**Consul注册中心详解**

数工研发部：夏神保

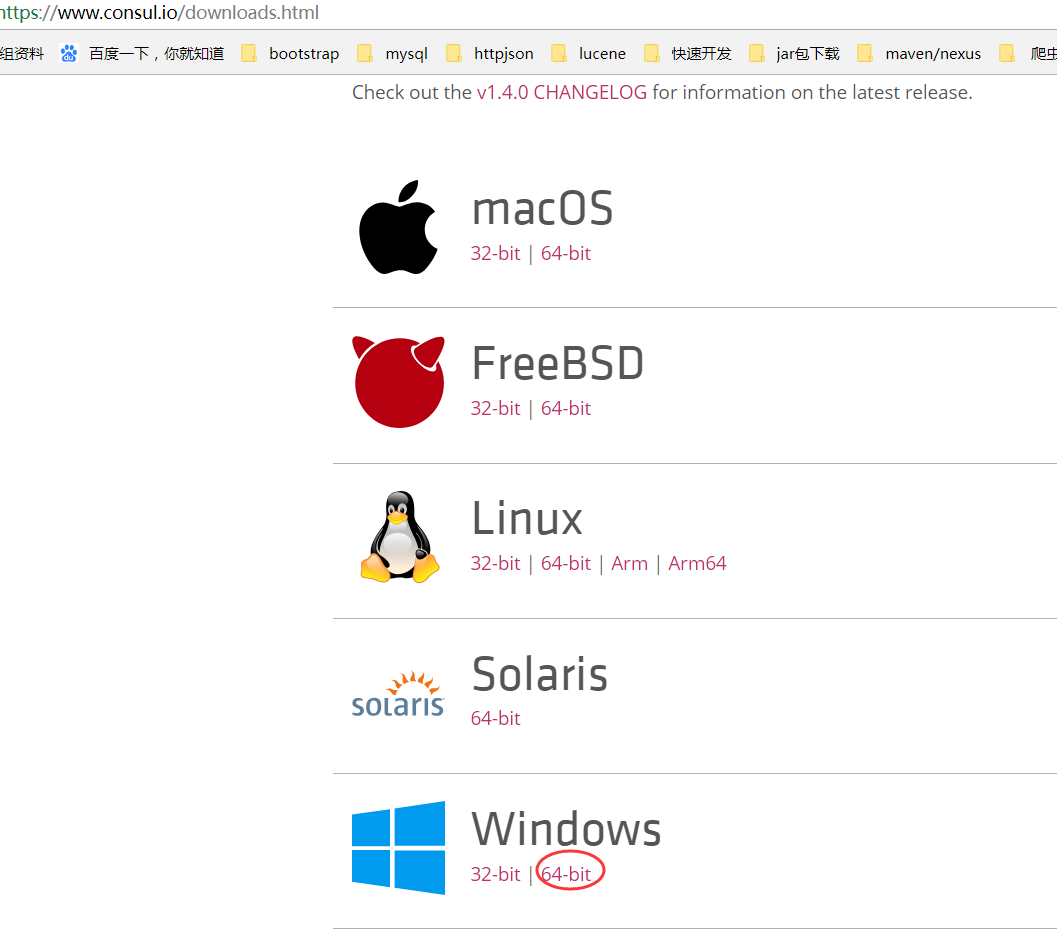
2018年12月4日

# 概述

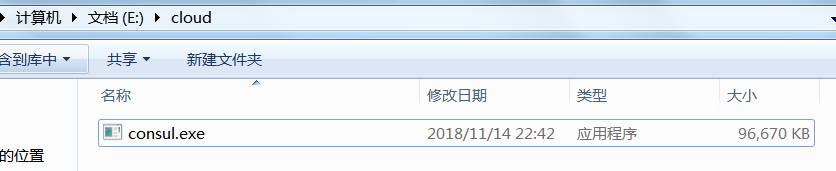
Consul是一个分布式服务，可以运行于任何的公有云、私有云平台，是google开源的一个使用go语言开发的服务发现、配置管理中心服务，内置了服务注册与发现框架、分布一致性协议实现、健康检查、键值对存储、多数据中心方案，不再需要依赖其他工具（如Zookeeper等），服务部署简单，只有一个二进制的包，每个服务器节点都要运行agent，他有2中运行模式server服务端和client客户端，每个数据中心官方建议需要3或5个server节点，以保证数据安全，同时保证server-leader的选举能够正确的进行。

# 下载consul

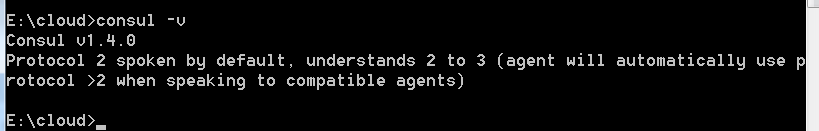
下载地址：<https://www.consul.io/downloads.html>



