Spring Cloud Consul 服务注册中心

历史修订

本次修订日期: 2023-02-26	下次修订日期:

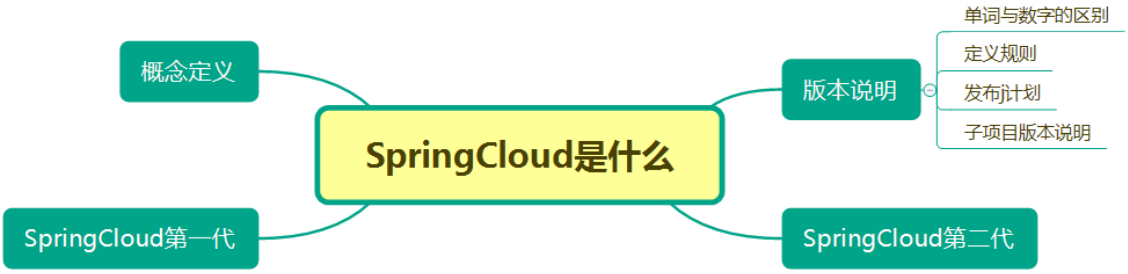
修订编号	修订日期	变更描述	说明
V0.1	2023-02-26	起草	李宗在
V0.2	2023-02-28	验证	李宗在
V0.3			

1. 微服务介绍

1.1 技术介绍

- spring boot
- ubuntu
- mysql
- docker
- mybatis/mybatis-plus
- github
- consul注册中心
- swagger接口文档管理

1.2 学习目



2. Consul服务注册中心

Netflix Eureka 2.X https://github.com/Netflix/eureka/wiki 官方宣告停止开发，但其实对国内的用户影响甚小，一方面国内大都使用的是 Eureka 1.X 系列，并且官方也在积极维护 1.X https://github.com/Netflix/eureka/releases。

> The existing open source work on eureka 2.0 is discontinued. The code base and artifacts that were released as part of the existing repository of work on the 2.x branch is considered use at your own risk.
>
> Eureka 1.x is a core part of Netflix's service discovery system and is still an active project.

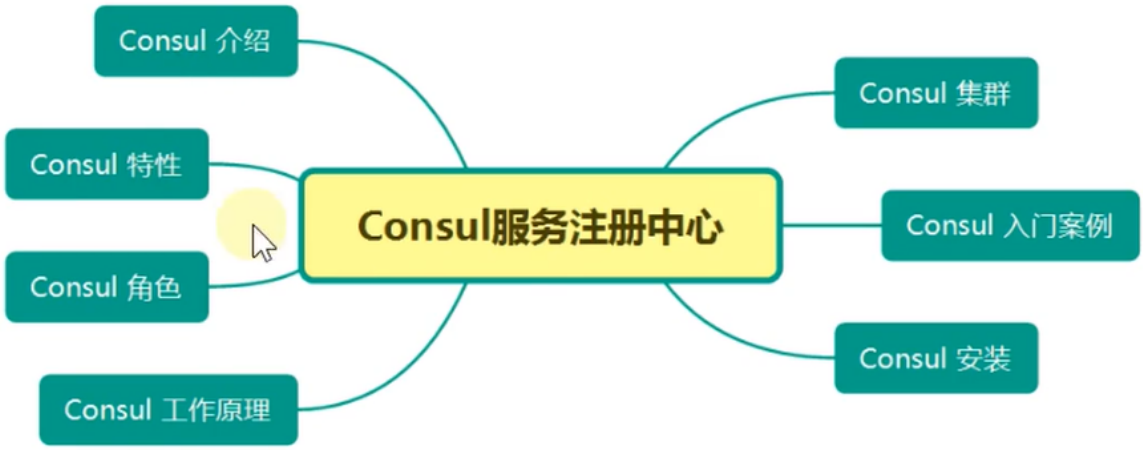
翻译：

> 有关 eureka 2.0 的现有开源工作已停止。在 2.x 分支上作为现有工作资料库的一部分发布的代码库和工件被视为使用后果自负。
>
> Eureka 1.x 是 Netflix 服务发现系统的核心部分，仍然是一个活跃的项目。

> 虽然 Eureka，Hystrix 等不再继续开发或维护，但是目前来说不影响使用，不管怎么说感谢开源，向 Netflix 公司的开源致敬。

另一方面 Spring Cloud 支持很多服务发现的软件，Eureka 只是其中之一，下面是 Spring Cloud 支持的服务发现软件以及特性对比。

2.1 学习目的



2.2 常见注册中心

- Netflix Eureka
- Alibaba Nacos
- HashiCorp Consul
- Apache ZooKeeper
- CoreOS ETCD
- CNCF CoreDNS

特性	Eureka	Nacos	Consul	Zookeeper
CAP	AP	CP + AP	CP	CP
健康检查	Client Beat	TCP/HTTP/MYSQL/Client Beat	TCP/HTTP/gRPC/Cmd	Keep Alive
雪崩保护	有	有	无	无
自动注销实例	支持	支持	不支持	支持
访问协议	HTTP	HTTP/DNS	HTTP/DNS	TCP
监听支持	支持	支持	支持	支持
多数据中心	支持	支持	支持	不支持
跨注册中心同步	不支持	支持	支持	不支持
SpringCloud集成	支持	支持	支持	支持

2.3 Consul介绍

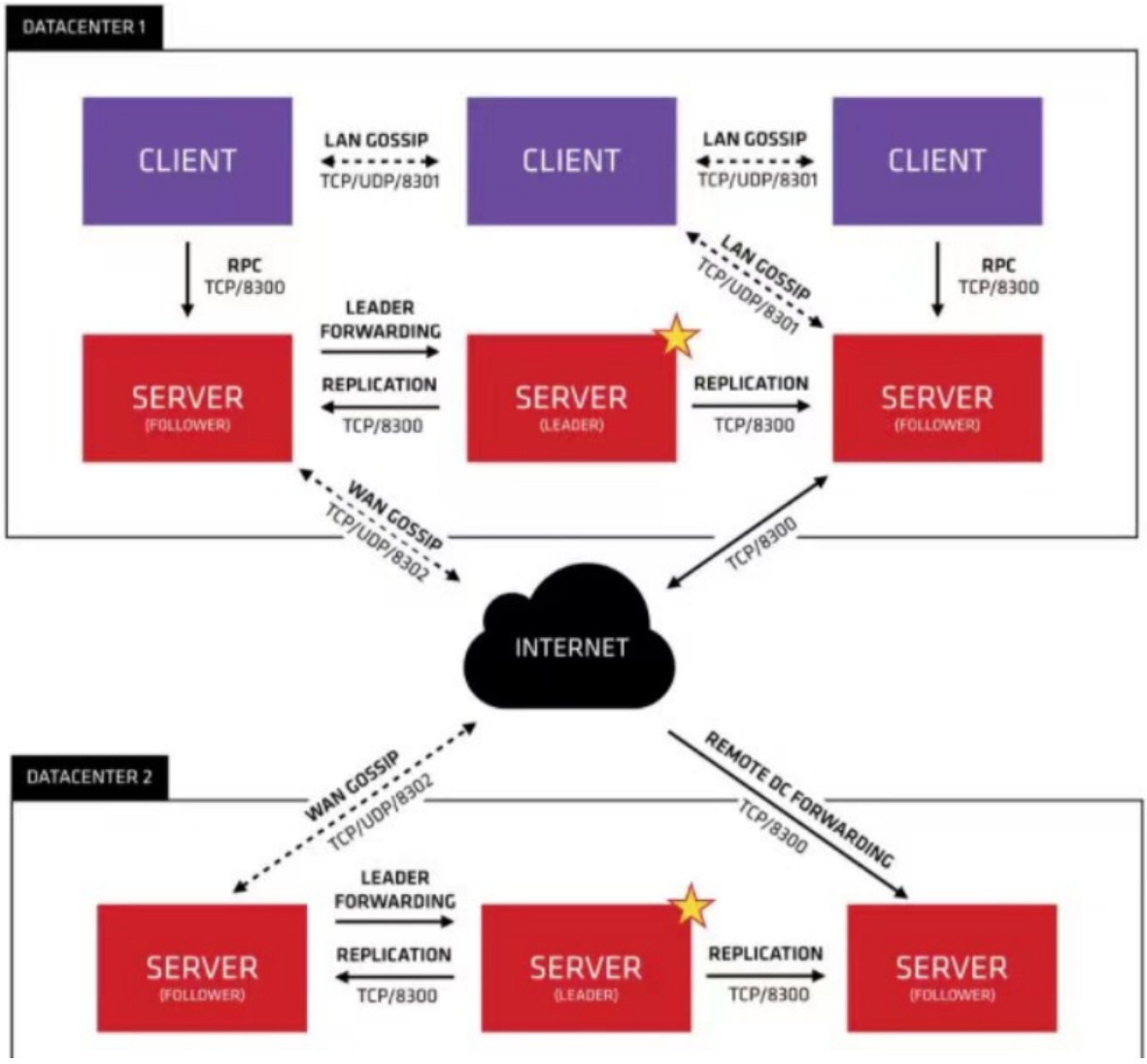
Consul 是 HashiCorp 公司推出的开源工具，用于实现分布式系统的服务发现与配置。与其它分布式服务注册与发现的方案，Consul 的方案更“一站式”，内置了服务注册与发现框架、分布一致性协议实现、健康检查、Key/Value 存储、多数据中心方案，不再需要依赖其它工具（比如 ZooKeeper 等），使用起来也较为简单。

