利用Docker实现

0侵入方式的服务注册与发现

朱清 冰鉴科技高级研发经理

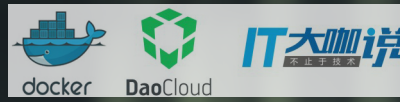
关于我

About Me

- 冰鉴科技信息技术部总监（公司业务：个人和小微企业信用评估，包含全周期风控数据、机器学习模型评估、反欺诈规则引擎）
- Spring Cloud中国社区联合创始人
- 电子科技大学通信学院学士，2015年“成电杰出学生”
- 腾讯视频客厅产品后台开发

目录

CONTENTS



1 项目背景

项目背景

- 2016年第四季度，随着业务规模扩大，单个系统包含的功能不断增长，系统复杂度显著增加，给开发和运维工作带来了很大的挑战，于是我们展开了服务化的研究与探索。

项目背景

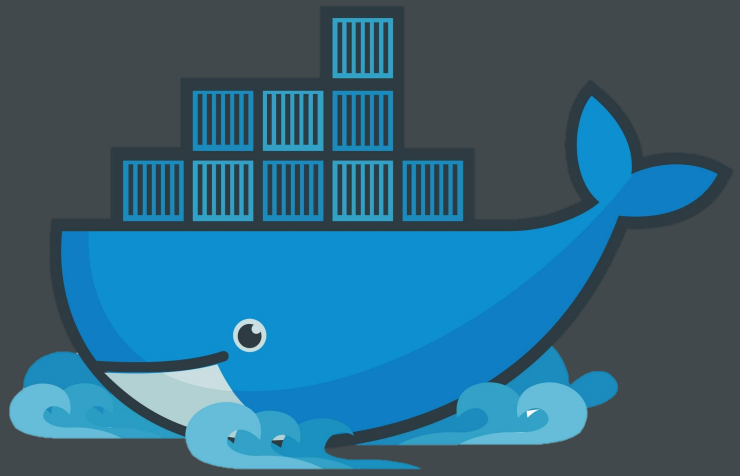
- 我们的业务系统主要用Java编写，并大规模使用了Spring框架。
- 我们的服务主要是基于Spring MVC开发的Web应用。
- 我们已经开始推广使用Spring Boot来提高应用开发效率。

→ 采用Spring Cloud作为我们微服务架构的基础



项目背景

- 在开始服务化以前，我们已经有了一些容器化的实践，并收到了较好的效果；以云主机为主的基础设施也能够较好地发挥容器技术的优势，因此我们决定将微服务架构搭建在Docker容器集群中。



2 技术选型

Consul vs Eureka

技术选型

Consul vs Eureka

Eureka是Netflix的服务注册于发现工具，Spring Cloud Netflix提供了Eureka的服务器和客户端。

优势：

- 完全基于Java实现，部署要求和现有其他服务相近。
- 定制开发更便于Java程序员掌握。
- 和我们选用的其他组件（Zuul, Hystrix,Ribbon）都源自Netflix，具备先天的良好适配性。

