**HisMicroserviceSample**

**开发文档**

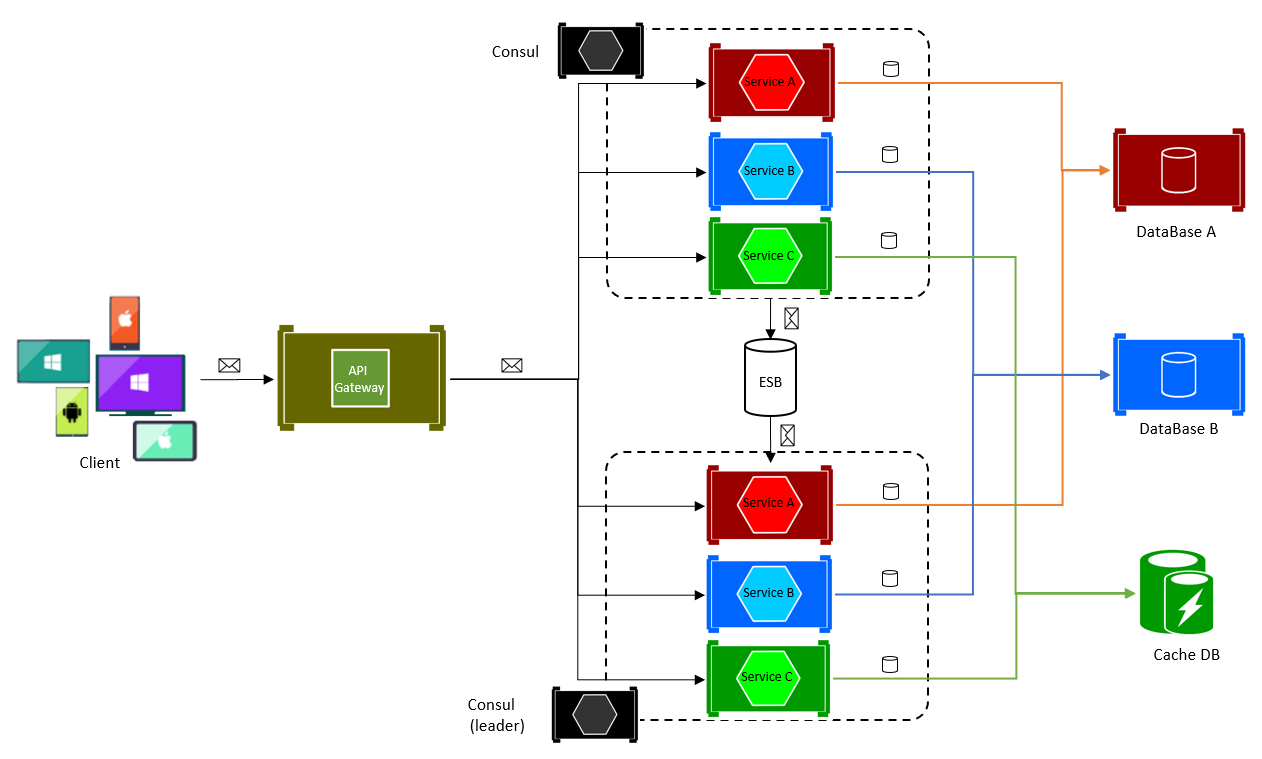
（版本 v1.0.0.0）

大连医卫信息技术有限公司

**文档变更记录**

|  |  |  |
| --- | --- | --- |
| 更改人 | 日期 | 更改内容 |
| 桂素伟 | 2017-12-26 | 创建文档 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 系统架构图



# 技术资料

## Consul（https://www.consul.io/）

### Consul说明

是一个服务管理软件。支持多数据中心下，分布式高可用的，服务发现和配置共享。consul支持健康检查，允许存储键值对。一致性协议采用 Raft 算法,用来保证服务的高可用。成员管理和消息广播 采用GOSSIP协议，支持ACL访问控制。

**服务注册**：一个服务将其位置信息在“中心注册节点”注册的过程。该服务一般会将它的主机IP地址以及端口号进行注册，有时也会有服务访问的认证信息，使用协议，版本号，以及关于环境的一些细节信息。

**服务发现**：服务发现可以让一个应用或者组件发现其运行环境以及其它应用或组件的信息。用户配置一个服务发现工具就可以将实际容器跟运行配置分离开。常见配置信息包括：ip、端口号、名称等。

### 述语

#### 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。由于每个节点都必须运行agent，所有节点要么是client要么是server。所有的Agent都可以调用DNS或HTTP API，并负责检查和维护服务同步。

