



# 注册中心之Nacos详细解析

● 君哥 ● 2021-05-31 15:44:15

📍 注册中心 配置中心 Nacos

你的支持是君哥更新的动力

阿里巴巴开源注册配置中心详解

加入君哥微信粉丝群：coding178

点击观看全套视频



## 注册中心

注册中心在微服务项目中扮演着非常重要的角色，是微服务架构中的纽带，类似于“通讯录”，它记录了服务和服务地址的映射关系。在分布式架构中，服务会注册到这里，当服务需要调用其它服务时，就到这里找到服务的地址，进行调用。

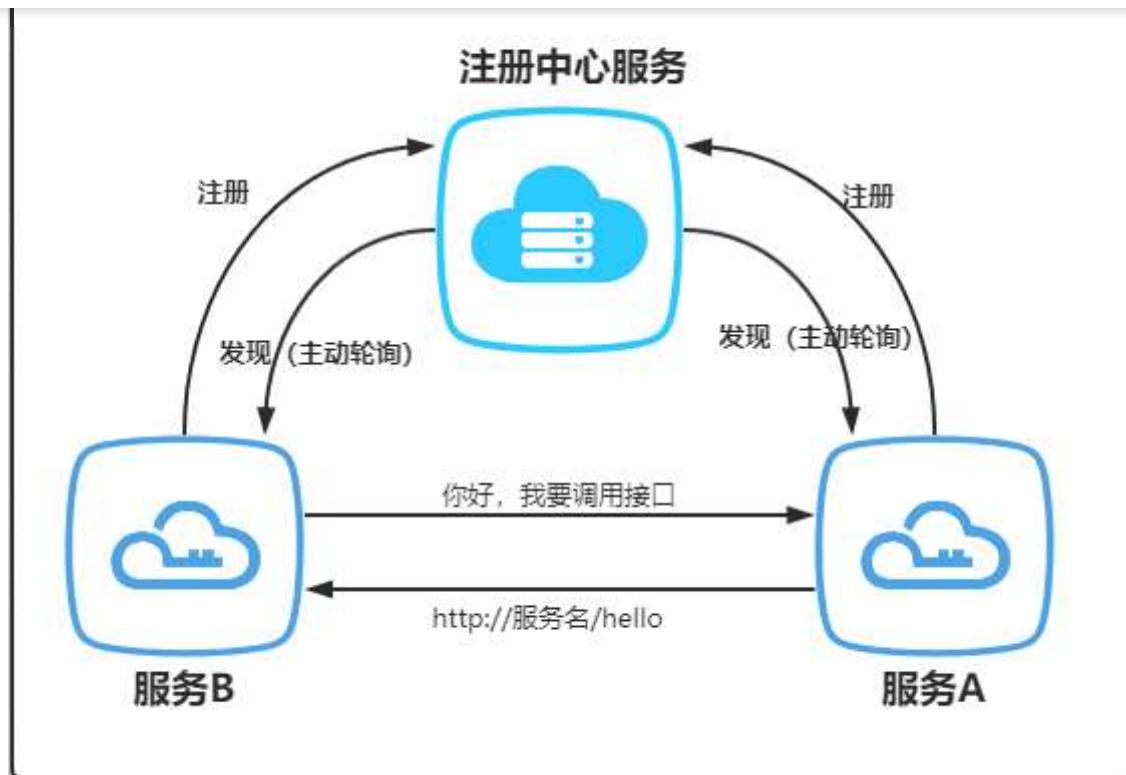
在分布式系统中，我们面临着以下几个问题

如何管理庞大的网状服务结构？

服务宕机后，如何及时下线？

流量高峰时，服务如何有效的水平扩展？





除了基本的服务注册与发现机制，从开发和运维角度，至少还要考虑如下五个方面：

- 测活：服务注册之后，如何对服务进行测活以保证服务的可用性？
- 负载均衡：当存在多个服务提供者时，如何均衡各个提供者的负载？
- 集成：在服务提供端或者调用端，如何集成注册中心？
- 运行时依赖：引入注册中心之后，对应用的运行时环境有何影响？
- 可用性：如何保证注册中心本身的可用性，特别是消除单点故障？



## Nacos注册中心

### 快速上手

Nacos [\[1\]](#) 是阿里巴巴开源的一个更易于构建云原生应用的动态服务发现、配置管理和服务管理平台。中文文档非常齐全，[\[2\]](#) 文档地址戳 [\[3\]](#)





## 什么是 Nacos

### 概览

欢迎来到 Nacos 的世界！

Nacos 致力于帮助您发现、配置和管理微服务。Nacos 提供了一组简单易用的特性集，帮助您快速实现动态服务发现、服务配置、服务元数据及流量管理。

Nacos 帮助您更敏捷和容易地构建、交付和管理微服务平台。Nacos 是构建以“服务”为中心的现代应用架构（例如微服务范式、云原生范式）的服务基础设施。

### 什么是 Nacos?

服务（Service）是 Nacos 世界的一等公民。Nacos 支持几乎所有主流类型的“服务”的发现、配置和管理：

Kubernetes Service

gRPC & Dubbo RPC Service

Spring Cloud RESTful Service

Nacos 的关键特性包括：

## 下载与启动

- 源码下载

```

1 $ git clone https://github.com/alibaba/nacos.git
2 $ cd nacos/
3 $ mvn -Prelease-nacos -Dmaven.test.skip=true clean install -U
4 $ ls -al distribution/target/
5
6 // change the $version to your actual path
7 $ cd distribution/target/nacos-server-$version/nacos/bin

```

- 安装包下载



Windows 版本: 所有用户 (4.10) , 日期: 2021-08-28 23:35:16 , 地点: 上海市徐汇区漕河泾开发区。

C:\WINDOWS\system32\cmd.exe

```
Nacos
Running in stand alone mode, All function modules
Port: 8848
Pid: 23468
Console: http://192.168.1.4:8848/nacos/index.html
https://nacos.io
```

```
2020-08-28 23:35:05,298 INFO Bean 'org.springframework.security.config.annotation.configuration.ObjectPostProcessorConfiguration' configuration. ObjectPostProcessorConfiguration$$EnhancerBySpringCGLIB$$6bf9c417] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
2020-08-28 23:35:07,049 INFO Bean 'objectPostProcessor' of type [org.springframework.security.config.annotation.configuration.AutoProxyingObjectPostProcessor] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
2020-08-28 23:35:07,098 INFO Bean 'org.springframework.security.access.expression.method.DefaultMethodSecurityExpressionHandler@5f1a2e3d' configuration. DefaultMethodSecurityExpressionHandler] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
2020-08-28 23:35:07,107 INFO Bean 'org.springframework.security.config.annotation.method.configuration.GlobalMethodSecurityConfiguration' configuration. GlobalMethodSecurityConfiguration$$EnhancerBySpringCGLIB$$90ce66c9] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
2020-08-28 23:35:07,219 INFO Bean 'methodSecurityMetadataSource' of type [org.springframework.security.access.method.DelegatingMethodSecurityMetadataSource] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
2020-08-28 23:35:14,598 INFO Tomcat initialized with port(s): 8848 (http)
2020-08-28 23:35:16,395 INFO Root WebApplicationContext: initialization completed in 44388 ms
```

如果不小心报了如下错：



```
1  java.io.IOException: java.lang.IllegalArgumentException: db.num is null
2  org.springframework.boot.web.embedded.tomcat.TomcatStarter.onStartup
3  work.boot.web.server.WebServerException: Unable to start embedded Tomcat
```

那说明你的Nacos默认是集群模式，需要改为单机模式

- 找到 startup.cmd 文件，并编辑
- 把集群模式改为单机模式

java

```
1  set MODE="cluster"
2  改为:
3  set MODE="standalone"
```





```

2020-09-05 17:20:35,653 INFO The server IP list of Nacos is []
2020-09-05 17:20:36,655 INFO Nacos is starting...
2020-09-05 17:20:37,656 INFO Nacos is starting...
2020-09-05 17:20:38,658 INFO Nacos is starting...
2020-09-05 17:20:39,659 INFO Nacos is starting...
2020-09-05 17:20:40,660 INFO Nacos is starting...
2020-09-05 17:20:41,661 INFO Nacos is starting...
2020-09-05 17:20:42,663 INFO Nacos is starting...

java.io.IOException: java.lang.IllegalArgumentException: db.num is null
    at com.alibaba.nacos.config.server.service.datasource.ExternalDataSourceServiceImpl.reload(ExternalDataSourceService
    viceImpl.java:141)
    at com.alibaba.nacos.config.server.service.datasource.ExternalDataSourceServiceImpl.init(ExternalDataSourceServ
    iceImpl.java:115)
    at com.alibaba.nacos.config.server.service.datasource.DynamicDataSource.getDataSource(DynamicDataSource.java:53) ~
    [nacos-server-1.1.0.jar!/:na]

```

- Linux/Unix/Mac

```

1 $ unzip nacos-server-$version.zip 或者 tar -xvf nacos-server-$version.tar.gz
2 $ cd nacos/bin

```

启动命令(standalone代表着单机模式运行，非集群模式):

```

1 $ sh startup.sh -m standalone

```

如果您使用的是ubuntu系统，或者运行脚本报错提示[[符号找不到，可尝试如下运行：

```

1 $ bash startup.sh -m standalone

```



三 君哥的学习笔记



- 打开控制台：

Nacos提供了一个可视化的操作平台，安装好之后，在浏览器中输入

<http://localhost:8848> 就可以访问了,默认的用户名和密码都是nacos (我使用的是1.3.0版本)

The screenshot shows the Nacos 1.3.0 web interface. The top navigation bar includes links for 首页 (Home), 文档 (Documentation), 博客 (Blog), 社区 (Community), and En (English). A logo for 'NACOS.' is on the left. On the far right is a large circular icon with a double-headed arrow. The main content area has a sidebar with tabs: 配置管理 (Configuration Management), 服务管理 (Service Management), 服务列表 (Service List), 订阅者列表 (Subscriber List), 权限控制 (Permission Control), 命名空间 (Namespace), and 集群管理 (Cluster Management). The '服务管理' tab is active, showing a search bar for '服务名称' (Service Name) and '分组名称' (Group Name), a toggle switch for '隐藏空服务' (Hide empty services), and a '查询' (Search) button. Below this is a table with columns: 服务名 (Service Name), 分组名 (Group Name), 集群数目 (Cluster Count), 实例数 (Instance Count), 健康实例数 (Healthy Instance Count), 触发保护阈值 (Trigger Protection Threshold), and 操作 (Operations). A message '没有数据' (No data) is displayed at the bottom of the table.

