Docker+Consul+Registrator

实现服务注册与发现

目录

文档信息

文档作者	房佳亮
文档版本	Version1.0
文档版权	内部资料禁止传播
文档归类	Linux 运维架构师系列
系统环境	CentOS-7.X-x86_64
作者邮箱	crushlinux@163.com
修订信息	2021-03-05
技术交流	

文档约定

[绿色背景] 知识重点
[红色背景] 错误警告
[黄色背景] 注意事项

执行命令

服务注册中心引言

服务注册中心本质上是为了解耦服务提供者和服务消费者。对于任何一个微服务，原则上都应存在或者支持多个提供者，这是由微服务的分布式属性决定的。更进一步，为了支持弹性扩缩容特性，一个微服务的提供者的数量和分布往往是动态变化的，也是无法预先确定的。因此，原本在单体应用阶段常用的静态 LB 机制就不再适用了，需要引入额外的组件来管理微服务提供者的注册与发现，而这个组件就是服务注册中心。

CAP 理论是分布式架构中重要理论

- 一致性(Consistency) (所有节点在同一时间具有相同的数据)
- 可用性(Availability) (保证每个请求不管成功或者失败都有响应)
- 分隔容忍(Partition tolerance) (系统中任意信息的丢失或失败不会影响系统的继续运作)

服务注册中心软件

- ZooKeeper

ZooKeeper 是一个分布式的，开放源码的分布式应用程序协调服务，是 Google 的 Chubby 一个开源的实现，是 Hadoop 和 Hbase 的重要组件。它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、域名服务、分布式同步、组服务等。

ZooKeeper 的目标就是封装好复杂易出错的关键服务，将简单易用的接口和性能高效、功能稳定的系统提供给用户。

- Eureka

Eureka 是基于 REST（Representational State Transfer）服务，主要以 AWS 云服务为支撑，提供服务发现并实现负载均衡和故障转移。我们称此服务为 Eureka 服务。Eureka 提供了 Java 客户端组件，Eureka Client，方便与服务端的交互。客户端内置了基于 round-robin 实现的简单负载均衡。在 Netflix，为 Eureka 提供更为复杂的负载均衡方案进行封装，以实现高可用，它包括基于流量、资源利用率以及请求返回状态的加权负载均衡。

- Etcd

etcd 是一个分布式键值对存储系统，由 coreos 开发，内部采用 raft 协议作为一致性算法，用于可靠、快速地保存关键数据，并提供访问。通过分布式锁、leader 选举和写屏障(write barriers)，来实现可靠的分布式协作。etcd 集群是为高可用、持久化数据存储和检索而准备。

- Consul

Consul 是 HashiCorp 公司推出的开源工具，用于实现分布式系统的服务发现与配置。与其他分布式服务注册与发现的方案比如 Airbnb 的 SmartStack 等相比，Consul 的方案更“一站式”，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、Key/Value 存储、多数据中心方案，不再需要依赖其他工具（比如 ZooKeeper 等）。使用起来也较为简单。Consul 用 Golang 语言实现，因此具有天然可移植性(支持 Linux、windows 和 Mac OS X)； 安装包仅包含一个可执行文件，方便部署，与 Docker 等轻量级容器可无缝配合。

常用的服务发现产品之间的比较

Feature	Consul	zookeeper	etcd	euerka
服务健康检查	服务状态，内存，硬盘等	(弱)长连接，keepalive	连接心跳	可配支持
多数据中心	支持	—	—	—
kv存储服务	支持	支持	支持	—
一致性	raft	paxos	raft	—
cap	ca	cp	cp	ap
使用接口(多语言能力)	支持http和dns	客户端	http/grpc	http（sidecar）
watch支持	全量/支持long polling	支持	支持 long polling	支持 long polling/大部分增量
自身监控	metrics	—	metrics	metrics
安全	acl /https	acl	https支持（弱）	—
spring cloud集成	已支持	已支持	已支持	已支持

