

整合Consul

什么是consul?

What is Consul?

Consul is a service mesh solution providing a full featured control plane with service discovery, configuration, and segmentation functionality. Each of these features can be used individually as needed, or they can be used together to build a full service mesh. Consul requires a data plane and supports both a proxy and native integration model. Consul ships with a simple built-in proxy so that everything works out of the box, but also supports 3rd party proxy integrations such as Envoy.

Review the video below to learn more about Consul from HashiCorp's co-founder Armon.

Consul 是一套开源的分布式服务发现和配置管理系统，由 HashiCorp 公司用 Go 语言开发。

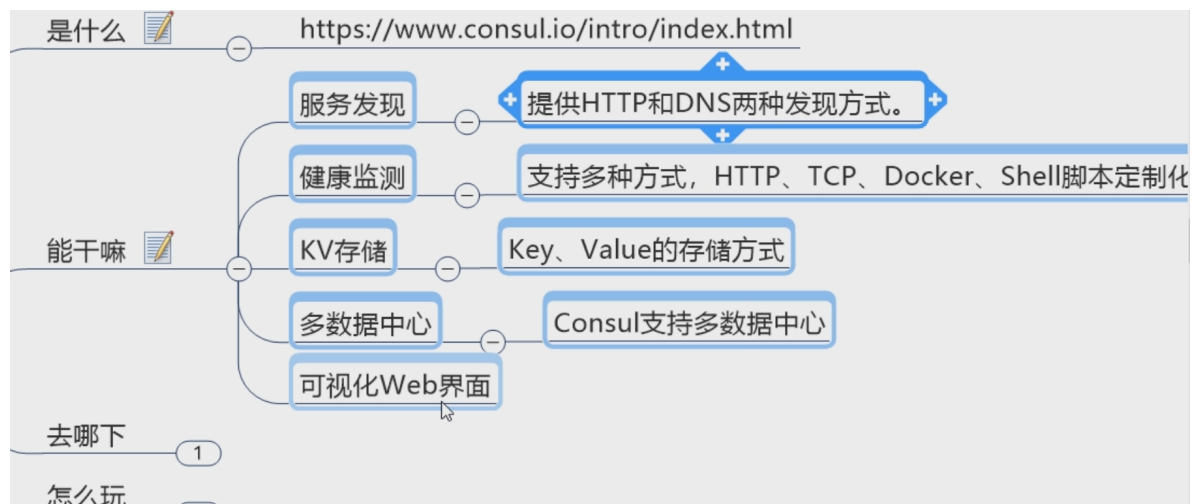
提供了微服务系统中的服务治理、配置中心、控制总线等功能。这些功能中的每一个都可以根据需要单独使用，也可以一起使用以构建全方位的服务网格，总之Consul提供了一种完整的服务网格解决方案。

Spring Cloud Consul 具有如下特性：

The key features of Consul are:

- **Service Discovery:** Clients of Consul can register a service, such as `api` or `mysql`, and other clients can use Consul to discover providers of a given service. Using either DNS or HTTP, applications can easily find the services they depend upon.
- **Health Checking:** Consul clients can provide any number of health checks, either associated with a given service ("is the webserver returning 200 OK"), or with the local node ("is memory utilization below 90%"). This information can be used by an operator to monitor cluster health, and it is used by the service discovery components to route traffic away from unhealthy hosts.
- **KV Store:** Applications can make use of Consul's hierarchical key/value store for any number of purposes, including dynamic configuration, feature flagging, coordination, leader election, and more. The simple HTTP API makes it easy to use.
- **Secure Service Communication:** Consul can generate and distribute TLS certificates for services to establish mutual TLS connections. `Intentions` can be used to define which services are allowed to communicate. Service segmentation can be easily managed with intentions that can be changed in real time instead of using complex network topologies and static firewall rules.
- **Multi Datacenter:** Consul supports multiple datacenters out of the box. This means users of Consul do not have to worry about building additional layers of abstraction to grow to multiple regions.

能干嘛



安装consul 配置环境变量 测试

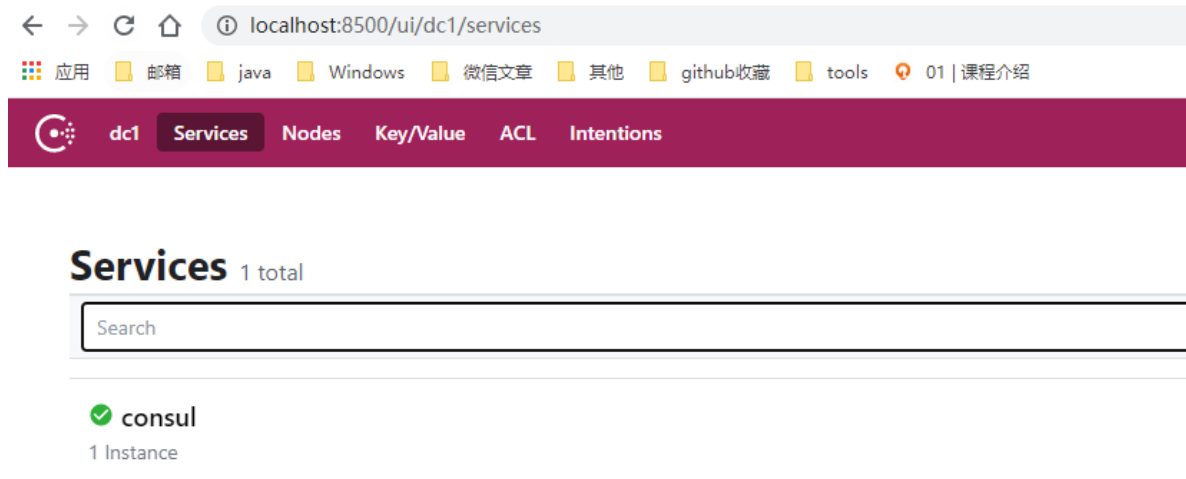
```
Microsoft Windows [版本 10.0.18363.959]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Users\秒度>consul --version
Consul v1.8.0
Protocol 2 spoken by default, understands 2 to 3 (agent will automatically use protocol >2 when speaking to compatible agents)

C:\Users\秒度>consul.exe agent -dev
** Creating Consul agent
```

