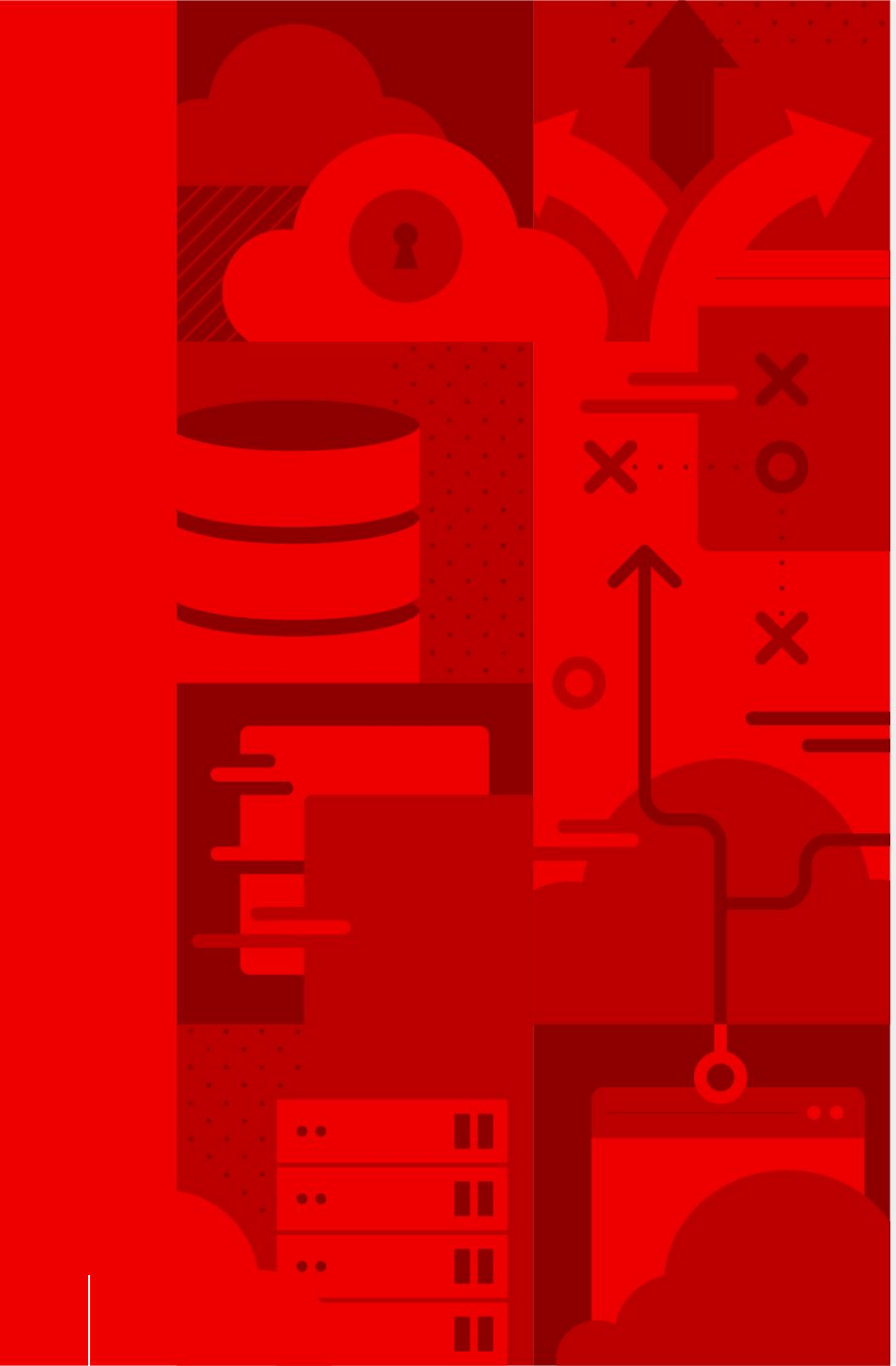


Knative

Kubernetes-based platform to manage modern serverless workloads.

Matthias Wessendorf @mwessendorf
Roland Huß @ro14nd
Principal Software Engineers, Red Hat

Red Hat

What is serverless again ?

*"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"*

-- CNCF Definition, https://www.cncf.io/blog/2018/02/14/cncf-takes-first-step-towards-serverless-computing/

Wait... wat ?