Consul 介绍

Consul 是一个分布式，高可用且支持多数据中心的服务发现，配置和编排工具。Consul 支持大规模部署，配置和维护面向服务的体系结构。

Consul 主要功能：

服务发现

通过 DNS 或 HTTP 接口使得消费者发现服务，应用程序可以轻松找到所依赖的服务。

健康检查

防止将请求转发不健康的主机。

键值存储

可以使用分层键/值存储，比如功能标记、动态配置等。

多数据中心

开箱即用，不需要复杂的配置。这就意味这不用建立抽象的逻辑来扩展多个地区。

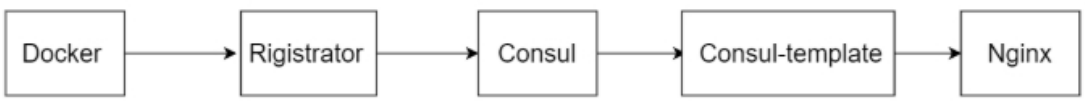
1) Consul agent 是 Consul 核心组件，分为 client 和 server 两种工作模式。
   a) 默认以 client 模式运行，提供服务注册、健康检查、转发查询给 server leader。
   b) server 模式启动时使用-server 选项指定，用于维护 Consul 集群状态、Raft 协议进行选举。
2) agent 必须在每个 Consul 节点运行，所有运行 Consul agent 节点构成 Consul 集群。
3) 官方建议 Consul 集群至少 3 或 5 个节点运行 Consul agent server 模式，client 节点不限。
4) 通过 join 或 rejoin 选项加入集群。一旦加入，集群信息使用 gossip 算法同步到整个集群节点。

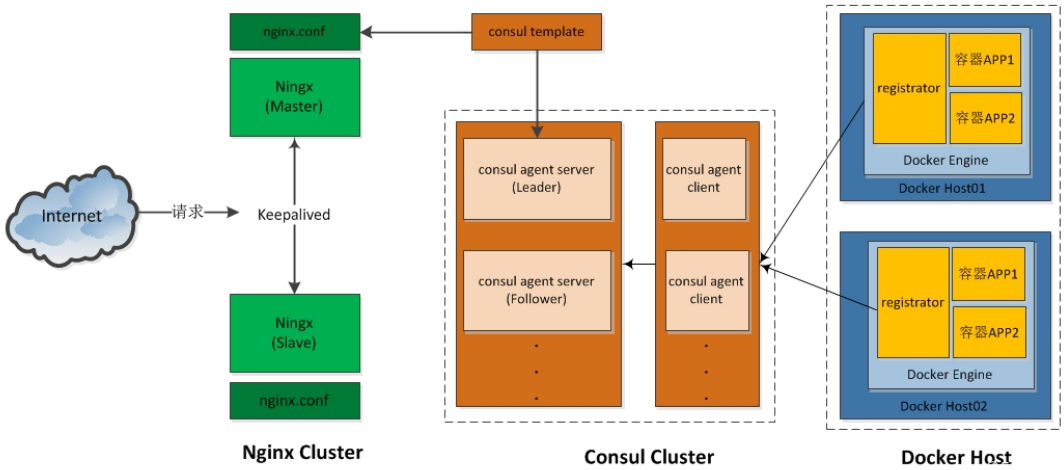
架构设计

在现实工作中，我们一直渴望着追求提供高质量、高可用的服务架构体系，同时减少不必要的部署和维护代价，减少容错率。面对如此高的要求，可以有两种架构方案：

- Docker+Etcd+Confd+Nginx
- Docker+Consul+Nginx

使用 Docker 将 Consul、Consul Template、Registrator 和 Nginx 组装成一个值得信任且可扩展的服务框架，这套架构在这个框架中添加和移除服务，不需要重写任何配置，也不需要重启任何服务，一切都能正常运行。工作流程很简单：





架构优势

Docker+Consul+Nginx 虽然看起来是三个组件的运用，但却证明是一个有机的整体。它们互相联系、互相作用，完全满足我们对高可用、高效服务架构方案的需求，是 Docker 生态圈中最理想的组合之一，具有以下优势：

1. 自动发现与注册组件 consul 使用 Raft 算法来保证一致性，比复杂的 Paxos 算法更直接。相比较而言，zookeeper 采用的是 Paxos，而 etcd 使用的则是 Raft；
2. 支持多数据中心，多数据中心集群可以避免单数据中心的单点故障，zookeeper 和 etcd 均不提供多数据中心功能的支持；
3. 自动、实时发现及无感知服务刷新，具备资源弹性，伸缩自如；
4. 支持健康检查，负载能动态在可用的服务实例上进行均衡，etcd 不提供此功能；
5. 支持足够多台 Docker 容器(前提架构资源足以保证性能支撑)；
6. 支持 http 和 dns 协议接口，zookeeper 的集成较为复杂，etcd 只支持 http 协议；
7. 服务规模方便进行快速调整，官方提供 web 管理界面，etcd 无此功能；
8. Consul template 搭配 consul 使用，支持多种接入层，如 Nginx、Haproxy。

构建自动发现的 Docker 服务架构

1、建立 Consul 服务

要想利用 Consul 提供的服务实现服务的注册与发现，我们需要建立 Consul 服务。在 Consul 方案中，每个提供服务的节点（Docker 主机）上都要部署和运行 Consul 的 client，所有运行 Consul agent 节点的集合构成 Consul Cluster。Consul agent 有两种运行模式：Server 和 Client。 这里的 Server 和 Client 只是 Consul 集群层面的区分，与搭建在 Cluster 之上的应用服务无关。以 Server 模式运行的 Consul agent 节点用于维护 Consul 集群的状态，官方建议每个 Consul Cluster 至少有 3 个或以上的运行在 Server mode 的 Agent，Client 节点不限。

