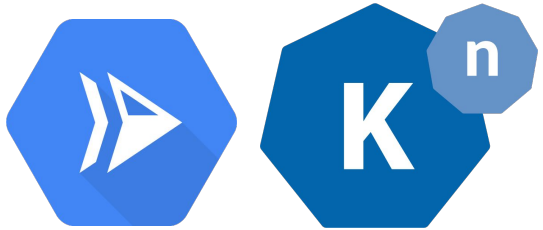
Knative and Cloud Run

Nick Brandaleone
Google CE
@brandaleone

June 2021



Serverless

Operational Model

No Infra Management

Managed Security

Pay only for usage

Programming Model

Service-based

Event-driven

Stateless

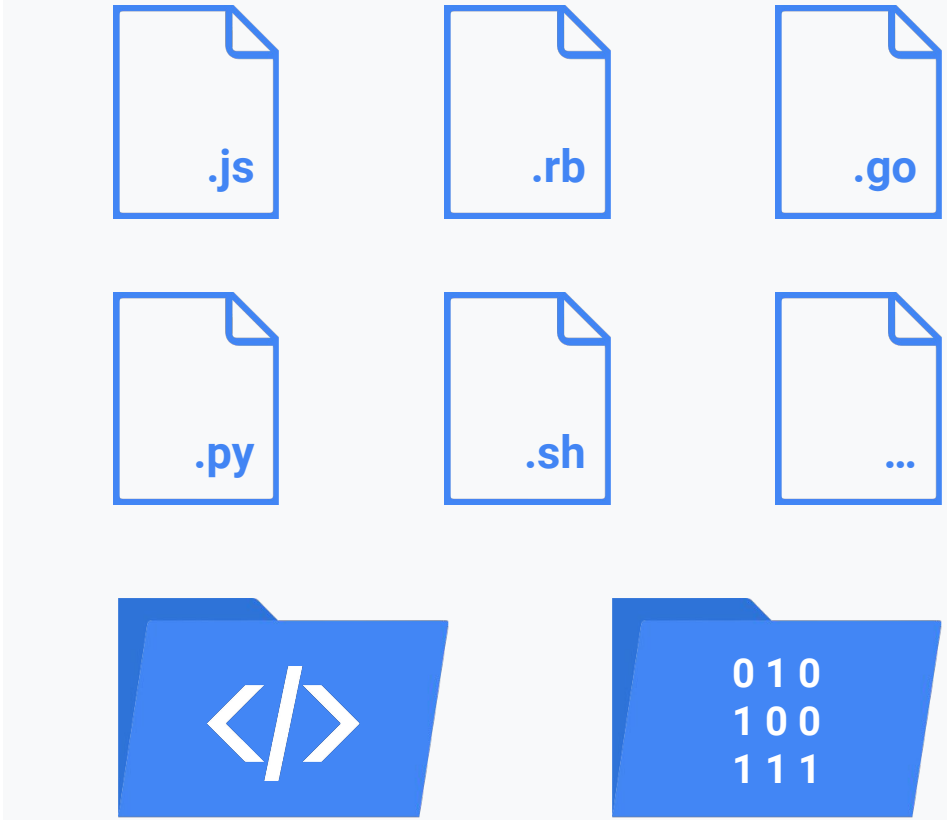
Google Cloud

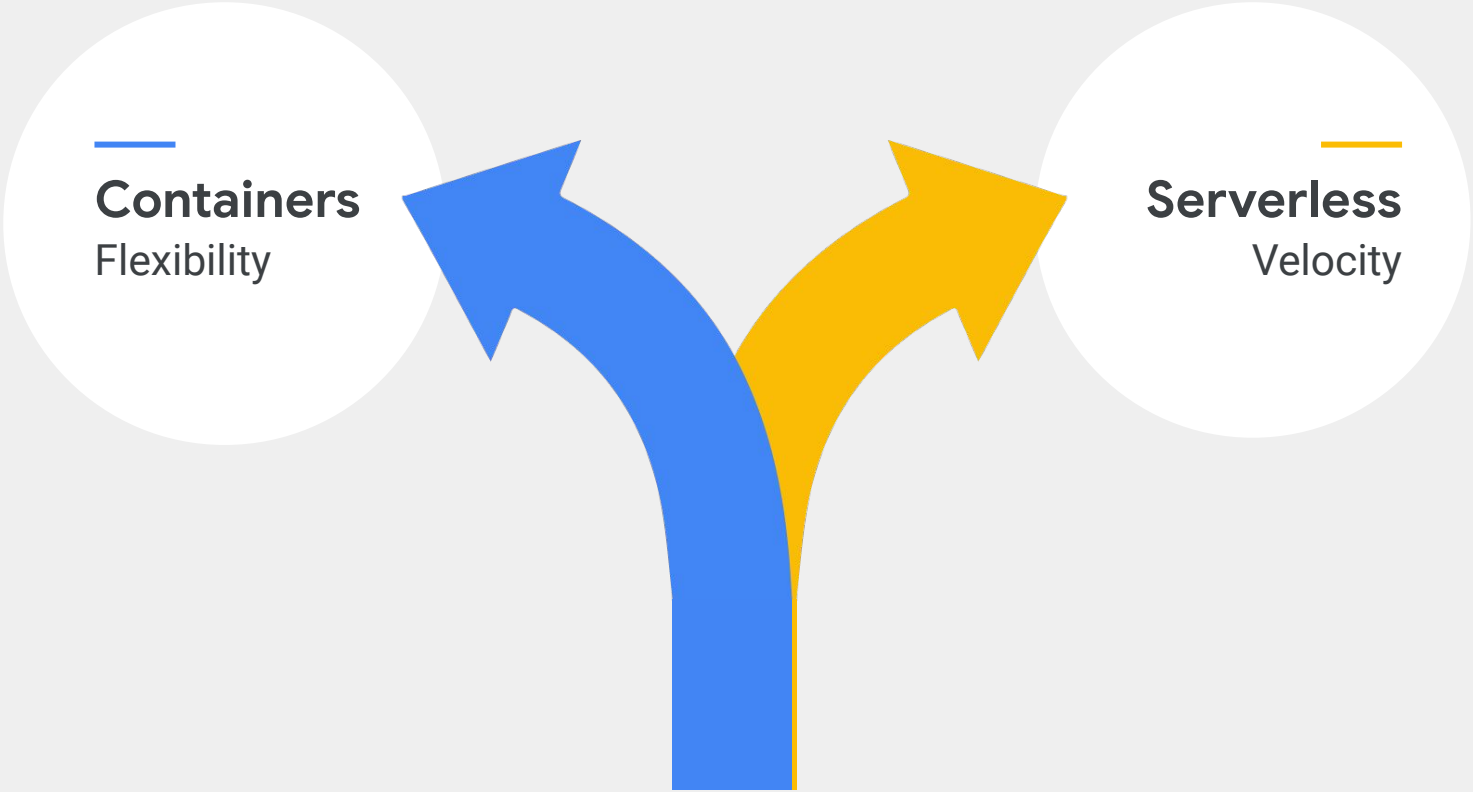
Containers

Any language

Any library

Ecosystem around containers

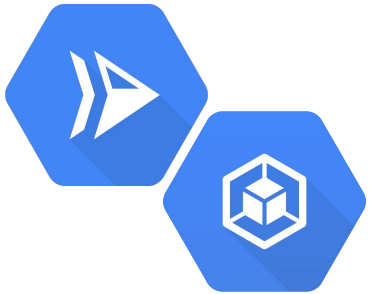



Containers
Flexibility
Serverless
Velocity

Serverless containers with Knative and Cloud Run

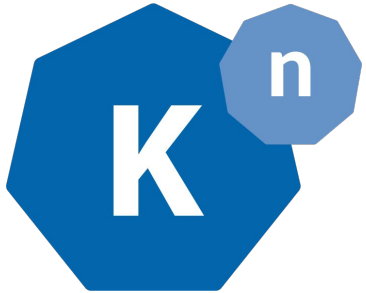
Cloud Run

Fully managed, deploy your workloads and don't see the cluster.



