Knative

Kubernetes-based platform to manage modern serverless workloads.

Roland Huß @ro14nd
Principal Software Engineer, 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

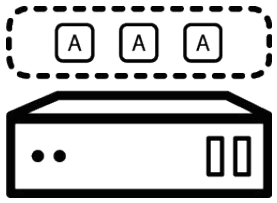
Kubernetes-based platform to **deploy** and **manage** modern serverless workloads.

https://knative.dev

Knative Components

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.

Background Information

- Started as Open Source Project mid-2018 by Google
- Community + Company Driven
  - https://github.com/knative
  - https://knative.dev
  - Support by Google, Red Hat, IBM, VMware, Triggermesh, SAP ...
  - Organized in 9 Working Groups with weekly meetings
  - No foundation
- Releases
  - Current: Serving v0.13.0, Eventing v0.13.0, Client v0.13.0
  - 6 week release cadence (client one week shifted)

Try Knative

- Install from resource descriptors on Kubernetes Cluster
  - https://knative.dev/docs/install/
- Google Cloud Run (managed and on GKE)
  - https://cloud.google.com/run/
  - Free tier: (180k CPU-s, 360k GB-s, 2M Requests) per month
- Red Hat OpenShift Serverless - Technical Preview
  - https://www.openshift.com/learn/topics/serverless
  - GA in April


Serving

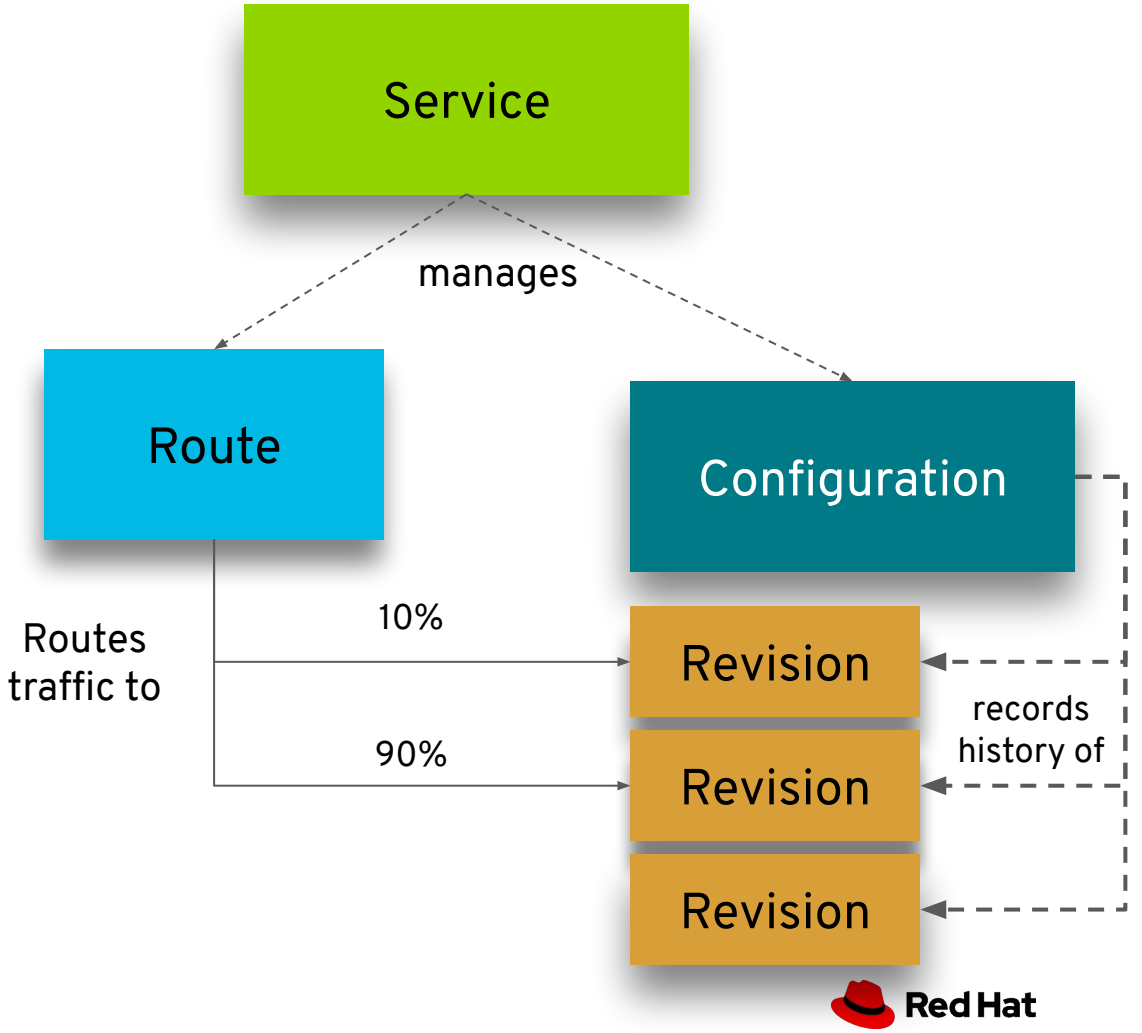
Route, scale-to-zero and track application revisions with ease.

