前言

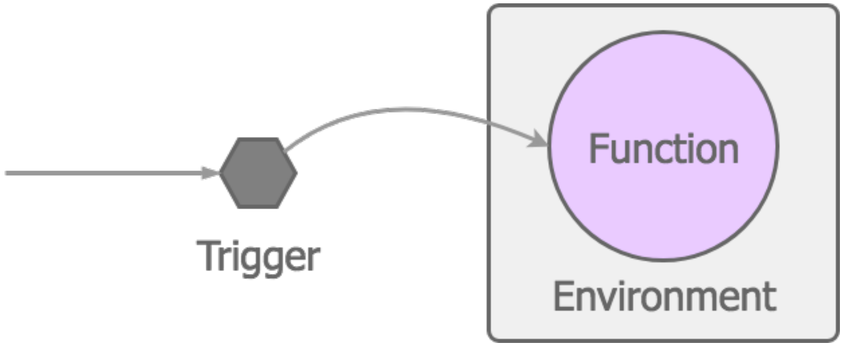
faas框架的核心任务

- 一个faas框架必须要：（1）把function转换为services（2）管理这些services的生命周期

fission简介

fission核心概念

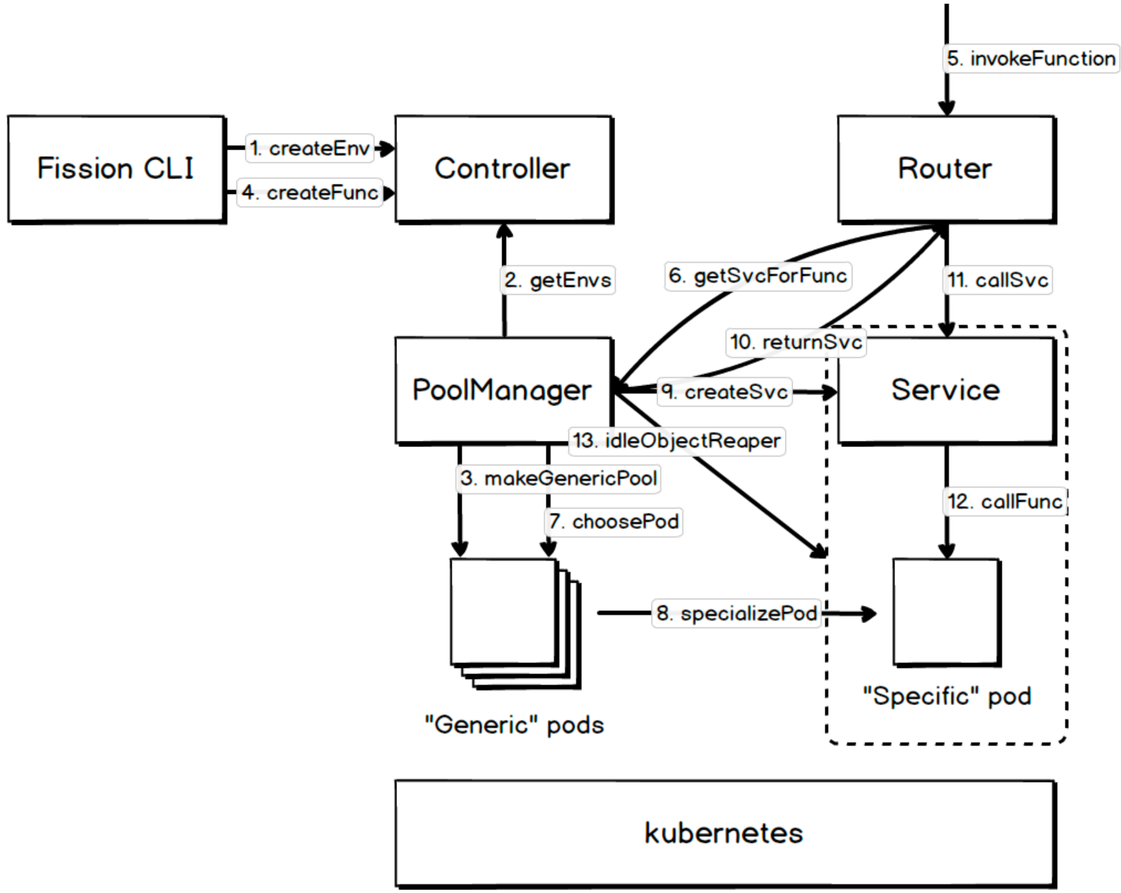
- Function
  - 特定语言编写的需要被执行的**代码片段**
- Environment
  - 用于运行用户函数的**特定语言**环境
- Trigger
  - 用于关联函数和**事件源**



fission简介

fission关键组件

- Controller
  - 它是**fission CLI**的主要交互对象
  - 提供了针对**fission资源**的增删改查的操作接口，包括functions、event triggers、HTTP route以及environment images。
    - fission资源都存储在k8s CRD(custom resource)上；
- Router
  - **函数访问入口**，同时也实现了**HTTP触发器**。它负责将用户**HTTP请求**以及各种事件源产生的**事件**转发至**目标函数**。
- Executor
  - 包含**Pool Manager**和**NewDeploy**两类执行器，它们控制着fission函数的生命周期（创建 -> 销毁）
    - Pool Manager 很好地平衡了函数的**冷启动时间**与**闲置成本**；但无法让函数根据**度量指标 自动伸缩；**
    - NewDeploy 实现了函数pod的**自动伸缩**与**负载均衡**



fission的基本原理

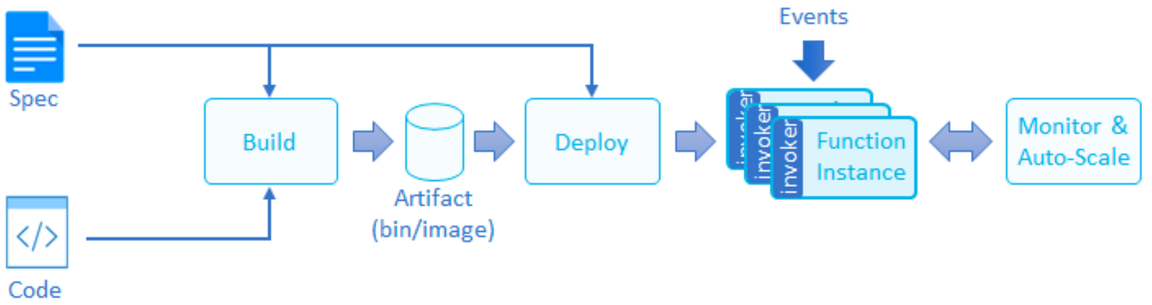
概述

- 函数执行器(executor)：理解**fission工作原理**的基础；
- Pod特化（specialize）：理解fission如何根据**用户源码**构建出**可执行函数**的关键；
- 触发器(trigger)：理解fission函数各种**触发原理**的入口；
- 自动伸缩：理解 fission 如何根据**负载动态**调整**函数个数**的捷径