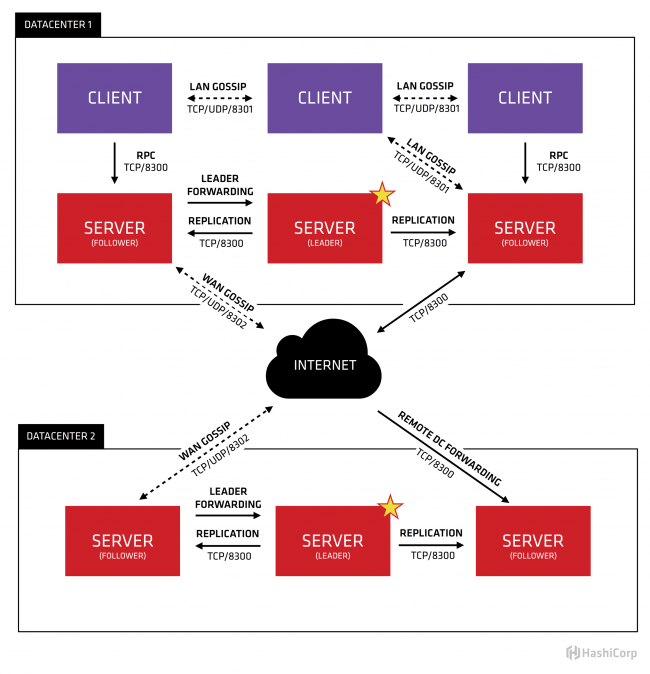
下载完成后，解压：



在此文件夹打开CMD窗口，查看consul版本：



# 内部原理



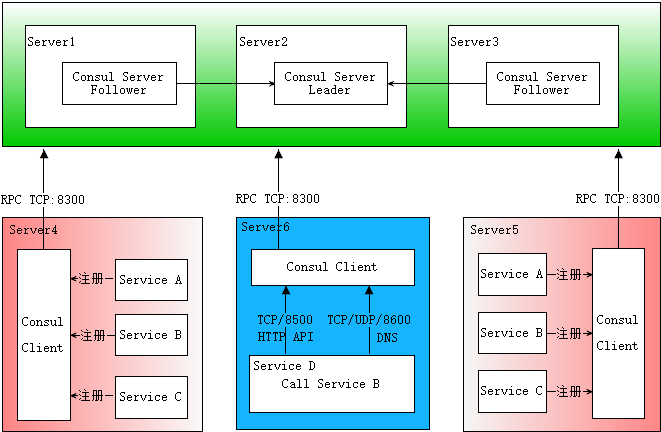
首先Consul支持多数据中心，在上图中有两个DataCenter，他们通过Internet互联，同时请注意为了提高通信效率，只有Server节点才加入跨数据中心的通信。

在单个数据中心中，Consul分为Client和Server两种节点（所有的节点也被称为Agent），Server节点保存数据，Client负责健康检查及转发数据请求到Server；Server节点有一个Leader和多个Follower，Leader节点会将数据同步到Follower，Server的数量推荐是3个或者5个，在Leader挂掉的时候会启动选举机制产生一个新的Leader。

集群内的Consul节点通过gossip协议（流言协议）维护成员关系，也就是说某个节点了解集群内现在还有哪些节点，这些节点是Client还是Server。单个数据中心的流言协议同时使用TCP和UDP通信，并且都使用8301端口。跨数据中心的流言协议也同时使用TCP和UDP通信，端口使用8302。

集群内数据的读写请求既可以直接发到Server，也可以通过Client使用RPC转发到Server，请求最终会到达Leader节点，在允许数据轻微陈旧的情况下，读请求也可以在普通的Server节点完成，集群内数据的读写和复制都是通过TCP的8300端口完成。

# 服务发现原理



首先需要有一个正常的Consul集群，有Server，有Leader。这里在服务器Server1、Server2、Server3上分别部署了Consul Server，假设他们选举了Server2上的Consul Server节点为Leader。这些服务器上最好只部署Consul程序，以尽量维护Consul Server的稳定。

然后在服务器Server4和Server5上通过Consul Client分别注册Service A、B、C，这里每个Service分别部署在了两个服务器上，这样可以避免Service的单点问题。服务注册到Consul可以通过HTTP API（8500端口）的方式，也可以通过Consul配置文件的方式。Consul Client可以认为是无状态的，它将注册信息通过RPC转发到Consul Server，服务信息保存在Server的各个节点中，并且通过Raft实现了强一致性。

最后在服务器Server6中Program D需要访问Service B，这时候Program D首先访问本机Consul Client提供的HTTP API，本机Client会将请求转发到Consul Server，Consul Server查询到Service B当前的信息返回，最终Program D拿到了Service B的所有部署的IP和端口，然后就可以选择Service B的其中一个部署并向其发起请求了。如果服务发现采用的是DNS方式，则Program D中直接使用Service B的服务发现域名，域名解析请求首先到达本机DNS代理，然后转发到本机Consul Client，本机Client会将请求转发到Consul Server，Consul Server查询到Service B当前的信息返回，最终Program D拿到了Service B的某个部署的IP和端口。

图中描述的部署架构笔者认为是最普适最简单的方案，从某些默认配置或设计上看也是官方希望使用者采用的方案，比如8500端口默认监听127.0.0.1，当然有些同学不赞同，后边会提到其他方案。

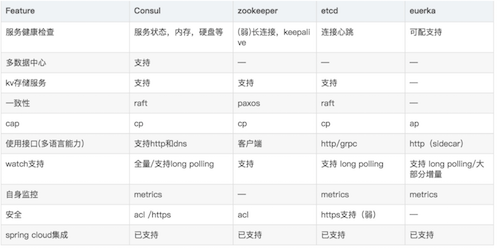
# Consul 的使用场景

1. Docker 实例的注册与配置共享
2. 与 Consul template 服务集成，动态生成 Nginx 和 HAProxy 等配置文件
3. Spring-Cloud-Consul 服务发现和配置文件存储

# Consul 的优势

1. 使用 Raft 算法来保证一致性, 比 ZooKeeper 的 Paxos 算法更简单直接。
2. 支持多数据中心，内外网的服务采用不同的端口进行监听。 ZooKeeper 和 etcd 均不提供多数据中心功能的支持。
3. 支持健康检查，etcd 不提供此功能。
4. 支持 http 和 dns 协议接口。ZooKeeper 的集成较为复杂，etcd 只支持 http 协议。
5. 官方提供 web 管理界面，etcd 无此功能。
6. Consul 1.2 新增 Service Mesh 解决方案。