#### client

运行client模式的Agent，将所有的RPCs转发到Server。Client是相对无状态的。Client唯一所做的是在后台参与LAN gossip pool。只消耗少量的资源，少量的网络带宽。

#### Server

运行Server模式的Agent，参与Raft quorum，维护集群的状态，响应RPC查询，与其他数据中心交互WAN gossip，转发查询到Leader或远程数据中心。

#### Datacenter

数据中心的定义似乎是显而易见的，有一些细节是必须考虑的。例如，在EC2，多个可用性区域是否被认为组成了单一的数据中心？我们定义数据中心是在同一个网络环境中——私有的，低延迟，高带宽。这不包括基于公共互联网环境，但是对于我们而言，在同一个EC2的多个可用性区域会被认为是一个的数据中心。

#### Consensus

当本系列文档中，consensus，意味着leader election协议，以及事务的顺序。由于这些事务是基于一个有限状态机，consensus的定义意味着复制状态机的一致性。

#### Gossip – consul

是建立在Serf之上，提供了完成的Gossip协议，用于成员维护故障检测、事件广播。详细细节参见[gossip documentation](https://www.consul.io/docs/internals/gossip.html" \t "_blank)。这足以知道gossip是基于UDP协议实现随机的节点到节点的通信，主要是在UDP。