fission的基本原理

函数生命周期

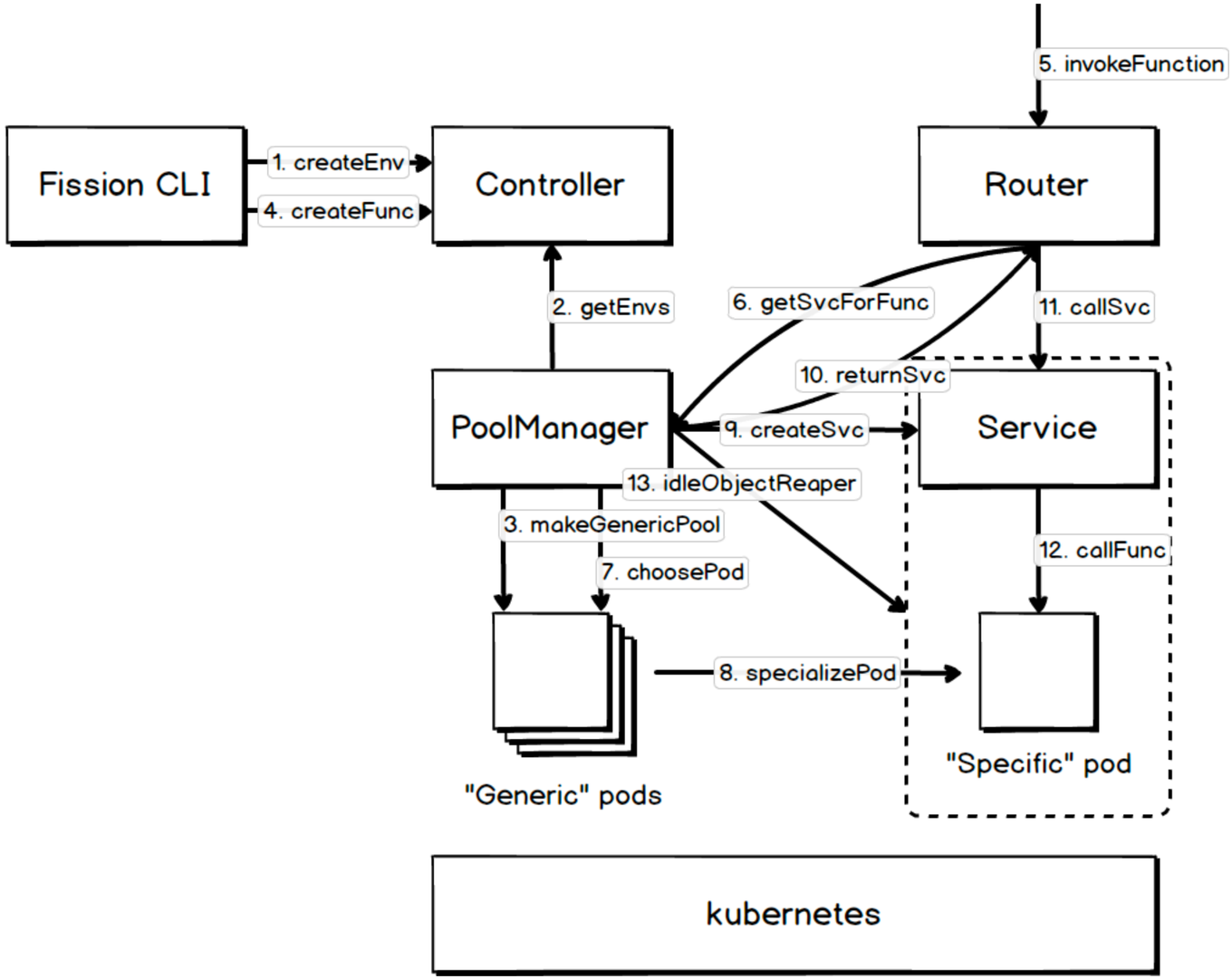
- CNCF对函数生命周期的定义
  - 函数构建、部署、运行的一般流程



fission的基本原理

函数执行器 -- Pool Manager

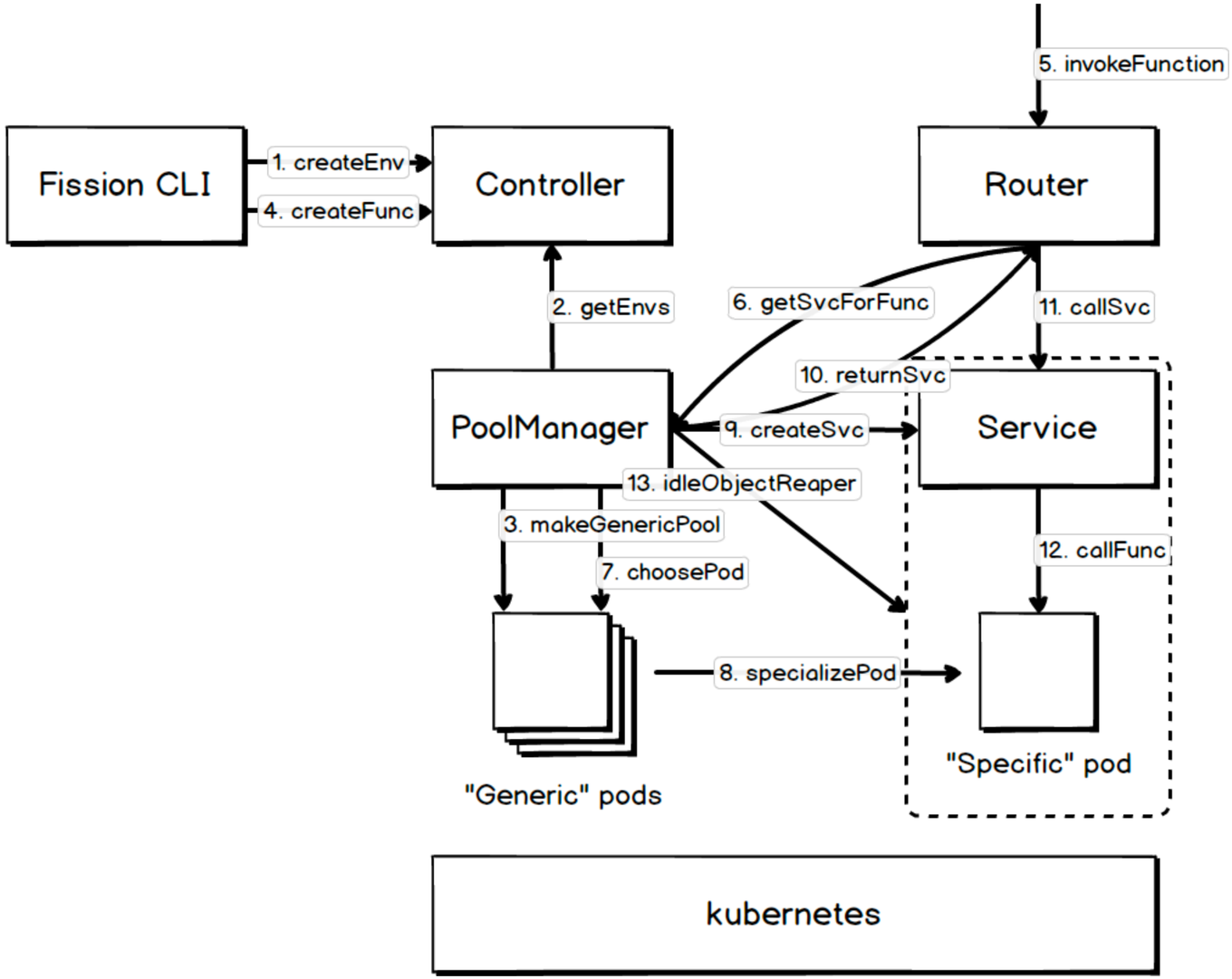
- 使用**池化**技术
  - 为每个 environment 维持了一定数量的**通用 pod** 并在函数被触发时将 pod **特化**，**大大降低**了函数的**冷启动**的时间
  - 自动清理一段时间内未被访问的函数（闲置函数）