Cloud Run on Anthos

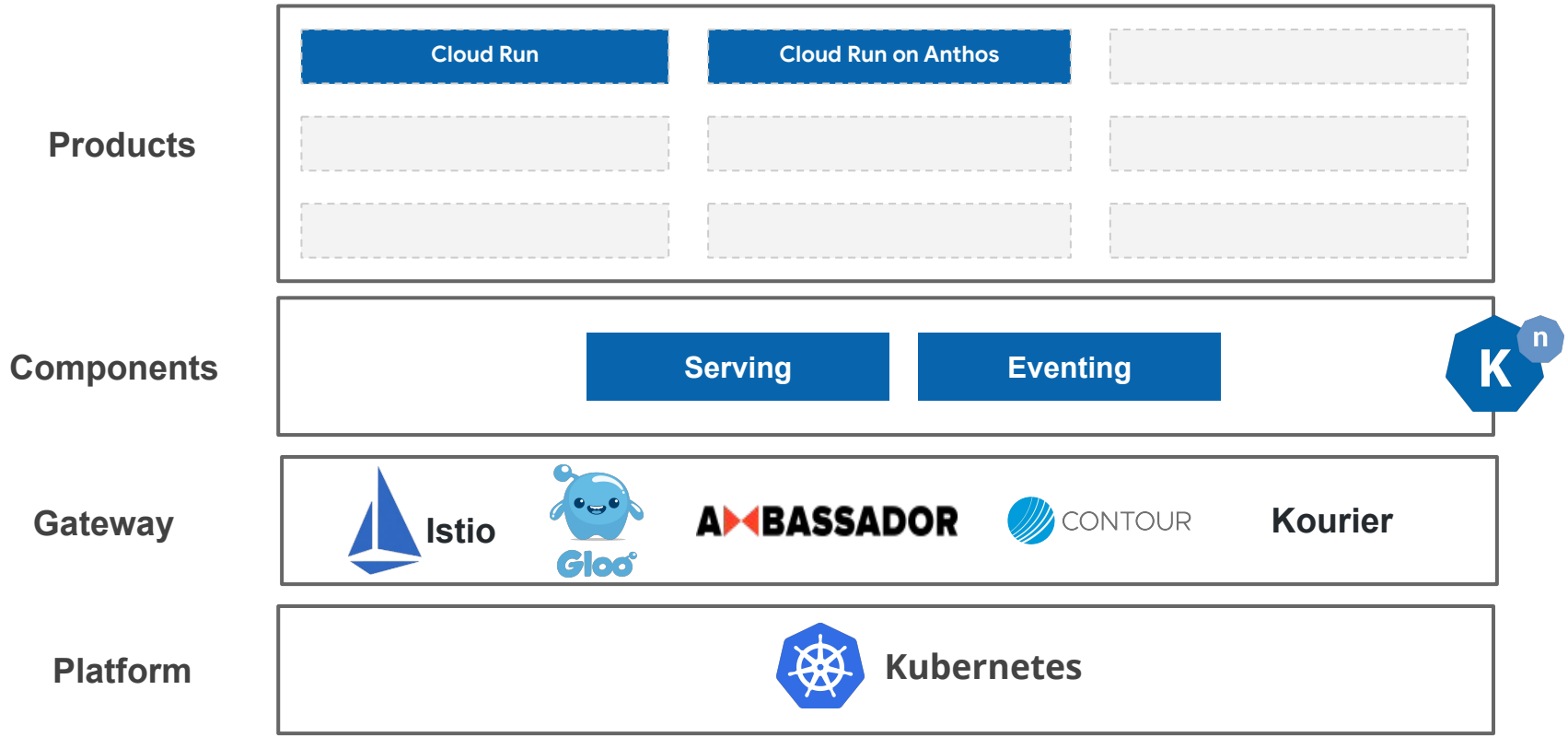
Deploy into Anthos, run serverless side-by-side with your existing workloads.



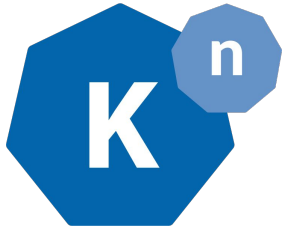
Knative Everywhere

Use the same APIs and tooling anywhere you run Kubernetes with Knative.

Knative Stack



Knative Serving



What is it?

Rapid deployment of serverless containers

Automatic (0-n) scaling

Configuration and revision management

Traffic splitting between revisions

Pluggable

Connect to your own logging and monitoring platform, or use the built-in system

Auto-scaler can be tuned or swapped out for custom code

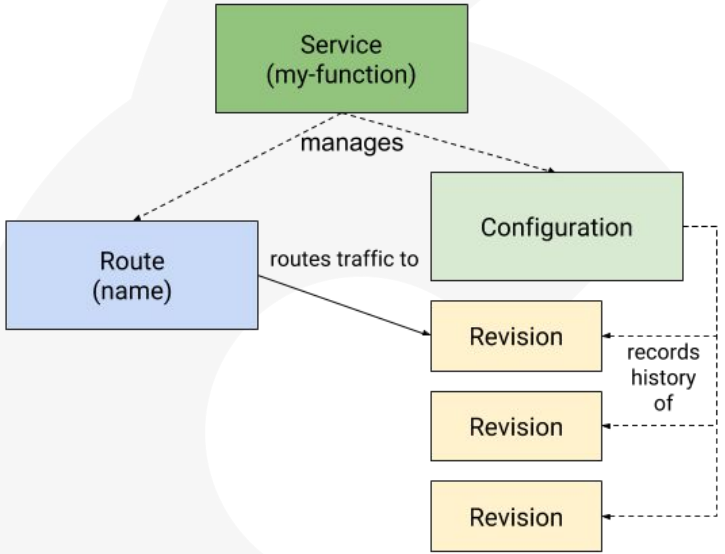
Knative Serving

Knative Service
High level abstraction for the application

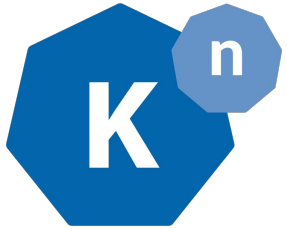
Configuration
Current/desired state of an application
Code & configuration separated (a la 12-factor)

