

# IBM开源技术微讲堂

Istio系列

第八讲

Istio使用案例：Serverless平台Knative

<http://ibm.biz/opentech-ma>

# “Istio”系列公开课

- 每周四晚8点档
  - Istio 101
  - 上手istio: 基本概念，安装并利用istio进行微服务流量管控
  - Envoy
  - Istio Mixer
  - 使用istio监控你的微服务
  - Istio Security
  - Istio跨云方案解析
  - Istio使用案例：Serverless平台Knative

# 讲师介绍



- 郭迎春
  - IBM工程师，开源贡献者
  - 多个开源社区经历：OpenOffice，OpenStack，Apache OpenWhisk
  - 首任OpenStack I18n PTL
  - Apache OpenWhisk Committer

# 议程

- Serverless概念介绍
- Knative简介
- Knative中的日志、监控和追踪
- Knative中的网络管控

# 议程

- Serverless概念介绍
- Knative简介
- Knative中的日志、监控和追踪
- Knative中的网络管控

# 什么是Serverless?

## What is Serverless?

Like many trends in software, there's no one clear starters, it encompasses two different but overlapping

1. Serverless was first used to describe application third-party, cloud-hosted applications and services in state. These are typically "rich client" applications—think single-page web apps, or mobile apps—that use the vast ecosystem of cloud-accessible databases (e.g., Parse, Firebase), authentication services (e.g., Auth0, AWS Cognito), and so on. These types of services have been previously described as "(Mobile) Backend as a Service", and I use "BaaS" as shorthand in the rest of this article.
2. Serverless can also mean applications where server-side logic is still written by the application developer, but, unlike traditional architectures, it's run in stateless compute containers that are event-triggered, ephemeral (may only last for one invocation), and fully managed by a third party. One way to think of this is "Functions as a Service" or "FaaS". (Note: The [original source](#) for this name—a tweet by @marak—is no longer publicly available.) AWS Lambda is one of the most popular implementations of a Functions-as-a-Service platform at present, but there are many others, too.



Mike Roberts

Mike Roberts is a partner, and co-founder, of **Symphonia** - a consultancy specializing in Cloud Architecture and the impact it has on companies and teams.

<https://www.martinfowler.com/articles/serverless.html>

[cncf / wg-serverless](#)

<> Code

🔔 Issues 0

🔗 Pull requests 1

📊 Insights

Branch: master ▾

[wg-serverless](#) / [whitepapers](#) / [serverless-overview](#) /

## Serverless Computing

### What is serverless computing?

Serverless computing refers to the concept of building and running applications that do not require server management. It describes a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment.

the developer to be serverless.  
 abstraction because those APIs are provided as a service that auto-scales and operates transparently. This appears  
 3. Backend-as-a-Service (BaaS) which are third-party API-based services that replace core subsets of functionality in  
 underlying infrastructure.  
 the BaaS which are executed as needed as discrete actions, scaling without the need to manage servers or any other  
 application code with functions that are triggered by events or HTTP requests. Developers deploy small units of code  
 4. Functions-as-a-Service (FaaS), which typically provides event-driven computing. Developers run and manage  
 A serverless computing platform may provide one or both of the following:

<https://github.com/cncf/wg-serverless/tree/master/whitepapers/serverless-overview>

# 什么是无服务器 Serverless ?

Serverless =

Functions as a Service

Backend as a Service

- Function-as-a-Service (FaaS)
  - 小段代码，按需执行，按需扩展，无需管理任何基础实施相关的部分。
  - 事件驱动型计算。函数被事件触发或者被HTTP请求调用。
- Backend-as-a-Service (BaaS)
  - 第三方基于API的服务，实现应用开发中的基础功能模块。
  - 这些API像服务一样，自动扩展，无需管理。

The goal of serverless is to abstract the complexity out of software development until it's accessible to all.

By Austen Collins, Founder & CEO, Serverless Inc.

# 无服务器计算的要点

## Function-as-a-Service

无需管理  
按需扩展  
按需执行  
按使用计费

函数无状态的、短暂的、有限制的  
事件驱动  
API网关

## Backend-as-a-Service

无需管理  
按需扩展  
按需执行  
按使用计费










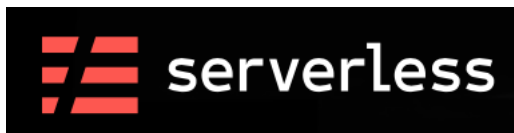
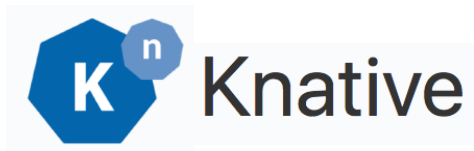
# 无服务器计算的优点

- 快速开发
- 无限扩展
- “绿色”计算
- 对开发者友好
- 费用低
- .....

# 无服务器应用的典型应用

	<b>micro-service</b>	Easily implement fine-grained, micro-service APIs.
	<b>IoT</b>	Power various mobile, web and IoT app use cases by scaling and simplifying the programming model of orchestrating various services.
	<b>Batch and Stream Processing</b>	Automate and control batch and stream processing
	<b>DevOps</b>	Automate DevOps pipeline based on events triggered from successful builds or completed staging or a go-live event.
	<b>IT/Ops</b>	Allow an easier deployment model for administrative functions (bots) to run for IT/Ops.

# Serverless开源项目



## Microcule

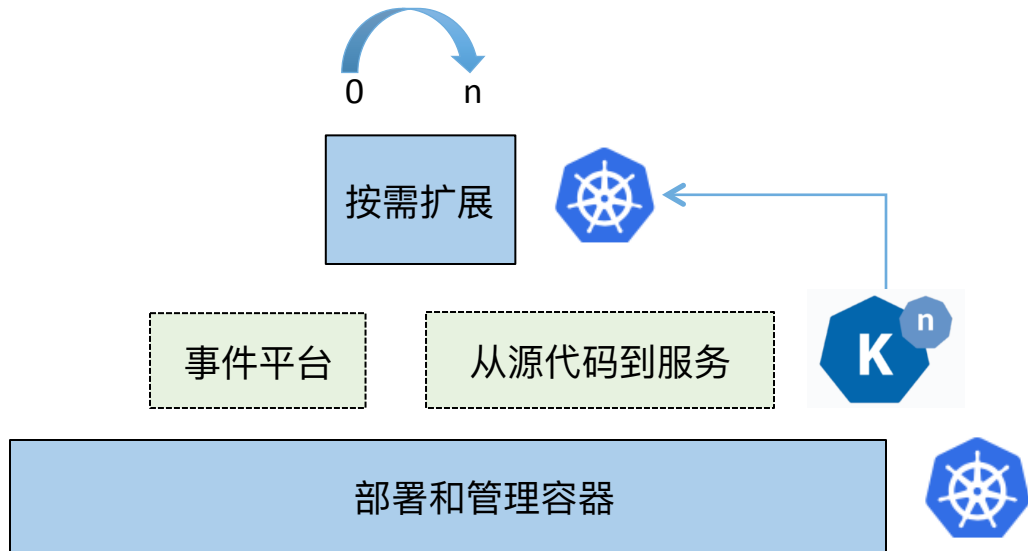


# 议程

- Serverless概念介绍
- Knative简介
- Knative中的日志、监控和追踪
- Knative中的网络管控
- Eventing

# Knative的由来

从Kubernetes到Serverless还缺什么？



# Knative - Kubernetes原生Serverless平台

Knative extends Kubernetes to provide a set of **middleware** components that are essential to build **modern, source-centric**, and **container-based applications** that can run anywhere: on premises, in the cloud, or even in a third-party data center.