Knative Serving Concepts

- Separation of code and configuration
- Immutable Revisions
- Autoscaling including scale-to-zero
  - https://github.com/knative/serving/blob/master/docs/scaling/SYSTEM.md
- Traffic Splitting
- Simplified and opinionated deployment model
  - Single Port
  - No PersistentVolumes
  - Single Container (about to change)
- Callable by Knative eventing

Knative Serving Resources

- **Configuration** represent the ‘floating HEAD’ of a history of **Revisions**
- **Revision** represents an immutable snapshot of code and configuration
- **Route** 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: cyberland/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: cyberland/lotto
        name: lotto
        ports:
        - containerPort: 8080
```

Demo


Eventing

Universal subscription, delivery, and management of events.

Knative Eventing Key Features

Benefits

- Declarative API for event distribution
- Based on CRD
- Flexible ways to connect Event **sources** to **sinks**
- Highly Scalable
- Powered by

cloudevents

Extendable

- Multitude of Event Sources
- SinkBinding to connect core K8s applications to Event sinks
- Plugable event transport via channels
  - In-Memory (default)
  - Apache Kafka
  - Google Pub-Sub

Red Hat

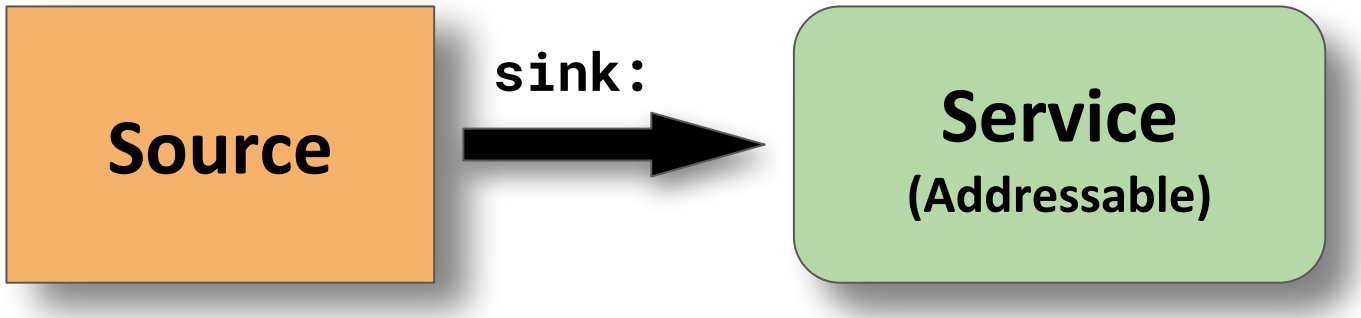
Knative Event Sources

- Integrating 3rd party systems with Knative
- More often “Adapter” than an original event source
- Declared with a Custom Resource
- Evaluated by an Operator
- Push or Pull based
- Converting custom event formats to CloudEvents

Knative Eventing Sources

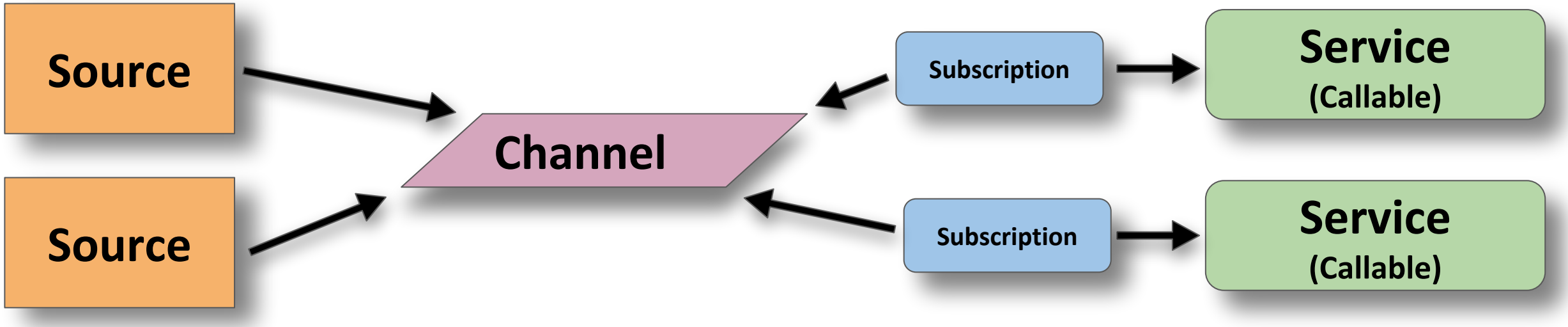
- Existing
  - **Ping:** Constant events created at a fixed schedule
  - **Apiserver:** Watching for Kubernetes resource lifecyle events
  - **SinkBinding:** Arbitrary container instantiated
