OpenFaaS 调研分析

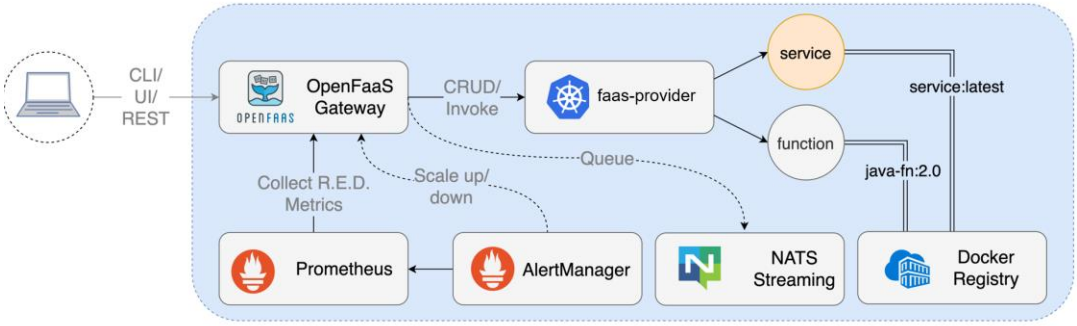
一、 OpenFaaS 简介

Serverless 的全称是 Serverless computing 无服务器运算，又被称为函数即服务（Function-as-a-Service，缩写为 FaaS），是云计算的一种计算范式。以平台即服务（PaaS）为基础，无服务器运算提供一个微型的架构，终端客户不需要部署、配置或管理服务器服务，代码运行所需要的服务器服务皆由云端平台来提供。

OpenFaaS 是一个开源的事件驱动的 serverless 框架， 为用户提供 FaaS 计算平台. OpenFaaS 当前官方的支持都是基于 docker 容器的运行， 可以直接利用 Kubernetes, Swarm 等编排系统进行管理， 也可以利用 faasd 在资源有限的单机环境下执行.

OpenFaaS 框架的整体设计与 micropos 非常类似， 每个功能都单独作为一个微服务， 对外提供 REST API， 并且使用 gateway 进行转发. Openfaas 是一个开放的框架， 对于后端如何运行云函数可以有多个不同的实现. 当前为了强隔离与缩放的因素， 最主流的都是利用容器， 每个函数都是在容器里面运行， 并且也是作为微服务的组件. 这些容器的管理编排工作主流的是使用 Kubernetes 来进行管理， 相应的管理部分 faas-netes 也构成了 OpenFaaS 整个框架的核心. Faas-provider 只规范了函数管理的接口， 这就极大地增加了整个框架的灵活性，不仅能利用容器，还可以使用其他隔离方案来部署运行云函数.

二、 OpenFaaS 架构



图一: OpenFaaS workflow

OpenFaaS 架构如图一所示，主要分为三个大部分：

Gateway：提供一系列用于函数管理，运行指标记录，容器缩放的 REST API，并且将其转发给相应的微服务组件.与 spring cloud gateway 功能差不多，能够根据配置提供网关服务.

CLI/UI：内置的 ui 和 faas-cli 负责为用户提供管理接口,并向 gateway 发送对应的请求

Faas-provider：提供一系列管理部署调用函数的 REST API，faas-provider 准确说是一个接口规范，具体使用哪些编排工具是需要具体的实现，如最流行的官方实现是 faas-netes，是利用 kubernetes 来编排容器进行管理函数.

另外 openfaas 还集成了其他工具，NATS 负责异步函数的执行与消息通信,Prometheus 负责运行指标记录，性能监控以及自动缩放等

OpenFaaS 的工作流：用户通过 REST API，如利用 faas-cli 或者内置的 UI 来访问 gateway，然后 gateway 转发到相应的微服务组件中. 每个微服务组件有默认的端口路由，也可以通过自定义来设置. Prometheus 通过 gateway 收集运行的指标，监控性能，并且根据相关条件发出自动缩放的请求.利用 gateway 转发相应的函数部署，调用请求. 函数调用可以使用 NATS 流的队列进行异步返回. Faas-provider 当前最主要的是使用 faas-netes，利用 kubernetes 进行容器编排，控制函数的部署以及调用.

三、 FaaS-provider

FaaS-provider 是用 go 编写的 SDK，利用 HTTP REST API 与 Gateway 进行交互通信，并与相关的接口兼容. 每个 faas-provider 接口都需要有以下的行为规范：

1. 函数的 CRUD
2. 调用运行函数
3. 接收缩放指令，通过缩放容器等行为缩放函数实例.
4. 密钥的 CRUD 以及日志流管理

当前 Faas-provider 的主要官方实现：

1. Faas-netes：与 Kubernetes 容器编排服务集成，使 OpenFaaS 能够通过

docker, Kubernetes 提供完整的无服务器功能支持. 与 Kubernetes 生态系统进行集成, 也能通过 kubectl 进行管理, 并且提供内置的 UI, 可用的商业支持, 异步调用等功能.

2. Faasd: 相比于 faas-netes 对于集群的支持, faasd 面向的是单个节点资源有限的机器. 能够在单个主机(如树莓派)下提供相同的功能(容器支持). Faasd 通过修改 contained 和利用 CNI, 使容器的部署执行比 docker 更节省资源, 更能适合 serverless 场景.
3. Faas-swarm: 利用 docker swarm 编排工具实现的容器管理, 函数管理.

四、 Watchdog

Openfaas 中最重要的是 watchdog. 每个云函数都作为一个 HTTP 微服务运行, watchdog 就是作为云函数微服务的反向代理, 将 http 的 request 内容作为函数的 stdin, 将函数输出的 stdout 作为 http 的 response 返回. 或者将 http 请求转发到函数监听的 http 端口. 同时 watchdog 还利用 Prometheus 来监控当前性能, 并且返回相应的 metrics.

实际的部署过程中, watchdog 与相应的云函数在一起作为一个容器镜像, 运行的时候 watchdog 监听 http 请求, 并且启动相应的云函数进程. 云函数可以使用任意语言, 只要是可以运行的二进制文件即可.

Watchdog 在执行函数的时候, 可以用多种模式来运行. 主要模式是 http 模式和 Serializing 模式. 对于 Http 模式, 将用 fork 启动云函数进程, watchdog 将 http 请求转发给云函数相应的端口. Serializing 模式也是利用 fork 启动云函数进程, 利用管道与云函数的进程通信, 重定向云函数的 stdio. 模式的选用根据函数的功能来选择.

五、 Autoscaling

自动缩放可以自动根据负载调整函数的实例副本的数量. 自动缩放也是实现了多种模式:

RPS: 根据每秒完成请求的个数, 适用于非常短时间的函数.

Capacity: 容量, 根据当前正在执行的个数, 适用于长时间的函数.

CPU: 根据当前 cpu 的负载来决定是否缩放.