Red Hat


Knative

Kubernetes-based platform to **build**, **deploy**, and **manage** modern serverless workloads.

https://knative.dev

Knative Components

~~Build~~

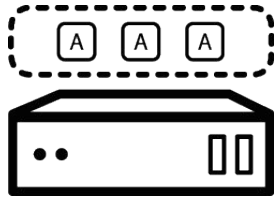
Replaced by
Tekton pipelines

Serving

A request-driven model that serves the container with your application and can "scale to zero".

Eventing

Common infrastructure for consuming and producing events that will stimulate applications.




Build

Proposal: Knative Build deprecation in favor of Tekton Pipelines #614

Open **vdemeester** opened this issue 16 days ago · 5 comments

vdemeester 16 days ago • edited Member

Objective

Propose to mark knative/build as deprecated and drive user to the Tekton project as their building engine. This doc should be used as a forum for external comment and questions, to resolve any concerns before the Steering Committee decides whether to adopt the proposal.

Background

Starting with `v1beta1` API of knative (serving), the embedded build feature will be removed from the Knative Service definition, see the proposal here.
This removes Serving *optional* dependency on Knative Build, making Knative Build fully decoupled from the rest of the Knative components and only responsible to build images that will be using in services later on. This responsability is shared with any projects capable of building images in Kubernetes.

The Tekton Pipeline (tektoncd/pipeline) project originates from Knative project (was `knative/build-pipeline`) and is enhancing Knative Build functionality, providing advanced CI/CD features on top of the base that Knative laid.

https://github.com/knative/build/issues/614


Serving

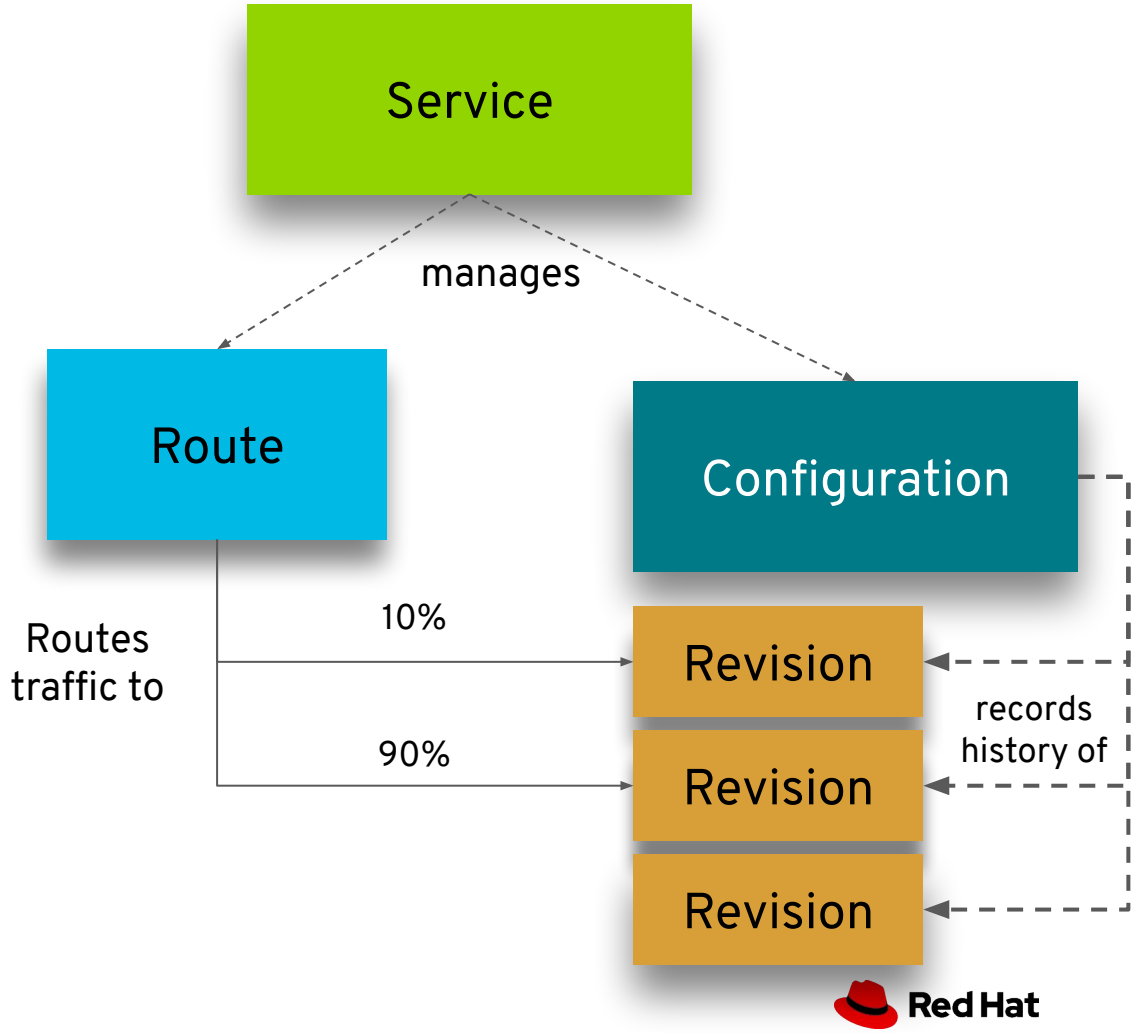
Route, scale-to-zero and track application revisions with ease.

