# 第一部分：简单介绍，与使用

## Consul是什么

Consul是一个服务网格（微服务间的 TCP/IP，负责服务之间的网络调用、限流、熔断和监控）解决方案，它是一个一个分布式的，高度可用的系统，而且开发使用都很简便。它提供了一个功能齐全的控制平面，主要特点是：服务发现、健康检查、键值存储、安全服务通信、多数据中心。

consul用于微服务下的服务治理，主要特点有：服务发现、服务配置、健康检查、键值存储、安全服务通信、多数据中心等。

什么叫服务治理发现？起初我们的服务比较单一，各服务之间通过接口就能访问。后面服务越来越复杂出现了分布式，为了不引起单点问题，必然是多服务部署，如果还用原来的方式直接连接，那么在某个服务挂掉或者修改了信息，就会导致连接失败。如果连接端能够不去关心具体的服务配置，他只要连接到那个服务，后续的工作由其它服务保证，包括负载均衡、健康检查等，保证总有可用的连接那就行了，consul就是做这个的，当然，它的功能远不止这些，这里只是以服务发现为例。

与它同类的东西，还有Eureka、zooKeeper、etcd等也能做这些事，说不上谁好谁坏，看场景挑合适的吧，不过Eureka现在已经闭源了，这个还是建议不要去用这个了。

## Consul做什么

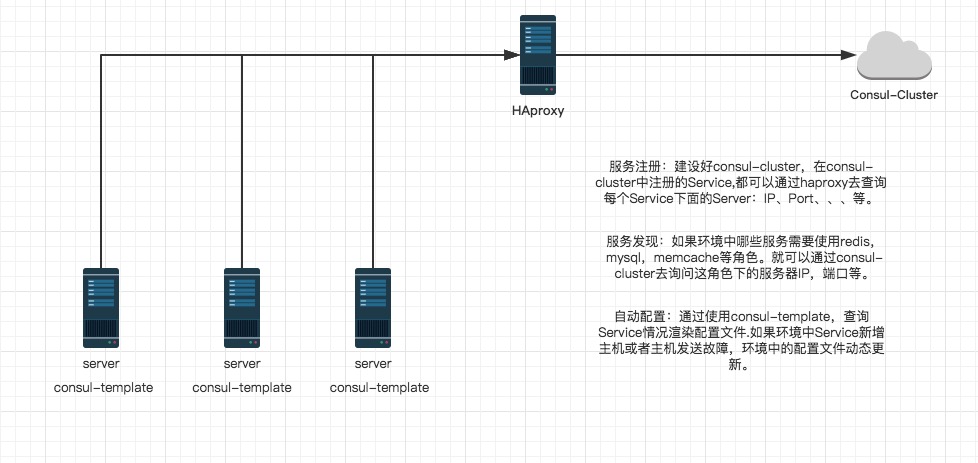
**服务发现**

Consul的客户端可用提供一个服务,比如 api 或者mysql ,另外一些客户端可用使用Consul去发现一个指定服务的提供者.通过DNS或者HTTP应用程序可用很容易的找到他所依赖的服务.  
**健康检查**

Consul客户端可用提供任意数量的健康检查,指定一个服务(比如:webserver是否返回了200 OK 状态码)或者使用本地节点(比如:内存使用是否大于90%). 这个信息可由operator用来监视集群的健康.被服务发现组件用来避免将流量发送到不健康的主机.  
**Key/Value存储**

应用程序可用根据自己的需要使用Consul的层级的Key/Value存储.比如动态配置,功能标记,协调,领袖选举等等,简单的HTTP API让他更易于使用.  
**多数据中心**

Consul支持开箱即用的多数据中心.这意味着用户不需要担心需要建立额外的抽象层让业务扩展到多个区域.  
Consul面向DevOps和应用开发者友好.是他适合现代的弹性的基础设施.



## 基础架构

Consul是一个分布式高可用的系统. 这节将包含一些基础,我们忽略掉一些细节这样你可以快速了解Consul是如何工作的.如果要了解更多细节,请参考深入的架构描述.

每个提供服务给Consul的阶段都运行了一个Consul agent . 发现服务或者设置和获取 key/value存储的数据不是必须运行agent.这个agent是负责对节点自身和节点上的服务进行健康检查的.

Agent与一个和多个Consul Server 进行交互.Consul Server 用于存放和复制数据.server自行选举一个领袖.虽然Consul可以运行在一台server , 但是建议使用3到5台来避免失败情况下数据的丢失.每个数据中心建议配置一个server集群.