fission的基本原理

函数执行器 -- Pool Manager -- 函数的生命周期

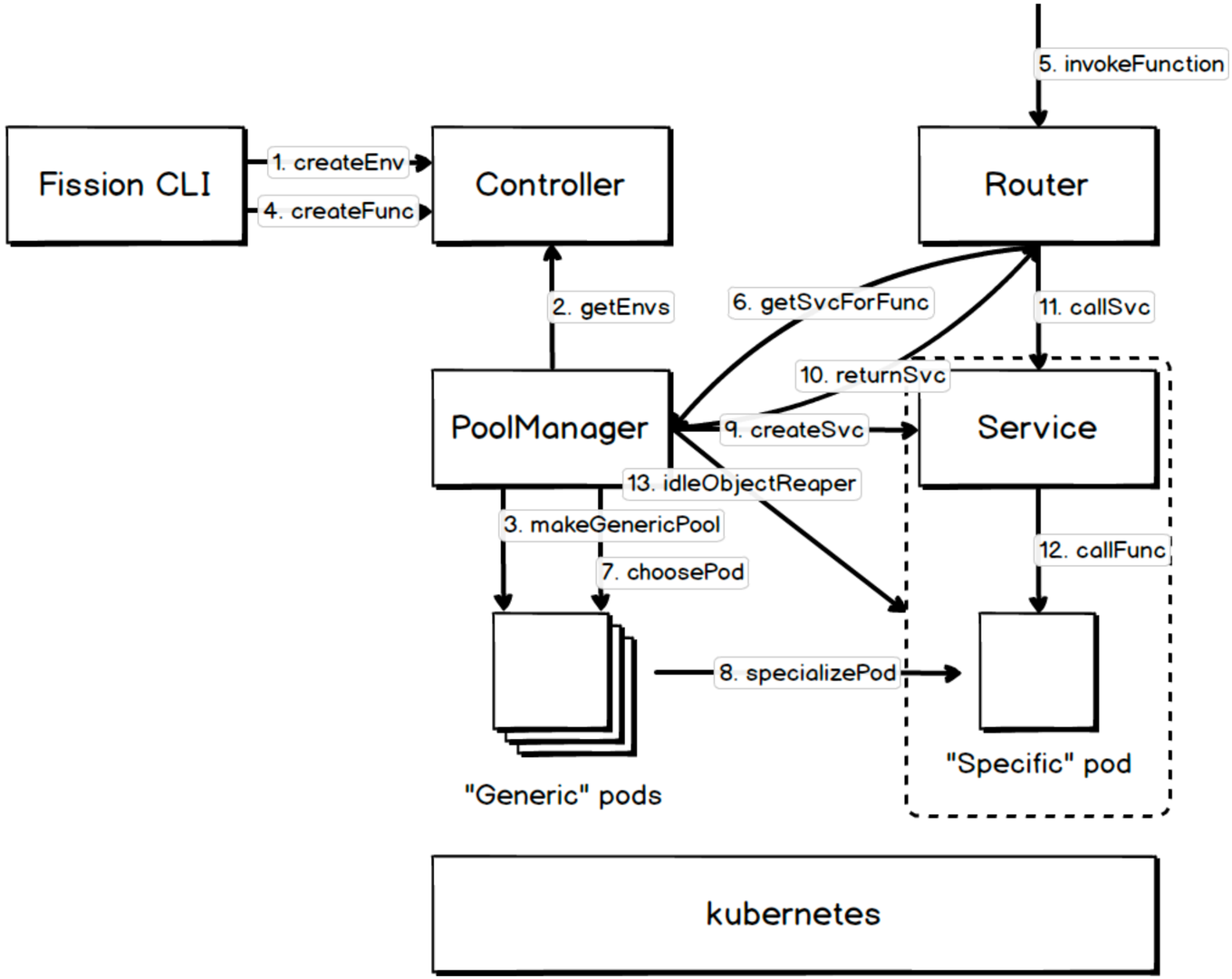
- fission向controller发出请求，创建函数运行所需的**语言环境**；
  - fission environment create --name **python** --image fission/python-env
- poolmgr定期同步env资源列表
- poolmgr遍历**env列表**，为**每个env**创建一个通用pod池



fission的基本原理

函数执行器 -- Pool Manager -- 函数的生命周期

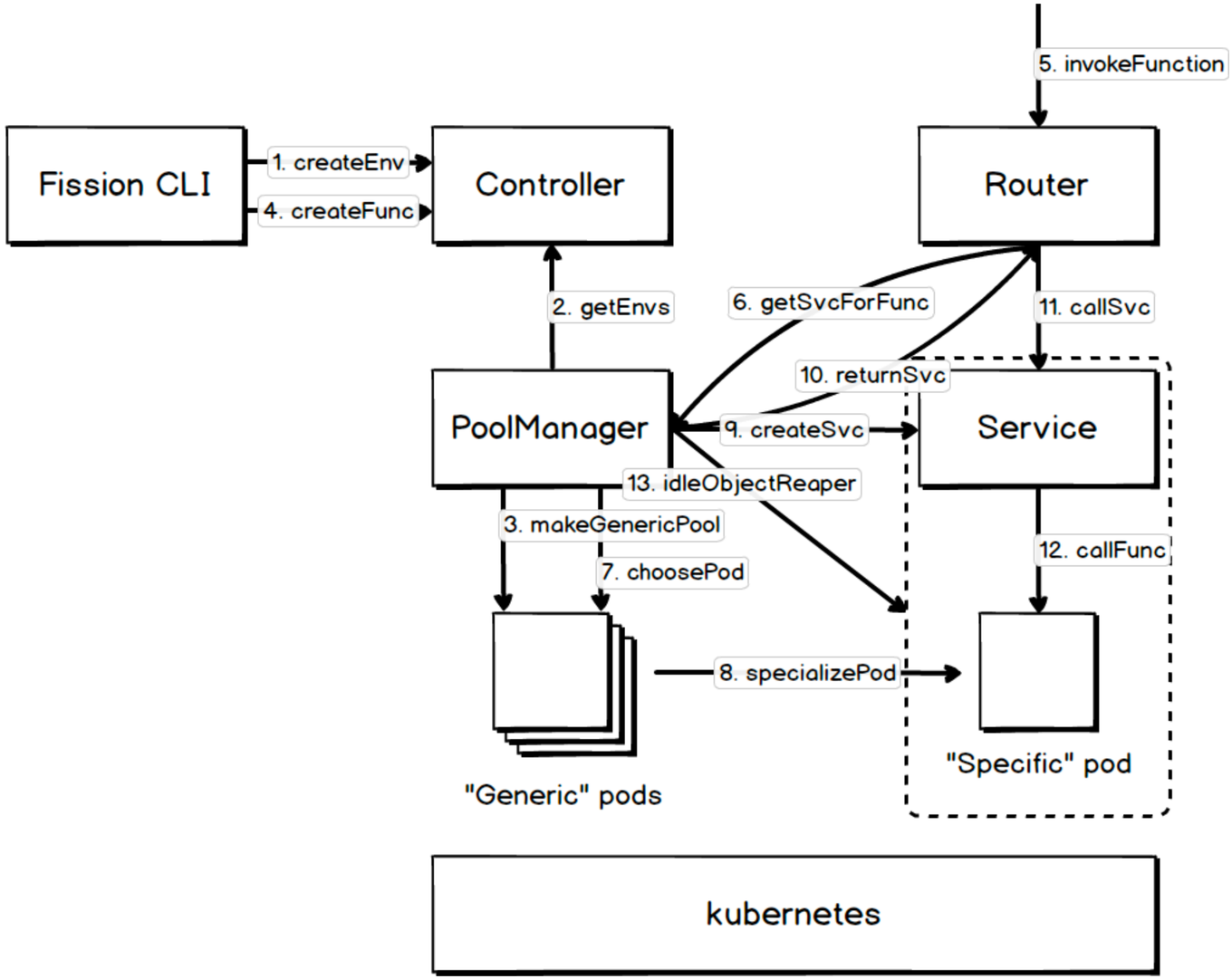
- 使用CLI发送**创建函数**的请求
  - controller将函数**源码**等信息**持久化存储**（未构建可执行函数）
- router接受触发**函数执行**的请求
- router向executor发送**请求**获取**函数访问入口**