Consul 集群部署

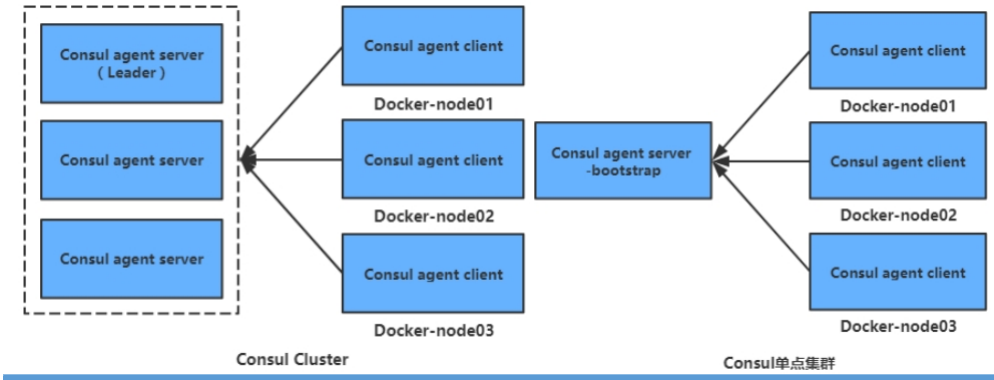
如下图所示，左边三台 Consul agent server 集群通过选举，选出一台 Leader。来管理

右边的 client。获取集群状态信息。

Consul agent server -bootstrap 自动选举本机为 Leader。管理台 client。获取集群信息。

Consul 高可用

- 3 台允许坏 1 台
- 5 台允许坏 2 台



操作系统	主机名	IP 地址	软件
CentOS7.x-1	consul	192.168.200.111	nginx consul consul-template
CentOS7.x-2	docker01	192.168.200.112	docker-ce registrator
CentOS7.x-3	docker02	192.168.200.113	docker-ce registrator

2、安装 consul

下载 consul 二进制包

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

```
[root@consul ~]# unzip consul_0.9.2_linux_amd64.zip
Archive:  consul_0.9.2_linux_amd64.zip
  inflating: consul
[root@consul ~]# mv consul /usr/bin/
[root@consul consul]# nohup consul agent -server -bootstrap -ui -data-dir=/var/lib/consul-data -bind=192.168.200.111 -client=0.0.0.0 -node=consul-server01 &> /var/log/consul.log &
```

配置项说明：

- agent:运行一个 consul 代理。
- -server :切换代理到服务器模式。
- -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。
- -data-dir 参数指定数据存储目录
- -bind -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。

- -ui 参数指定开启 UI 界面，这样可以通过 http://localhost:8500/ui 这样的地址访问 consul 自带的 web UI 界面。
- -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。
- -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。

安装 consul 是用于服务注册，也就是容器本身的一些信息注册到 consul 里面，其他程 序可以通过 consul 获取注册的相关服务信息，这是就是所为的服务注册与发现。

3、查看集群信息

查看 consul 集群成员信息

```
[root@consul consul]# consul members
Node                Address                Status  Type    Build  Protocol  DC
consul-server01     192.168.200.111:8301   alive   server  0.9.2  2         dc1
```

查看 consul 集群中的 leader

```
[root@consul consul]# consul info | grep leader
        leader = true
        leader_addr = 192.168.200.111:8300
```

