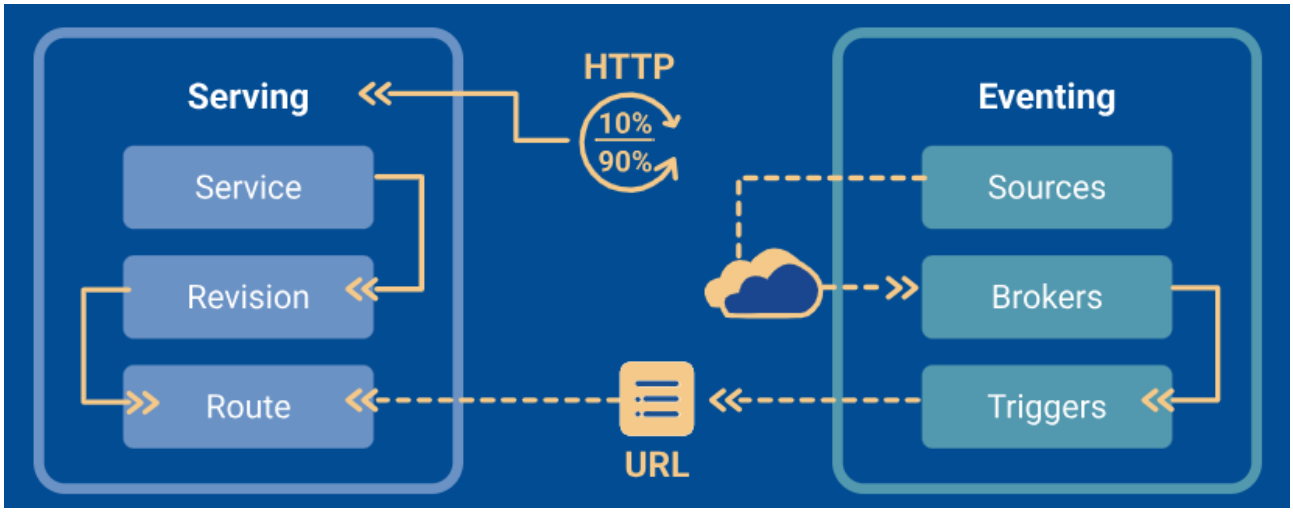
FAAS PLATFORMS

Introduction:

Before we start talking about how to implement dependency injection for serverless applications, we present FaaS platforms, focusing on two open-source solutions quite spread in the market. In particular, we will discuss about Knative and OpenFunction, their architectures and their main components. Being aware of such concepts permit us to identify and to evaluate the best strategy to implement dependency injection.

Knative:

Knative is an open-source enterprise-level solution to build serverless applications that are runnable in any cloud environment. It has been developed as a Kubernetes extension and it is, essentially, a set of APIs that enable to build and deploy event-driven application in general. Knative is composed by two independent blocks added just above Kubernetes: Serving and Eventing.



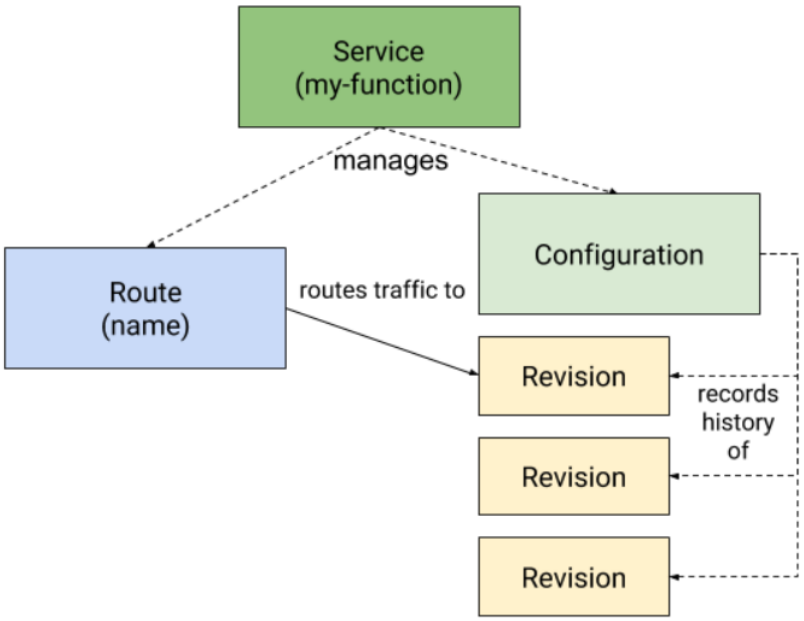
Serving:

It defines a set of objects as Kubernetes Custom Definitions (CRDs). These resources are used to define and control how our serverless workload behaves on the cluster. The main types of resources are:

- Route: it provides a long-lived, stable, named, HTTP-addressable network endpoint for a user's service that is backed by one or more Revisions.
- Revision: it represents a stateless, autoscaling snapshot-in-time of application and Configuration. Revisions enable progressive rollout and rollback of application

changes by changing the HTTP routing between Service names and Revision instances. As such, Revisions are generally immutable.

- Configuration: it describes the desired state of the latest Revision. Modifying a Configuration spec, a new Revision is created.
- Service: A Service encapsulates a Route and Configuration which together provide a software component. A Service exists to provide a singular abstraction which can be access controlled, reasoned about, and which encapsulates software lifecycle decisions. A Service acts only as an orchestrator of the underlying Route and Configuration.

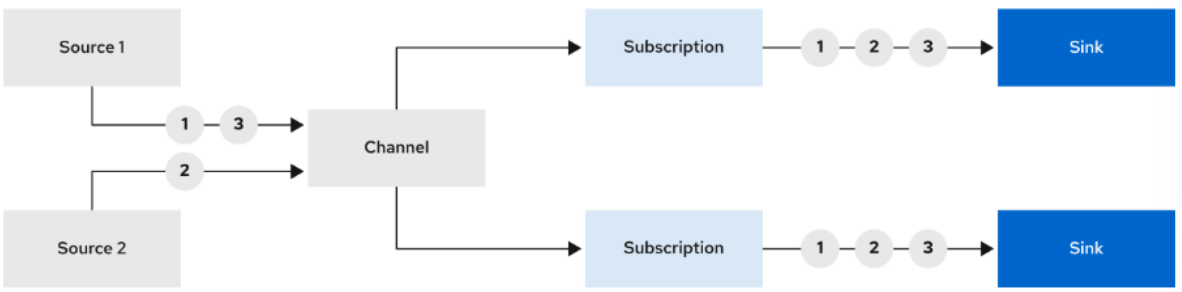


When a HTTP request arrives to an application that is running on Knative Serving,

Eventing:

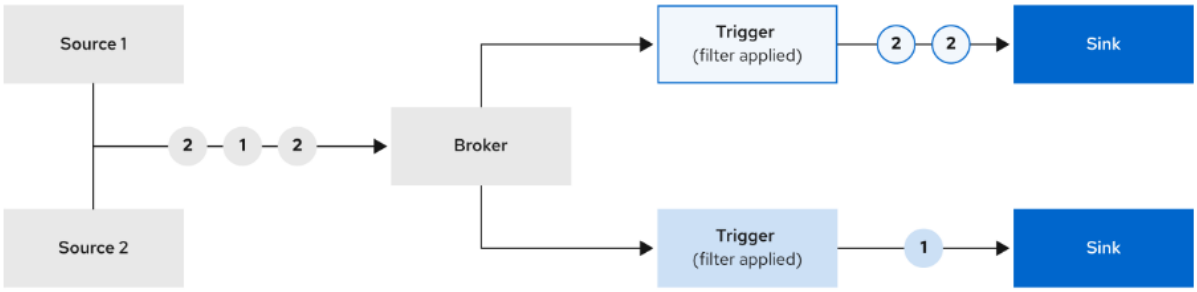
It is a collection of APIs that provides primitives for two types of event-processing patterns:

- Topology-based event routing: events are routed based on connections between objects. They are managed like flows of water through pipes. These pipes are called Channels and they redirect the events independently to each Subscription.



- Content-based event routing: events are selected for routing based on attributes. There are Brokers that provide a central event-routing hub which exposes a URL

address that is used to submit events. Triggers define delivery options to route events to local or external Destinations.



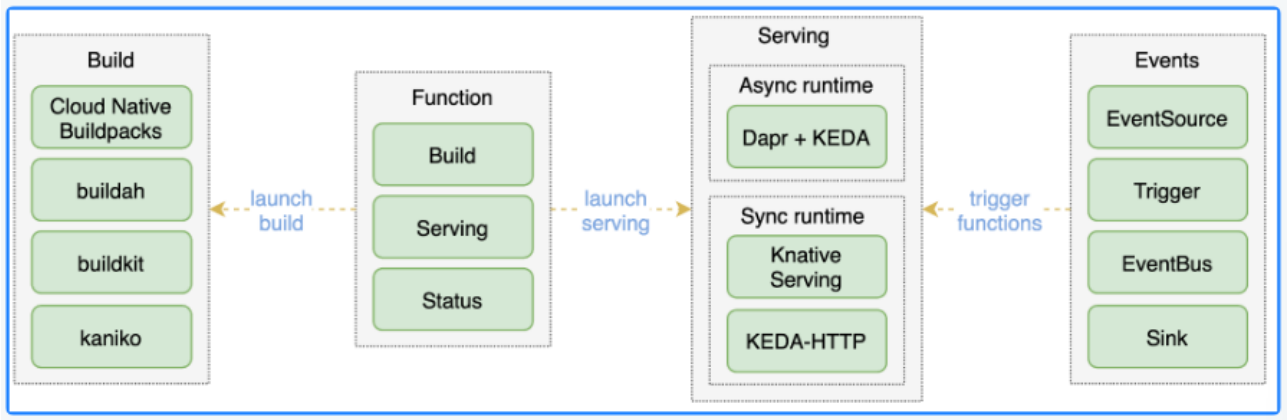
An event source is a Kubernetes custom resource (CR), created by a developer or cluster administrator, that acts as a link between an event producer and an event sink. A sink can be a k8s service, including Knative Services, a Channel, or a Broker that receives events from an event source.

Functions:

Knative Functions provide a simple programming model for using functions on Knative, without requiring in-depth knowledge of Knative, Kubernetes, containers or dockerfiles. They enable to easily create, build and deploy stateless, event-driven functions as Knative Services. When we build a function, a container image is automatically generated and stored in a registry.

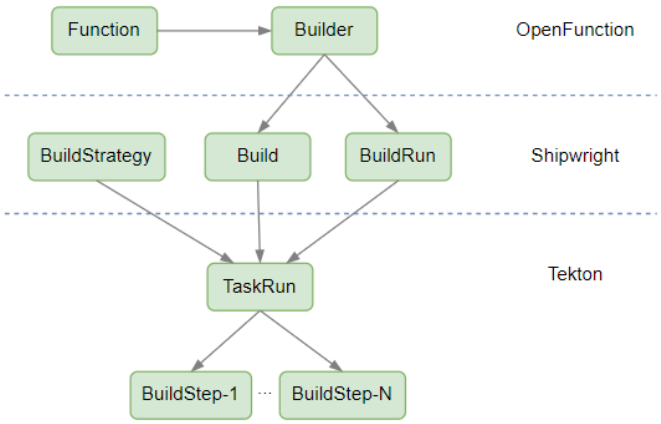
OpenFunction:

OpenFunction is a cloud-native open-source FaaS platform composed by 4 main components:

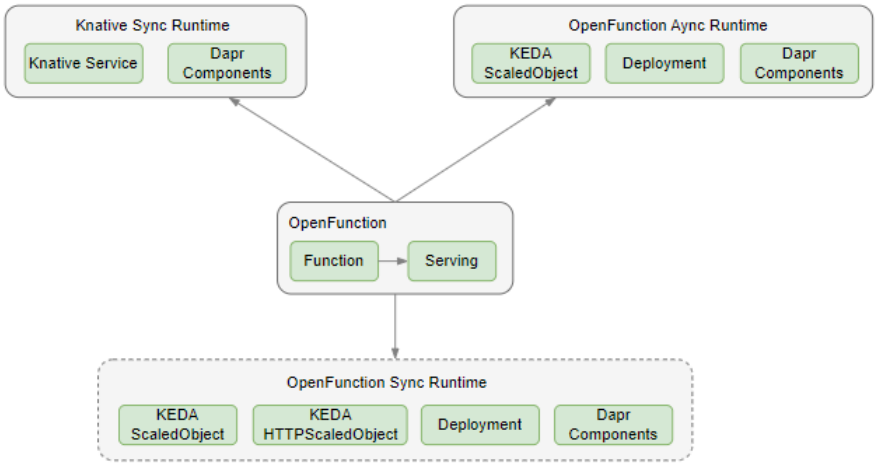


- Function: it is the control plane of Build and Serving and it's also the interface for users to work with OpenFunction. Once a function is created, it will control the lifecycle of Build and Serving without user intervention: if Build is defined in the function, a builder resource will be createdto build the container image for the function's deployment; if Serving is defined, a serving resource will be created to control function's scaling and autoscaling.

- Build: OpenFunction uses Shipwright and Cloud Native Buildpacks to build the function source code into container images. Once a function is created with Build spec in it, a builder custom resource will be created which will use Shipwright to manage the build tools and strategy. Shipwright will then use Tekton to control the process of building container imaged.



- Serving: Once a function is created with Serving spec in ti, a serving custom resource will be created to control a function's serving phase. Two runtimes are supported: sync runtime using Knative Serving and async runtime which uses event-driven runtime based on KEDA and Dapr.

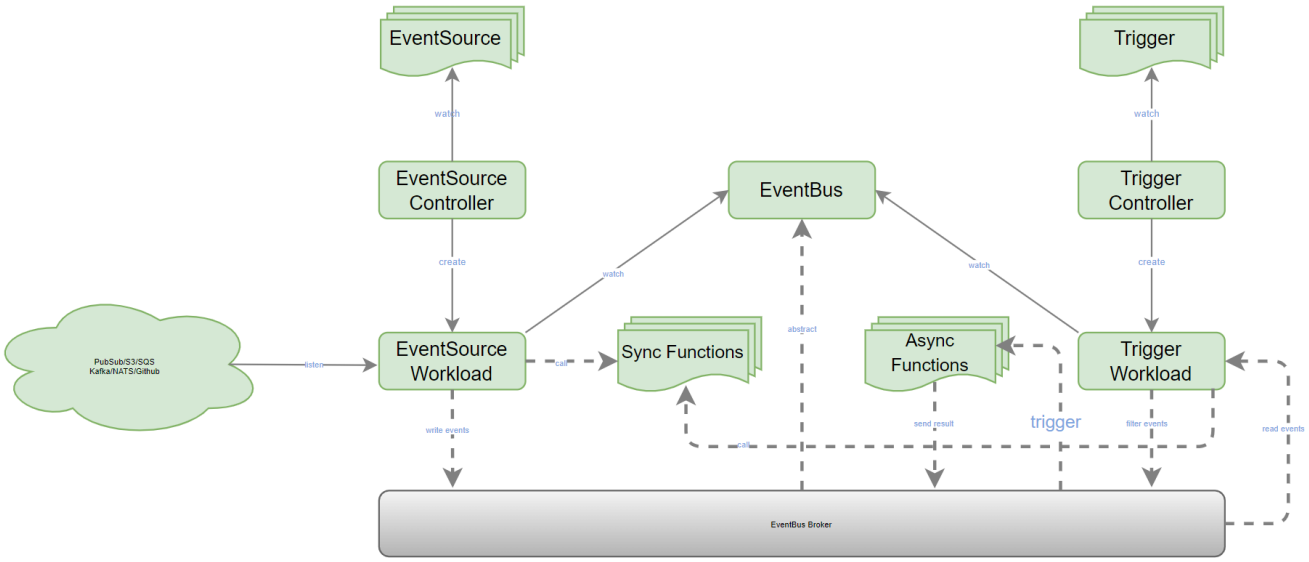


- Events: it is OpenFunction's event management framework. It provides support for triggering target functions and defining trigger judgement logic. It is based on three main concepts:

    1) EventSource: it defines the producer of an event. It can be a Kafka service, an object storage service or a function.

    2) EventBus: it is responsible for aggregating events and making them persisten.

    3) Trigger: it is an abstraction of the purpose of an event, such as what need to be done when a message is received.



References:

https://knative.dev/docs/

https://github.com/knative/specs/tree/main/specs

https://openfunction.dev/docs/