fission的基本原理

函数执行器 -- Pool Manager -- 函数的生命周期

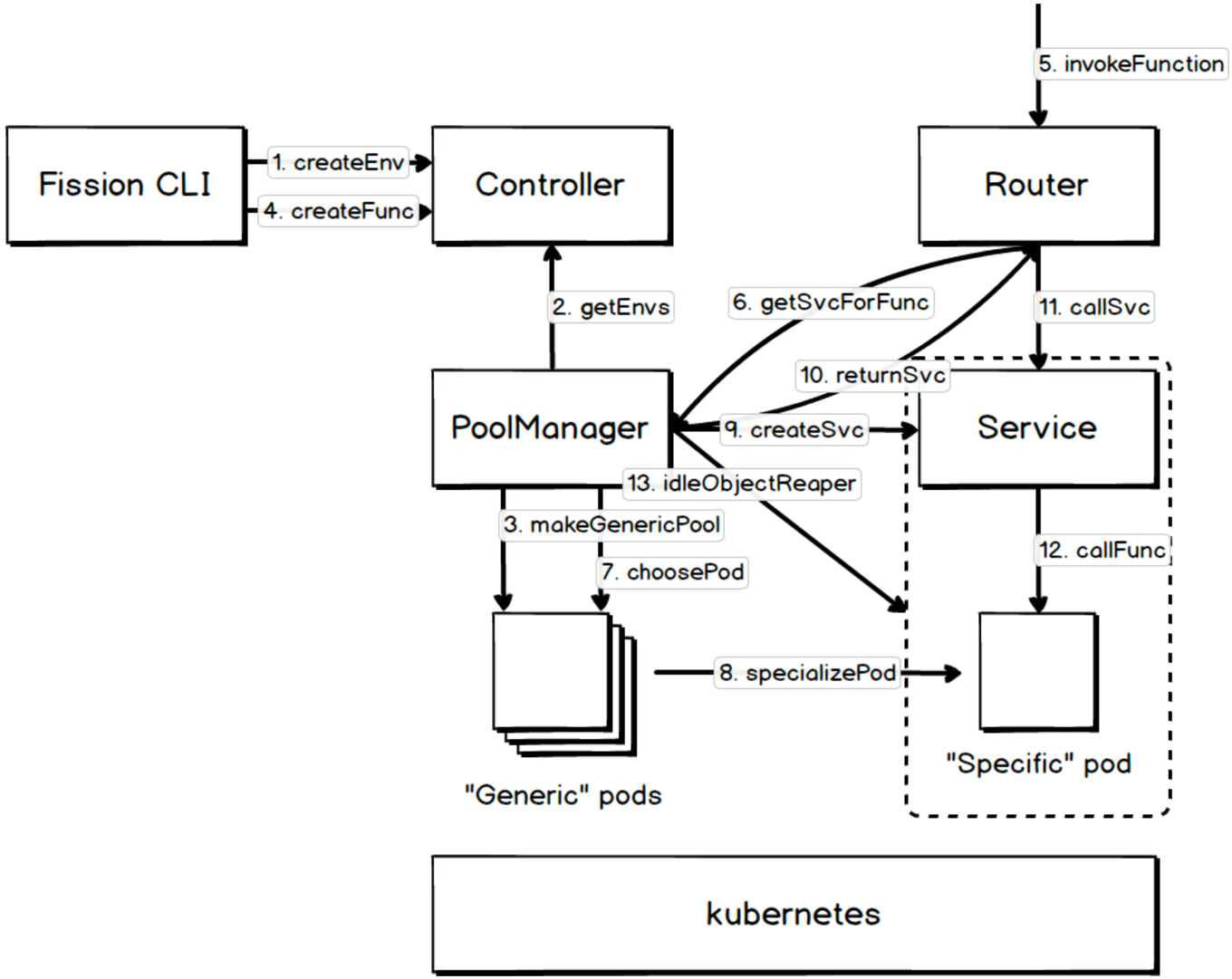
- poolmgr从通用pod池**随机选择**一个pod作为载体
  - 这里会**更改pod标签**让其从deployment中**独立**出来，k8s发现deployment管理的pod副本变少之后会**自行补充**pod，**维持pod池中的数量**
- **特化处理**挑选出来的pod
  - 构建可执行函数
- 为pod创建custerIP类型的service
  - ClusterIP：集群内部使用的service
- 将函数的**service信息**返回给router，router会将**serviceUrl缓存**，以免频繁向executor发请求



fission的基本原理

函数执行器 -- Pool Manager -- 函数的生命周期

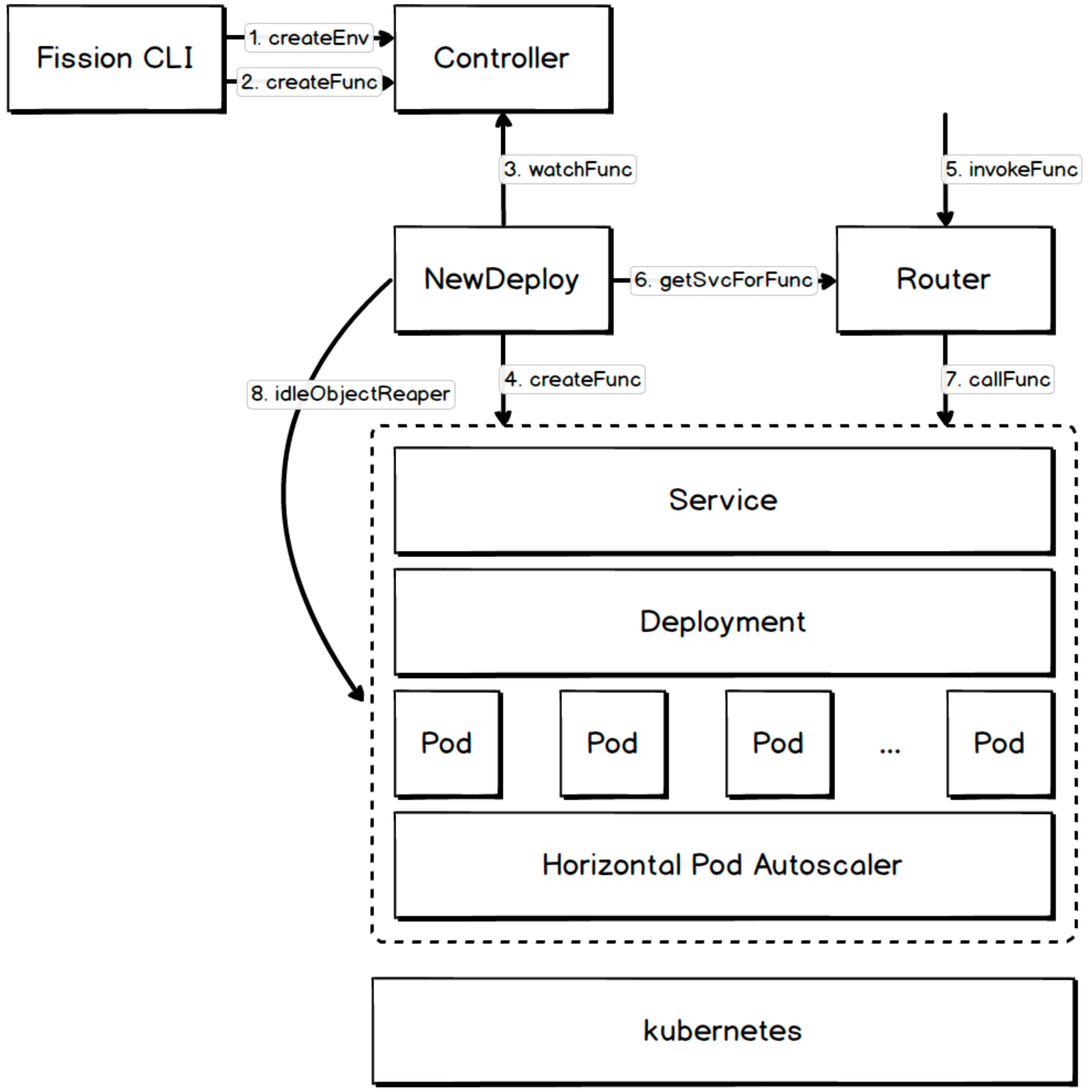
- router使用返回的serviceUrl**访问函数**
- 请求**最终被路由到运行函数的pod**
- 如果该一段时间没有被运行，就会被自动清理（包括pod与service）