查看 consul 集群中管理的服务

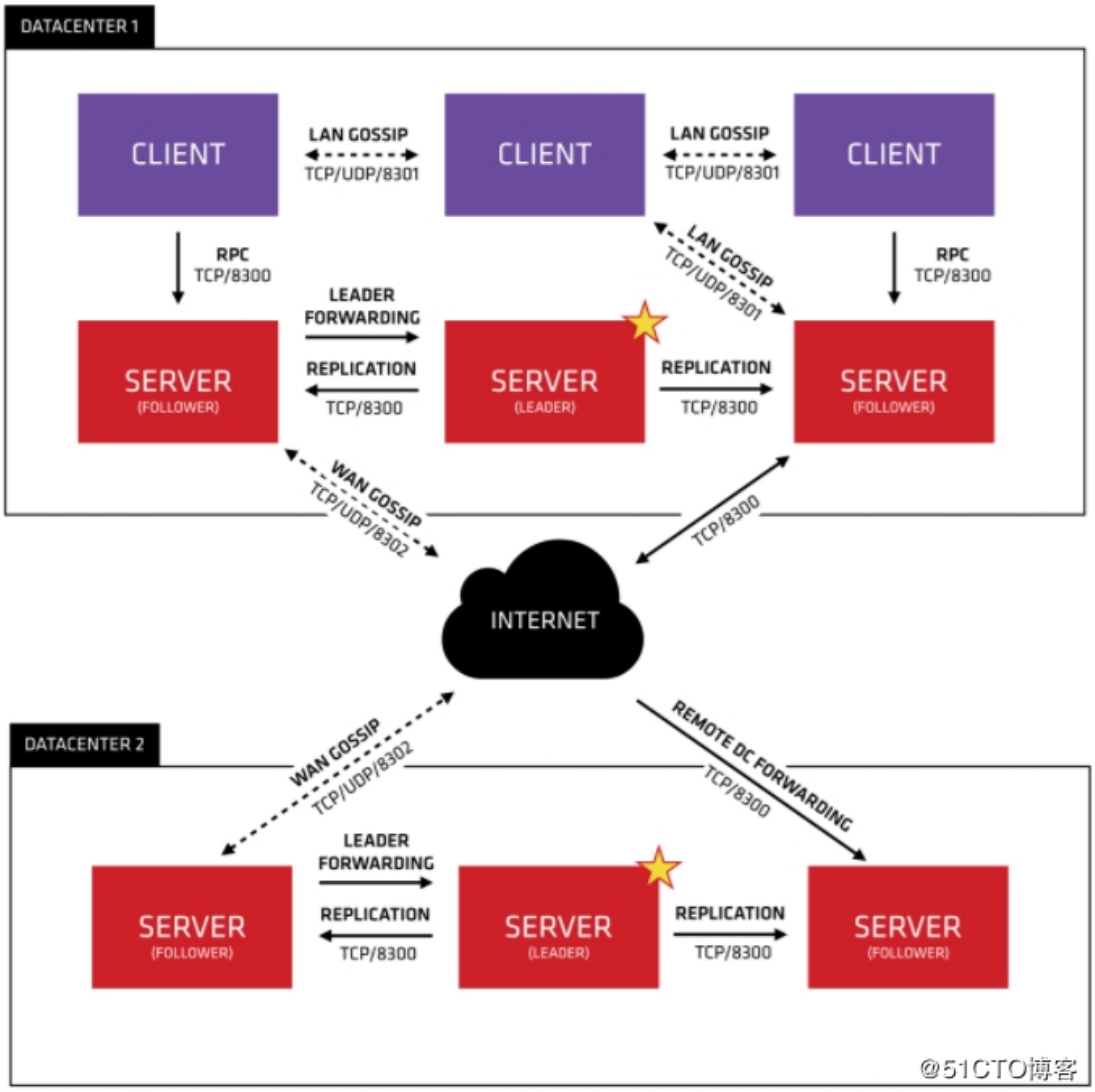
```
[root@consul ~]# consul catalog services
consul
```

```
[root@consul ~]# netstat -lnpt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 192.168.200.111:8300    0.0.0.0:*               LISTEN      1368/consul
tcp        0      0 192.168.200.111:8301    0.0.0.0:*               LISTEN      1368/consul
tcp        0      0 192.168.200.111:8302    0.0.0.0:*               LISTEN      1368/consul
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      680/rpcbind
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      4813/nginx: master
tcp        0      0 0.0.0.0:83              0.0.0.0:*               LISTEN      4813/nginx: master
tcp        0      0 192.168.122.1:53        0.0.0.0:*               LISTEN      1262/dnsmasq
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      996/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      1000/cupsd
tcp        0      0 127.0.0.1:6010          0.0.0.0:*               LISTEN      1286/sshd: root@pts
tcp        0      0 127.0.0.1:6011          0.0.0.0:*               LISTEN      4920/sshd: root@pts
tcp6       0      0 :::111                  :::*                    LISTEN      680/rpcbind
tcp6       0      0 :::8500                 :::*                    LISTEN      1368/consul
tcp6       0      0 :::22                   :::*                    LISTEN      996/sshd
tcp6       0      0 ::1:631                 :::*                    LISTEN      1000/cupsd
tcp6       0      0 :::8600                 :::*                    LISTEN      1368/consul
tcp6       0      0 ::1:6010                :::*                    LISTEN      1286/sshd: root@pts
tcp6       0      0 ::1:6011                :::*                    LISTEN      4920/sshd: root@pts
[root@consul ~]#
```

启动 consul 后默认会监听 5 个端口：

- 8300： replication、leader farwarding 的端口
- 8301： lan cossip 的端口
- 8302： wan gossip 的端口
- 8500： web ui 界面的端口
- 8600： 使用 dns 协议查看节点信息的端口

