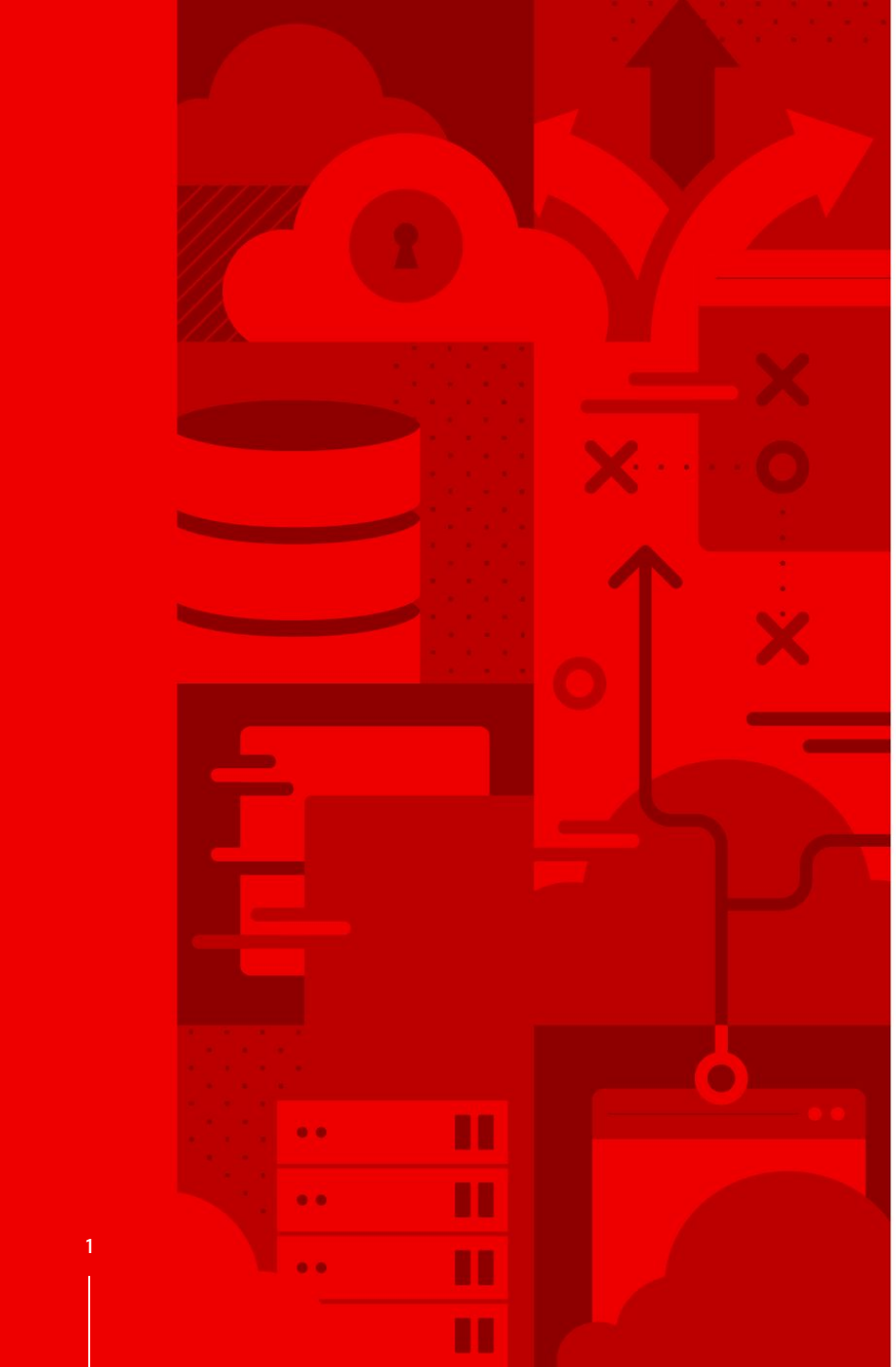


Knative: Serverless and Eventing

Ludovic Cavajani
Platform Engineer

Red Hat

What we'll discuss today

- Intro
- Knative Serving
- Demo
- Knative Eventing
- Demo

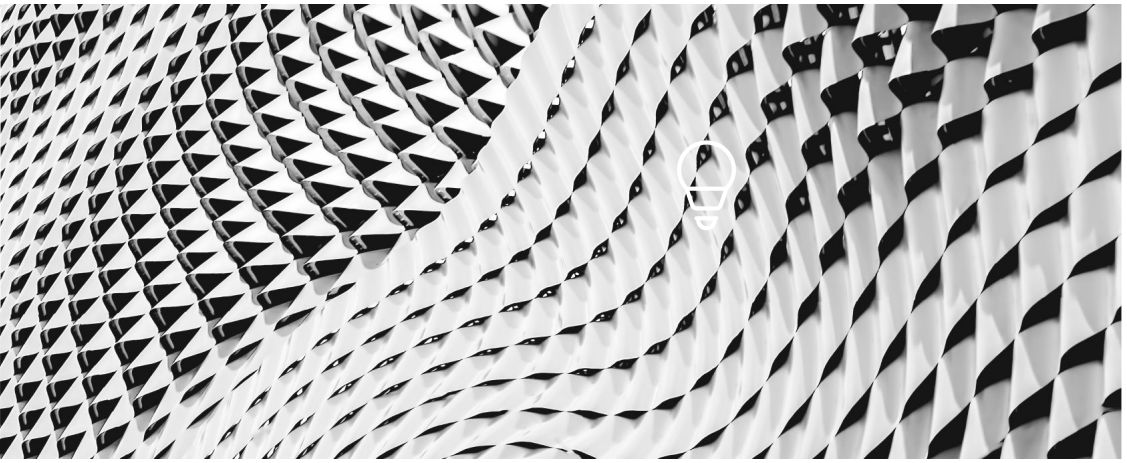
What's Knative ?

Open-Source Enterprise-level solution to build **Serverless** and **Event Driven Applications** using respectively two independent components **Serving** and **Eventing**

- Reached GA on Nov. 2, 2021
- CNCF incubating project since March. 2, 2022