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）；安装包仅包含一个可执行文件，方便部署，与 Docker 等轻量级容器可无缝配合。

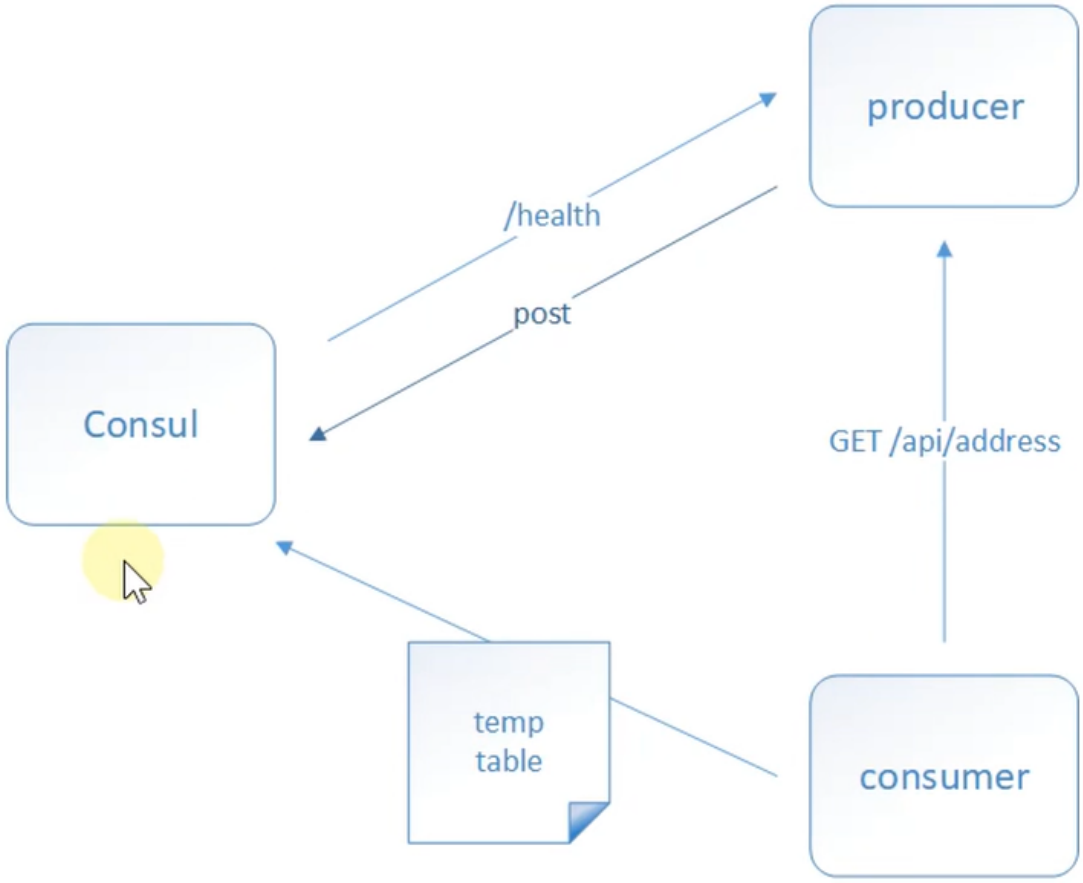
2.4 Consul特性

- Raft 算法
- 服务发现
- 健康检查
- Key/Value 存储
- 多数据中心
- 支持 http 和 dns 协议接口
- 官方提供 web 管理界面

2.5 Consul 角色



2.6 Consul工作原理

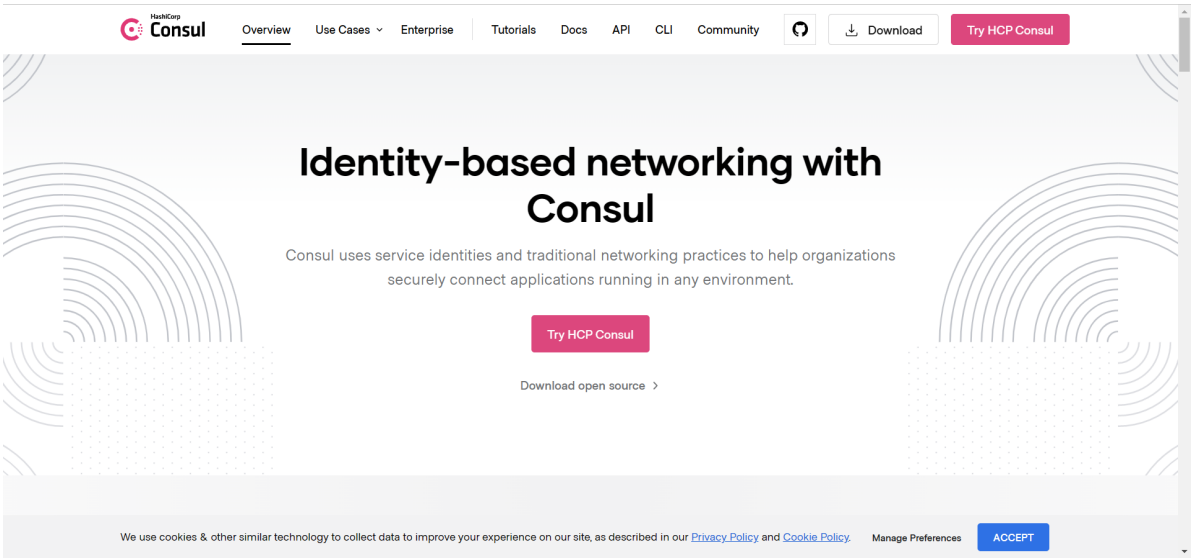


2.7 Consul安装

Eureka其实就是一个Servlet程序，运行在Servlet容器中。Consul是使用go语言编写的第三方工具需要单独安装使用。

2.7.1 下载

访问官方 https://consul.io



2.7.2 单节点

容器化部署consul服务注册中心

```
$ sudo docker run --name consul01 -d -p 8500:8500 -p 8300:8300 -p 8301:8301 -p
8302:8302 -p 8600:8600 consul:1.6.1 agent -server -bootstrap-expect=1  -ui -
bind=0.0.0.0 -client=0.0.0.0
```

apt安装包consul服务注册中心

```
$ sudo wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor | sudo tee
/usr/share/keyrings/hashicorp-archive-keyring.gpg

$ sudo echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee
/etc/apt/sources.list.d/hashicorp.list

$ sudo apt update && sudo apt install consul
```