#### 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。

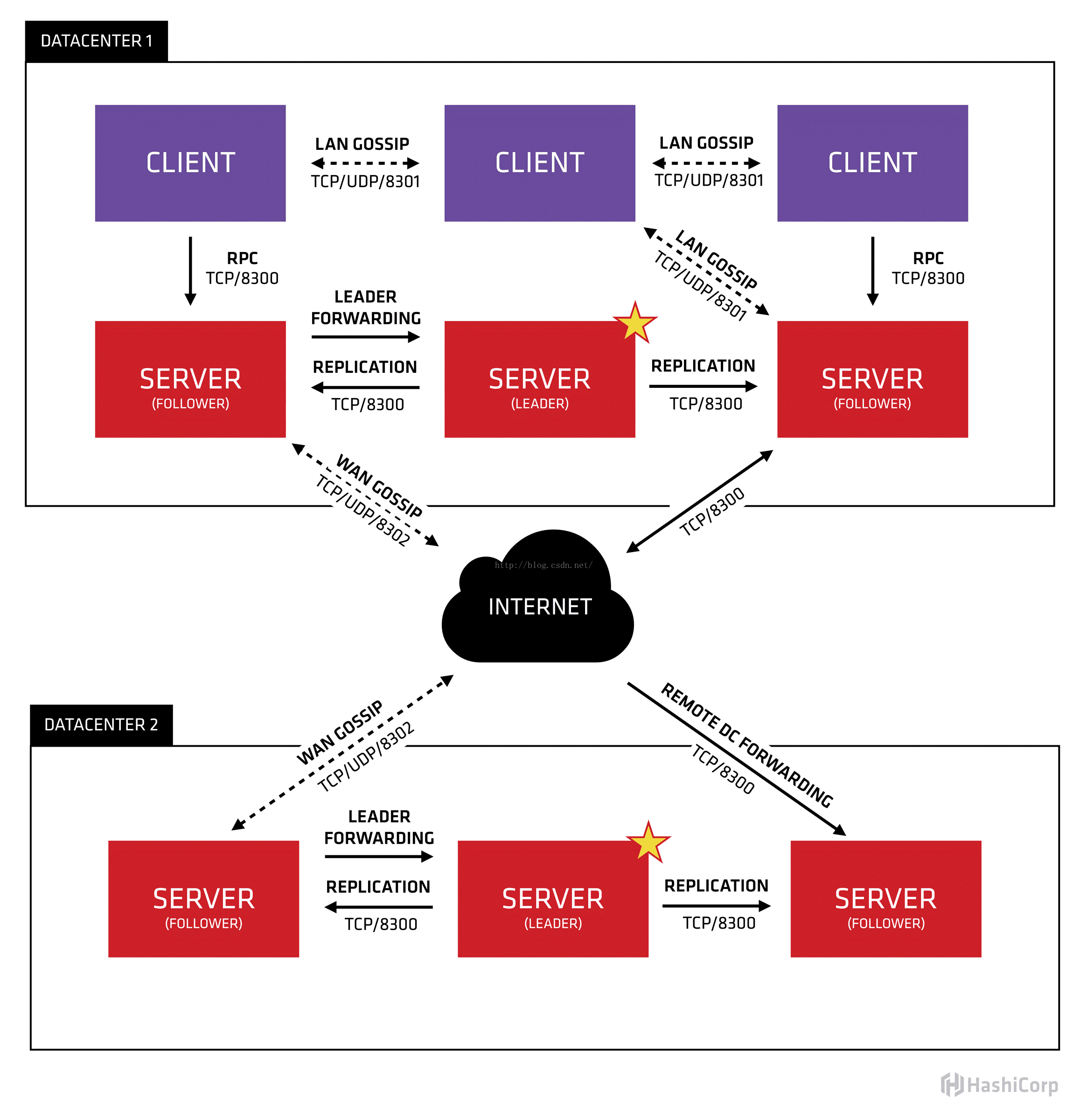
#### WAN Gossip

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。

### 部署结构图



### 命令

#### -advertise

通知展现地址用来改变我们给集群中的其他节点展现的地址，一般情况下-bind地址就是展现地址

#### -bootstrap

用来控制一个server是否在bootstrap模式，在一个datacenter中只能有一个server处于bootstrap模式，当一个server处于bootstrap模式时，可以自己选举为raft leader。

#### -bootstrap-expect

在一个datacenter中期望提供的server节点数目，当该值提供的时候，consul一直等到达到指定sever数目的时候才会引导整个集群，该标记不能和bootstrap公用

#### -bind

该地址用来在集群内部的通讯，集群内的所有节点到地址都必须是可达的，默认是0.0.0.0

#### -client

consul绑定在哪个client地址上，这个地址提供HTTP、DNS、RPC等服务，默认是127.0.0.1

#### -config-file

明确的指定要加载哪个配置文件

#### -config-dir

配置文件目录，里面所有以.json结尾的文件都会被加载

#### -data-dir

提供一个目录用来存放agent的状态，所有的agent允许都需要该目录，该目录必须是稳定的，系统重启后都继续存在

#### -datacenter

该标记控制agent允许的datacenter的名称，默认是dc1

#### -encrypt

指定secret key，使consul在通讯时进行加密，key可以通过consul keygen生成，同一个集群中的节点必须使用相同的key

#### -join

加入一个已经启动的agent的ip地址，可以多次指定多个agent的地址。如果consul不能加入任何指定的地址中，则agent会启动失败，默认agent启动时不会加入任何节点。

#### -retry-join

和join类似，但是允许你在第一次失败后进行尝试。

#### -retry-interval

两次join之间的时间间隔，默认是30s

#### -retry-max

尝试重复join的次数，默认是0，也就是无限次尝试

#### -log-level

consul agent启动后显示的日志信息级别。默认是info，可选：trace、debug、info、warn、err。

#### -node

节点在集群中的名称，在一个集群中必须是唯一的，默认是该节点的主机名

#### -protocol

consul使用的协议版本

#### -rejoin

使consul忽略先前的离开，在再次启动后仍旧尝试加入集群中。

#### -server

定义agent运行在server模式，每个集群至少有一个server，建议每个集群的server不要超过5个

#### -syslog

开启系统日志功能，只在linux/osx上生效

#### -ui-dir

提供存放web ui资源的路径，该目录必须是可读的

#### -pid-file

提供一个路径来存放pid文件，可以使用该文件进行SIGINT/SIGHUP(关闭/更新)agent

### 常用API

consul的主要接口是RESTful HTTP API，该API可以用来增删查改nodes、services、checks、configguration。所有的endpoints主要分为以下类别：

**kv** - Key/Value存储

**agent** - Agent控制

**catalog** - 管理nodes和services

**health** - 管理健康监测

**session** - Session操作

**acl** - ACL创建和管理

**event** - 用户Events

**status** - Consul系统状态

#### agent endpoints：agent endpoints用来和本地agent进行交互，一般用来服务注册和检查注册，支持以下接口

/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

#### catalog endpoints：catalog endpoints用来注册/注销nodes、services、checks

/v1/catalog/register : Registers a new node, service, or check

/v1/catalog/deregister : Deregisters a node, service, or check

/v1/catalog/datacenters : Lists known datacenters

/v1/catalog/nodes : Lists nodes in a given DC

/v1/catalog/services : Lists services in a given DC

/v1/catalog/service/<service> : Lists the nodes in a given service

/v1/catalog/node/<node> : Lists the services provided by a node

#### health endpoints：health endpoints用来查询健康状况相关信息，该功能从catalog中单独分离出来

/v1/healt/node/<node>: 返回node所定义的检查，可用参数?dc=

/v1/health/checks/<service>: 返回和服务相关联的检查，可用参数?dc=

/v1/health/service/<service>: 返回给定datacenter中给定node中service

/v1/health/state/<state>: 返回给定datacenter中指定状态的服务，state可以是"any", "unknown", "passing", "warning", or "critical"，可用参数?dc=