Knative Serving Concepts

- Separation of code and configuration
- Immutable Revisions
- Autoscaling include scale-to-zero
- Traffic Splitting
- Simplified and opinionated deployment model
  - Single Port
  - No PersistentVolumes
  - Single Container
- Callable by Knative eventing

Knative Serving Resources

- **Configurations** represent the ‘floating HEAD’ of a history of **Revisions**
- **Revisions** represent immutable snapshot of code and configuration
- **Routes** configure ingress over a collection of Revisions
- **Services** (not K8s services) are top-level entities that manage a set of Routes and Configurations



Service Pealing

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: lotto
spec:
  replicas: 1
  selector:
    matchLabels:
      app: lotto
  template:
    metadata:
      labels:
        app: lotto
    spec:
      containers:
      - image: cds19/lotto
        name: lotto
        ports:
        - containerPort: 8080
```

```yaml
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: lotto
spec:
  replicas: 1
  selector:
    matchLabels:
      app: lotto
  template:
    metadata:
      labels:
        app: lotto
    spec:
      containers:
      - image: cds19/lotto
        name: lotto
        ports:
        - containerPort: 8080
```

Red Hat

Demo


Eventing

Universal subscription, delivery, and management of events.

Knative Eventing Features

Benefits

- Event orchestration
  - Declaratively API to distribute events
- Scales from just few events to live data-streaming pipelines

Powered by:

cloudevents

Extendable

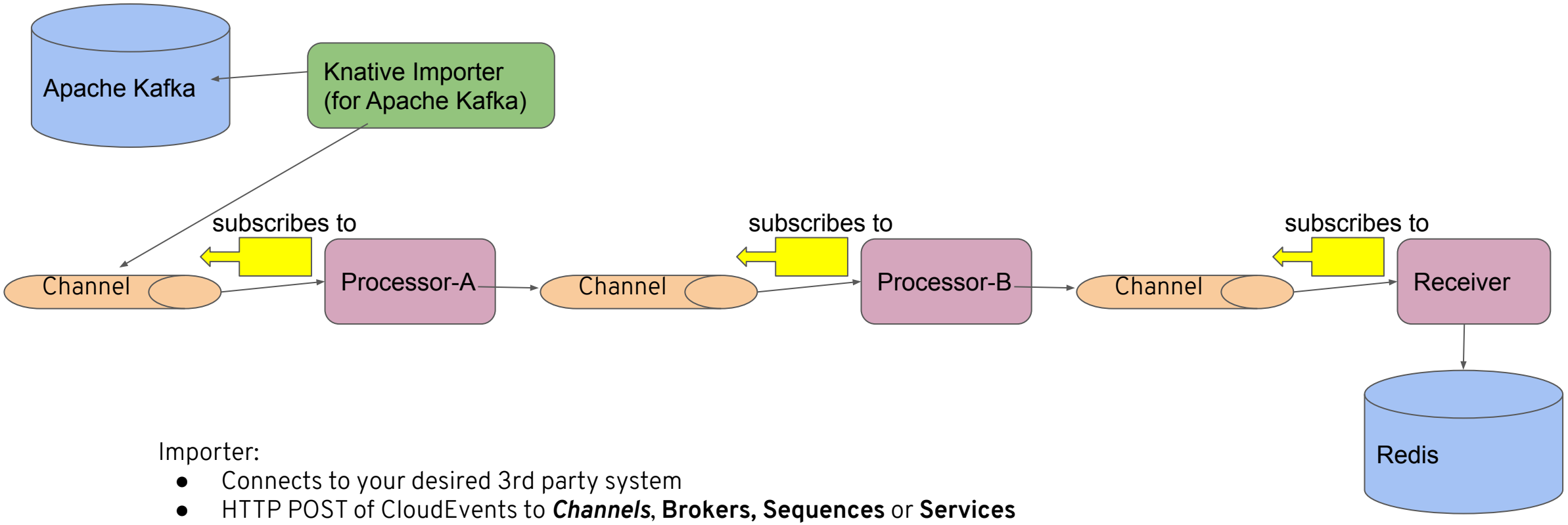
- Eventing Sources/Importers
  - Github
  - Apache Kafka
  - ...
  - Build your own
- Plugable channel implementation
  - In-Memory(default)
  - Apache Kafka
  - Google Pub-Sub
  - ...

Red Hat

Knative Eventing Source (Importer)

- Integrating 3rd party system with Knative
- Control plane
  - Create a Apache Kafka topic subscription
- Data plane
  - Pull Events from Apache Kafka topic
- Push/Pull based