2.8 入门级案例

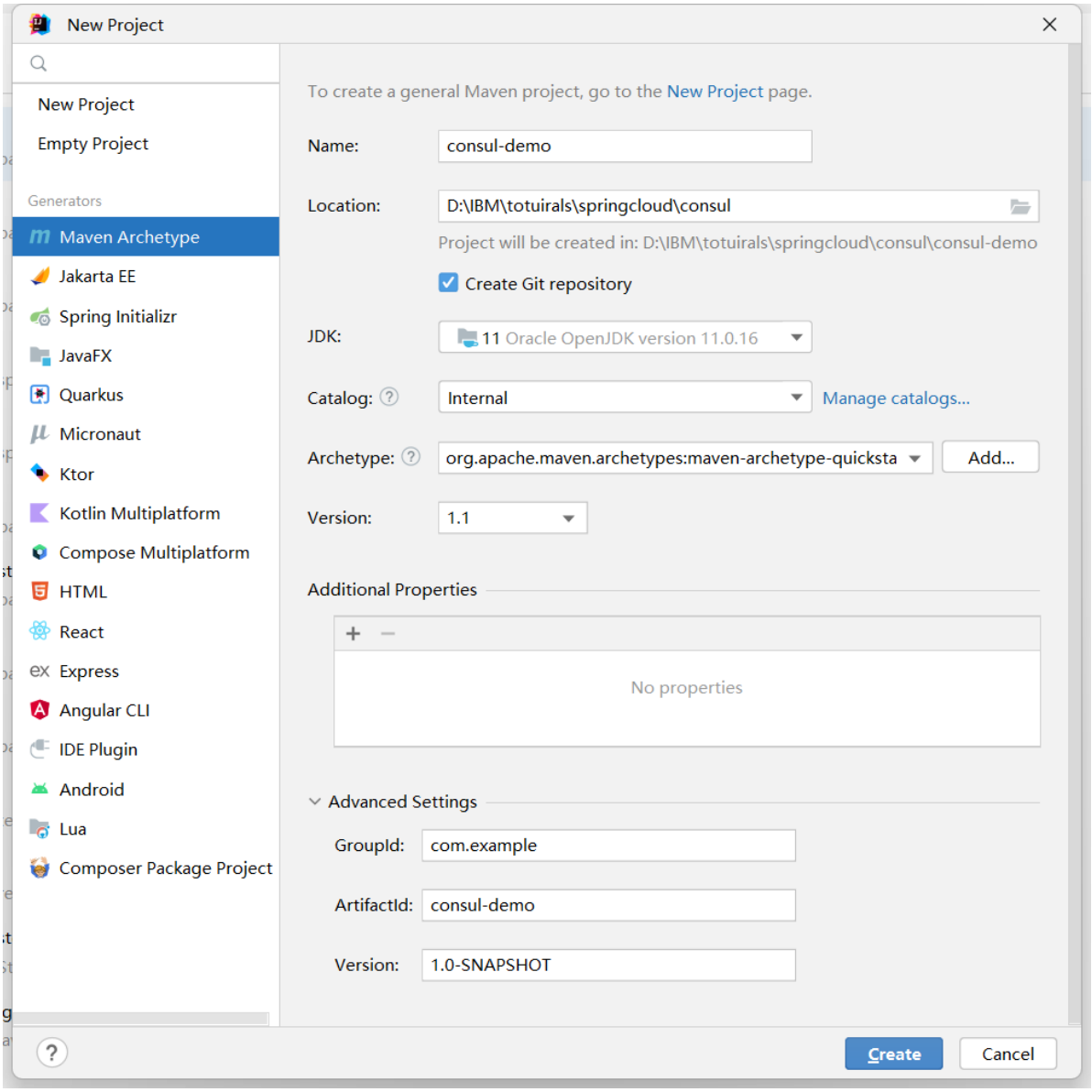
SpringCloud与SpringBoot的版本对应

官网版本对应地址： https://start.spring.io/actuator/info

https://blog.csdn.net/u012272367/article/details/125094938

SpringCloud版本	SpringBoot版本
2022.0.0-M2	Spring Boot >=3.0.0-M2 and <3.1.0-M1
2022.0.0-M1	Spring Boot >=3.0.0-M1 and <3.0.0-M2
2021.0.3	Spring Boot >=2.6.1 and <3.0.0-M1
2021.0.0-RC1	Spring Boot >=2.6.0-RC1 and <2.6.1
2021.0.0-M3	Spring Boot >=2.6.0-M3 and <2.6.0-RC1
2021.0.0-M1	Spring Boot >=2.6.0-M1 and <2.6.0-M3
2020.0.5	Spring Boot >=2.4.0.M1 and <2.6.0-M1
Hoxton.SR12	Spring Boot >=2.2.0.RELEASE and <2.4.0.M1
Hoxton.BUILD-SNAPSHOT	Spring Boot >=2.2.0.BUILD-SNAPSHOT
Hoxton.M2	Spring Boot >=2.2.0.M4 and <=2.2.0.M5
Greenwich.BUILD-SNAPSHO	Spring Boot >=2.1.9.BUILD-SNAPSHOT and <2.2.0.M4
Greenwich.SR2	Spring Boot >=2.1.0.RELEASE and <2.1.9.BUILD-SNAPSHOT
Greenwich.M1	Spring Boot >=2.1.0.M3 and <2.1.0.RELEASE

2.8.1 创建父工程

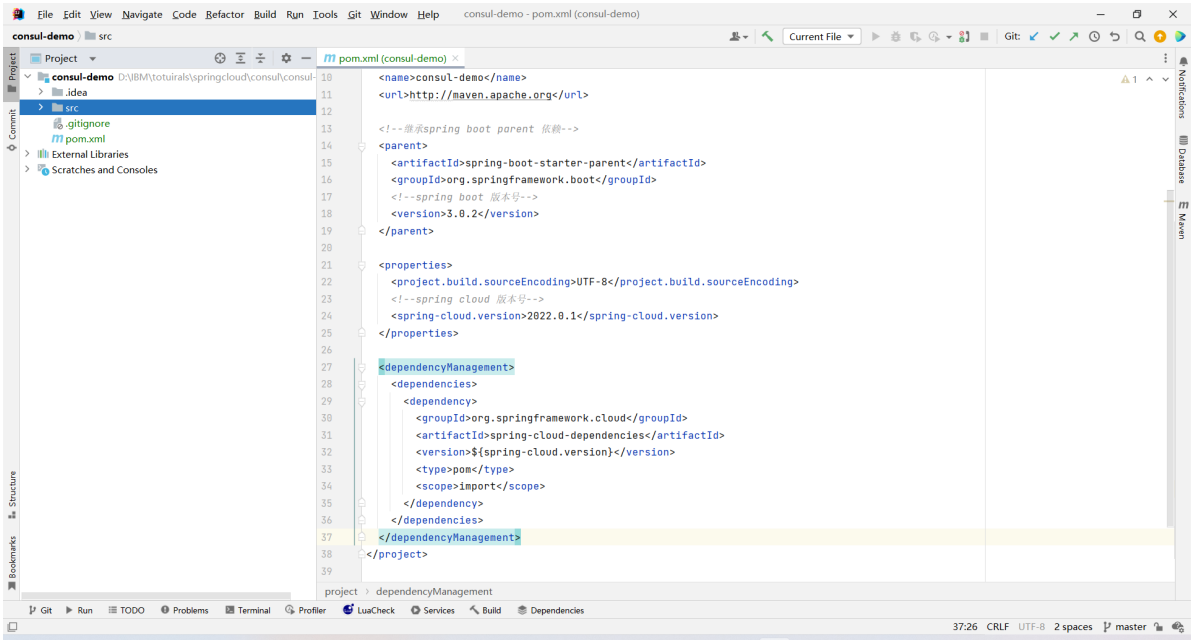


2.8.2 添加依赖

```
<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.example</groupId>
  <artifactId>consul-center</artifactId>
  <version>1.0-SNAPSHOT</version>
  <modules>
    <module>service-provider</module>
    <module>code-generator</module>
  </modules>
  <packaging>pom</packaging>

  <name>consul-center</name>
  <url>http://maven.apache.org</url>

  <!--继承spring boot parent 依赖-->
  <parent>
```

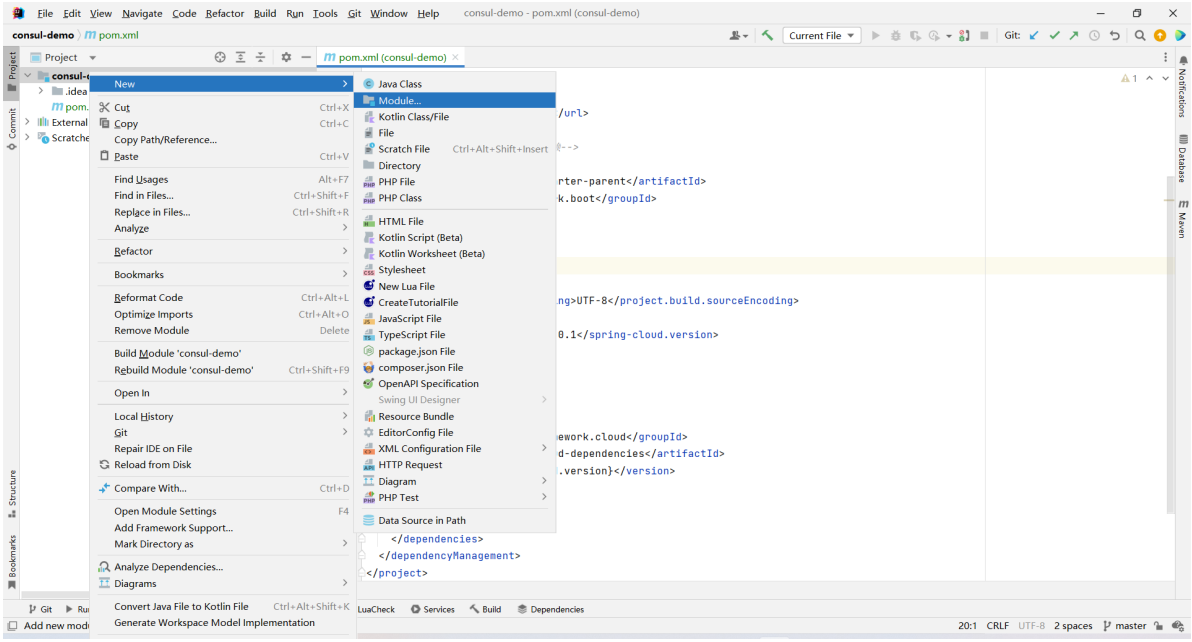
```
    <artifactId>spring-boot-starter-parent</artifactId>
    <groupId>org.springframework.boot</groupId>
    <!--spring boot 版本号-->
    <version>2.6.1</version>
  </parent>

  <properties>
    <java.version>11</java.version>
    <!--spring cloud 版本号-->
    <spring-cloud.version>2022.0.1</spring-cloud.version>
  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-dependencies</artifactId>
        <version>${spring-cloud.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>

</project>
```