可参考下图查看端口的意思：



4、通过 API 获取集群信息

```
[root@consul ~]# curl 127.0.0.1:8500/v1/status/peers //查看集群 server 成员
["192.168.200.111:8300"]
[root@consul ~]# curl 127.0.0.1:8500/v1/status/leader //集群 Raf leader
"192.168.200.111:8300"
[root@consul ~]# curl 127.0.0.1:8500/v1/catalog/services //注册的所有服务
{"consul":[]}
[root@consul ~]# curl 127.0.0.1:8500/v1/catalog/nginx //查看 nginx 服务信息
[root@consul ~]# curl 127.0.0.1:8500/v1/catalog/nodes //集群节点详细信息
[{"ID":"6256726f-c53e-ddc6-9b88-f4402a8baf33","Node":"consul-
server01","Address":"192.168.200.111","Datacenter":"dc1","Tagg
edAddresses":{"lan":"192.168.200.111","wan":"192.168.200.111"},"Meta":{},"CreateIndex":5,
"ModifyIndex":6}]
```

容器服务自动加入 Nginx 集群

1. 基于 Registrator 镜像部署容器

Registrator 服务会检查应用服务容器的运行状态，进行自动注册和注销 docker 容器服务到服务配置中心 Consul 上。目前支持与 Consul、etcd 和 SkyDNS2 等软件进行整合。

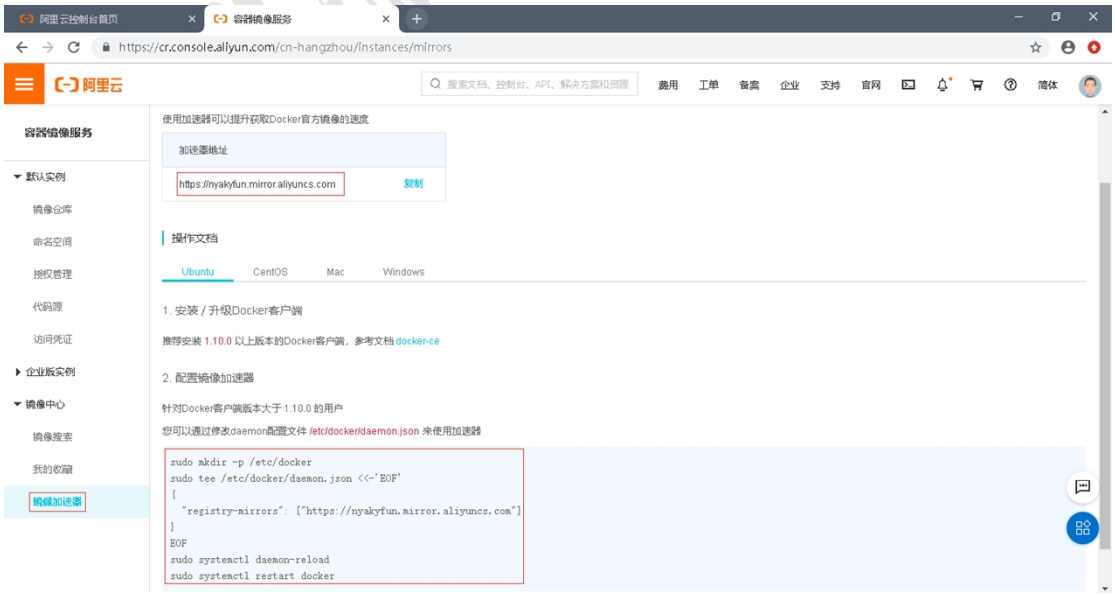
在 192.168.200.112/192.168.200.113 节点操作

1）安装 docker-ce

```
[root@localhost ~]# wget -O /etc/yum.repos.d/CentOS-Base.repo
http://mirrors.aliyun.com/repo/Centos-7.repo

[root@localhost ~]# yum -y install yum-utils device-mapper-persistent-data lvm2
[root@localhost ~]# yum-config-manager --add-repo http://mirrors.aliyun.com/docker-
ce/linux/centos/docker-ce.repo

[root@localhost ~]# ls /etc/yum.repos.d/
backup  Centos-aliyun.repo  CentOS-Media.repo  docker-ce.repo

[root@localhost ~]# yum -y install docker-ce
[root@localhost ~]# systemctl start docker
[root@localhost ~]# systemctl enable docker
```

