

OpenFunction

“云无关的”函数计算框架的应用探索

Press Space for next page →



张海立

KubeSphere Ambassador, CNCF OpenFunction TOC Member

驭势科技 UISEE© 云平台研发总监，开源爱好者

云原生专注领域: Kubernetes, DevOps, FaaS, Observability, Service Mesh

👤 zhanghaili

🐱 webup

🐦 zhanghaili0610

✉️ haili.zhang@uisee.com

🏢 ServiceUP · 语雀



1 CNCF OpenFunction 函数计算框架简介

2 OpenFunction 的一些应用场景探讨

3 OpenFunction 函数网关的演进及展望

Open? Function! ☁

一个 CNCF Sandbox 函数计算平台项目

OpenFunction 的云原生特性：

- “云无关”：与云商的 BaaS 松耦合
 - 通过 Dapr，简化了对于云商 BaaS 的集成
- 同时支持同步和异步函数
 - 同步函数 Kubernetes Gateway API 支持的高级函数入口和流量管理
 - 异步函数可以直接从事件源消耗事件
- 支持直接从函数源码生成符合 OCI 镜像
 - 基于 Cloud Native Buildpacks 生态实现
- 在 0 和 N 之间灵活的自动缩放
 - 充分利用 KEDA、Knative



[Blog](#) [Community](#)



[Docs ➔](#)

[GitHub 🐳](#)



KubeCon



CloudNativeCon

Europe 2022

分享

Keynote: Landscape Sustainability: The Pillars of Cloud Native Growth

Dave Zolotusky, Software Engineer, Spotify & Katie Gamanji, Senior Kubernetes Field Engineer, Apple



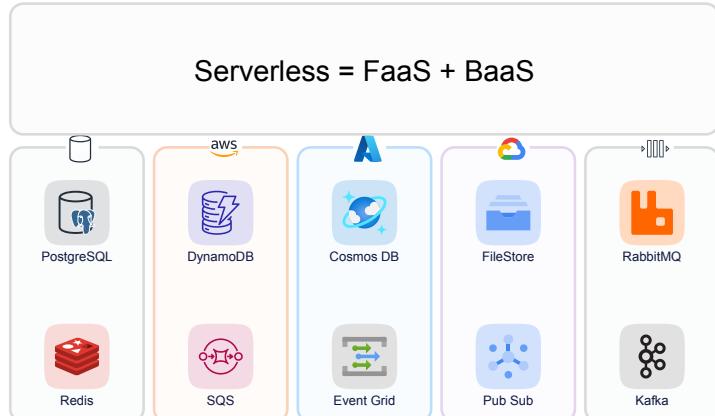
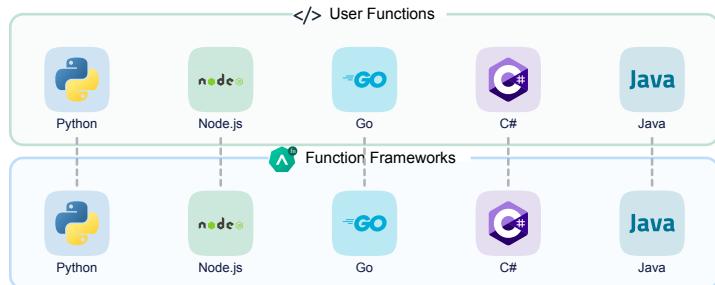
“云无关”的价值

Cloud Agnostic — 与云商的 BaaS 解耦

"Serverless computing = FaaS + BaaS." [1]

- 云函数是函数计算的主体
 - 各类后端服务是函数实现业务的重要依托
- 🔒 云商通过提供“托管的”(Hosted) 函数计算框架(FaaS)和各类中间件服务(BaaS)，把开发者锁定在托管的云平台上。
- 💡 即使函数计算框架是跨语言跨平台的，谁又来填补各云商平台 BaaS 服务的接口差异？

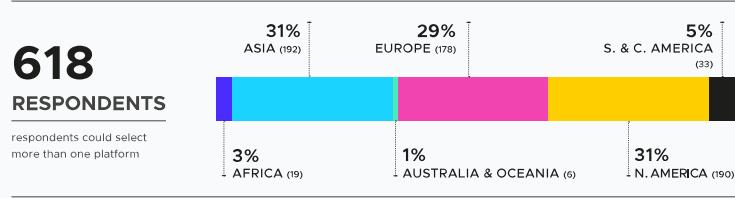
1. Cloud Programming Simplified: A Berkeley View on Serverless Computing. Feb 10, 2019.