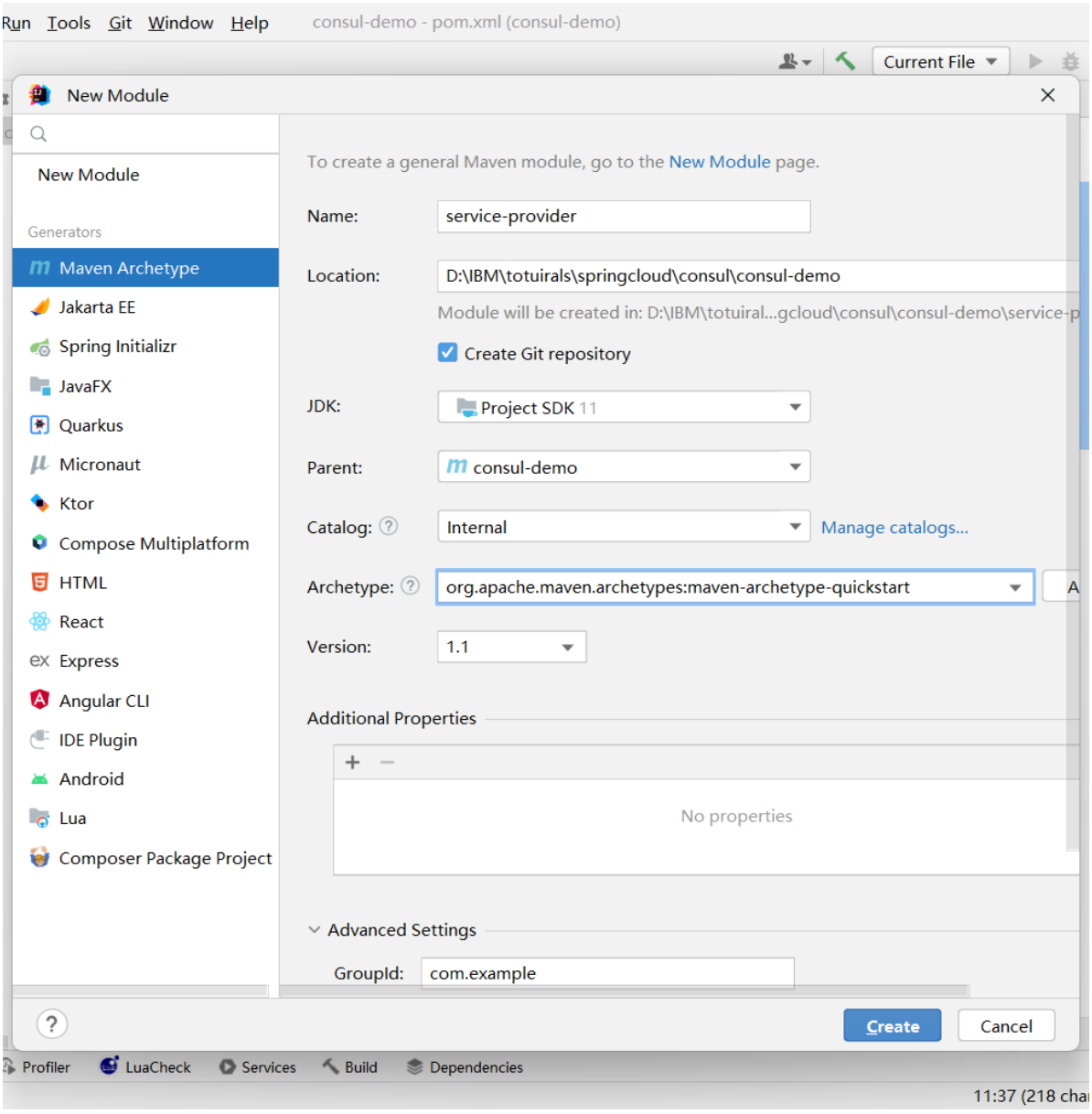
删除src和gitingnore目录和文件



2.8.3 创建子项目

2.8.3.1 服务提供者


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
consul-demo - pom.xml (consul-demo)
New
Java Class
Module...
Kotlin Class/File
File
Scratch File
Directory
PHP File
PHP Class
HTML File
Kotlin Script (Beta)
Kotlin Worksheet (Beta)
Stylesheet
New Lua File
CreateTutorialFile
JavaScript File
TypeScript File
package.json File
composer.json File
OpenAPI Specification
Swing UI Designer
Resource Bundle
EditorConfig File
XML Configuration File
HTTP Request
Diagram
PHP Test
Data Source in Path
Cut
Copy
Copy Path/Reference...
Paste
Find Usages
Find in Files...
Replace in Files...
Analyze
Refactor
Bookmarks
Reformat Code
Optimize Imports
Remove Module
Build Module 'consul-demo'
Rebuild Module 'consul-demo'
Open In
Local History
Git
Repair IDE on File
Reload from Disk
Compare With...
Open Module Settings
Add Framework Support...
Mark Directory as
Analyze Dependencies...
Diagrams
Convert Java File to Kotlin File
Generate Workspace Model Implementation


Run Tools Git Window Help
consul-demo - pom.xml (consul-demo)
Current File
New Module
New Module
Generators
Maven Archetype
Jakarta EE
Spring Initializr
JavaFX
Quarkus
Micronaut
Ktor
Compose Multiplatform
HTML
React
Express
Angular CLI
IDE Plugin
Android
Lua
Composer Package Project
To create a general Maven module, go to the New Module page.
Name: service-provider
Location: D:\IBM\totuirals\springcloud\consul\consul-demo
Module will be created in: D:\IBM\totuiral...gcloud\consul\consul-demo\service-p
Create Git repository
JDK: Project SDK 11
Parent: consul-demo
Catalog: Internal
Manage catalogs...
Archetype: org.apache.maven.archetypes:maven-archetype-quickstart
Version: 1.1
Additional Properties
No properties
Advanced Settings
GroupId: com.example
Create
Cancel
Profiler
LuaCheck
Services
Build
Dependencies
11:37 (218 cha