# 与其他注册中心比较



# 启动agent

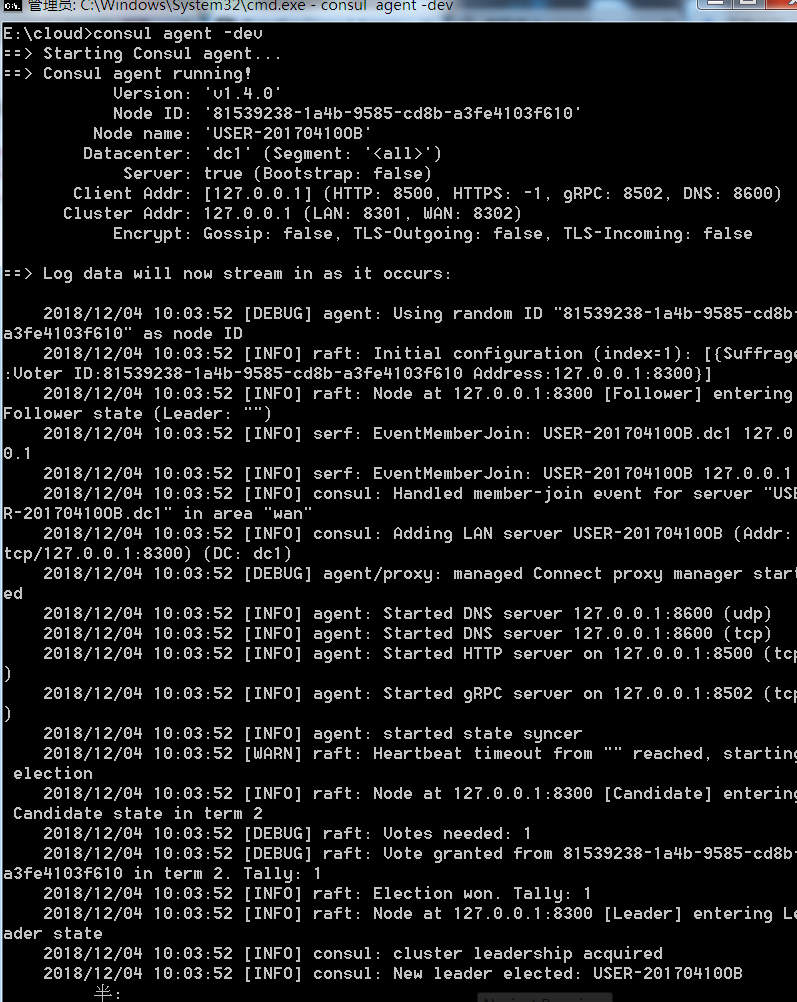
下载完Consul后，必须要先启动agent，agent有server和client两种模式，每个数据中心必须至少有一台agent服务器，但是建议集群中有3台或是5台server服务器，不建议单个服务器部署；

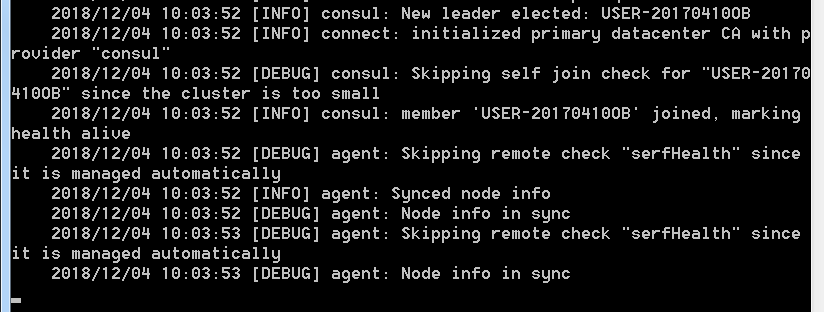
所有其他agent都以客户端模式运行，客户端是一个非常轻量级的流程，可以注册服务、运行健康检查，并将查询结果转发到服务器，代理必须在集群的每个节点上运行；

## 启动agent

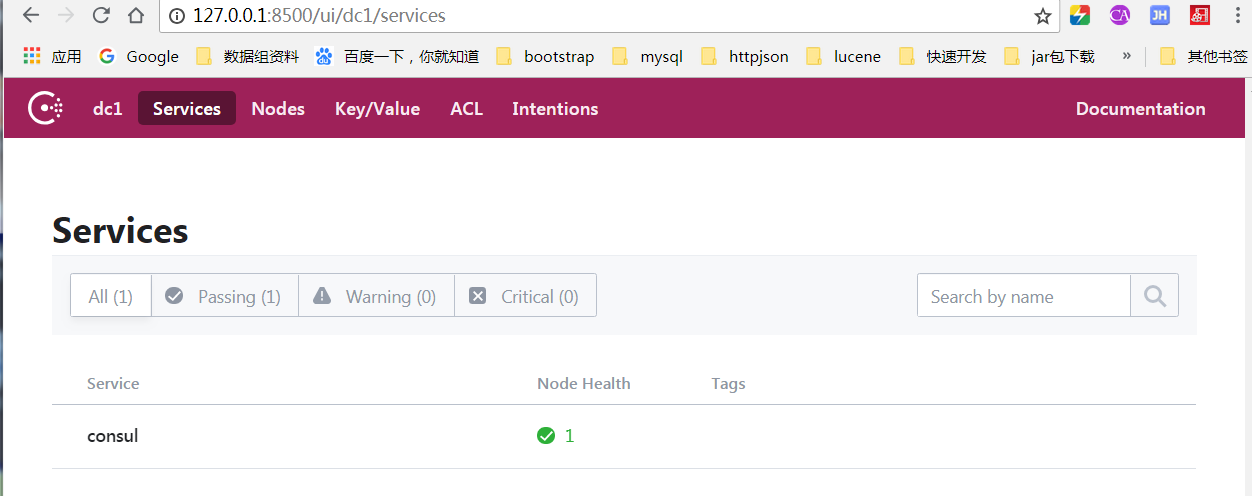
启动开发模式：consul agent -dev

为了简单起见，我们在开发模式中启动agent，该模式有助于快速、轻松地生成单节点agent环境，开发模式不适合于生产环境，因为它不保存任何状态；



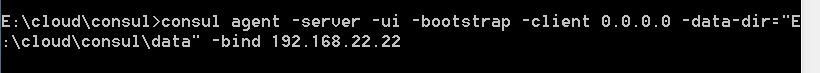


这时，在浏览器输入<http://127.0.0.1:8500>即可访问

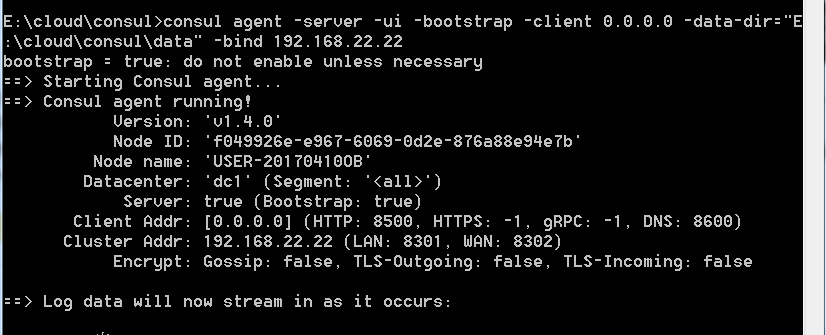


备注：本机IP是192.168.22.22，如果通过192.168.22.22:8500访问consul，则访问不成功，需要通过以下命名启动才可以通过IP访问：

先创建data文件夹，然后执行以下命令：

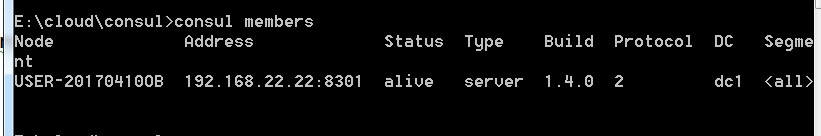


consul agent -server -ui -bootstrap -client 0.0.0.0 -data-dir="E:\cloud\consul\data" -bind 192.168.22.22



