodejs,OpenAPI
分布式高可用最小集群数量	Config-Server2+Git+MQ	Config2+Admin3+Portal*2+Mysql=8	Nacos*3+MySql=4
配置格式校验	不支持	支持	支持
通信协议	HTTP和AMQP	HTTP	HTTP
数据一致性	Git保证数据一致性，Config-Server从Git读取数据	数据库模拟消息队列，Apollo定时读消息	HTTP异步通知
单机读（tps）	7（限流所制）	9000	15000
单机写（tps）	5（限流所制）	1100	1800
3节点读	21（限流所制）	27000	45000
3节点写	5（限流所制）	3300	5600

1.1 快速开始

准备配置，nacos server中新建nacos-config.properties

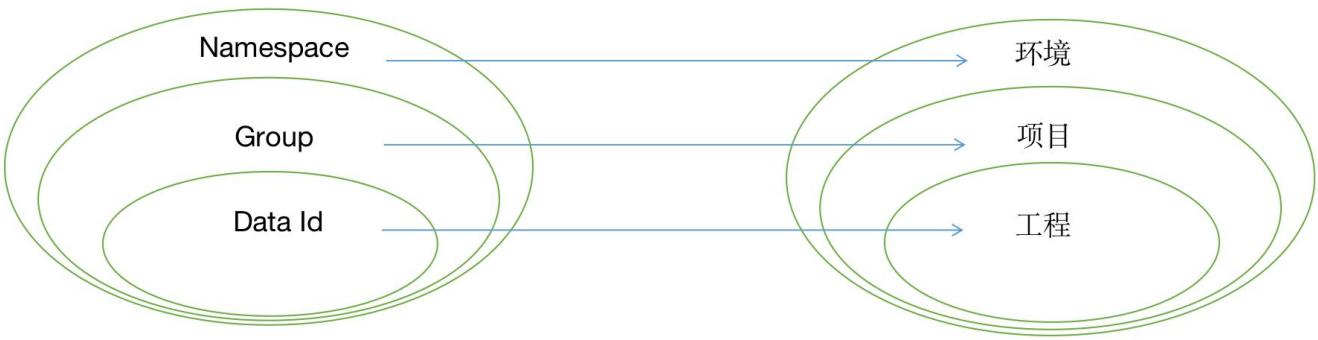


最佳实践：

Namespace：代表不同**环境**，如开发、测试、生产环境。

Group：代表某**项目**，如XX医疗项目、XX电商项目

DataId：每个项目下往往有若干个**工程（微服务）**，每个配置集(DataId)是一个**工程（微服务）**的**主配置文件**



启动权限：修改application.properties

```
### If turn on auth system:
nacos.core.auth.enabled=true
```

1.2 搭建nacos-config服务

通过 Nacos Server 和 spring-cloud-starter-alibaba-nacos-config 实现配置的动态变更

1）引入依赖

```
<dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
```

2）添加bootstrap.properties

```
spring.application.name=nacos-config
# 配置中心地址
spring.cloud.nacos.config.server-addr=127.0.0.1:8848

# dataid 为 yaml 的文件扩展名配置方式
# `${spring.application.name}.${file-extension:properties}`
spring.cloud.nacos.config.file-extension=yaml
#profile粒度的配置    `${spring.application.name}-${profile}.${file-extension:properties}`
```