2.8.3.2 添加依赖

```
<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">
  <parent>
    <artifactId>consul-center</artifactId>
    <groupId>com.example</groupId>
    <version>1.0-SNAPSHOT</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>

  <artifactId>service-provider</artifactId>
  <packaging>jar</packaging>

  <name>service-provider</name>
  <url>http://maven.apache.org</url>

  <properties>
    <java.version>1.8</java.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <!--spring cloud consul 依赖-->
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-consul-discovery</artifactId>
    </dependency>

    <!-- spring cloud consul config 配置中心依赖 -->
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-consul-config</artifactId>
    </dependency>

    <!--spring boot actuator 依赖-->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>

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

    <!--lombok 依赖-->
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
    </dependency>

    <!--junit 依赖-->
    <dependency>
      <groupId>junit</groupId>
```

```
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <!--mysql 依赖-->
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.18</version>
    </dependency>

    <!--mybatis-plus 依赖-->
    <dependency>
      <groupId>com.baomidou</groupId>
      <artifactId>mybatis-plus-boot-starter</artifactId>
      <version>3.4.0</version>
    </dependency>

    <!--swagger2 依赖-->
    <dependency>
      <groupId>io.springfox</groupId>
      <artifactId>springfox-swagger2</artifactId>
      <version>2.7.0</version>
    </dependency>

    <!--swagger 第三方ui依赖-->
    <dependency>
      <groupId>com.github.xiaoymin</groupId>
      <artifactId>swagger-bootstrap-ui</artifactId>
      <version>1.9.6</version>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

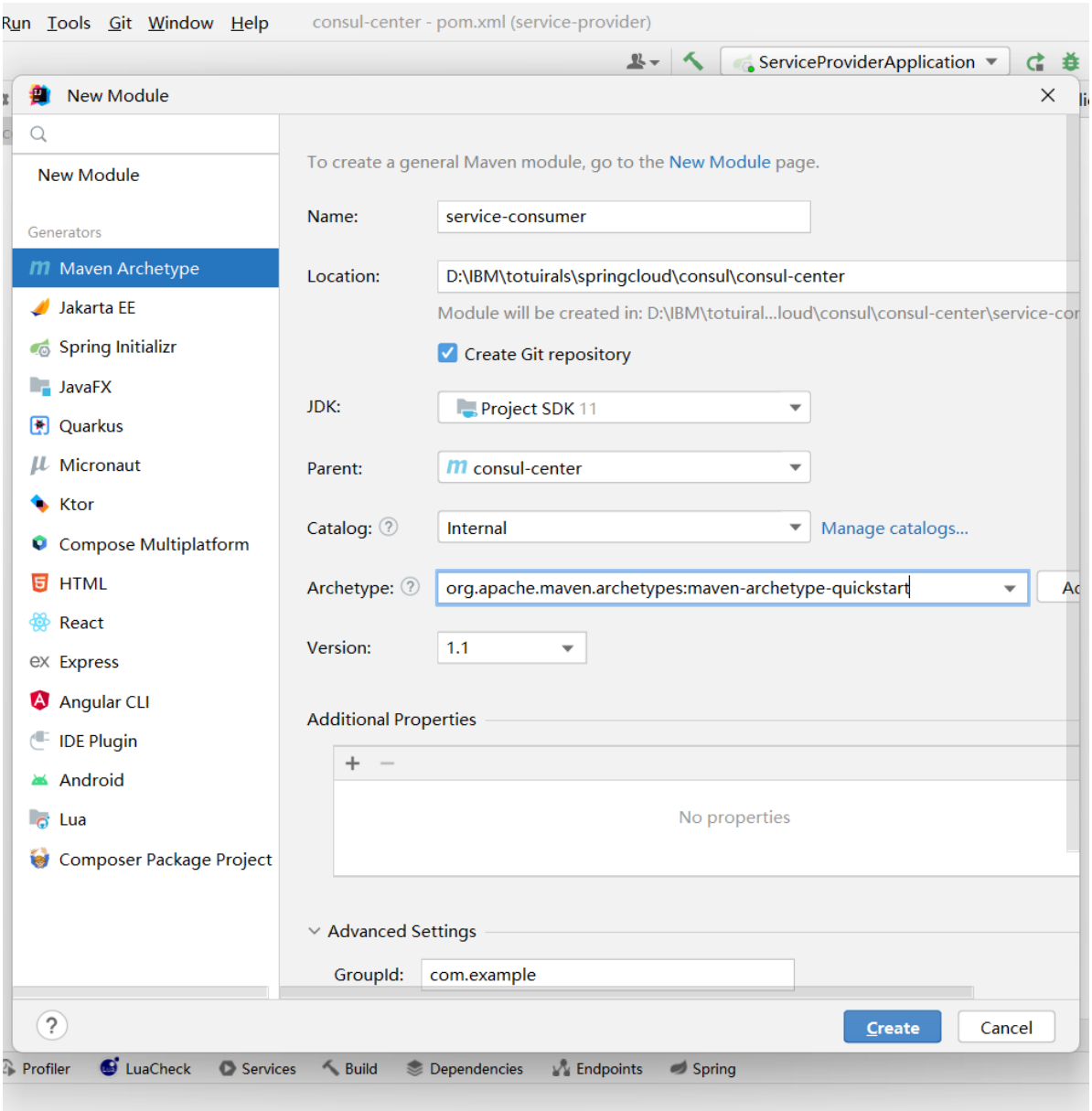
2.8.3.3 配置文件

```yaml
server:
  port: 8080  # 端口号

spring:
  application:
    name: service-provider # 应用名称
  main:
    allow-circular-references: true
  mvc:
    pathmatch:
      matching-strategy: ANT_PATH_MATCHER
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: "jdbc:mysql://localhost:3306/micro_service?
useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai"
    username: root
    password: password

  # 配置consul注册中心
  cloud:
    consul:
      # 服务注册中心的访问地址和端口号
      host: 192.168.126.65
      port: 8500
      # 服务提供者信息
      discovery:
        register: true                                  # 是否需要注册
        instance-id: ${spring.application.name}-01      # 注册实例ID(必须唯一)
        service-name: ${spring.application.name}        # 服务名称
        port: ${server.port}                            # 服务端口
        prefer-agent-address: true                      # 是否使用IP地址注册
        ip-address: ${spring.cloud.client.ip-address} # 服务请求IP
```

2.8.3.4 商品列表

忽略

2.8.3.5 服务消费者



忽略步骤，具体可以看github

consul集群安装

Setup Consul Cluster on Ubuntu 20.04|18.04|16.04 & Debian 10/9 | ComputingForGeeks

https://computingforgeeks.com/how-to-install-consul-cluster-18-04-lts/

https://computingforgeeks.com/how-to-install-consul-cluster-18-04-lts/

https://www.how2shout.com/linux/how-to-install-consul-by-hashicorp-on-ubuntu-20-04-lts/

2.10 consul集群安装

Step 1: Download and install Consul on Ubuntu 18.04

Install Consul on all the three nodes. Check the latest release of Consul from the releases page. Here we will download and install **v1.8.4**

```
$ sudo export VER="1.8.4"
$ sudo wget https://releases.hashicorp.com/consul/${VER}/consul_${VER}_linux_amd64.zip
```

Extract the file

```
$ sudo apt update
$ sudo apt install unzip
$ unzip consul_${VER}_linux_amd64.zip
```

Move extracted `consul` binary to `/usr/local/bin` directory

```
$ sudo chmod +x consul
$ sudo mv consul /usr/local/bin/
```

To print consul help page, use `--help` option

```
$ sudo consul --help
```

To verify Consul is properly installed, run `consul -v` on your system.

```
$ sudo consul  version
```

Step 2: Bootstrap and start Consul Cluster

Since we have three nodes to use for our Consul cluster setup, we will bootstrap one by one. If you want to do a single node Consul setup, you can skip the other two.

Create a `consul` system user/group.

```
$ sudo groupadd --system consul
$ sudo useradd -s /sbin/nologin --system -g consul consul
```

Create consul data directory and set ownership to `consul` user

```
$ sudo mkdir -p /var/lib/consul
$ sudo chown -R consul:consul /var/lib/consul
$ sudo chmod -R 775 /var/lib/consul
```

Create Consul configurations directory

```
$ sudo mkdir /etc/consul.d
$ sudo chown -R consul:consul /etc/consul.d
```

Setup DNS or edit `/etc/hosts` file to configure hostnames for all servers (set on all nodes),Replace `example.com` with your actual domain name.

```
$ sudo vim /etc/hosts

192.168.126.65 consul01.lab.example.com consul01
192.168.126.67 consul02.lab.example.com consul02
192.168.126.68 consul03.lab.example.com consul03
```

For a three node cluster:

Bootstrap Consul first node – consul01

Create a systemd service file `/etc/systemd/system/consul.service` and add:

```
$ sudo vim /etc/systemd/system/consul.service
```

```
[Unit]
Description=Consul Service Discovery Agent
Documentation=https://www.consul.io/
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
User=consul
Group=consul
ExecStart=/usr/local/bin/consul agent \
    -node=<strong>consul01</strong> \
    -config-dir=/etc/consul.d

ExecReload=/bin/kill -HUP $MAINPID
KillSignal=SIGINT
TimeoutStopSec=5
Restart=on-failure
SyslogIdentifier=consul

[Install]
WantedBy=multi-user.target
```

Generate Consul secret

```
$ sudo consul keygen
qzxCTInDpwGJI65KfRAbijhJdHQBjlHkcmxCTbyxkmc
```

Then create a json configuration file for the node in `/etc/consul.d/config.json`

```
$ sudo vim /etc/consul.d/config.json
```

```
{
    "advertise_addr": "192.168.126.65",
    "bind_addr": "192.168.126.65",
    "bootstrap_expect": 3,
    "client_addr": "0.0.0.0",
    "datacenter": "DC1",
    "data_dir": "/var/lib/consul",
    "domain": "consul",
    "enable_script_checks": true,
    "dns_config": {
        "enable_truncate": true,
        "only_passing": true
    },
    "enable_syslog": true,
    "encrypt": "qzxCTInDpwGJI65KfRAbijhJdHQBjlHkcmxCTbyxkmc",
    "leave_on_terminate": true,
    "log_level": "INFO",
    "rejoin_after_leave": true,
    "retry_join": [
     "consul01",
     "consul02",
     "consul03"
    ],
    "server": true,
    "start_join": [
        "consul01",
        "consul02",
        "consul03"
    ],
    "ui": true
}
```

Replace all occurrences of `192.168.126.65` with the correct IP address of this node and value of **encrypt** with your generated secret. You need to have DNS or hosts file configured for the short DNS names (**consul01, consul02,** and **consul03**) to work.

Consul Node 2

Consul systemd service:

```
# cat   /etc/systemd/system/consul.service
[Unit]
Description=Consul Service Discovery Agent
Documentation=https://www.consul.io/
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
User=consul
Group=consul
ExecStart=/usr/local/bin/consul agent \
    -node=consul01 \
    -config-dir=/etc/consul.d