## 集群成员

命令：consul members



Members命令是基于gossip协议，最终是一致的，但是本地看到的信息可能与服务器上的状态不完全匹配，要获得一致性的信息，可以通过HTTP API将请求转发给Consul服务器：http://localhost:8500/v1/catalog/nodes



## 停止agent

在cmd窗口使用Ctrl+C，可以正常暂停代理，Consul会通知其他集群成员，该节点离开，会从目录中删除其服务和检查；

如果直接关闭CMD窗口强制离开，则集群的其他成员将检测到该节点已失败，其健康状况仅标记为严重，但不会从目录中删除，Consul将自动尝试重新连接到故障节点，允许它从某些网络状况回复；

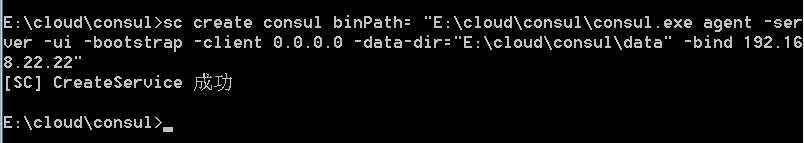
# 注册服务

## 注册系统服务命令

E:\cloud\consul>sc create consul binPath= "E:\cloud\consul\consul.exe agent -ser

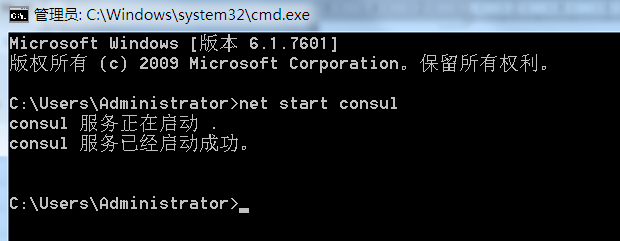
ver -ui -bootstrap -client 0.0.0.0 -data-dir="E:\cloud\consul\data" -bind 192.16

8.22.22"