# Knative设计理念

01

**Idiomatic**

- Feel native on Kubernetes
- Meet the developer

02

**Extensible**

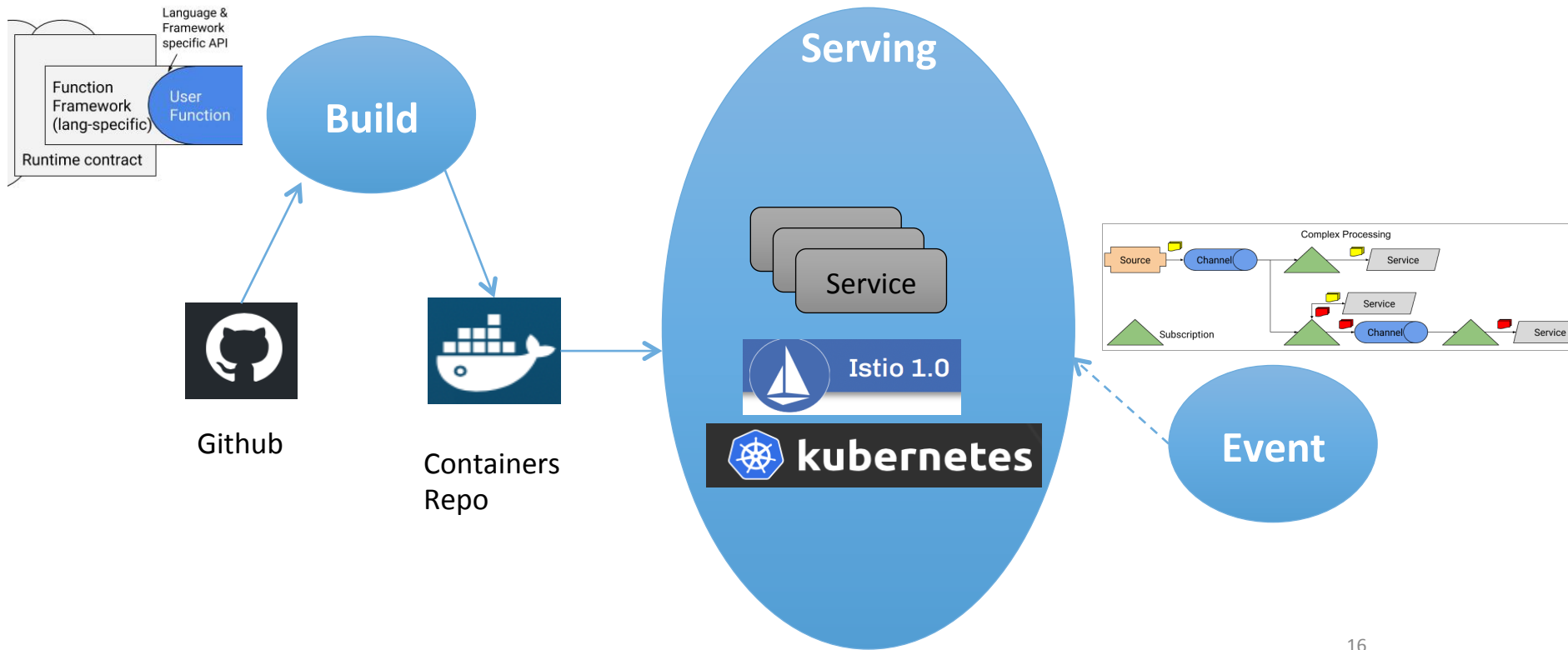
- Loose coupling at the top
- Pluggable at the bottom

03

**Organic**

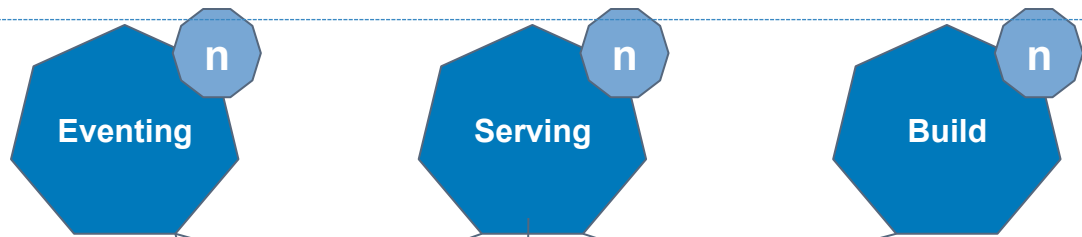
- Codify the commonalities
- Build a stable platform

