GOTC

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

OPEN SOURCE , OPEN WORLD

「开源云原生计算时代论坛」专场

本期议题：Knative Eventing系统应用实践

毕小红 2021年7月10日

自我介绍

GOTC

毕小红：

中移（苏州）软件技术有限公司

软件开发工程师

目前主要负责移动云函数计算的研发，

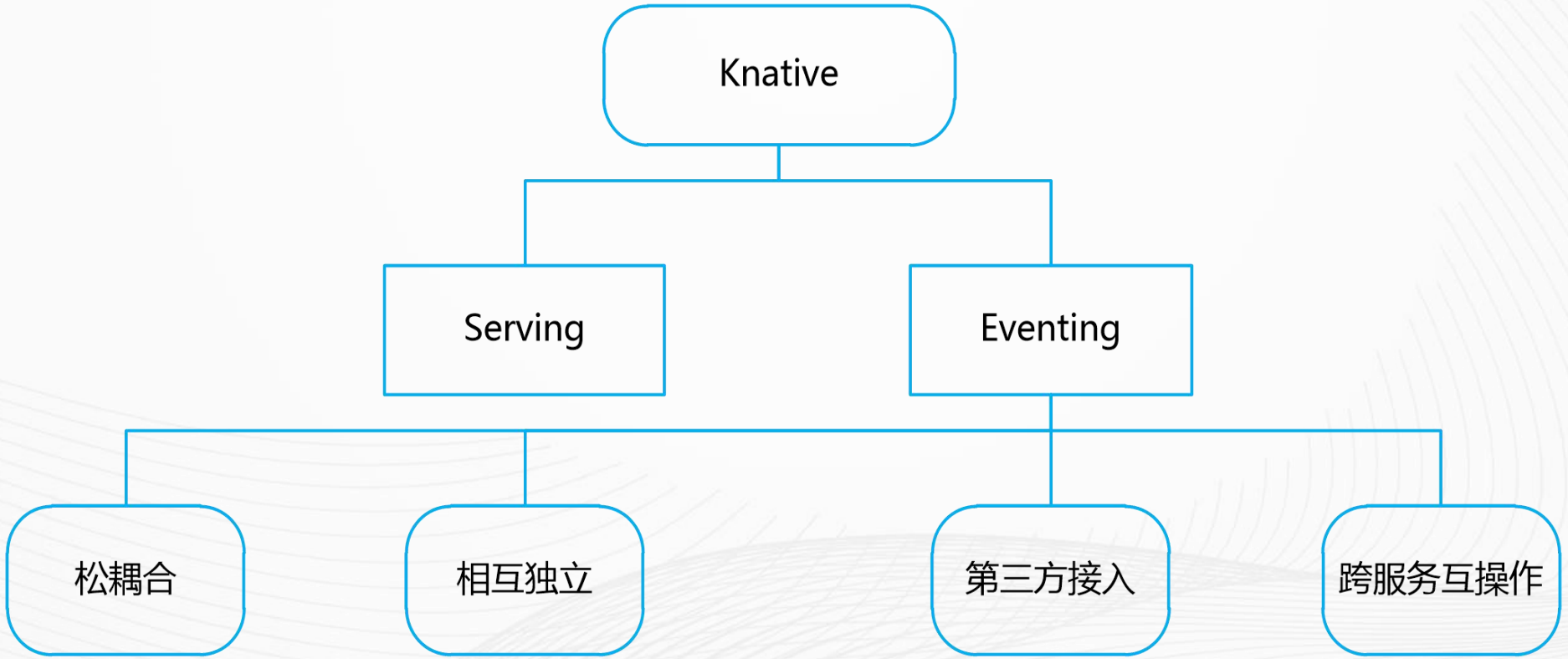
容器服务监控研发。

目录

Knative简介

GOTC

The **Knative** project provides a set of Kubernetes components that introduce event-driven and serverless capabilities for Kubernetes clusters.



如何实现事件生产和消费者相互独立？

CloudEvents！

标准数据格式CloudEvents

GOTC

CloudEvents：是一种以通用方式描述事件数据的规范，该规范旨在简化跨服务、跨平台事件的声明和交互。

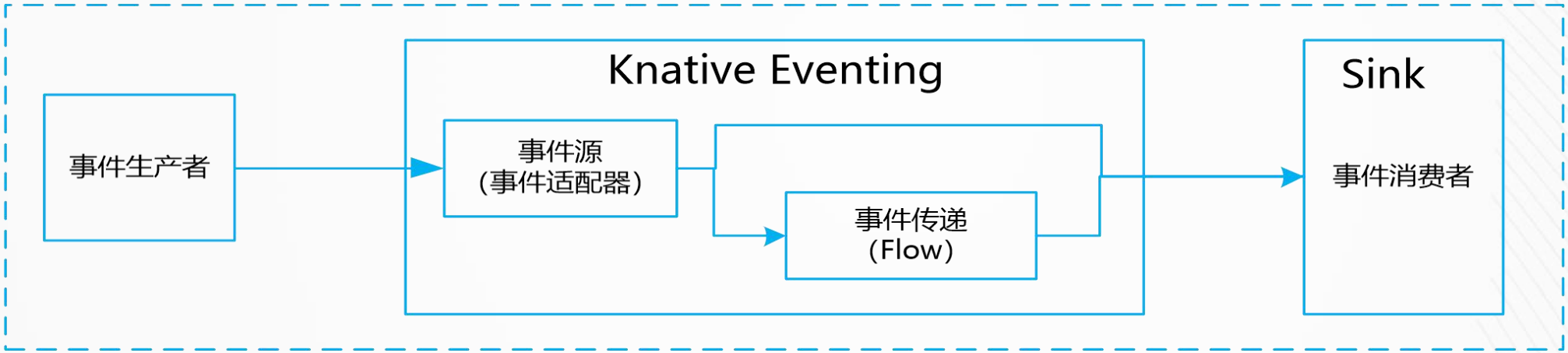
```
{
    "specversion" : "1.0",
    "type" : "com.github.pull.create",
    "source" : "https://xx/xxx",
    "subject" : "123",
    "id" : "A234-1234-1234",
    "time" : "2021-04-05T17:31:00Z",
    "comexampleextension1" : "value",
    "comexampleothervalue" : 5,
    "datacontenttype" : "text/xml",
    "data" : "<much wow=\"xml\"/>"
}
```