阿里云镜像加速器

https://cr.console.aliyun.com/




```
[root@localhost ~]# cat << END > /etc/docker/daemon.json
{
        "registry-mirrors":[ "https://nyakyfun.mirror.aliyuncs.com" ]
```

```
}
END
[root@localhost ~]# systemctl daemon-reload
[root@localhost ~]# systemctl restart docker
[root@localhost ~]# docker version
Client: Docker Engine - Community
 Version:           19.03.8
 API version:       1.40
 Go version:        go1.12.17
 Git commit:        afacb8b
 Built:             Wed Mar 11 01:27:04 2020
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.8
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.17
  Git commit:       afacb8b
  Built:            Wed Mar 11 01:25:42 2020
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.13
  GitCommit:        7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc:
  Version:          1.0.0-rc10
  GitCommit:        dc9208a3303feef5b3839f4323d9beb36df0a9dd
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
```

3）部署 registrator 容器服务

注意 192.168.200.113 主机需要注意 IP 地址

```
[root@docker01 ~]# docker run -d --name=registrator --net=host -v
/var/run/docker.sock:/tmp/docker.sock --restart=always gliderlabs/registrator:latest -
ip=192.168.200.112 consul://192.168.200.111:8500

[root@docker02 ~]# docker run -d --name=registrator --net=host -v
/var/run/docker.sock:/tmp/docker.sock --restart=always gliderlabs/registrator:latest -
ip=192.168.200.113 consul://192.168.200.111:8500
```

2. 测试服务发现功能是否正常

创建 Tomcat 测试页面，并部署 tomcat 测试容器

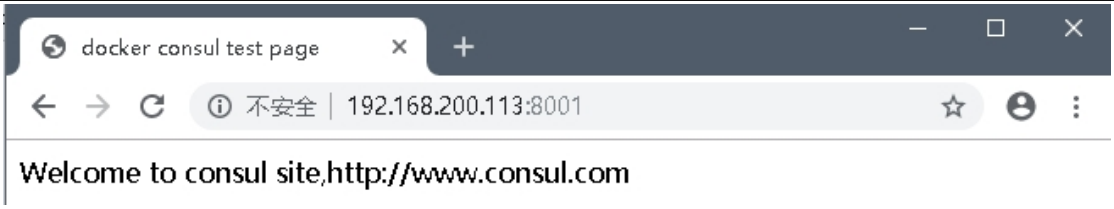
docker01 主机

```
[root@docker01 ~]# mkdir /web
[root@docker01 ~]# vim /web/index.jsp
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<html>
    <head>
        <title>docker consul test page</title>
    </head>
    <body>
        <% out.println("Welcome to consul site,http://www.consul.com");%>
    </body>
</html>
[root@docker01 ~]# docker run -itd -p:8001:8080 -v /web:/usr/local/tomcat/webapps/ROOT --
name docker01-t1 -h docker01-t1 tomcat
[root@docker01 ~]# docker run -itd -p:8002:8080 -v /web:/usr/local/tomcat/webapps/ROOT --
name docker01-t2 -h docker01-t2 tomcat
```

docker02 主机

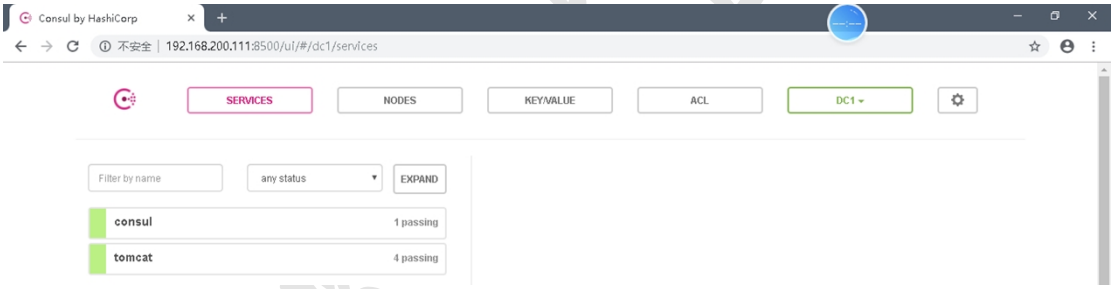
```
[root@docker02 ~]# mkdir /web
[root@docker02 ~]# vim /web/index.jsp
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<html>
    <head>
        <title>docker consul test page</title>
    </head>
    <body>
        <% out.println("Welcome to consul site,http://www.consul.com");%>
    </body>
</html>
[root@docker02 ~]# docker run -itd -p:8001:8080 -v /web:/usr/local/tomcat/webapps/ROOT --
```