  - **GitHub:** Listen on GitHub API calls
  - **Kafka:** Connect to Kafka topics of a given Kafka Cluster
  - **Camel-K:** Camel components for connecting to external systems
  - and many more: https://knative.dev/docs/eventing/sources/
- Custom
  - Write your own Source (CRD + Operator)
  - Use SinkBinding with a deployment
  - Talk to Broker directly via HTTP

Source → Service : Direct Connection



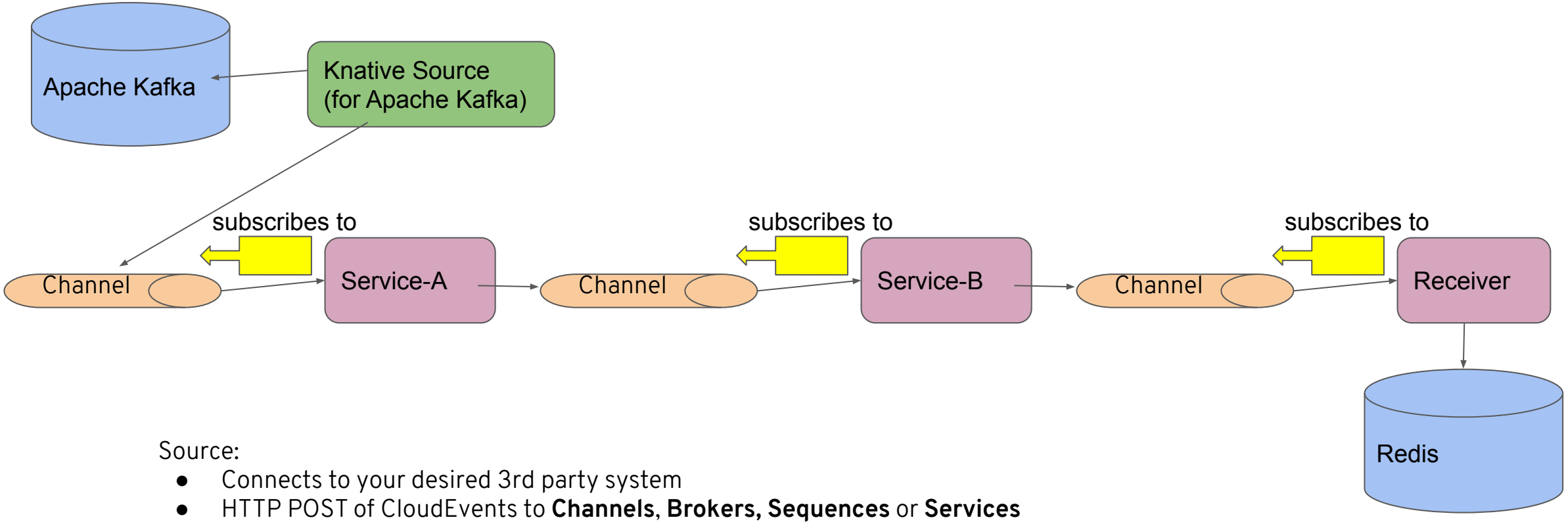
- Simplest way to get CloudEvents to a service
- Drawbacks:
  - No queuing support when service is unavailable
  - No back pressure support
  - Only one Service can consume
  - No filtering, Service gets always all events

Source → Service : Channel & Subscription



- Multiple Services can consume the same event
- Subscription can point to a reply channel (not shown here)
- Various Channel Backends available