- ID属性：事件id；
- Source属性：标示事件发生的上下文；
- Type属性： 该属性包含一个描述事件类型的值，描述与源事件相关的事件类型；
- Specversion属性：CloudEvents规范的版本；
- Datacontenttype属性：数据类型的内容，例如：“application/json”格式、“application/xml”格式；
- Dataschema属性：指明事件数据所遵循的统一资源标识；
- Subject属性：事件主题；
- Time属性：事件发生的时间戳。

Knative Eventing系统实现原理

GOTC

Eventing功能架构



- 事件生产者： 产生事件的系统；
- 事件适配器： 数据处理的实体，将外部事件数据接入转化为CloudEvents格式，并将消息传入事件平台中；
- 事件传递：Eventing系统中实现了多种消息转发机制，主要包括直接转发，通道订阅模式和Broker/Trigger消息传递模式，复杂模式；
- 事件消费者： 接受事件的一方，如Knative服务或核心Kubernetes服务。 实现通用接口：**可寻址对象**，能够接收和通过 HTTP 传递到其status.address.url 地址的事件；**可调用对象**，能够接收通过 HTTP 传递的事件并转换事件，在 HTTP 响应中返回 0 或 1 个新事件。

Knative Eventing系统实现原理

GOTC

事件源：把事件生产者接入knative事件平台中，并把事件传送给消费者

事件源主要功能：

① 接收事件

② 将事件转化为CloudEvents格式

③ 将事件发送出去

预定义的事件源：

- ContainerSource: 实例化一个容器，通过该容器产生事件
- ApiserverSource：每次创建或更新 Kubernetes 资源时，ApiserverSource 都会触发一个新事件
- GitHubSource：GitHub 操作触发一个新事件
- CronJobSource: 通过 CronJob 产生事件
- KafkaSource: 接收 Kafka 事件并触发一个新事件
- RabbitMQ：将RabbitMQ消息带入Knative
- SinkBinding：使用Kubernetes提供的资源（例如，Deployment，Job，DaemonSet，StatefulSet）作事件源

Knative Eventing系统实现原理

GOTC

Knative Eventing自定义的资源类型

- Channel：实现事件的转发和持久存储，支持不同的技术如Kafka channel；
- Subscription：事件订阅者，即事件转发的目的地；
- Parallel: 事件同时转发给多个订阅者的一种机制；
- Sequence: 事件依次经过多个订阅者的一种机制；
- Broker：可以接受事件、并把事件转发到订阅者；
- Trigger：对某种事件的订阅；
- ContainerSource: 实例化一个容器，通过该容器产生事件；

Knative Eventing系统实现原理

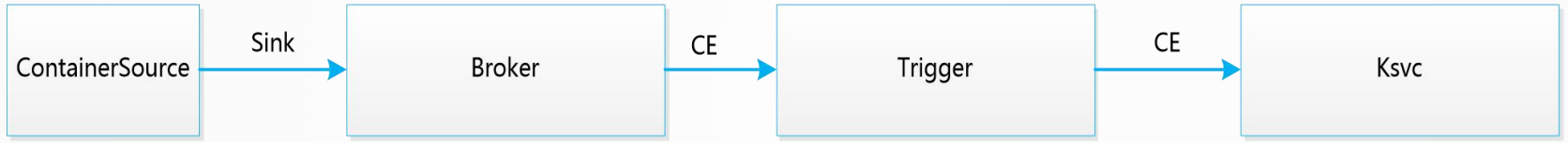
GOTC

Knative Eventing组件

Knative Eventing系统实现原理

GOTC

Broker/Trigger创建实例说明




```
apiVersion: sources.knative.dev/v1
kind: ContainerSource
metadata:
  name: onest-source
  namespace: b4be0...0ad
spec:
  template:
    spec:
      containers:
        - image: receive_adapter:v0.1
          name: receive-adapter
  sink:
    ref:
      apiVersion: eventing.knative.dev/v1
      kind: Broker
      name: onest-broker
```

```
apiVersion: eventing.knative.dev/v1
kind: Broker
metadata:
  annotations:
    eventing.knative.dev/broker.class:
      MTChannelBasedBroker
  name: onest-broker
  namespace: b4be0...0ad
spec:
  config:
    apiVersion: v1
    kind: ConfigMap
    name: config-br-kafka-channel
    namespace: knative-eventing
```

```
apiVersion: eventing.knative.dev/v1
kind: Trigger
metadata:
  name: test-trigger
  namespace: b4be0...0ad
spec:
  broker: onest-broker
  filter:
    attributes:
      type: dev.knative.sources.ping
  subscriber:
    ref:
      apiVersion:serving.knative.dev/v1
      kind: Service
      name: my-service
```

Knative Eventing系统实现原理

GOTC

Knative Eventing ingress转发事件

```
[root@master1 ~]# kubectl get containersource -n b4be0500-c445-11ea-8af1-4c32758b00ad
NAME           READY   REASON   SINK
           AGE
onest-source   True             http://broker-ingress.knative-eventing.svc.cluster.local/b4be0500-c445-11ea-8af1-4c32758b00ad/one
st-broker   211d
```

```
start := time.Now()
channelURI := &url.URL{
    Scheme: "http",
    Host:   fmt.Sprintf( format: "%s-kne-trigger-kn-channel.%s.svc.%s", brokerName, brokerNamespace, utils.GetClusterDomain
    Path:   "/",
}
sendingCTX := utils.SendingContextFrom(ctx, tctx, channelURI)

rctx, _, err := h.CeClient.Send(sendingCTX, event)
```

```
[root@master1 ~]# kubectl get kafkachannel -n b4be0500-c445-11ea-8af1-4c32758b00ad
NAME                        READY   REASON   URL
              AGE
onest-broker-kne-trigger   True             http://onest-broker-kne-trigger-kn-channel.b4be0500-c445-11ea-8af1-4c32758b00ad.svc.c
luster.local   170d
```

```
[root@master1 ~]# kubectl get svc -n b4be0500-c445-11ea-8af1-4c32758b00ad | grep onest-broker-kne-trigger-kn-channel
onest-broker-kne-trigger-kn-channel                    ExternalName   <none>          kafka-ch-dispatcher.knative-eventing.svc.
cluster.local   <none>                                 170d
```