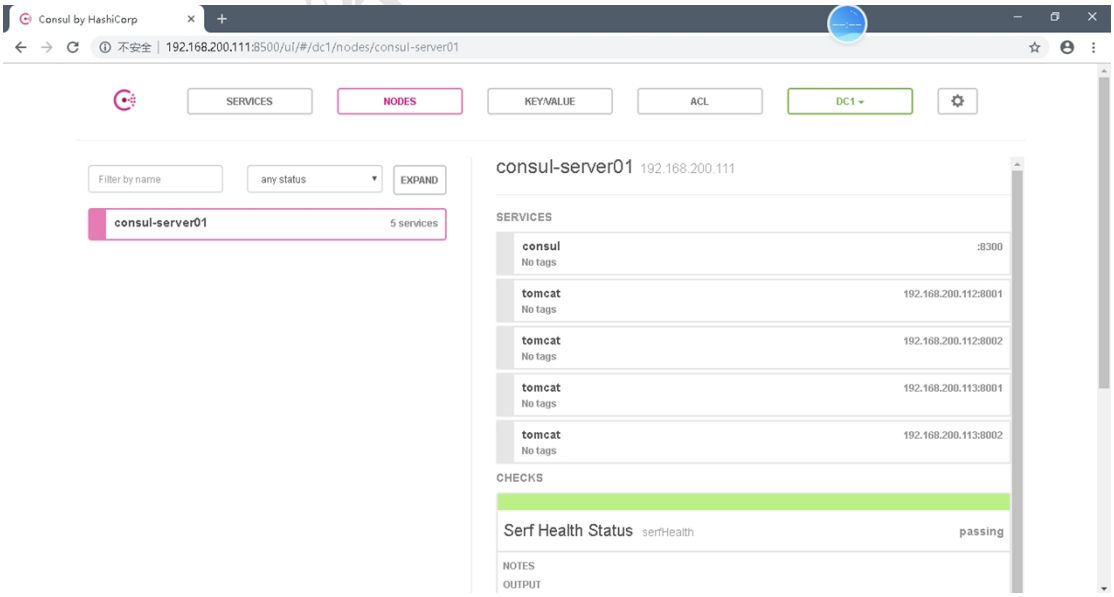
```
name docker02-t1 -h docker02-t1 tomcat
[root@docker02 ~]# docker run -itd -p:8002:8080 -v /web:/usr/local/tomcat/webapps/ROOT --
name docker02-t2 -h docker02-t2 tomcat
```



3. 验证 Tomcat 服务是否注册到了 consul 集群

浏览器输入 http://192.168.200.111:8500，点击”NODES“，再点击“consul-server01””， 会出现 5 个服务。






```
[root@consul ~]# curl 127.0.0.1:8500/v1/catalog/services
{"consul":[],"tomcat":[]}
```

从以上结果中发现 Tomcat 服务已经注册到 consul 里面，说明服务正常。

4. 安装 consul-template

Consul-Template 是基于 Consul 的自动替换配置文件的应用。在 Consul-Template 没出现之前，构建服务大多采用的是 Zookeeper、Etcd+Confd 这样类似的系统。

Consul-template 是一个守护进程，用于实时查询 consul 集群信息，并更新文件系统上的任意数量的指定模板，生成配置文件，更新完成以后可以选择运行 shell 命令执行更新操作，例如：重新加载 nginx。

Consul-Template 可以查询 Consul 中的服务目录、Key、Key-values 等。这种强大的抽象功能和查询语言模板可以使 Consul-Template 特别适合动态的创建配置文件。例如：创建 Apache/Nginx Proxy Balancers、 Haproxy Backends、 Varnish Servers、 Application Configurations 等。

5. 准备 template nginx 模板文件

```
[root@consul ~]# mkdir consul
[root@consul ~]# cd consul/
[root@consul consul]# vim /root/consul/nginx.tmp
upstream http_backend {
    {{range service "tomcat"}}
    server {{ .Address }}:{{ .Port }};
    {{ end }}
}
server {
    listen 8080;
    server_name localhost 192.168.200.111;
    access_log /usr/local/nginx/logs/crushlinux-access.log;
    index index.html index.jsp index.php;
    location / {
        proxy_set_header HOST $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Client-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://http_backend;
    }
}
```

6. 编译安装 nginx

```
[root@consul ~]# yum -y install gcc gcc-c++ make pcre-devel    zlib-devel openssl-devel
[root@consul ~]# tar xf nginx-1.12.1.tar.gz
[root@consul ~]# cd nginx-1.12.1/
[root@consul nginx-1.12.0]# ./configure --prefix=/usr/local/nginx --with-
```

```
http_stub_status_module --with-http_realip_module --with-pcre --with-http_ssl_module &&
make -j 2 && make install
```

7. 配置 nginx

```
[root@consul ~]# vim /usr/local/nginx/conf/nginx.conf
include vhost/*.conf;    //在 http 段里面添加虚拟主机目录
[root@consul ~]# mkdir /usr/local/nginx/conf/vhost/
[root@consul ~]# /usr/local/nginx/sbin/nginx
```

