36 Nacos 基础用法

更新时间：2019-08-01 09:31:35

> 立志是事业的大门，工作是登堂入室的旅程。
>
> ——巴斯德

上篇文章我们和大家介绍了 Spring Cloud Alibaba 的基本概念以及发展前景，使大家对 Spring Cloud Alibaba 有一个基本的认知，本文我们就来看看 Spring Cloud Alibaba 中 Nacos 的具体用法。

Nacos 简介

先来看看 Nacos 到底是什么！

基本介绍

Nacos 提供了动态服务发现、服务配置和服务管理功能，乍一看你可能会觉得这不就是 Eureka 或者 Consul 的翻版嘛！这句话也没错，不过 Nacos 的功能可不止这些！用官方的话来说，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。

关键特性

那么 Nacos 都包含那些基本特性呢？

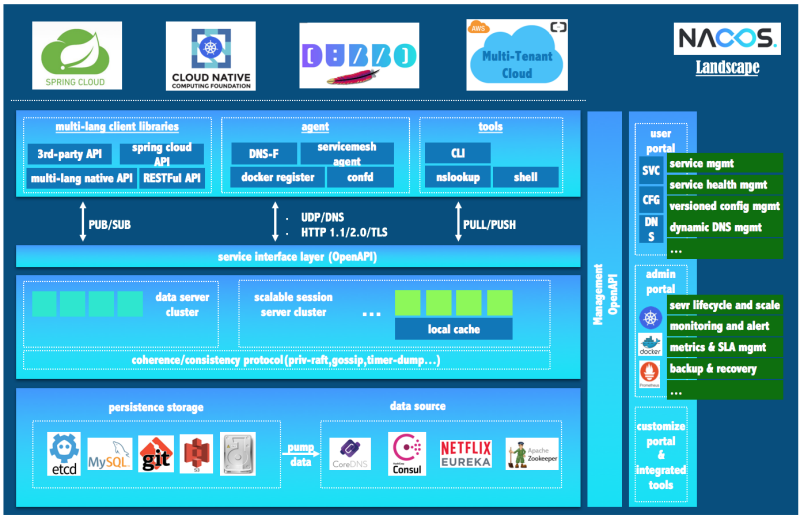
1. 服务发现和服务健康监测：Nacos 支持基于 DNS 和 RPC 的服务发现，并且提供了基于传输层和应用层的服务健康检查，自动剔除不健康的主机或者实例，Nacos 还提供了统一的健康检查仪表盘，使运维工程师可以快速发现问题。
2. 动态配置服务，这个功能有点类似于我们前面学过的 Spring Cloud Config 的作用，不过这里 Nacos 也提供了一样的功能。
3. 动态 DNS 服务，动态 DNS 服务支持权重路由，这样开发者可以更容易地实现中间层负载均衡、更灵活的路由策略、流量控制以及数据中心内网的简单DNS解析服务。动态 DNS 服务还能让开发者通过 DNS 协议实现服务发

现，这样可以避免耦合到厂商私有服务发现 API。

4. 服务及其元数据管理，Nacos 能让开发者从微服务平台建设的视角管理数据中心的所有服务及元数据，包括管理服务的描述、生命周期、服务的静态依赖分析、服务的健康状态、服务的流量管理、路由及安全策略、服务的 SLA 以及最首要的 metrics 统计数据。