Revision
Point in time snapshots for your code and configuration

Route
Maps traffic to revisions



Knative Eventing



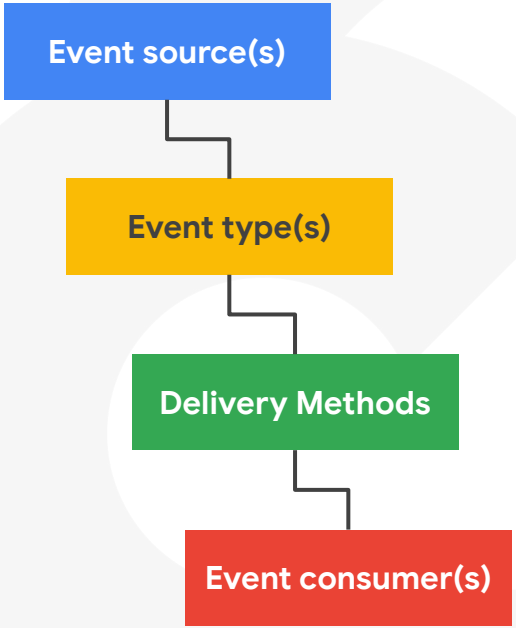
What is it?

For loosely coupled, event-driven services

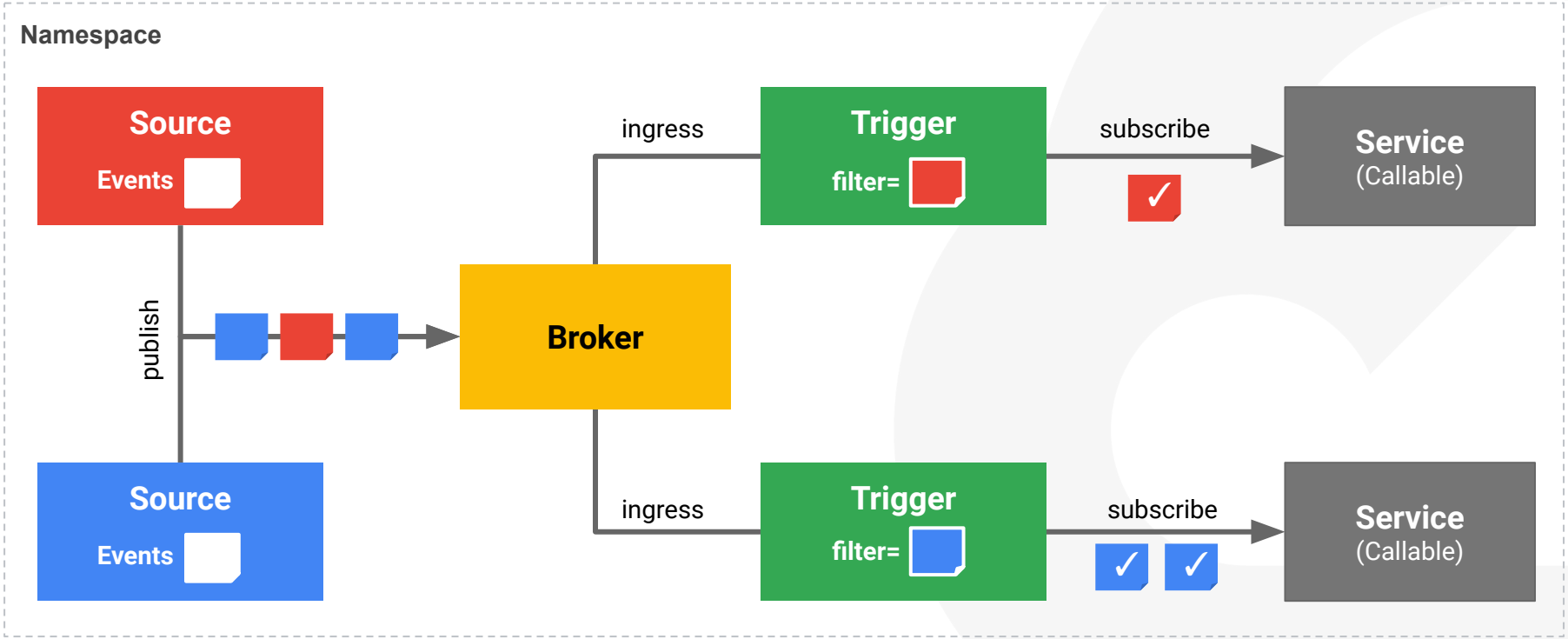
A number of different delivery methods

Scales from just few events to live streams

Uses standard CloudEvents



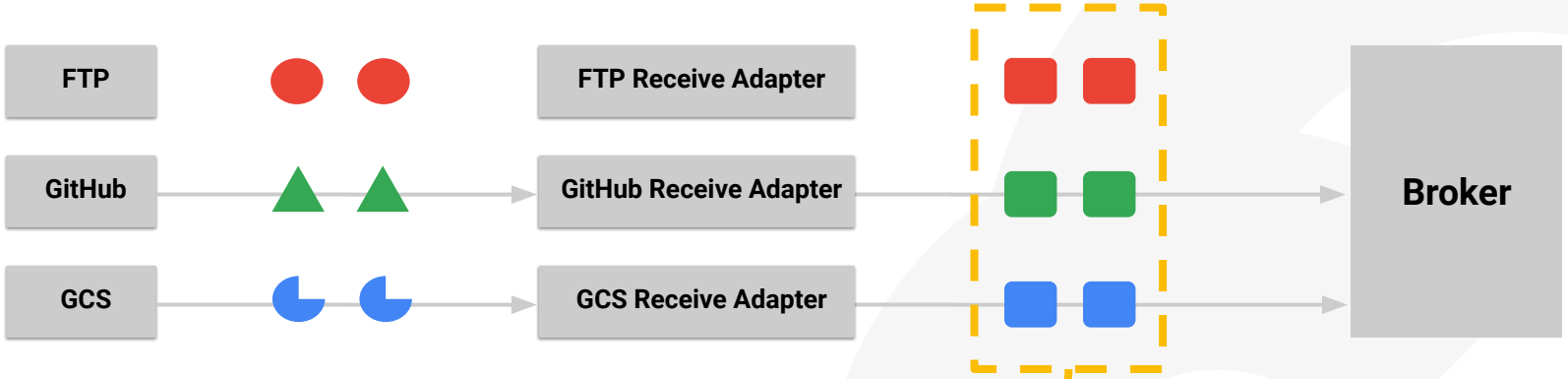
Knative Eventing



Terminology of Knative Eventing

CloudEvents → Format
Event Source → Producer
Broker → Event mesh in the namespace
Trigger → Interest in messages from Broker & filter
Service → Consumer
Channel → Persistence layer
Subscription → Interest in messages from channel

CloudEvents - cloudevents.io



CloudEvent

```
{
    "specversion": "1.0",
    "type": "com.github.pull.create",
    "source": "https://github.com/cloudevents/spec/pull/123",
    "id": "A234-1234-1234",
    "time": "2019-04-08T17:31:00Z",
    "datacontenttype": "application/json",
    "data": "{ GitHub Payload... }"
}
```

Event Sources

Name	Description
Apache Camel	Allows to use Apache Camel components for pushing events into Knative
Apache Kafka	Brings Apache Kafka messages into Knative
AWS SQS	Brings AWS Simple Queue Service messages into Knative
Cron Job	Uses an in-memory timer to produce events on the specified Cron schedule.
GCP PubSub	Brings GCP PubSub messages into Knative
GitHub	Brings GitHub organization/repository events into Knative
GitLab	Brings GitLab repository events into Knative.
Google Cloud Scheduler	Google Cloud Scheduler events in Knative when jobs are triggered
Google Cloud Storage	Brings Google Cloud Storage bucket/object events into Knative
Kubernetes	Brings Kubernetes cluster/infrastructure events into Knative

https://github.com/knative/docs/tree/master/docs/eventing/sources

Broker

Combines Channel, reply, and filter functionality into a single resource

Typically injected one per namespace

Broker

```
apiVersion: eventing.knative.dev/v1beta1
  kind: Broker
    labels:
      eventing.knative.dev/namespaceInjected: "true"
    name: default
    namespace: default
  status:
    address:
      Url:http://default-broker.default.svc.cluster.local
```

Trigger

Subscribes a Service to Broker

Filtering

Trigger

```
apiVersion: eventing.knative.dev/v1beta1
kind: Trigger
metadata:
  name: trigger-filter
spec:
  filter:
    attributes:
      type: com.google.cloud.storage.object.finalize
  subscriber:
    ref:
      apiVersion: serving.knative.dev/v1
      kind: Service
      name: filter
```

Service

Receives events

Knative or Kubernetes Service

Subscriber of a trigger or a sink of a source

Knative Service

```
apiVersion: eventing.knative.dev/v1alpha1
kind: Trigger
metadata:
  name: trigger-event-display
spec:
  subscriber:
    ref:
      # apiVersion: v1
      apiVersion: serving.knative.dev/v1
      kind: Service
      name: event-display
```

Channel

Persistence layer

In-memory, PubSub, Kafka implementations

Default channel

Channel

```
apiVersion: messaging.knative.dev/v1beta1
kind: InMemoryChannel
metadata:
  name: channel

apiVersion: messaging.knative.dev/v1alpha1
kind: KafkaChannel
metadata:
  name: my-kafka-channel
spec:
  numPartitions: 1
  replicationFactor: 1
```

Subscription

Subscribes Service to Channel

Also defines the notion of **event replies**

Subscription

```
apiVersion: messaging.knative.dev/v1alpha1
kind: Subscription
metadata:
  name: subscription1
spec:
  channel:
    apiVersion: messaging.knative.dev/v1alpha1
    kind: InMemoryChannel
    name: channel
  subscriber:
    ref:
      apiVersion: serving.knative.dev/v1
      kind: Service
      name: service1
```

Delivery Methods

Simple Delivery

Event Source ⟶ Service, 1:1

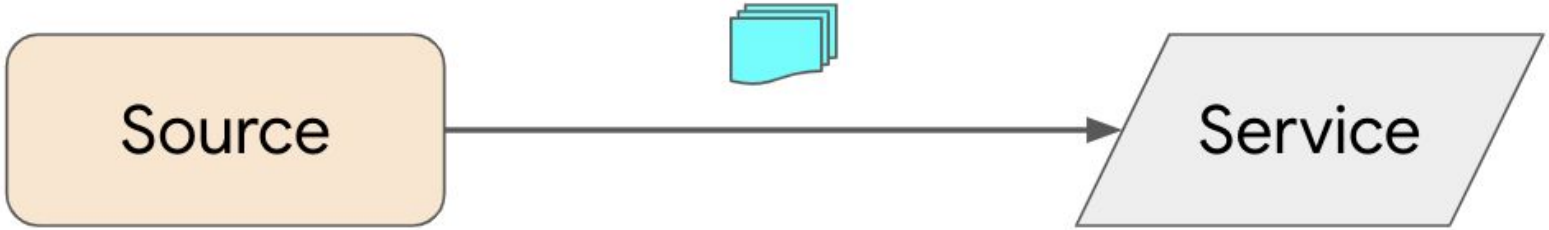
Complex Delivery with optional reply

Event Source ⟶ Channels ⟶ Subscription ⟶ Services, 1:N

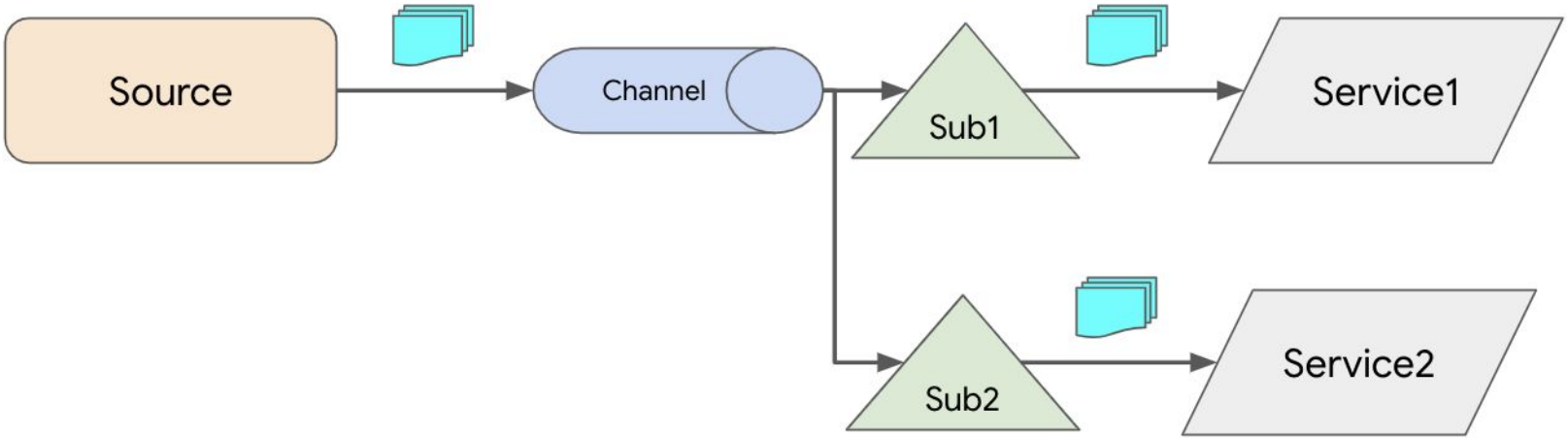
Broker Trigger Delivery

Event Source ⟶ Broker ⟶ Triggeer ⟶ Services, 1:N

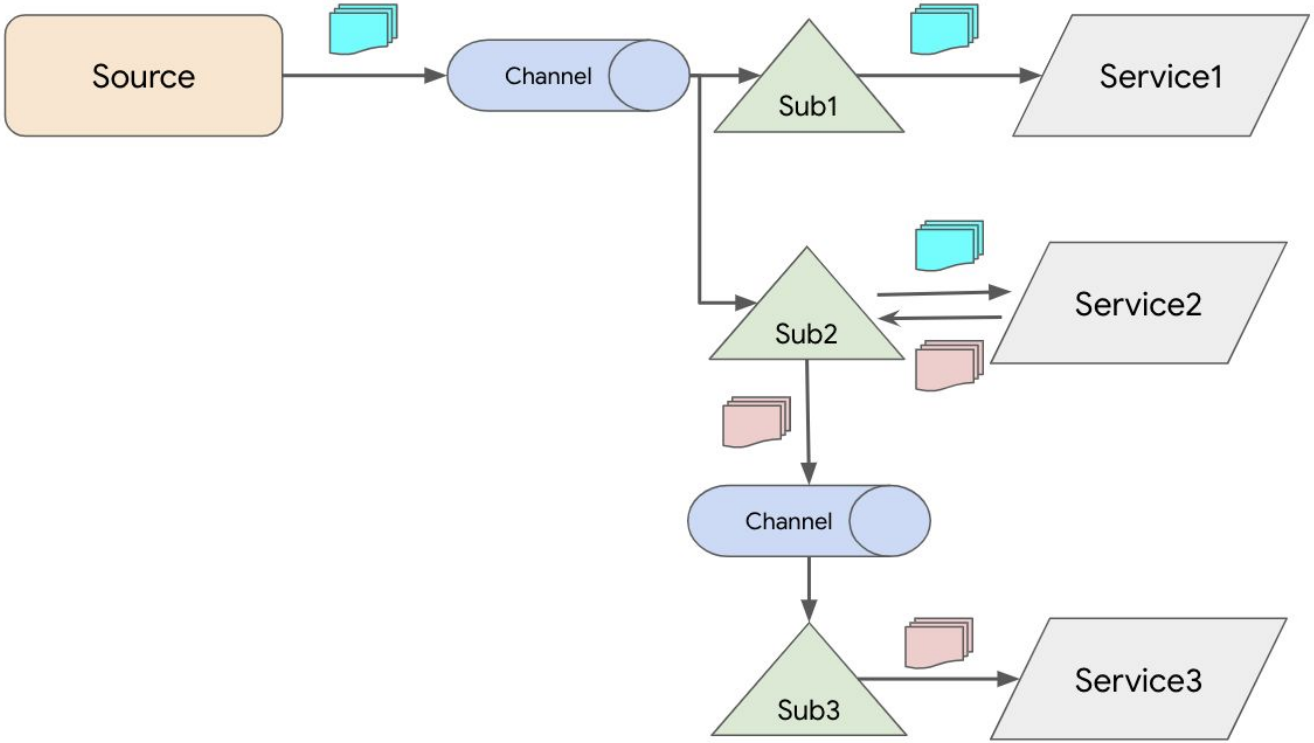
Simple Delivery



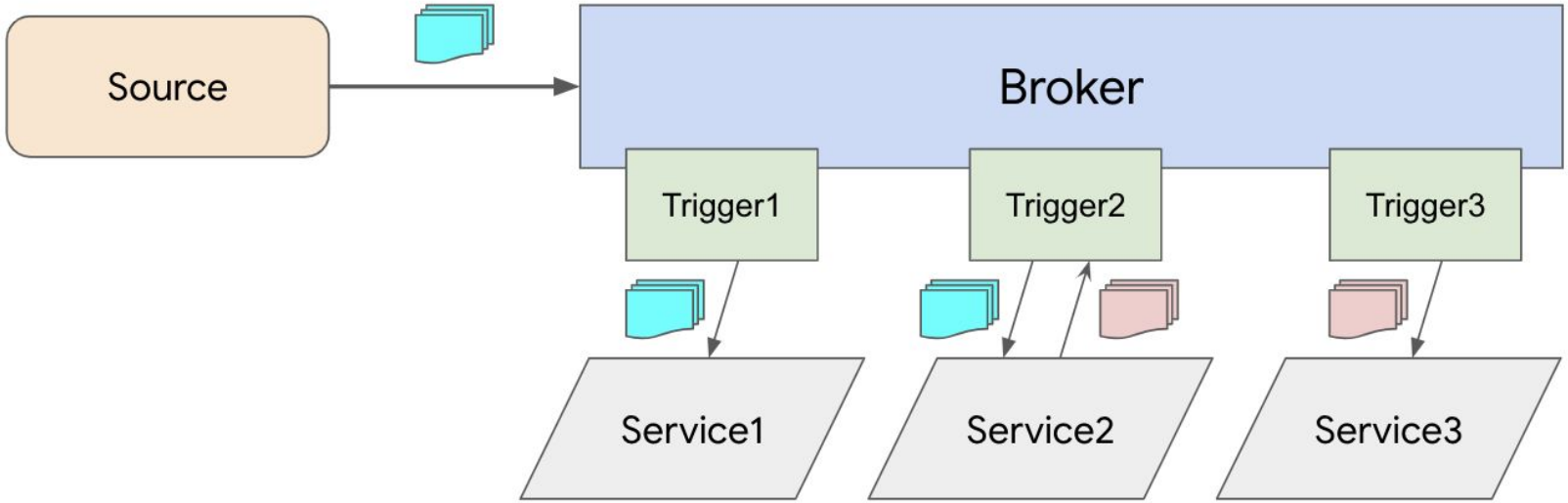
Complex Delivery



Complex Delivery with reply



Broker Trigger Delivery



Knative GCP Project - github.com/google/knative-gcp

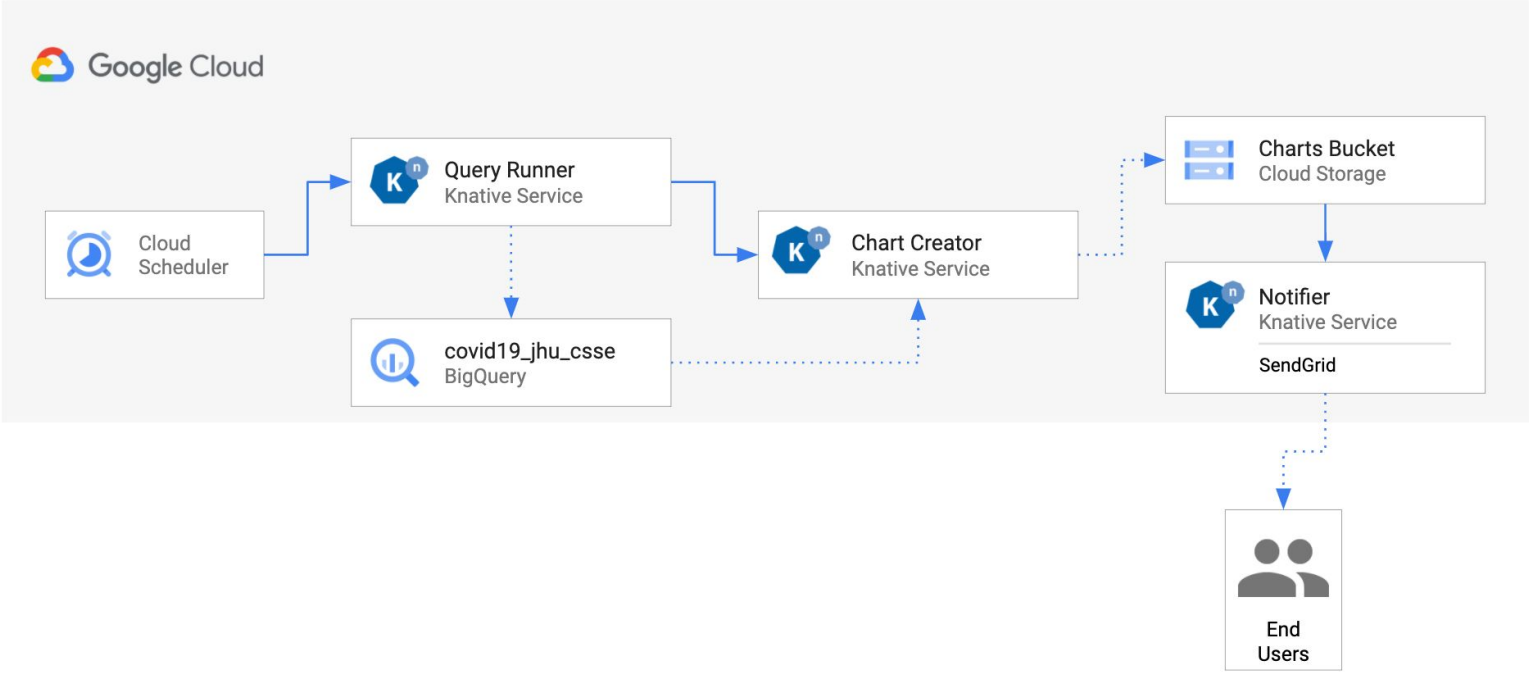
Easy configuration and consumption of Google Cloud Platform events in Knative

Ready to use event sources:

1. CloudPubSubSource
2. CloudStorageSource
3. CloudSchedulerSource
4. CloudAuditLogsSource
5. CloudBuildSource

BigQuery Processing Pipeline

Schedule BigQuery jobs to query Covid-19 public dataset and generate charts to share with users over email



Thank you!

https://github.com/nbrandaleone/knative-tutorial