Knative Eventing系统实现原理

GOTC

Knative Eventing dispatcher转发事件

```
for _, cc := range config.ChannelConfigs {
    channelRef := eventingchannels.ChannelReference{
        Name:      cc.Name,
        Namespace: cc.Namespace,
    }
    for _, subSpec := range cc.Subscriptions {
        newSubs = append(newSubs, subSpec.UID)
        // Check if sub already exists
        exists := false
        for _, s := range d.channelSubscriptions[channelRef] {
            if s == subSpec.UID {
                exists = true
            }
        }
        if !exists {
            if err := d.subscribe(channelRef, subSpec); err != nil {
                failedToSubscribe[subSpec.UID] = err
            }
        }
    }
```

```
  resourceVersion: "508175962"
  selfLink: /apis/messaging.knative.dev/v1beta1/namespaces/default/inmemory
  uid: 0a815476-4e10-462e-a1ef-cfaed6b48bab
spec:
  subscribers:
  - generation: 1
    replyUri: http://broker-ingress.knative-eventing.svc.cluster.local/defa
    subscriberUri: http://broker-filter.knative-eventing.svc.cluster.local/
-29d
    uid: 52659027-c4f3-44c2-b297-1b22f783cb18
  - generation: 1
    replyUri: http://broker-ingress.knative-eventing.svc.cluster.local/defa
    subscriberUri: http://broker-filter.knative-eventing.svc.cluster.local/
-2f
status:
  address:
    url: http://test-b-kne-trigger-kn-channel.default.svc.cluster.local
  conditions:
```

Knative Eventing系统实现原理

GOTC

Knative Eventing filter转发事件

```
  selfLink: /apis/messaging.knative.dev/v1beta1/namespaces/default/subscriptions/test-b-hello-display-dcb13422-bae9-4d64-8184-29
f87a7ba53
  uid: 52659027-c4f3-44c2-b297-1b22f783cb18
spec:
  channel:
    apiVersion: messaging.knative.dev/v1beta1
    kind: InMemoryChannel
    name: test-b-kne-trigger
  reply:
    ref:
      apiVersion: eventing.knative.dev/v1alpha1
      kind: Broker
      name: test-b
      namespace: default
  subscriber:
    uri: http://broker-filter.knative-eventing.svc.cluster.local/triggers/default/hello-display/dcb13422-bae9-4d64-8184-29df87a7
a53
```

subscription

```
    // tctx.URI is actually the path...
    triggerRef, err := path.Parse(tctx.URI)
    if err != nil {
        r.logger.Info( msg: "Unable to parse path as a trigger", zap.Error(err), zap.String( key: "pat
        return errors.New( text: "unable to parse path as a Trigger")
    }
    ...
    responseEvent, err := r.sendEvent(ctx, tctx, triggerRef, &event)
    if err != nil {
```