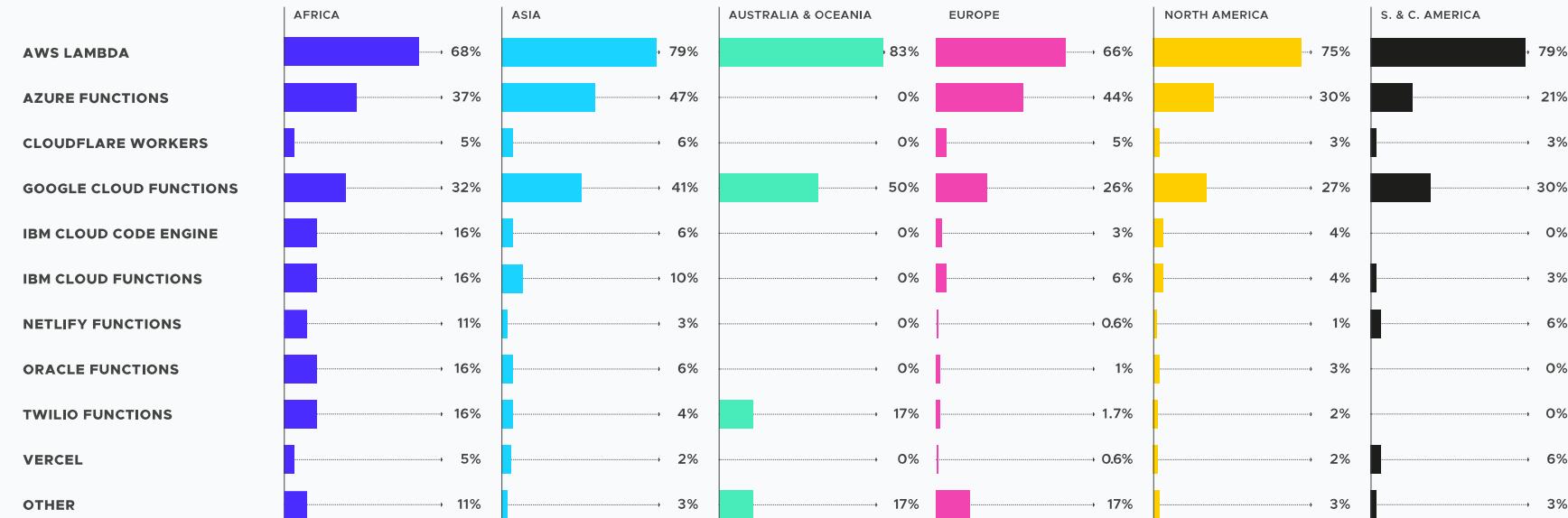
Serverless 平台的战争？

CNCF Annual Survey 2021

"使用 Serverless 技术的受访者，75% 选择了托管平台"



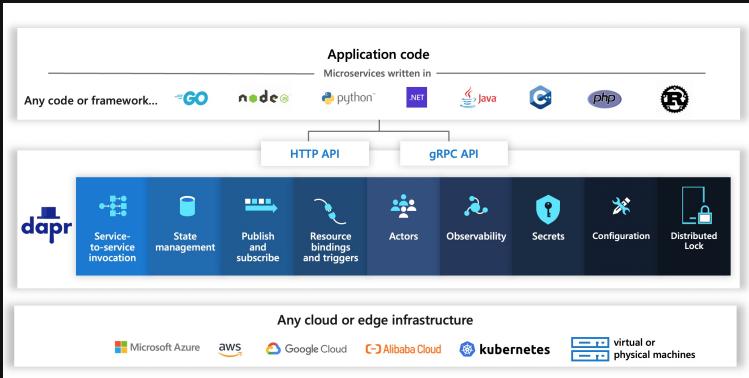
IF YOUR ORGANIZATION IS USING SERVERLESS VIA A 'HOSTED PLATFORM', WHICH HOSTED SERVERLESS PLATFORMS DO YOU USE?



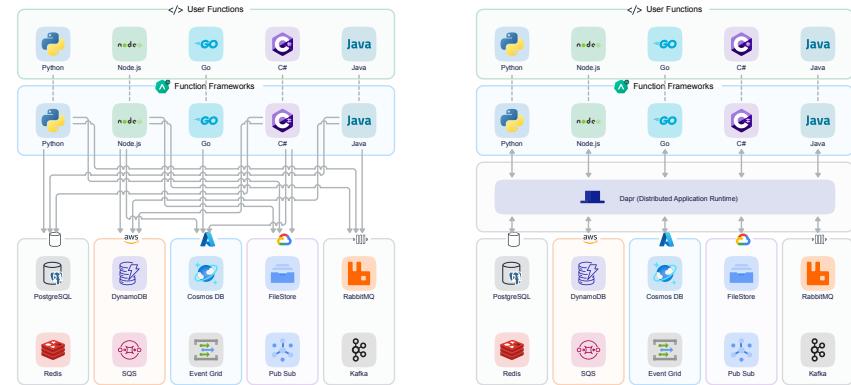
破局之道



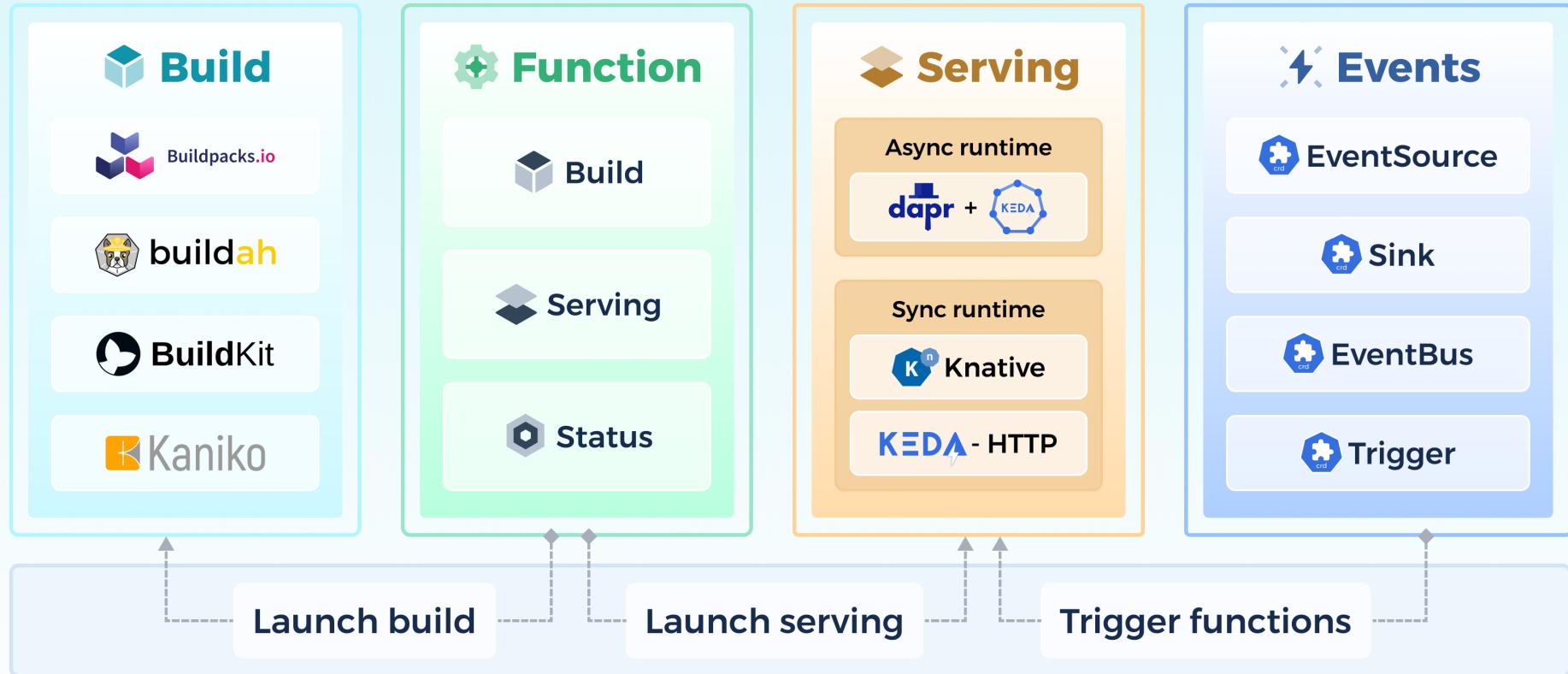
借助 Dapr 来集成各云商平台的 BaaS 服务

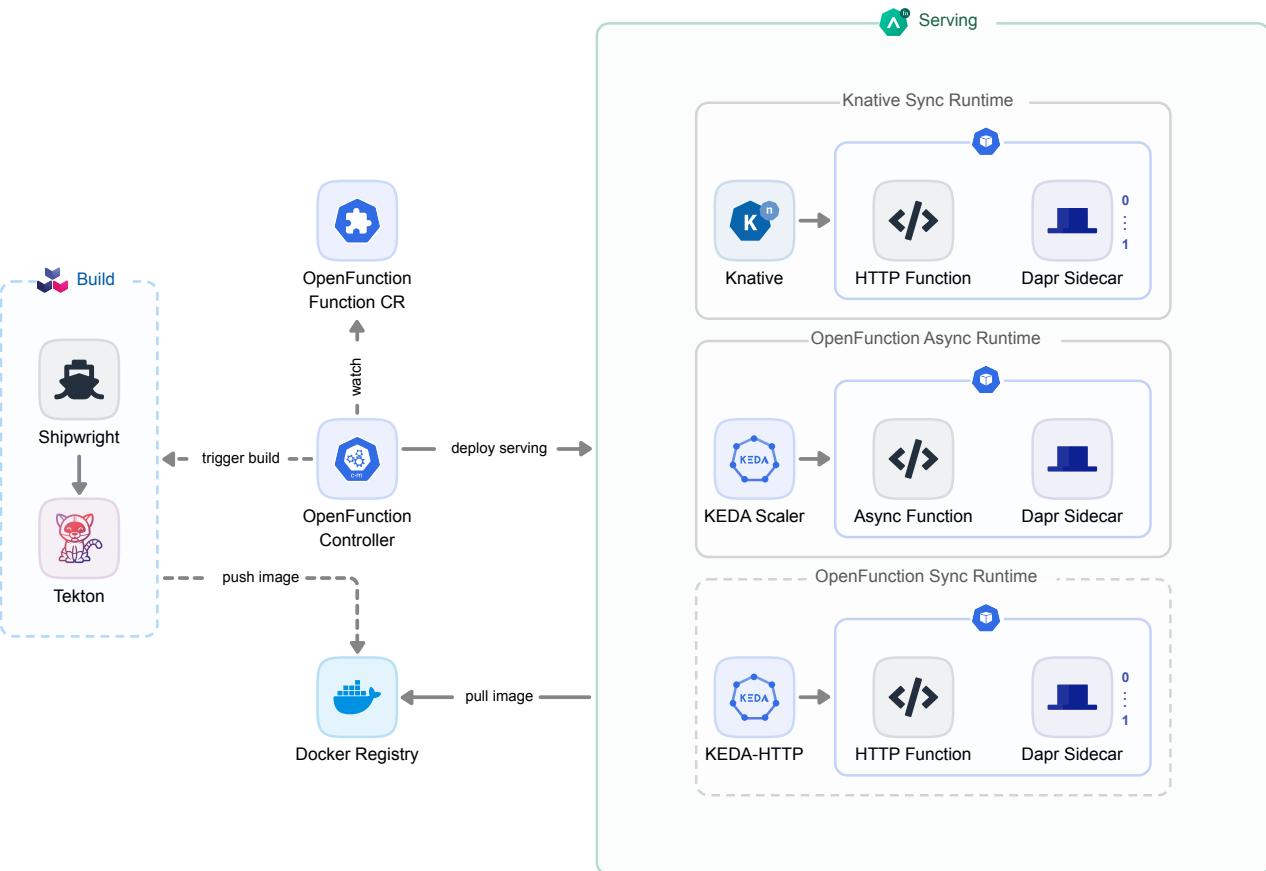


Dapr provides you with APIs that abstract away the complexity of common challenges developers encounter regularly when building distributed applications. These API building blocks can be leveraged as the need arises - use one, several or all to develop your application faster and deliver your solution on time.



Cloud-Native FaaS Platform





举几个函数的例子



OpenFunction 现已支持 Go, Node.js, Python, Java, .Net 等多种语言

一个 Node.js 同步函数

INDEX.MJS

```
// Standard Express style HTTP sync function
export const tryKnative = (req, res) => {
  res.send(`Hello, ${req.query.u || 'World'}!`);
};
```

🔍 参见 Express 的 `request` 和 `response` 对象的使用

PACKAGE.JSON

```
{
  "main": "index.mjs",
  "scripts": {
    "start": "functions-framework --target=tryKnative"
  },
  "dependencies": {
    "@openfunction/functions-framework": "^0.6.0"
  }
}
```

FUNCTION CR (RAW MANIFEST)

```
apiVersion: core.openfunction.io/v1beta1
kind: Function
metadata:
  name: sample-node-knative
spec:
  version: v2.0.0
  image: '<image-repo>/<image-name>:<image-tag>'
  build:
    builder: openfunction/builder-node:v2-16.15
  env:
    FUNC_NAME: tryKnative
  srcRepo:
    url: https://github.com/OpenFunction/samples.git
    sourceSubPath: functions/async/mqtt-io-node
    revision: main
  # app port default to "8080"
  port: 8080
  serving:
    runtime: knative
    template:
      containers:
        - name: function
          imagePullPolicy: IfNotPresent
```