生态

我们来看一张 Nacos 官方给出的 Nacos 生态图：



根据上图，我们可以看出，Nacos 可以无缝集成到主流的开源生态体系中，例如：

- Spring Cloud
- Apache Dubbo and Dubbo Mesh TODO
- Kubernetes and CNCF TODO

使用 Nacos 简化服务发现、配置管理、服务治理及管理的解决方案，让微服务的发现、管理、共享、组合更加容易。

基本用法

说了这么多官方的介绍，接下来我们就来简单实践下。

Nacos 安装

安装方式主要给大家介绍两种，一种就是直接安装，另一种则是通过 Docker 来安装。

直接安装

首先我们来看看 Nacos 的安装步骤，我们可以下载源码包编译安装也可以直接下载官方编译好的压缩包，这里我们采用第二种方式，下载地址：

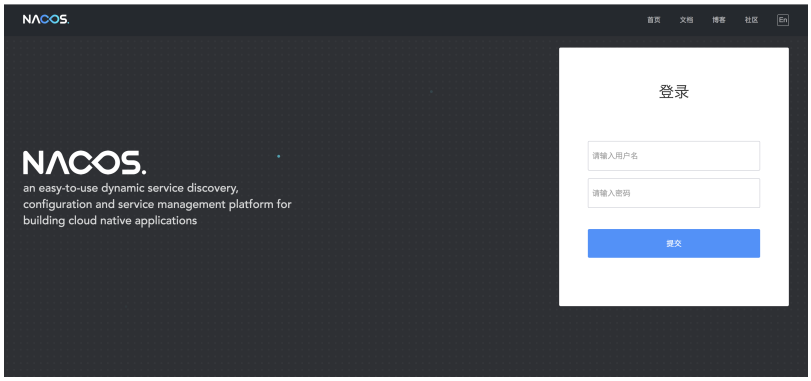
- Nacos 下载地址

下载完成后，解压进入 bin 目录下，如果是 Windows 系统，直接双击 startup.cmd 启动项目，如果是 Linux/Unix/Mac 系统，执行如下命令启动项目：

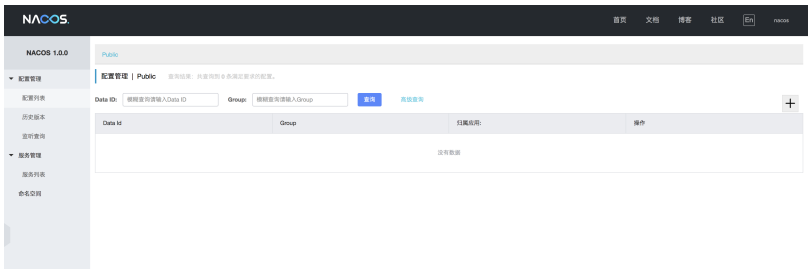
```
sh startup.sh -m standalone
```

启动命令中的 standalone 表示项目以单机模式启动，项目启动 Logo 如下：

项目启动成功之后，浏览器输入 http://localhost:8848/nacos ，我们可以看到如下管理台页面：



默认的登录用户名和密码都是 nacos ，登录成功后，就可以看到后台管理页面：



可以看到左侧菜单栏的两个选项**配置管理**和**服务管理**，这就是 Nacos 两大核心功能，我们一会来和大家仔细介绍。

Docker 安装

使用 Docker 安装，比较容易，首先我们先下载相关脚本：

- nacos docker 下载脚本

下载完成后，解压，进入解压目录中，可以以单机的形式启动 nacos ，也可以以集群方式启动 nacos ，单机版启动命令如下：

```
docker-compose -f example/standalone-derby.yaml up
```

以集群形式启动 nacos 命令如下：

```
docker-compose -f example/cluster-hostname.yaml up
```

我们通过观察下载的 nacos-docker 中的 example 目录下的 standalone-derby.yaml 和 cluster-hostname.yaml 脚本，发现两者都集成了 prometheus 和 grafana 进来。因此无论是集群版启动，还是单机版启动，启动成功之后，我们也都可以直接访问 Prometheus 和 Grafana ，关于 Prometheus 和 Grafana 的用法，读者可以参考本专栏前面的文章。

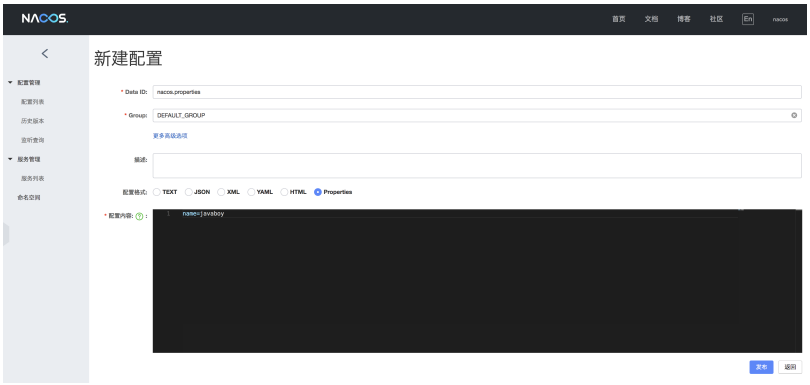
运行完 Docker 命令后，会有一个下载过程，稍等片刻，Docker 版的 Nacos 就启动成功了。