3) 启动服务，测试微服务是否使用配置中心的配置

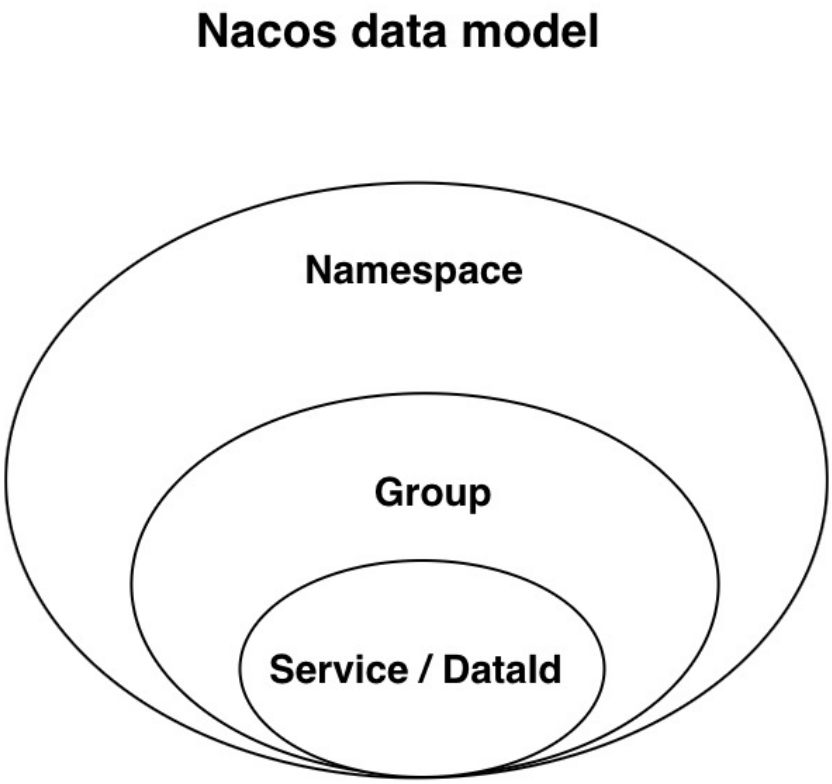
```
@SpringBootApplication
public class NacosConfigApplication {

    public static void main(String[] args) {
        ConfigurableApplicationContext applicationContext = SpringApplication.run(NacosConfigApplication.cla
        String userName = applicationContext.getEnvironment().getProperty("common.name");
        String userAge = applicationContext.getEnvironment().getProperty("common.age");
        System.out.println("common name :"+userName+"; age: "+userAge);
    }
```

```
2020-07-31 15:33:43.474  WARN 45352 --- [           main] c.a.c.n.c.NacosPropertySourceBuilder     : Ignore the empty
2020-07-31 15:33:43.479  INFO 45352 --- [           main] c.a.c.n.c.NacosPropertySourceBuilder     : Loading nacos da
user.age=30
2020-07-31 15:33:43.481  INFO 45352 --- [           main] b.c.PropertySourceBootstrapConfiguration : Located property
2020-07-31 15:33:43.484  INFO 45352 --- [           main] bat.ke.qq.com.NacosConfigApplication     : No active profil
2020-07-31 15:33:43.634  INFO 45352 --- [           main] o.s.cloud.context.scope.GenericScope     : BeanFactory id=6
2020-07-31 15:33:43.638  INFO 45352 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'org.spring
2020-07-31 15:33:45.202  INFO 45352 --- [           main] o.s.cloud.commons.util.InetUtils         : Cannot determine
2020-07-31 15:33:45.243  INFO 45352 --- [           main] bat.ke.qq.com.NacosConfigApplication     : Started NacosCon
user name :fox; age: 30
```

1.3 Config相关配置

Nacos 数据模型 Key 由三元组唯一确定, Namespace默认是空串，公共命名空间（public），分组默认是DEFAULT_GROUP



- **支持配置的动态更新**

```
@SpringBootApplication
public class NacosConfigApplication {

    public static void main(String[] args) throws InterruptedException {
        ConfigurableApplicationContext applicationContext = SpringApplication.run(NacosConfigApplication.cla

         while(true) {
        //当动态配置刷新时，会更新到 Enviroment中，因此这里每隔一秒中从Enviroment中获取配置
         String userName = applicationContext.getEnvironment().getProperty("common.name");
        String userAge = applicationContext.getEnvironment().getProperty("common.age");
        System.err.println("common name :" + userName + "; age: " + userAge);
            TimeUnit.SECONDS.sleep(1);
        }

    }

}

```

ps：除了默认的配置文件， 其他dataId都要加上后缀

- **支持profile粒度的配置**

spring-cloud-starter-alibaba-nacos-config 在加载配置的时候，不仅仅加载了以 dataid 为 `${spring.application.name}.${file-extension:properties}` 为前缀的基础配置，还加载了dataid为 `${spring.application.name}-${profile}.${file-`

`extension:properties}` 的基础配置。在日常开发中如果遇到多套环境下的不同配置，可以通过Spring 提供的 `${spring.profiles.active}` 这个配置项来配置。

```
spring.profiles.active=dev
```

profile 的配置文件 大于 默认配置的文件。 并且形成互补

ps：只有默认的配置文件， 才会应用profile

- **支持自定义 namespace 的配置**

用于进行租户粒度的配置隔离。不同的命名空间下，可以存在相同的 Group 或 Data ID 的配置。Namespace 的常用场景之一是不同环境的配置的区分隔离，例如开发测试环境和生产环境的资源（如配置、服务）隔离等。

在没有明确指定 `${spring.cloud.nacos.config.namespace}` 配置的情况下， 默认使用的是 Nacos 上 Public 这个namespace。如果需要使用自定义的命名空间，可以通过以下配置来实现：

```
spring.cloud.nacos.config.namespace=71bb9785-231f-4eca-b4dc-6be446e12ff8
```

- **支持自定义 Group 的配置**

Group是组织配置的维度之一。通过一个有意义的字符串（如 Buy 或 Trade ）对配置集进行分组，从而区分 Data ID 相同的配置集。当您在 Nacos 上创建一个配置时，如果未填写配置分组的名称，则配置分组的名称默认采用 DEFAULT_GROUP 。配置分组的常见场景：不同的应用或组件使用了相同的配置类型，如 database_url 配置和 MQ_topic 配置。

在没有明确指定 `${spring.cloud.nacos.config.group}` 配置的情况下，默认是DEFAULT_GROUP 。如果需要自定义自己的Group，可以通过以下配置来实现：

```
spring.cloud.nacos.config.group=DEVELOP_GROUP
```

- **支持自定义扩展的 Data Id 配置**

Data ID 是组织划分配置的维度之一。Data ID 通常用于组织划分系统的配置集。一个系统或者应用可以包含多个配置集，每个配置集都可以被一个有意义的名称标识。Data ID 通常采用类 Java 包（如 com.taobao.tc.refund.log.level）的命名规则保证全局唯一性。此命名规则非强制。

通过自定义扩展的 Data Id 配置，既可以解决多个应用间配置共享的问题，又可以支持一个应用有多个配置文件。

```
# 自定义 Data Id 的配置
#不同工程的通用配置 支持共享的 DataId
spring.cloud.nacos.config.sharedConfigs[0].data-id= common.yaml
spring.cloud.nacos.config.sharedConfigs[0].group=REFRESH_GROUP
spring.cloud.nacos.config.sharedConfigs[0].refresh=true

# config external configuration
# 支持一个应用多个 DataId 的配置  一定要加扩展名
spring.cloud.nacos.config.extensionConfigs[0].data-id=ext-config-common01.properties
spring.cloud.nacos.config.extensionConfigs[0].group=REFRESH_GROUP
spring.cloud.nacos.config.extensionConfigs[0].refresh=true

spring.cloud.nacos.config.extensionConfigs[1].data-id=ext-config-common02.properties
spring.cloud.nacos.config.extensionConfigs[1].group=REFRESH_GROUP
```

1.4 配置的优先级

Spring Cloud Alibaba Nacos Config 目前提供了三种配置能力从 Nacos 拉取相关的配置。

- A: 通过 `spring.cloud.nacos.config.shared-configs` 支持多个共享 Data Id 的配置
- B: 通过 `spring.cloud.nacos.config.ext-config[n].data-id` 的方式支持多个扩展 Data Id 的配置
- C: 通过内部相关规则(应用名、应用名+ Profile)自动生成相关的 Data Id 配置

当三种方式共同使用时，他们的一个优先级关系是:A < B < C

优先级从高到低:

1) nacos-config-product.yaml 精准配置 2) nacos-config.yaml 同工程不同环境的通用配置 3) ext-config: 不同工程 扩展配置 4) shared-dataids 不同工程通用配置

1.5 @RefreshScope

@Value注解可以获取到配置中心的值，但是无法动态感知修改后的值，需要利用@RefreshScope注解

```
@RestController
@RefreshScope
public class TestController {

    @Value("${common.age}")
    private String age;

    @GetMapping("/common")
    public String hello() {
        return age;
    }

```