一个 Node.js 异步函数

INDEX.MJS

```
// Standard Express style HTTP sync function
export const tryKnative = (req, res) => {
  res.send(`Hello, ${req.query.u || 'World'}!`);
};

// Async function
export const tryAsync = (ctx, data) => {
  console.log('Data received: %o', data);
  ctx.send(data);
};
```

💡 多个函数可以被放在同一个 JS 文件中，并在 CR 中动态指定

PACKAGE.JSON

```
{
  "main": "index.mjs",
  "dependencies": {
    "@openfunction/functions-framework": "^0.6.0"
  }
}
```

FUNCTION CR (RAW MANIFEST)

```
apiVersion: core.openfunction.io/v1beta1
kind: Function
metadata:
  name: sample-node-async
spec:
  version: v2.0.0
  image: '<image-repo>/<image-name>:<tag>'
  serving:
    # default to knative
    runtime: async
  annotations:
    # default to "grpc"
    dapr.io/app-protocol: http
  template:
    containers:
      - name: function
  params:
    # default to FUNC_NAME value
    FUNCTION_TARGET: tryAsync
  inputs:
    - name: mqtt-input
      # default to "topic" value
      # type: mqtt-topic
```

一个可以触发异步函数的 Node.js 同步函数

INDEX.MJS

```
// Standard Express style HTTP sync function
export const tryKnative = (req, res) => {
  res.send(`Hello, ${req.query.u || 'World'}!`);
};

// OpenFunction standard function which could
// set bridge between sync and async function
export const tryKnativeAsync = async (ctx, data) => {
  console.log('Data received: %o', data);

  // Forward HTTP body to Dapr outputs
  await ctx.send(data);

  // Send something back as HTTP response
  // when necessary
  // ctx.res.send(data);
};


```



`openfunction` 函数形态 可以同时支持同步和异步函数

FUNCTION CR (RAW MANIFEST)

```
apiVersion: core.openfunction.io/v1beta1
kind: Function
metadata:
  name: sample-node-knative-async
spec:
  version: v2.0.0
  image: '<image-repo>/<image-name>:<image-tag>'
  serving:
    runtime: knative
  params:
    FUNCTION_TARGET: tryKnativeAsync
    FUNCTION_SIGNATURE_TYPE: openfunction
  outputs:
    - name: mqtt-output
      component: mqtt-out
      operation: create
  bindings:
    mqtt-out:
      type: bindings.mqtt
```

案例分享：数据桥接

OpenFunction 简化数据预处理

同步: GITLAB WEBHOOK ➡ CHATOPS

```
import (
    // ...
    "github.com/OpenFunction/functions-framework-go"
    "github.com/xanzy/go-gitlab"
)

var GitlabClient *Client

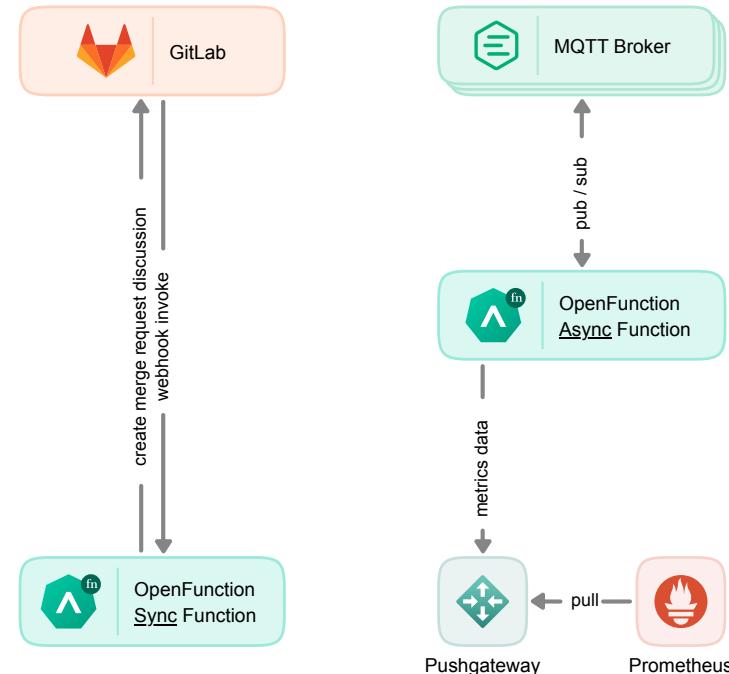
func init() {
    functions.HTTP("HelloWorld", SendBeginUnittest,
        OpenFunctionSyncFunction)
}
```