  - In-Memory, Kafka, GCP PubSub, (write your own)
- Drawbacks:
  - Channel Infrastructure needs to be set up manually
  - No filtering, Service gets always all events

Event Driven Flow

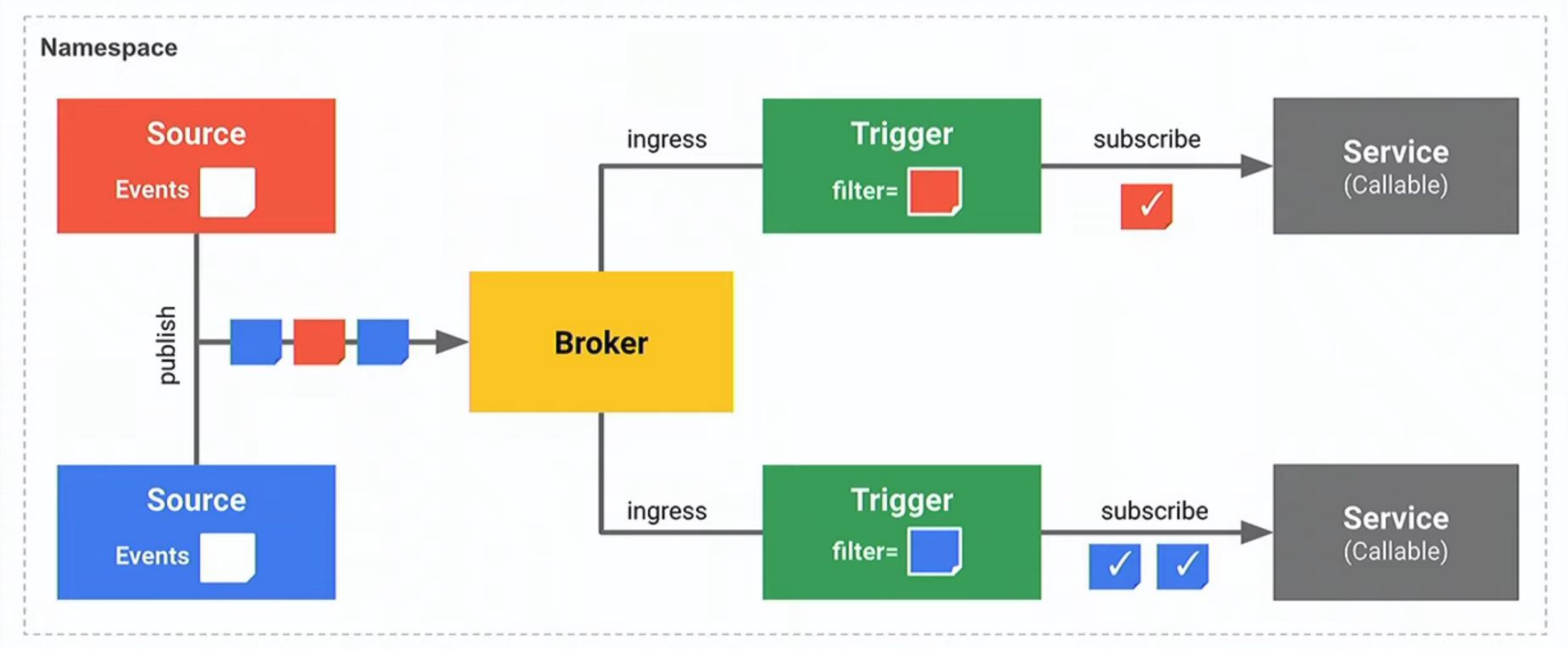


Source:
- Connects to your desired 3rd party system
- HTTP POST of CloudEvents to **Channels**, **Brokers, Sequences** or **Services**

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)

Source → Service: Broker & Trigger



By Ville Aikas (Google)

Source → Service: Broker & Trigger

- **Broker**
  - Eventing Mesh (or Event Delivery System)
  - Connects Sources with Sinks
  - Uses Channels internally
- **Trigger**
  - Filter events (e.g. type and/or source)
  - Can produce new events (returned to “Broker”)
  - Delivered as CloudEvents

Demo

More Knative Eventing

- **EventRegistry**
  - EventType CRD
  - Discoverability of Events
- **Sequence**
  - Chaining multiple Services