配置中心

将 Nacos 作为配置中心，作用类似于我们之前讲过的 Spring Cloud Config ，但是很明显这里要比 Spring Cloud Config 方便一些，因为配置中心+注册中心二合一，一个服务做两件事。而且都是可视化操作，我们就先来看一下配置中心要怎么使用。

Nacos 服务搭建

首先我们登录到 Nacos 后台管理页面，在 `配置管理->配置列表` 选项中，点击右边的添加按钮，先来配置一个基本的数据项，如下：



这里的配置我们主要配置了三个动心，分别是 Data ID、Group 以及配置内容。

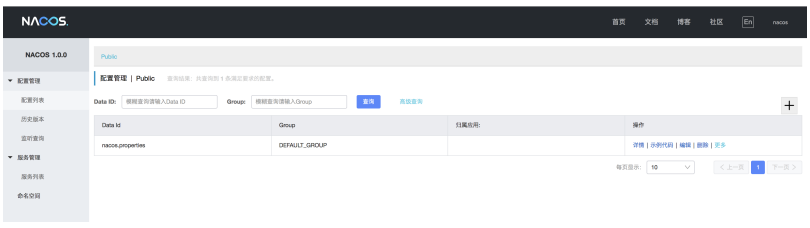
Data ID 的完成格式是：

```
${prefix}-${spring.profile.active}.${file-extension}
```

- `${prefix}` 默认为 spring.application.name 的值，当然开发者也可以直接在 Spring Boot 项目中通过 spring.cloud.nacos.config.prefix 属性来指定。
- `${spring.profile.active}` 表示当前项目所处的环境，即对应 Spring Boot 中的 spring.profile.active 属性，这是可选的，如果这个选项省略了，对应的 `-` 也将不复存在。
- `${file-extension}` 表示配置的数据格式。

经过这样的配置之后，每个服务连接上来之后，都能找到自己的配置了。

配置完成后，点击右下角的 发布 按钮，完成发布：



好了，这里就算先告一段落。

Spring Boot 服务创建

接下来，我们再来创建一个 Spring Boot 项目，创建时引入 Web 依赖即可，创建成功之后再手动加入 Nacos 依赖，由于 Spring Cloud Alibaba 目前尚未纳入到 Spring Cloud 的主版本库中，因此我们需要手动配置一下 Nacos 版本，最终的 pom.xml 文件如下：

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-alibaba-nacos-config</artifactId>
        <version>0.9.0.RELEASE</version>
    </dependency>
</dependencies>
```

创建成功之后，由于我们要加载远程配置，和前面的 Spring Cloud Config 一样，我们需要在 resources 目录下创建一个名为 bootstrap.properties 的文件，在该文件中配置远程配置中心的地址以及当前服务的名称：

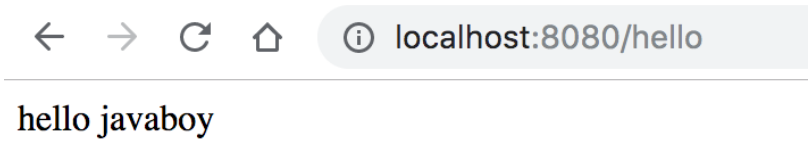
```
spring.application.name=nacos
spring.cloud.nacos.config.server-addr=127.0.0.1:8848
spring.cloud.nacos.config.file-extension=properties
```

这样就可以确保在 Spring Boot 项目启动时自动去 Nacos 上加载相关的配置文件，我们再创建一个新的 Controller，来展示加载到的数据，如下：