构建服务

Eureka 和 Eureka 不一样，并不需要创建新的web项目，而是和 Zookeeper 和 Consul 一样，只需要下载安装启动后，将我们的微服务注册进去就可以了。创建两个微服务，一个客户端（调用者）和一个服务端（提供者）

这里我先创建一个名为 `it235-nacos-parent` 的聚合模块父项目，后续所有模块均在该项目中完成



# 君哥的学习笔记



com.atlassian.maven.archetypes:bamboo-plugin-archetype  
com.atlassian.maven.archetypes:confluence-plugin-archetype  
com.atlassian.maven.archetypes:jira-plugin-archetype  
com.rfc.maven.archetypes:jpa-maven-archetype  
de.akquinet.jbosscc:jbosscc-seam-archetype  
net.databinder:data-app  
net.liftweb:lift-archetype-basic  
net.liftweb:lift-archetype-blank  
net.sf.maven-har:maven-archetype-har  
net.sf.maven-sar:maven-archetype-sar  
org.apache.camel.archetypes:camel-archetype-activemq  
org.apache.camel.archetypes:camel-archetype-component  
org.apache.camel.archetypes:camel-archetype-java  
org.apache.camel.archetypes:camel-archetype-scala  
org.apache.camel.archetypes:camel-archetype-spring  
org.apache.camel.archetypes:camel-archetype-war  
org.apache.cocoon:cocoon-22-archetype-block  
org.apache.cocoon:cocoon-22-archetype-block-plain  
org.apache.cocoon:cocoon-22-archetype-webapp  
org.apache.maven.archetypes:maven-archetype-j2ee-simple

Previous Next Cancel Help

New Project

GroupId: com.it235.nacos

ArtifactId: it235-nacos-parent

Version: 1.0.0-SNAPSHOT

Next

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help it235-nacos-parent [D:\self\zmx\code\it235-nacos-parent] ~...\pom

it235-nacos-parent pom.xml

Project Explorer

it235-nacos-parent D:\self\zmx\code\it235-nacos-parent  
|.idea  
| it235-nacos-parent.iml  
| pom.xml  
> External Libraries  
Scatches and Consoles

pom.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.it235.nacos</groupId>
    <artifactId>it235-nacos-parent</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <!-- Nacos 讲解的父项目-->
</project>

```

空壳子即可

## 1. 调用端（消费者）

- 创建项目



# 君哥的学习笔记



Choose Initializr Service URL.

Default: <https://start.spring.io>

Custom:

Make sure your network connection is active before continuing.

利用spring脚手架创建

New Module

**Project Metadata**

Group: com.it235.nacos

Artifact: order-service-consumer

Type: Maven Project (Generate a Maven based project archive.)

Language: Java

Packaging: Jar

Java Version: 8

Version: 1.0.0-SNAPSHOT

Name: order-service-consumer

Description: 订单服务作为消费者

Package: com.it235.nacos.order.service



Project: it235-nacos-parent > order-service-consumer > src > main > java > com > it235 > nacos > order > service > consumer > OrderServiceConsumerApplication

File: OrderServiceConsumerApplication.java

```

package com.it235.nacos.order.service.consumer;

import ...

@SpringBootApplication
public class OrderServiceConsumerApplication {

    public static void main(String[] args) { SpringApplication.run(OrderServiceConsumerApplication.class, args); }
}

```

建立子项目后，清理掉一些不需要的mvn等文件

Project Explorer:

- it235-nacos-parent
- .idea
- order-service-consumer
  - src
    - main
      - java
        - com
          - it235
            - nacos
              - order
                - service
                  - consumer

Resources:

  - static
  - templates
  - application.properties

Test:

  - order-service-consumer.iml
  - pom.xml
  - .gitignore
  - it235-nacos-parent.iml
  - pom.xml

External Libraries

Scratches and Consoles



- 引入依赖



```
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://mave
4      <modelVersion>4.0.0</modelVersion>
5      <!-- 继承springboot-->
6      <parent>
7          <groupId>org.springframework.boot</groupId>
8          <artifactId>spring-boot-starter-parent</artifactId>
9          <version>2.2.5.RELEASE</version>
10         <relativePath/> <!-- lookup parent from repository -->
11     </parent>
12
13     <groupId>com.it235.nacos</groupId>
14     <artifactId>order-service-consumer</artifactId>
15     <version>1.0.0-SNAPSHOT</version>
16     <name>order-service-consumer</name>
17     <description>订单服务作为消费者</description>
18
19     <!-- 声明相关依赖版本-->
20     <properties>
21         <java.version>1.8</java.version>
22         <spring-cloud-alibaba.version>2.2.1.RELEASE</spring-cloud-alibab
23     </properties>
24
25     <dependencies>
26         <!-- 作为web容器启动-->
27         <dependency>
28             <groupId>org.springframework.boot</groupId>
29             <artifactId>spring-boot-starter-web</artifactId>
30         </dependency>
31         <!-- 加入阿里服务发现依赖 重要-->
32         <dependency>
33             <groupId>com.alibaba.cloud</groupId>
34             <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artif
35         </dependency>
36     </dependencies>
37
38     <!-- alibaba cloud 相关依赖管理-->
39     <dependencyManagement>
40         <dependencies>
41             <dependency>
42                 <groupId>com.alibaba.cloud</groupId>
43                 <artifactId>spring-cloud-alibaba-dependencies</artifactId>
44                 <version>${spring-cloud-alibaba.version}</version>
45                 <type>pom</type>
46                 <scope>import</scope>
```





```

50
51      <!--springboot 打包构建插件-->
52      <build>
53          <plugins>
54              <plugin>
55                  <groupId>org.springframework.boot</groupId>
56                  <artifactId>spring-boot-maven-plugin</artifactId>
57              </plugin>
58          </plugins>
59      </build>
60  </project>

```

- 添加Nacos配置

- 修改 application.properties 文件为 bootstrap.yml (Nacos首先加载该命名文件)
- bootstrap.yml 中加入如下配置

```

1  server:
2      port: 6010
3  spring:
4      application:
5          name: order-service
6      cloud:
7          nacos:
8              discovery:
9                  #必须配置ip地址
10                 server-addr: localhost:8848
11                 # 将自己的服务注册到注册中心
12                 register-enabled: true
13                 #namespace: all-register-service-namespace

```



- 主 Application 启动类加入服务发现注解 @EnableDiscoveryClient

```

1  @SpringBootApplication
2  @EnableDiscoveryClient //重点
3  public class OrderServiceConsumerApplication {
4      public static void main(String[] args) {
5          SpringApplication.run(OrderServiceConsumerApplication.class, arg

```





# 君哥的学习笔记



```

13 <version>1.0.0-SNAPSHOT</version>
14 <name>course-service-provider</name>
15 <description>课程服务作为消费者</description>
16
17 <properties>
18   <java.version>1.8</java.version>
19   <spring-cloud-alibaba.version>2.2.1.RELEASE</spring-cloud-alibaba.version>
20 </properties>
21
22 <dependencies>
23   <dependency>
24     <groupId>org.springframework.boot</groupId>
25     <artifactId>spring-boot-starter-web</artifactId>
26   </dependency>

```

project > dependencies > dependency > artifactId

Text Dependency Analyzer

开启服务面板（重要）

**Services**  
Multiple Spring Boot run configurations were detected.  
Services allows to manage multiple run configurations at once.

[Show run configurations in Services](#)

Do not show again for this project

Java Enterprise 0: Messages 25: S 英 Event Log

java

resources

- static
- templates
- bootstrap.yml

Services

Spring Boot

Running

CourseServiceProviderApplication :6020/

OrderServiceConsumerApplication

Loaded classes are up to date. Nothing to reload.

服务面板方便管理SpringBoot应用程序

- 启动服务，查看 Nacos 控制台的服务列表



The screenshot shows the Nacos service list interface. On the left is a sidebar with options like '配置管理', '服务管理' (highlighted with a red box), '订阅者列表', '权限控制', '命名空间', and '集群管理'. The main area has a title '服务列表 | public'. It includes search fields for '服务名称' and '分组名称', a '隐藏空服务' switch, and a '查询' button. A table lists services with columns: 服务名, 分组名称, 集群数目, 实例数, 健康实例数, and 触发保护阈值. One row is highlighted with a red box: 'order-service' under '服务名', 'DEFAULT\_GROUP' under '分组名称', '1' under '集群数目', '1' under '实例数', '1' under '健康实例数', and 'false' under '触发保护阈值'. Below the table, a message says '该服务已注册到注册中心'.

## 2. 提供者 (生产者)

- 创建项目

依葫芦画瓢创建 course-service-provider 项目，并配置，此处忽略

The screenshot shows the IntelliJ IDEA interface with two projects: 'course-service-provider' and 'order-service-consumer'. The 'course-service-provider' project structure is displayed in the Project tool window, showing packages like 'com.it235.course.provider' and files like 'CourseServiceProviderApplication.java' and 'bootstrap.yml'. A note in the center says: '课程服务, 供订单服务调用 端口: 6020 应用名: course-service'. The 'bootstrap.yml' file content is shown in the editor:

```

1 server:
2   port: 6020
3
4 spring:
5
6   application:
7     name: course-service
8
9   cloud:
10
11   nacos:
12     discovery:
13       #必须配置ip地址
14       server-addr: localhost:8848
15       # 将自己的服务注册到注册中心
16       register-enabled: true
17       #namespace: all-register-service-namespace

```

The 'order-service-consumer' project structure is also shown, with a note: '订单服务, 主动调用课程服务 端口: 6010 应用名: order-service'. The 'bootstrap.yml' file content is partially visible:

```

1 server:
2   port: 6010
3
4 spring:
5
6   application:
7     name: order-service
8
9   cloud:
10
11   nacos:
12     discovery:
13       #必须配置ip地址
14       server-addr: localhost:8848
15       # 将自己的服务注册到注册中心
16       register-enabled: true
17       #namespace: all-register-service-namespace

```

- 引入依赖

与 order-service-consumer 的依赖保持一致



- 添加 Nacos 配置

```

1 server:
2   port: 6020
3
4 spring:
5
6   application:
7     name: course-service
8
9   cloud:
10
11   nacos:
12     discovery:
13       #必须配置ip地址
14       server-addr: localhost:8848
15       # 将自己的服务注册到注册中心
16       register-enabled: true
17       #namespace: all-register-service-namespace

```



```

11 # 将自己的服务注册到注册中心
12 register-enabled: true
13 #namespace: all-register-service-namespace

```

- 课程服务主 Application 启动类添加服务发现注解

```

1 @SpringBootApplication
2 @EnableDiscoveryClient //重要
3 public class CourseServiceProviderApplication {
4     public static void main(String[] args) {
5         SpringApplication.run(CourseServiceProviderApplication.class, args)
6     }
7 }

```

### 3. 启动服务

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows the project tree with modules like `course-service-provider.iml`, `pom.xml`, and `order-service-consumer`.
- Dependency Analyzer:** A floating window on the right displays the dependency tree for the `order-service-consumer` module, specifically focusing on the `spring-boot-starter` dependency.
- Services:** The bottom-left panel shows the running services: `CourseServiceProviderApplication :6020/` and `OrderServiceConsumerApplication :6010/`. A message "2个服务都启动成功" (Both services started successfully) is displayed in red.
- Console:** The bottom-right panel shows the application logs with INFO and WARN levels.
- Status Bar:** A green message at the bottom says "Loaded classes are up to date. Nothing to reload."



服务列表 | public

| 服务名            | 分组名称          | 集群数目 | 实例数 | 健康实例数 | 触发保护阈值 |
|----------------|---------------|------|-----|-------|--------|
| course-service | DEFAULT_GROUP | 1    | 1   | 1     | false  |
| order-service  | DEFAULT_GROUP | 1    | 1   | 1     | false  |

刷新控制台，此时2个服务都已经注册到Nacos

## 添加服务调用模块

为最简单演示，这里我直接使用restTemplate对象来进行远程服务调用

### 1. 给course-service模块添加controller控制器，用来提供给远程服务调用

```

1  @RestController
2  @RequestMapping("course")
3  public class CourseController {
4
5      @GetMapping("list")
6      public String get(){
7          return "已经访问到课程服务";
8      }
9  }

```

it235-nacos-parent > course-service-provider > src > main > java > com > it235 > nacos > course > service > provider > controller > CourseController.java

Project

1-Project

1-Project

CourseController.java

```

1  package com.it235.nacos.course.service.provider.controller;
2
3  import org.springframework.web.bind.annotation.GetMapping;
4  import org.springframework.web.bind.annotation.RequestMapping;
5  import org.springframework.web.bind.annotation.RestController;
6
7  @RestController
8  @RequestMapping("course")
9  public class CourseController {
10
11     @GetMapping("list")
12     public String get() { return "已经访问到课程服务"; }
13
14 }

```



```

1  @SpringBootApplication
2  @EnableDiscoveryClient
3  public class OrderServiceConsumerApplication {
4
5      // 新增restTemplate对象注入方法，注意，此处LoadBalanced注解一定要加上，否则
6      @Bean
7      @LoadBalanced
8      public RestTemplate restTemplate(){
9          return new RestTemplate();
10     }
11     // 新增 end
12
13     public static void main(String[] args) {
14         SpringApplication.run(OrderServiceConsumerApplication.class, args);
15     }
16 }
```

java

- 添加 OrderController，并使用 restTemplate 对象调用远程的课程服务方法



```

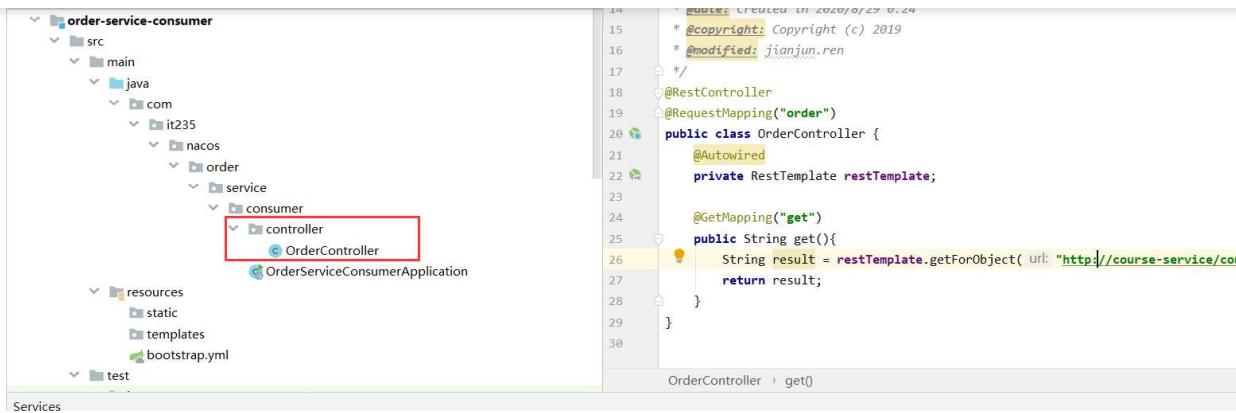
1  @RestController
2  @RequestMapping("order")
3  public class OrderController {
4
5      @Autowired
6      private RestTemplate restTemplate;
7
8      @GetMapping("get")
9      public String get(){
10         //通过服务名的方式调用远程服务（非ip端口）
11         String result = restTemplate.getForObject(
12             "http://course-service/course/list", String.class);
13         return result;
14     }
15 }
```

java





# 君哥的学习笔记

```

order-service-consumer
├── src
│   ├── main
│   │   ├── java
│   │   │   └── com
│   │   │       └── it235
│   │   │           └── nacos
│   │   │               ├── order
│   │   │               │   └── service
│   │   │               └── consumer
│   │   │                   └── controller
│   │   │                       └── OrderController
│   │   └── resources
│   └── test
└── bootstrap.yml

```

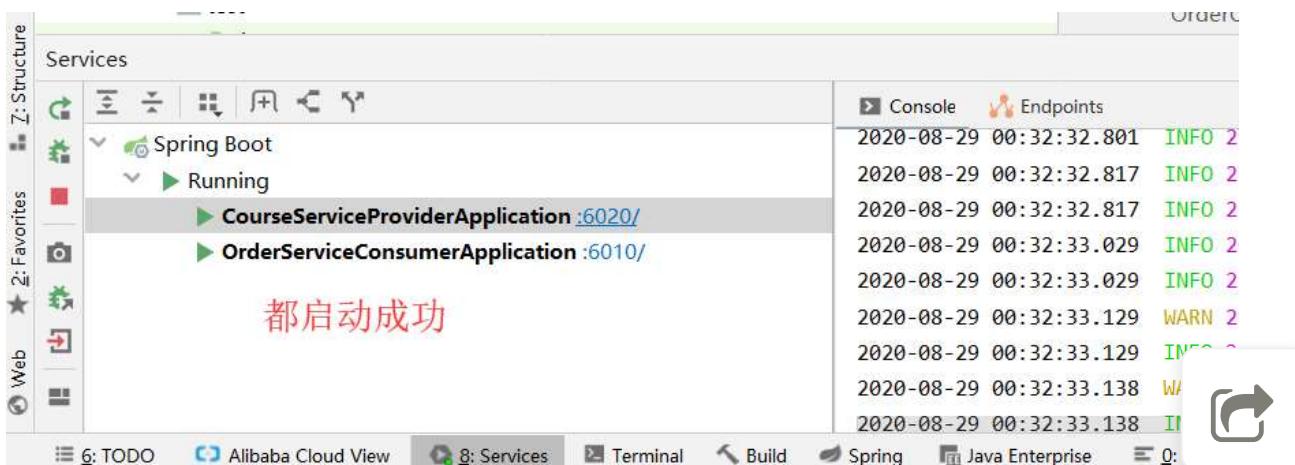
```