ExecReload=/bin/kill -HUP $MAINPID
KillSignal=SIGINT
TimeoutStopSec=5
Restart=on-failure
SyslogIdentifier=consul

[Install]
WantedBy=multi-user.target2
```

Consul json configuration file

```
# cat /etc/consul.d/config.json
{
    "advertise_addr": "192.168.126.67",
    "bind_addr": "192.168.126.67",
    "bootstrap_expect": 3,
    "client_addr": "0.0.0.0",
    "datacenter": "DC1",
    "data_dir": "/var/lib/consul",
    "domain": "consul",
    "enable_script_checks": true,
    "dns_config": {
        "enable_truncate": true,
        "only_passing": true
    },
    "enable_syslog": true,
    "encrypt": "bnRHLmJ6TeLomirgEOWP2g==",
    "leave_on_terminate": true,
    "log_level": "INFO",
    "rejoin_after_leave": true,
    "retry_join": [
     "consul01",
     "consul02",
     "consul03"
    ],
    "server": true,
    "start_join": [
```

```
        "consul01",
        "consul02",
        "consul03"
    ],
    "ui": true
}
```

Consul Node 3

Consul systemd service:

```
# cat   /etc/systemd/system/consul.service
[Unit]
Description=Consul Service Discovery Agent
Documentation=https://www.consul.io/
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
User=consul
Group=consul
ExecStart=/usr/local/bin/consul agent \
    -node=consul03 \
    -config-dir=/etc/consul.d

ExecReload=/bin/kill -HUP $MAINPID
KillSignal=SIGINT
TimeoutStopSec=5
Restart=on-failure
SyslogIdentifier=consul

[Install]
WantedBy=multi-user.target2
```

Consul json configuration file

```
# cat /etc/consul.d/config.json
{
    "advertise_addr": "192.168.126.68",
    "bind_addr": "192.168.126.68",
    "bootstrap_expect": 3,
    "client_addr": "0.0.0.0",
    "datacenter": "DC1",
    "data_dir": "/var/lib/consul",
    "domain": "consul",
    "enable_script_checks": true,
    "dns_config": {
        "enable_truncate": true,
        "only_passing": true
    },
    "enable_syslog": true,
    "encrypt": "bnRHLmJ6TeLomirgEOWP2g==",
```

```
    "leave_on_terminate": true,
    "log_level": "INFO",
    "rejoin_after_leave": true,
    "retry_join": [
     "consul01",
     "consul02",
     "consul03"
    ],
    "server": true,
    "start_join": [
        "consul01",
        "consul02",
        "consul03"
    ],
    "ui": true
}
```

Start consul service on all nodes

```
$ sudo systemctl start consul
$ sudo systemctl enable consul
```

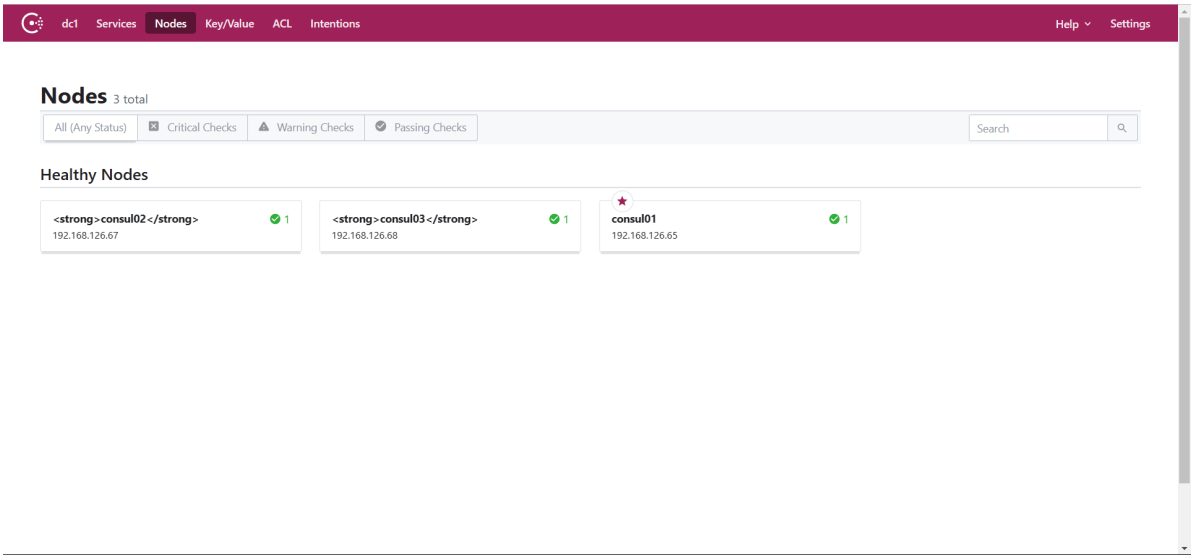
Check cluster members

```
root@consul03:~#
root@consul03:~#
root@consul03:~# consul members
Node                  Address             Status  Type    Build  Protocol  DC   Segment
<strong>consul02</strong>  192.168.126.67:8301  alive   server  1.8.4  2         dc1  <all>
<strong>consul03</strong>  192.168.126.68:8301  alive   server  1.8.4  2         dc1  <all>
consul01              192.168.126.65:8301  alive   server  1.8.4  2         dc1  <all>
root@consul03:~# consul members -detailed
Node                  Address             Status  Tags
<strong>consul02</strong>  192.168.126.67:8301  alive   acls=0,build=1.8.4:12b16df3,dc=dc1,expect=3,ft_fs=1,id=b6146f82-d4e6-a87c-f122-fa168a87be0d,port=8300,raft_vsn=3,role=consul,seg
ment=<all>,vsn=2,vsn_max=3,vsn_min=2,wan_join_port=8302
<strong>consul03</strong>  192.168.126.68:8301  alive   acls=0,build=1.8.4:12b16df3,dc=dc1,expect=3,ft_fs=1,id=2ae6f0cc-a601-8243-79b6-341f0756c890,port=8300,raft_vsn=3,role=consul,seg
ment=<all>,vsn=2,vsn_max=3,vsn_min=2,wan_join_port=8302
consul01              192.168.126.65:8301  alive   acls=0,build=1.8.4:12b16df3,dc=dc1,expect=3,ft_fs=1,id=63e630c5-bcfc-f4da-c73e-d564a53025f2,port=8300,raft_vsn=3,role=consul,seg
ment=<all>,vsn=2,vsn_max=3,vsn_min=2,wan_join_port=8302
root@consul03:~#
```

The output shows the address, health state, role in the cluster, and consul version of each node in the cluster. You can obtain additional metadata by providing the `-detailed` flag.

Access Consul UI

You can access the Consul in-built Web interface using the URL `http://<consul-IP>:8500/ui` ,for example **http://192.168.126.65:8500/ui**



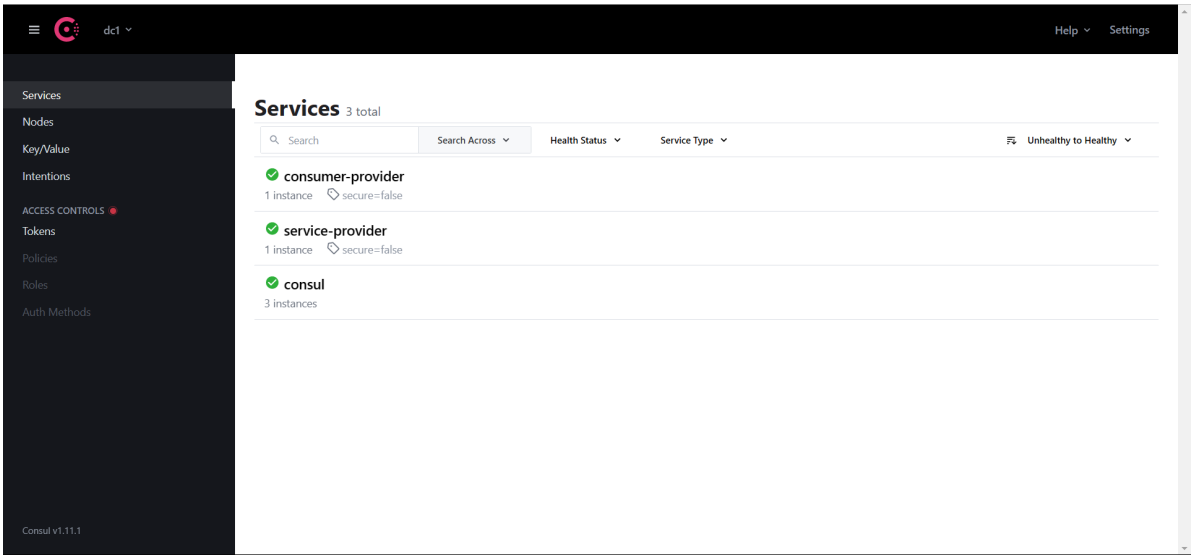
Step 3: Docker Consul Cluster

基于docker安装consul集群搭建server+client - 夜空中的萤火虫 - 博客园 (cnblogs.com)

https://www.cnblogs.com/zhouganqing/p/14207291.html

