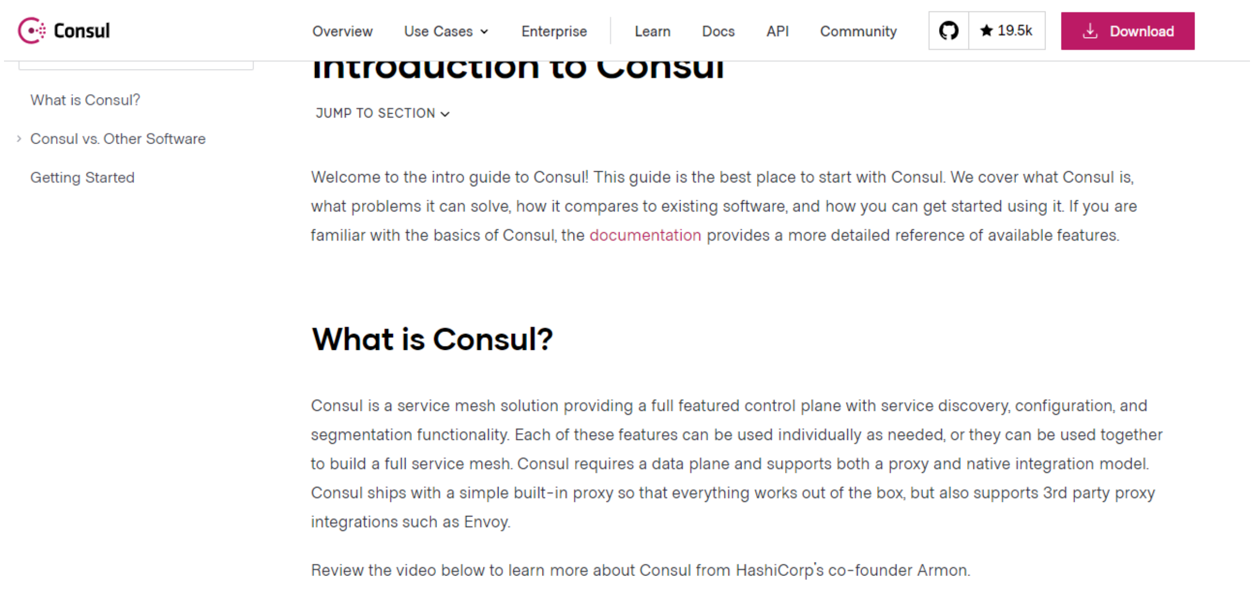
1. Consul简介

1.1. 是什么

官网: https://www.consul.io/intro



Introduction to Consul

Welcome to the intro guide to Consul! This guide is the best place to start with Consul. We cover what Consul is, what problems it can solve, how it compares to existing software, and how you can get started using it. If you are familiar with the basics of Consul, the documentation provides a more detailed reference of available features.

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

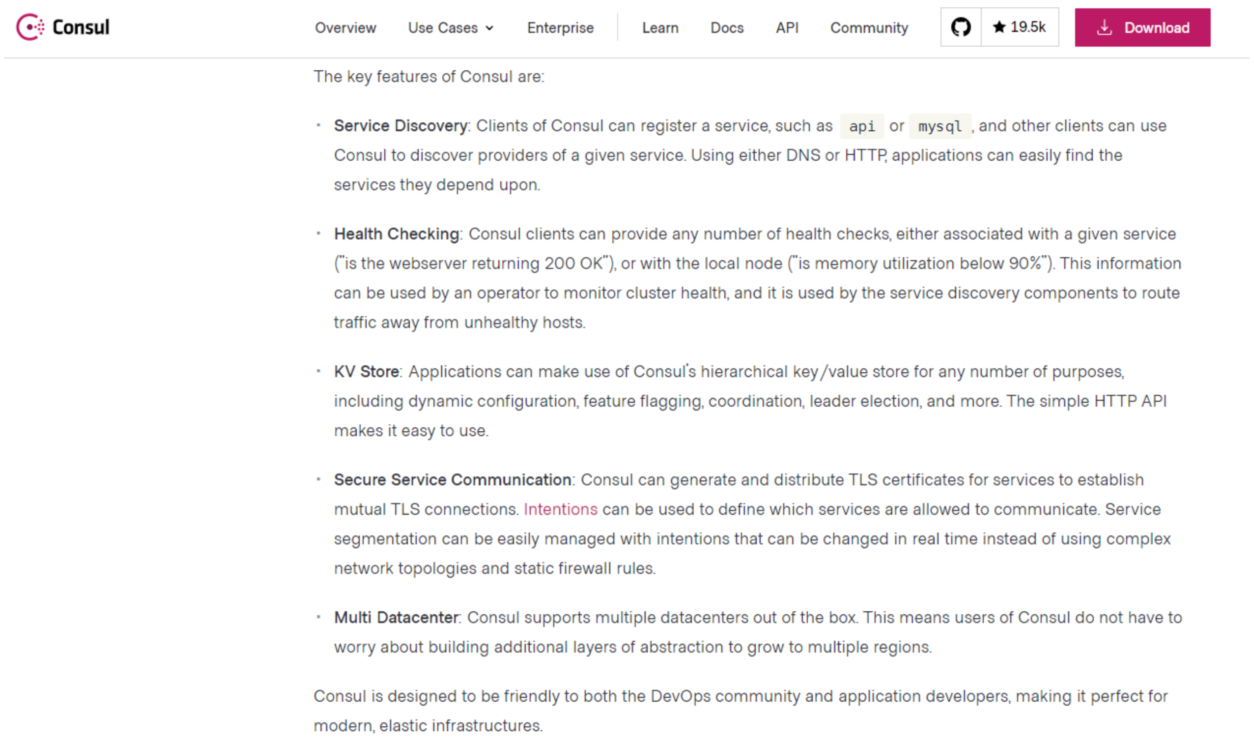
提供了微服务系统中的服务治理、配置中心、控制总线等功能。这些功能中的每一个都可以根据需要单独使用，也可以一起使用以构建全方位的服务网格，总之 Consu提供了一种完整的服务网格解决方案。

它具有很多优点。包括：基于raft协议，比较简洁；支持健康检查，同时支持HTTP和DNS协议支持跨数据中心的WAN集群提供图形界面跨平台，支持 Linux、Mac、Windows

1.2. 能干啥

SpringCloud Consul具体如下特性:

- 服务发现-提供HTTP/DNS两种发现方式
- 健康检测-支持多种方式, HTTP, TCP, Docker, Shell脚本定制化
- KV存储-Key,Value的存储方式
- 多数据中心-Consul支持多数据中心
- 可视化界面