fission的基本原理

函数执行器 -- NewDeploy -- 函数的生命周期

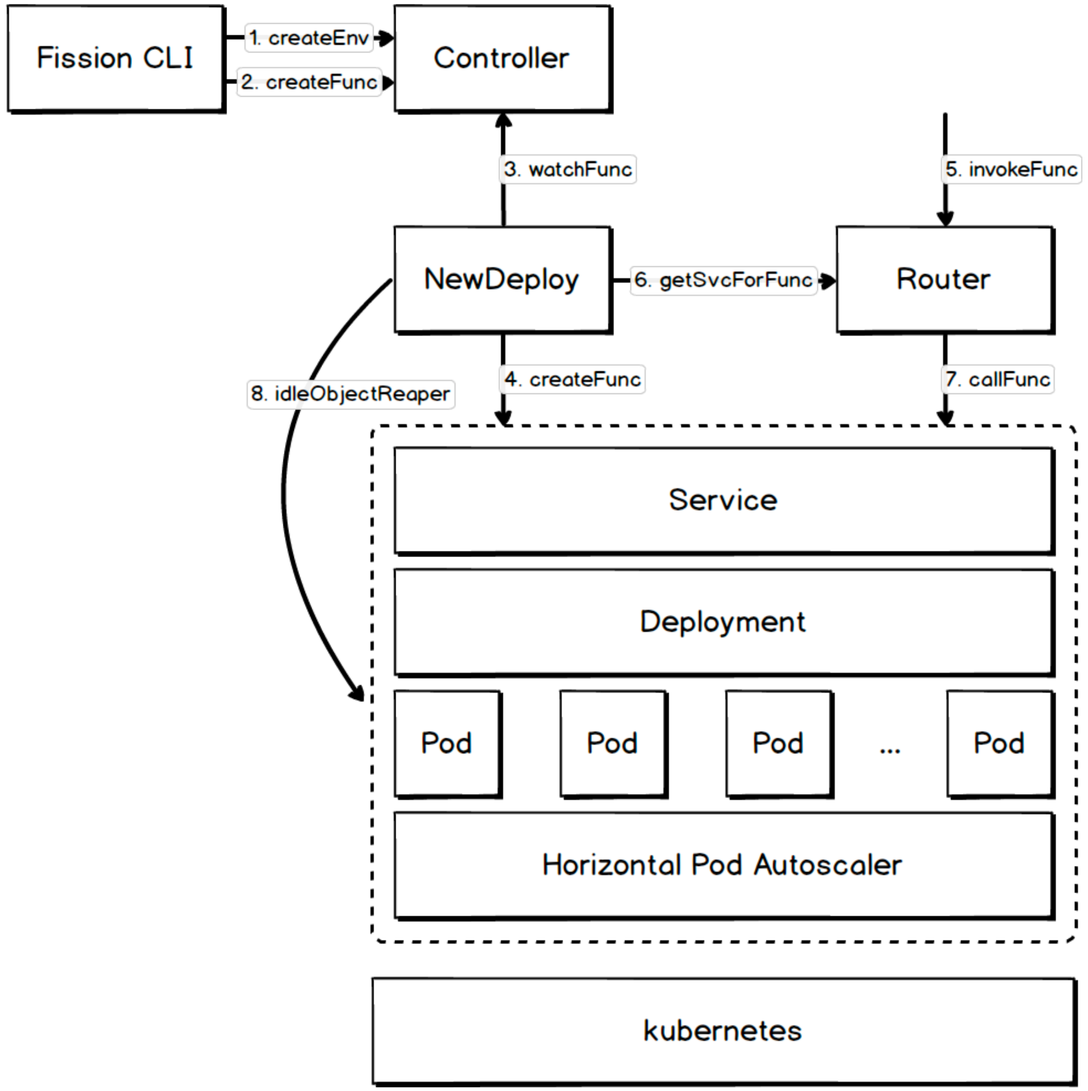
- 请求创建语言环境
- 使用CLI向controller请求**创建函数**
  - fission fn create --name hello --**env** python --**code** hello.py --**executortype** newdeploy --**minscale** 1 --**maxscale** 3 --**targetcpu** 50
  - 选择已经创建好的python环境
  - 源码来自hello.py
  - 执行器类型为newdeploy
  - 目标副本为1～3
  - 目标cpu使用率是50%



fission的基本原理

函数执行器 -- NewDeploy -- 函数的生命周期

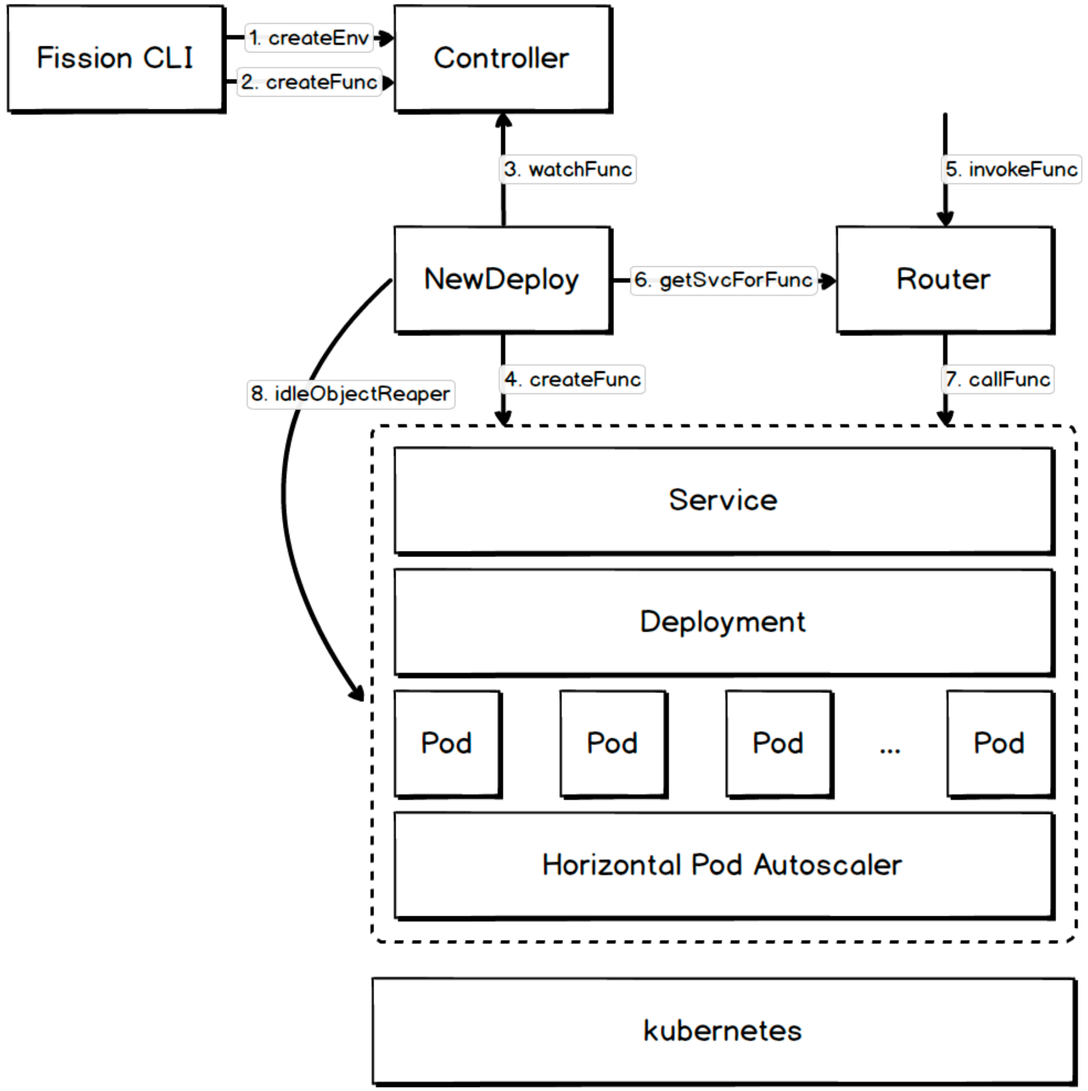
- newdeploy注册一个funcController持续**监听**对function的增、改、删；
- newdeploy监听到**函数的Add事件**后，会根据**minscale的取值**判断是否**立即**为该函数创建相关资源
  - 若minscale **> 0**，则立即创建
  - 若 **<= 0**，则延迟到函数真正被触发时创建
- router接受调用函数的请求



fission的基本原理

函数执行器 -- NewDeploy -- 函数的生命周期

- router向newdeploy发送请求，获取函数访问入口
  - 如果资源已被创建，直接返回访问入口
  - 否则，创建好相关资源后再返回
- router使用返回的serviceURL访问函数
- 如果该函数一段时间内未被访问，函数的**目标副本数**会被调整为minScale，但**不会删除**servive、deployment、HPA等资源



fission的基本原理

函数执行器的比较

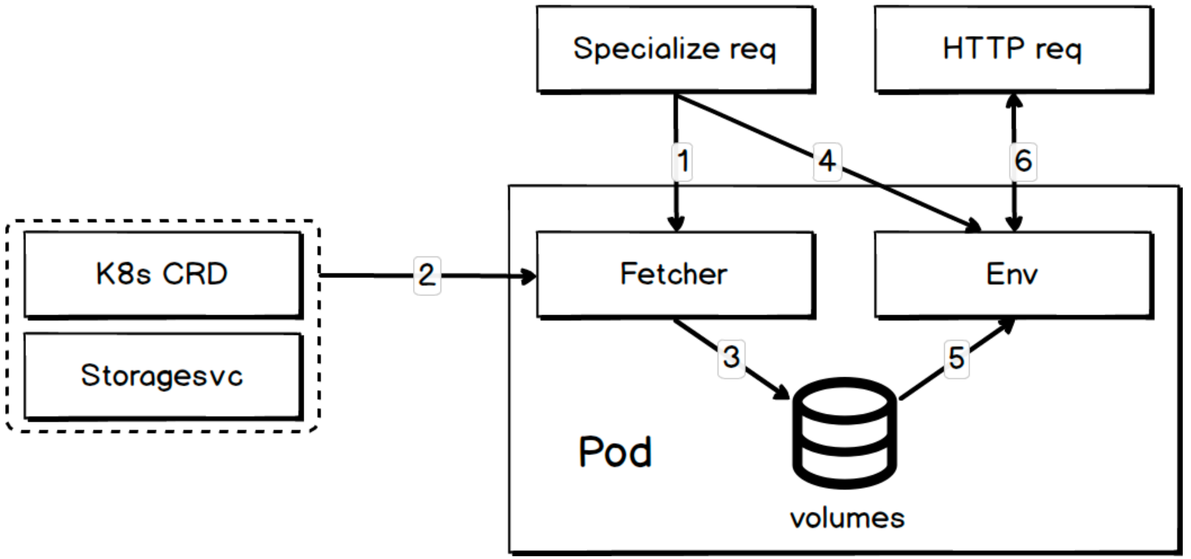
- “实际上可以将 poolmgr 和 newdeploy 技术相结合，通过创建 deployment 将特化后的 pod 管理起来，这样可以很自然地利用 HPA(horizontal pod autoscaling) 来实现对函数的自动伸缩。”