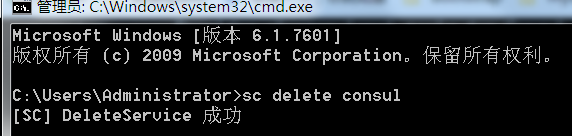
## 通过服务启动

net start consule



## 删除服务

sc delete consul



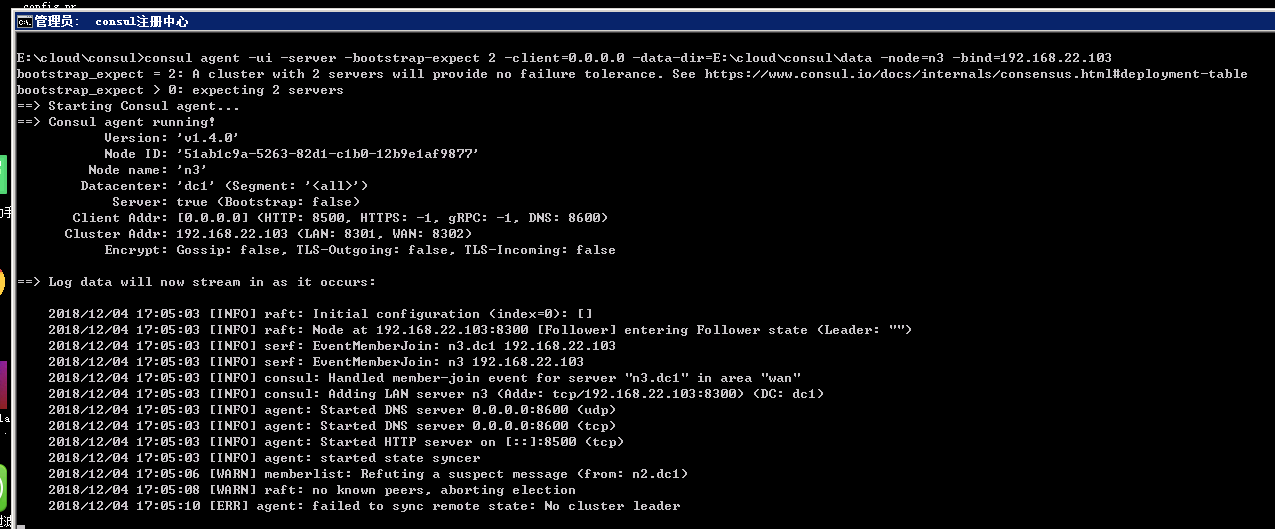
# 集群搭建

## 选3台电脑做server

Server：192.168.22.108,192.168.22.105,192.168.22.103

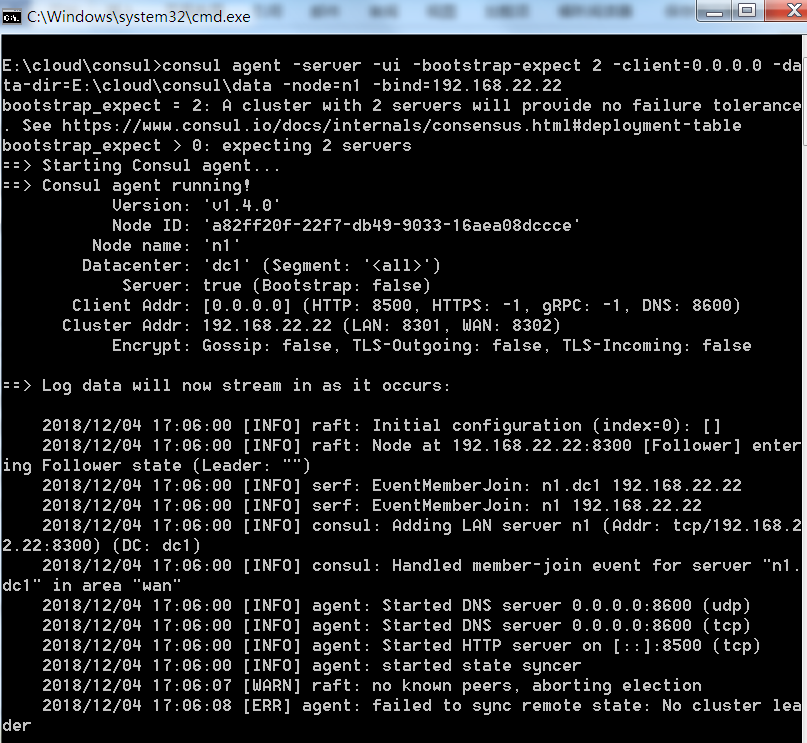
Client：192.168.22.106

在22.103电脑执行以下命令：consul agent –server -bootstrap-expect 3 -client=0.0.0.0 -data-dir=E:\cloud\consul\data -node=n3 -bind=192.168.22.103



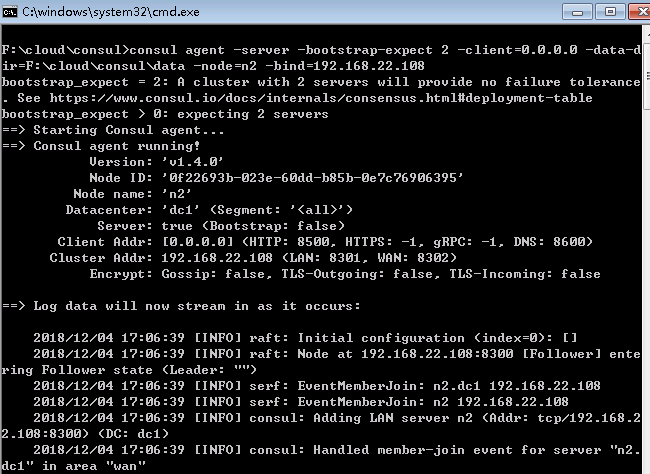
在22.105电脑执行以下命令：

consul agent –server -bootstrap-expect 3 -client=0.0.0.0 -data-dir=E:\cloud\consul\data -node=n1 -bind=192.168.22.105