异步: 消息队列实时数据 ➡ PROMETHEUS 指标

```
import (
    "github.com/OpenFunction/functions-framework-go"
    "github.com/prometheus/client_golang/prometheus"
)

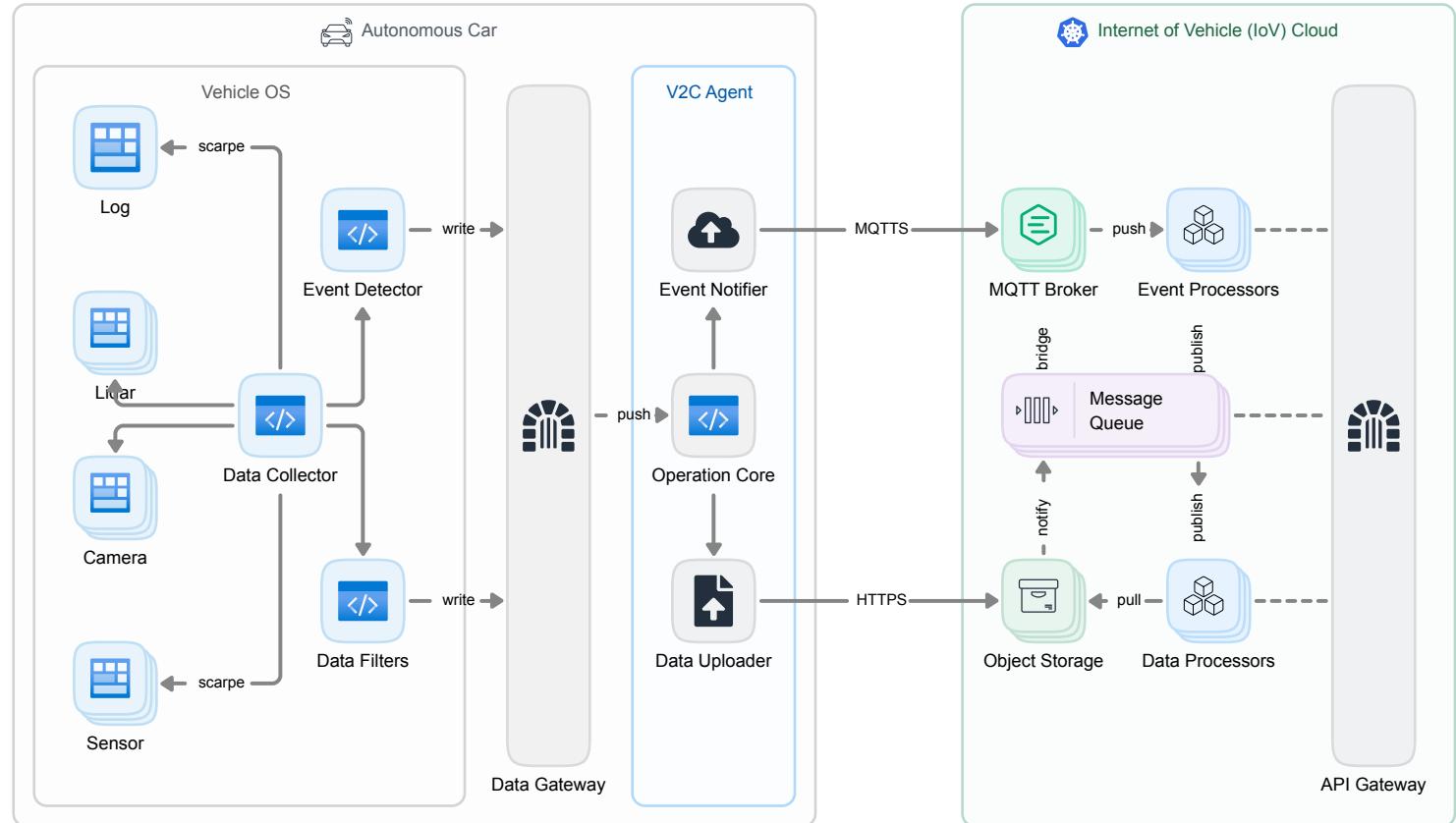
var speedGauge prometheus.Gauge