执行器类型	最小副本数	延迟	闲置成本
Newdeploy	0	高	非常低 - pods 一段时间未被访问会被自动清理掉。
Newdeploy	>0	低	中等 - 每个函数始终会有一定数量的 pod 在运行。
Poolmgr	0	低	低 - 通用池中的 pod 会一直运行。

fission的基本原理

Pod特化

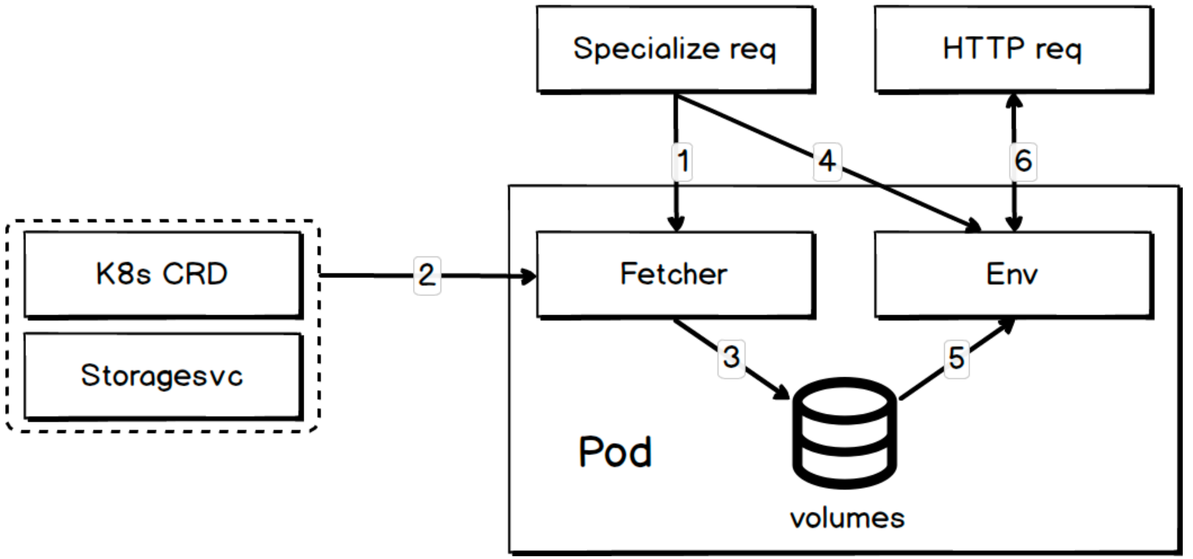
- 作用是将**环境容器**变成**函数容器**。
  - 通过向容器发送特化请求，让其**加载用户函数**；
- 一个函数pod由下面两种容器组成
  - fetcher：**下载**用户函数并把它放到共享的共享的volume中
  - env：用户**函数运行的载体**；当它成功加载共享volume 中的函数后，便可以接受用户请求



fission的基本原理

Pod特化

- 具体步骤
  - fetcher接受到拉取用户函数的**请求**
  - fetcher从**k8s CRD**(custom resource)或**storagesvc**处获取用户函数
  - fetcher将函数文件放置在共享的volume
  - env**容器**接收到加载用户函数的命令
  - env从volume从加载用户函数
  - 流程结束，env**开始处理用户请求**

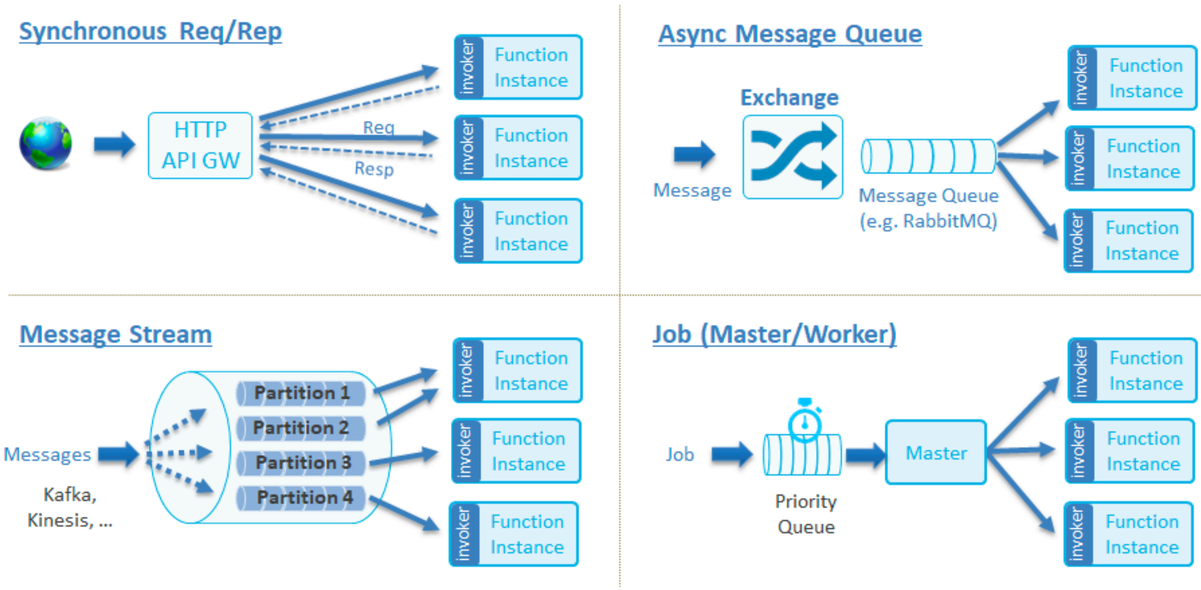


fission的基本原理

触发器

- fission支持的触发方式

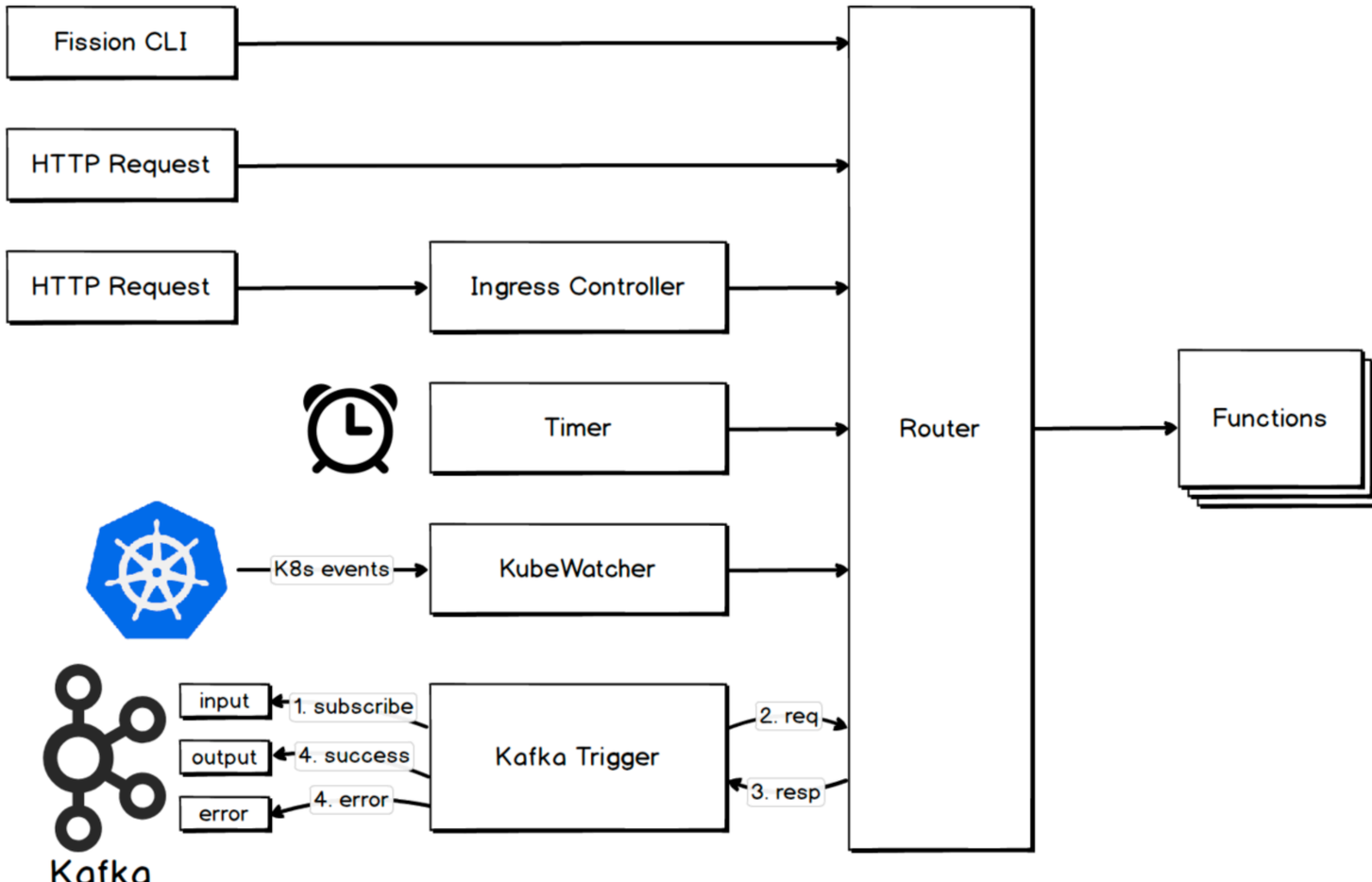
Triggering method	Category
fission CLI	Synchronous Req/Rep
HTTP Trigger	Synchronous Req/Rep
Time Trigger	Job (Master/Worker)
Message Queue Trigger 1. nats-streaming 2. azure-storage-queue 3. kafka	Async Message Queue
Kubernetes Watch Trigger	Async Message Queue



fission的基本原理

触发器

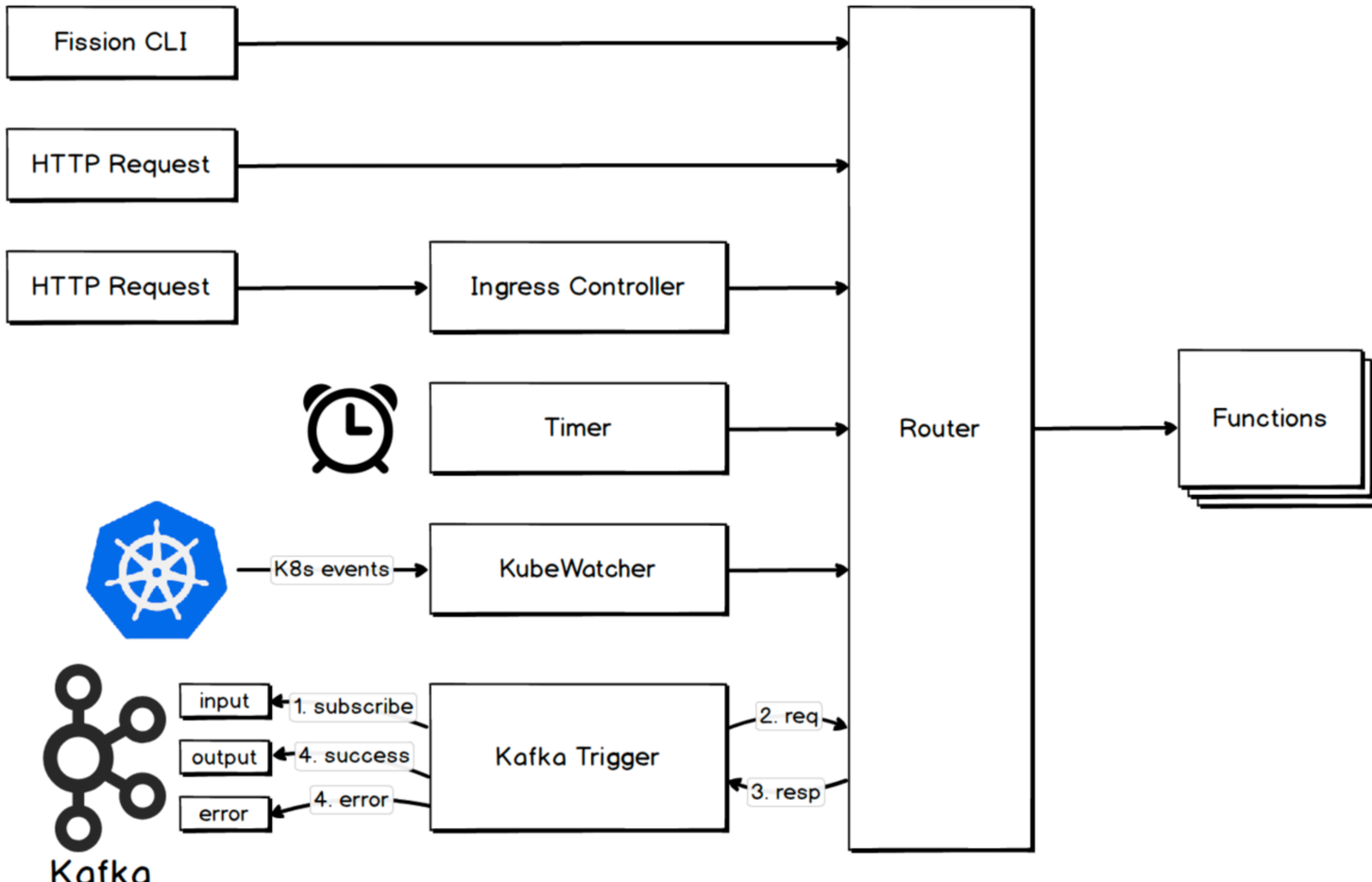
- 所有发往fission函数的请求都会由router转发；
- fission通过为router创建NodePort或LoadBalancer类型的**(k8s) service**能够让它被外界访问
  - NodePort：把服务暴露在每个Node IP的**静态port**上
  - LoadBalancer：通过云服务提供商的load balancer 把服务暴露给**外部**