- Originally came from Google with now many major contributors (Red Hat, IBM, Cisco, VMware...)
- Powers cloud commercial offering (Google Cloud Run, IBM Cloud Code Engine, Scaleway Serverless...)
- Runs on any Kubernetes platform (available on OpenShift with Red Hat OpenShift Serverless)

Source:
https://github.com/knative/community/blob/main/ADOPTERS.MD

Knative Serving



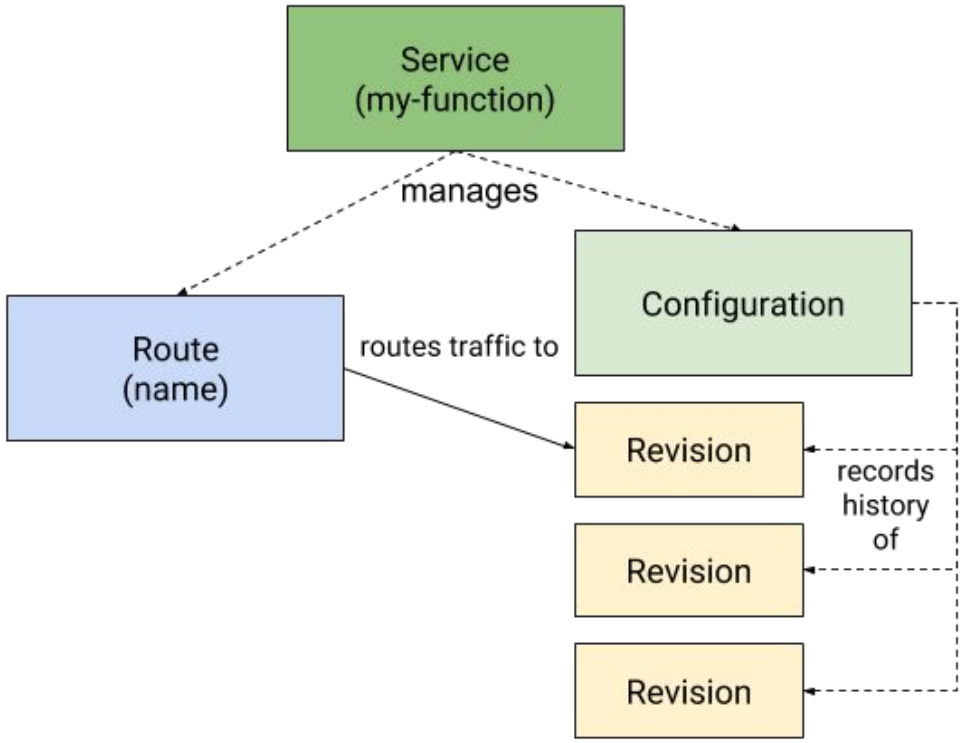
Knative Serving

Component responsible for running Serverless containers and functions.