在22.108电脑执行以下命令：

consul agent –server -bootstrap-expect 3 -client=0.0.0.0 -data-dir=F:\cloud\consul\data -node=n2 -bind=192.168.22.108



现在启动了3台server，但是还没有join，所以都是独立的，不是集群：

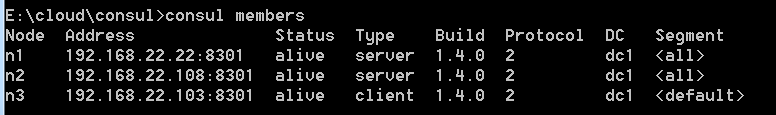


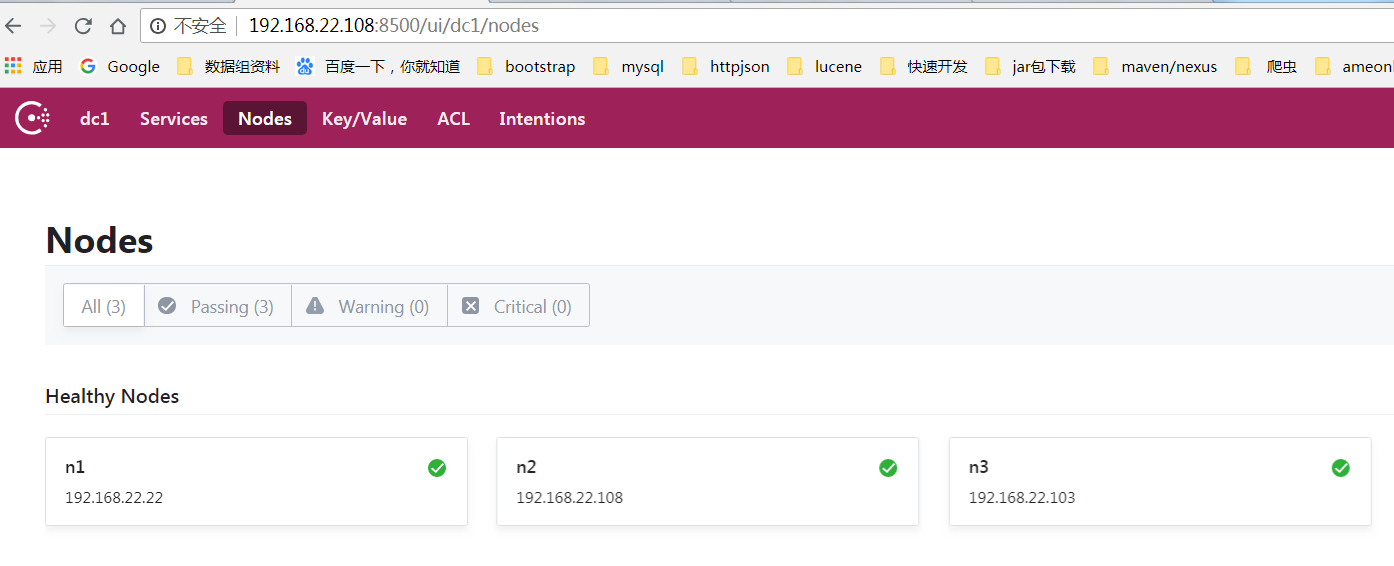
分别在22.22和22.108服务器上join 22.103，命令如下：

consol join 192.168.22.103



查看集群：





可以通过consul info查看当前节点是否为主节点；

命令详解：

-ui表示可以通过<http://192.168.22.22:8500/ui>访问web界面，不加此命令，则没有web界面；

-server： 以server身份启动。  
-bootstrap-expect：集群要求的最少server数量，当低于这个数量，集群即失效。  
-data-dir：data存放的目录，更多信息请参阅consul数据同步机制  
-node：节点id，在同一集群不能重复。  
-bind：监听的ip地址。  
-client 客户端的ip地址