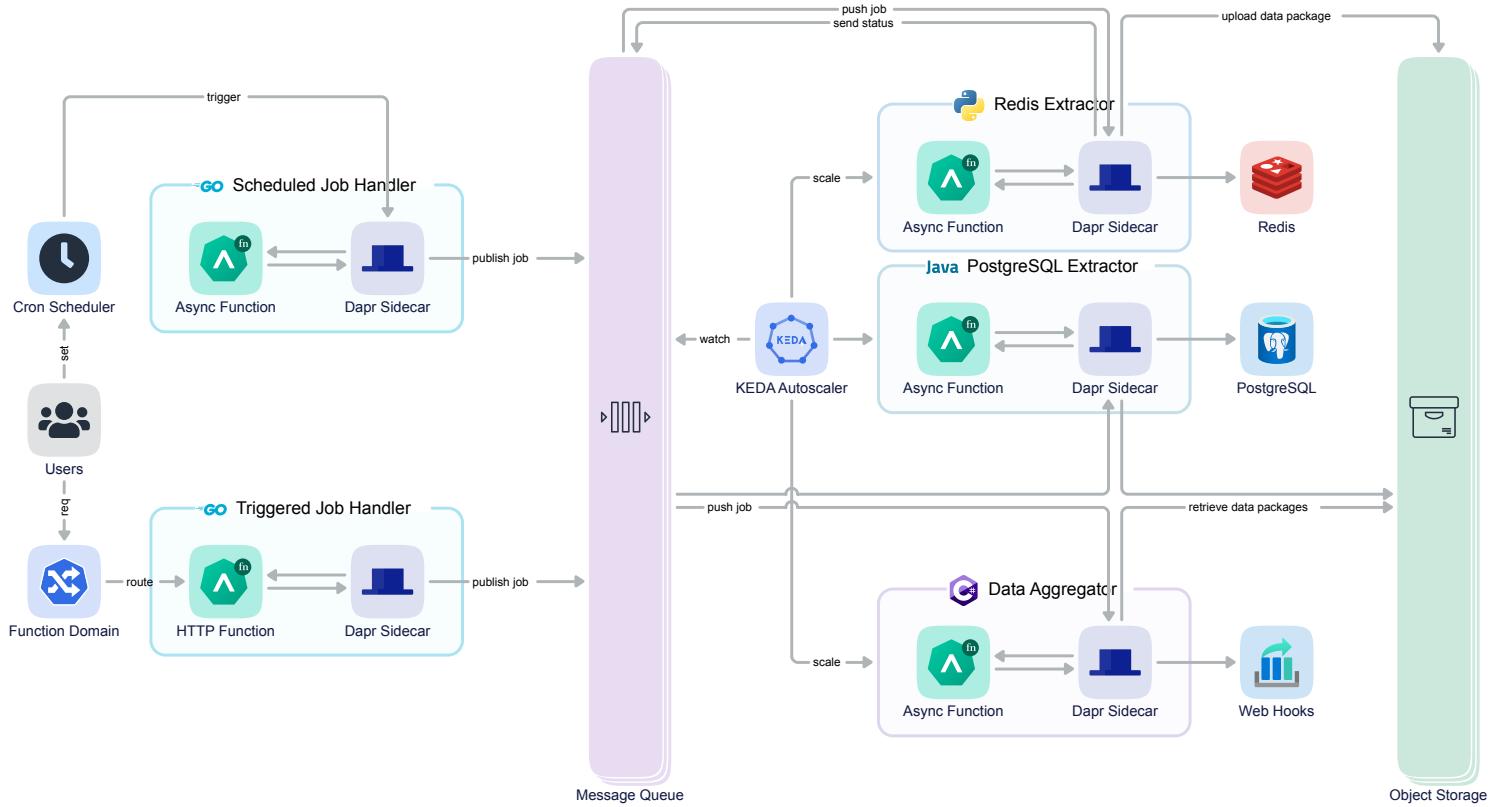
func init() {
    speedGauge = prometheus.NewGauge(prometheus.GaugeOpts{
        Namespace: namespace,
        Subsystem: subsystem,
        Name:      "speed",
    })
}
```