8. 配置并启动 consul-template

手动上传 consul-template_0.19.3_linux_amd64.zip 包到/root 目录下

```
[root@consul ~]# https://releases.hashicorp.com/consul-template/0.19.3/consul-
template_0.19.3_linux_amd64.zip
[root@consul ~]# unzip consul-template_0.19.3_linux_amd64.zip
Archive:  consul-template_0.19.3_linux_amd64.zip
  inflating: consul-template
[root@consul ~]# mv consul-template /usr/bin/
```

在前台启动 template 服务，需要指定 template 模板文件及生成路径即可，启动后不要按 ctrl+c 中止。

```
[root@consul ~]# consul-template -consul-addr 192.168.200.111:8500 -template
"/root/consul/nginx.tmp:/usr/local/nginx/conf/vhost/crushlinux.conf:/usr/local/nginx/sbin/n
ginx -s reload" --log-level=info

2019/06/03 15:24:22.615243 [INFO] consul-template v0.19.3 (ebf2d3d)
2019/06/03 15:24:22.615276 [INFO] (runner) creating new runner (dry: false, once: false)
2019/06/03 15:24:22.616181 [INFO] (runner) creating watcher
2019/06/03 15:24:22.617030 [INFO] (runner) starting
2019/06/03 15:24:22.617056 [INFO] (runner) initiating run
2019/06/03 15:24:22.623225 [INFO] (runner) initiating run
```

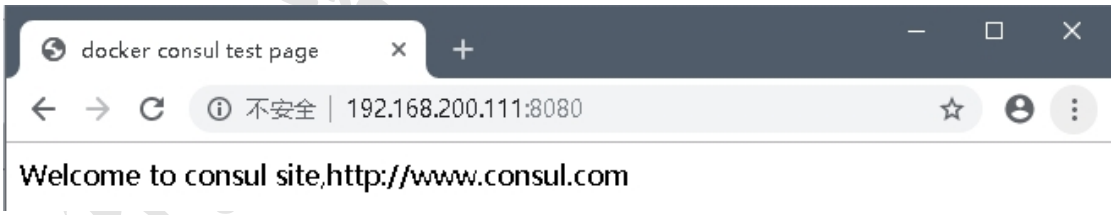
生成的配置文件如下。

```
[root@consul ~]# cat /usr/local/nginx/conf/vhost/crushlinux.conf
upstream http_backend {

    server 192.168.200.112:8001;

    server 192.168.200.112:8002;

    server 192.168.200.113:8001;
```

```
        server 192.168.200.113:8002;

}
server {
        listen 8080;
        server_name localhost 192.168.200.111;
        access_log /usr/local/nginx/logs/crushlinux-access.log;
        index index.html index.php;
        location / {
                proxy_set_header HOST $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header Client-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_pass http://http_backend;
        }
}
[root@consul ~]# netstat -lnpt | grep nginx
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN
66875/nginx: master
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
66875/nginx: master
```

9. 访问 template-nginx 配置文件

通过浏览器访问 nginx 监听的 8080 端口



10. 增加一个测试的容器节点

1）增加一个 tomcat 容器节点，测试服务发现及配置更新功能。

```
[root@docker01 ~]# docker run -itd -p:8003:8080 -v /web:/usr/local/tomcat/webapps/ROOT --
name docker01-t3 -h docker01-t3 tomcat7
ea096462d0283db455d450ad31d730130dd84fff49af685511bc047636b4d6e
```

2）观察 consul-template 服务，他会自动从模板更新 /usr/local/nginx/conf/vhost/crushlinux.conf 文件内容，并且重载 nginx 服务。

```
2020/02/05 08:48:05.742642 [INFO] (runner) initiating run
2020/02/05 08:48:05.744772 [INFO] (runner) rendered "/root/consul/nginx.tmp" =>
```

```
"/usr/local/nginx/conf/vhost/crushlinux.con
f"2020/02/05 08:48:05.744858 [INFO] (runner) executing command
"/usr/local/nginx/sbin/nginx -s reload" from "/root/consul/ngi
nx.tmp" => "/usr/local/nginx/conf/vhost/crushlinux.conf"2020/02/05 08:48:05.745105 [INFO]
(child) spawning: /usr/local/nginx/sbin/nginx -s reload
```

3）查看/usr/local/nginx/conf/vhost/crushlinux.conf 文件内容

```
[root@consul ~]# cat /usr/local/nginx/conf/vhost/crushlinux.conf
upstream http_backend {

    server 192.168.200.112:8001;

    server 192.168.200.112:8002;

    server 192.168.200.112:8003;

    server 192.168.200.113:8001;

    server 192.168.200.113:8002;

}
server {
    listen 8080;
    server_name localhost 192.168.200.111;
    access_log /usr/local/nginx/logs/crushlinux-access.log;
    index index.html index.php;
    location / {
        proxy_set_header HOST $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Client-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://http_backend;
    }
}
```