启动: consul.exe agent -dev

localhost:8500



新建服务提供者consul模块

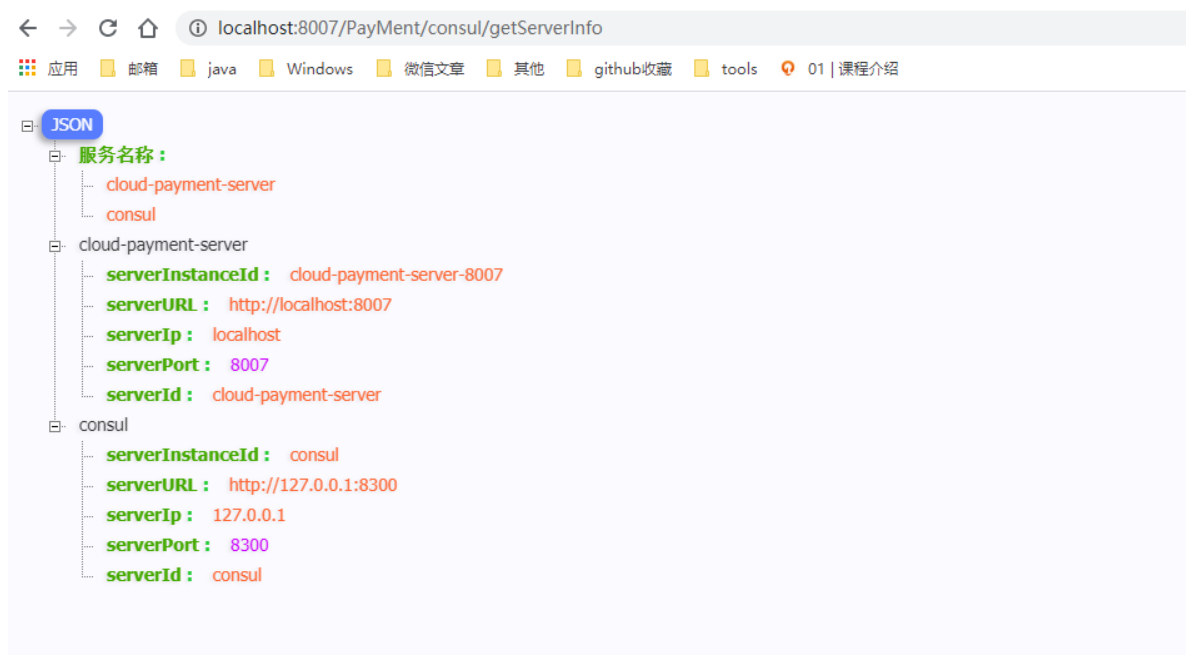
```
<modelVersion>4.0.0</modelVersion>

<artifactId>cloud-providerconsul-payment8006</artifactId>

<dependencies>
    <!--SpringCloud consul-server -->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-consul-discovery</artifactId>
    </dependency>
    <!-- SpringBoot整合Web组件 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
    <!-- 日常通用jar包配置 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
```

```
server:
  port: 8007

spring:
  application:
    name: cloud-payment-server
  cloud:
    consul:
      host: localhost
      port: 8500
    discovery:
      #hostname:127.0.0.1
      service-name: ${spring.application.name}
  datasource:
    driver-class-name: com.mysql.jdbc.Driver
    url: jdbc:mysql:///cfx?useUnicode=true&characterEncoding=utf-8&useSSL=false
    username: root
    password: a5268413
```



构建consul服务消费者模块

JSON

服务名称 :

cloud-consumer-80
cloud-payment-server
consul

cloud-consumer-80

serverInstanceId : cloud-consumer-80-80
serverURL : http://localhost:80
serverIp : localhost
serverPort : 80
serverId : cloud-consumer-80

cloud-payment-server

serverInstanceId : cloud-payment-server-8007
serverURL : http://localhost:8007
serverIp : localhost
serverPort : 8007
serverId : cloud-payment-server

consul

serverInstanceId : consul
serverURL : http://127.0.0.1:8300
serverIp : 127.0.0.1
serverPort : 8300
serverId : consul