Knative

-going server less

Gang of Two (Suchitra Reddy Chinnamail and Varun Lingabathini)

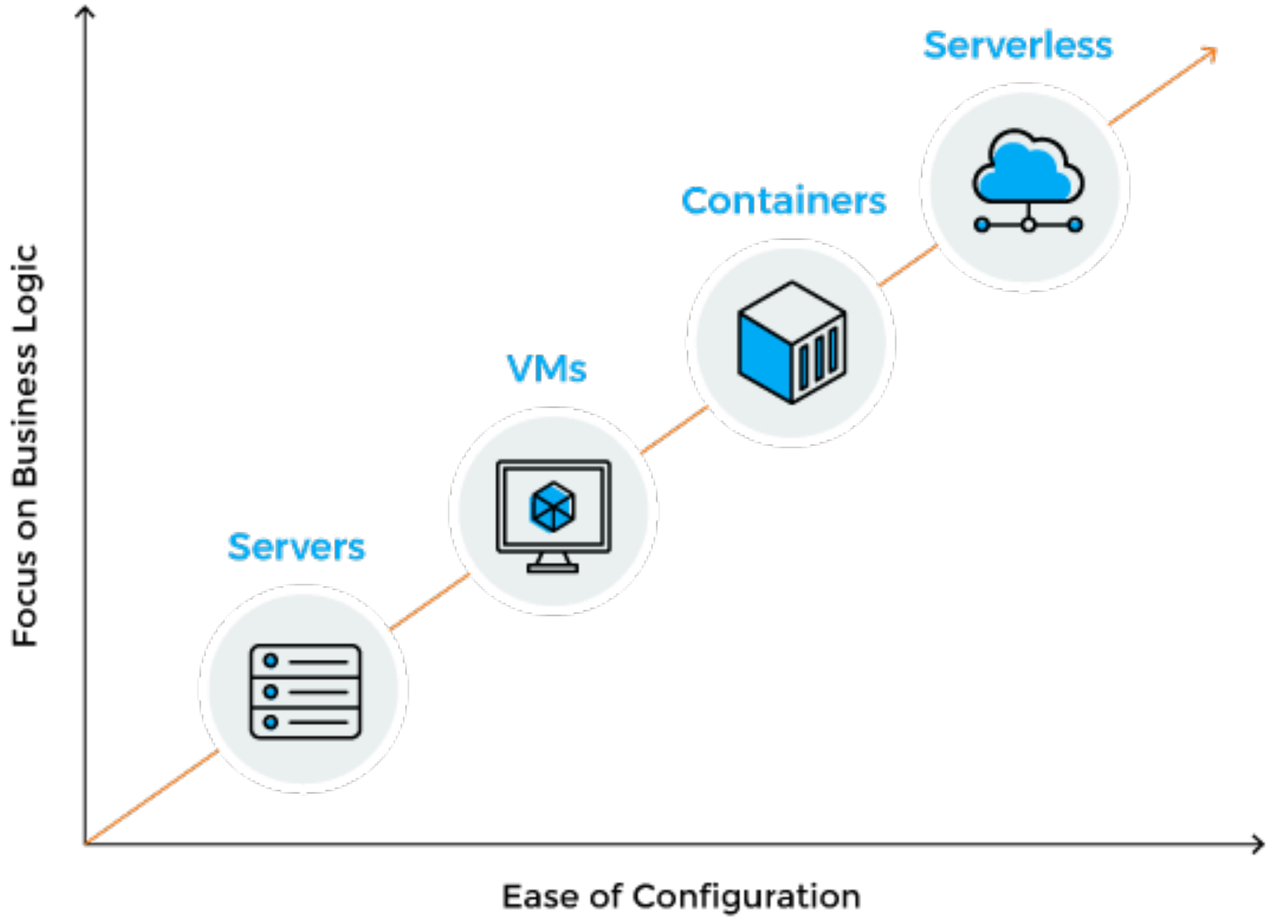
Agenda

- Serverless - Introduction
- Knative -Introduction
- Knative Components
  - Knative Serving
  - Knative Eventing
- Autoscaling

Serverless

Serverless

- Write small, single purpose functions that are invoked via events.
- Use compute resources only while serving requests.
- Focus on writing code itself - faster time to market.



Knative

Knative - general

Goal: Bring serverless capabilities and running serverless workloads to Kubernetes

- Platform on top of Kubernetes
- Kubernetes under the covers
- Access to full Kubernetes if needed

Installed as set of Custom Resource Definitions (CRDs) for Kubernetes

- Main components:
  - Serving:
    - Run serverless containers on Kubernetes
    - Takes care of the details of networking, autoscaling (even to zero), and revision tracking
  - Eventing
    - Universal subscription, delivery, and management of events
    - Build apps by attaching compute to a data stream with declarative event connectivity and developer-friendly object model

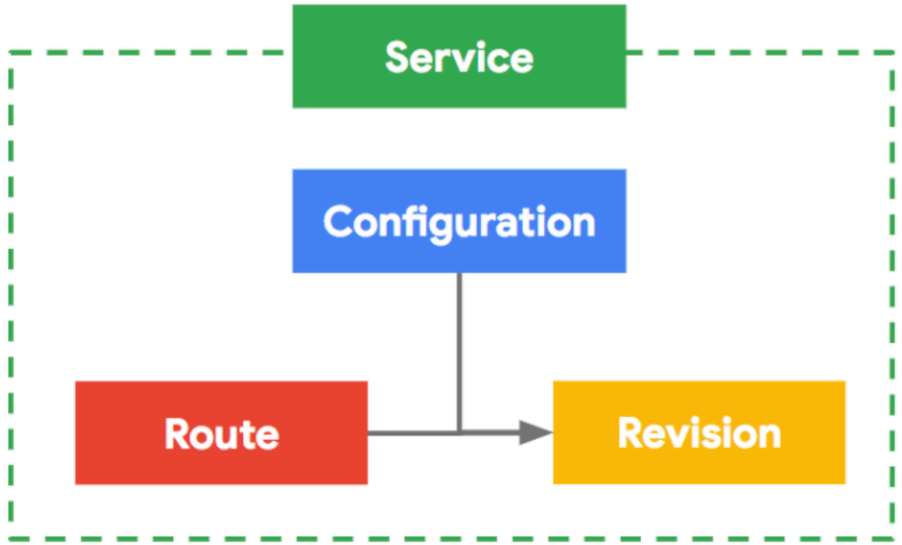
Knative Components

Knative Serving

Knative Serving builds on Kubernetes to support deploying and serving of serverless applications and functions.

Provided capabilities:

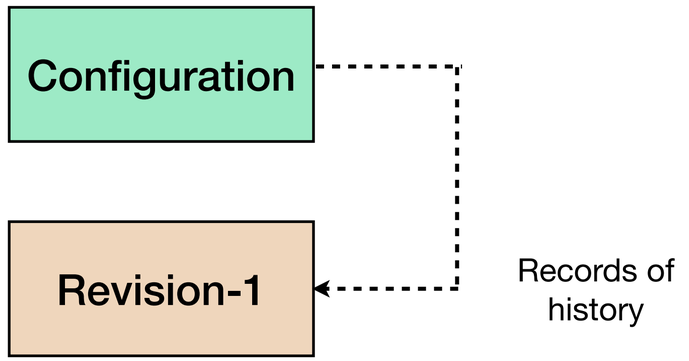
- Handle and respond to HTTP requests
- Manage deployment and serving
- Maintain point-in-time snapshots
- Provide automatic scaling (up and down to zero)
- Handles routing and network programming
- Objects to control this functionality: Configuration, Revision, Route and Service



Knative Serving Object Model - Configuration

Deploy app as pod/revision

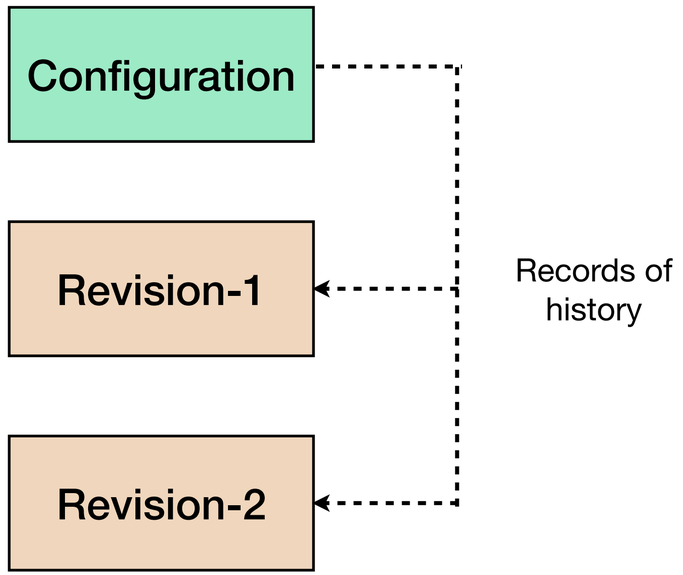
- Revision: immutable version of your app
  - Example image, environment variables, scale
- Revisions scaled up/down based on load



Knative Serving Object Model - Configuration

Deploy app as pod/revision

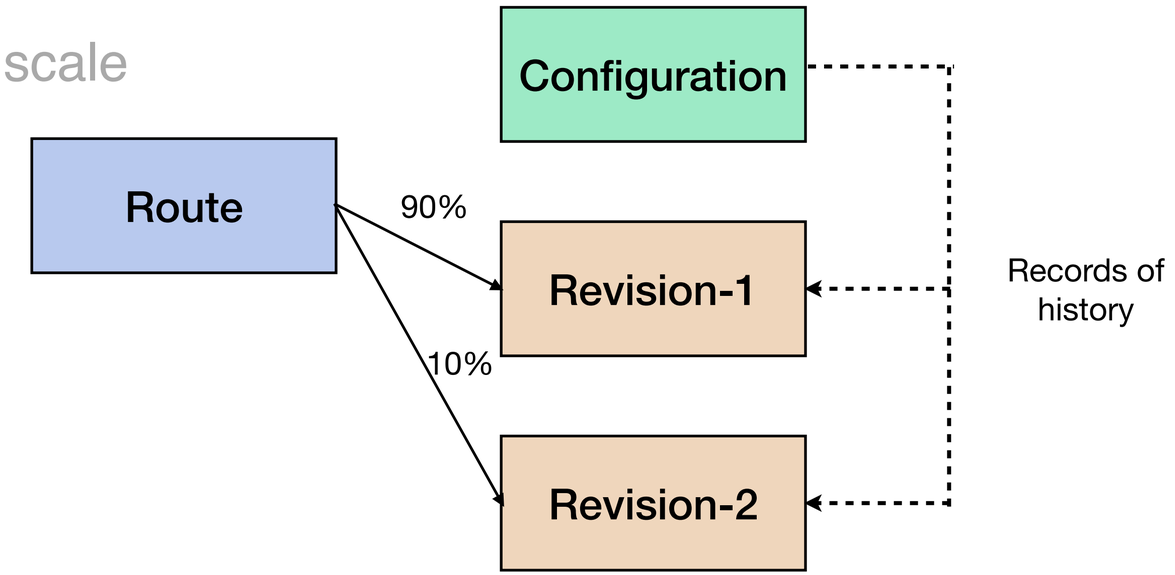
- Revision: immutable version of your app
  - Example image, environment variables, scale
- Revisions scaled up/down based on load
- Updates create revisions



Knative Serving Object Model - Configuration

Deploy app as pod/revision

- Revision: immutable version of your app
  - Example image, environment variables, scale
- Revisions scaled up/down based on load
- Updates create revisions
- Routes: manage traffic
  - Networking auto-setup
- Traffic splitting based on %

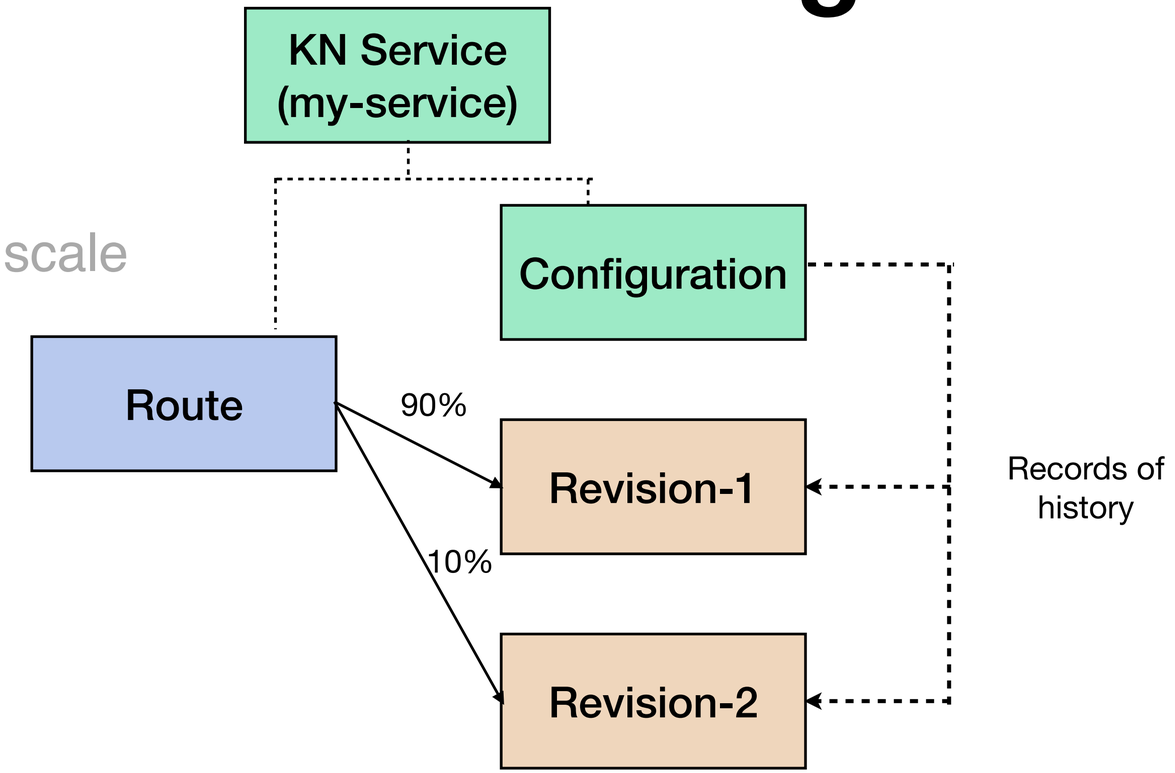


Knative Serving Object Model - Configuration

Deploy app as pod/revision

- Revision: immutable version of your app
  - Example image, environment variables, scale
- Revisions scaled up/down based on load
- Updates create revisions
- Routes: manage traffic
  - Networking auto-setup
- Traffic splitting based on %

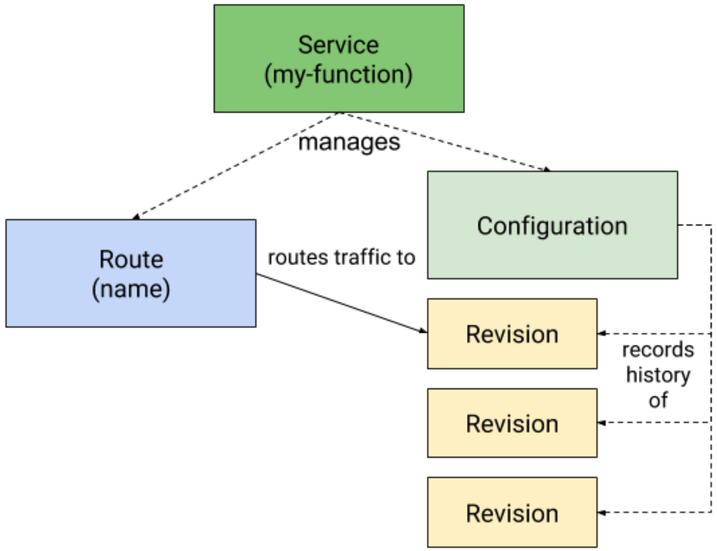
- KN Service: manages life cycle of a workload



Demo

Knative Serving Summary

- Deploy KN service with name and image
- Access via HTTP
- Rolling upgrade to new version of service
- Routing traffic to different revisions

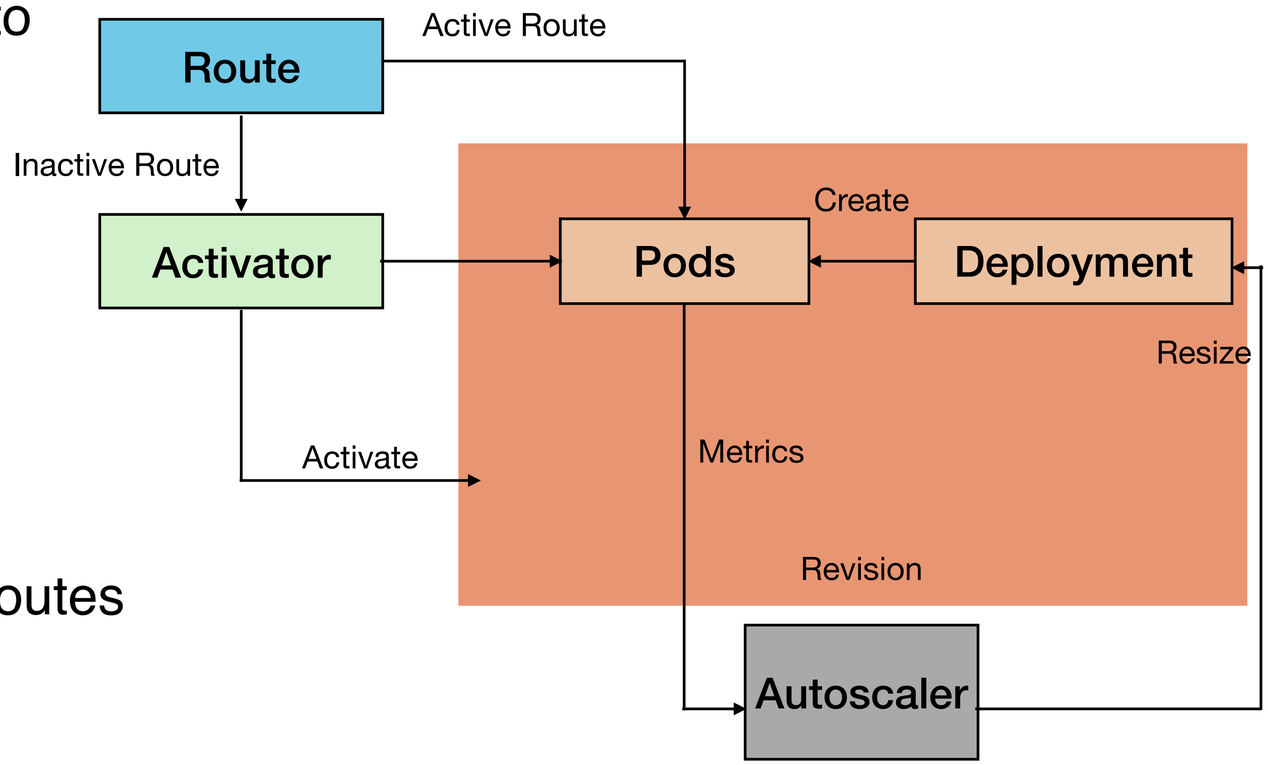


Autoscaling

Autoscaling

Autoscaler:

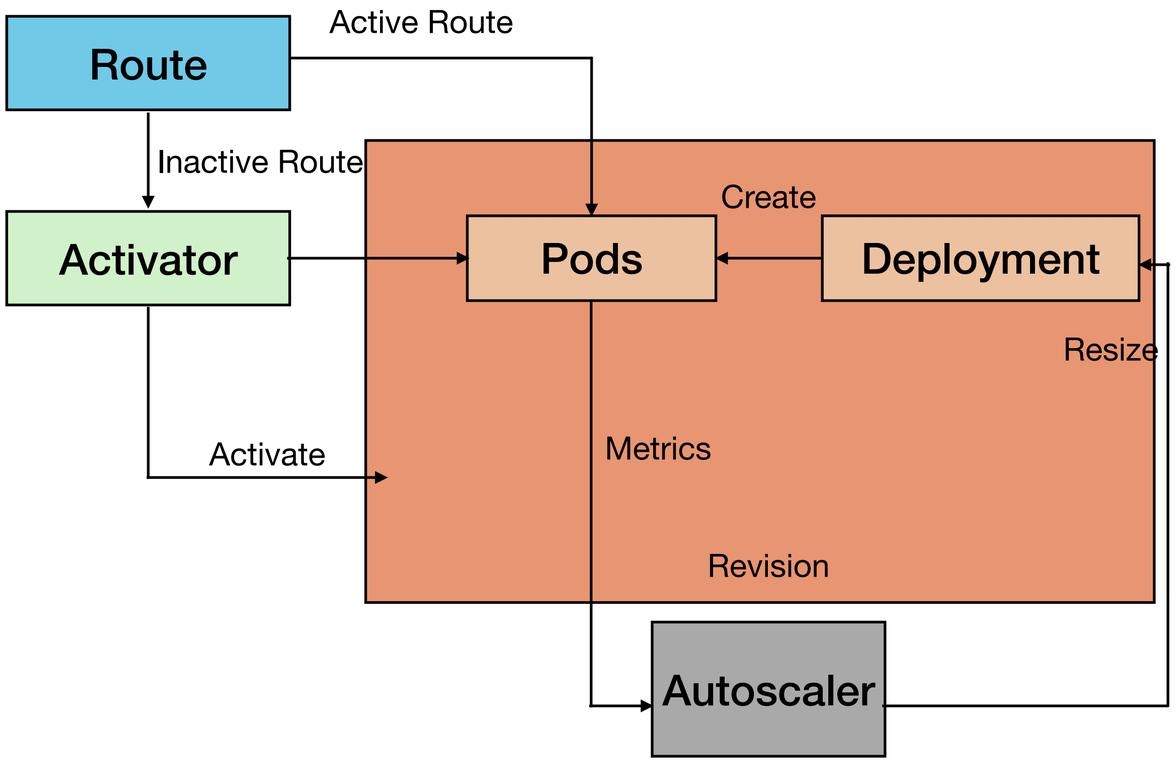
- Gather information about concurrent requests to revision
- Evaluate metrics and adjust size of revision's underlying deployment
- Revision states:
  - Active: Receives Traffic
  - Reserve: Deployment scales to zero, traffic routes to activator



Autoscaling

Activator

- Horizontally scalable, shared component that catches all traffic for Reverse revisions.
- Transition revision to Active and proxy request to appropriate pod.



Autoscaling - How it works?

- Algorithm based on average number of in-flight requests per pod (concurrency).
- Average data points over two intervals: 60-second and 6-second window.
- Operate in two different modes:
  - Stable: Determine average concurrency using 60-second window.
  - Panic: Enter mode if panic window reaches 2x the target concurrency.
    - Use 6-second window for scaling more responsive to sudden increase in traffic.
    - Only scale up during panic mode to prevent rapid fluctuations in pod count.
    - Transition back to stable mode after panic conditions no longer met for 60 seconds.

Demo

Knative Eventing

Knative Eventing -Introduction

- Manage coordination & delivery of events
  - Subscribe to event producers
  - Manage events when they come to your Knative environment
  - Send events between your Knative services
  - Orchestrate workflow

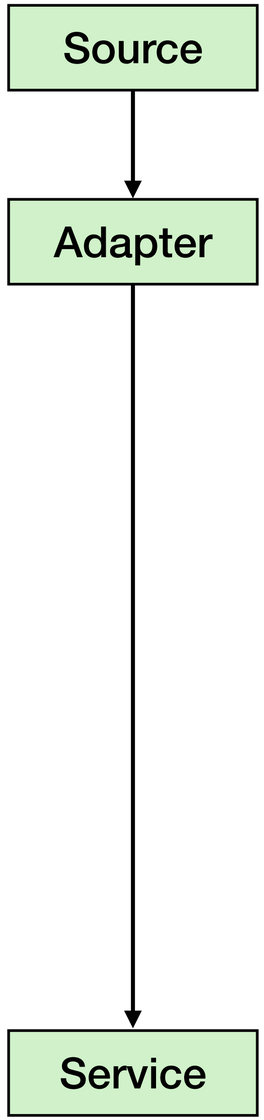
Knative Eventing Components

Event Source (producer):

- Read events from source and forward downstream
- Adapter to convert events into CloudEvents

Service (consumer)

- Knative service consuming the event stream



Knative Eventing Components

Event Source (producer):

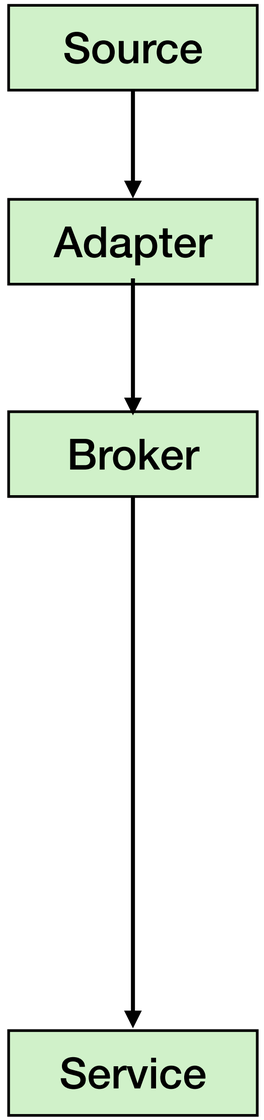
- Read events from source and forward downstream
- Adapter to convert events into CloudEvents

Broker:

- Receives events from source

Service (consumer)

- Knative service consuming the event stream



Knative Eventing Components

Event Source (producer):

- Read events from source and forward downstream
- Adapter to convert events into CloudEvents

Broker:

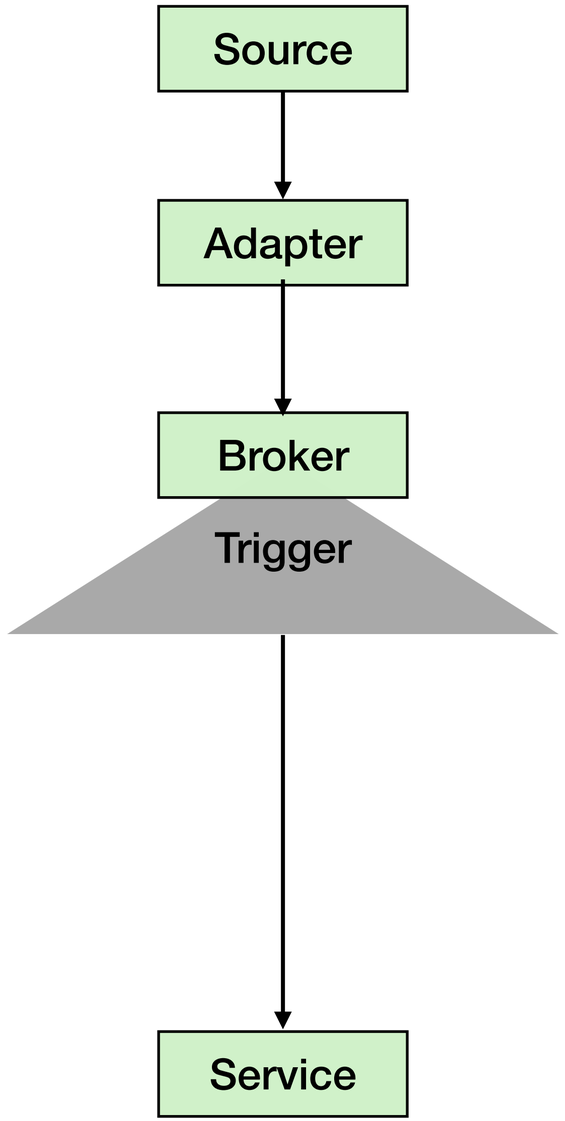
- Receives events from source

Trigger (subscription):

- Bridge between broker and service

Service (consumer)

- Knative service consuming the event stream



Delivery Methods

Approach 1- Source to Service

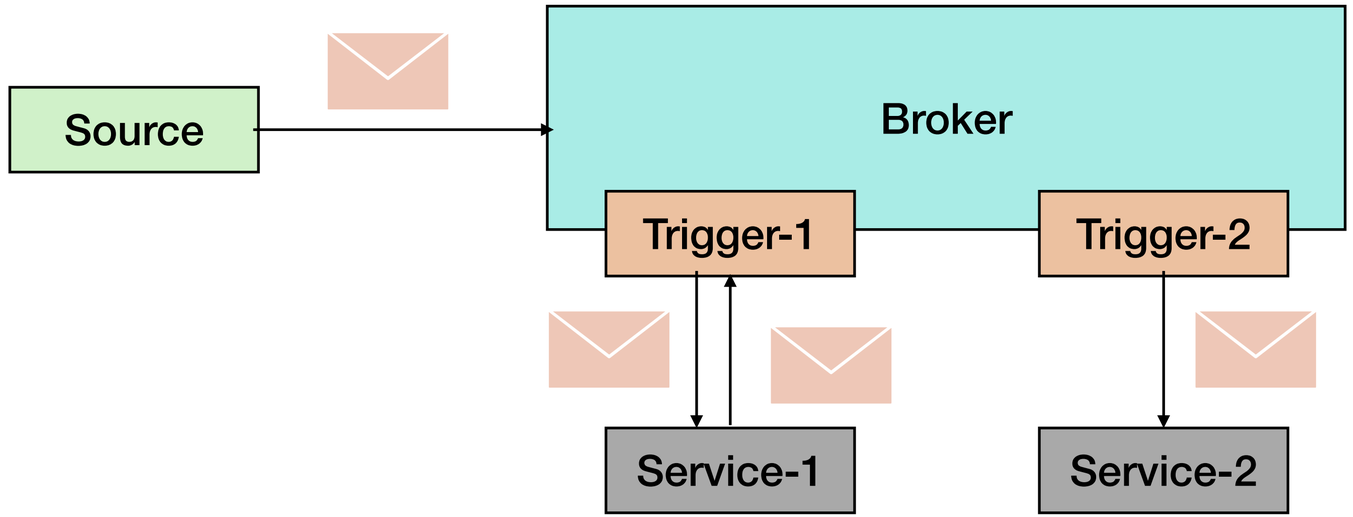
Event source sends messages directly to a service

- Simplest approach
- No delivery guarantees
- No queuing, no back pressure, no filtering
- No replies

Approach 2- Broker and Trigger

Receive event from source, save to underlying storage and fan out to all subscribers

- Fan-Out (Multiple sinks, multiple event receiving services)
- Handles persistence
- Supports replies
- Filtering



Demo

Thank You!!