不足：

- 需要在服务中添加Eureka的客户端。
- 心跳包机制，服务状态改变不及时。

技术选型

Consul vs Eureka

Consul是HashiCorp出品的服务注册与发现工具，Spring Cloud提供了支持Consul的注册和发现客户端。

优势：

- 服务注册独立于服务本身
- 主动检测服务状态
- 通过Registrator可以提供基于容器的服务注册
- 多数据中心支持

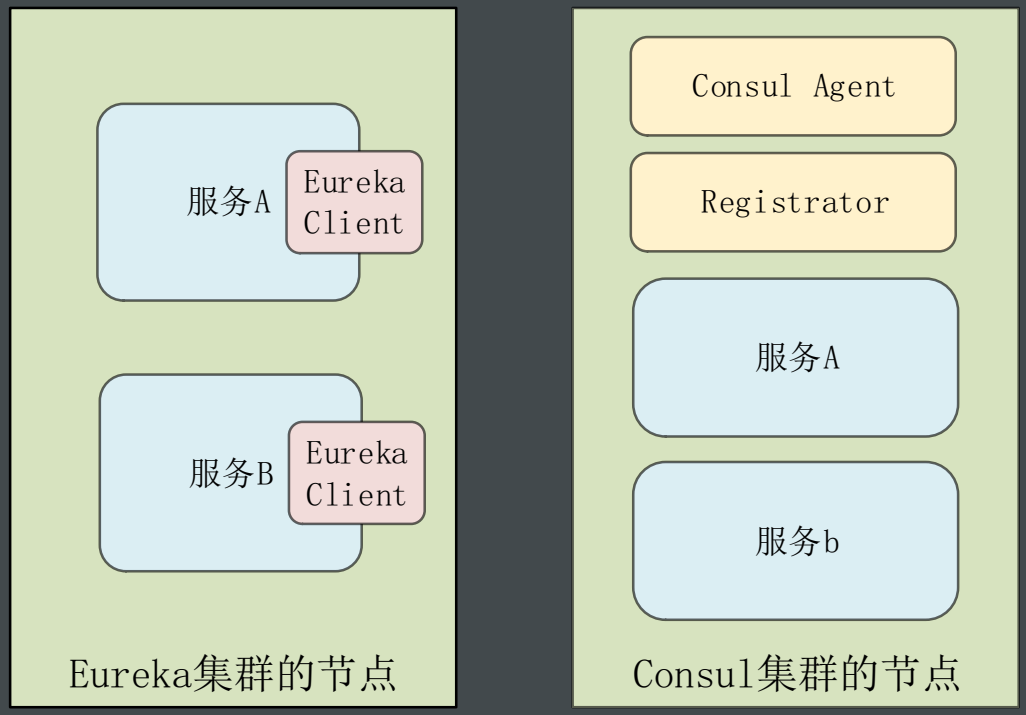
不足：

- Consul用GO语言开发，定制开发需要重新学习
- 需要单独运行，无法直接加入其它组件

技术选型

Consul vs Eureka

Eureka需要在服务内添加Eureka客户端以实现注册和心跳。
Consul的注册由Registrator完成，不需要修改服务。



3 基于容器的服务注册

Registrator

基于容器的服务注册
Registrator

- 服务启动时，需要向注册中心提供自己的信息，包括地址，端口，服务标识符等。然而在Docker容器中，服务一般无法了解自己在宿主机上开放的端口，也无法准确判断该服务的真实网络地址。这就给服务自身完成注册操作造成了一些困难。

- 从另一个方面来看，服务所在的地址和端口等信息其实是和业务无关的，不应该也不适合由服务去处理。否则不仅背离了服务化的初衷，也加深了对服务的侵入。

基于容器的服务注册
Registrator

- Registrator通过监控Docker事件，根据docker容器的启动和停止向注册中心发送注册和注销的消息，无需服务自身参与即可完成。
- Registrator能够掌握服务真正对外开放的端口。
- 即使在服务无响应，甚至直接删除容器的情况下也可以完成对服务的注销。

4 图形化容器集群管理

Rancher

图形化容器集群管理

Rancher

- Rancher是一个图形化的集群管理工具，支持多种集群作为后端，包括Kubernetes、Docker Swarm、Mesos和Rancher内建的Cattle等。

5 Consul+Docker 服务注册与发现实践

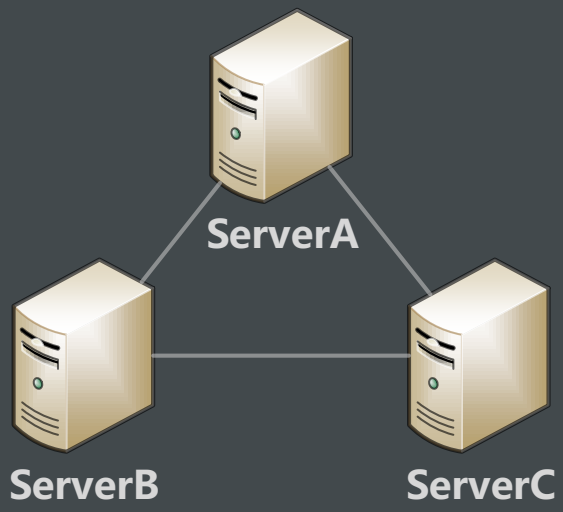
Consul+Docker服务注册与发现实践

硬件环境：

- 云主机x3：4核CPU，8GB内存

软件环境：

- CentOS 7 64位
- Docker 1.12.5
- Rancher V1.4.1



Consul+Docker服务注册与发现实践

- 安装Docker
- 先在一台主机启动 rancher-server 容器：

```
docker run -d --restart=unless-stopped  -p 8080:8080 rancher/server:stable
```

- 开启访问控制
- 在主机管理页面添加三台主机
- 等待Rancher自动初始化集群

Consul+Docker服务注册与发现实践

Consul集群的基本单位是agent，agent运行于主机集群中每个节点上，多个Consul agent组成集群。

Consul的两个运行模式：

Server：强一致性，保存服务状态，WAN通信

Client：无状态，转发HTTP请求，健康检查

每个数据中心建议由3-5个Server模式的agent组成高可用的集群。

大规模集群中Server模式的agent资源需求较高，可运行于专用的主机上，服务所在主机上则部署仅启用Client模式的agent。

Consul+Docker服务注册与发现实践

在Rancher中新建一个应用栈Consul，添加一个如下设定的服务：

数量：总是在每台主机上运行一个此容器的示例
名称：consul
选择镜像：consul
命令：agent -ui -data-dir /tmp/data -server -bootstrap-expect 3 -client 0.0.0.0 -retry-join [ServerA的地址]
网络：主机

-ui ：开启网页GUI。
-server：开启server模式。
-bootstrap-expect 3：有三个节点加入集群时开始leader选举。
-client 0.0.0.0：开启client模式，绑定到所有端口。
-retry-join [ServerA的地址]：通过连接ServerA加入集群

Consul+Docker服务注册与发现实践

当主机有多个IP地址时，Consul容器启动报错，提示有多个IP，请指定一个绑定IP。

对每个主机单独设置绑定IP虽然可行，但会导致每台主机的容器配置不同，不便于扩展。

Consul官方docker镜像的入口脚本中我们发现，该脚本支持使用环境变量来指定绑定IP。

设定容器的环境变量：CONSUL_BIND_INTERFACE=eth1

在容器启动时会自动绑定eth1端口的IP。

Consul+Docker服务注册与发现实践

Consul集群启动后，需要大约几十秒时间完成leader选举和状态同步。然后就可以通过8500端口打开Consul的页面查看Consul集群中节点和服务的状态。

Consul+Docker服务注册与发现实践

Registrator部署

Consul集群搭建完成后，即可提供服务注册和发现的服务，但为了实现基于容器的服务自动注册，我们还需要安装Registrator。

和Consul agent一样，Registrator也需要在集群中每个节点部署一个容器。通过挂载docker.sock，容器中运行的Registrator可以控制主机的Docker守护进程，通过Docker events达到监控容器变化的目的，并向本节点的Consul agent发送注册消息。

Consul+Docker服务注册与发现实践

Registrator部署

Consul集群搭建完成后，即可提供服务注册和发现的服务，但为了实现基于容器的服务自动注册，我们还需要安装Registrator。

和Consul agent一样，Registrator也需要在集群中每个节点部署一个容器。通过挂载docker.sock，容器中运行的Registrator可以控制主机的Docker守护进程，通过Docker events达到监控容器变化的目的，并向本节点的Consul agent发送注册消息。

Consul+Docker服务注册与发现实践

在Consul应用栈中添加服务。服务的各项设置如下：

数量：总是在每台主机上运行一个此容器的示例
名称：registrator
选择镜像：gliderlabs/registrator
命令：consul://localhost:8500
网络：主机
卷：/var/run/docker.sock:/tmp/docker.sock

此处值得注意的是要把主机的/var/run/docker.sock挂载进容器中。
registrator的其他详细设定可以通过修改启动命令来修改。

Consul+Docker服务注册与发现实践

Registrator部署

完成registrator的部署后，框架的搭建就基本完成了，可以在该环境中新建容器，如果容器通过nat开放了端口，会自动被注册到consul中；在容器停止或被删除时，则会从consul中注销。

Consul+Docker服务注册与发现实践

基于容器的自动注册虽然方便，但有些时候，我们需要对注册的过程进行一些定制。为了在自动注册时加入我们自己设定的一些信息，我们需要对服务所在的容器做一些改动。

Consul+Docker服务注册与发现实践

定制服务名称：

服务名称是Consul服务目录划分同一种服务的依据，在运用zuul作为服务网关进行反向代理时，如果服务列表取自consul的服务发现，服务名也是访问服务的默认路径，因此我们很有可能需要对服务名称进行定制。

要设置自动注册时采用的服务名，只需要给服务容器添加标签SERVICE_NAME=[服务名] 即可，也可以通过设置环境变量SERVICE_NAME=[服务名]来达到同样的效果。

Consul+Docker服务注册与发现实践

配置健康检查：

我们可以通过如下方式让Registrator在注册服务的同时，注册该服务的健康检查。以Http检查为例，设置标签或环境变量SERVICE_CHECK_HTTP=/health设置健康检查路径为/health；设置标签或环境变量SERVICE_CHECK_INTERVAL=15s设置健康检查间隔为15秒；设置标签或环境变量SERVICE_CHECK_TIMEOUT=5s设置健康检查请求超时时间为5秒。

通过上述设定，在服务被Registrator注册到Consul时，会附带注册一个间隔为15秒的http方式的健康检查，如果请求超时或失败，Consul会将服务置为故障状态，在取可用服务时不会提供该实例。

Consul+Docker服务注册与发现实践

修改服务程序：

将服务运行的容器进行上述设定后（建议通过Dockerfile进行构建，直接在镜像中包含这些设置），通过Rancher部署到之前建好的集群中运行，网络模式选择桥接，设定好开放端口，待服务启动后即可在Consul中看到服务的信息，在服务开始启动到服务能够正常处理请求之间的这段时间，由于健康检查失败，在consul中服务将显示为故障状态，直到第一次成功执行健康检查。

后记

后续有待研究的问题有：

- 如何在服务发布时平滑切换，通过Consul的API设定服务为维护状态可以达到目的，但这个操作是针对agent的，有一定局限性。
- 混合云环境下的跨数据中心集群如何部署。
- 其他使用中遇到的问题。

本文完成时Docker已经迎来重大变革，来到了17.03版本；Rancher也发布了1.5版本。这次更新带来了很多新特性有待我们进一步去探索。

docker DaoCloud IT大咖说 不止于技术

谢谢

THANKS

冰鉴 技术说

微信搜索公众号： 冰鉴技术说