- Provides a set of CRDs that **allows to quickly and securely run serverless (any http app) applications**
- Enables microservice architecture
- Supports any language
- Leverages Service Mesh for Networking/Ingress (Kourier, Istio...)

Source:
https://knative.dev/docs/serving/

Main CustomResourceDefinitions (CRDs)



Service

Top-level abstraction of a software application deployment

Route

Manage traffic to a Service container

Configuration

Stream of environments for Revisions

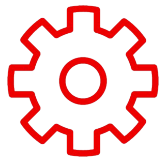
Revision

Immutable snapshots of configuration

Source:
lhttps://knative.dev/docs/serving/

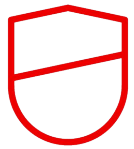
```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: hello
spec:
  template:
    spec:
      containers:
      - env:
        - name: TARGET
          value: World
        image: gcr.io/knative-samples/helloworld-go
        ports:
        - containerPort: 8080
```

Features



Autoscaling

Auto scale from zero to n containers, supports different scaling metrics (concurrency, rps, cpu), highly customizable scaling configuration.



Security

Auto TLS, mutual TLS, Traffic isolation, automatic digest resolution

Traffic Management

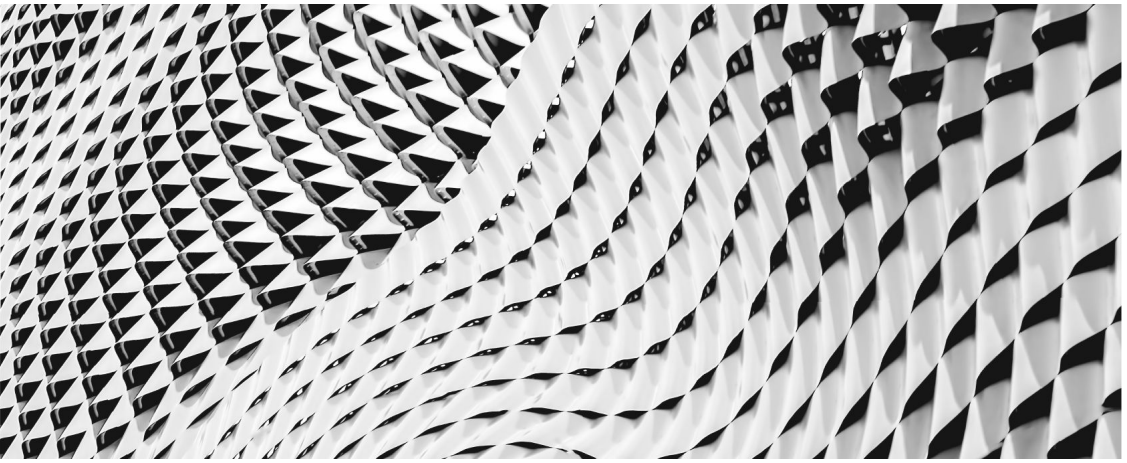
Route traffic between different version of a Knative Service.

Observability

Exposes Prometheus metrics and OpenTelemetry data.


STOP TALKING
SHOW ME A DEMO
memo-arsenal.ru

Knative Eventing



Knative Eventing

Component responsible for routing events from producers to consumers

- Provides a set of CRDs that enables event-driven architecture
- Event producer (event sources) can be internal (PingSource, ApiServerSource…) or external (GitHub, Kafka, AWS SQS…) to the cluster
- Event consumer (sink) can be any Kubernetes or Knative Services
- Allows different architectures to deliver events

Source:
https://knative.dev/docs/eventing
https://knative.dev/docs/eventing/sources/#knative-sources
https://knative.dev/docs/eventing/sinks/