14 * @copyright: Copyright (c) 2019
15 * @modified: jianjun.ren
16 */
17
18 @RestController
19 @RequestMapping("order")
20 public class OrderController {
21     @Autowired
22     private RestTemplate restTemplate;
23
24     @GetMapping("get")
25     public String get(){
26         String result = restTemplate.getForObject( url: "http://course-service/con", 
27         return result;
28     }
29
30 }

```

OrderController > get()

- 启动服务，并查看Nacos注册情况



都启动成功



| 服务名            | 分组名称          | 集群数目 | 实例数 | 健康实例数 |
|----------------|---------------|------|-----|-------|
| course-service | DEFAULT_GROUP | 1    | 1   | 1     |
| order-service  | DEFAULT_GROUP | 1    | 1   | 1     |

都注册成功

- 

3. 浏览器访问URL，进行测试 <http://localhost:6010/order/get>

如果能正常返回 course-service 提供的值，则表示Demo运行成功



### 三 君哥的学习笔记

已经访问到课程服务

该值由course-service提供

#### 4. 错误解析

如出现 nested exception is java.net.UnknownHostException: course-service 错误, 请在 order-service 项目的主Application中加上 @LoadBalanced 注解

课件代码详见: <https://gitee.com/appdoc/it235.git>, tag名: 1.0.0 ↗

| 标签名   | 描述                                     | 提交信息                       | 操作          |
|-------|--|----------------------------|-------------|
| 1.2.0 | nacos配置中心完善, 加入prefix-active.extension | 5cce949 · 2020-09-14 12:10 | 下载 创建发行版 删除 |
| 1.1.0 | nacos配置中心前置准备, value读取数据               | 697ffe4 · 2020-09-14 12:08 | 下载 创建发行版 删除 |
| 1.0.0 | nacos远程调用服务实现                          | 86aa401 · 2020-09-14 10:45 | 下载 创建发行版 删除 |

#### 流程剖析

浏览器->order-service服务->course-service服务

### MySQL支持

在前面的学习中, 我们对于Nacos服务端自身并没有做过什么特殊的配置, 一切均以默认的单机模式运行, 完成了上述所有功能的学习。但是, Nacos的单机运行模式仅适用于学习与测试环境, 对于有高可用要求的生产环境显然是不合适的。

在搭建Nacos集群之前, 我们需要先修改Nacos的数据持久化配置为MySQL存储。默认情况下, Nacos使用嵌入式数据库实现数据的存储。所以, 如果启动多个默认配置下的Nacos节点, 数据存储是存在一致性问题的。为了解决这个问题, Nacos采用了集中式存储的方式来支持集群化部署, 目前只要支持MySQL的存储。

### 三 君哥的学习笔记



MySQL8.0+，具体的操作步骤：

- 安装数据库，版本要求：5.6.5+
- 初始化 mysql 数据库
- 修改 conf/application.properties 文件，增加支持 mysql 数据源配置（目前只支持 mysql），添加 mysql 数据源的url、用户名和密码。

```

1 # 此项一定要启用，默认是注释掉的
2 spring.datasource.platform=mysql
3
4 # 注意MySQL8.0以上版本指定url时一定要带入serverTimezone参数
5 db.num=1
6 db.url.0=jdbc:mysql://localhost:3306/nacos_config?serverTimezone=Asia/Shanghai&c
7 db.user=root
8 db.password=123456
9
10 # 可选启用配置
11 nacos.cmdb.dumpTaskInterval=3600
12 nacos.cmdb.eventTaskInterval=10
13 nacos.cmdb.labelTaskInterval=300
14 nacos.cmdb.loadDataAtStart=false

```



1. 这个的 application.properties 指nacos的解压目录 nacos/conf 目录下的文件
2. 这里的db具体配置根据自身情况而变

再以单机模式启动 nacos，nacos 所有写嵌入式数据库的数据都写到了 mysql，访问UI查看是否部署成功，并进行数据库验证。

1. 添加一行配置

Nacos 主界面->配置管理->右边+号新建





\* Data ID: test

\* Group: DEFAULT\_GROUP

[更多高级选项](#)

描述: test

配置格式:  TEXT  JSON  XML  YAML  HTML  Properties

\* 配置内容: 1 aaa 1 右下角保存发布 ?

## 2. 保存发布

NACOS 1.3.2

public

配置管理 | public

查询结果: 共查询到 1 条满足要求的配置。

|                          | Data Id | Group         |
|--------------------------|---------|---------------|
| <input type="checkbox"/> | test    | DEFAULT_GROUP |

删除 导出选中的配置 克隆

## 3. 数据库查看





|   | id | data_id | group_id      | content | md5     | gmt_c               |
|---|----|---------|---------------|---------|---------|---------------------|
| ▶ | 1  | test    | DEFAULT_GROUP | aaa     | 47bce5c | 2020-08-24 14:45:20 |

## 启动异常分析

若同学出现如下错误，请检查MySQL相关配置

### 1. db.url未指定 serverTimezone

```

1 org.springframework.jdbc.CannotGetJdbcConnectionException: Failed to obtain JI
2 ...
3 Caused by: com.mysql.cj.exceptions.InvalidConnectionAttributeException: The si

```



### 2. Nacos早期版本不兼容 MySQL8.0 数据库

Nacos 1.2.0 之前的版本使用的 mysql-connector-java-version.jar 为5.x，如果要使 Nacos 1.2.0 支持 MySQL8.0+ 数据库，需要替换该驱动包。

1.2.0  
-o 842818f

Compare ▾

### 1.2.0(Mar 4th, 2020) [Not Suggest](Have MD5 Problem)

nkorange released this on 4 Mar · 610 commits to develop since this release

#2370 com.alibaba.nacos.naming.misc.GlobalConfig taskDispatchPeriod 参数初始值疑问  
#2258 HealthCheckReactor is not isolated by namespace  
#2248 Nacos server application.properties reloadable  
#2232 Distro load data failed cause start failed.  
#2184 nacos 1.1.4 删除配置ui展示列表为空  
#2171 支持同一个 namesapce 中的克隆  
#2168 page bug in configuration list  
#2145 支持自定义namespace id  
#2123 Client heart beat package is too large  
#2056 整合dubbo采用nacos作为注册中心报错 [NACOS SocketTimeoutException httpPost] currentServerAddr: I  
err: Read timed out  
#2042 mysql8.0 support error  
#2020 com.alibaba.nacos.client.config.utils.IOUtils.toString NPE  
#2018 http connection Keepalive is invalid, dubbo consumer has many TIME\_WAIT  
#2006 Choose instance for nacos  
#2000 nacos 1.1.3版本, 可能产生死循环bug  
#1993 集群模式bug, 使用discovery会报错java.lang.IllegalStateException: failed to req API:/nacos/v1/ns/instance

### 三 君哥的学习笔记



版本的删除，替换为8.x的jar

- 替换后结果如下

支持MySQL 8

| 名称                                       | 压缩前           | 压缩后           | 类型                         | 修改日期                    |
|--|---------------|---------------|----------------------------|-------------------------|
| metrics-core-4.0.3.jar                   | 95.4 KB       | 95.4 KB       | Executable Jar File        | 2018-07-18 22:29        |
| micrometer-core-1.1.1.jar                | 411.4 KB      | 411.4 KB      | Executable Jar File        | 2018-11-29 13:29        |
| micrometer-registry-elastic-1.1.1.jar    | 12.4 KB       | 12.4 KB       | Executable Jar File        | 2018-11-29 15:58        |
| micrometer-registry-influx-1.1.1.jar     | 15.8 KB       | 15.8 KB       | Executable Jar File        | 2018-11-29 15:59        |
| micrometer-registry-prometheus-1.1.1.jar | 25.2 KB       | 25.2 KB       | Executable Jar File        | 2018-11-29 16:00        |
| mina-core-2.0.0-RC1.jar                  | 623.3 KB      | 623.3 KB      | Executable Jar File        | 2009-10-19 10:43        |
| <b>mysql-connector-java-8.0.16.jar</b>   | <b>2.2 MB</b> | <b>2.2 MB</b> | <b>Executable Jar File</b> | <b>2019-03-20 20:08</b> |
| nacos-api-1.3.2.jar                      | 62.0 KB       | 62.0 KB       | Executable Jar File        | 2020-08-04 19:26        |
| nacos-client-1.3.2.jar                   | 233.4 KB      | 233.4 KB      | Executable Jar File        | 2020-08-04 19:26        |
| nacos-cmdb-1.3.2.jar                     | 16.2 KB       | 16.2 KB       | Executable Jar File        | 2020-08-04 19:26        |
| nacos-common-1.3.2.jar                   | 119.6 KB      | 119.6 KB      | Executable Jar File        | 2020-08-04 19:26        |
| nacos-config-1.3.2.jar                   | 458.0 KB      | 458.0 KB      | Executable Jar File        | 2020-08-04 19:26        |
| nacos-consistency-1.3.2.jar              | 62.6 KB       | 62.6 KB       | Executable Jar File        | 2020-08-04 19:26        |
| nacos-core-1.3.2.jar                     | 203.3 KB      | 203.3 KB      | Executable Jar File        | 2020-08-04 19:26        |
| nacos-istio-1.3.2.jar                    | 310.5 KB      | 310.5 KB      | Executable Jar File        | 2020-08-04 19:26        |
| nacos-naming-1.3.2.jar                   | 331.0 KB      | 331.0 KB      | Executable Jar File        | 2020-08-04 19:26        |

大小: 77.5 MB 共 506 个文件和 110 个文件夹 压缩率 90.7% 已经选择 2.2 MB (1 个文件)

双核压缩引擎已开

## 配置中心

### 框架整合

配置中心知名的有 Apollo , Spring Cloud Config , 基于以上项目，我们继续来完成配置中心

application.yml 里边配置的一些系统变量数据, 通常会再 Controller 里边用 @Value 取出使用, 但是你要是想改变他, 就要重新改代码, 打包, 部署, 十分麻烦, 我们需要让配置文件的值变得动起来, Nacos 也采用了 Spring Cloud 原生注解 @RefreshScope 实现配置自动更新,

1. 在需要添加配置中心的服务中添加mvn依赖