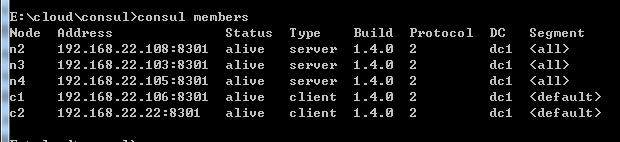
-datacenter 表示数据中心名字

## 选一台电脑做client

consul agent -ui -client=0.0.0.0 -data-dir=F:\cloud\consul\data -node=c1 -bind=192.168.22.22

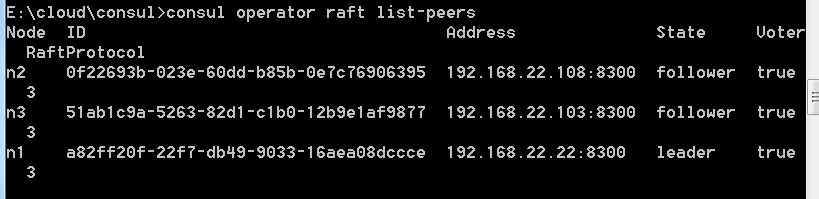
只要是有部署微服务的服务器，都要启动client，client端加-ui；





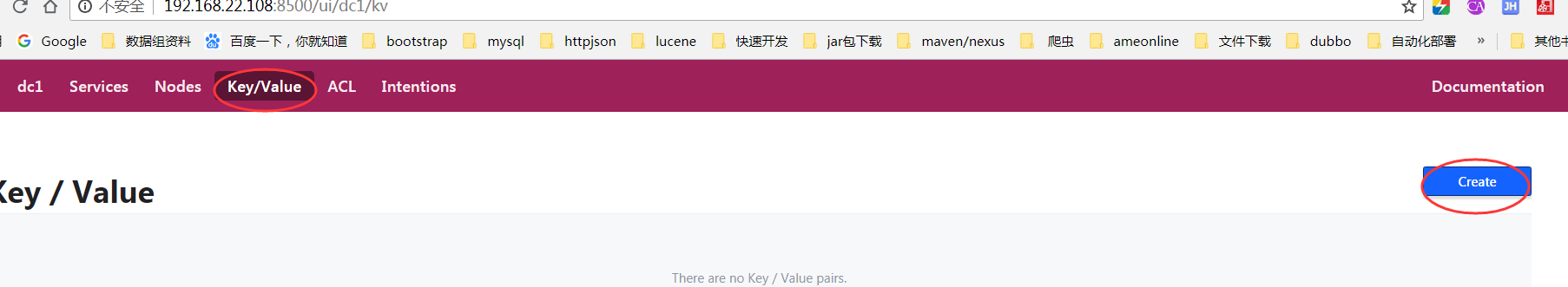
## 查看集群状态

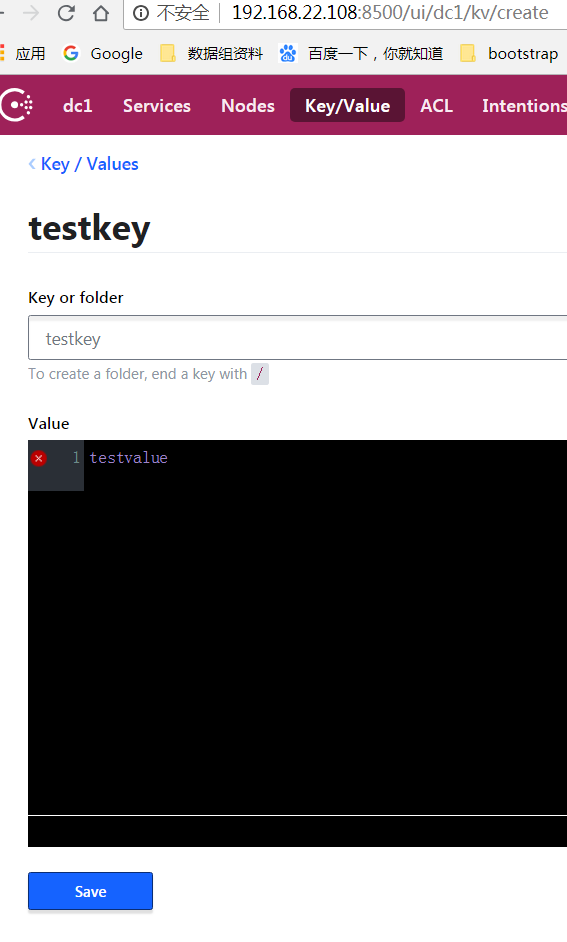
consul operator raft list-peers

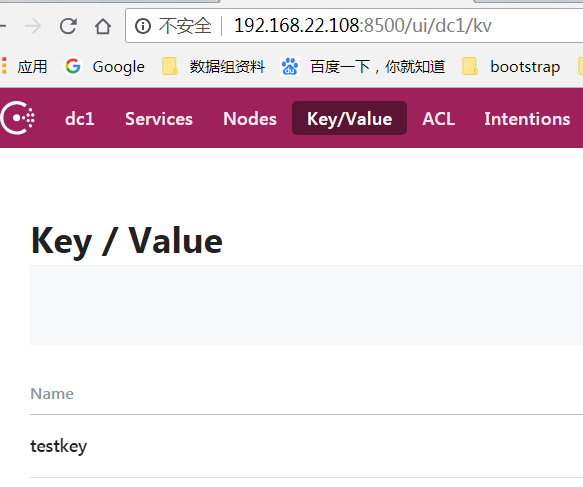


## 配置key/value

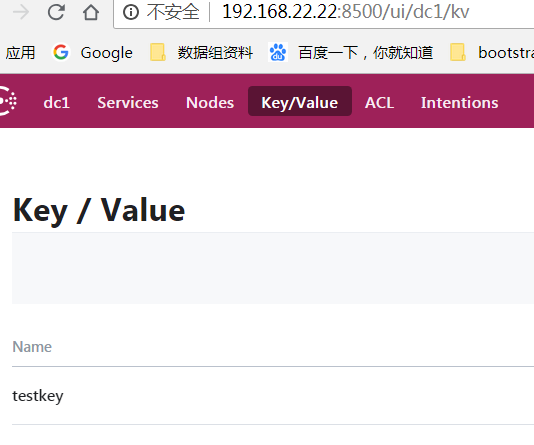
在108服务器上配置key/value：

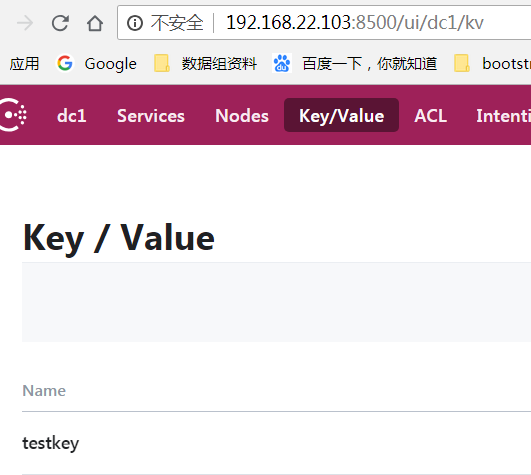






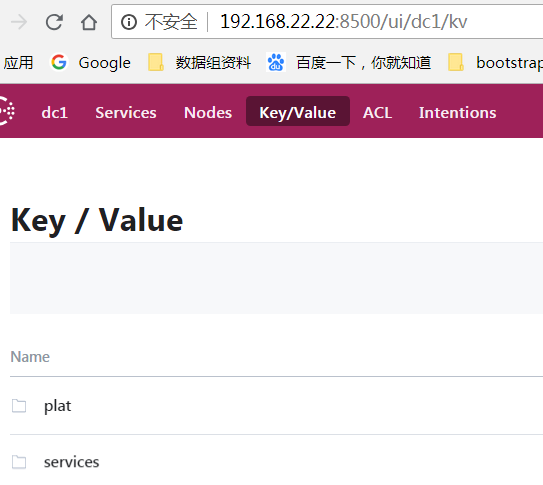
在108上配置key/value，现在到105和103服务器看是否同步添加了key/value：

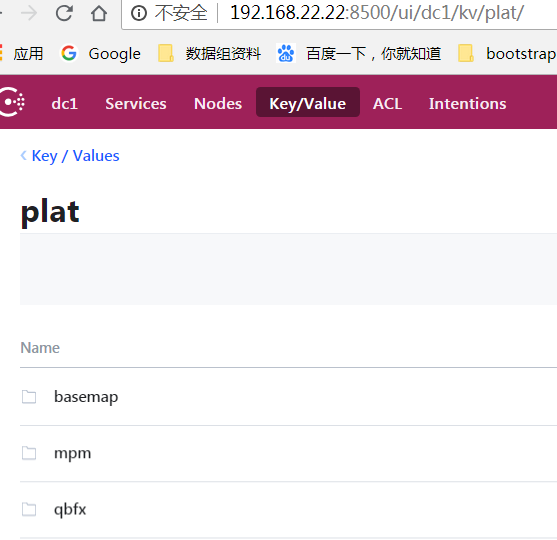




由此可见集群key/value环境搭建成功；

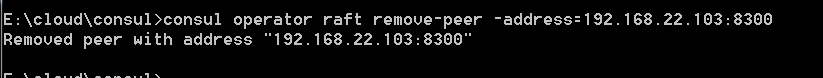
实际工作中，数工部创建key的方式为plat/平台名称/，services/服务名/