Eventing系统消息传递机制

GOTC

01 直接转发

02 通道订阅模式

03 Broker/Trigger消息传递模式

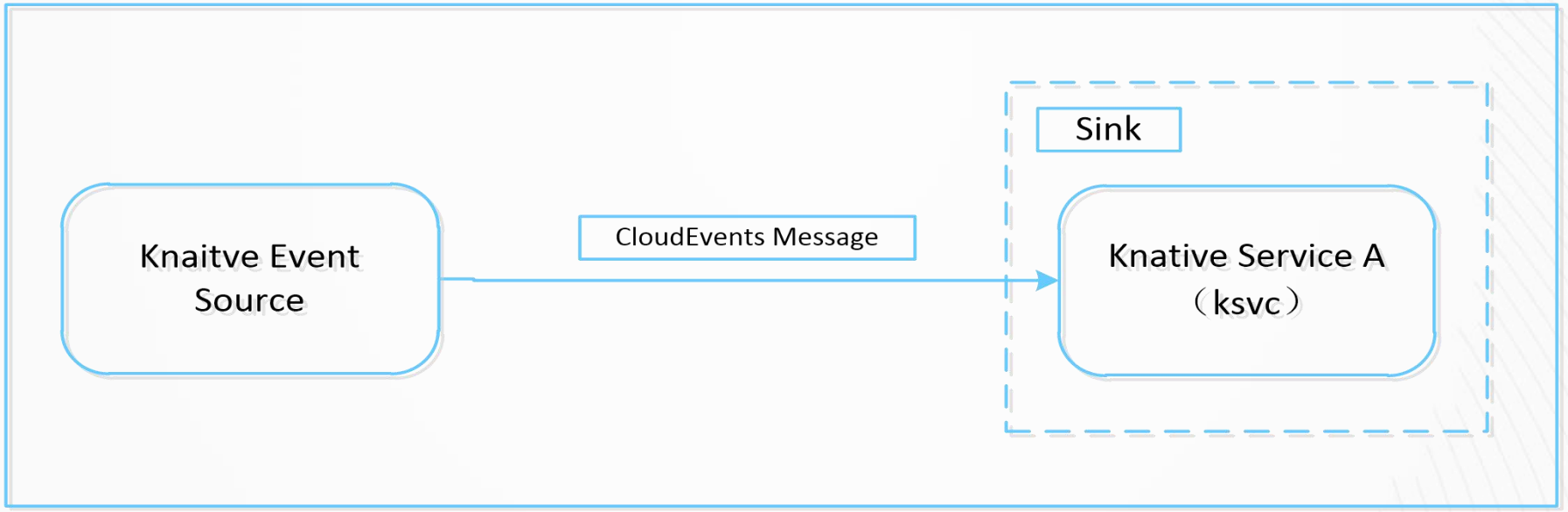
04 序列转发

05 并行转发

Eventing系统消息传递机制

GOTC

直接转发

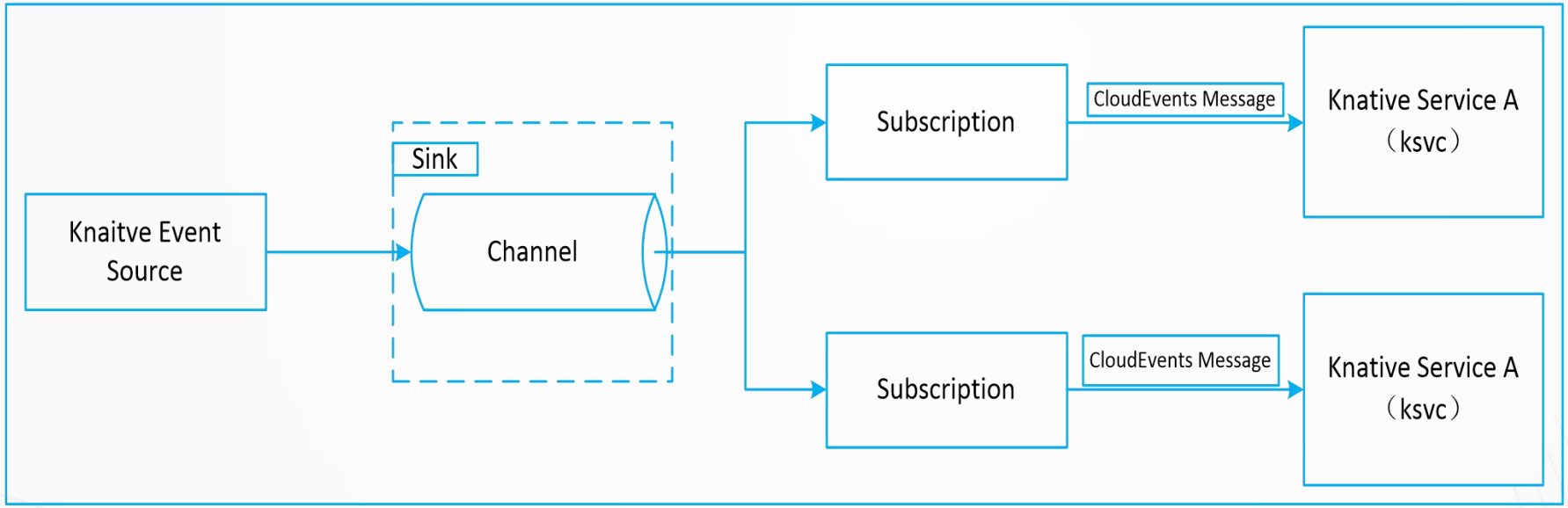


事件从源直接传递到单个服务，该服务必须是可寻址端点，包括Knative服务或核心Kubernetes服务。在这种情况下，如果目标服务不可用，则源负责重试或排队事件。如PingSource等自定义资源。

Eventing系统消息传递机制

GOTC

通道订阅模式

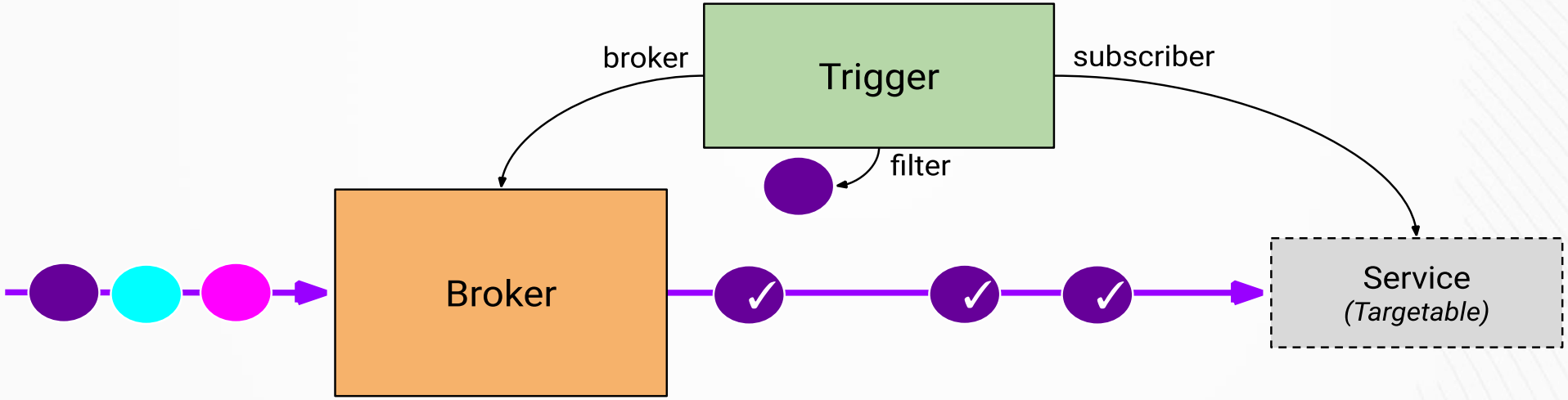


Channel是事件生产者与事件消费者的中间层，它从事件源收集事件并分发给事件订阅者。Subscription对象指定一个事件消费者对事件的订阅，通过Subscription来指定需要分发给哪些消费者。通道和订阅模式无法过滤消息。