你基础设施中需要发现其他服务的组件可以查询任何一个Consul 的server或者 agent.Agent会自动转发请求到server .

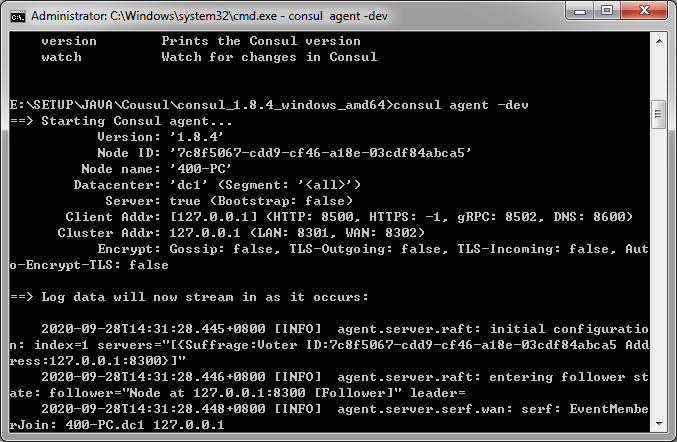
每个数据中运行了一个Consul server集群.当一个跨数据中心的服务发现和配置请求创建时.本地Consul Server转发请求到远程的数据中心并返回结果.