```
-- 创建并启动第一个节点
$ sudo docker run -d -p 8500:8500 --name=ConsulServer-A consul agent -server -ui -
node=Server-A -bootstrap-expect=3 -client=0.0.0.0

--查看第一个节点IP
$ sudo docker inspect -f '{{.NetworkSettings.IPAddress}}' ConsulServer-A

--创建并启动第二个节点
$ sudo docker run -d -p 8501:8500 --name=ConsulServer-B consul agent -server -ui -
node=Server-B -bootstrap-expect=3 -client=0.0.0.0 -join=172.17.0.2

--创建并启动第三个节点
$ sudo docker run -d -p 8502:8500 --name=ConsulServer-c consul agent -server -ui -
node=Server-c -bootstrap-expect=3 -client=0.0.0.0 -join=172.17.0.2

--第二种方式：创建变量存放IP
$ sudo JOIN_IP="$(docker inspect -f '{{.NetworkSettings.IPAddress}}' ConsulServer-A)"
--创建并启动第二个节点，第三个节点
$ sudo docker run -d -p 8502:8500 --name=ConsulServer-c consul agent -server -ui -
node=Server-c -bootstrap-expect=3 -client=0.0.0.0 -join=$JOIN_IP

--创建第四个节点(client模式)
$ sudo docker run -d -p 8503:8500 --name=ConsulClient-a consul agent -ui -
node=ConsulClient-a -client=0.0.0.0 -join=172.17.0.2去掉 -server 即为client模式
```

http://192.168.126.69:8500/ui/dc1/services

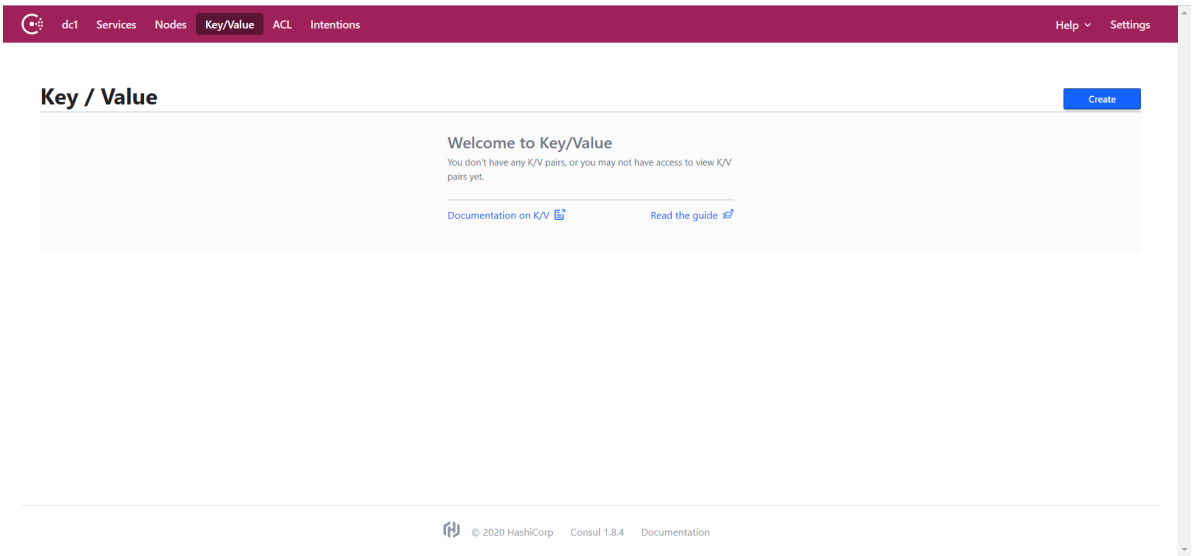


3. 初始化配置

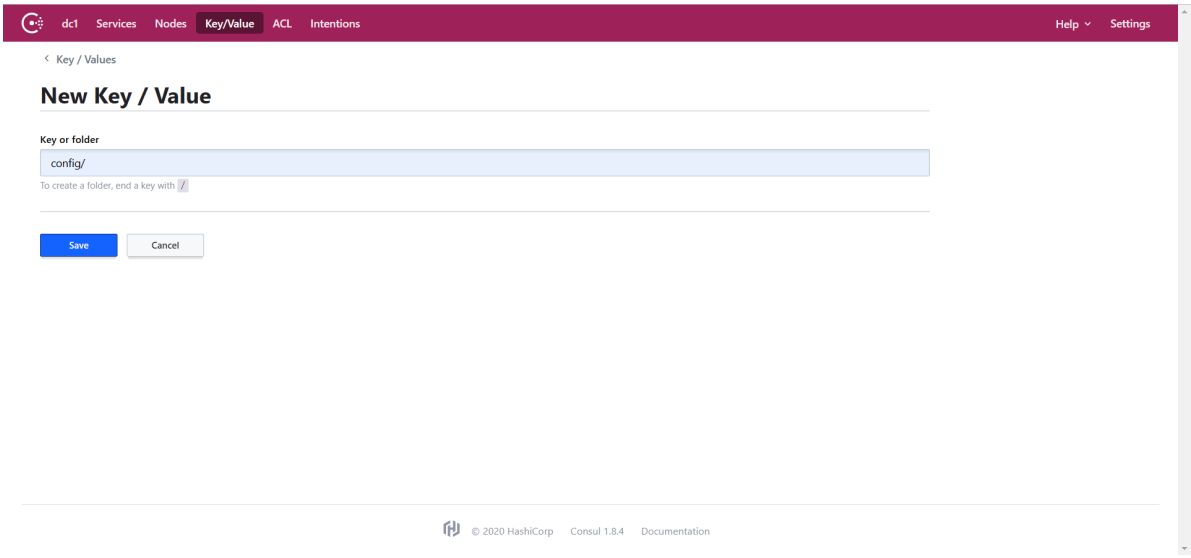
3.1 创建基本目录

使用 Consul 作为配置中心，第一步我们先创建目录，把配置信息存储至 Consul。

点击菜单 `Key/Value` 再点击 `Create` 按钮。

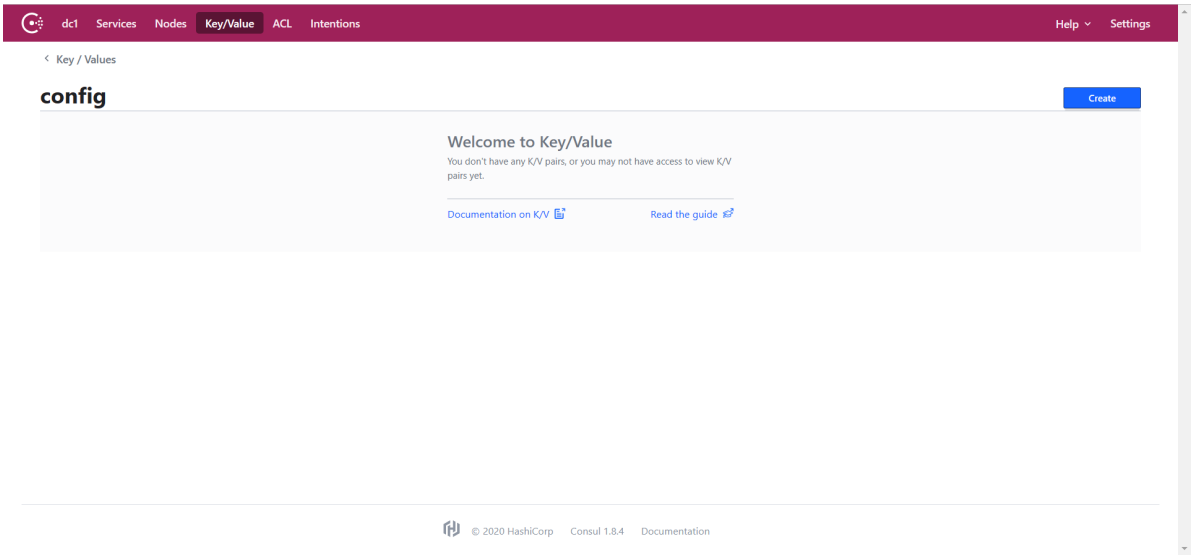


创建 `config/` 基本目录，可以理解为配置文件所在的最外层文件夹。

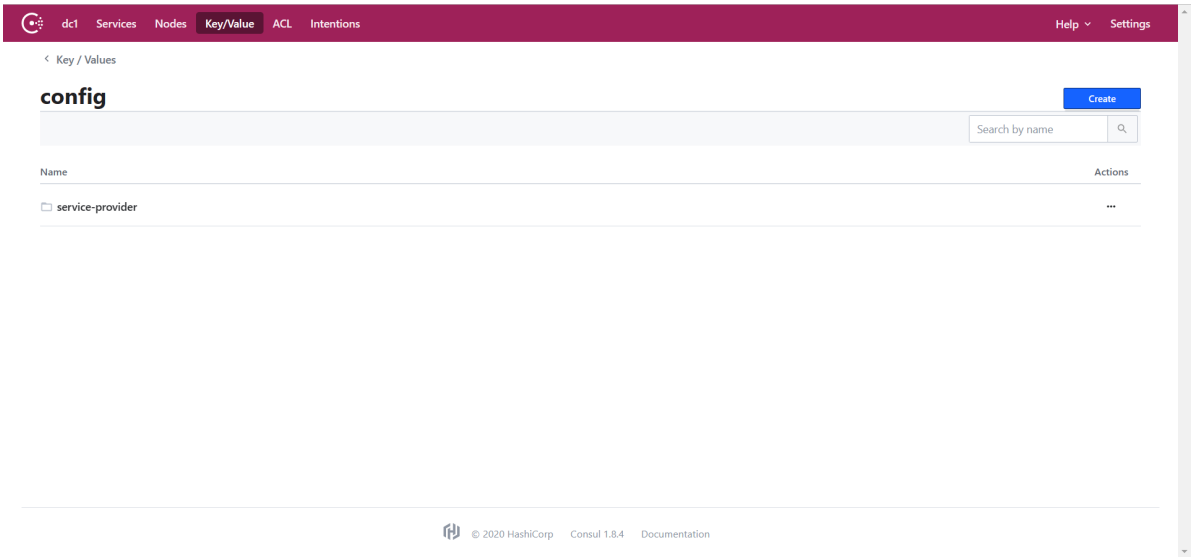


3.2 创建应用目录

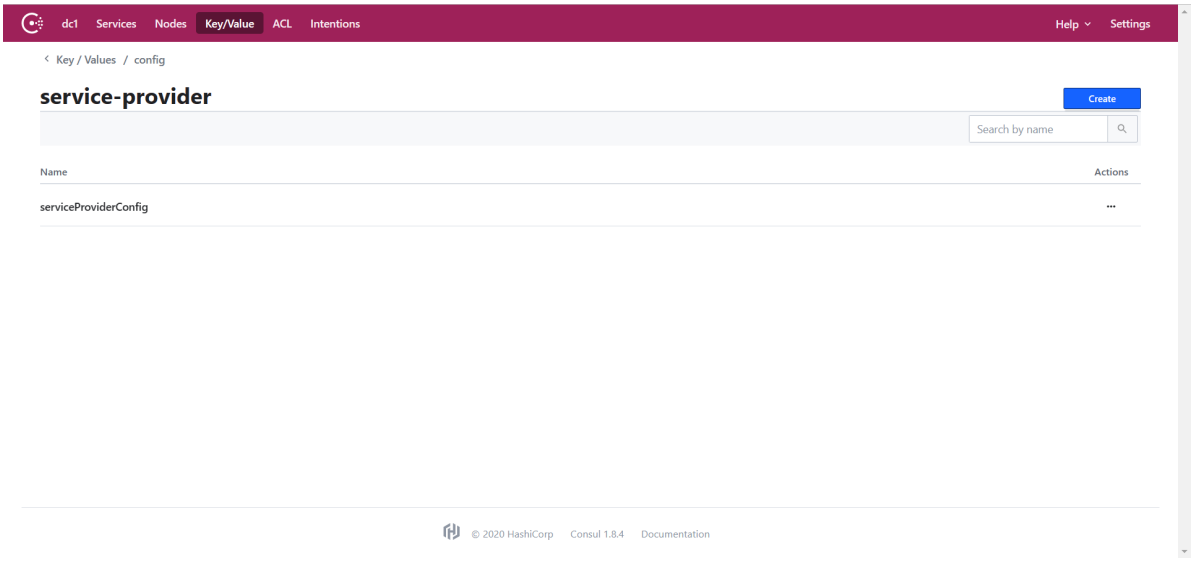
点击 `config` 进入文件夹。



再点击 `Create` 按钮。

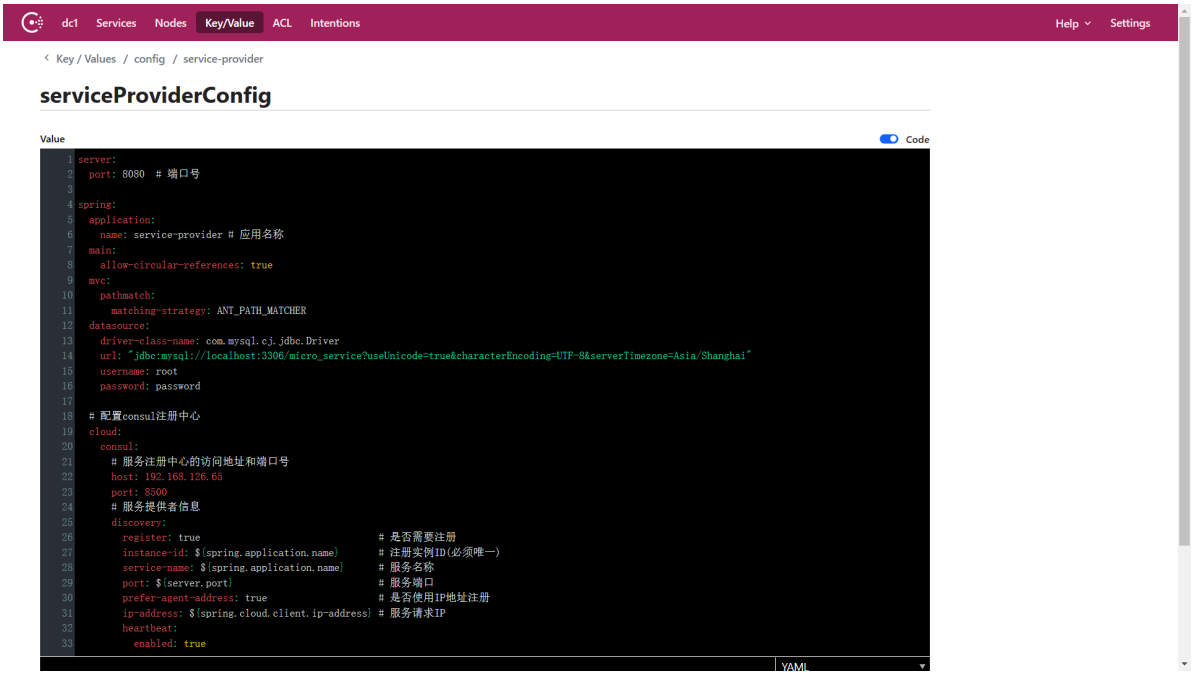


创建 `service-provider/` 应用目录，存储对应微服务应用的 `default` 环境配置信息。