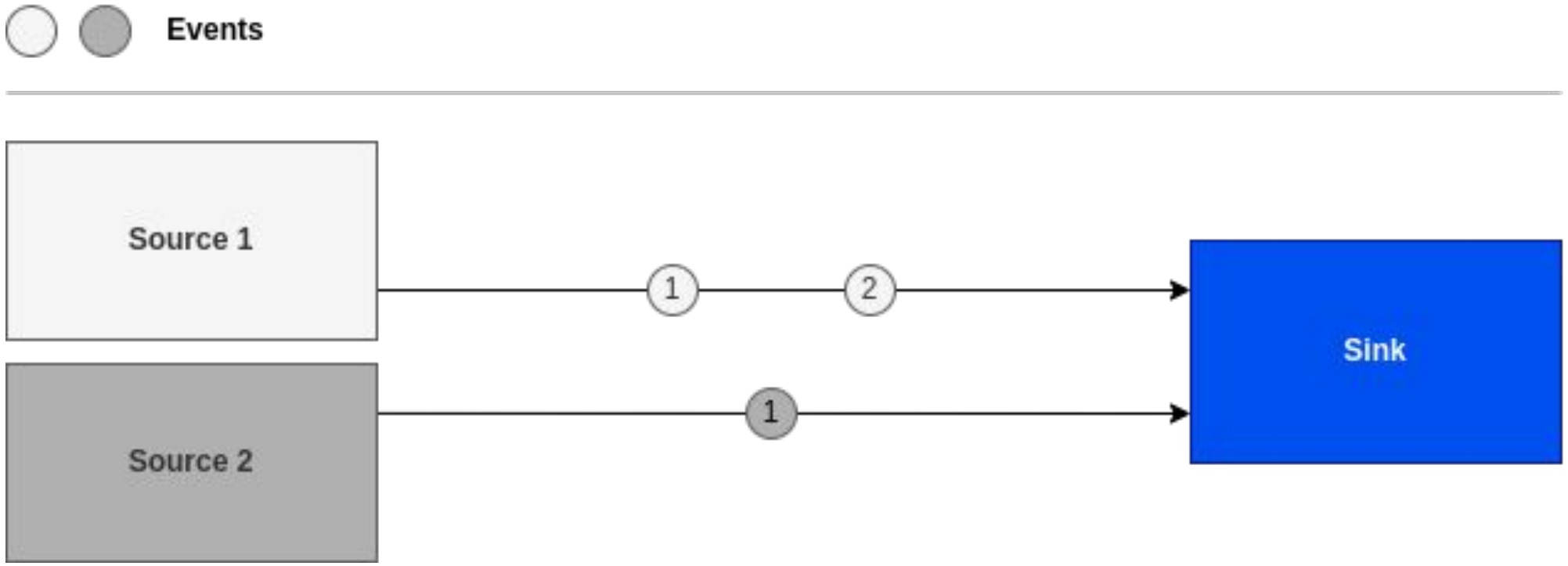
CloudEvents (cloudevents.io)

- Events are standard, they follow the CloudEvents specifications
- Sending event is standard, they are sent with HTTP POST