  - Sinking to an “Addressable” (Service, Channel, Sequence, Broker …)
- **Parallel**
  - Branching of events with filters
  - Allows to implement conditional processing

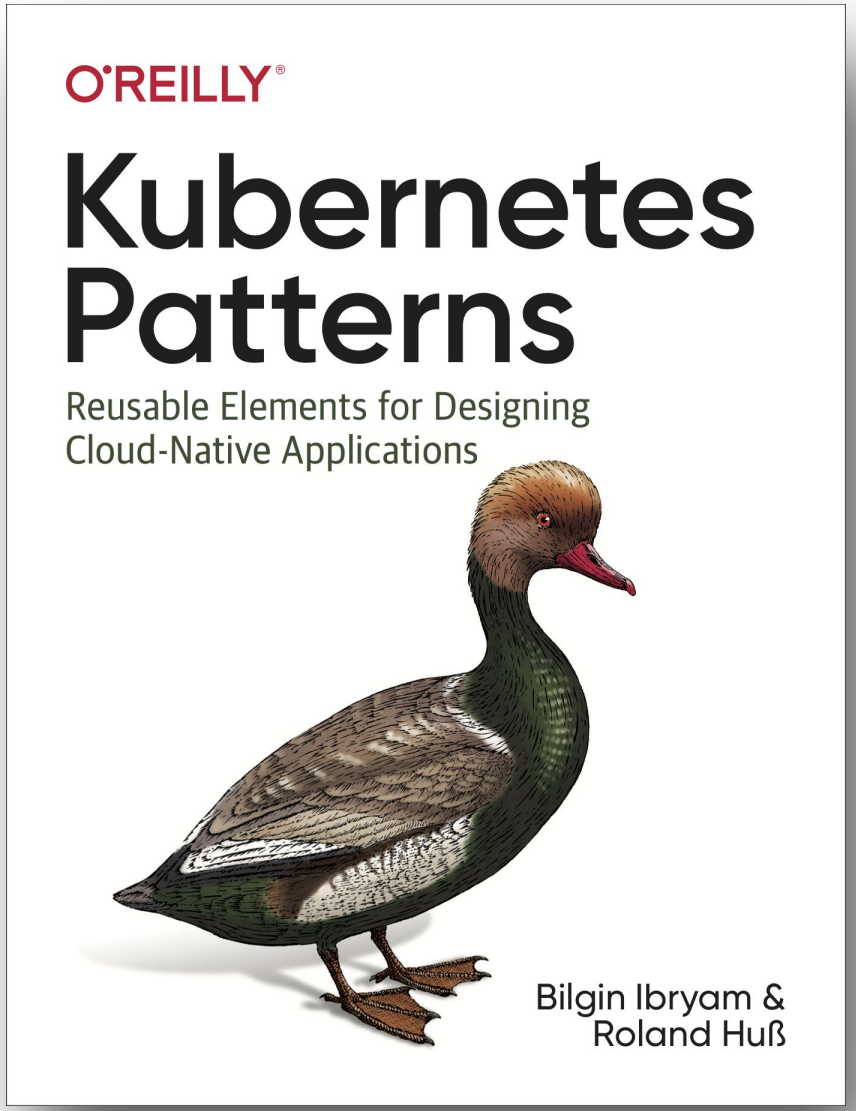
Summary

Summary

- Knative Serving
  - Simplified Deployment for stateless workloads
  - Traffic based autoscaling including Scale-to-Zero
  - Traffic splitting for custom rollout / rollback scenarios
- Knative Eventing
  - External Triggers for feeding Knative Services
  - Based on CloudEvents
  - Backed by proven messaging systems
  - Flexible messaging setup

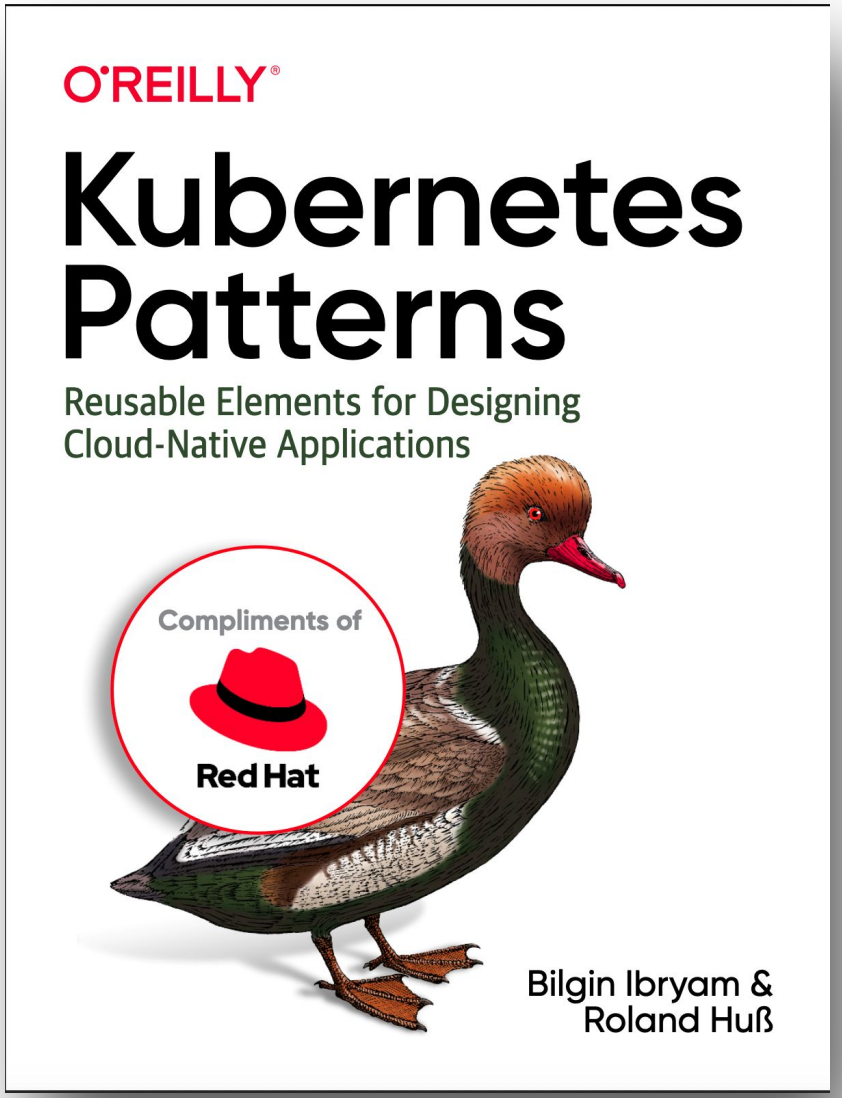

Javamagazin
SERVERLESS WORKLOADS AUF KUBERNETES
native
EVENTING | SERVING | TEKTON
Inklusive Knative-Infografik

Red Hat



https://k8spatterns.io

Red Hat



http://bit.ly/kubernetes-patterns-red-hat

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

Red Hat