## 下载地址：

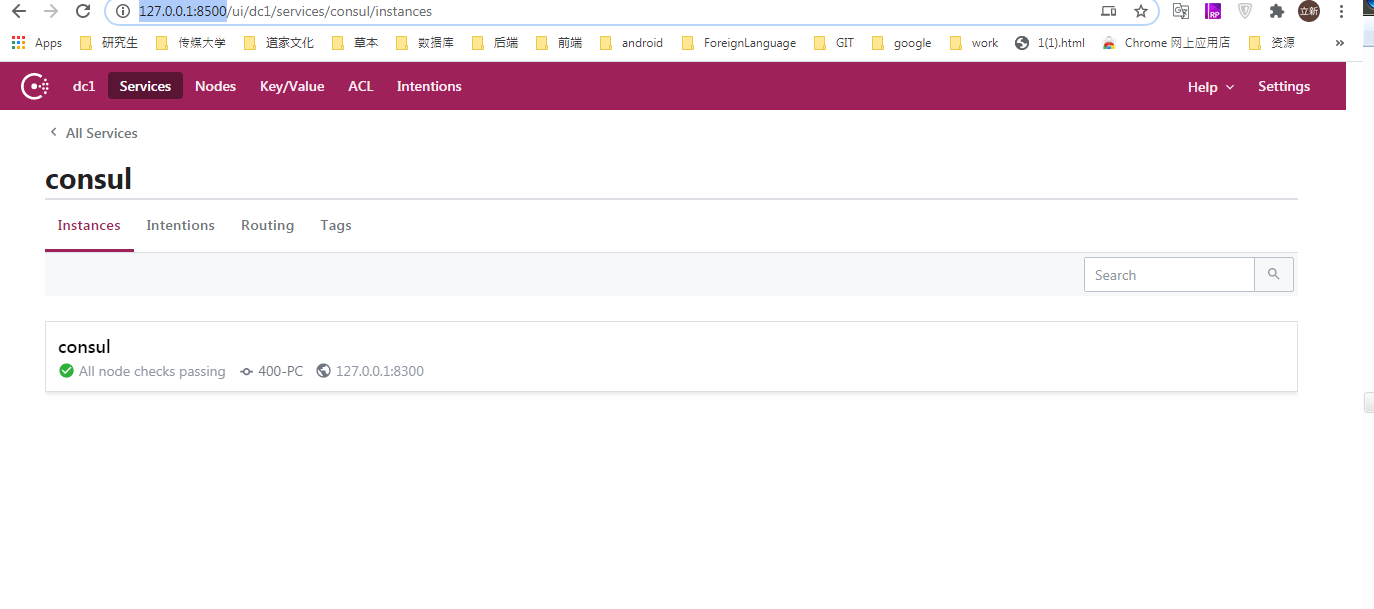
<https://www.consul.io/downloads.html>

## 启动consul

consul agent -dev



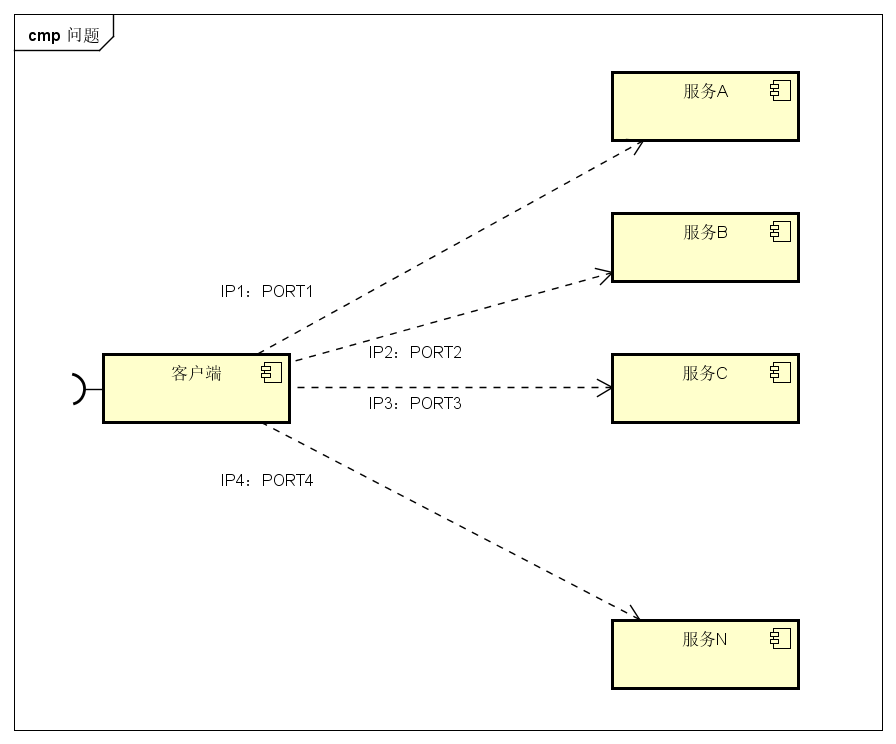
<http://127.0.0.1:8500/>



# 第二：了解微服务

## 什么是服务发现

微服务的框架体系中，服务发现是不能不提的一个模块。我相信了解或者熟悉微服务的童鞋应该都知道它的重要性。这里我只是简单的提一下，毕竟这不是我们的重点。我们看下面的一幅图片：