#### session endpoints：session endpoints用来create、update、destory、query sessions

/v1/session/create: Creates a new session

/v1/session/destroy/<session>: Destroys a given session

/v1/session/info/<session>: Queries a given session

/v1/session/node/<node>: Lists sessions belonging to a node

/v1/session/list: Lists all the active sessions

#### acl endpoints：acl endpoints用来create、update、destory、query acl

/v1/acl/create: Creates a new token with policy

/v1/acl/update: Update the policy of a token

/v1/acl/destroy/<id>: Destroys a given token

/v1/acl/info/<id>: Queries the policy of a given token

/v1/acl/clone/<id>: Creates a new token by cloning an existing token

/v1/acl/list: Lists all the active tokens

#### event endpoints：event endpoints用来fire新的events、查询已有的events

/v1/event/fire/<name>: 触发一个新的event，用户event需要name和其他可选的参数，使用PUT方法

/v1/event/list: 返回agent知道的events

#### status endpoints：status endpoints用来或者consul 集群的信息

/v1/status/leader : 返回当前集群的Raft leader

/v1/status/peers : 返回当前集群中同事

### 使用consul

#### 启动

consul agent -server -datacenter=数据中心名称 -bootstrap -data-dir 数据存放路径 -config-file 配置文件路径 -ui-dir UI存放路径 -node=n1 -bind 本机IP

例：

consul agent -server -datacenter=dc1 -bootstrap -data-dir /tmp/consul -config-file ./conf -ui-dir ./dist -node=n1 -bind 127.0.0.1

web.json配置文件如下：

{

"encrypt": "7TnJPB4lKtjEcCWWjN6jSA==",

"services": [

{

"id": "lisapi001",

"name": "lisapi",

"tags": [ "lisapi" ],

"address": "192.168.1.101",

"port": 5002,

"checks": [

{

"id": "lisapicheck001",

"name": "lisapicheck001",

"http": "http://192.168.1.101:5002/health",

"interval": "10s",

"tls\_skip\_verify": false,

"method": "GET",

"timeout": "1s"

},

{

"id": "lisapicheck003",

"name": "lisapicheck003",

"http": "http://192.168.1.101:5002",

"interval": "10s",

"tls\_skip\_verify": false,

"method": "GET",

"timeout": "1s"

}

]

}

]

}

Dist文件： 

#### 查看成员

consul members

#### 查看集群状态

consul operator raft list-peers

#### 加入consul集群

把192.168.1.126加入到当前集群

consul join 192.168.1.126

## 监控App Metrics（https://app-metrics.io）

### Influxdb（https://www.influxdata.com）

#### influxdb介绍

下载InfluxDB：https://portal.influxdata.com/downloads

influxd.exe是主程序，influx.exe是客户端，可以客户端里执行命令。

InfluxDB 是用Go语言编写的一个开源分布式时序、事件和指标数据库，无需外部依赖。

其主要特色功能

基于时间序列，支持与时间有关的相关函数（如最大，最小，求和等）

可度量性：你可以实时对大量数据进行计算

基于事件：它支持任意的事件数据

InfluxDB的主要特点

无结构（无模式）：可以是任意数量的列

可拓展的

支持min, max, sum, count, mean, median 等一系列函数，方便统计

原生的HTTP支持，内置HTTP API

强大的类SQL语法

HTTP API操作地址：http://localhost:8086

#### 常用命令

显示数据库：show databases

新建数据库：create database test

删除数据库：drop database test

使用某个数据库：use test

显示所有表：show measurements

新建表：insert disk\_free,hostname=server01 value=442221834240i 1435362189575692182，其中 disk\_free 就是表名，hostname是索引，value=xx是记录值，记录值可以有多个，最后是指定的时间