```
@RestController
@RefreshScope
public class HelloController {
    @Value(value = "${name}")
    String name;
    @GetMapping("/hello")
    public String hello() {
        return "hello " + name;
    }
}
```

这里我们使用到的都是 Spring 的原生注解，最大限度和 Nacos 解耦。其中，为了实现动态刷新配置文件，我们添加了 @RefreshScope 注解。

接下来我们启动 Nacos 项目，启动成功之后，我们访问 `/hello` 接口，结果如下：

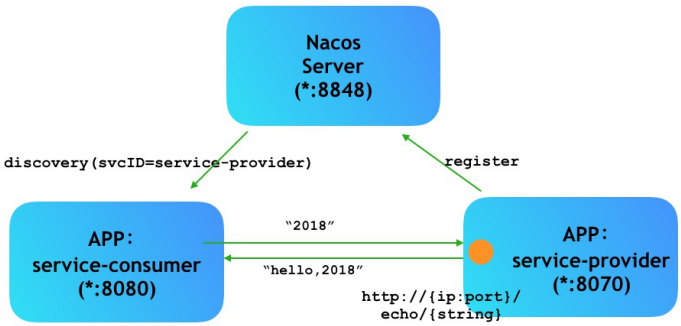


此时如果我们通过 Nacos 后台管理页面动态修改配置数据，不用重启 Spring Boot 项目，直接刷新页面，我们就可以看到数据已经更新。

大家有没有觉得很方便呢？还是比 Spring Cloud Config 要方便那么一丢丢！

注册中心

接下来我们再来看 Nacos 第二个使用场景，就是作为注册中心。扮演的角色相当于 Eureka。先来看 Nacos 官网的一张架构图：



和我们之前使用 Eureka 或者 Consul 的架构基本一致，首先我们搭建 Nacos Server 作为服务注册中心，然后 provider 注册到服务注册中心，consumer 再从服务注册中心获取到 provider 的信息，然后 consumer 就可以通过 RestTemplate 等工具调用 provider 了，就是这样一个简单的架构，我们来看下如何实现。

服务提供者

先来看服务提供者。

Nacos 本身不用变，还是上面的 Nacos ，我们首先在 Spring Boot 项目中引入服务发现依赖（这里就不创建新的 Spring Boot 项目了，继续在上文的基础上完成）：

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
    <version>0.9.0.RELEASE</version>
</dependency>
```

然后在 Spring Boot 的 application.properties 中添加一行配置，就可以将 Spring Boot 注册到 Eureka 上，如下：

```
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848
```

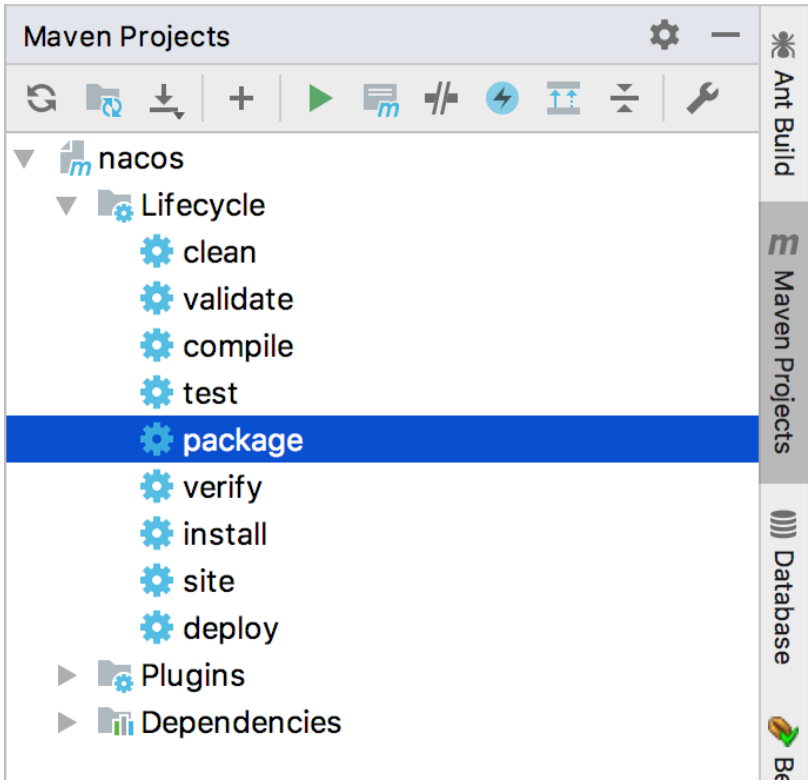
因为我们一会要将这个项目以集群的方式启动，在服务调用时，为了辨别是哪一个实例提供的服务，我将 HelloController 稍作修改：

```
@RestController
@RefreshScope
public class HelloController {
    @Value(value = "${name}")
    String name;
    @Value("${server.port}")
    Integer port;

    @GetMapping("/hello")
    public String hello() {
        return "hello " + name + ">>>" + port;
    }
}
```

