How it works

Table of Contents

The `funktion operator` watches for `Flow` and `Function` resources.

When a new `function` is created then the operator will spin up a matching `Deployment` for running the `function source code` along with a `Service` to expose the service as a HTTP or HTTPS endpoint.

When a new `flow` is created then this operator will spin up a matching `Deployment` which consumes from some `Connector` and typically invokes a function using HTTP.

The following kubernetes resources are used:

Kubernetes Resources

A `function` is modelled as a Kubernetes `ConfigMap` with the label `kind.funktion.fabric8.io: "Function"` which contains the source code of the function inside the `Data['source']` entry.

A `flow` is modelled as a Kubernetes `ConfigMap` with the label `kind.funktion.fabric8.io: "Flow"`. A ConfigMap is used so that the entries inside the `ConfigMap` can be mounted as files inside the `Deployment`. For example this will typically involve storing the `funktion.yml` file or maybe a Spring Boot `application.properties` file inside the `ConfigMap` like this example flow

A `Connector` is generated for every Camel Component and each connector has an associated `ConfigMap` resource like this example which uses the label `kind.funktion.fabric8.io: "Connector"`. The `Connector` stores the `Deployment` metadata, the `schema.yml` for editing the connectors endpoint URL and the `documentation.adoc` documentation for using the Connector.

So a `Connector` can have `0..N flows` associated with it. For those who know Apache Camel this is like the relationship between a `Component` having `0..N Endpoints`.

For example we could have a Connector called `kafka` which knows how to produce and consume messages on Apache Kafka with the Connector containing the metadata of how to create a consumer, how to configure the kafka endpoint and the documetnation. Then a flow could be created for `kafka://cheese` to subscribe on the `cheese` topic and post messages to `http://foo/`.

Typically a number of `Connector` resources are shipped as a package; such as inside the Red Hat iPaaS or as an app inside fabric8. Though a `Connector` can be created as part of the CD Pipeline by an expert Java developer who takes a Camel component and customizes it for use by `Funktion` or the `iPaaS`.

The collection of `Connector` resources installed in a kubernetes namespace creates the `integration palette` thats seen by users in tools like CLI or web UIs.

Then a `flow` can be created at any time by users from a `Connector` with a custom configuration (e.g. choosing a particular queue or topic in a messaging system or a particular table in a database or folder in a file system).

Debugging

If you ever need to you can debug any `flow` as each flow matches a `Deployment` of one or more pods. So you can just debug that pod which typically is a regular Spring Boot and camel application.

Otherwise you can debug the pod thats exposing an HTTP endpoint using whatever the native debugger is; e.g. using Java or NodeJS or whatever.