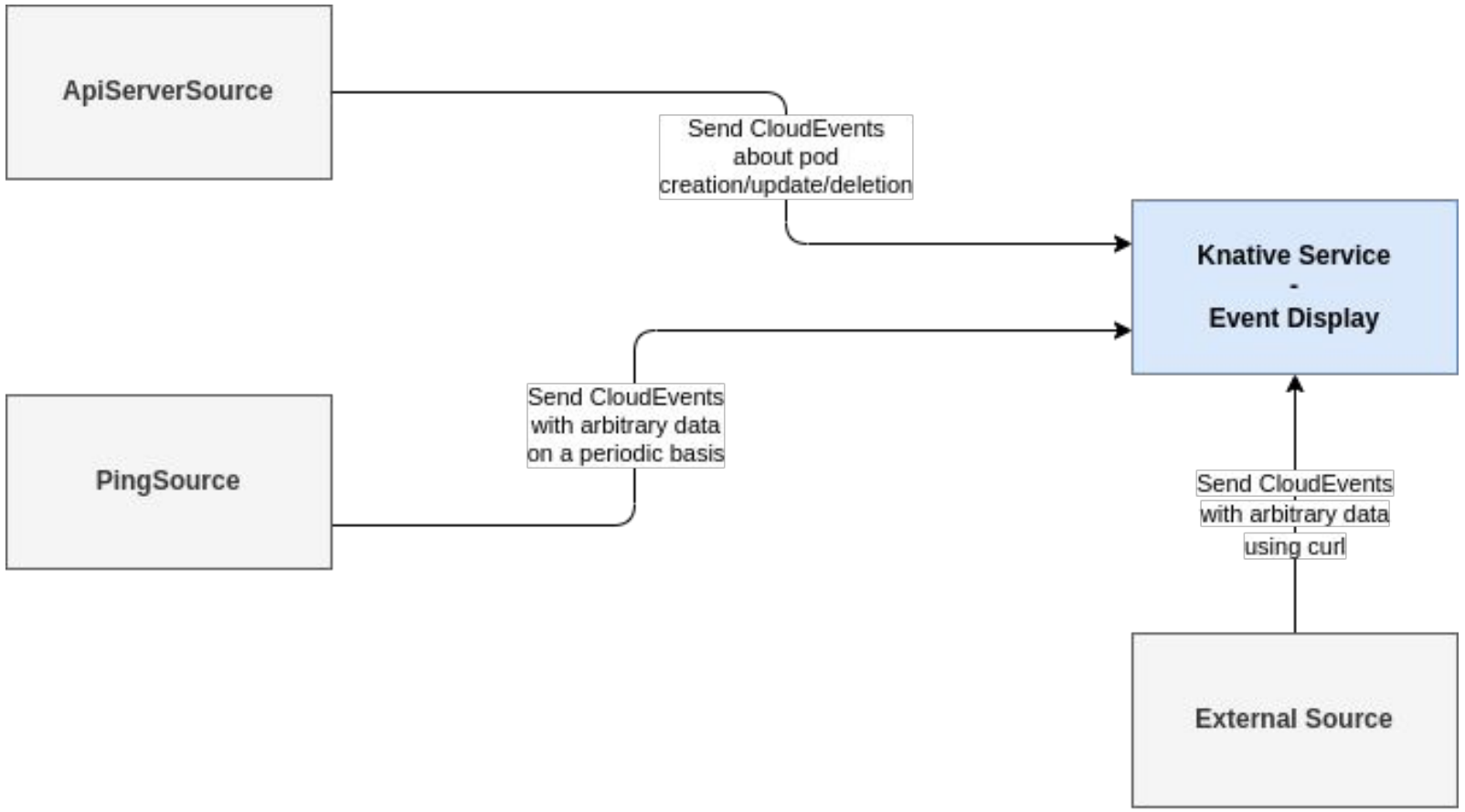
```
{
    "specversion": "1.0",
    "type": "falco.rule.output.v1",
    "source": "syscall",
    "id": "b179bf43-ef62-4808-81e9-058d2cd0edcd",
    "time": "2022-09-09T10:02:40.435537425Z",
    "datacontenttype": "application/json",
    "data": "\{\"output\": \"Notice Unexpected connection to K8s API Server from container\"\}"
}
```


Source:
https://github.com/cloudevents/spec/blob/v1.0.2/cloudevents/spec.md

Source to Sink (n:1)