这里多注入了一个字段 port ，通过 port 就可以区分到底是哪一个实例提供的服务了。

配置完成后，我们将 Spring Boot 项目打包，启动两个实例，方便我们后期测试：

首先点击右边的 `Maven Project -> Lifecycle -> package` 进行打包：

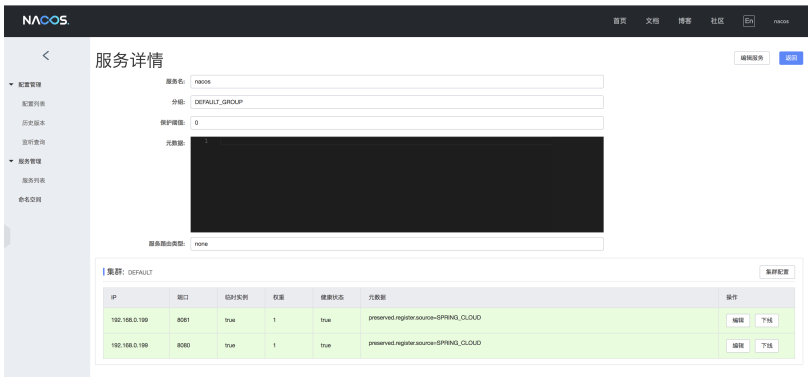


打包成功后，进入到打包目录中，执行如下命令，分别启动两个 Spring Boot 项目实例：

```
java -jar nacos-0.0.1-SNAPSHOT.jar --server.port=8080
java -jar nacos-0.0.1-SNAPSHOT.jar --server.port=8081
```

启动成功之后，我们再去观察 Nacos 管理页面，可以看到服务已经成功注册上来了：

点击详情，可以查看详细信息：



如此之后，我们的服务提供者就算是配置成功了。

服务消费者

接下来，我们需要配置服务消费者。

此时，我们创建一个新的 Spring Boot 工程，创建时引入 Web 依赖，创建成功后，引入 Nacos 服务发现依赖，最终的 pom.xml 文件如下图：

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
        <version>0.9.0.RELEASE</version>
    </dependency>
</dependencies>
```

然后在 application.properties 文件中配置一下 Nacos 地址，如下：

```
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848
server.port=8082
```

配置完成后，我们就可以像以前一样，使用 RestTemplate 去调用另一个微服务提供的服务了。

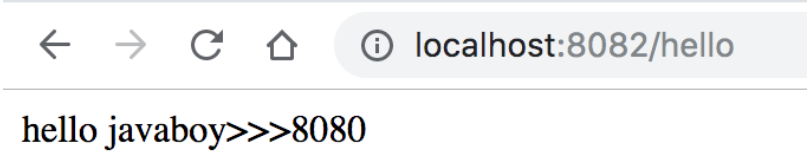
我们首先在配置类中提供一个 RestTemplate 实例，并且添加负载均衡功能：

```
@SpringBootApplication
public class NacosClientApplication {

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

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

然后在 Controller 中调用相关的服务，如下：

```
@RestController
public class HelloController {
    @Autowired
    RestTemplate restTemplate;
    @GetMapping("/hello")
    public String hello() {
        return restTemplate.getForObject("http://nacos/hello", String.class);
    }
}
```

在这里我们注入 RestTemplate ，然后通过调用 RestTemplate 中的 getForObject 方法去调用另外一个服务。



刷新一下，就换一个实例提供服务，即 8080 和 8081 来回切换显示。但是我们的所有用法都和前面学过的一样，并没有变化，这就是 Spring Cloud 的魅力之一，对调用 API 做了统一规范，无论是 Netflix 还是 Spring Cloud Alibaba ，都只是其中一个实现而已。

总结

本文主要向大家介绍了 Spring Cloud Alibaba 中 Nacos 组件的基本用法，通过本文，希望大家不仅掌握 Nacos 的基本用法，更重要的是可以加深对前面学习知识点的理解，前面我们和大家聊的负载均衡注解 @LoadBalanced 以及 RestTemplate 工具，在 Nacos 中都可以无缝集成进来使用。

参考资料：

1. Nacos 官方文档

35 Spring Cloud Alibaba 现状

37 Sentinel 基本用法