Eventing系统消息传递机制

GOTC

Broker/Trigger消息传递模式

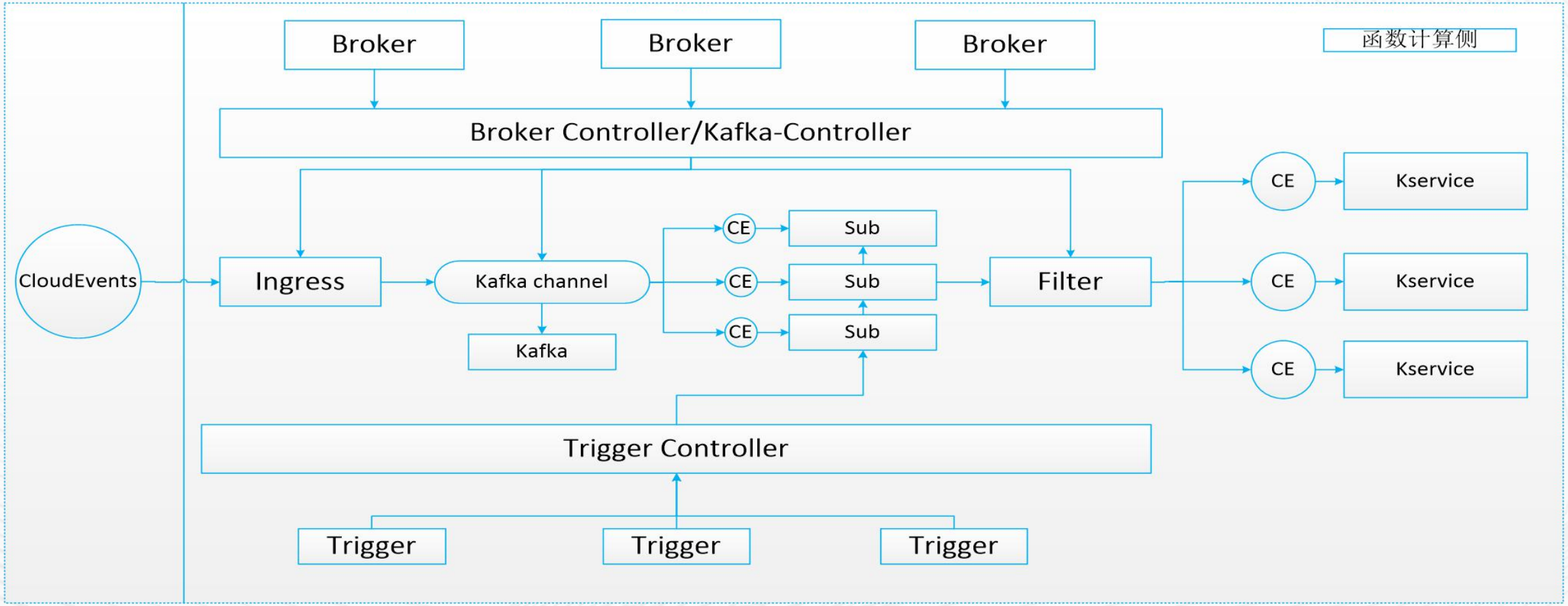


- 引入Broker和Trigger的目的，是为了搭建一个黑盒子，将具体的实现隐藏起来；
- Broker如同事件桶，接入各种不同的事件，这些事件通过不同的属性来过滤；
- Trigger描述了一个过滤器，只有通过了过滤器选择的事件，才可以被发送给消费者。

Eventing系统消息传递机制

GOTC

Broker/Trigger原理



Eventing系统消息传递机制

GOTC

Sequence

- **Sequence** 提供一种定义调用功能有序列表的方法，每个Step都可以修改，过滤或创建新的事件。序列在后台创建Channels和Subscriptions。

- **Sequence Spec 包括 3 个部分**：

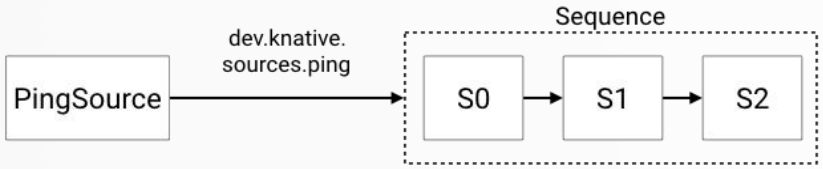
- **steps**： 在 step 中定义了按照顺序执行的服务，每个服务会对应创建 Subscription；
- **channelTemplate**：指定了使用具体的那个 Channel；
- **reply**： （可选）定义了将最后一个 step 服务结果转发到的目标服务。

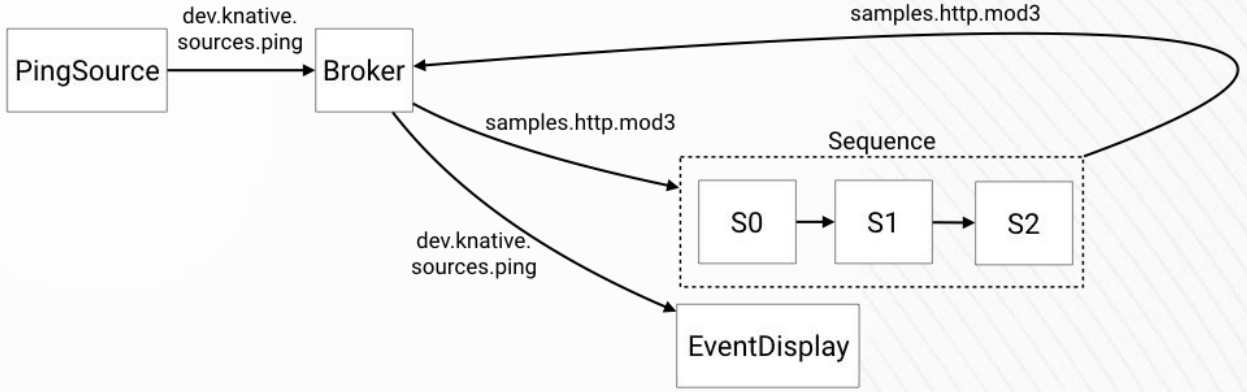
```
apiVersion: messaging.knative.dev/v1alpha1
kind: Sequence
metadata:
  name: test
spec:
  channelTemplate:
    apiVersion: messaging.knative.dev/v1alpha1
    kind: InMemoryChannel
  steps:
    - ref:
        apiVersion: serving.knative.dev/v1alpha1
        kind: Service
        name: test
  reply:
    kind: Broker
    apiVersion: eventing.knative.dev/v1alpha1
    name: test
```