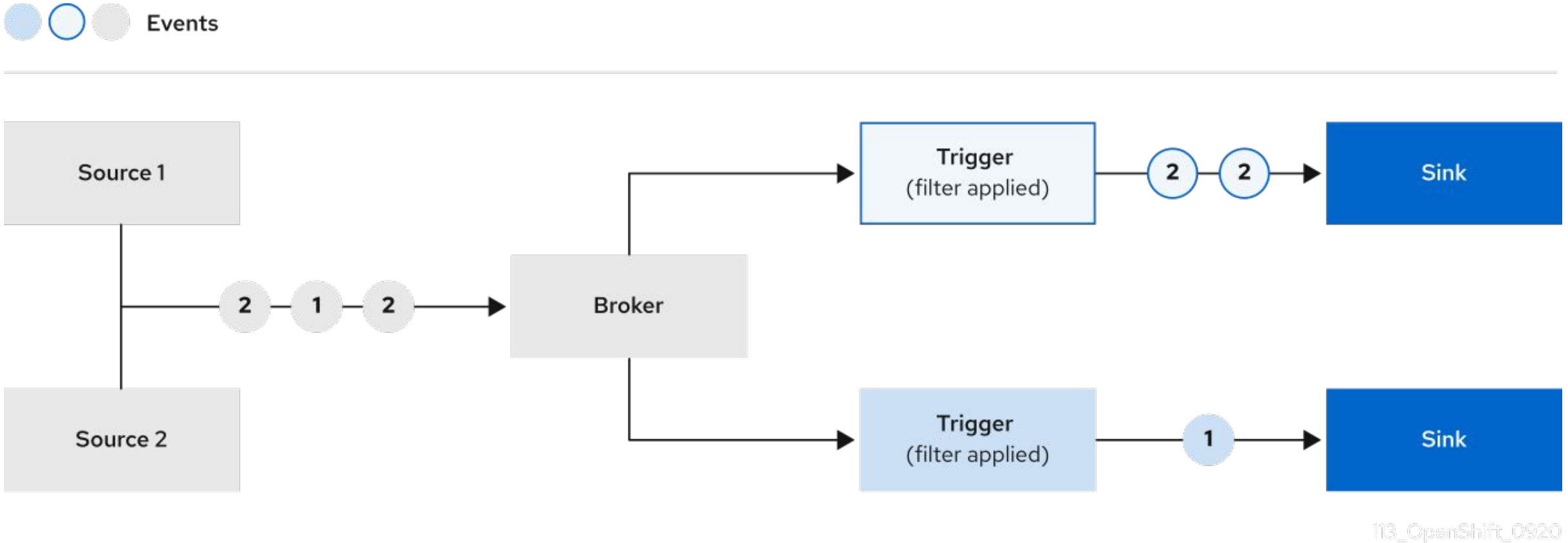


Demo



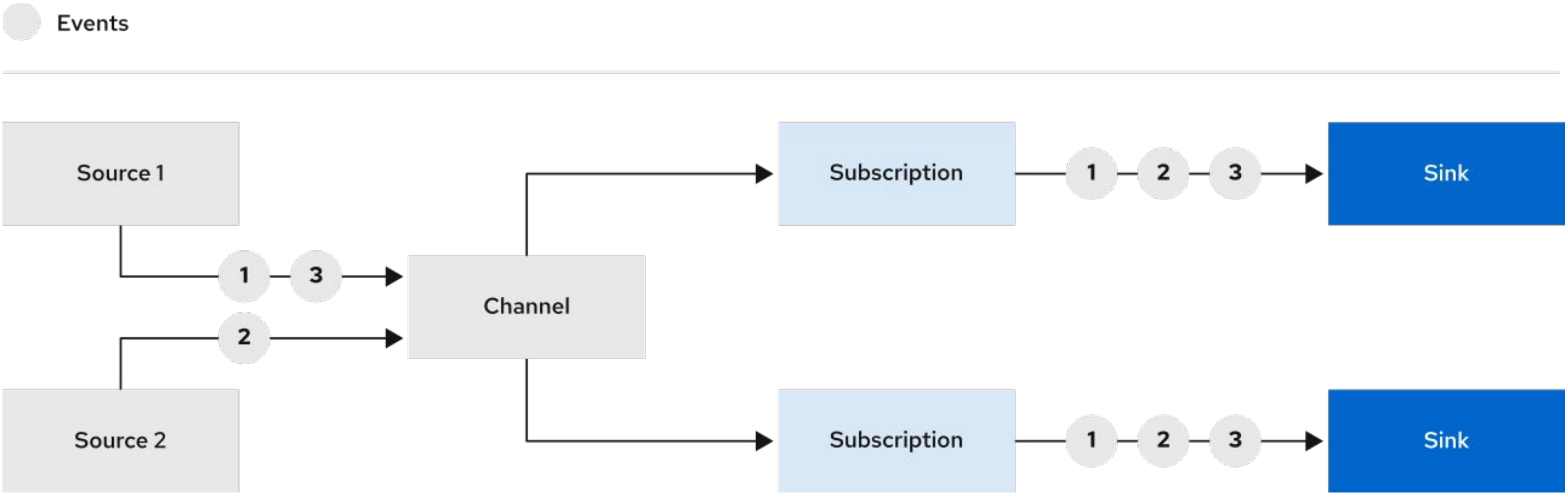
The **ApiServerSource** needs RBAC to be configured in order to read pods information in the namespace or at the cluster level.