# Knative三大模块

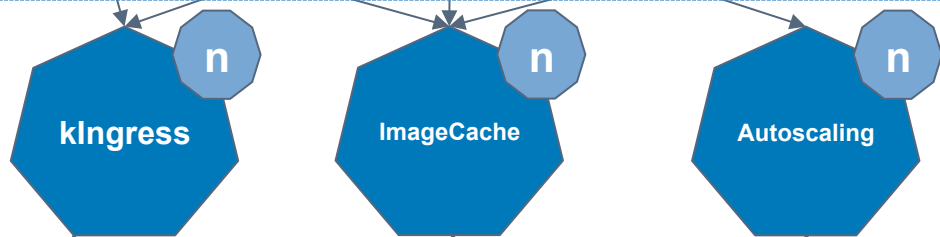




# Pluggability



Knative三大组件



基础功能模块



Istio

NGINX

可以自己选择或开发...



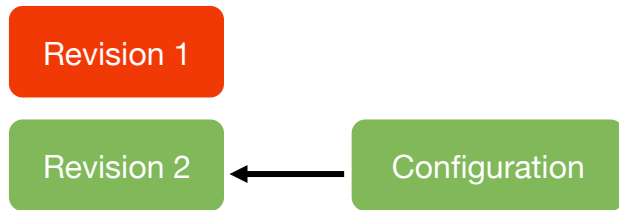
具体实现

## 演示: Knative安装之后

- 查看knative-serving名称空间
- 查看istio-system名称空间
- 查看knative-monitoring名称空间

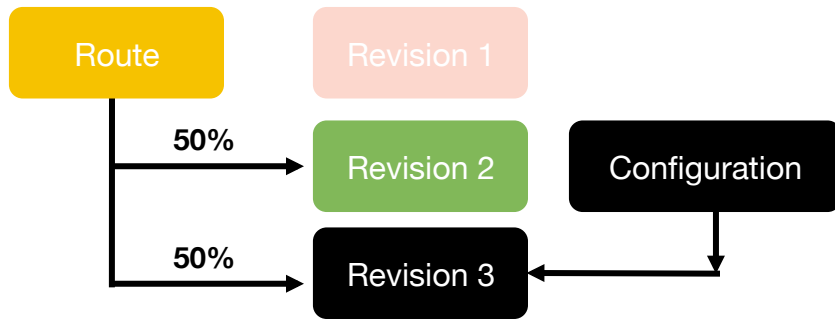
# Knative的概念： Configuration和Revision

- Configuration
  - 描述用户服务的最新状态
  - 用户直接创建
- Revision
  - 实现用户逻辑的具体程序
  - 在K8s中表现为deployment， 包含用户容器及若干个sidecar
  - 用户**不可以**直接创建
  - 当Configuration创建或者更新时， 自动创建



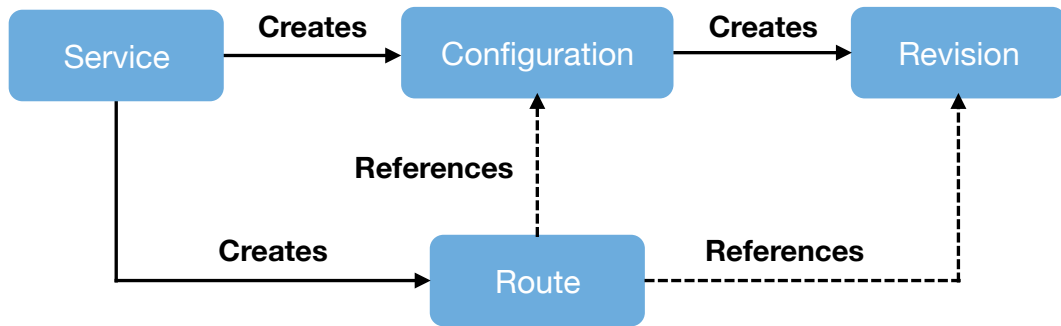
# Knative的概念： Route

- 可访问用户服务的HTTP地址
- 表现为 应用名+域名
- 后端映射到一个或多个Revision
- 由Istio实现



# Knative的概念：Service

- 高级管理模块
- 帮助创建configuration和route



# 演示：动态路由



Powered by Istio

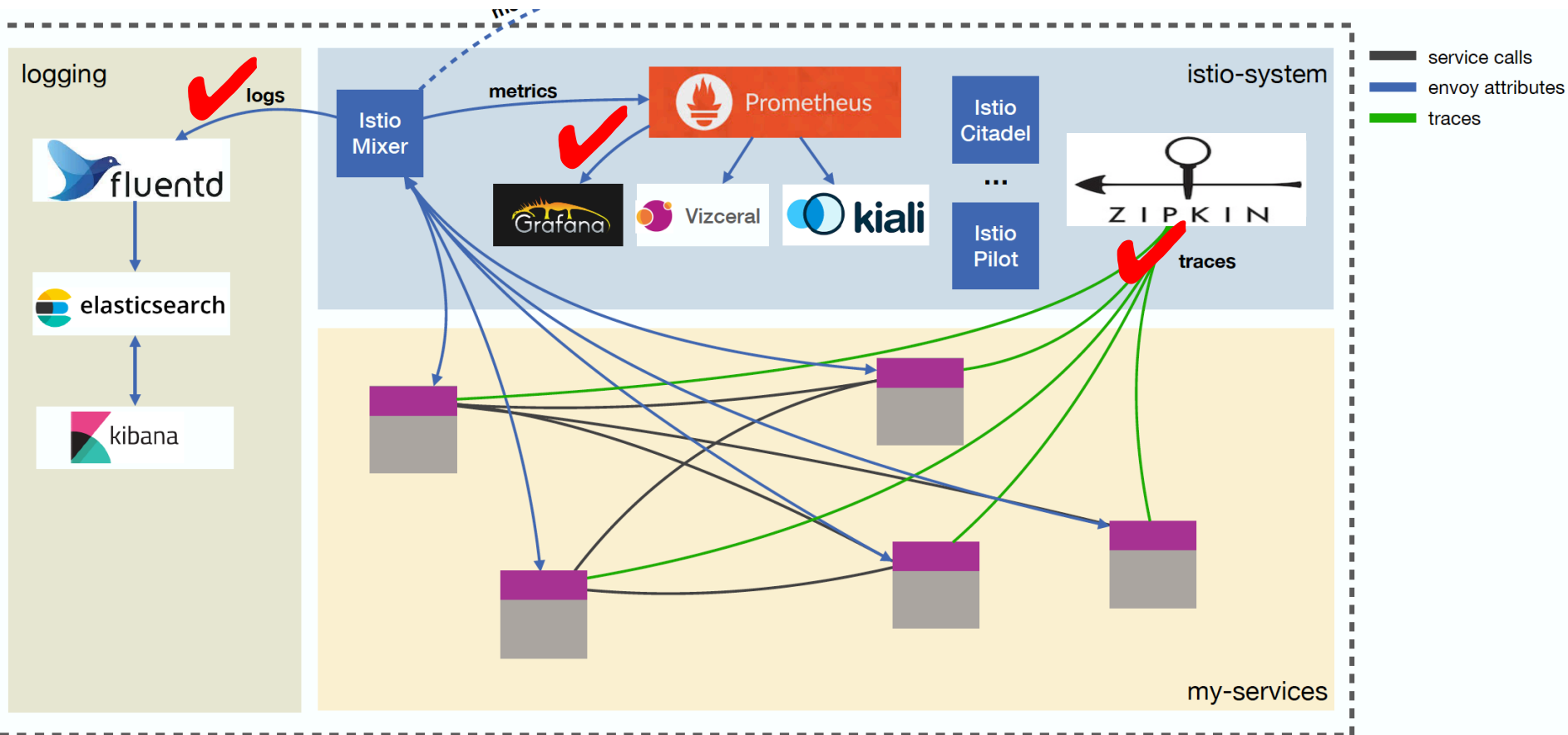
## Service

```
apiVersion: serving.knative.dev/v1alpha1 # Current version of Knative
kind: Service
metadata:
  name: helloworld-go # The name of the app
  namespace: default # The namespace the app will use
spec:
  runLatest:
    configuration:
      revisionTemplate:
        spec:
          container:
            image: gcr.io/knative-samples/helloworld-go # The URL to the image of the app
            env:
              - name: TARGET # The environment variable printed out by the sample app
                value: "Go Sample v1"
```

# 议程

- Serverless概念介绍
- Knative简介
- Knative中的日志、监控和追踪
- Knative中的网络管控

# 架构总览



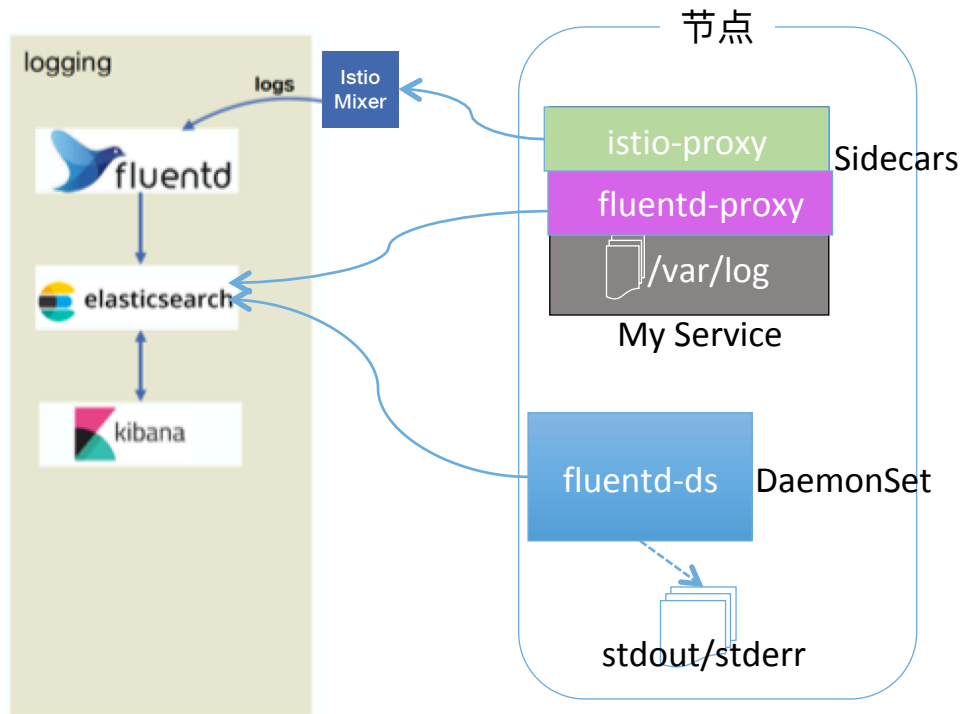


# 日志

Powered by fluentd



- Istio Mixer
- fluentd-ds (DaemonSet )
- fluentd-proxy (Sidecar)



# 监控



Powered by Istio

## Metrics:

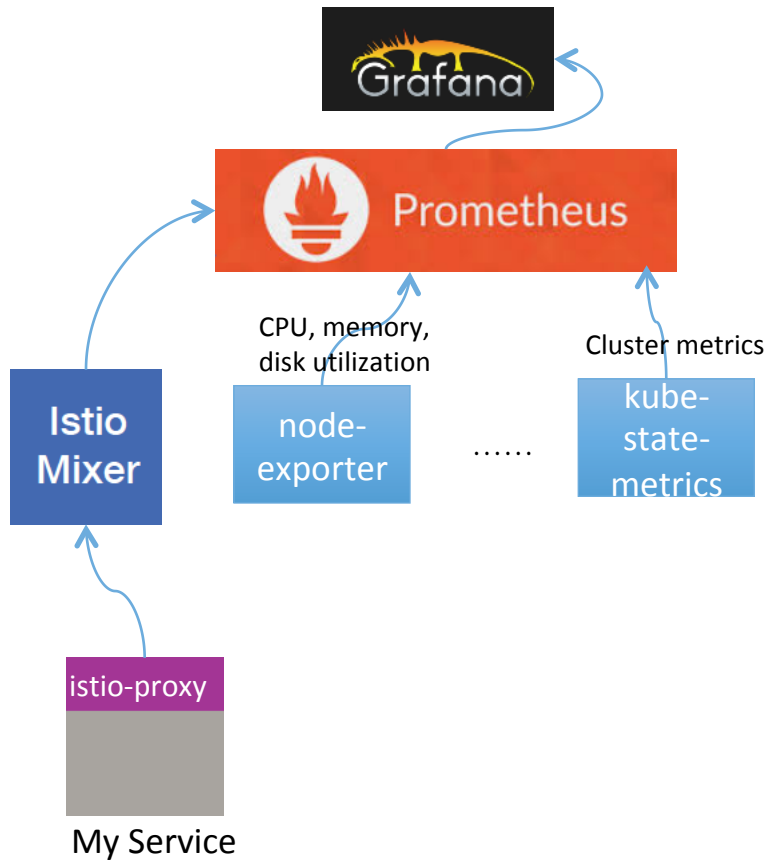
- requestcount
- requestduration
- requestsize
- responsesize
- tcpbytesent
- tcpbytereceived
- [revisionrequestcount](#)
- [revisionrequestduration](#)
- [revisionrequestsize](#)
- [revisionresponsesize](#)

## Handler:

- handler.prometheus

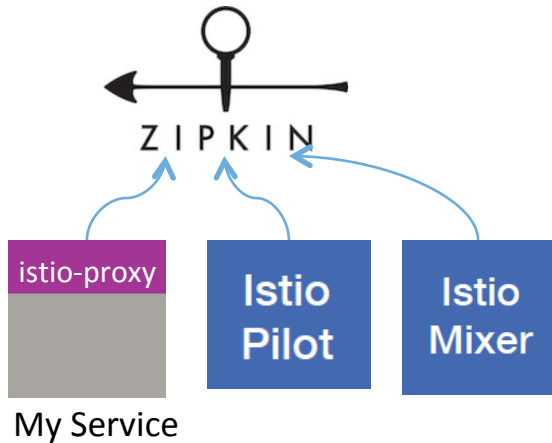
## Rule:

- promhttp
- promtcp
- [revisionpromhttp](#)



# 分布式追踪

- OpenZipkin管理trace



# 演示：telemetry-go

apiVersion: serving.knative.dev/v1alpha1

## Configuration

kind: Configuration

metadata:

name: [telemetrysample-configuration](#)

namespace: default

labels:

app: telemetrysample

spec:

revisionTemplate:

metadata:

labels:

knative.dev/type: [app](#)

spec:

container:

image: [docker.io/daisyycguo/knative-telemetry:latest](#)

apiVersion: serving.knative.dev/v1alpha1

## Route

kind: Route

metadata:

name: [telemetrysample-route](#)

namespace: default

spec:

traffic:

- configurationName: [telemetrysample-configuration](#)

percent: 100

# 议程

- Serverless概念介绍
- Knative简介
- Knative中的日志、监控和追踪
- Knative中的网络管控

# 演示：蓝绿发布



Powered by Istio

## Route

apiVersion: serving.knative.dev/v1alpha1

kind: Route

metadata:

name: blue-green-demo

namespace: default

spec:

traffic:

- revisionName: blue-green-demo-00001

percent: 50

- revisionName: blue-green-demo-00002

percent: 50

name: v2

# 回顾

- Serverless概念介绍
- Knative简介
- Knative中的日志、监控和追迹
- Knative中的网络管控

<https://github.com/daisy-ycguo/knative-learning>

## 参考资料

- 本讲中的演示: <https://github.com/daisy-ycguo/knative-learning>
- Knative的源码: <https://github.com/knative>



# IBM开源技术微讲堂

Istio系列

完



关注公众号，期待更多精彩