```

1  <!-- SpringCloud Alibaba Nacos Config -->
2  <dependency>
3    <groupId>com.alibaba.cloud</groupId>

```



2. 修改 bootstrap.yml 配置文件，这里我以 order-service 应用为例，新增配置中心属性，如下：

```

1 server:
2   port: 6010
3 spring:
4   application:
5     name: order-service
6   cloud:
7     nacos:
8       discovery:
#必须配置ip地址
9         server-addr: localhost:8848
# 将自己的服务注册到注册中心
10      register-enabled: true
# 配置中心新增部分
11      config:
12        server-addr: localhost:8848
13        file-extension: yaml
14 profiles:
15   active: dev
# 新加属性，作为配置测试
16 member:
17   name: 订单服务中的配置名称
18   age: 18

```



### 配置文件加载的优先级（由高到低）

bootstrap.properties ->bootstrap.yml -> application.properties -> application.yml



3. OrderController 类中添加 @RefreshScope 注解

```

1 @RestController
2 @RequestMapping("order")
3 @RefreshScope //动态刷新配置，重要
4 public class OrderController {
5
6   //Spring常规取值
7   @Value("${merber.name}")

```

java





```

11     private Integer merberAge;
12
13     @Autowired
14     private RestTemplate restTemplate;
15
16     @GetMapping("get")
17     public String get(){
18         String result = restTemplate.getForObject("http://course-service/cour:
19         return result + ", merberName: " + merberName + ", merberAge: " + merb
20     }
21 }

```

#### 4. 给配置中心默认添加一个数据集 (Data Id)

NACOS 1.3.0

public | 服务注册 | 课程服务 | 订单服务 | 公共项目配置

配置管理 | public  
查询结果：共查找到 0 条满足要求的配置。

Data ID: 模糊查询请输入 Data ID | Group: 模糊查询请输入 Group | 搜索 | 高级查询 | 导出查询结果 | 导入配置

| Data ID | Group | 归属应用: | 操作 |
|---------|-------|-------|----|
| 没有数据    |       |       |    |

+  
新增一个配置





\* Data ID:  (highlighted by red box)

\* Group:

[更多高级选项](#)

描述: 订单服务开发环境的配置

配置格式:  TEXT  JSON  XML  YAML  HTML  Properties

\* 配置内容: [?](#) :

```
1: member:
  2:   name: 订单服务中的配置名称A
  3:   age: 188
```

## 新建配置



## 新建配置

Data ID: **order-service-dev.yaml**Group: **DEFAULT\_GROUP****确定**

NACOS 1.3.0

public | 服务注册 | 课程服务 | 订单服务 | 公共项目配置

配置管理 | public | 查询结果: 共查询到 1 条满足要求的配置。

|                          | Data Id                | Group         | 归属应用: | 操作   |
|--------------------------|------------------------|---------------|-------|--|
| <input type="checkbox"/> | order-service-dev.yaml | DEFAULT_GROUP |       | <a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a> |

[删除](#) [导出选中的配置](#) [克隆](#)

## 5. 配置中心命名规则



从 Nacos Spring Cloud 下，数据采集（Data ID）的配置元注释如下：

`${spring.cloud.nacos.config.prefix}-${spring.profiles.active}.${spring.cloud.nacos.cor  
extension}` , 通俗一点就是 前缀-环境-扩展名

- prefix: 默认为 `spring.application.name` 的值, 也可以通过配置项 `spring.cloud.nacos.config.prefix` 来配置

```

1 # 手动指定
2 spring:
3   cloud:
4     nacos:
5       config:
6         prefix: order-service-config
7
8 # 若不指定, 默认采用应用名的方案
9 spring:
10   application:
11     name: order-sevice #服务名

```



- active: 是配置开发环境的值, 一个程序不可能总是在开发环境, 可能需要切换到测试环境, 上线环境, 他们的配置文件都是不同的, 所以为了方便环境切换, 我们配置不同的开发环境文档。比如以前我们在 `application.yml` 中有配置dev、test、prod, 其中dev就是开发环境。。

\*\*注意:\*\* 当 `spring.profile.active` 为空时, 对应的连接符 - 也将不存在, dataId的拼接格式变成  `${prefix}.${file-extension}`

```

1 spring:
2   profiles:
3     active: dev #表示开发环境

```



- file-extension: 最后我们需要指定配置文件类型, 默认是 `properties` 。我们可以自己指定文件类型, 比如配置:

```

1 spring:
2   cloud:
3     nacos:

```





Nacos为我们提供了： TEXT、JSON、XML、YAML、HTML、Properties 几种类型

### ■ 最终配置

指定好配置文件类型，我们最终在配置中心新增配置文件就是： order-service-dev.yaml

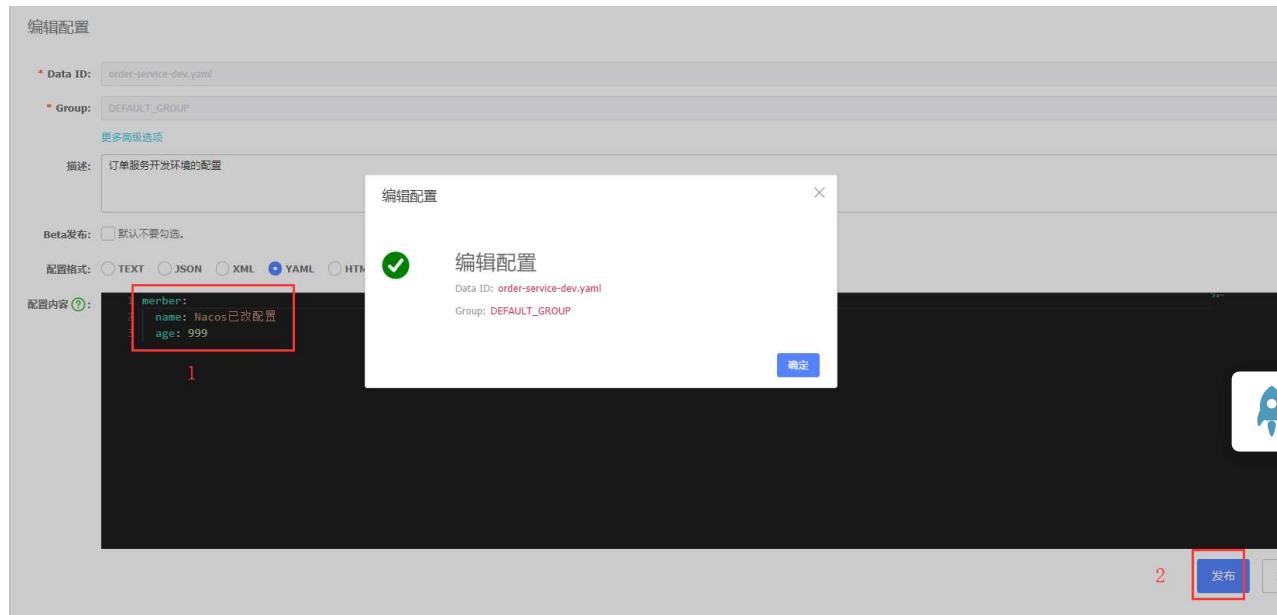
- Group