Broker and Trigger (n:n)



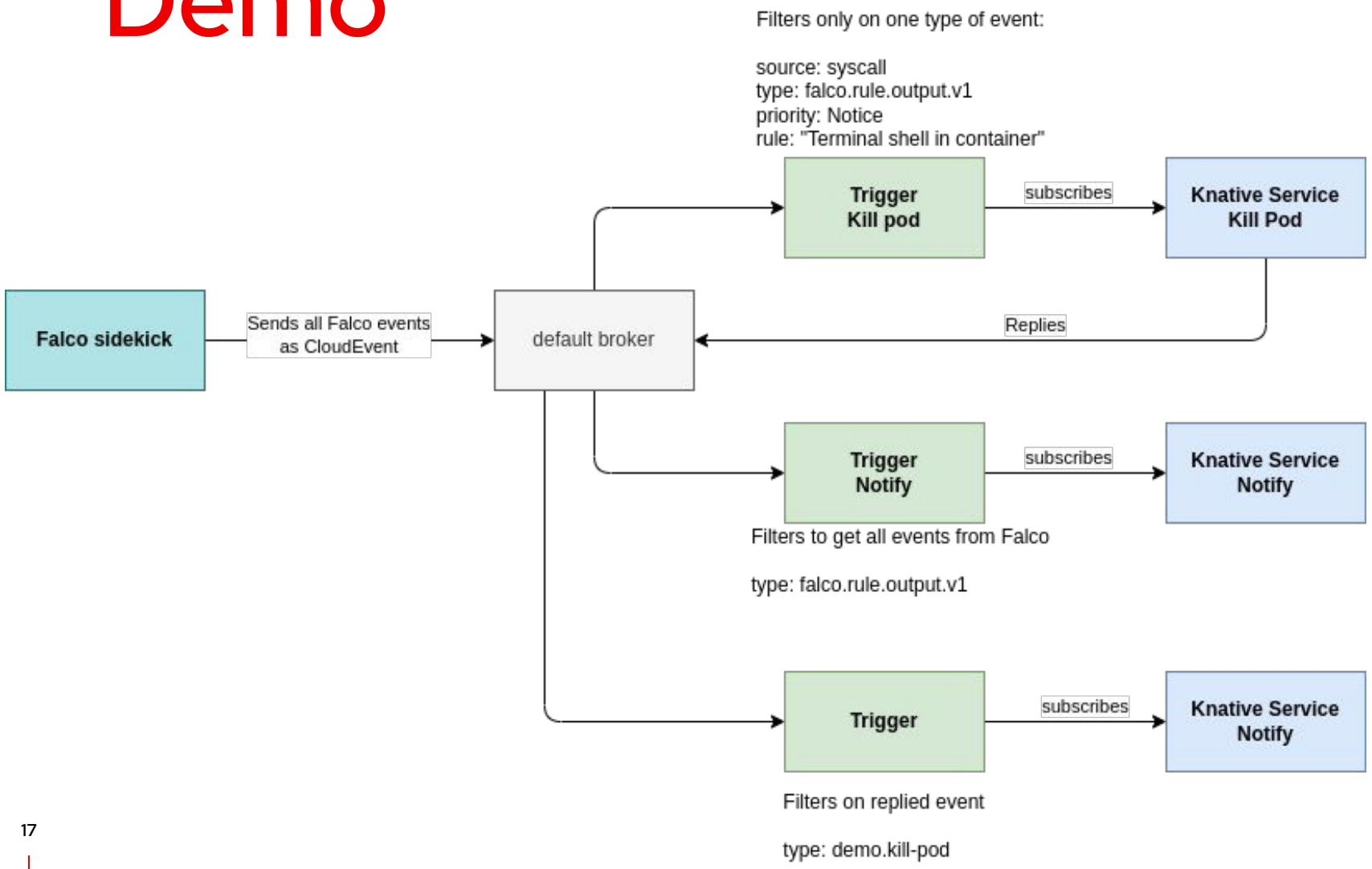
Source:
https://knative.dev/docs/eventing/brokers/

Channel and Subscription (n:n)



Source:
https://knative.dev/docs/eventing/channels/

Demo


Filters only on one type of event:
source: syscall
type: falco.rule.output.v1
priority: Notice
rule: "Terminal shell in container"
Trigger
Kill pod
subscribes
Knative Service
Kill Pod
Falco sidekick
Sends all Falco events
as CloudEvent
default broker
Replies
Trigger
Notify
subscribes
Knative Service
Notify
Filters to get all events from Falco
type: falco.rule.output.v1
Trigger
subscribes
Knative Service
Notify
Filters on replied event
type: demo.kill-pod

Other topics

Custom event sources

Create new sources from producers that does not emit CloudEvents or that is simply not included by default in Knative

Handling Delivery Failure

Configure where to send event when delivery failed

Flows

Define order of service invocation using Sequence or Parallel

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat