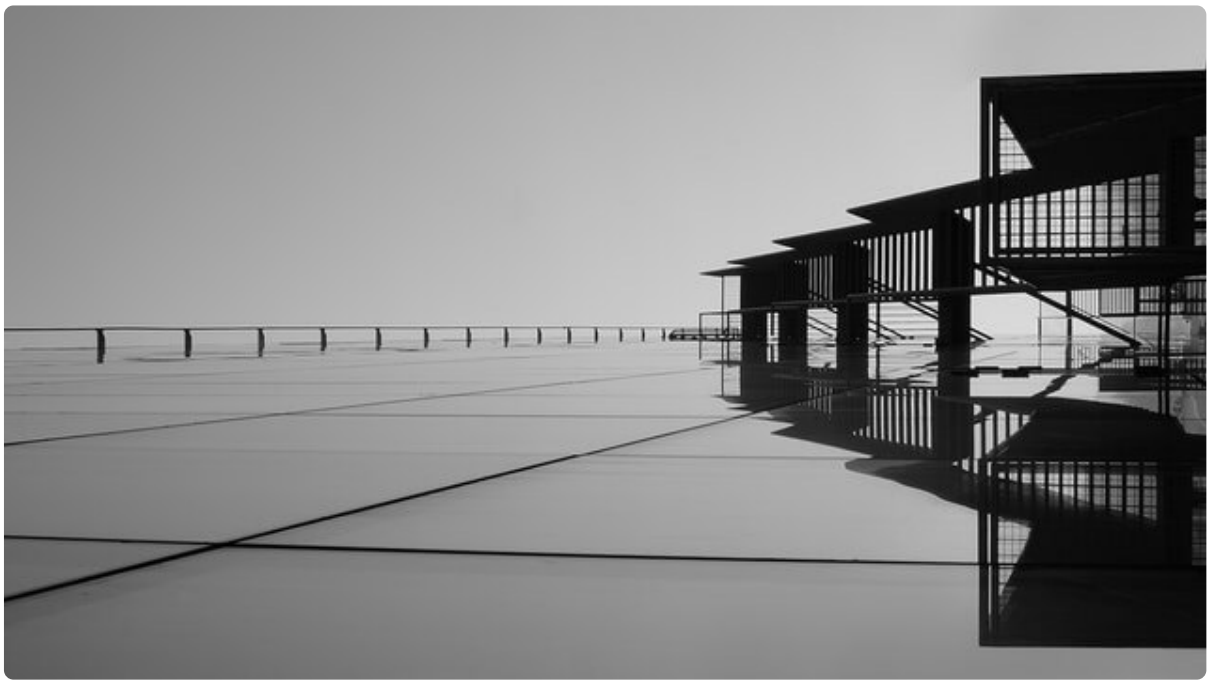
10 Consul 架构原理和实践

更新时间：2019-06-19 17:55:22



> 上天赋予的生命，就是要为人类的繁荣和平和幸福而奉献。
>
> ——松下幸之助

上篇文章我们学习了注册中心 Consul 的相关特性。本节内容我们学习 Consul ，了解它的架构原理，以及如何使用 Consul 进行服务发现和调用。

Consul 核心概念

我们先来了解一下 Consul 架构中的几个核心概念：

- Agent，Agent 是长期运行在每个 Consul 集群成员节点上守护进程。通过命令 Consul agent 启动。Agent 有 Client 和 Server 两种模式。
- Client，客户端是运行 Client 模式的 Agent，负责转发所有的 RPC 到 Server，Client 是相对无状态的。
- Server，服务的是运行 Server 模式的 Agent，包括参与 Raft 仲裁，维护集群的状态，响应 RPC 查询，与其它数据中心交互 WAN gossip，以及将其它查询转发给 Leader 或远程数据中心。
- Datacenter，一个数据中心代表了 Consul 的一个本地部署集群。
- Consensus - Consensus 协议，负责维护不同节点之间状态的一致性。
- Gossip – Consul 是建立在 Serf 之上，提供了完整的 Gossip 协议，用于成员维护故障检测、事件广播。
- LAN Gossip，指的是 LAN gossip pool，包含位于同一个局域网或者数据中心的节点。
- WAN Gossip，指的是 WAN gossip pool，只包含 Server 节点，这些 Server 主要分布在不同的数据中心或者通信是基于互联网或广域网的。
- RPC，远程过程调用，是允许 Client 请求服务器的请求/响应机制。

> Gossip 是一个带冗余的容错算法，更进一步，Gossip 是一个最终一致性算法。虽然无法保证在某个时刻所有节点状态一致，但可以保证在”最终“所有节点一致，”最终“是一个现实中存在，但理论上无法证明的时间点。

Consul 基础架构

每一个 Consul 节点都要运行一个 Consul agent(Consul 代理)。运行一个 Agent 不需要做服务发现或者设置 key/value 数据。这个代理只是负责对当前节点自己还有它上面的服务做健康检查。

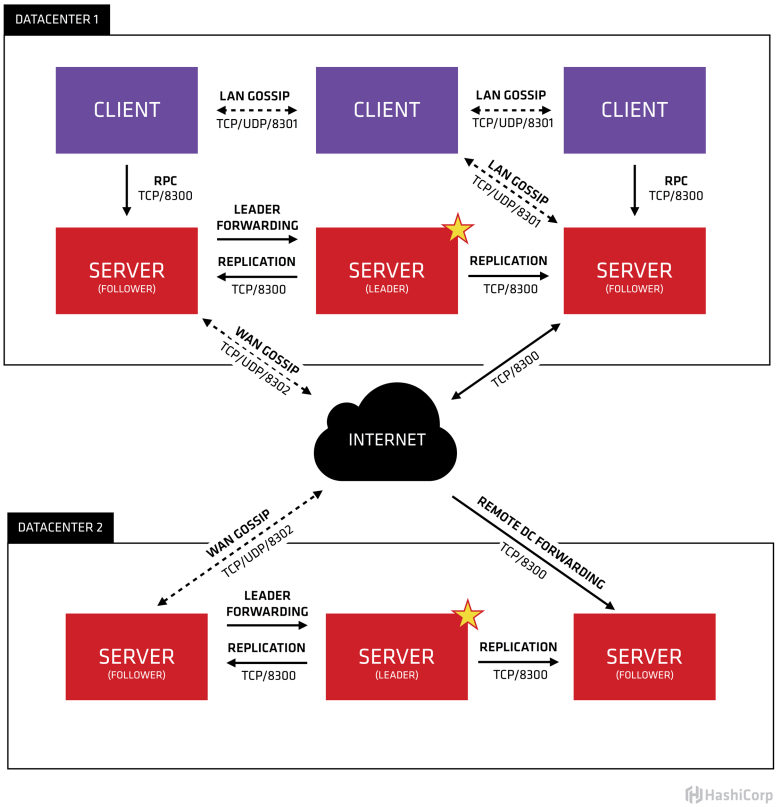
Agent 与一个或多个 Consul servers(Consul 服务)通信。Consul servers 是数据存储和备份的地方。Servers 内部选举一个 Leader。虽然 Consul 可以只有一个 Server，但是为了避免单节点故障造成数据丢失，我们建议最好部署 3-5 个 Server 节点。每个数据中心都需要一个 Consul server 集群。

服务发现组件可以通过 Consul server 或者任意一个 Consul agent 来查询服务或者节点。Agent 可以将这些查询自动转发给 Server。

每个数据中心都要运行一个 Consul server 集群。当一个跨数据中心的服务发现或者配置请求发出时，本地的 Consul Server 会将这些请求转发给远程的数据中心，然后将结果返回。

Consul 集群

Consul 客户端、服务端还支持跨中心的使用，更加提高了它的高可用性。



> 此图来源于 Consul 官网：https://www.consul.io/docs/internals/architecture.html

从上图可以看出，有两个数据中心，分别标记为 DATACENTER 1 和 DATACENTER 2 ,两个数据中心通过 WAN GOSSIP 在 Internet 上交互报文，由此可知 Consul 是支持多数据中心的。

每个数据中心都有两个角色 Server 和 Client，它们之间通过 GRPC 通信。Server 端参与 Raft 仲裁，维护集群的状态，响应 RPC 查询，Client 负责转发 RPC 到 Server 。

Server 和 Client 之间，还有一条 LAN GOSSIP 通信，当 LAN 内部发生了拓扑变化时，存活的节点能够及时感知，比如 Server 节点 Down 掉后，Client 就会触发将对应 Server 节点从可用列表中剥离出去。

官方建议每个 Consul Cluster 有 3 到 5 台 Server，兼顾故障处理和性能的平衡。如果增加越多的机器，则 Consensus 会越来越慢。对 Client 没有限制，可以很容易地扩展到成千上万或数万。

同一个数据中心的所有节点都要加入 Gossip 协议，这意味着 Gossip pool 包含给定数据中心的所有节点。有以下目的：

没有必要为 Client 配置服务器地址参数；发现是自动完成的。

Server 集群之间运行 Raft 协议，通过共识仲裁选举出 Leader。Leader server 负责处理所有的查询和事务，事务也必须通过 Consensus 协议复制到所有的伙伴。当非 Leader Server 接收到一个 RPC 请求，会转发到集群的 Leader。

所有的业务数据都通过 Leader 写入到集群中做持久化，当有半数以上的节点存储了该数据后，Server 集群才会返回 ACK，从而保障了数据的强一致性。

Server 节点和其它数据中心通过 WAN gossip 通讯，WAN gossip 针对互联网传输做了优化，具备更高延迟的网络响应。将一个新的数据中心加入现有的 WAN Gossip 是很容易的，因为相互之间都是通过 WAN gossip 通讯。

当一个 Server 接收到不同数据中心的请求时，它把这个请求转发给对应数据中心随机的一个 Server。然后，接收到请求的 Server 可能会转发给 Leader。

多个数据中心之间是低耦合，但由于故障检测、连接缓存复用、跨数据中心要求快速和可靠的响应。

快速入手

接下来我们使用一个简单的示例，来了解 Consul 的服务注册和调用。首先我们构建一个 Consul 的服务提供者，只提供一个简单的 hello 调用方法。

服务提供者

添加依赖

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-consul-discovery</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
</dependencies>
```

- `spring-boot-starter-actuator` 健康检查依赖于此包
- `spring-cloud-starter-consul-discovery` Spring Cloud Consul 的支持

完整的 pom.xml 文件大家可以参考课程示例源码。

配置文件

```
spring.application.name=consul-producer
server.port=8501
# consul 的地址和端口
spring.cloud.consul.host=localhost
spring.cloud.consul.port=8500
#注册到 consul 的服务名称
spring.cloud.consul.discovery.serviceName=service-producer
```

`spring.cloud.consul.discovery.serviceName` 是指注册到 Consul 的服务名称，客户端可根据这个名称进行服务调用。

启动类

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

注解`@EnableDiscoveryClient`代表开启了服务发现的功能。

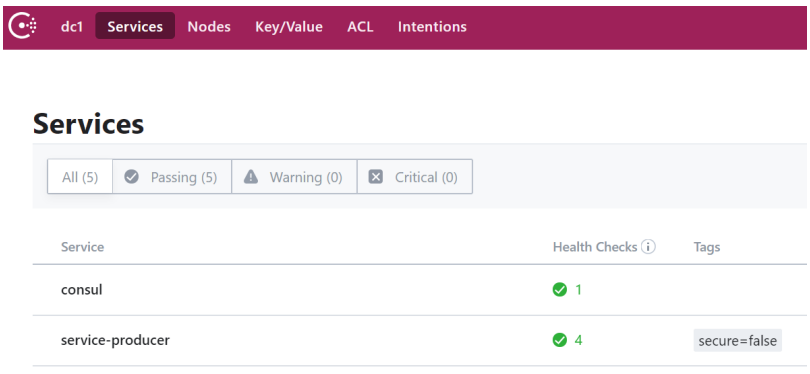
添加服务接口

创建一个 Controller 来提供 hello 服务接口，代码如下:

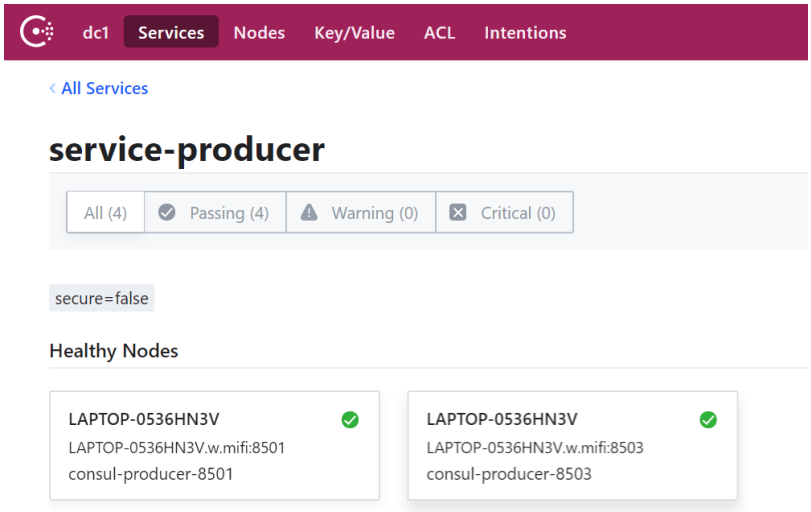
```
@RestController
public class HelloController {

    @RequestMapping("/hello")
    public String hello() {
        return "hello consul";
    }
}
```

为了模拟多服务提供调用，复制上述项目重命名为 consul-producer-2 ,修改对应的端口为 8503，修改 hello 方法的返回值为：”hello consul two”，修改完成后依次启动两个项目。

当两个项目都启动完成之后，我们再来访问地址:`http://localhost:8500`，看看服务注册的情况：



我们发现页面多了 service-producer 服务，点击进去后页面显示有两个服务提供者：



这样服务提供者就准备好了。

服务调用者

我们创建项目 consul-consumer 作为服务的调用者，依赖文件和上述项目相同。

调用者配置文件如下：

配置文件

```
spring.application.name=consul-consumer
server.port=8601
spring.cloud.consul.host=localhost
spring.cloud.consul.port=8500
#设置不需要注册到 consul
spring.cloud.consul.discovery.register=false
```

客户端可以设置是否注册到 Consul 中，大家根据业务选择配置即可。

启动类

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

启动类比较简单，和正常的 Spring Boot 项目相同。

服务调用

我们创建一个 CallHelloController ， 调用我们上面创建好的 hello 方法。

```
@RestController
public class CallHelloController {

    @Autowired
    private LoadBalancerClient loadBalancer;

    @RequestMapping("/call")
    public String call() {
        ServiceInstance serviceInstance = loadBalancer.choose("service-producer");
        System.out.println("服务地址: " + serviceInstance.getUri());
        System.out.println("服务名称: " + serviceInstance.getServiceId());

        String callServiceResult = new RestTemplate().getForObject(serviceInstance.getUri().toString() + "/hello", String.c
lass);
        System.out.println(callServiceResult);
        return callServiceResult;
    }

}
```

启动 consul-consumer 项目，等待启动完成之后，在浏览器多次访问地址：http://localhost:8601/call，通过 call() 方法调用服务提供者的 hello 接口，并且打印接口返回信息。

多次调用后，IED 输出栏中会打印如下信息：

```
服务地址：http://LAPTOP-0536HN3V.w.mifi:8501
服务名称：service-producer
hello consul
服务地址：http://LAPTOP-0536HN3V.w.mifi:8503
服务名称：service-producer
hello consul two
...
```

可以看出服务调用者已经成功获得了服务提供者的响应信息，并且两个服务提供者交替返回信息，服务调用自动实现了均衡负载的功能。

小结

本节给大家介绍了 Consul 的核心概念和工作原理，最后给大家介绍了如何在项目中使用 Consul 的服务注册和调用功能。Consul 是一款非常稳定和成熟的注册中心产品，最新版本的 Consul 提供了一系列的特性，以更好支持未来在服务网格下的使用，如果大家感兴趣可以进一步地关注。

参考链接：

https://www.consul.io/docs/internals/architecture.html

本文作者：纯洁的微笑、江南一点雨

09 Consul 注册中心介绍

11 服务注册与消费