```
1 ${spring.cloud.nacos.config.group}
2 若未配置则取Nacos默认值: DEFAULT_GROUP
```

## 6. 启动服务验证



### Nacos中的配置





已经访问到课程服务, merberName: Nacos已改配置, merberAge: 999

在不重启 服务器的情况下, 配置已经发生改变

至此配置中心接入完毕, 课件代码详见: <https://gitee.com/appdoc/it235.git>, tag名:

**1.2.0**

| 标签名   | 描述                                     | 提交信息                       | 操作          |
|-------|--|----------------------------|-------------|
| 1.2.0 | nacos配置中心完善, 加入prefix-active.extension | 5cce949 · 2020-09-14 12:10 | 下载 创建发行版 删除 |
| 1.1.0 | nacos配置中心前置准备, value读取数据               | 697ffe4 · 2020-09-14 12:08 | 下载 创建发行版 删除 |
| 1.0.0 | nacos远程调用服务实现                          | 86aa401 · 2020-09-14 10:45 | 下载 创建发行版 删除 |

## 命名空间

Nacos 引入了命名空间( Namespace )的概念来进行多环境配置和服务的管理及隔离, Namespace 也是官方推荐的多环境支持方案。

例如, 你可能有 dev, test 和 prod 三个不同的环境, 那么使用一套 nacos 集群可以分别建以下三个不同的 namespace 。在没有明确指定命名空间配置的情况下, 默认新增的所有配置都在 public 空间, Nacos 控制台对不同的 Namespace 做了 Tab 栏分组展示, 如下图:



## 君哥的学习笔记



NACOS 1.3.2

public | prod | **dev** | test ① TAB栏被官方推荐用作环境隔离

配置管理 ② namespace的id

配置列表

历史版本

监听查询

服务管理

权限控制

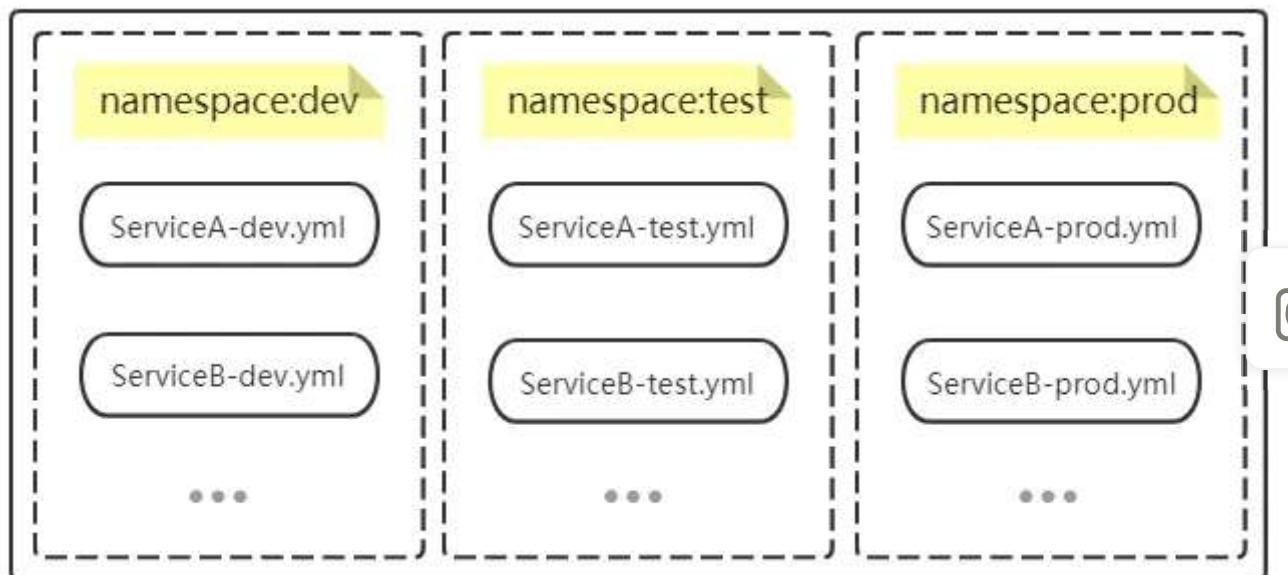
命名空间

集群管理

配置管理 | dev 1bd766d7-9941-4b9b-b94b-a1aa0a6d3305 | 搜索结果：共查询到 0 条满足要求的配置。

Data ID: 模糊查询请输入Data ID | Group: 模糊查询请输入Group | 查询 | 高级

| <input type="checkbox"/> | Data Id | Group | 操作 |
|--------------------------|---------|-------|----|
| 没有数据                     |         |       |    |



隔离开发，测试，生产环境可以添加命名空间，然后在 `bootstrap.yml` 配置文件添加命名空间的id即可切换到对应的命名空间，使用对应空间下的配置文件：

```

1 cloud:
2   nacos:
3     config:
4       namespace: a65a17de-e8f3-4d00-8d87-91549aaa0f02 #对应创建的命名空间的UUID

```



## 君哥的学习笔记



The screenshot shows the Nacos UI interface. On the left, there's a sidebar with options like '配置列表', '历史版本', '监听查询', '服务管理', '权限控制', and '命名空间'. The '命名空间' option is highlighted with a red box. The main area has a title '新建命名空间' (Create Namespace) and fields for '命名空间ID(不填则自动生成)' (Namespace ID (auto-generated if empty)) and '命名空间名:' (Namespace Name: dev). Below it is a '描述:' (Description: 开发环境) field. A blue '确定' (Confirm) button is at the bottom right.

也可以基于微服务来创建命名空间，用每一个微服务名来命名，达到隔离每一个微服务的目的，哪一个微服务需要配置直接去对应的微服务空间下找配置即可，使得项目更加结构化。

The screenshot shows the Nacos UI interface. The left sidebar has '命名空间' selected. The main area shows a table of namespaces:

| 命名空间名称       | 命名空间ID                         | 默认存在一个public保留命名空间 | 配置数 | 操作   |
|--------------|--------------------------------|--------------------|-----|--|
| public(保留空间) |                                |                    | 1   | <a href="#">详情</a> <a href="#">删除</a> <a href="#">编辑</a> |
| 服务注册         | all-register-service-namespace |                    | 0   | <a href="#">详情</a> <a href="#">删除</a> <a href="#">编辑</a> |
| 课程服务         | course-service                 |                    | 0   | <a href="#">详情</a> <a href="#">删除</a> <a href="#">编辑</a> |
| 订单服务         | order-service                  |                    | 1   | <a href="#">详情</a> <a href="#">删除</a> <a href="#">编辑</a> |
| 公共项目配置       | pub-config                     |                    | 0   | <a href="#">详情</a> <a href="#">删除</a> <a href="#">编辑</a> |

A note '命名空间主要用于环境隔离' (Namespaces are mainly used for environment isolation) is displayed below the table. A large circular refresh icon is on the right.

命名空间课件代码详见：<https://gitee.com/appdoc/it235.git>, tag名：1.3.0

The screenshot shows a GitHub repository page for '建君 / it235-nacos-parent'. The top bar includes links for 'Unwatch' (1), 'Star' (1), 'Fork' (0), and navigation tabs for '代码', 'Issues (0)', 'Pull Requests (0)', '附件 (0)', 'Wiki (0)', '统计' (highlighted in orange), 'DevOps', '服务', and '管理'.

The repository has 153 commits, 1 issue, and 0 pull requests. The '统计' tab shows a chart with the following data:

| 访问统计  | 仓库数据统计                                | 仓库网络图 | 发行版                             | 标签 | 提交 | + 新建标签             |
|-------|---------------------------------------|-------|---------------------------------|----|----|--------------------|
| 1.3.0 | nacos配置中心完善，加入命名空间区分环境                | 提交信息  | 153dfdf (1)<br>2020-09-14 12:55 |    |    | <a href="#">操作</a> |
| 1.2.0 | nacos配置中心完善，加入prefix-active.extension | 提交信息  | 5cce949 (1)<br>2020-09-14 12:10 |    |    | <a href="#">操作</a> |
| 1.1.0 | nacos配置中心前置准备，value读取数据               | 提交信息  | 697ffe4 (1)<br>2020-09-14 12:08 |    |    | <a href="#">操作</a> |
| 1.0.0 | nacos远程调用服务实现                         | 提交信息  | 86aa401 (1)<br>2020-09-14 10:45 |    |    | <a href="#">操作</a> |



## Data ID

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 package 的命名规则保证全局唯一性。此命名规则非强制。

大多数时候我们可能更加倾向于将不同的配置分开写到不同的配置文件中，将一个配置文件按功能拆分成不同的文件，然后在程序组合加载到一起组成一个完整的配置文件。比如我想把 ==DB类== 和 ==日志类== 的配置拆分开写到两个配置中，nacos也是支持这种一个配置中心多个配置集这种写法的。

1. 我们在nacos中新建两个 Data ID 分别是 db.yaml 和 log.yaml 的文件。
2. 在配置文件中分别加入部分配置内容
3. 更改 bootstrap.yml 中的nacos配置为

```

1   spring:
2     cloud:
3       nacos:
4         config:
5           extension-configs[0]:
6             data-id: db.yaml
7             group: DEFAULT_GROUP    # 默认为DEFAULT_GROUP
8             refresh: true    # 是否动态刷新，默认为false
9           extension-configs[1]:
10             data-id: log.yaml
11             group: DEFAULT_GROUP
12             refresh: true

```



为了更加清晰的在多个应用间配置共享的 Data Id，官方推荐使用 shared-configs，配置如下：

```

1   spring:
2     cloud:
3       nacos:
4         config:
5           shared-configs[0]:
6             data-id: db.yaml

```





```

10      data-id: log.yaml
11      group: DEFAULT_GROUP
12      refresh: true

```

#### 4. 思考：在这2个文件中出现相同配置，nacos如何选取？

多个 Data Id 同时配置时，他的优先级关系是 `spring.cloud.nacos.config.extension-configs[n].data-id` 其中 n 的值越大，优先级越高。

**==注意==：** `spring.cloud.nacos.config.extension-configs[n].data-id` 的值必须带文件扩展名，文件扩展名既可支持 `properties`，又可以支持 `yaml/yml`。此时 `spring.cloud.nacos.config.file-extension` 的配置对自定义扩展配置的 Data Id 文件扩展名没有影响。

#### 5. 不同方式配置加载优先级

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。



1. 通过 `spring.cloud.nacos.config.shared-configs[n].data-id` 支持多个共享 Data Id 配置
2. 通过 `spring.cloud.nacos.config.extension-configs[n].data-id` 的方式支持多个扩展 Data Id 的配置
3. 通过内部相关规则(`spring.cloud.nacos.config.prefix`、`spring.cloud.nacos.config.file-extension`、`spring.cloud.nacos.config.group`)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



#### 6. 配置集拆分实践

以下配置集可以拆分为：db、log、mybatis 三块



#### Group的使用

这是一个很灵活的配置项，并没有固定的规定，可以用作多环境、多模块、多版本之间区分配置，`namespace` 因其TAB栏的设定就是用作环境隔离。



## 君哥的学习笔记

GROUP;

### 常见的分组场景（维度）

不同的应用或组件使用了相同的配置类型，比如：

- database\_url 配置和 MQ\_topic 配置
- 对内配置组，对外配置组

## 总结

- dataid：配置文件的名字，相当于主键的作用；
- group: 不同的系统或微服务的配置文件可以放在一个组里。比如用户系统和订单系统的配置文件都可以放在同一个组中；
- namespace: 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；

## Nacos集群模式



集群模式适用于生产环境，ZK、Eureka等注册中心都能实现集群模式，集群模式的重心无外乎在节点间的相互通信和选举策略。

Nacos支持三种部署模式

- 单机模式 - 用于测试和单机试用。
- 集群模式 - 用于生产环境，确保高可用。
- 多集群模式 - 用于多数据中心场景。

1. 集群需要依赖mysql，单机可不必
2. 3个或3个以上Nacos节点才能构成集群。



在nacos的解压目录nacos/conf目录下，有配置文件cluster.conf，请每行配置成ip:port。  
(请配置3个或3个以上节点)

首先我们进入conf目录下，默认只有一个cluster.conf.example文件，我们需要自行复制一份，修改名称为cluster.conf





然后使用vi编辑器 打开cluster.config，按a/i/o 键可进入插入模式，输入以下内容

```
1 #ip:port
2 192.168.0.244:8848
3 192.168.0.245:8848
4 192.168.0.246:8848
```

然后按ESC键返回到命令模式，再按shift+:进入末行模式，输入wq敲回车（保存并退出）。

没有可视化界面的Linux我们不便操作，所以对mysql的操作，我在windows下使用navicat连接到之前在linux中安装的mysql再进行操作

注：需要对linux进行联网和mysql远程连接授权，再放行我们所需的端口（图方便的伙伴在练习的时候可直接关闭防火墙）

新建一个名为nacos\_config的数据库，在裤子执行下面的sql脚本文件。



SQL源文件如下

```
1 SQL脚本
```

执行后的navicat图

配置application.properties

```
1 # spring
2
3 server.contextPath=/nacos
4 server.servlet.contextPath=/nacos
5 server.port=8848
6
7 nacos.cmdb.dumpTaskInterval=3600
8 nacos.cmdb.eventTaskInterval=10
9 nacos.cmdb.labelTaskInterval=300
10 nacos.cmdb.loadDataAtStart=false
```





```

13
14 #management.endpoints.web.exposure.include=*
15
16 # metrics for elastic search
17 management.metrics.export.elasticsearch.enabled=false
18 #management.metrics.export.elasticsearch.host=http://localhost:9200
19
20 # metrics for influx
21 management.metrics.export.influx.enabled=false
22 #management.metrics.export.influx.db=springboot
23 #management.metrics.export.influx.uri=http://localhost:8086
24 #management.metrics.export.influx.auto-create-db=true
25 #management.metrics.export.influx.consistency=one
26 #management.metrics.export.influx.compressed=true
27
28 server.tomcat.accesslog.enabled=true
29 server.tomcat.accesslog.pattern=%h %l %u %t "%r" %s %b %D
30 # default current work dir
31 #server.tomcat.basedir=
32
33 db.num=1
34 db.url.0=jdbc:mysql://10.51.10.128:3306/nacos_config?characterEncoding=utf8&
35 db.user=root
36 db.password=123456
37
38 ## spring security config
39 ##### turn off security
40 #spring.security.enabled=false
41 #management.security=false
42 #security.basic.enabled=false
43 #nacos.security.ignore.urls=/*
44
45 nacos.security.ignore.urls=/,/**/*.*.css,/**/*.*.js,/**/*.*.html,/**/*.*.map,/**/*.*.svg,/

```



此时我们的一台linux上的nacos server就配置完成了，剩余两台linux的配置和上述一样，三台linux都连接同一mysql，cluster.conf内容一样

## 启动服务



三台linux机都输入以下命令启动nacos server服务

```
sh startup.sh
```



```
1 #=====
2 # JVM Configuration
3 =====
4 if [[ "${MODE}" == "standalone" ]]; then
5     JAVA_OPT="${JAVA_OPT} -Xms512m -Xmx512m -Xmn256m"
6     JAVA_OPT="${JAVA_OPT} -Dnacos.standalone=true"
7 else
8     JAVA_OPT="${JAVA_OPT} -server -Xms512m -Xmx512m -Xmn256m -XX:MetaspaceSize=6
9     JAVA_OPT="${JAVA_OPT} -XX:-OmitStackTraceInFastThrow -XX:+HeapDumpOnOutOfMem
10    JAVA_OPT="${JAVA_OPT} -XX:-UseLargePages"
11
12 fi
```

若出现内存不足的问题，可适当将-server后的jvm内存分配参数调小，系统默认的是

```
1 -server -Xms2g -Xmx2g -Xmn1g -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m
```



使用登入Nacos Server的可视化界面

在windows中用浏览器分别访问

- <http://10.51.10.128:8848/nacos/index.html>
- <http://10.51.10.129:8848/nacos/index.html>
- <http://10.51.10.130:8848/nacos/index.html>



### 三 君哥的学习笔记



#### 启动服务提供方和消费方

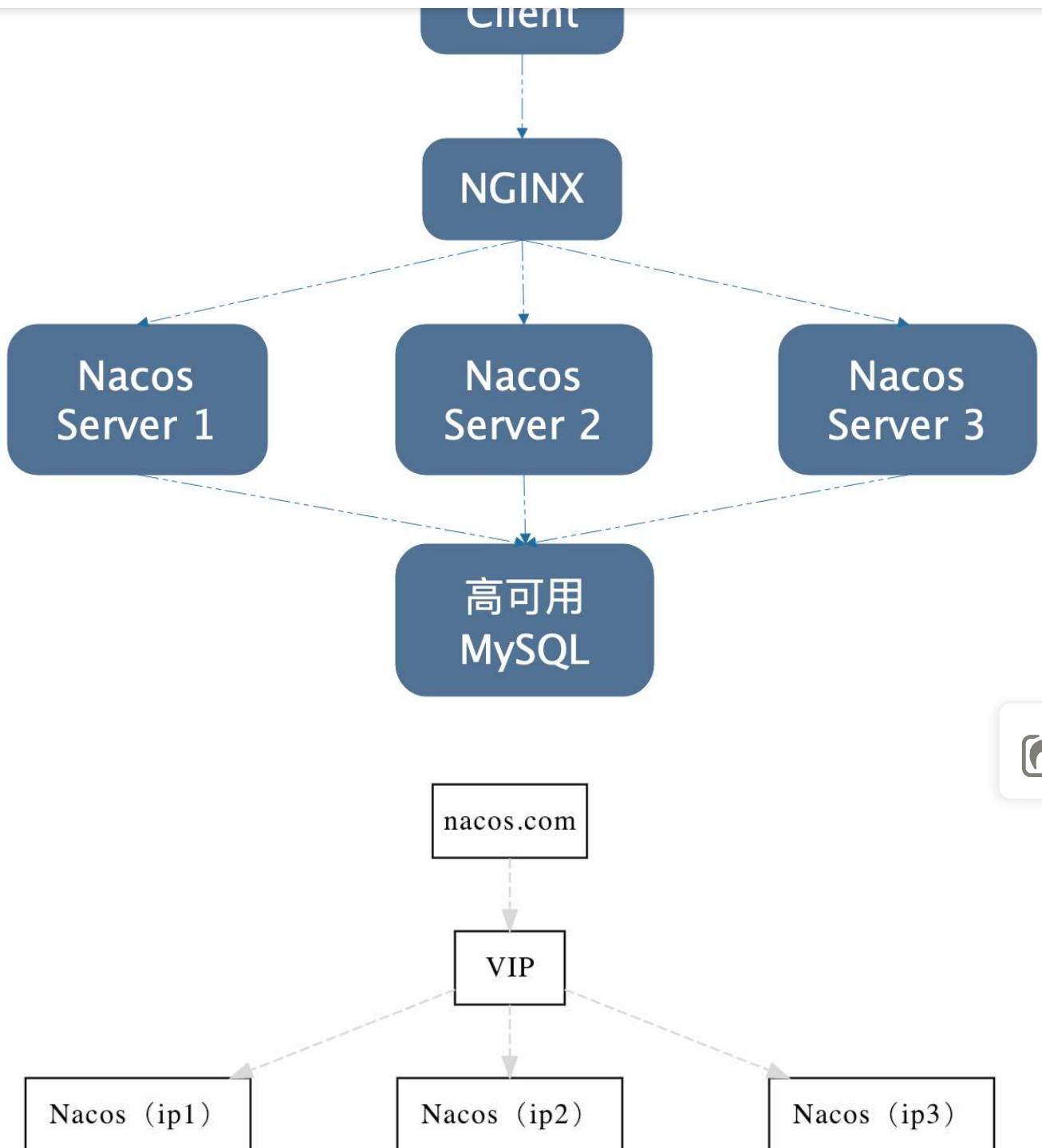
只需对上述两个项目的 `bootstrap.yml` 做局部修改:

```
1 spring.cloud.nacos.config.server-addr: 10.51.10.128:8848,10.51.10.129:8848,10.51.
```

#### 查看Nacos server的管理Web

Nacos集群模式架构图如下:





## Nacos多分支环境

Nacos提供对多分支环境的支持



## Nacos+Dubbo



## Nacos源码分析

### 附录：主流注册中心对比

#### Eureka AP

Spring Cloud Netflix 在设计 Eureka 时就紧遵AP原则（尽管现在2.0发布了，但是由于其闭源的原因，但是目前 Eureka 1.x 任然是比较活跃的）。

Eureka Server 也可以运行多个实例来构建集群，解决单点问题，但不同于 ZooKeeper 的选举 leader 的过程，Eureka Server 采用的是 Peer to Peer 对等通信。这是一种去中心化的架构，无 master/slave 之分，每一个 Peer 都是对等的。在这种架构风格中，节点通过彼此互相注册来提高可用性，每个节点需要添加一个或多个有效的 serviceUrl 指向其他节点。每个节点都可被视为其他节点的副本。

在集群环境中如果某台 Eureka Server 宕机，Eureka Client 的请求会自动切换到新的 Eureka Server 节点上，当宕机的服务器重新恢复后，Eureka 会再次将其纳入到服务器集管理之中。当节点开始接受客户端请求时，所有的操作都会在节点间进行复制（replicate To Peer）操作，将请求复制到该 Eureka Server 当前所知的其它所有节点中。

当一个新的 Eureka Server 节点启动后，会首先尝试从邻近节点获取所有注册列表信息，并完成初始化。Eureka Server 通过 getEurekaServiceUrls() 方法获取所有的节点，并且会通过心跳契约的方式定期更新。

默认情况下，如果 Eureka Server 在一定时间内没有接收到某个服务实例的心跳（默认周期为30秒），Eureka Server 将会注销该实例（默认为90秒，eureka.instance.lease-expiration-duration-in-seconds 进行自定义配置）。

当 Eureka Server 节点在短时间内丢失过多的心跳时，那么这个节点就会进入自我保护模式。

Eureka的集群中，只要有一台Eureka还在，就能保证注册服务可用（保证可用性），只不过查到的信息可能不是最新的（不保证强一致性）。除此之外，Eureka还有一种自我保护机制，如果在15分钟内超过85%的节点都没有正常的心跳，那么Eureka就认为客户端与注册中心出现了网络故障，此时会出现以下几种情况：

1. Eureka不再从注册表中移除因为长时间没有收到心跳而过期的服务；



### 三 君哥的学习笔记

3. 当网络稳定时，当前实例新注册的信息会被同步到其它节点中；

因此，Eureka可以很好的应对因网络故障导致部分节点失去联系的情况，而不会像zookeeper那样使得整个注册服务瘫痪。

#### Zookeeper CP

与 Eureka 有所不同，Apache Zookeeper 在设计时就紧遵CP原则，即任何时候对 Zookeeper 的访问请求能得到一致的数据结果，同时系统对网络分割具备容错性，但是 Zookeeper 不能保证每次服务请求都是可达的。

从 Zookeeper 的实际应用情况来看，在使用 Zookeeper 获取服务列表时，如果此时的 Zookeeper 集群中的 Leader 宕机了，该集群就要进行 Leader 的选举，又或者 Zookeeper 集群中半数以上服务器节点不可用（例如有三个节点，如果节点一检测到节点三挂了，节点二也检测到节点三挂了，那这个节点才算是真的挂了），那么将无法处理该请求。所以说，Zookeeper 不能保证服务可用性。

当然，在大多数分布式环境中，尤其是涉及到数据存储的场景，数据一致性应该是首先被保证的，这也是 Zookeeper 设计紧遵CP原则的另一个原因。

但是对于服务发现来说，情况就不太一样了，针对同一个服务，即使注册中心的不同节点保存的服务提供者信息不尽相同，也并不会造成灾难性的后果。

因为对于服务消费者来说，能消费才是最重要的，消费者虽然拿到可能不正确的服务实例信息后尝试消费一下，也要胜过因为无法获取实例信息而不去消费，导致系统异常变好（淘宝的双十一，京东的618就是紧遵AP的最好参照）。

当master节点因为网络故障与其他节点失去联系时，剩余节点会重新进行leader选举。问题在于，选举leader的时间太长，30~120s，而且选举期间整个zk集群都是不可用的，这就导致在选举期间注册服务瘫痪。

 在云部署环境下，因为网络问题使得zk集群失去master节点是大概率事件，虽然服务能最终恢复，但是漫长的选举事件导致注册长期不可用是不能容忍的。

#### Consul CP

Consul 是 HashiCorp 公司推出的开源工具，用于实现分布式系统的服务发现与配置。Consul 使用 Go 语言编写，因此具有天然可移植性（支持Linux、windows和Mac OS X）。



Consul 遵循CAP原理中的CP原则，保证了强一致性和分区容错性，且使用的是Raft算法，比zookeeper 使用的 Paxos 算法更加简单。虽然保证了强一致性，但是可用性就相应下降了，例如服务注册的时间会稍长一些，因为 Consul 的 raft 协议要求必须过半数的节点都写入成功才认为注册成功；在leader挂掉了之后，重新选举出leader之前会导致Consul 服务不可用。

## Nacos

Nacos是阿里开源的，Nacos 支持基于 DNS 和基于 RPC 的服务发现。在Spring Cloud中使用 Nacos，只需要先下载 Nacos 并启动 Nacos server，Nacos只需要简单的配置就可以完成服务的注册发现。

Nacos除了服务的注册发现之外，还支持动态配置服务。动态配置服务可以让您以中心化、外部化和动态化的方式管理所有环境的应用配置和服务配置。动态配置消除了配置变更时重新部署应用和服务的需要，让配置管理变得更加高效和敏捷。配置中心化管理让实现无状态服务变得更简单，让服务按需弹性扩展变得更容易。

一句话概括就是Nacos = Spring Cloud注册中心 + Spring Cloud配置中心。

### 附录：主流注册中心对比

| 对比项目  | Nacos                      | Eureka      | Consul            | Core |
|-------|----------------------------|-------------|-------------------|------|
| 一致性协议 | CP+AP                      | AP          | CP                | —    |
| 健康检查  | TCP/HTTP/MYSQL/Client Beat | Client Beat | TCP/HTTP/gRPC/Cmd | —    |

|        |                       |        |          |            |
|--------|-----------------------|--------|----------|------------|
| 负载均衡策略 | 权重/ metadata/Selector | Ribbon | Fabio    | RoundRobin |
| 雪崩保护   | 有                     | 有      | 无        | 无          |
| 自动注销实例 | 支持                    | 支持     | 不支持      | 不支持        |
| 访问协议   | HTTP/DNS              | HTTP   | HTTP/DNS | DNS        |
| 监听支持   | 支持                    | 支持     | 支持       | 不支持        |
| 多数据中心  | 支持                    | 支持     | 支持       | 不支持        |



## 君哥的学习笔记



|                   |    |     |     |     |
|-------------------|----|-----|-----|-----|
| SpringCloud<br>集成 | 支持 | 支持  | 支持  | 不支持 |
| Dubbo集成           | 支持 | 不支持 | 不支持 | 不支持 |
| K8S集成             | 支持 | 不支持 | 支持  | 支持  |

参考: <https://blog.csdn.net/fly910905/article/details/100023415>

参考: <https://developer.aliyun.com/article/698930>

2021-04-07: 6/15/2021, 5:17:04 PM

分布式事务之Seata →

昵称

邮箱

网址(<http://>)

来都来了，冒个泡再走呗...

表情 | 预览



M

回复

3 评论



Anonymous

Chrome 92.0.4515.107

Windows 10.0

2021-08-01



回复

牛皮



## 君哥的学习笔记



Anonymous

Chrome 90.0.4430.93

Mac OS 10.15.7

2021-05-04

回复

整挺好，细节满满



Anonymous

Chrome 89.0.4389.114

Windows 10.0

2021-04-09

回复

好的

Powered By Valine

v1.3.6