fission的基本原理

触发器 -- HTTP trigger

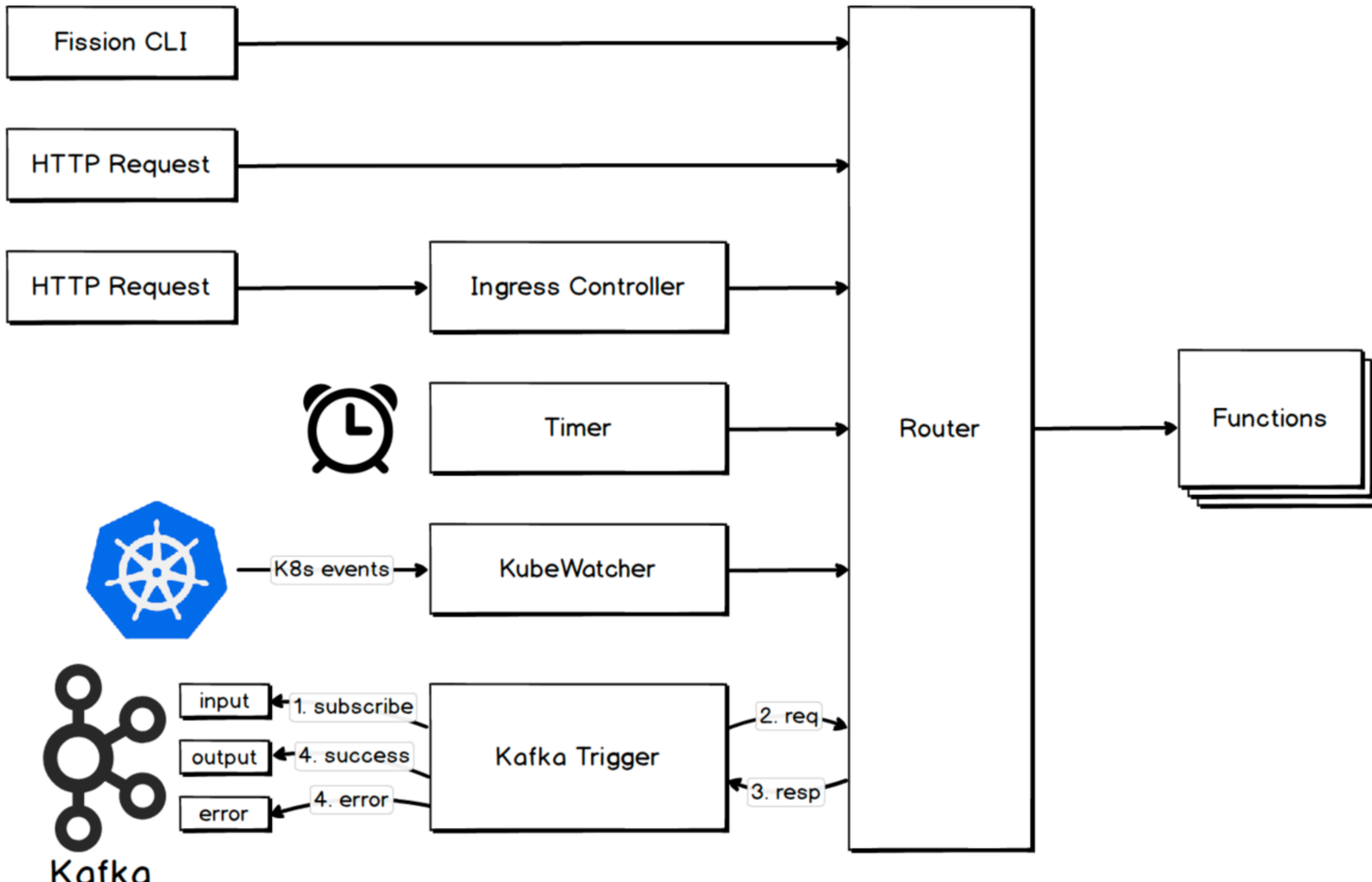
- 可以利用k8s ingress机制实现http trigger



fission的基本原理

触发器 -- Time trigger

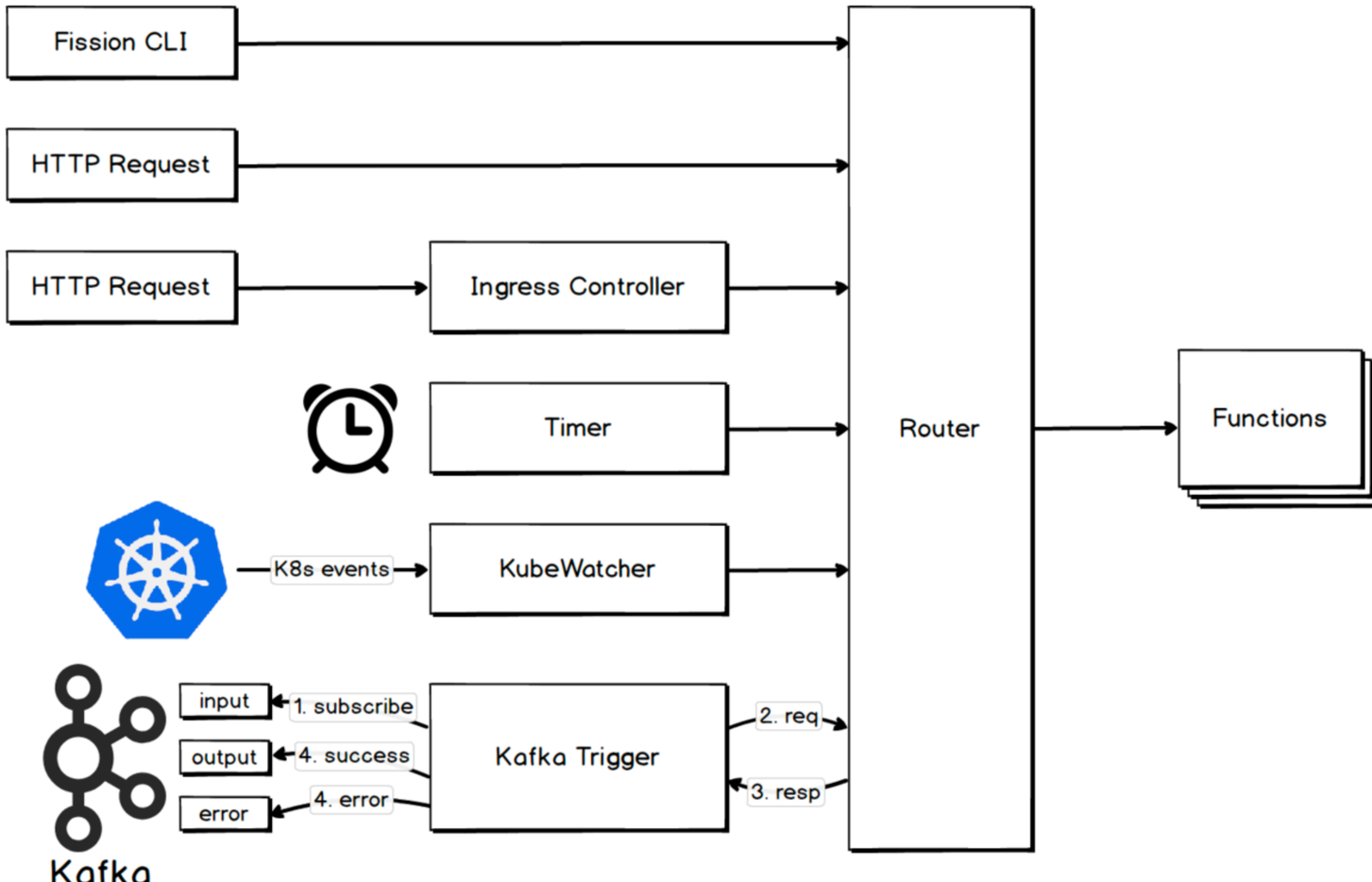
- **定期**触发函数执行；
- fission通过deployment部署组件**timer** —— 负责管理用户创建的timer trigger



fission的基本原理

触发器 -- Message queue trigger

- 用于支持**异步触发**
- 以kafka为例
  - 触发器**订阅了input** 中的消息
  - 每当有消息到达，便会触发函数执行；如果执行成功，会将结果写入output中，不成功，则写入error中；



fission的基本原理

自动伸缩

- k8s通过horizontal pod autoscaler(**HPA**) 实现pod的自动水平伸缩；
- fission只有通过**newdeploy**方式创建的函数才能利用**HPA**实现自动伸缩；
- 利用hpa可以指定**容器上使用资源的百分比**（如cpu利用率、内存）

在k8s上实现fission的细节

Packaging, source code, and image

- source level
  - 优势：为了方便用户使用，fission能够在**源码级别**操作。这样用户可以**不用处理**构建container image、把image上传到registry、管理registry证书、image版本等等工作
  - 劣势：一个源码级别的操作方式不能允许用户**打包依赖**
- container images
  - 是最灵活的打包app的方式。允许用户打包二进制依赖。
- 所以fission采取混合的方式 -- 采用包含了dynamic function loader的container image
  - 这种方式允许用户在**源码级别**使用，也允许它们**自定义container image**
  - 这些image（被称为环境镜像），包含了语言需要的runtime、一组**通用**的依赖、一个dynamic loader for function
    - 如果通用依赖已经够用了，那么就不用rebuild image；如果不够用，可以修改依赖列表，然后rebuild image。

在k8s上实现fission的细节

Cold start performance

- 为了优化冷启动的性能，fission为每个环境保持了一个处于运行中的container pool（**Generic pod**）（使用Pool manager）
- 对于NodeJS和Python函数，这个过程的开销约为**100ms**

使用方法

例子

- # Add the stock Python env to your Fission deployment
- $ fission env create --name python --image fission/python-env

- # A Python function that prints "hello world"
- $ curl -LO https://raw.githubusercontent.com/fission/examples/main/python/hello.py

- # Upload your function code to fission
- $ fission function create --name hello-py --env python --code hello.py

- # Test your function. This takes about 100msec the first time.
- $ fission function test --name hello-py
- Hello, world!

参考

- Fission: Serverless Functions as a Service for Kubernetes
- 开源 serverless 产品原理剖析（二） - Fission