图中，客户端的一个接口，需要调用服务A-N。客户端必须要知道所有服务的网络位置的，以往的做法是配置是配置文件中，或者有些配置在数据库中。这里就带出几个问题：

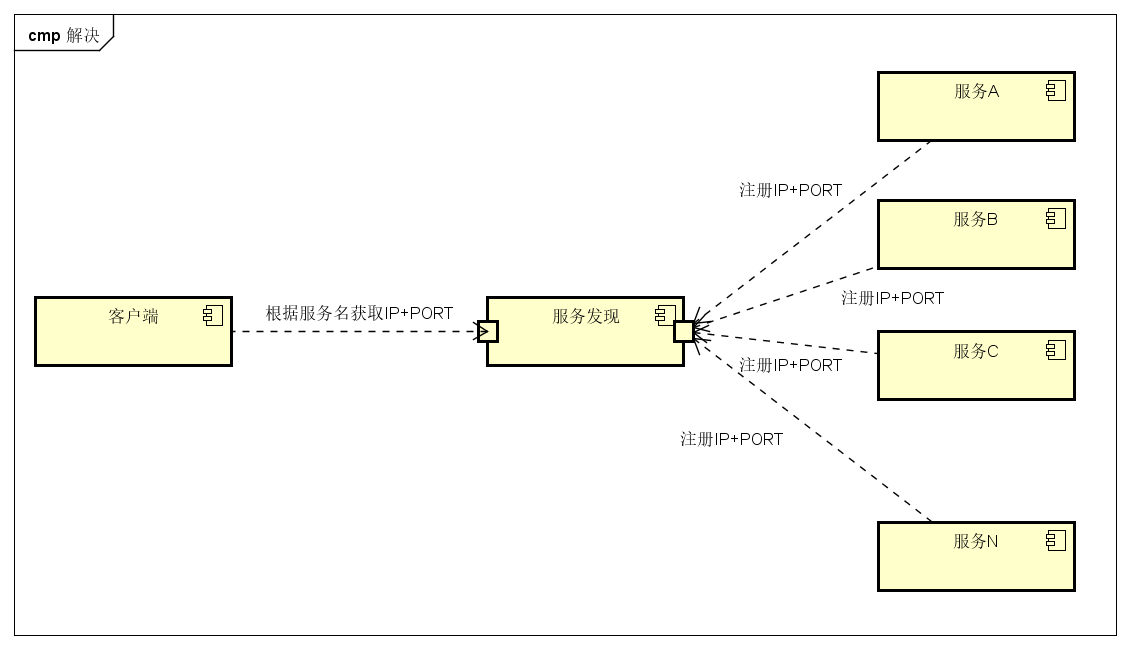
需要配置N个服务的网络位置，加大配置的复杂性

服务的网络位置变化，都需要改变每个调用者的配置

集群的情况下，难以做负载（反向代理的方式除外）

总结起来一句话：服务多了，配置很麻烦，问题多多

既然有这些问题，那么服务发现就是解决这些问题的。话说，怎么解决呢？我们再看一张图



与之前一张不同的是，加了个服务发现模块。图比较简单，这边文字描述下。服务A-N把当前自己的网络位置注册到服务发现模块（这里注册的意思就是告诉），服务发现就以K-V的方式记录下，K一般是服务名，V就是IP:PORT。服务发现模块定时的轮询查看这些服务能不能访问的了（**这就是健康检查**）。客户端在调用服务A-N的时候，就跑去服务发现模块问下它们的网络位置，然后再调用它们的服务。这样的方式是不是就可以解决上面的问题了呢？客户端完全不需要记录这些服务网络位置，客户端和服务端完全解耦！

这个过程大体是这样，当然服务发现模块没这么简单。里面包含的东西还很多。这样表述只是方便理解。

图中的服务发现模块基本上就是微服务架构中服务发现的作用了。

## consul 简介

做服务发现的框架常用的有

zookeeper

eureka

etcd

consul

这里就不比较哪个好哪个差了，需要的童鞋自己谷歌百度。

那么consul是啥？**consul就是提供服务发现的工具**。然后下面是简单的介绍：

consul是分布式的、高可用、横向扩展的。consul提供的一些关键特性：

service discovery：consul通过DNS或者HTTP接口使服务注册和服务发现变的很容易，一些外部服务，例如saas提供的也可以一样注册。

health checking：健康检测使consul可以快速的告警在集群中的操作。和服务发现的集成，可以防止服务转发到故障的服务上面。

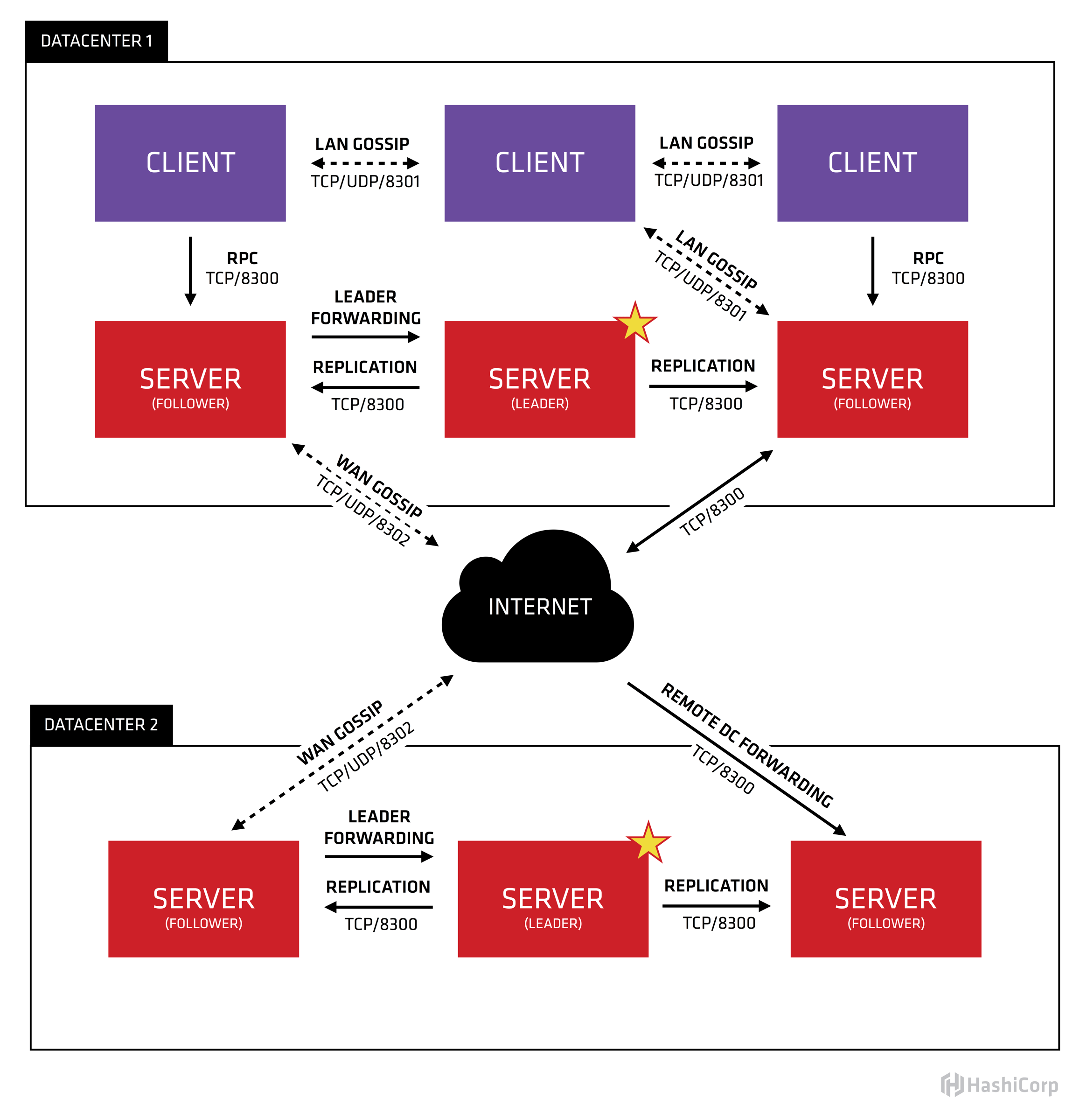
key/value storage：一个用来存储动态配置的系统。提供简单的HTTP接口，可以在任何地方操作。

multi-datacenter：无需复杂的配置，即可支持任意数量的区域。

我们这里会介绍服务发现，健康检查，还有一些基本KV存储。多数据中心有机会另一篇文章再说。

总结：只要知道它是解决我上一部分提出的问题就行，其它的东西慢慢理解

## consul的几个概念



上图来自于consul官方文档

我们只看数据中心1，可以看出consul的集群是由N个SERVER，加上M个CLIENT组成的。而不管是SERVER还是CLIENT，都是consul的一个节点，所有的服务都可以注册到这些节点上，正是通过这些节点实现服务注册信息的共享。除了这两个，还有一些小细节，一一简单介绍。

CLIENT

CLIENT表示consul的client模式，就是客户端模式。是consul节点的一种模式，这种模式下，所有注册到当前节点的服务会被转发到SERVER，本身是不持久化这些信息。

SERVER

SERVER表示consul的server模式，表明这个consul是个server，这种模式下，功能和CLIENT都一样，唯一不同的是，它会把所有的信息持久化的本地，这样遇到故障，信息是可以被保留的。

SERVER-LEADER

中间那个SERVER下面有LEADER的字眼，表明这个SERVER是它们的老大，它和其它SERVER不一样的一点是，它需要负责同步注册的信息给其它的SERVER，同时也要负责各个节点的健康监测。

其它信息其它信息包括它们之间的通信方式，还有一些协议信息，算法。它们是用于保证节点之间的数据同步，实时性要求等等一系列集群问题的解决。这些有兴趣的自己看看官方文档。

## 运行 Consul Agent

Consul安装之后，代理必须运行。 代理可以在服务器或客户端模式下运行。 每个数据中心都必须至少有一台服务器，但推荐使用3台或5台服务器。 一个单一的服务器部署是非常不推荐的，因为在故障情况下数据丢失是不可避免的。

　　所有其他代理以客户端模式运行。 客户端是一个非常轻量级的进程，它注册服务，运行健康检查，并将查询转发给服务器。 代理程序必须在集群中的每个节点上运行。

　　为了简单起见，我们现在将以开发模式启动Consul代理。 这种模式对于快速简单地启动单节点Consul环境非常有用。 它并不打算在生产中使用，因为它不会持续任何状态。