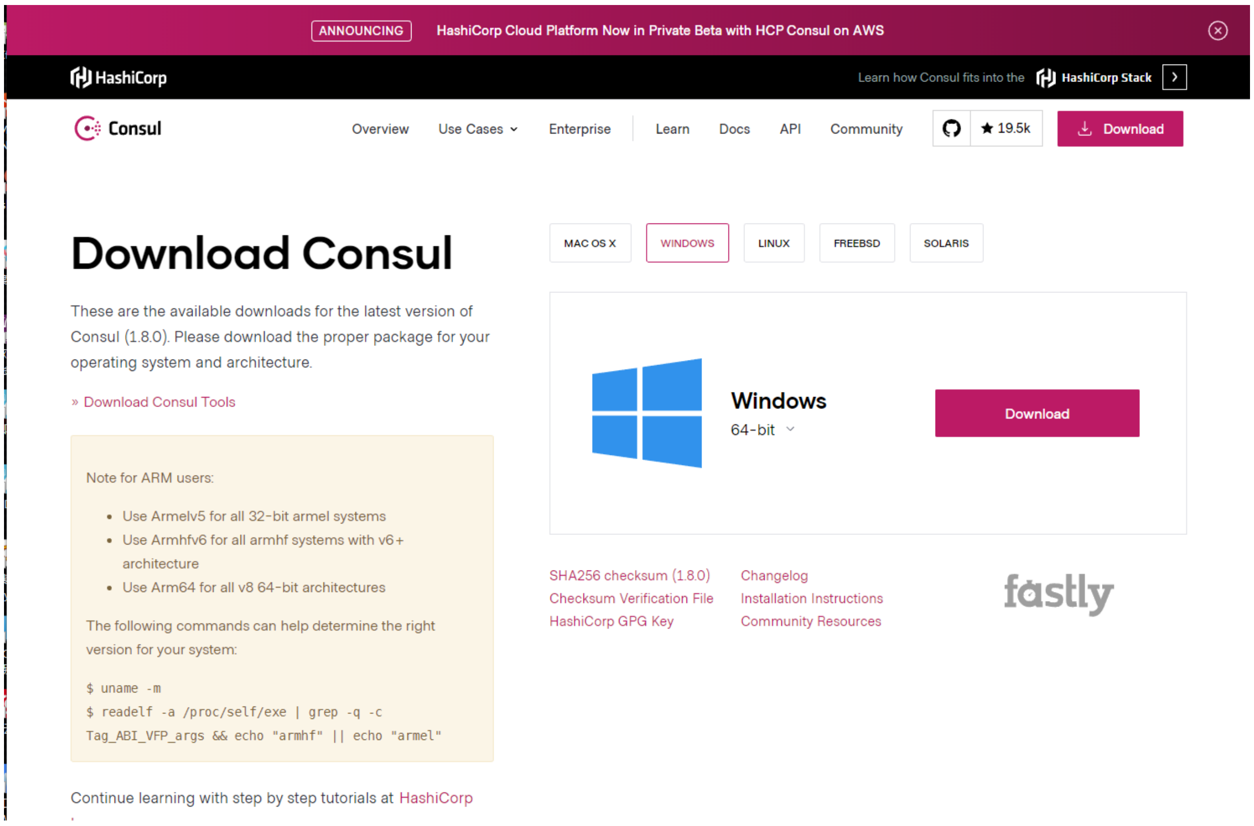
The key features of Consul are:

- **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.

Consul is designed to be friendly to both the DevOps community and application developers, making it perfect for modern, elastic infrastructures.

1.3. 哪里下

下载地址: https://www.consul.io/downloads.html



Download Consul

These are the available downloads for the latest version of Consul (1.8.0). Please download the proper package for your operating system and architecture.

» Download Consul Tools

Note for ARM users:

- Use Armelv5 for all 32-bit armel systems
- Use Armhfv6 for all armhf systems with v6+ architecture
- Use Arm64 for all v8 64-bit architectures

The following commands can help determine the right version for your system:

```
$ uname -m
$ readelf -a /proc/self/exe | grep -q -c
Tag_ABI_VFP_args && echo "armhf" || echo "armel"
```

Continue learning with step by step tutorials at HashiCorp

> 注意事项: 目前Consul最新版本为 1.8.0(截止20200627)
>
> 课程中使用的版本是 consul_1.6.1_windows_amd64.zip
>
> 我们这里使用的版本是 consul_1.7.1_windows_amd64.zip

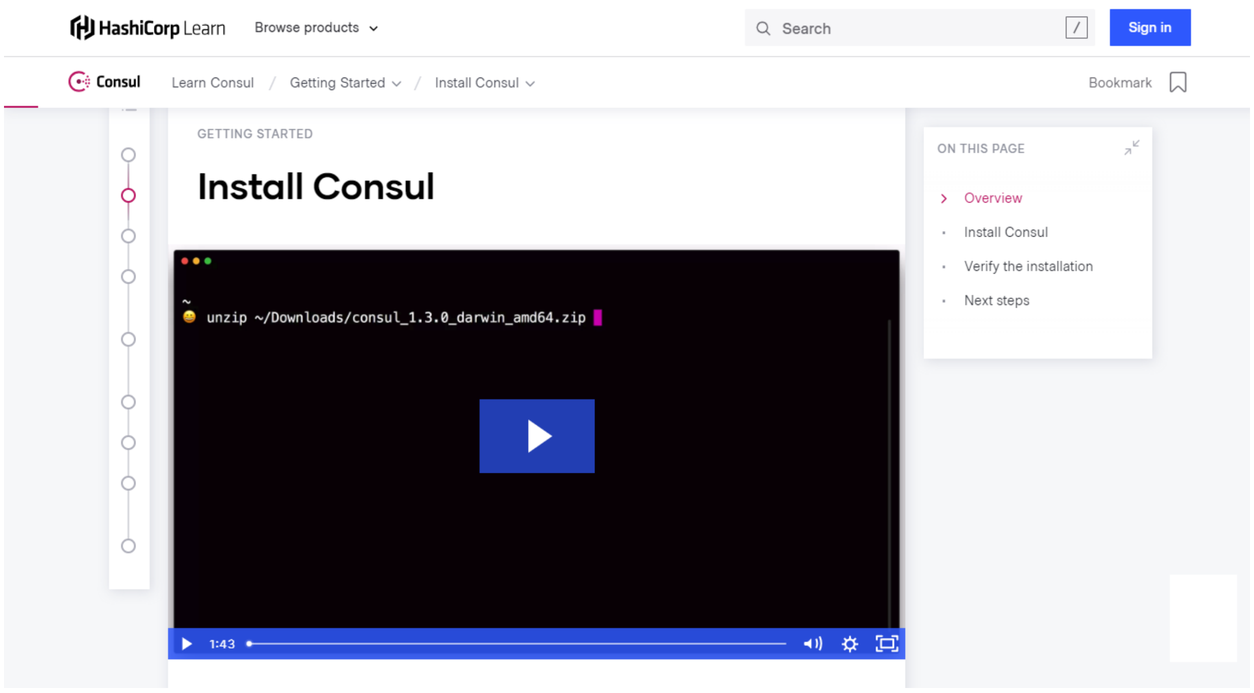
1.4. 怎么玩

参考文档: https://www.springcloud.cc/spring-cloud-consul.html

2. 安装并运行Consul

2.1. 官网说明

https://learn.hashicorp.com/consul/getting-started/install.html

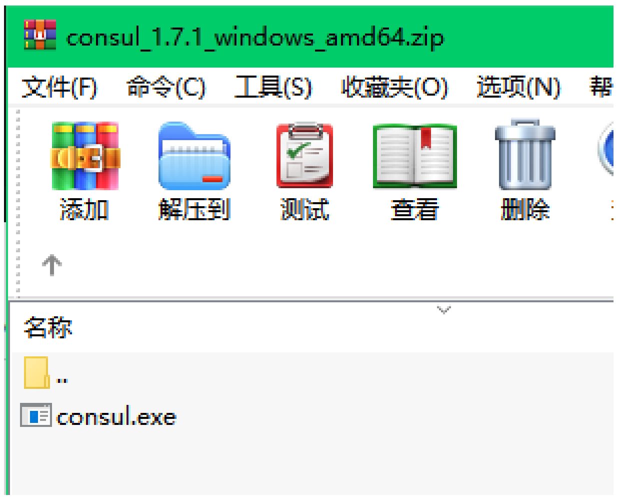


Install Consul

```
~
unzip ~/Downloads/consul_1.3.0_darwin_amd64.zip
```

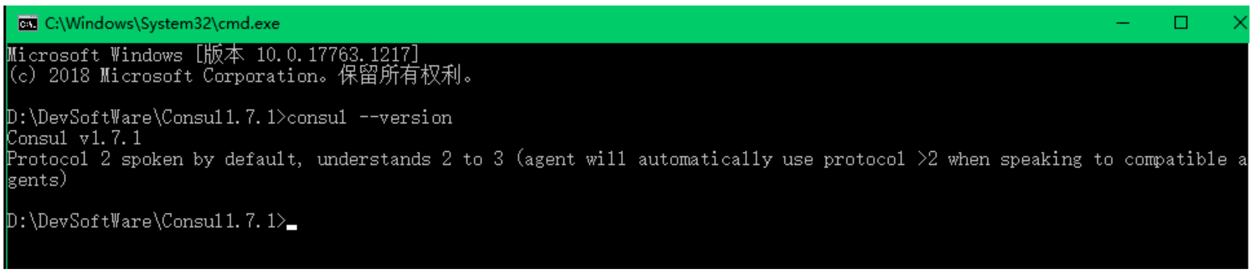
2.2. 安装/运行

下载完成后只有一个consul.exe文件



查看版本信息

```
consul --version
```



```
Microsoft Windows [版本 10.0.17763.1217]
(c) 2018 Microsoft Corporation。保留所有权利。

D:\DevSoftWare\Consul1.7.1>consul --version
Consul v1.7.1
Protocol 2 spoken by default, understands 2 to 3 (agent will automatically use protocol >2 when speaking to compatible a
gents)

D:\DevSoftWare\Consul1.7.1>_
```

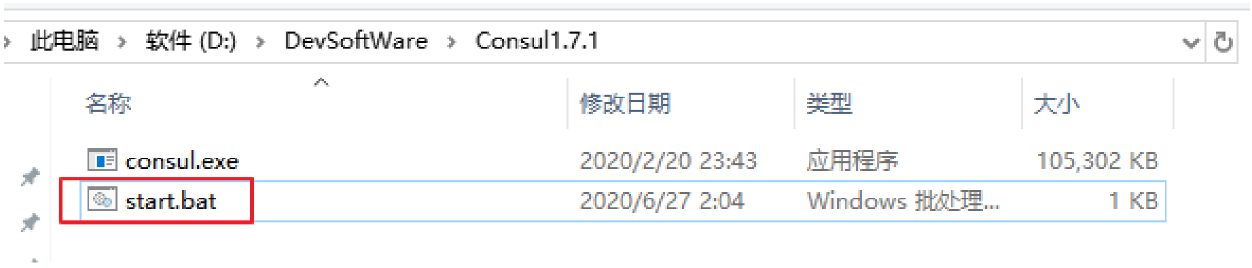
2.3. 使用开发者模式启动

使用开发者模式启动

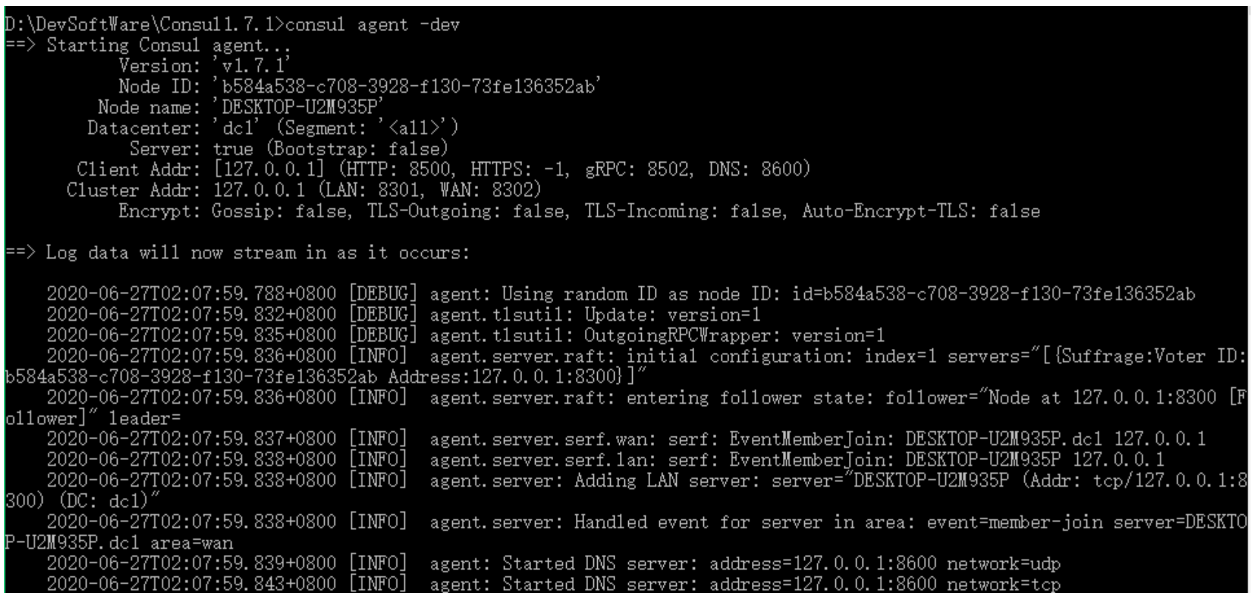
```
consul agent -dev
```

可以在与consul.exe同级文件夹下创建一个.bat脚本,以实现快速启动

```
start consul agent -dev
```

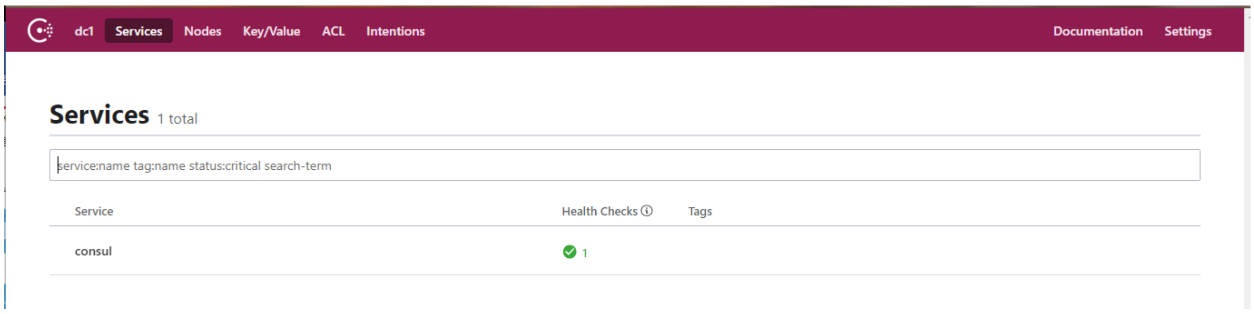


名称	修改日期	类型	大小
consul.exe	2020/2/20 23:43	应用程序	105,302 KB
start.bat	2020/6/27 2:04	Windows 批处理...	1 KB



```
D:\DevSoftWare\Consul1.7.1>consul agent -dev
==> Starting Consul agent...
           Version: 'v1.7.1'
           Node ID: 'b584a538-c708-3928-f130-73fe136352ab'
         Node name: 'DESKTOP-U2M935P'
        Datacenter: 'dc1' (Segment: '<all>')
            Server: true (Bootstrap: false)
       Client Addr: [127.0.0.1] (HTTP: 8500, HTTPS: -1, gRPC: 8502, DNS: 8600)
      Cluster Addr: 127.0.0.1 (LAN: 8301, WAN: 8302)
           Encrypt: Gossip: false, TLS-Outgoing: false, TLS-Incoming: false, Auto-Encrypt-TLS: false

==> Log data will now stream in as it occurs:

    2020-06-27T02:07:59.788+0800 [DEBUG] agent: Using random ID as node ID: id=b584a538-c708-3928-f130-73fe136352ab
    2020-06-27T02:07:59.832+0800 [DEBUG] agent.tlsutil: Update: version=1
    2020-06-27T02:07:59.835+0800 [DEBUG] agent.tlsutil: OutgoingRPCWrapper: version=1
    2020-06-27T02:07:59.836+0800 [INFO]  agent.server.raft: initial configuration: index=1 servers="[{Suffrage:Voter ID:
b584a538-c708-3928-f130-73fe136352ab Address:127.0.0.1:8300}]"
    2020-06-27T02:07:59.836+0800 [INFO]  agent.server.raft: entering follower state: follower="Node at 127.0.0.1:8300 [F
ollower]" leader=
    2020-06-27T02:07:59.837+0800 [INFO]  agent.server.serf.wan: serf: EventMemberJoin: DESKTOP-U2M935P.dc1 127.0.0.1
    2020-06-27T02:07:59.838+0800 [INFO]  agent.server.serf.lan: serf: EventMemberJoin: DESKTOP-U2M935P 127.0.0.1
    2020-06-27T02:07:59.838+0800 [INFO]  agent.server: Adding LAN server: server="DESKTOP-U2M935P (Addr: tcp/127.0.0.1:8
300) (DC: dc1)"
    2020-06-27T02:07:59.838+0800 [INFO]  agent.server: Handled event for server in area: event=member-join server=DESKTO
P-U2M935P.dc1 area=wan
    2020-06-27T02:07:59.839+0800 [INFO]  agent: Started DNS server: address=127.0.0.1:8600 network=udp
    2020-06-27T02:07:59.843+0800 [INFO]  agent: Started DNS server: address=127.0.0.1:8600 network=tcp
```

通过以下地址可以访问Consul的首页: http://localhost:8500


dc1 Services Nodes Key/Value ACL Intentions Documentation Settings

Services 1 total

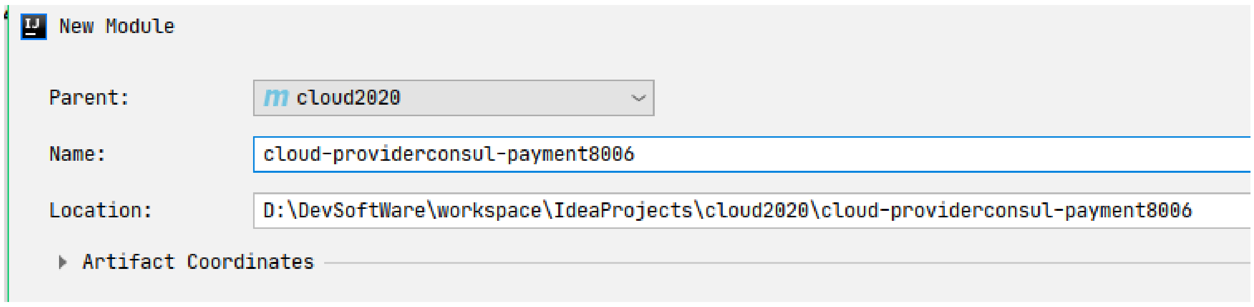
service:name tag:name status:critical search-term

Service	Health Checks	Tags
consul	✓ 1	

3. 服务提供者

3.1. 新建Module支付服务provider8006

新建模块cloud-providerconsul-payment8006


New Module

Parent: cloud2020

Name: cloud-providerconsul-payment8006

Location: D:\DevSoftWare\workspace\IdeaProjects\cloud2020\cloud-providerconsul-payment8006

Artifact Coordinates

3.2. 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.o
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloud-providerconsul-payment8006</artifactId>
    <description>支付服务的提供者之注册中心consul</description>

    <dependencies>
        <!--SpringCloud consul-server-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-consul-discovery</artifactId>
        </dependency>
        <dependency>
            <groupId>cn.sitedev.springcloud</groupId>
            <artifactId>cloud-api-commons</artifactId>
            <version>${project.version}</version>
        </dependency>
        <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>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

3.3. YML

```yaml
server:
  # consul服务端口
  port: 8006
spring:
  application:
    name: cloud-provider-payment
  cloud:
    consul:
      # consul注册中心地址
      host: localhost
      port: 8500
      discovery:
        hostname: 127.0.0.1
        service-name: ${spring.application.name}
```

3.4. 主启动类

```java
package cn.sitedev.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
@EnableDiscoveryClient
public class PaymentMain8006 {
    public static void main(String[] args) {
        SpringApplication.run(PaymentMain8006.class, args);
    }
}
```

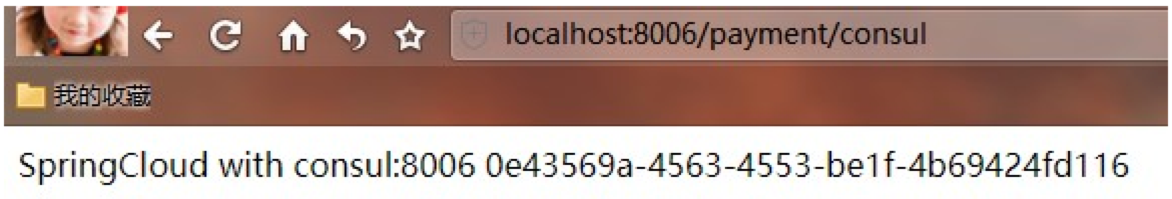
3.5. 业务类Controller

```java
package cn.sitedev.springcloud.controller;

import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.UUID;

@RestController
```

```java
@Slf4j
public class PaymentController {
    @Value("${server.port}")
    private String serverPort;

    @RequestMapping(value = "payment/consul")
    public String paymentConsul() {
        return "SpringCloud with consul:" + serverPort + "\t" + UUID.randomUUID().to
    }
}
```

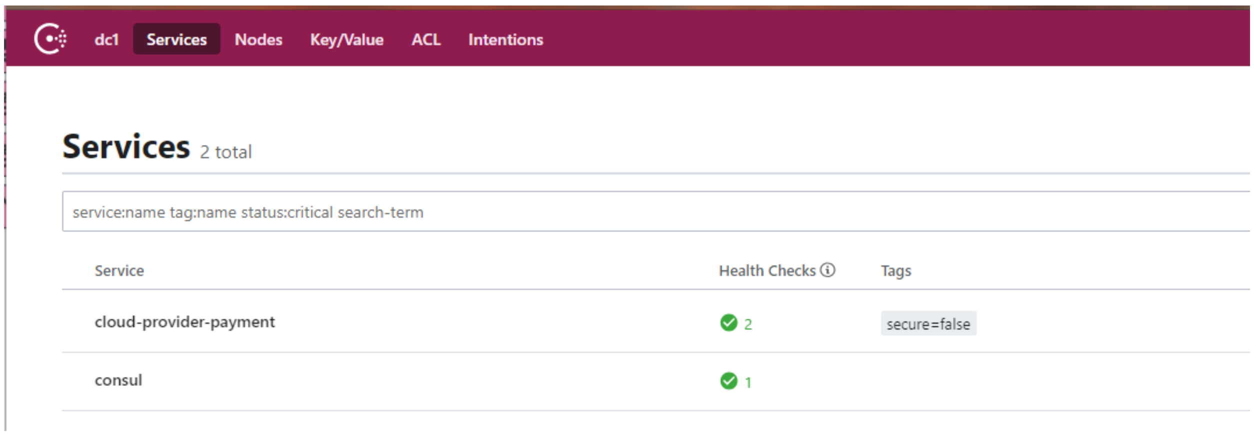
3.6. 验证测试

启动8006服务, 浏览器访问 http://localhost:8006/payment/consul



SpringCloud with consul:8006 0e43569a-4563-4553-be1f-4b69424fd116

浏览器访问 http://localhost:8500/



Services 2 total

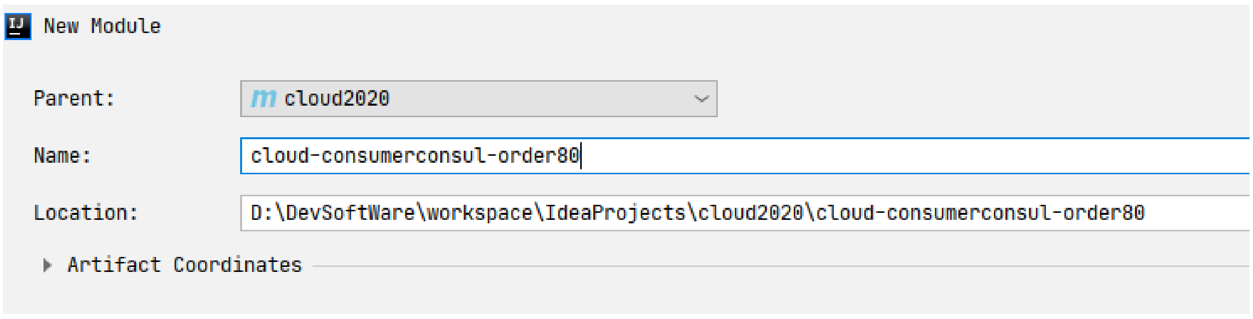
service:name tag:name status:critical search-term

Service	Health Checks	Tags
cloud-provider-payment	✓ 2	secure=false
consul	✓ 1	

4. 服务消费者

4.1. 新建Module 消费服务order80

新建模块cloud-consumerconsul-order80



Parent: cloud2020

Name: cloud-consumerconsul-order80

Location: D:\DevSoftWare\workspace\IdeaProjects\cloud2020\cloud-consumerconsul-order80

4.2. 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.c
    <parent>
        <artifactId>cloud2020</artifactId>
        <groupId>cn.sitedev.springcloud</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>cloud-consumerconsul-payment80</artifactId>
    <description>服务消费者之注册中心consul</description>

    <dependencies>
        <!--SpringCloud consul-server-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-consul-discovery</artifactId>
        </dependency>
        <dependency>
            <groupId>cn.sitedev.springcloud</groupId>
            <artifactId>cloud-api-commons</artifactId>
            <version>${project.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!--监控-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <!--热部署-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>


</project>
```

4.3. YML

```yaml
server:
  port: 80
spring:
  application:
    name: cloud-consumer-order
  cloud:
    consul:
      # consul注册中心地址
      host: localhost
      port: 8500
      discovery:
        hostname: 127.0.0.1
        service-name: ${spring.application.name}
```

4.4. 主启动类

```java
package cn.sitedev.springcloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
@EnableDiscoveryClient
public class OrderConsulMain80 {
    public static void main(String[] args) {
        SpringApplication.run(OrderConsulMain80.class, args);
    }
}
```

4.5. 配置bean

```java
package cn.sitedev.springcloud.config;

import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;

@Configuration
public class ApplicationContextConfig {
    @Bean
    @LoadBalanced
    public RestTemplate getRestTemplate() {
        return new RestTemplate();
    }
}
```

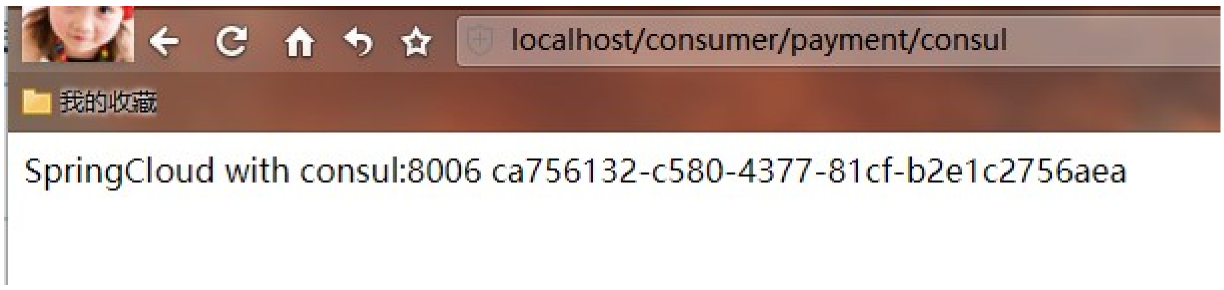
4.6. Controller

```java
package cn.sitedev.springcloud.controller;
```

```java

import lombok.extern.slf4j.Slf4j;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

import javax.annotation.Resource;

@RestController
@Slf4j
public class OrderConsulController {
    public static final String INVOKE_URL = "http://cloud-provider-payment";

    @Resource
    private RestTemplate restTemplate;

    @GetMapping(value = "/consumer/payment/consul")
    public String paymentInfo() {
        String result = restTemplate.getForObject(INVOKE_URL + "/payment/consul", St
        return result;
    }
}
```

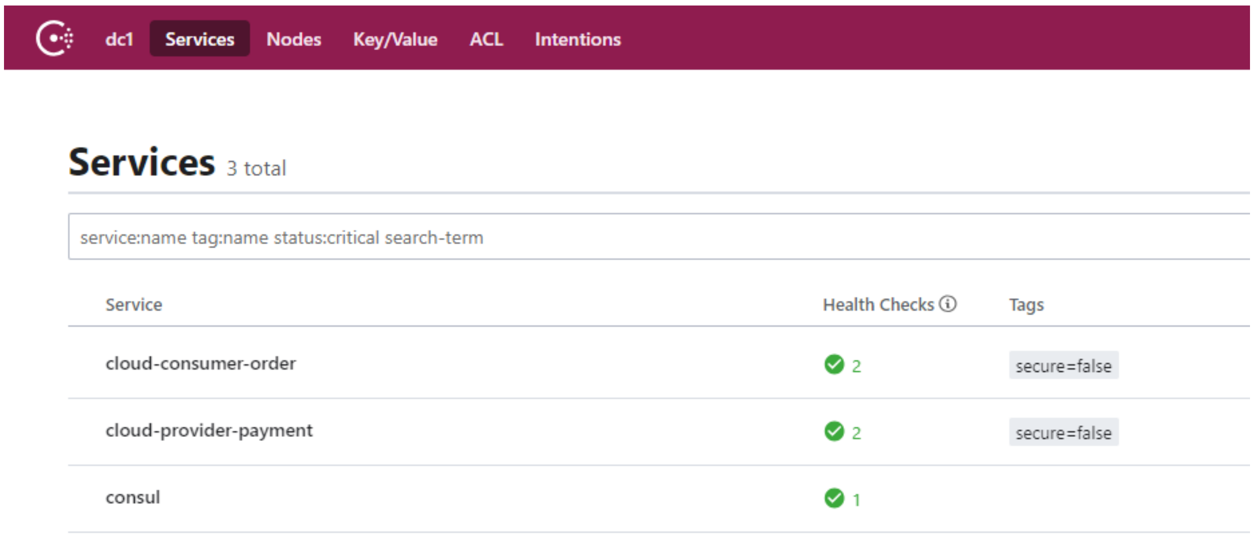
4.7. 验证测试

浏览器访问 http://localhost/consumer/payment/consul



```
SpringCloud with consul:8006 ca756132-c580-4377-81cf-b2e1c2756aea
```

浏览器访问 http://localhost:8500



Services 3 total

service:name tag:name status:critical search-term

Service	Health Checks	Tags
cloud-consumer-order	✓ 2	secure=false
cloud-provider-payment	✓ 2	secure=false
consul	✓ 1	

5. 三种注册中心异同点

5.1. CAP

分区容错性要保证,所以要么是CP,要么是AP

C:Consistency(强一致性)

A:Availability(可用性)

P:Partition tolerance(分区容错性)

CAP理论关注粒度是否是数据,而不是整体系统设计的策略

5.2. 经典CAP图

最多只能同时较好的满足两个。

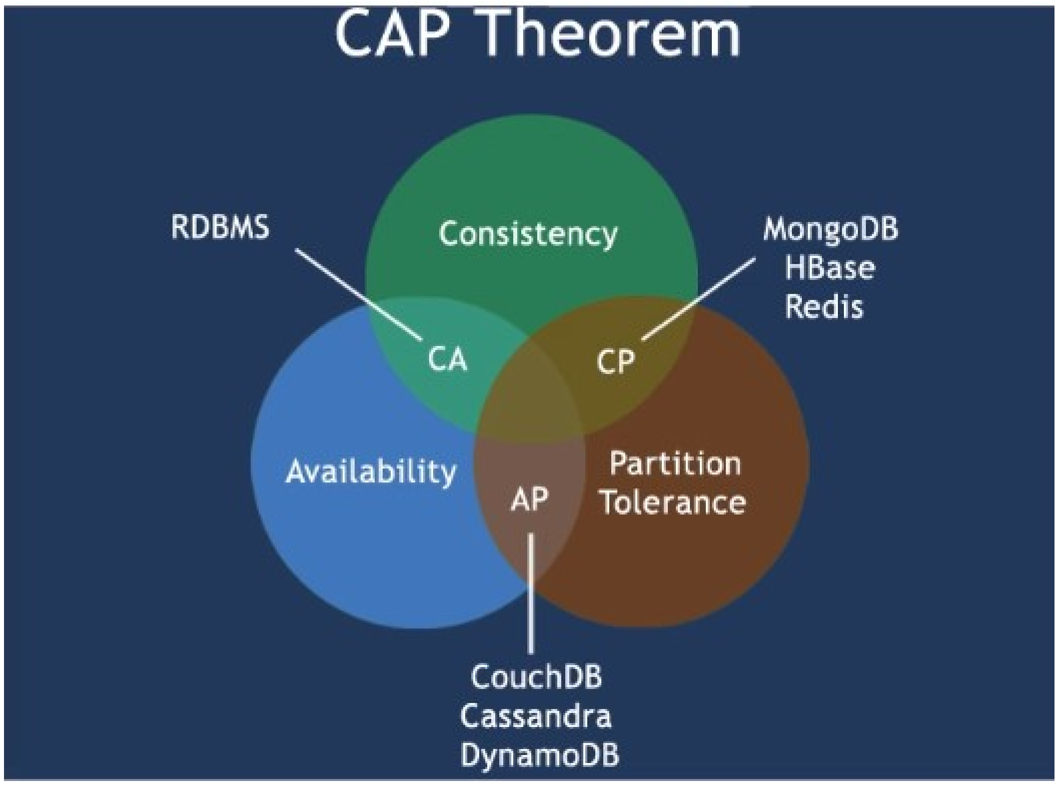
CAP理论的核心是：一个分布式系统不可能同时很好的满足一致性，可用性和分区容错性这三个需求

因此，根据CAP原理将NoSQL数据库分成了满足CA原则、满足CP原则和满足AP原则三大类:

CA-单点集群，满足一致性，可用性的系统，通常在可扩展性上不太强大。

CP-满足一致性，分区容忍性的系统，通常性能不是特别高。

AP-满足可用性，分区容忍性的系统，通常可能对一致性要求低一些



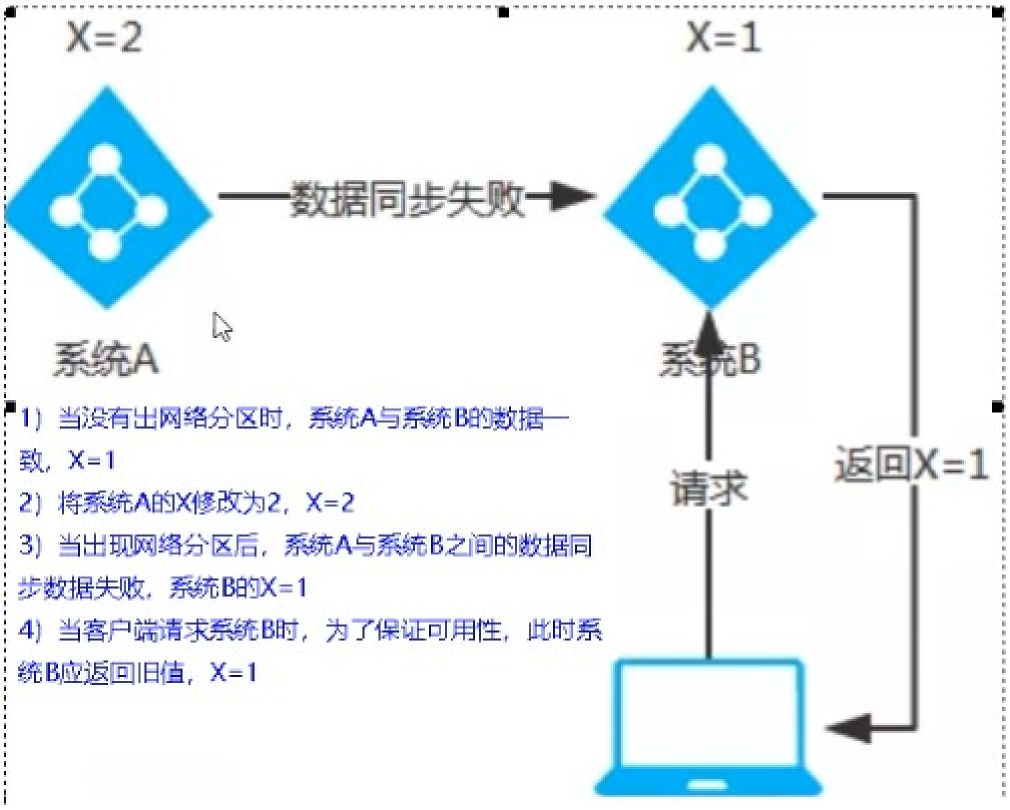
5.2.1. AP (Eureka)

1）当没有出现网络分区时，系统A与系统B的数据一致，X=1

2）将系统A的X修改为2，X=2

3）当出现网络分区后，系统A与系统B之间的数据同步数据失败，系统B=1

4）当客户端请求系统B，为了保证可用性，此时系统B应返回旧值，X=1



1）当没有出网络分区时，系统A与系统B的数据一致，X=1
2）将系统A的X修改为2，X=2
3）当出现网络分区后，系统A与系统B之间的数据同步数据失败，系统B的X=1
4）当客户端请求系统B时，为了保证可用性，此时系统B应返回旧值，X=1

5.2.2. CP (Zookeeper/Consul)

CP架构

当网络分区出现后,为了保证一致性,就必须拒绝请求,否则无法保证一致性

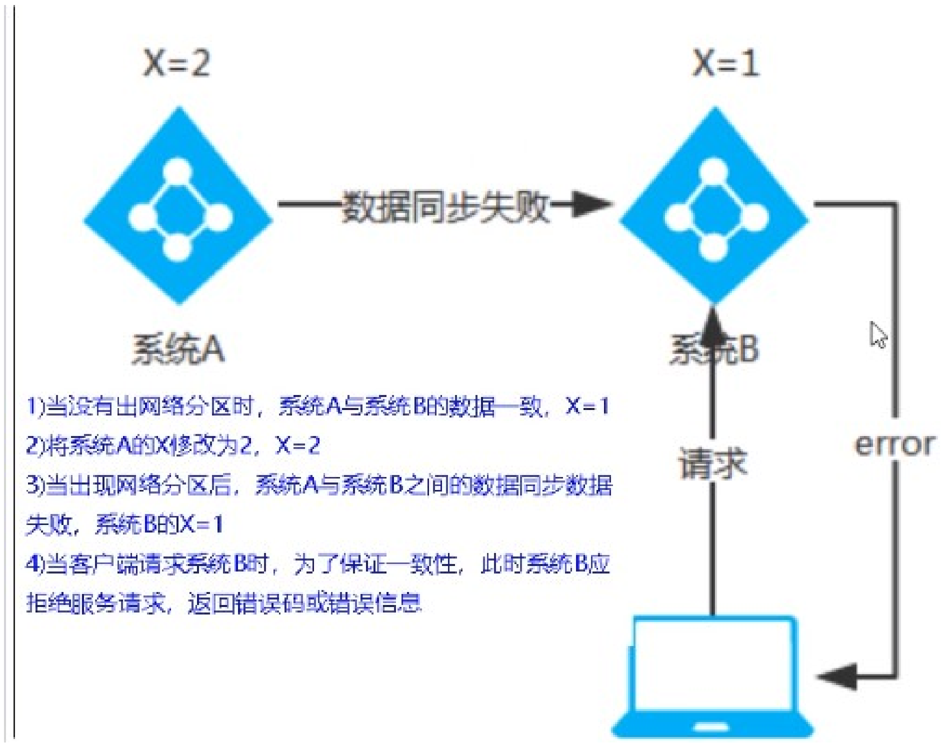
结论:违背了可用性A的要求,只满足一致性和分区容错,即CP

1）当没有出现网络分区时，系统A与系统B数据一致，X=1

2) 将系统A的修改为2，X=2

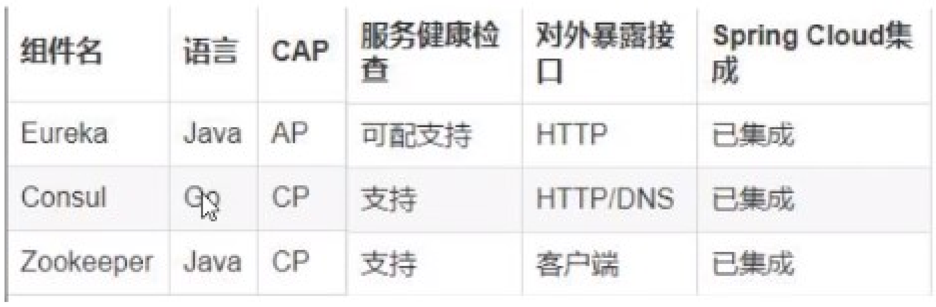
3）当出现网络分区后，系统A与系统B之间的数据同步数据失败，系统B的X=1

4）当客户端请求系统B时，为了保证一致性，此时系统B应拒绝服务请求，返回错误码或错误信息

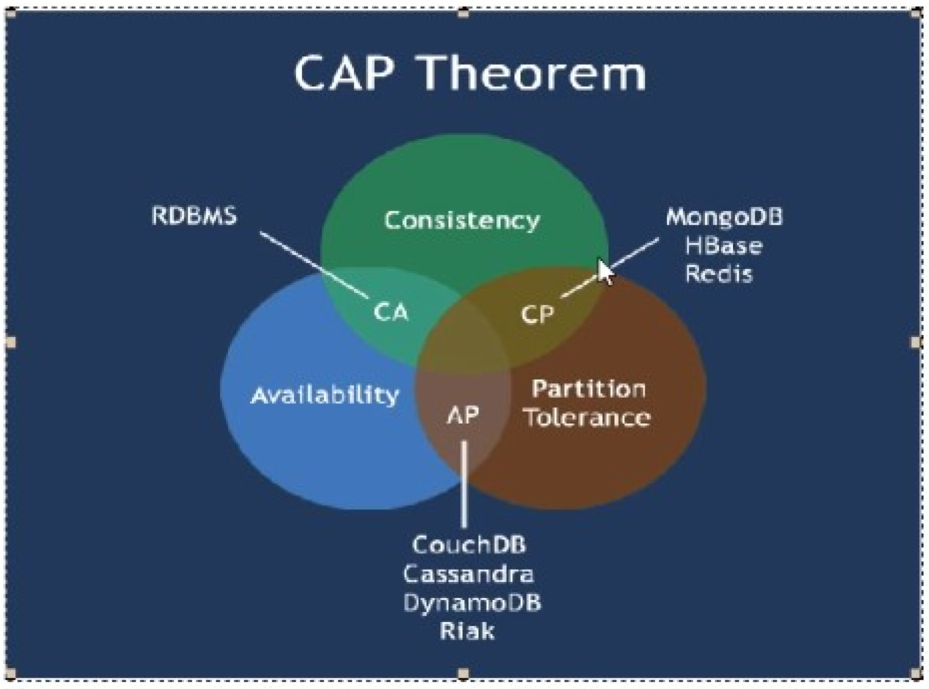


1)当没有出现网络分区时，系统A与系统B的数据一致，X=1
2)将系统A的X修改为2，X=2
3)当出现网络分区后，系统A与系统B之间的数据同步数据失败，系统B的X=1
4)当客户端请求系统B时，为了保证一致性，此时系统B应拒绝服务请求，返回错误码或错误信息

5.3. 三种注册中心异同点



组件名	语言	CAP	服务健康检查	对外暴露接口	Spring Cloud集成
Eureka	Java	AP	可配支持	HTTP	已集成
Consul	Go	CP	支持	HTTP/DNS	已集成
Zookeeper	Java	CP	支持	客户端	已集成


CAP Theorem
Consistency
RDBMS
MongoDB
HBase
Redis
CA
CP
Availability
AP
Partition Tolerance
CouchDB
Cassandra
DynamoDB
Riak