3.4 初始化配置

以 `default` 环境为例，点击 `service-provider` 进入文件夹。点击 `Create` 按钮准备创建 `Key/Value` 配置信息。




```yaml
server:
  port: 8080  # 端口号

spring:
  application:
```

```yaml
    name: service-provider # 应用名称
  main:
    allow-circular-references: true
  mvc:
    pathmatch:
      matching-strategy: ANT_PATH_MATCHER
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: "jdbc:mysql://localhost:3306/micro_service?
useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai"
    username: root
    password: password

  # 配置consul注册中心
  cloud:
    consul:
      # 服务注册中心的访问地址和端口号
      host: 192.168.126.65
      port: 8500
      # 服务提供者信息
      discovery:
        register: true                                  # 是否需要注册
        instance-id: ${spring.application.name}         # 注册实例ID(必须唯一)
        service-name: ${spring.application.name}        # 服务名称
        port: ${server.port}                            # 服务端口
        prefer-agent-address: true                      # 是否使用IP地址注册
        ip-address: ${spring.cloud.client.ip-address} # 服务请求IP
        heartbeat:
          enabled: true
```

3.5 访问接口

http://localhost:8080/product/list


```
[
  - {
      id: 1,
      productName: "Apple iPhone 14pro",
      productNum: 1,
      productPrice: 8699
    },
  - {
      id: 2,
      productName: "笔记本电脑",
      productNum: 1,
      productPrice: 29999
    },
  - {
      id: 3,
      productName: "海尔冰箱",
      productNum: 1,
      productPrice: 15999
    },
  - {
      id: 165,
      productName: "海尔冰箱",
      productNum: 1,
      productPrice: 15999
    }
]
```