## 删除集群中的某一台服务器

consul operator raft remove-peer -address=192.168.22.103:8300



# 常用命令

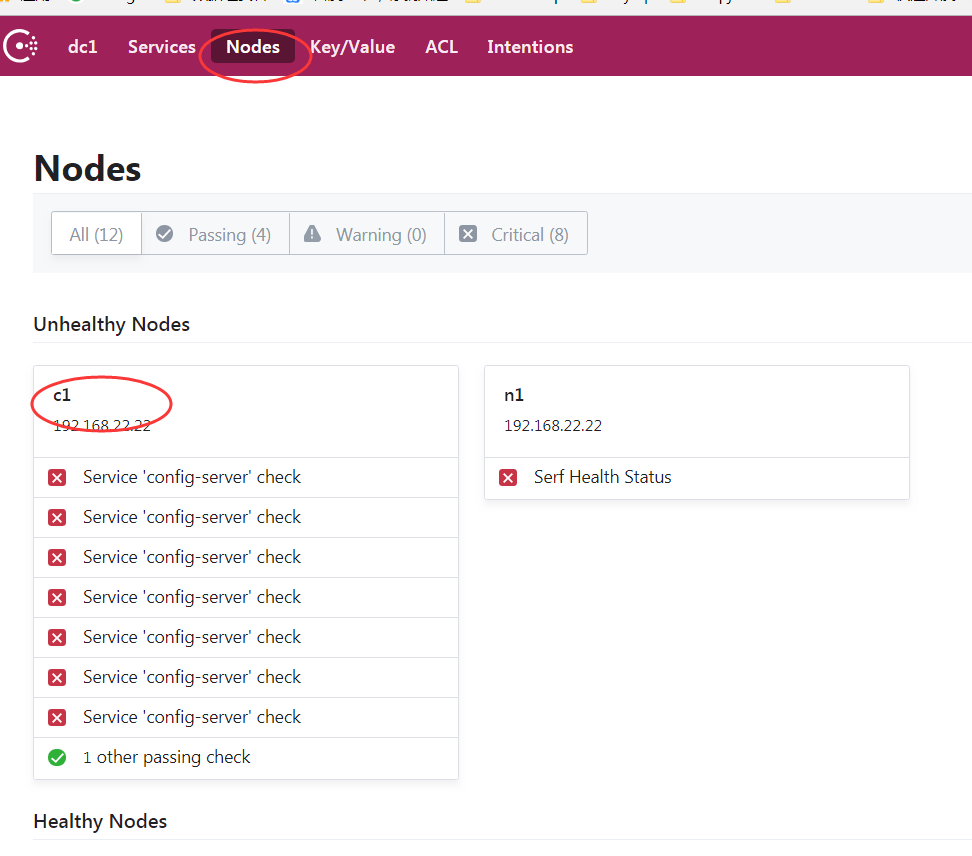
## 服务

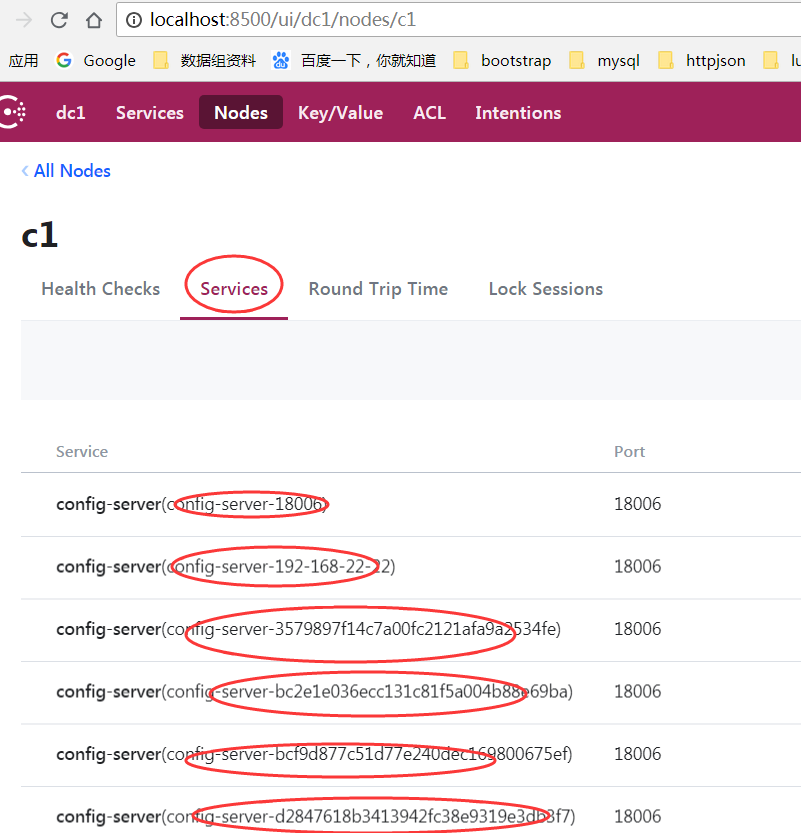
服务API: https://www.consul.io/api/agent/service.html

获取所有服务：<http://192.168.22.103:8500/v1/agent/services>

## 删除服务

如何找服务ID：



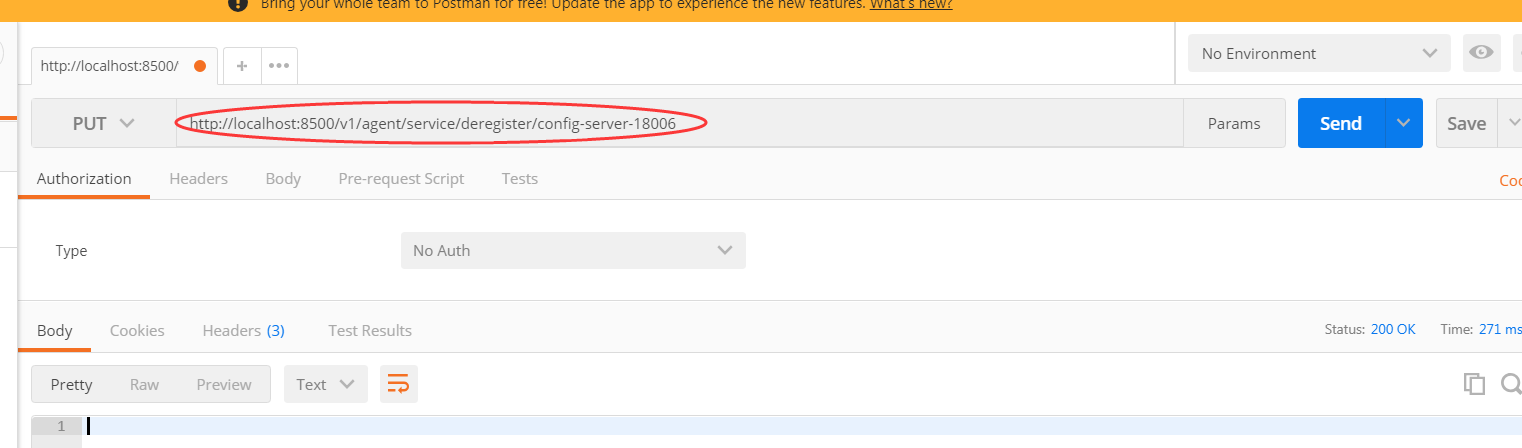


命令行方式删除：

consul services deregister -id=config-server-15958c2c4019f08f9c034080d6e106ae



利用postman发送PUT请求删除：



备注：如果有的服务删除不了，就找到该服务属于哪个节点服务器，然后把localhost改成对应节点服务器的IP即可；

## 服务链接

/v1/agent/checks : 返回本地agent注册的所有检查(包括配置文件和HTTP接口)

/v1/agent/services : 返回本地agent注册的所有 服务

/v1/agent/members : 返回agent在集群的gossip pool中看到的成员

/v1/agent/self : 返回本地agent的配置和成员信息

/v1/agent/join/<address> : 触发本地agent加入node

/v1/agent/force-leave/<node>>: 强制删除node

/v1/agent/check/register : 在本地agent增加一个检查项，使用PUT方法传输一个json格式的数据

/v1/agent/check/deregister/<checkID> : 注销一个本地agent的检查项

/v1/agent/check/pass/<checkID> : 设置一个本地检查项的状态为passing

/v1/agent/check/warn/<checkID> : 设置一个本地检查项的状态为warning

/v1/agent/check/fail/<checkID> : 设置一个本地检查项的状态为critical

/v1/agent/service/register : 在本地agent增加一个新的服务项，使用PUT方法传输一个json格式的数据

/v1/agent/service/deregister/<serviceID> : 注销一个本地agent的服务项