删除表：drop measurement disk\_free

增加数据：insert disk\_free,hostname=server01 value=442221834240i 1435362189575692182

查询数据：select \* from disk\_free

修改和删除数据：InfluxDB属于时序数据库，没有提供修改和删除数据的方法。

### Grafana(https://grafana.com/)

#### Grafana介绍

下载：https://grafana.com/get

Grafana 是基于JS开发的，功能齐全的度量仪表盘和图形编辑器，帮助开发人员发现问题的工具。

访问地址：http://localhost:3000，用户名:admin，密码：admin

#### 述语

datasource ：数据的存储源，它定义了将用什么方式来查询数据展示在grafana上面，不同的datasource拥有不同的查询语法，grafana支持多种类型例如es,influxdb,

dashboard ：row的集合

row ：panel的集合

panel ：最小的可视化单位，支持多种数据展示方式，table，graph

playlist ：dashboard的集合，当控制台数量太多时，用来快速在目标群中切换

## 网关Ocelot（https://github.com/TomPallister/Ocelot）

### 介绍

Ocelot的目标是使用.NET运行微服务/面向服务架构，我们需要一个统一的入口进入我们的服务，提供监控、鉴权、负载均衡等机制，也可以通过编写中间件的形式，来扩展Ocelot的功能。  Ocelot是一堆特定顺序的中间件。

## 依赖项目

### DapperPlus

是对Dapper的一层封装，便于集中处理和单元测试。

具体使用方法参见<https://github.com/StackExchange/Dapper>

### Ocelot.ConfigEditor

是对每个API项目api接口的自动获取，并保存到网关的配置文件中。

通过访问http://localhost:5000/create访问api网关内的所有项目，并进行自动配置。

### RouteManage

收集业务项目的API，以便供api网关配置。

此dll下有个ActionHandle.GetActions();可以获取当前项目中全部Controller下的路由

### YWJWTAuthorizePolicy

Api网关项目，统一验证项目，各业务API项目中权限验证的基类。

各项目统一配置文件appsetings.json中：

"Audience": {

"Secret": "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890",

"Issuer": "yiwei",

"Audience": "everone"

},

AddOcelotJwtBearer：在ocelot网关的Startup的ConfigureServices中调用

//读取配置文件

var audienceConfig = Configuration.GetSection("Audience");

//注入OcelotJwtBearer

services.AddOcelotJwtBearer(audienceConfig["Issuer"],audienceConfig["Issuer"], audienceConfig["Secret"], "YiWeiBearer");

AddJTokenBuild：在token生成项目的Startup的ConfigureServices中使用

//读取配置文件

var audienceConfig = Configuration.GetSection("Audience");

//注入OcelotJwtBearer

services.AddJTokenBuild(audienceConfig["Issuer"], audienceConfig["Issuer"], audienceConfig["Secret"], "/api/denied");

AddOcelotPolicyJwtBearer：在业务API应用中的Startup的ConfigureServices调用

//读取配置文件

var audienceConfig = Configuration.GetSection("Audience");

//注入OcelotJwtBearer

services.AddOcelotPolicyJwtBearer(audienceConfig["Issuer"], audienceConfig["Issuer"], audienceConfig["Secret"], "GSWBearer", "Permission", "/lisapi/denied");

//这个集合模拟用户权限表,可从数据库中查询出来

var permission = new List<Permission> {

new Permission { Url="/", Name="admin"},

new Permission { Url="/lisapi/lisuser", Name="admin"}

};

services.AddSingleton(permission);

### ProjectExtension

AOP功能：

RepositoryInterceptorAttribute为仓储层AOP特性类，主要为仓储层各方法添加统一的日志等功能。使用时在仓储接口设置此特性

UnifyJsonAttribute为Controller统一数据返回标准JSON，返回格式为

增删改一般返回：new{result=1,message=”添加成功”}的json对象

带值返回返回：new{result=1,message=””,data=返回的对象或集合}

**注意**：

**result=1时为成功，result=0是为失败，result其他值根具体方法定义**

YiWeiExceptionAttribute为Controller异常时返回SON，返回格式为

{“result”:0,”message”:”异常信息”}

使用案例：

需要在Nuget中添加AspectCore.Extensions.DependencyInjection

在业务项目中的StartUp里

public IServiceProvider ConfigureServices(IServiceCollection services)

{

AddRepository(services);

services.AddMvc();

services.AddDynamicProxy();

return services.BuildAspectCoreServiceProvider();

}

# 项目资料

## 服务(Service)项目

|  |  |  |  |
| --- | --- | --- | --- |
| **服务名称** | **项目名称** | **服务端口** | **服务说明** |
| 认证服务 | AuthenticationService | 6800 | 负责用户认证，通过返回token |
| 基础服务 | BasicService | 6801 | 微服务的基础资料管理 |
| 进销存服务 | InvoicingService | 6802 | 负责药房药局的数据信息管理 |
| 门诊服务 | OutpatientService | 6803 | 门诊业务管理 |
| 住院服务 | InpatientService | 6804 | 住院业务管理 |
| 结算服务 | SettlementService | 6805 | 门诊，住院结算管理 |