案例分享：实时数据处理

OpenFunction 在无人驾驶业务中的一种应用





聊一聊函数网关



如何让同步函数可以更方便和更灵活的被访问？

基于 Ingress 的网关

OpenFunction 0.5.0 - 0.6.0 中可使用

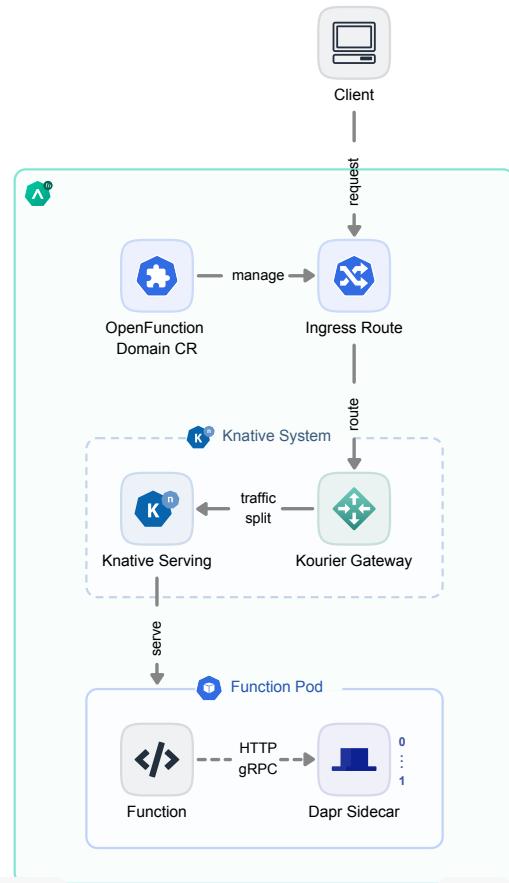
🚧 每个函数对应的 Service 不便于外部直接访问，如 `sample-serving-d55zz-ksvc-gd64w`

💡 直接的想法：由 Ingress 统一函数访问入口

💡 通过 Domain CR 为 Knative 同步函数提供 Ingress 访问入口：`http://<domain_name>. <domain_ns>/<function_ns>/<function_name>`

❗ 这个方案可用，但也存在一些问题：

- 强依赖于 Knative 的网关（OpenFunction 默认随 Knative 安装使用 Courier）
 - Courier Domain 配置比较麻烦
- 不易适配未来要引入的其它同步函数运行时



基于 Gateway 的网关

OpenFunction 0.7.0 开始可使用

💡 进一步的想法：直接使用 K8s Gateway API

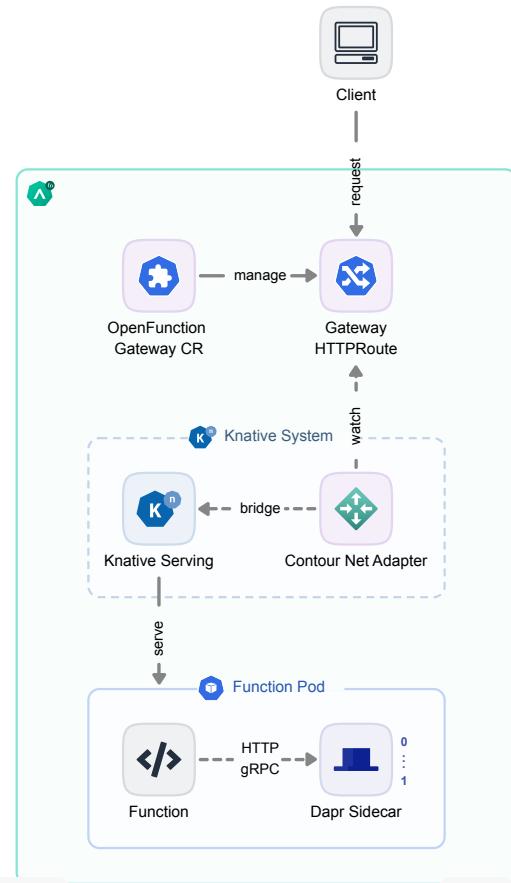
更强大更灵活的 Kubernetes 原生网关，越来越多的
社区支持：Contour ，Istio，Apache APISIX 等等

👤 构建新的 OpenFunction Gateway 体系

Domain 体系从 OpenFunction 0.7.0 开始被移除

🚀 OpenFunction Gateway 的一些特性：

- 开发者可以自定义函数入口，如 `{{.Name}}`。
`{{.NS}}.{{.Domain}}`
- 开发者可以自定义每个函数的路由规则
- 未来可以支持多个函数版本的流量切分



Apache APISIX®

使用 APISIX Ingress Controller 作为函数网关

- APISIX Ingress Controller 已经从 1.5.0 版本开始 初步支持 Kubernetes Gateway API:
- 需要通过 `enable_gateway_api=true` 开启 OpenFunction Gateway 所需要的 `HTTPRouteFilter` 计划在 1.6.0 中支持
- APISIX 强大而丰富的插件生态也正在助力 OpenFunction 的应用和发展:
- “最近，APISIX 又新增了不少生态插件，其中就包括与 OpenFunction 集成的无服务插件 `openfunction`” ↗ 详见 集成细节



Apache APISIX®



Version: 1.5.0

On this page

Getting Started

What is apisix-ingress-controller

apisix-ingress-controller is yet another Ingress controller for Kubernetes using **Apache APISIX** as the high performance reverse proxy.

It's configured by using the declarative configurations like **AbisixRoute**, **AbisixUpstream**, **Ingress**. All these resources are



Learn More



Discord



Slack

OpenFunction · Node.js Functions Framework