- Validation of events
- Build your own*
  - CRD, using ContainerSource or emit events to (http) transport (to feed the broker)

Event Driven Flow

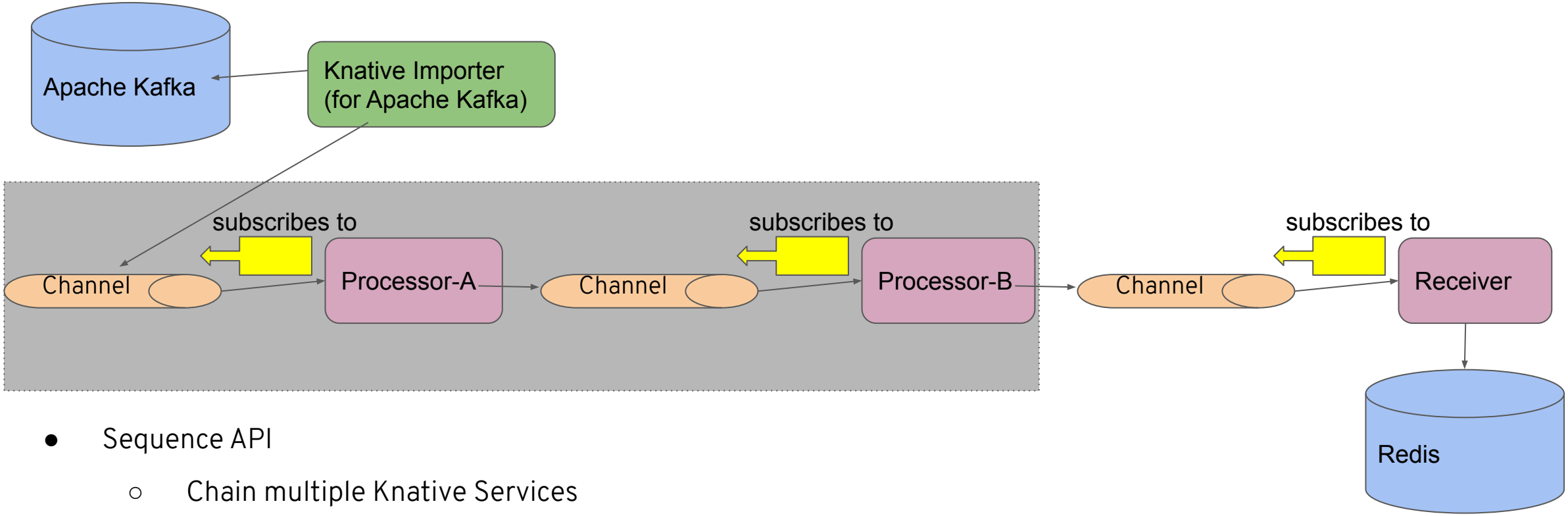


Importer:
- Connects to your desired 3rd party system
- HTTP POST of CloudEvents to ***Channels***, **Brokers, Sequences** or **Services**

Channel:
- Has n subscribers (of Knative Services)
- “Persisting” messages for consumption by Subscribers

Service:
- Receives the HTTP POST of the (CloudEvent) message
  - Optionally returns processed data (replyChannel)

Event Driven Flow



- Sequence API
  - Chain multiple Knative Services
  - Sink to Channel, Broker, Sequence or Service

A Sequence

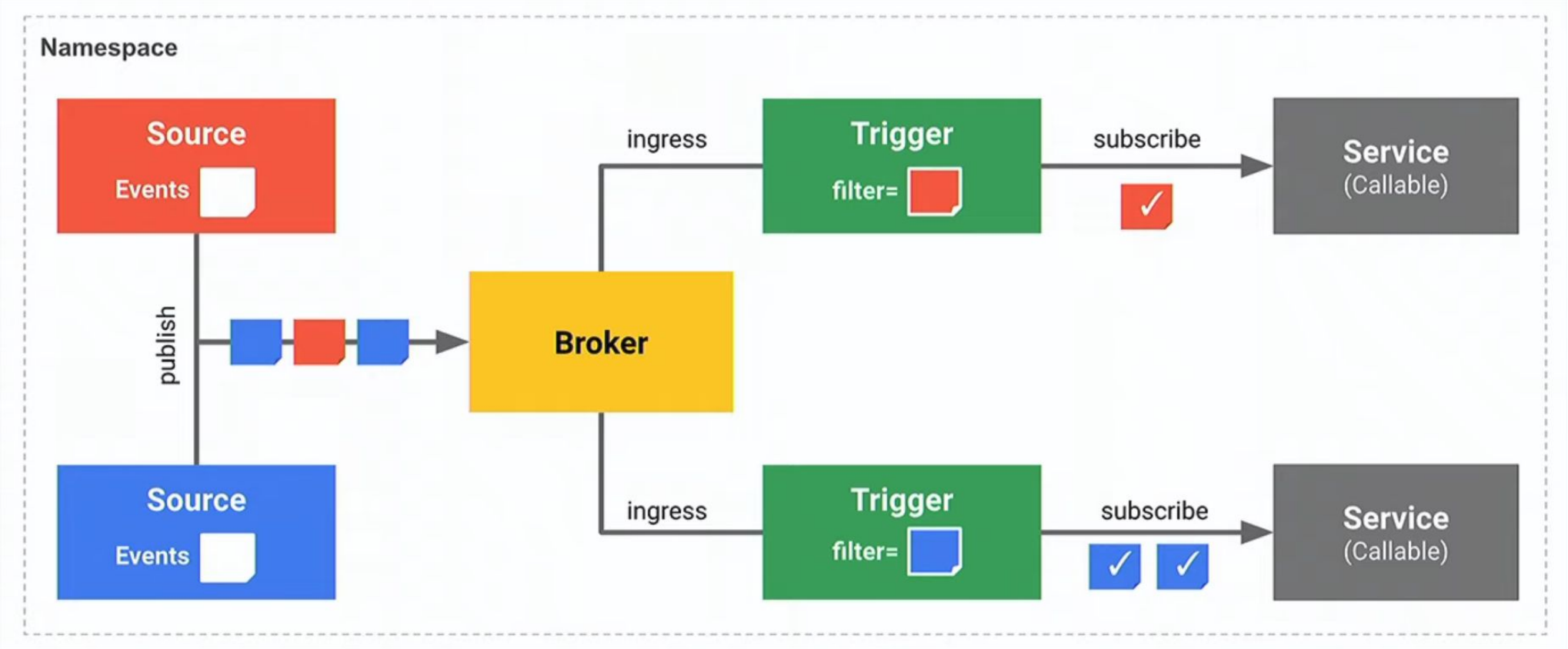
```yaml
apiVersion: sources.eventing.knative.dev/v1alpha1
kind: CronJobSource
metadata:
  name: cronjob-source
spec:
  schedule: "* * * * *"
  data: '{"message": "Hello world!"}'
  sink:
    apiVersion: messaging.knative.dev/v1alpha1
    kind: Sequence
    name: sequence
```

```yaml
apiVersion: messaging.knative.dev/v1alpha1
kind: Sequence
metadata:
  name: sequence
spec:
  channelTemplate:
    apiVersion: messaging.knative.dev/v1alpha1
    kind: InMemoryChannel
  steps:
  - ref:
      apiVersion: serving.knative.dev/v1beta1
      kind: Service
      name: processor-a
  - ref:
      apiVersion: serving.knative.dev/v1beta1
      kind: Service
      name: processor-b
  reply:
    kind: Service
    apiVersion: serving.knative.dev/v1beta1
    name: receiver
```

Knative Eventing Resources

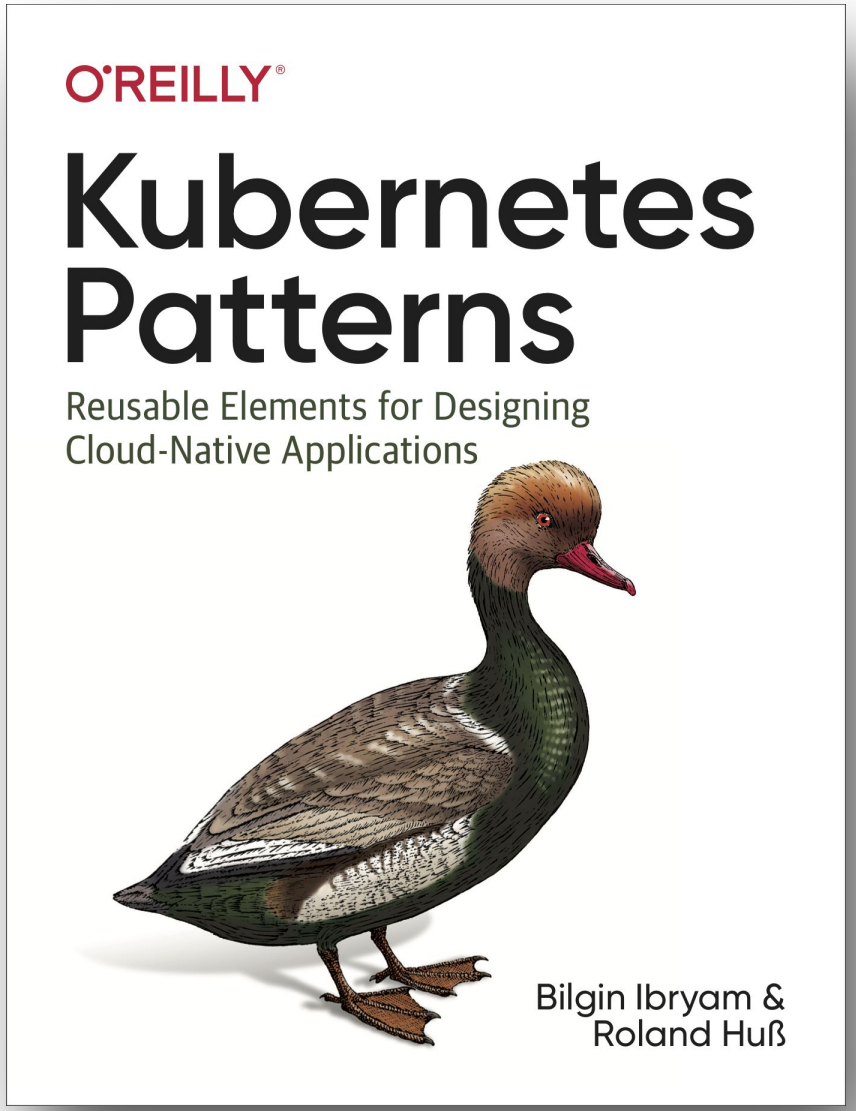
- **Broker**
  - Eventing Mesh (or Event Delivery System)
  - Connects Sources
  - Uses Channels internally
- **Trigger**
  - Filter events (e.g. type and/or source)
  - Can produce new events (returned to “Broker”)
  - Delivered as CloudEvents
- **EventRegistry**
  - EventType CRD
  - Discoverability of Events
- **Sequence**
  - chaining multiple Services, sinking to Service, Channel, Sequence or Broker

Broker & Trigger



By Ville Aikas (Google)

Demo



https://k8spatterns.io

Red Hat

Thank you

https://k8spatterns.io

https://twitter.com/ro14nd

https://twitter.com/mwessendorf

https://twitter.com/k8spatterns

Red Hat

Picture Credits

https://www.pexels.com/photo/boat-island-ocean-sea-218999/

https://unsplash.com/photos/t6t2-gXKxXM

https://unsplash.com/photos/UGMf30W28qc

https://pixabay.com/photos/hamburg-speicherstadt-channel-2976711/

https://pixabay.com/photos/beer-machine-alcohol-brewery-1513436/

https://unsplash.com/photos/9SWHIgu8A8k

https://me.me/i/aws-lambda-is-just-glorified-cgi-bin-imgflip-com-change-my-mind-d0b715592ba34b08b79452ad02783ca2

https://unsplash.com/photos/dodn_0TESN0