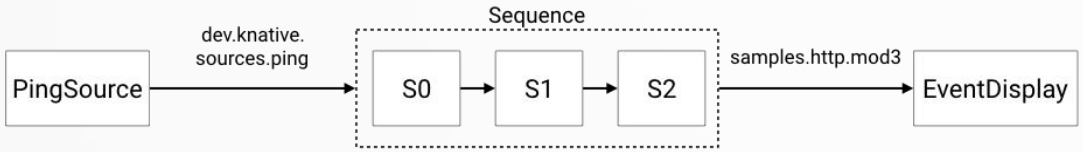
Eventing系统消息传递机制

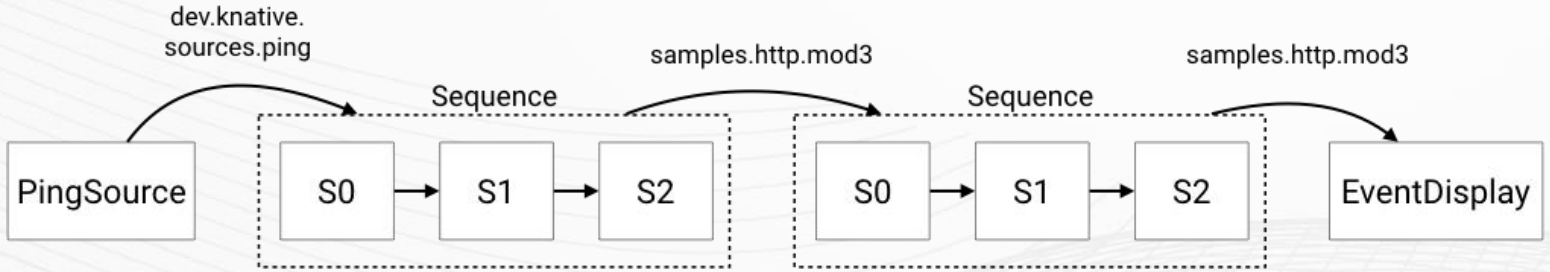
GOTC

Sequence









- 无返回事件处理
- 有返回事件处理
- 级联 Sequence
- 与Broker/Trigger模型结合使用

Eventing系统消息传递机制

GOTC

Parallel

- Parallel CRD提供了一种定义分支列表的方法，每个分支都接收相同CloudEvent。每个分支都包含一个过滤函数。

- Parallel Spec 包括 3 个部分：

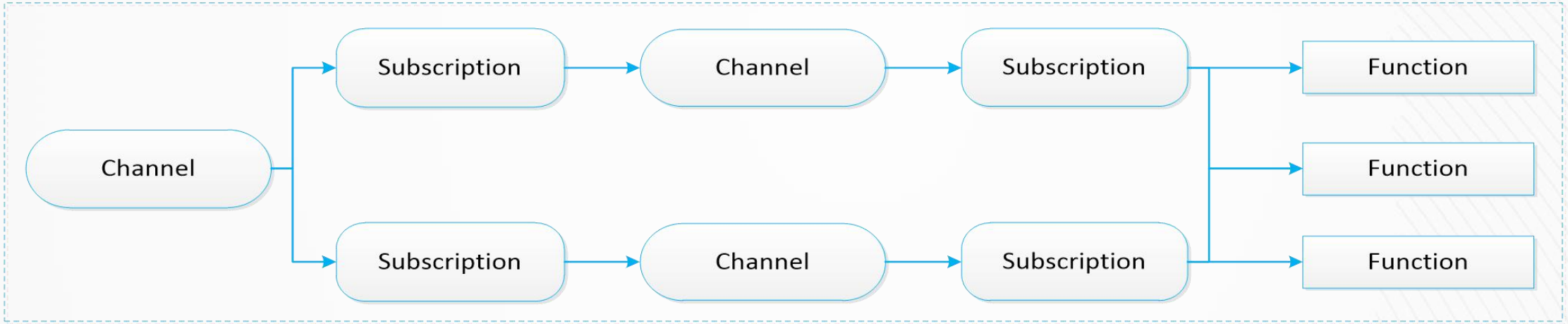
  - branches： 定义了过滤器和订阅者的列表
  - channelTemplate：指定了使用具体的Channel；
  - reply： （可选）定义当分支没有自己的结果时，将每个分支的处理结果发送到该处。

```
apiVersion: flows.knative.dev/v1
kind: Parallel
metadata:
  name: me-odd-even-parallel
spec:
  channelTemplate:
    apiVersion: messaging.knative.dev/v1
    kind: InMemoryChannel
  branches:
    - filter:
        uri: "http://me-even-odd-switcher.default.svc.cluster.local/0"
      subscriber:
        ref:
          apiVersion: serving.knative.dev/v1
          kind: Service
          name: me-even-transformer
    - filter:
        uri: "http://me-even-odd-switcher.default.svc.cluster.local/1"
      subscriber:
        ref:
          apiVersion: serving.knative.dev/v1
          kind: Service
          name: me-odd-transformer
  reply:
    ref:
      apiVersion: serving.knative.dev/v1
      kind: Service
      name: me-event-display
```

Eventing系统消息传递机制

GOTC

Parallel



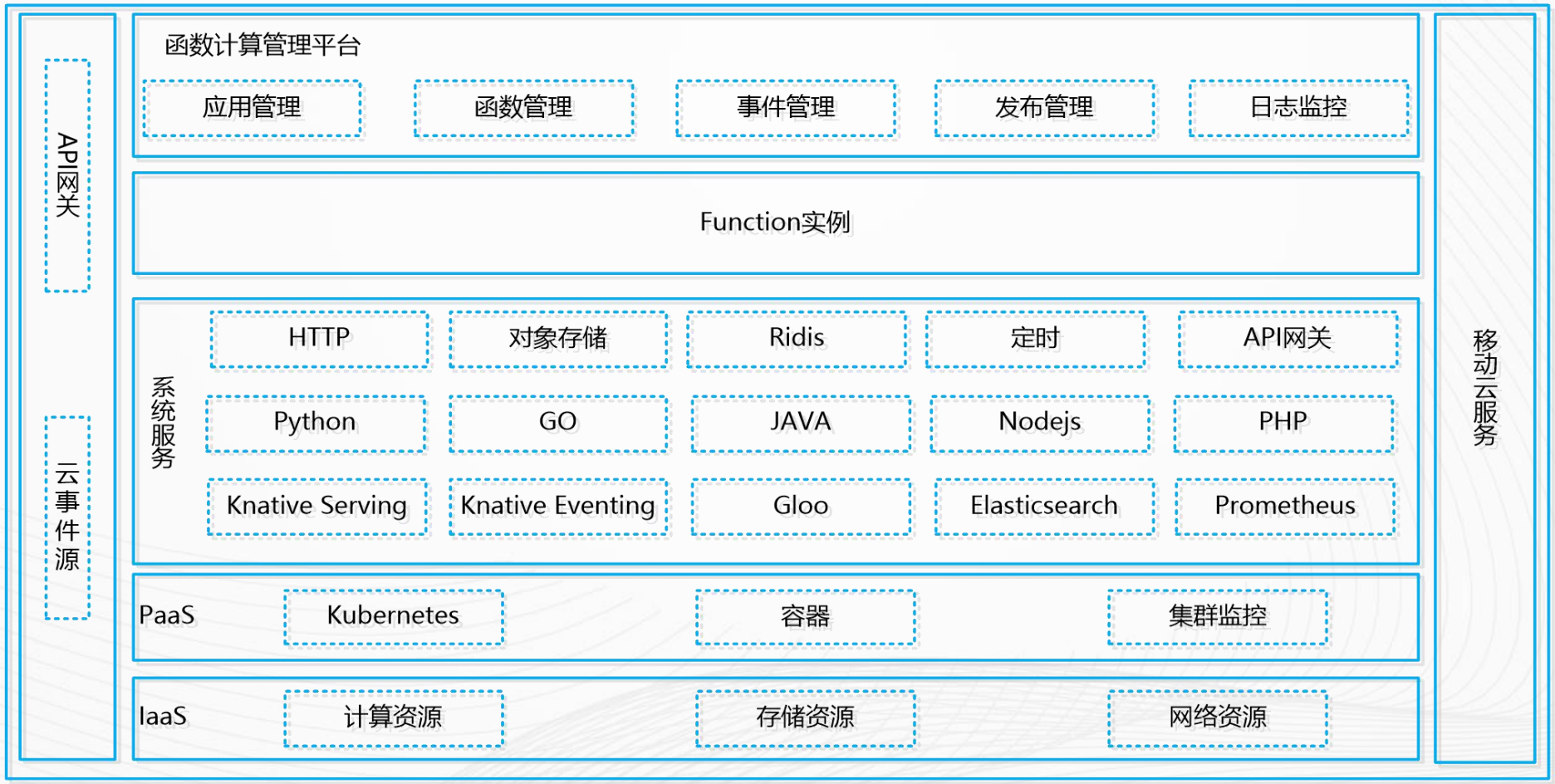
- 首先为 Parallel 创建一个全局的 Channel。然后为每一个实例创建一个过滤 Channel；
- 在每个分支中做了如下处理：
  - 为全局的 Channel创建一个 Subscription，订阅条件为filter信息，并且把 reply 响应发送给当前示例中的过滤 Channel；
  - 为过滤 Channel 创建一个 Subscription，将订阅信息发送给每个示例的 Reply。如果当前示例中没有设置Reply，则发送的全局Reply。

Eventing在移动云函数计算
应用实践

移动云函数计算的应用实践

GOTC

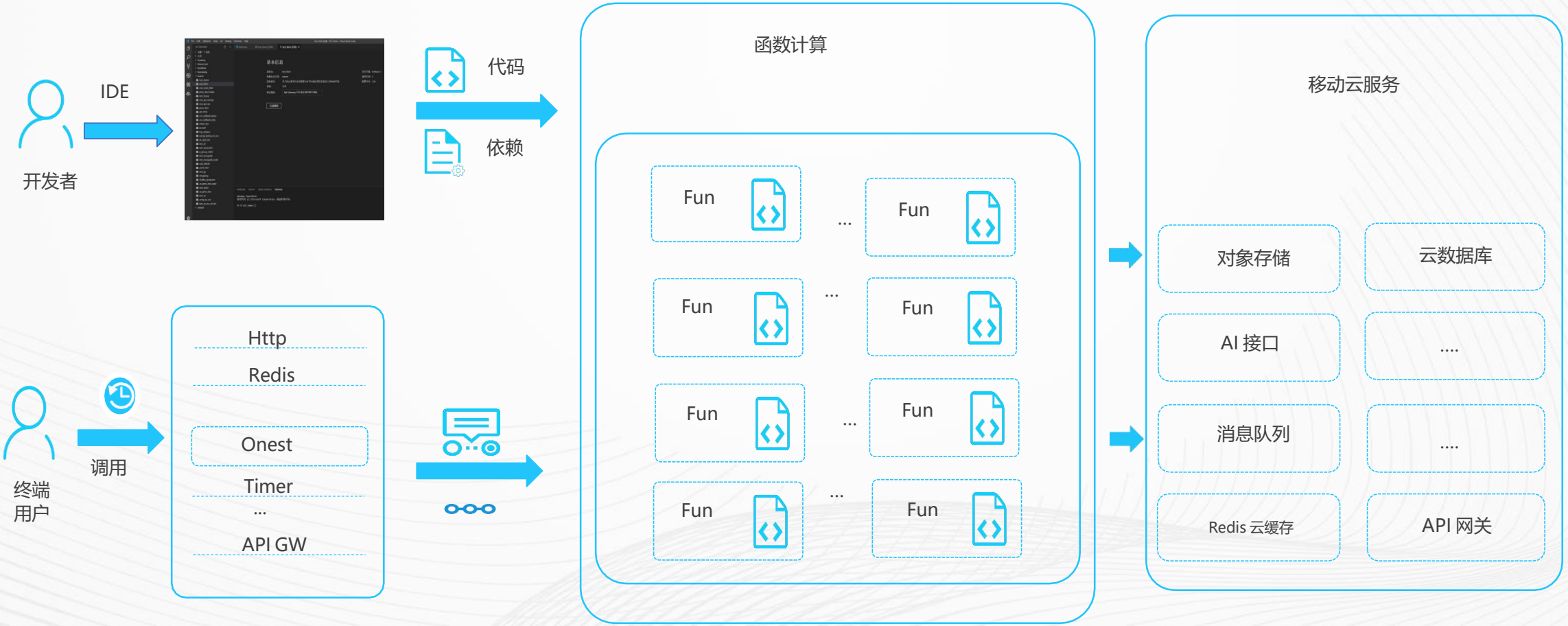
移动云函数计算技术架构



移动云函数计算的应用实践

GOTC

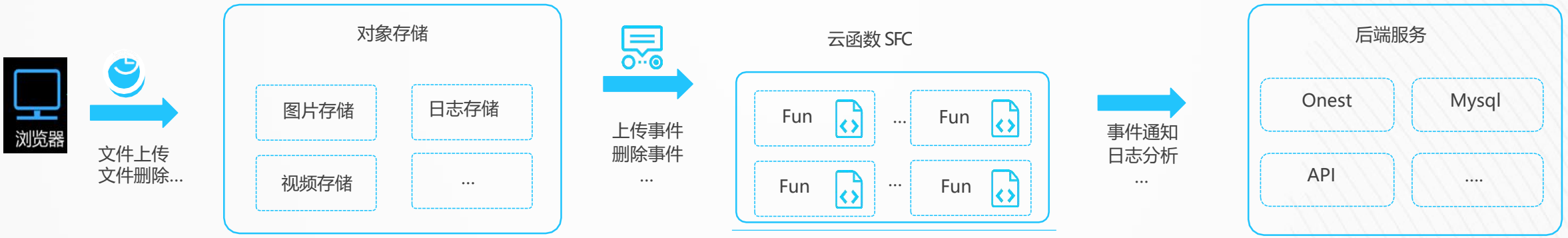
移动云函数计算应用



移动云函数计算的应用实践

GOTC

移动云函数计算对象存储触发器



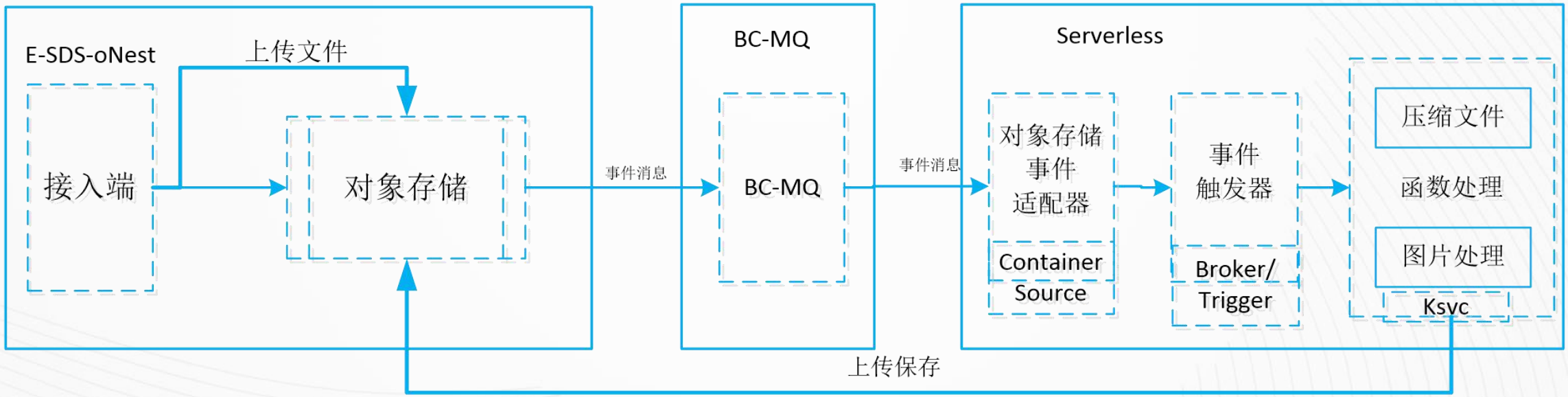
客户端上传/删除图片/视频等文件时，会在对象存储产生上传/删除事件，该事件可以转发到云函数，触发函数运行具体的业务逻辑。

在移动云函数中，可以基于不同的编程语言（Python/NodeJS/PHP/JAVA/GO），撰写自定义业务逻辑，如文件解压， 日志分析等。

在移动云函数中，调用云上的其他服务，如事件通知，可以把自定义事件投递到某个数据库，或将对象传到对象存储。

移动云函数计算的应用实践

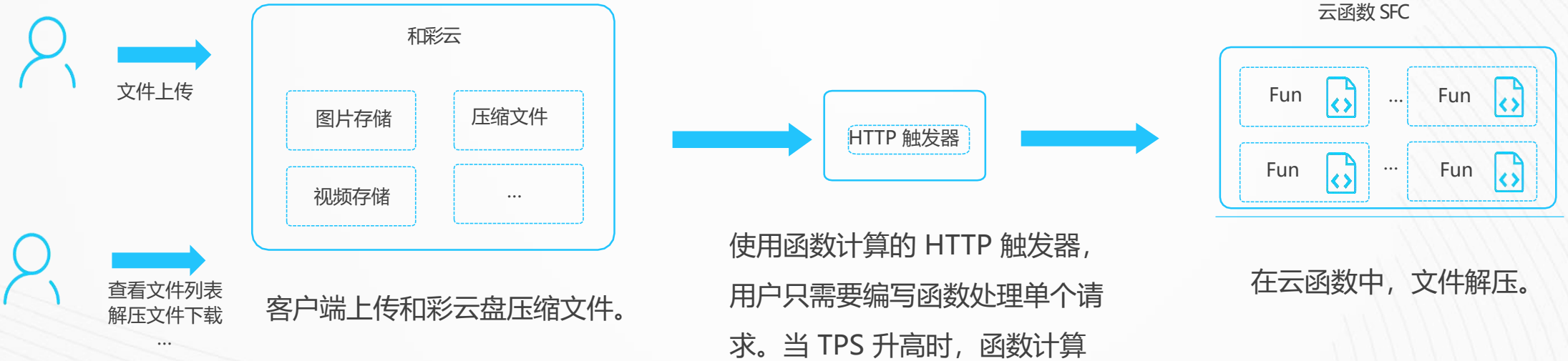
移动云函数计算对象存储触发器实现原理



移动云函数计算的应用实践

GOTC

移动云函数计算http触发器实践



客户端上传和彩云盘压缩文件。

使用函数计算的 HTTP 触发器，用户只需要编写函数处理单个请求。当 TPS 升高时，函数计算会自动扩容计算资源执行多个函数实例处理请求。

在云函数中，文件解压。

移动云函数计算的应用实践

GOTC

移动云函数计算图片处理实践

GOTC

THANKS

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE