Red Hat build of Apache Camel K 1.10.5

将应用程序与 Kamelets 集成

配置连接器以简化应用程序集成

Last Updated: 2024-03-06

Red Hat build of Apache Camel K 1.10.5 将应用程序与 Kamelets 集成

配置连接器以简化应用程序集成

法律通告

摘要

Kamelets 提供了应用程序集成的替代方法。您可以配置 Red Hat build of Apache Camel K Kamelets (opinionated route templates)来创建连接，而不是直接使用 Camel 组件。

目录

前言

kamelets 是可重复使用的路由组件，隐藏了创建连接到外部系统的数据管道的复杂性。

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

第 1 章 KAMELETS 概述

kamelets 是高级连接器，可作为事件驱动的架构解决方案中的构建块。它们是您可以在 OpenShift 集群上安装的自定义资源，并在 Camel K 集成中使用。kamelets 加快您的开发工作。它们简化了如何连接数据源（发出事件）和数据接收器（消耗事件）。由于您可以配置 Kamelet 参数而不是编写代码，因此您不需要熟悉 Camel DSL 以使用 Kamelets。

您可以使用 Kamelets 相互连接应用程序和服务，或将服务直接连接到：

- Kafka 主题，如 连接到 Kamelets 的 Kafka 所述。
- Knative destinations （通道或代理），如 使用 Kamelets 连接到 Knative 中所述。
- 特定的 Camel URI，如 连接到显式 Camel URI 所述。

1.1. 关于 KAMELETS

kamelets 是路由组件（封装的代码），这些组件作为 Camel 集成中的连接器工作。您可以将 Kamelets 视为模板，用于定义从哪里使用数据（源）以及将数据发送到（接收器）的位置 - 允许您编译数据管道。kamelets 也可以过滤、掩码，并对数据执行简单的计算逻辑。

Kamelets 有三种不同类型的：

- **Source** - 生成数据的路由。您可以使用 source Kamelet 从组件检索数据。
- **sink** - 消耗数据的路由。您可以使用 sink Kamelet 将数据发送到组件。
- **action** - 对数据执行操作的路由。当数据从源 Kamelet 传递给接收器 Kamelet 时，您可以使用一个 action Kamelet 操作数据。

1.1.1. 为什么使用 Kamelets？

在 微服务和 事件驱动的构架解决方案中，Kamelets 可以充当发出事件和使用事件的接收器的源的构建块。

kamelets 提供抽象（隐藏连接到外部系统的复杂性）和可重复利用性（它们是重复使用代码的简单方法，并将其应用到不同的用例）。

以下是一些用例示例：

- 您希望应用程序消耗来自 Telegram 的事件，您可以使用 Kamelets 将 Telegram 源绑定到事件频道。之后，您可以将应用程序连接到该频道，以便它对这些事件做出反应。
- 您希望您的应用程序将 Salesforce 直接连接到 Slack。

kamelets 允许您和您的集成开发团队更高效。您可以重复使用 Kamelets，并与可以为其特定需求配置实例的团队成员共享它们。底层 Camel K 操作器执行硬工作：它编译、构建、软件包并部署 Kamelet 定义的集成。

1.1.2. 谁使用 Kamelets？

因为 Kamelets 允许您减少 Camel 集成所需的编码量，所以它们非常适合不熟悉 Camel DSL 的开发人员。kamelets 可以帮助非 Camel 开发者更容易地学习相关的知识。不需要您了解另一个框架或语言来获取 Camel 运行。

kamelets 对于希望将复杂 Camel 集成逻辑封装为可重复使用的 Kamelet 的经验丰富的 Camel 开发人员，然后与其他用户共享它。

1.1.3. 使用 Kamelets 的先决条件是什么？

要使用 Kamelets，您需要以下环境设置：

- 您可以使用正确访问级别访问 OpenShift 4.6 （或更新版本）集群、创建项目和安装操作器的功能，以及在本地系统上安装 OpenShift 和 Camel K CLI 工具。
- 您在命名空间或集群范围内安装了 Camel K Operator，如 安装 Camel K 所述
- 已安装 OpenShift 命令行(**oc**)接口工具。
- 另外，您使用 Camel K 插件安装 VS 代码或其他开发工具。基于 Camel 的工具扩展包括基于嵌入式 Kamelet Catalog 自动完成 Camel URI 的功能。如需更多信息，请参阅 Camel K *入门中的 Camel K 开发工具* 部分。
  注意： Visice Studio (VS) Code Tooling 扩展只是社区。

1.1.4. 如何使用 Kamelets？

使用 Kamelet 通常涉及两个组件： Kamelet 本身，它定义了可重复使用的路由片断，以及 Kamelet Binding，您可以在其中引用并绑定一个或多个 Kamelets。Kamelet Binding 是一个 OpenShift 资源 (**KameletBinding**)。

在 Kamelet Binding 资源中，您可以：

- 将 sink 或 source Kamelet 连接到事件频道： Kafka 主题或 Knative 目标（频道或代理）。
- 将接收器 Kamelet 直接连接到 Camel 统一资源标识符(URI)。您还可以将源 Kamelet 连接到 Camel URI，但连接 URI 和 sink Kamelet 是最常见的用例。
- 将接收器和源 Kamelet 直接连接到相互，而无需将事件频道用作中间层。
- 在同一个 Kamelet Binding 中多次引用同一 Kamelet。
- 添加 action Kamelets，以便在从源 Kamelet 传递给接收器 Kamelet 时操作数据。
- 定义错误处理策略，以指定在发送或接收事件数据时应该做什么。

在运行时，Camel K operator 使用 Kamelet Binding 生成并运行 Camel K 集成。

注： 虽然 Camel DSL 开发人员可以在 Camel K 集成中使用 Kamelets，但实施 Kamelets 的更简单方法是通过指定一个 Kamelet Binding 资源来构建高级别事件流。

1.2. 连接源和接收器

当您要连接两个或多个组件（外部应用程序或服务）时，使用 Kamelets。每个 Kamelet 基本上是一个带有配置属性的路由模板。您需要知道您要从（源）获取数据的组件，以及您要将数据发送到哪个组件（接收器）。您可以通过在 Kamelet Binding 中添加 Kamelets 来连接源和接收器组件，如图 1.1 所示。

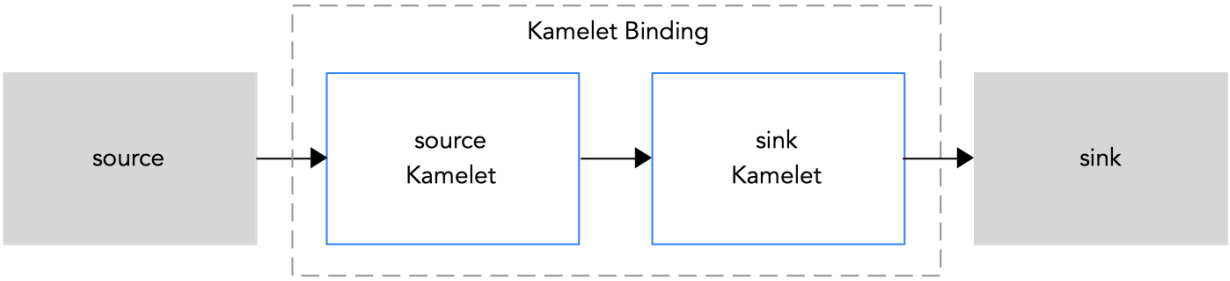


图 1.1: Kamelet Binding 源到 sink

以下是在 Kamelet Binding 中使用 Kamelets 的步骤概述：

1. 安装 Camel K operator。它包含一个 Kamelets 目录，作为 OpenShift 项目中的资源。
2. 创建一个 Kamelet Binding。确定您要在 Kamelet Binding 中连接的服务或应用程序。
3. 查看 Kamelet Catalog，以查找您要使用的源和接收器组件的 Kamelets。
4. 对于您要包含在 Kamelet Binding 中的每个 Kamelet，请确定您需要设置的配置属性。
5. 在 Kamelet Binding 代码中，添加对每个 Kamelet 的引用并配置必要的属性。
6. 将 Kamelet Binding 应用为 OpenShift 项目中的资源。

Camel K operator 使用 Kamelet Binding 生成并运行集成。

1.2.1. 安装 Camel K

您可以从 OperatorHub 在 OpenShift 集群上安装 Red Hat Integration - Camel K Operator。OperatorHub 由 OpenShift Container Platform Web 控制台获得，为集群管理员提供了一个界面，以发现和安装 Operator。

安装 Camel K Operator 后，您可以安装 Camel K CLI 工具以命令行访问所有 Camel K 功能。

先决条件

- 您可以访问具有正确访问级别的 OpenShift 4.6 （或更新版本）集群、创建项目和安装操作器的功能，以及在本地系统上安装 CLI 工具。

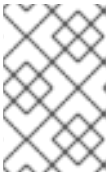
注意

> You do not need to create a pull secret when installing Camel K from the OpenShift OperatorHub. The Camel K Operator automatically reuses the OpenShift cluster-level authentication to pull the Camel K image from `registry.redhat.io`.

- 已安装 OpenShift CLI 工具(**oc**)，以便可以在命令行中与 OpenShift 集群交互。有关如何安装 OpenShift CLI 的详情，请参阅安装 OpenShift CLI。

流程

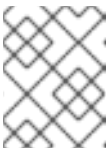
1. 在 OpenShift Container Platform Web 控制台中，使用具有集群管理员特权的帐户登录。
2. 创建新的 OpenShift 项目：
   a. 在左侧导航菜单中点 **Home** > **Project** > **Create Project**。
   b. 输入项目名称，如 **my-camel-k-project**，然后单击 **Create**。
3. 在左侧导航菜单中点 **Operators** > **OperatorHub**。
4. 在 **Filter by keyword** 文本框中，键入 **Camel K**，然后点 **Red Hat Integration - Camel K Operator** 卡。
5. 阅读有关操作器的信息，然后单击 **Install**。Operator 安装页面将打开。
6. 选择以下订阅设置：
   - **Update Channel** > **latest**
   - 选择以下 2 个选项：
     - **Installation Mode** > **A specific namespace on the cluster**> **my-camel-k-project**
     - **Installation Mode** > **All namespaces on the cluster (default)**> **Openshift operator**



注意

If you do not choose among the above two options, the system by default chooses a global namespace on the cluster then leading to openshift operator.

   - **Approval Strategy** > **Automatic**



注意

如果您的环境需要，可以使用 **Installation mode** > **All namespaces on the cluster** 和 **Approval Strategy** > **Manual** settings。

7. 点 **Install**，等待片刻，直到 Camel K Operator 准备就绪。
8. 下载并安装 Camel K CLI 工具：
   a. 在 OpenShift Web 控制台顶部的 **帮助菜单(?)** 中，选择 **Command line tools**。
   b. 向下滚动到 **kamel - Red Hat Integration - Camel K - Command Line Interface**部分。
   c. 单击链接以下载本地操作系统的二进制文件(Linux、Mac、Windows)。
   d. 在您的系统路径中解压并安装 CLI。
   e. 要验证您可以访问 Kamel K CLI，请打开一个命令窗口，然后输入以下内容：
   kamel --help

   此命令显示有关 Camel K CLI 命令的信息。

后续步骤

（可选） 指定 Camel K 资源限值

1.2.2. 查看 Kamelet Catalog

安装 Camel K operator 时，它包含一个可在 Camel K 集成中使用的 Kamelets 目录。

前提条件

您在工作空间或集群范围内安装了 Camel K 操作器，如 安装 Camel K 所述。

流程

查看使用 Camel K operator 安装的 Kamelets 列表：

1. 在终端窗口中，登录到您的 OpenShift 集群。
2. 查看可用 Kamelets 列表取决于 Camel K Operator 的安装方式（在一个特定命名空间或 cluster-mode 中）：
   - 如果在 cluster-mode 中安装了 Camel K Operator，请使用这个命令查看可用的 Kamelets：
     oc get kamelet -n openshift-operators
   - 如果 Camel K Operator 安装在特定命名空间中：
     a. 打开安装 Camel K Operator 的项目。
     oc project <camelk-project>

     例如，如果在 **my-camel-k-project** 项目中安装了 Camel K Operator：

     oc project my-camel-k-project

     b. 运行以下命令：
     oc get kamelets

注意

有关红帽支持的 Kamelets 列表，请参阅 Red Hat Integration 发行注记 。

另请参阅

在 Kamelet Catalog 中添加自定义 Kamelet

1.2.2.1. 在 Kamelet Catalog 中添加自定义 Kamelet

如果您在符合您要求的目录中没有看到 Kamelet，则 Camel DSL 开发人员可以创建自定义 Kamelet，如 *Apache Camel Kamelets Developers Guide* （community 文档所述。Kamelet 是以 **YAML** 格式编码的，惯例具有 **.kamelet.yaml** 文件扩展名。

先决条件

- Camel DSL 开发人员为您提供了一个自定义 Kamelet 文件。
- Kamelet 名称对于安装了 Camel K operator 的 OpenShift 命名空间必须是唯一的。

流程

要使自定义 Kamelet 作为 OpenShift 命名空间中的资源提供：

1. 将 Kamelet **YAML** 文件（例如 **custom-sink.kamelet.yaml**）下载到本地文件夹。
2. 登录您的 OpenShift 集群。
3. 在终端窗口中，打开安装 Camel K 操作器的项目，如 **my-camel-k-project** ：
   oc project my-camel-k-project
4. 运行 **oc apply** 命令将自定义 Kamelet 作为资源添加到命名空间中：
   oc apply -f <custom-kamelet-filename>

   例如，使用以下命令添加位于当前目录中的 **custom-sink.kamelet.yaml** 文件：

   oc apply -f custom-sink.kamelet.yaml
5. 要验证 Kamelet 是否作为资源可用，请使用以下命令查看当前命名空间中所有 Kamelets 的字母顺序列表，然后查找您的自定义 Kamelet ：
   oc get kamelets

1.2.2.2. 确定 Kamelet 的配置参数

在 Kamelet Binding 中，当添加对 Kamelet 的引用时，您可以指定 Kamelet 的名称，并配置 Kamelet 的参数。

前提条件

- 您在工作空间或集群范围内安装了 Camel K Operator。

流程

要确定 Kamelet 的名称和参数：

1. 在终端窗口中，登录到您的 OpenShift 集群。
2. 打开 Kamelet 的 YAML 文件：
   oc describe kamelets/<kamelet-name>

   例如，要查看 **ftp-source** Kamelet 的代码，如果 Camel K operator 在当前命名空间中安装了，请使用以下命令：

   oc describe kamelets/ftp-source

   如果 Camel K Operator 安装在 cluster-mode 中，使用以下命令：

   oc describe -n openshift-operators kamelets/ftp-source
3. 在 YAML 文件中，滚动到 **spec.definition** 部分（使用 JSON-schema 格式编写），以查看 Kamelet 的属性列表。在部分末尾，required 字段列出了在引用 Kamelet 时必须配置的属性。例如，以下代码是 **ftp-source** Kamelet 的 **spec.definition** 部分的摘录。本节详细介绍了 Kamelet 的配置属性。此 Kamelet 的必要属性是 **connectionHost**,**connectionPort**,**username**,**password**, 和 **directoryName** ：

```
spec:
  definition:
    title: "FTP Source"
```

```
  description: |-
    Receive data from an FTP Server.
  required:
    - connectionHost
    - connectionPort
    - username
    - password
    - directoryName
  type: object
  properties:
    connectionHost:
      title: Connection Host
      description: Hostname of the FTP server
      type: string
    connectionPort:
      title: Connection Port
      description: Port of the FTP server
      type: string
      default: 21
    username:
      title: Username
      description: The username to access the FTP server
      type: string
    password:
      title: Password
      description: The password to access the FTP server
      type: string
      format: password
      x-descriptors:
      - urn:alm:descriptor:com.tectonic.ui:password
    directoryName:
      title: Directory Name
      description: The starting directory
      type: string
    passiveMode:
      title: Passive Mode
      description: Sets passive mode connection
      type: boolean
      default: false
      x-descriptors:
      - 'urn:alm:descriptor:com.tectonic.ui:checkbox'
    recursive:
      title: Recursive
      description: If a directory, will look for files in all the sub-directories as well.
      type: boolean
      default: false
      x-descriptors:
      - 'urn:alm:descriptor:com.tectonic.ui:checkbox'
    idempotent:
      title: Idempotency
      description: Skip already processed files.
      type: boolean
      default: true
      x-descriptors:
      - 'urn:alm:descriptor:com.tectonic.ui:checkbox'
```

另请参阅

配置 Kamelet 实例参数

1.2.3. 在 Kamelet Binding 中连接源和接收器组件

在 Kamelet Binding 中，您将连接源和接收器组件。

此流程中的示例使用以下 Kamelets，如图 1.2 所示：

- 示例源 Kamelet 称为 **coffee-source**。这个简单 Kamelet 从网站目录中检索有关 coffee 类型的随机生成的数据。它有一个参数(**句点 - 整数值**)，用于决定检索 coffee 数据的频率（以秒为单位）。不需要该参数，因为有一个默认值(1000 秒)。
- sink Kamelet 示例名为 **log-sink**。它检索数据并将其输出到日志文件中。**log-sink** Kamelet 在 Kamelet Catalog 中提供。

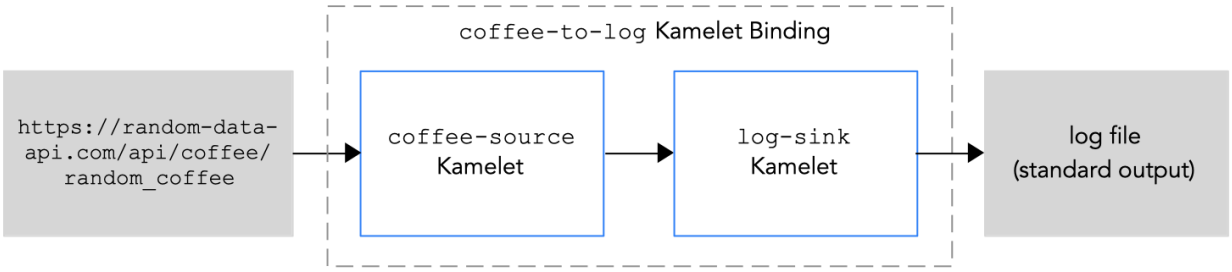


图 1.2: Kamelet Binding 示例

先决条件

- 您知道如何创建和编辑 Camel K 集成。
- **Red Hat Integration - Camel K**Operator 已安装在 OpenShift 命名空间或集群中，并下载 Red Hat Integration Camel K CLI 工具，如 安装 Camel K 所述。
- 您知道您要添加到 Camel K 集成中的 Kamelets 及其所需的实例参数。
- 您要使用的 Kamelets 在 Kamelet Catalog 中提供。
  在本例中，**log-sink** Kamelet 在 Kamelet Catalog 中提供。如果要在本示例中使用 source Kamelet，将 **coffee-source** 代码 复制并保持到一个名为 **coffee-source.kamelet.yaml** 的本地文件中，然后运行以下命令将其添加到 Kamelet 目录中：

  oc apply -f coffee-source.kamelet.yaml

流程

1. 登录您的 OpenShift 集群。
2. 打开安装 Camel K Operator 的工作项目。如果您在 cluster-mode 中安装了 Camel K operator，则集群的任何项目都可以使用它。
   例如，要打开名为 **my-camel-k-project** 的现有项目：

   oc project my-camel-k-project

3. 使用以下选项之一创建一个新的 Kamelet Binding：

- 使用 **kamel bind** 命令创建并运行 Kamelet Binding （对于命令行定义的简单 Kamelet Bindings 很有用）
- 创建一个 YAML 文件来定义 Kamelet Binding，然后使用 **oc apply** 命令运行它（当 Kamelet Binding 配置更为复杂时，此选项很有用）。

使用 kamel bind 命令创建一个新的 Kamelet Binding

使用以下 **kamel bind** 语法指定 source 和 sink Kamelets 和任何配置参数：

```
kamel bind <kamelet-source> -p “<property>=<property-value>” <kamelet-sink> -p “<property>=<property-value>”
```

例如：

```
kamel bind coffee-source -p “source.period=5000” log-sink -p "sink.showStreams=true"
```

Camel K operator 生成 **KameletBinding** 资源，并运行对应的 Camel K 集成。

使用 YAML 文件创建一个新的 Kamelet Binding

a. 在您选择的编辑器中，创建一个具有以下结构的 YAML 文件：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name:
spec:
  source:
  sink:
```

b. 为 Kamelet Binding 添加名称。
在本例中，名称是 **coffee-to-log**，因为绑定将 **coffee-source** Kamelet 连接到 **log-sink** Kamelet。

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffee-to-log
spec:
  source:
  sink:
```

c. 指定源 Kamelet （如 **coffee-source**）并为 Kamelet 配置任何参数。
注： 在本例中，参数在 Kamelet Binding 的 YAML 文件中定义。另外，您可以在属性文件、ConfigMap 或 Secret 中配置 Kamelet 的参数，如 配置 Kamelet 实例参数 中所述。

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffee-to-log
spec:
  source:
```

```
    ref
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: coffee-source
    properties:
      period: 5000
  sink:
```

d. 指定 sink Kamelet （如 **log-sink**）并为 Kamelet 配置任何参数。使用 **log-sink** Kamelet 的可选 **showStreams** 参数来显示消息正文。

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffee-to-log
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: coffee-source
    properties:
      period: 5000
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: log-sink
    properties:
      showStreams: true
```

e. 保存 YAML 文件（例如，**coffee-to-log.yaml**）。

f. 将 **KameletBinding** 作为资源添加到 OpenShift 命名空间中：
oc apply -f <kamelet-binding>.yaml

例如：

oc apply -f coffee-to-log.yaml

Camel K operator 使用 **KameletBinding** 资源生成并运行 Camel K 集成。

4. 查看 Kamelet Binding 的状态：
oc get kameletbindings

5. 查看对应集成的状态： **oc get integrations**

6. 查看输出：

- 要从命令行查看日志，请打开终端窗口，然后输入以下命令：
kamel log <integration-name>

例如，如果集成名称为 **coffee-to-log**，请使用以下命令：

kamel log coffee-to-log

- 查看 OpenShift Web 控制台的日志：
  a. 选择 **Workloads** > **Pods**。
  b. 单击 Camel K 集成 pod 的名称，然后单击 **Logs**。
  您应该看到类似以下示例的 coffee 事件列表：

> INFO [log-sink-E80C5C904418150-0000000000000001] (Camel (camel-1) thread #0 - timer://tick) {"id":7259,"uid":"a4ecb7c2-05b8-4a49-b0d2-d1e8db5bc5e2","blend_name":"Postmodern Symphony","origin":"Huila, Colombia","variety":"Kona","notes":"delicate, chewy, black currant, red apple, star fruit","intensifier":"balanced"}

7. 要停止集成，请删除 Kamelet Binding：
   oc delete kameletbindings/<kameletbinding-name>

   例如：

   oc delete kameletbindings/coffee-to-log

后续步骤

（可选）：

- 添加 action Kamelets 作为中间步骤，如 将操作添加到 Kamelet Binding 所述。
- 在 Kamelet Binding 中添加错误处理，如将 错误处理器策略添加到 Kamelet Binding 所述。

1.2.4. 配置 Kamelet 实例参数

引用 Kamelet 时，有定义 Kamelet 的实例参数的以下选项：

- 直接在指定 Kamelet URI 的 Kamelet Binding 中。在以下示例中，由 Telegram BotFather 提供的 bot 授权令牌为 **123456**:
  from("kamelet:telegram-source?authorizationToken=123456")
- 全局配置 Kamelet 属性（因此您不必使用以下格式在 URI 中提供值）：
  "camel.kamelet.<kamelet-name>.<property-name>=<value>"

  如 使用 Camel K 开发和管理集成 中的 配置 *Camel K* 集成 一章中所述，您可以通过以下方式配置 Kamelet 参数：
  - 将它们定义为属性
  - 在属性文件中定义它们
  - 在 OpenShift ConfigMap 或 Secret 中定义

另请参阅

确定 kamelet 的配置参数

1.2.5. 连接到事件频道

Kamelets 的最常见用例是使用 Kamelet Binding 将它们连接到事件的频道： Kafka 主题或 Knative destination （频道或代理）。这样做的好处是，数据源和接收器相互独立，并"不知道"。这种分离允许您

的企业方案中的组件单独开发和管理。如果您的数据 sink 和源作为您的业务场景的一部分，则更重要的是分离各种组件。例如，如果需要关闭事件 sink，事件源不会受到影响。另外，如果其他 sink 使用相同的源，则它们不会受到影响。

图 1.3 演示了将源和接收器 Kamelets 连接到事件频道的流程。

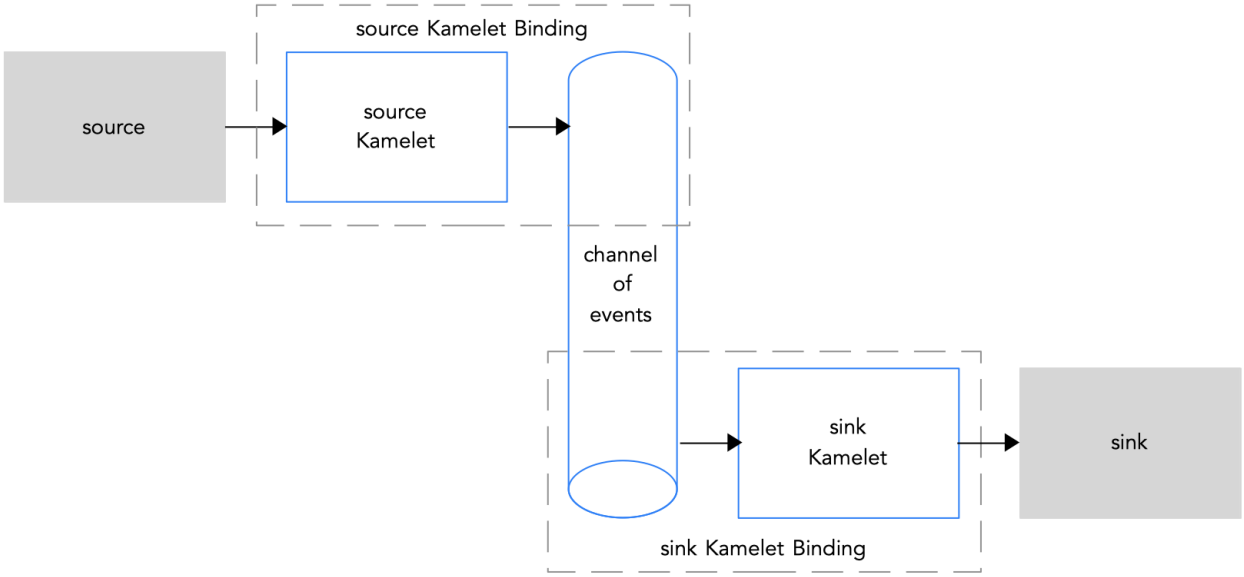


图 1.3：将源和接收器 Kamelets 连接到事件频道

如果您使用 Apache Kafka 流处理框架，请参阅 连接到 Kafka 主题，请参阅使用 Kamelets 连接到 Kafka。

如果使用 Knative 无服务器框架，请参阅 连接到 Knative 目标（频道或代理）的详情，请参阅使用 Kamelets 连接到 Knative。

1.2.6. 连接到显式 Camel URI

您可以创建一个 Kamelet Binding，其中 Kamelet 将事件发送到或从显式 Camel URI 接收事件。通常，您可以将源 Kamelet 绑定到一个可以接收事件（即，在 Kamelet Binding 中将 URI 指定为接收器）的 URI。接收事件的 Camel URI 示例是 HTTP 或 HTTPS 端点。

也可以将 URI 指定为 Kamelet Binding 中的源，但可能并不常见。发送事件的 Camel URI 示例包括计时器、邮件或 FTP 端点。

要将 Kamelet 连接到 Camel URI，请按照 Kamelet Binding 中的 Connecting source 和 sink 组件 以及 **sink.uri** 字段而不是 Kamelet 中的步骤，指定一个显式 Camel URI。

在以下示例中，sink 的 URI 是一个虚构 URI (https://mycompany.com/event-service)：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffee-to-event-service
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
```

```
    name: coffee-source
  properties:
    period: 5000
sink:
  uri: https://mycompany.com/event-service
```

1.3. 将操作应用到连接中的数据

如果要对 Kamelet 和事件频道之间传递的数据执行操作，请使用 action Kamelets 作为 Kamelet Binding 中的中间步骤。例如，您可以使用一个 action Kamelet 来序列化或反序列化数据，过滤数据或插入一个字段或消息标头。

操作操作（如过滤或添加字段）只适用于 JSON 数据（即，当 **Content-Type** 标头设置为 **application/json**时）。如果事件数据使用 JSON 以外的格式（如 Avro 或 Protocol Buffers），则在处理操作前需要一个额外的反序列化步骤来进行格式转换（例如，**protobuf-deserialize-action** 或 **avro-deserialize-action** Kamelet），并在处理操作后需要另外一个额外的序列化操作（例如 **protobuf-serialize-action** or **avro-serialize-action** Kamelet）。有关在连接中转换数据格式的更多信息，请参阅数据转换 Kamelets。

action Kamelets 包括：

- 数据过滤 Kamelets
- 数据转换 Kamelets
- 数据转换 Kamelets

1.3.1. 在 Kamelet Binding 中添加操作

要实现一个操作 Kamelet，在 Kamelet Binding 文件的 **spec** 部分中，在 source 和 sink 部分添加一个 **steps** 部分。

先决条件

- 您已创建了 Kamelet Binding，如 Kamelet Binding 中的 Connecting source 和 sink 组件 所述。
- 您知道您要添加到 Kamelet Binding 和 action Kamelet 的必要参数的 Kamelet 操作。
  对于此流程中的示例，**predicate-filter-action** Kamelet 的参数 **是一个字符串** 类型 expression，它提供 JSON Path Expression，它过滤 coffee data to only log coffees with a "deep" taste intensity。请注意，**predicate-filter-action** Kamelet 要求您在 Kamelet Binding 中设置 Builder 特征配置属性。

  这个示例还包括 deserialize 和 serialize 操作，在本例中是可选的，因为事件数据格式为 JSON。

流程

1. 在编辑器中打开 **KameletBinding** 文件。
   例如，以下是 **coffee-to-log.yaml** 文件的内容：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffee-to-log
spec:
  source:
```

```
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: coffee-source
    properties:
      period: 5000
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: log-sink
```

2. 在 **source** 部分上方添加一个 **integration** 部分，并提供以下 Builder 特征配置属性（如 **predicate-filter-action** Kamelet 需要）：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffee-to-log
spec:
  integration:
        traits:
          builder:
            configuration:
              properties:
                - "quarkus.arc.unremovable-
types=com.fasterxml.jackson.databind.ObjectMapper"
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: coffee-source
    properties:
      period: 5000
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: log-sink
```

3. 在 **source** 和 **sink** 部分之间添加一个 **steps** 部分，并定义 action Kamelet。例如：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffee-to-log
  spec:
    integration:
          traits:
            builder:
              configuration:
                properties:
                  - "quarkus.arc.unremovable-
types=com.fasterxml.jackson.databind.ObjectMapper"
  source:
```

```
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: coffee-source
    properties:
      period: 5000
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: predicate-filter-action
    properties:
      expression: "@.intensifier =~ /.*deep/"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-serialize-action
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: log-sink
```

4. 保存您的更改。

5. 使用 **oc apply** 命令更新 **KameletBinding** 资源，例如：
   oc apply -f coffee-to-log.yaml

   Camel K operator 会重新生成并运行 CamelK 集成，该集成会根据更新的 **KameletBinding** 资源生成。

6. 查看 Kamelet Binding 的状态：
   oc get kameletbindings

7. 查看其对应集成的状态：
   oc get integrations

8. 查看集成的日志文件输出：
   kamel logs <integration-name>

   例如，如果集成名称为 **coffee-to-log** ：

   kamel logs coffee-to-log

9. 要停止集成，请删除 Kamelet Binding：
   oc delete kameletbindings/<kameletbinding-name>

   例如：

   oc delete kameletbindings/coffee-to-log

1.3.2. 操作 kamelets

- 第 1.3.2.1 节 “数据过滤 Kamelets”
- 第 1.3.2.2 节 “数据转换 Kamelets”
- 第 1.3.2.3 节 “数据转换 Kamelets”

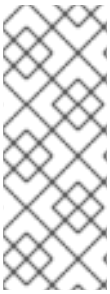
1.3.2.1. 数据过滤 Kamelets

您可以过滤源和接收器组件间传递的数据，例如防止泄漏敏感数据或避免生成不必要的网络收费。

您可以根据以下条件过滤数据：

- **Kafka 主题名称** - 通过配置主题名称 Matches Filter Kamelet (**topic-name-matches-filter-action**)来过滤带有与给定 Java 正则表达式匹配的 Kafka 主题的事件。如需更多信息，请参阅 过滤特定 Kafka 主题的事件数据。
- header **键 - 通过配置 Header** Filter Action Kamelet (**has-header-filter-action**)来过滤具有给定消息标头的事件。
- null 值 - 通过配置 Tombstone Filter Action Kamelet (**is-tombstone-filter-action**)，过滤器 tombstone 事件（带有 null payload 的事件）。
- predicate - 通过配置 **Predicate** Filter Action Kamelet (**predicate-filter-action**)，根据给定的 JSON 路径表达式过滤事件。**predicate-filter-action** Kamelet 要求您在 Kamelet Binding 中设置以下 Builder 特征 配置属性：

```
spec:
  integration:
    traits:
      builder:
        configuration:
          properties:
           - "quarkus.arc.unremovable-types=com.fasterxml.
                jackson.databind.ObjectMapper"
```



注意

数据过滤 Kamelets 使用 JSON 数据（即，当 Content-Type 标头设置为 application/json 时）可以正常工作。如果事件数据使用 JSON 以外的格式，则在处理操作前需要一个额外的反序列化步骤来进行格式转换（例如，**protobuf-deserialize-action** 或 **avro-deserialize-action**），并在处理操作后需要另外一个额外的序列化操作（例如 **protobuf-serialize-action** or **avro-serialize-action**）。有关在连接中转换数据格式的更多信息，请参阅数据转换 Kamelets。

1.3.2.2. 数据转换 Kamelets

使用以下数据转换 Kamelets，您可以序列化和反序列化源组件之间传递的数据格式。数据转换适用于事件数据的有效负载（不是密钥或标头）。

- **avro** - 为 Apache Hadoop 提供数据序列化和数据交换服务的开源项目。
  - avro Deserialize Action Kamelet (**avro-deserialize-action**)
  - avro Serialize Action Kamelet (**avro-serialize-action**)

- **协议缓冲器** - 由内部使用它的 Google 发明的高性能、紧凑的二进制线格式，以便它们可以与其内部网络服务通信。
  - protobuf Deserialize Action Kamelet (**protobuf-deserialize-action**)
  - protobuf Serialize Action Kamelet (**protobuf-serialize-action**)
- **JSON** (JavaScript 对象表示法)- 基于 JavaScript 编程语言的子集的数据交换格式。JSON 是一个完全独立于语言的文本格式。
  - JSON Deserialize Action Kamelet (**json-deserialize-action**)
  - JSON Serialize Action Kamelet (**json-serialize-action**)

注意

您必须在 Avro 和 Protobuf serialize/deserialize Kamelets 中指定 schema （使用 JSON 格式）。对于 JSON serialize/deserialize Kamelets，您不需要这样做。

1.3.2.3. 数据转换 Kamelets

通过以下数据转换 Kamelets，您可以在源和 sink 组件间传递的数据上执行简单的操作：

- extract **Field** - 使用 **extract-field-action** Kamelet 从数据的正文中拉取一个字段，并使用提取的字段替换整个数据正文。
- **Hoist 字段** - 使用 **hoist-field-action** Kamelet 将数据正文嵌套为一个字段。
- insert **Header** - 使用 **insert-header-action** Kamelet 使用静态数据或记录元数据添加标头字段。
- insert **Field** - 使用 **insert-field-action** Kamelet 使用静态数据或记录元数据添加字段值。
- **Mask Field** - 使用 **mask-field-action** Kamelet 为字段类型使用一个有效的 null 值（如 0 或空字符串）或一个给定的值（需要是一个非空的字符串或一个数字值）替换字段的值。
  例如，如果要从相关数据库捕获到 Kafka 的数据，数据包含受保护的(PCI / PII)信息，如果您的 Kafka 集群还没有认证，则必须屏蔽受保护的信息。
- **替换 Field** - 使用 **replace-field-action** Kamelet 过滤或重命名字段。您可以指定要重命名的字段、禁用（排除）或启用（包括）。
- **Value To Key** -(for Kafka)使用 **value-to-key-action** Kamelet 将记录键替换为来自有效负载中字段子集的新键。您可以将 event 键设置为基于事件信息的值，然后再将数据写入 Kafka。例如，当从数据库表读取记录时，您可以根据客户 ID 在 Kafka 中对记录进行分区。

1.4. 处理连接中的错误

要指定在发送或接收事件数据时运行集成遇到失败的集成，您可以选择在 Kamelet Binding 中添加以下错误处理策略之一：

- 没有错误处理程序 - 忽略集成中的所有失败。
- 日志错误处理程序 - 将日志消息发送到标准输出。
- 死信频道错误处理程序 - 将失败的事件重定向到另一个组件，如第三方 URI、队列或其他 Kamelet，它们可以使用失败的事件执行某些逻辑。还支持尝试重新设计消息交换的次数，然后再将其发送到死信端点。

- bean 错误处理程序 - 指定使用自定义 bean 来处理错误。
- ref error handler - 指定使用 bean 来处理错误。bean 必须在运行时在 Camel registry 中提供。

1.4.1. 在 Kamelet Binding 中添加错误处理器策略

要在源和接收器连接之间发送或接收事件数据时处理错误，请在 Kamelet Binding 中添加错误处理器策略。

先决条件

- 您知道您要使用的错误处理程序策略类型。
- 您有一个现有的 **KameletBinding** YAML 文件。

流程

在 Kamelet Binding 中实施错误处理：

1. 在编辑器中打开 **KameletBinding** YAML 文件。
2. 在 **sink** 定义后，在 **spec** 部分添加一个错误处理器部分：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: example-kamelet-binding
spec:
  source:
   ...
  sink:
   ...
  errorHandler: ...
```

例如，在 **coffee-to-log** Kamelet Binding 中，通过添加日志错误处理程序来指定将错误发送到日志文件的最大次数：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffee-to-log
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: coffee-source
    properties:
      period: 5000
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: log-sink
  errorHandler:
```

```
    log:
      parameters:
        maximumRedeliveries: 3
```

3. 保存您的文件。

1.4.2. 错误处理程序

- 第 1.4.2.1 节 “没有错误处理程序”
- 第 1.4.2.2 节 “日志错误处理程序”
- 第 1.4.2.3 节 “死信频道错误处理程序”
- 第 1.4.2.4 节 “Bean 错误处理程序”
- 第 1.4.2.5 节 “ref error handler”

1.4.2.1. 没有错误处理程序

如果要忽略集成中的任何失败，您可以在 Kamelet Binding 中包含 **errorHandler** 部分，或者将其设置为 **none**，如下例所示：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: my-kamelet-binding
spec:
  source:
...
  sink:
...
  errorHandler:
    none:
```

1.4.2.2. 日志错误处理程序

处理任何失败的默认行为是将日志消息发送到标准输出。另外，您可以使用日志错误处理器来指定其他行为，如重新传送或延迟策略，如下例所示：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: my-kamelet-binding
spec:
  source:
...
  sink:
...
  errorHandler:
    log:
      parameters:
        maximumRedeliveries: 3
        redeliveryDelay: 2000
```

1.4.2.3. 死信频道错误处理程序

Dead Letter Channel 允许您将任何失败的事件重定向到任何其他组件（如第三方 URI、队列或其他 Kamelet），它可以定义如何处理失败的事件，如下例所示：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: my-kamelet-binding
spec:
  source:
  ...
  sink:
  ...
  errorHandler:
    dead-letter-channel:
      endpoint:
        ref: 1
          kind: Kamelet
          apiVersion: camel.apache.org/v1alpha1
          name: error-handler
        properties: 2
          message: "ERROR!"
          ...
      parameters: 3
        maximumRedeliveries: 1
```

1. 对于 **端点**，您可以使用 **ref** 或 **uri**。根据 **kind**, **apiVersion** 和 **name** 的值的 Camel K operator interprets **ref**。您可以使用任何 Kamelet、Kafka Topic 频道或 Knative 目的地。
2. 属于端点的**属性**（本例中为一个名为 **error-handler** 的 Kamelet）。
3. 属于 dead-letter-channel 错误处理程序类型 **的参数**。

1.4.2.4. Bean 错误处理程序

通过 Bean 错误处理程序，您可以通过提供处理错误的自定义 bean 来扩展 Error Handler 的功能。对于 **类型**，指定 **ErrorHandlerBuilder** 的完全限定名称。对于 **属性**，请配置您在 **类型** 中指定的 **ErrorHandlerBuilder** 预期的属性。

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: my-kamelet-binding
spec:
  source:
...
  sink:
...
  errorHandler:
    bean:
      type: "org.apache.camel.builder.DeadLetterChannelBuilder"
      properties:
        deadLetterUri: log:error
```

1.4.2.5. ref error handler

使用 Ref 错误处理程序，您可以使用您希望在运行时在 Camel registry 中提供的任何 bean。在以下示例中，**my-custom-builder** 是要在运行时查看的 bean 的名称。

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: my-kamelet-binding
spec:
  source:
   ...
  sink:
   ...
  errorHandler:
    ref: my-custom-builder
```

另请参阅：

- Camel K 错误处理
- 各种错误处理程序支持的功能

第 2 章 使用 KAMELETS 连接到 KAFKA

Apache Kafka 是一个开源、分布式、发布订阅的消息系统，用于创建容错、实时数据源。Kafka 为大量消费者（外部连接）快速存储和复制数据。

Kafka 可帮助您构建处理流事件的解决方案。分布式、事件驱动的架构需要捕获、沟通和帮助处理事件的"后退"。Kafka 可以充当将数据源和事件连接到应用程序的通信主干。

您可以使用 Kamelets 来配置 Kafka 和外部资源之间的通信。kamelets 允许您配置数据如何在 Kafka 流处理框架中从一个端点移动到另一个端点，而无需编写代码。kamelets 是通过指定参数值来配置的路由模板。

例如，Kafka 以二进制形式存储数据。您可以使用 Kamelets 来序列化和反序列化数据以发送到，并从外部连接接收。使用 Kamelets 时，您可以验证模式并更改数据，如添加到其中、过滤或屏蔽数据。kamelets 也可以处理和处理错误。

2.1. 使用 KAMELETS 连接到 KAFKA 概述

如果使用 Apache Kafka 流处理框架，您可以使用 Kamelets 将服务和应用程序连接到 Kafka 主题。Kamelet Catalog 提供以下 Kamelets，专门用于进行到 Kafka 主题的连接：

- **kafka-sink** - 将事件从数据制作者移到 Kafka 主题。在 Kamelet Binding 中，将 **kafka-sink** Kamelet 指定为接收器。
- **kafka-source** - 将事件从 Kafka 主题移到数据消费者。在 Kamelet Binding 中，指定 **kafka-source** Kamelet 作为源。

图 2.1 演示了将源和接收器 Kamelets 连接到 Kafka 主题的流程。

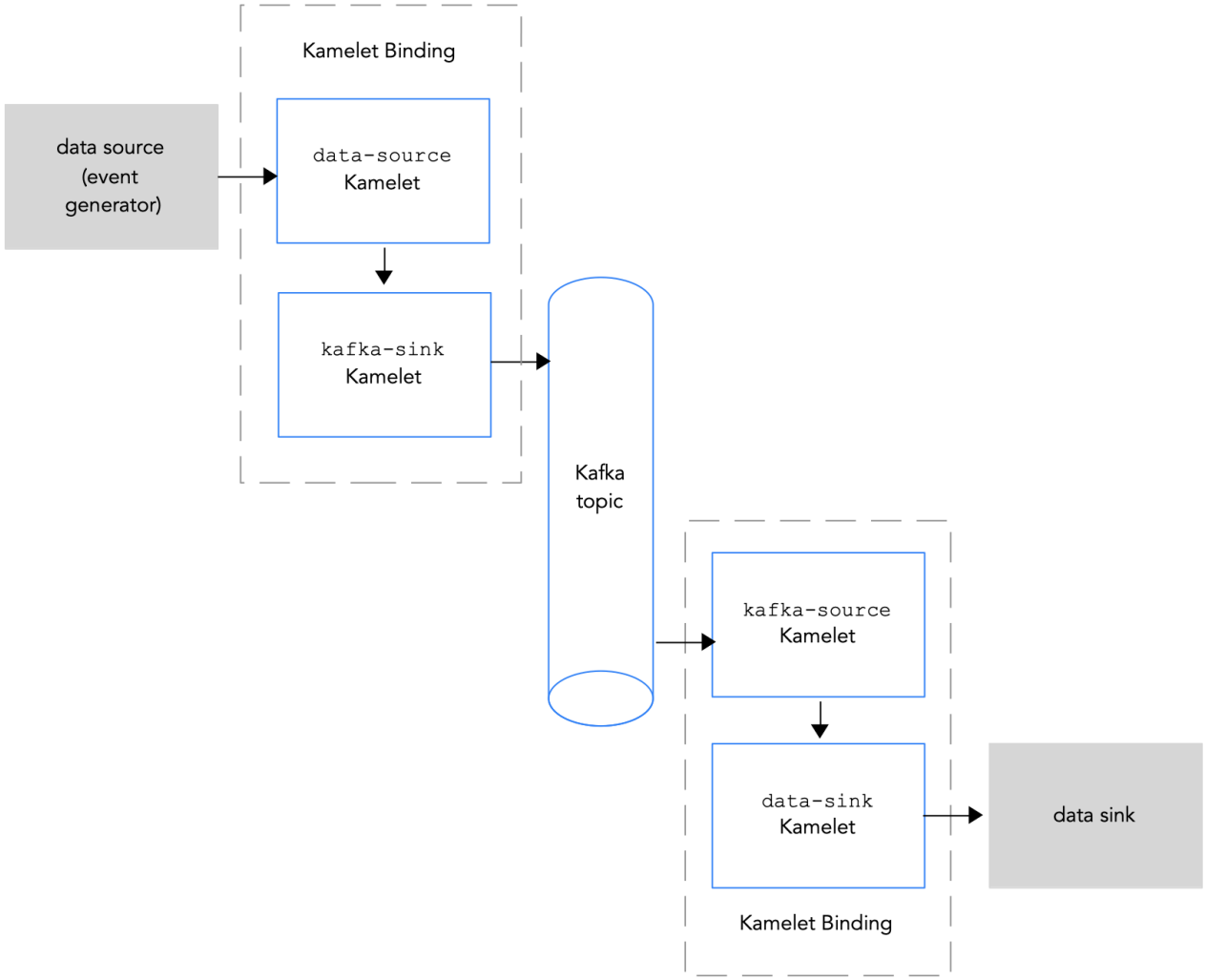


图 2.1：带有 Kamelets 和 Kafka 主题的数据流

以下是使用 Kamelets 和 Kamelet Bindings 将应用程序和服务连接到 Kafka 主题的基本步骤概述：

1. 设置 Kafka：
   a. 安装所需的 OpenShift 操作器。
      - 对于 OpenShift Streams for Apache Kafka，安装 Camel K operator、Camel K CLI 和 Red Hat OpenShift Application Services (RHOAS) CLI。
      - 对于 AMQ 流，请安装 Camel K 和 AMQ 流操作器和 Camel K CLI。
   b. 创建 Kafka 实例。Kafka 实例作为消息代理运行。代理包含主题，编配存储并传递信息。
   c. 创建 Kafka 主题。主题提供数据存储的目的地。
   d. 获取 Kafka 身份验证凭证。
2. 确定您要连接到 Kafka 主题的服务或应用程序。
3. 查看 Kamelet Catalog，以查找您要添加到集成的源和接收器组件的 Kamelets。另外，确定您要使用的每个 Kamelet 所需的配置参数。
4. 创建 Kamelet Bindings：

- 创建一个 Kamelet Binding，它将一个数据源（一个生成数据的组件）连接到 Kafka 主题（使用 **kafka-sink** Kamelet）。
- 创建一个 Kamelet Binding，它将 kafka 主题（使用 **kafka-source** Kamelet）连接到数据 sink（消耗数据的组件）。

5. （可选）通过在 Kamelet Binding 中添加一个或多个操作 Kamelets 作为中介步骤来操作 Kafka 主题和数据源或 sink 间的数据。
6. （可选）定义如何在 Kamelet Binding 中处理错误。
7. 将 Kamelet Bindings 作为资源应用到项目。
Camel K operator 为每个 Kamelet Binding 生成一个单独的 Camel K 集成。

2.2. 设置 KAFKA

要设置 Kafka，您必须：

1. 安装所需的 OpenShift operator
2. 创建 Kafka 实例
3. 创建 Kafka 主题

使用以下提到的红帽产品来设置 Kafka：

- **Red Hat Advanced 消息队列(AMQ)流** - 自我管理的 Apache Kafka 产品。AMQ Streams 基于开源 Strimzi，并作为 Red Hat Integration 的一部分包含在内。AMQ Streams 是一个分布式、可扩展的流平台，它基于 Apache Kafka，其中包括发布/订阅消息传递代理。Kafka Connect 提供了一个框架，用于将基于 Kafka 的系统与外部系统集成。使用 Kafka Connect，您可以配置源和接收器连接器，将来自外部系统的数据流传输到 Kafka 代理。

2.2.1. 使用 AMQ 流设置 Kafka

AMQ Streams 简化了在 OpenShift 集群中运行 Apache Kafka 的过程。

2.2.1.1. 为 AMQ Streams 准备 OpenShift 集群

要使用 Camel K 或 Kamelets 和 Red Hat AMQ Streams，您必须安装以下 operator 和工具：

- **Red Hat Integration - AMQ Streams**operator - 管理 Openshift Cluster 和 AMQ Streams for Apache Kafka 实例之间的通信。
- **Red Hat Integration - Camel K**operator - 安装和管理 Camel K - 在 OpenShift 上原生运行的轻量级集成框架。
- **Camel K CLI** 工具 - 允许您访问所有 Camel K 功能。

先决条件

- 熟悉 Apache Kafka 概念。
- 您可以使用正确访问级别访问 OpenShift 4.6 （或更新版本）集群、创建项目和安装操作器的功能，以及在本地系统上安装 OpenShift 和 Camel K CLI。
- 已安装 OpenShift CLI 工具(**oc**)，以便可以在命令行中与 OpenShift 集群交互。

流程

使用 AMQ Streams 设置 Kafka：

1. 登录您的 OpenShift 集群的 Web 控制台。
2. 创建或打开您要在其中创建集成的项目，如 **my-camel-k-kafka**。
3. 安装 Camel K operator 和 Camel K CLI，如 安装 Camel K 所述。
4. 安装 AMQ Streams Operator：
   a. 从任何项目中，选择 **Operators** > **OperatorHub**。
   b. 在 **Filter by Keyword** 字段中，键入 **AMQ Streams**。
   c. 点 **Red Hat Integration - AMQ Streams**卡，然后点 **Install**。
   此时会打开 **Install Operator** 页面。
   d. 接受默认值，然后点 **Install**。
5. 选择 **Operators** > **Installed Operators** 来验证是否安装了 Camel K 和 AMQ Streams operator。

后续步骤

使用 AMQ Streams 设置 Kafka 主题

2.2.1.2. 使用 AMQ Streams 设置 Kafka 主题

Kafka 主题提供在 Kafka 实例中存储数据的目标。在向它发送数据前，您必须设置 Kafka 主题。

先决条件

- 您可以访问 OpenShift 集群。
- 已安装 **Red Hat Integration - Camel K**和 **Red Hat Integration - AMQ Streams**operator，如 准备 OpenShift 集群 中所述。
- 已安装 OpenShift CLI (**oc**)和 Camel K CLI (**kamel**)。

流程

使用 AMQ Streams 设置 Kafka 主题：

1. 登录您的 OpenShift 集群的 Web 控制台。
2. 选择 **Projects**，然后单击在其中安装 **Red Hat Integration - AMQ Streams**operator 的项目。例如，单击 **my-camel-k-kafka** 项目。
3. 选择 **Operators** > **Installed Operators**，然后点 **Red Hat Integration - AMQ Streams**。
4. 创建 Kafka 集群：
   a. 在 **Kafka** 下，点 **Create instance**。
   b. 为集群输入一个名称，如 **kafka-test**。
   c. 接受其他默认值，然后单击 **Create**。

创建 Kafka 实例的过程可能需要几分钟时间来完成。

当状态就绪时，继续下一步。

5. 创建 Kafka 主题：

   a. 选择 **Operators** > **Installed Operators**，然后点 **Red Hat Integration - AMQ Streams**。

   b. 在 **Kafka 主题** 下，点 **Create Kafka Topic**。

   c. 输入主题的名称，如 **test-topic**。

   d. 接受其他默认值，然后单击 **Create**。

2.2.2. 使用 OpenShift 流设置 Kafka

要使用 OpenShift Streams for Apache Kafka，您必须登录到您的红帽帐户。

2.2.2.1. 为 OpenShift Streams 准备 OpenShift 集群

要使用受管云服务，您必须安装以下 operator 和工具：

- **OpenShift Application Services (RHOAS) CLI** - 允许您从终端管理应用程序服务。
- **Red Hat Integration - Camel K**operator 安装并管理 Camel K - 在 OpenShift 上原生运行的轻量级集成框架。
- **Camel K CLI 工具** - 允许您访问所有 Camel K 功能。

先决条件

- 熟悉 Apache Kafka 概念。
- 您可以使用正确访问级别访问 OpenShift 4.6 （或更新版本）集群、创建项目和安装操作器的功能，以及在本地系统上安装 OpenShift 和 Apache Camel K CLI。
- 已安装 OpenShift CLI 工具(**oc**)，以便可以在命令行中与 OpenShift 集群交互。

流程

1. 使用集群管理员帐户登录 OpenShift Web 控制台。
2. 为您的 Camel K 或 Kamelets 应用程序创建 OpenShift 项目。

   a. 选择 **Home** > **Projects**。

   b. 点击 **Create Project**。

   c. 键入项目的名称，如 **my-camel-k-kafka**，然后单击 **Create**。

3. 下载并安装 RHOAS CLI，如 Getting started with the rhoas CLI 所述。
4. 安装 Camel K operator 和 Camel K CLI，如 安装 Camel K 所述。
5. 要验证是否安装了 **Red Hat Integration - Camel K**Operator，点 **Operators** > **Installed Operators**。

后续步骤

使用 RHOAS 设置 Kafka 主题

2.2.2.2. 使用 RHOAS 设置 Kafka 主题

Kafka 整理有关 *主题的消息*。每个主题都有一个名称。应用向主题发送消息并从主题检索消息。Kafka 主题提供在 Kafka 实例中存储数据的目标。在向它发送数据前，您必须设置 Kafka 主题。

先决条件

- 您可以使用正确访问级别访问 OpenShift 集群、创建项目和安装操作器，以及在本地系统上安装 OpenShift 和 Camel K CLI。
- 已安装 OpenShift CLI (**oc**)、Camel K CLI (**kamel**)和 RHOAS CLI (**rhoas**)工具，如 准备 OpenShift 集群 中所述。
- 已安装 **Red Hat Integration - Camel K**operator，如 准备 OpenShift 集群 中所述。
- 您已登录到 Red Hat Cloud 站点。

流程

设置 Kafka 主题：

1. 在命令行中登录到您的 OpenShift 集群。
2. 打开您的项目，例如：
   oc project my-camel-k-kafka
3. 验证 Camel K Operator 是否已安装到项目中：
   oc get csv

   结果列出了 Red Hat Camel K operator，并表示它处于 **Succeeded** 阶段。
4. 准备 Kafka 实例并将其连接到 RHOAS：
   a. 使用以下命令登录到 RHOAS CLI：
      RHOAS 登录
   b. 创建一个 kafka 实例，如 **kafka-test** ：
      rhoas kafka create kafka-test

      创建 Kafka 实例的过程可能需要几分钟时间来完成。
5. 检查 Kafka 实例的状态：
   RHOAS 状态

   您还可以在 web 控制台中查看状态：

   https://cloud.redhat.com/application-services/streams/kafkas/

   当状态 **就绪时**，继续下一步。
6. 创建新的 Kafka 主题：
   rhoas kafka topic create --name test-topic

7. 将 Kafka 实例（集群）与 Openshift Application Services 实例连接：
   RHOAS 集群连接

8. 按照获取凭证令牌的脚本说明进行操作。
   您应该看到类似如下的输出：

```
Token Secret "rh-cloud-services-accesstoken-cli" created successfully
Service Account Secret "rh-cloud-services-service-account" created successfully
KafkaConnection resource "kafka-test" has been created
KafkaConnection successfully installed on your cluster.
```

后续步骤

- 获取 Kafka 凭证

2.2.2.3. 获取 Kafka 凭证

要将应用程序或服务连接到 Kafka 实例，您必须首先获取以下 Kafka 凭证：

- 获取 bootstrap URL。
- 使用凭证（用户名和密码）创建服务帐户。

对于 OpenShift Streams，身份验证协议是 SASL_SSL。

前提条件

- 您已创建了 Kafka 实例，它处于 ready 状态。
- 您已创建了 Kafka 主题。

流程

1. 获取 Kafka Broker URL (Bootstrap URL)：
   RHOAS 状态

   这个命令返回类似如下的输出：

```
Kafka
----------------------------------------------------------------
ID:                 1ptdfZRHmLKwqW6A3YKM2MawgDh
Name:                 my-kafka
Status:               ready
Bootstrap URL:        my-kafka--ptdfzrhmlkwqw-a-ykm-mawgdh.kafka.devshift.org:443
```

2. 要获取用户名和密码，请使用以下语法创建服务帐户：
   rhoas service-account create --name "<account-name>" --file-format json

注意

在创建服务帐户时，您可以选择文件格式和位置来保存凭证。如需更多信息，请键入 **rhoas service-account create --help**

例如：

rhoas service-account create --name "my-service-acct" --file-format json

服务帐户已创建并保存到 JSON 文件中。

3. 要验证您的服务帐户凭证，请查看 **credentials.json** 文件：
cat credentials.json

这个命令返回类似如下的输出：

```
{"clientID":"srvc-acct-eb575691-b94a-41f1-ab97-50ade0cd1094", "password":"facf3df1-3c8d-4253-aa87-8c95ca5e1225"}
```

4. 授予向 Kakfa 主题或从 Kakfa 发送和接收消息的权限。使用以下命令，其中 **clientID** 是 **credentials.json** 文件中提供的值（从第 3 步中）。

```
rhoas kafka acl grant-access --producer --consumer --service-account $CLIENT_ID --topic test-topic --group all
```

例如：

```
rhoas kafka acl grant-access --producer --consumer --service-account srvc-acct-eb575691-b94a-41f1-ab97-50ade0cd1094 --topic test-topic --group all
```

2.3. 将数据源连接到 KAMELET BINDING 中的 KAFKA 主题

要将数据源连接到 Kafka 主题，您可以创建一个 Kamelet Binding，*如图 2.2 所示*。

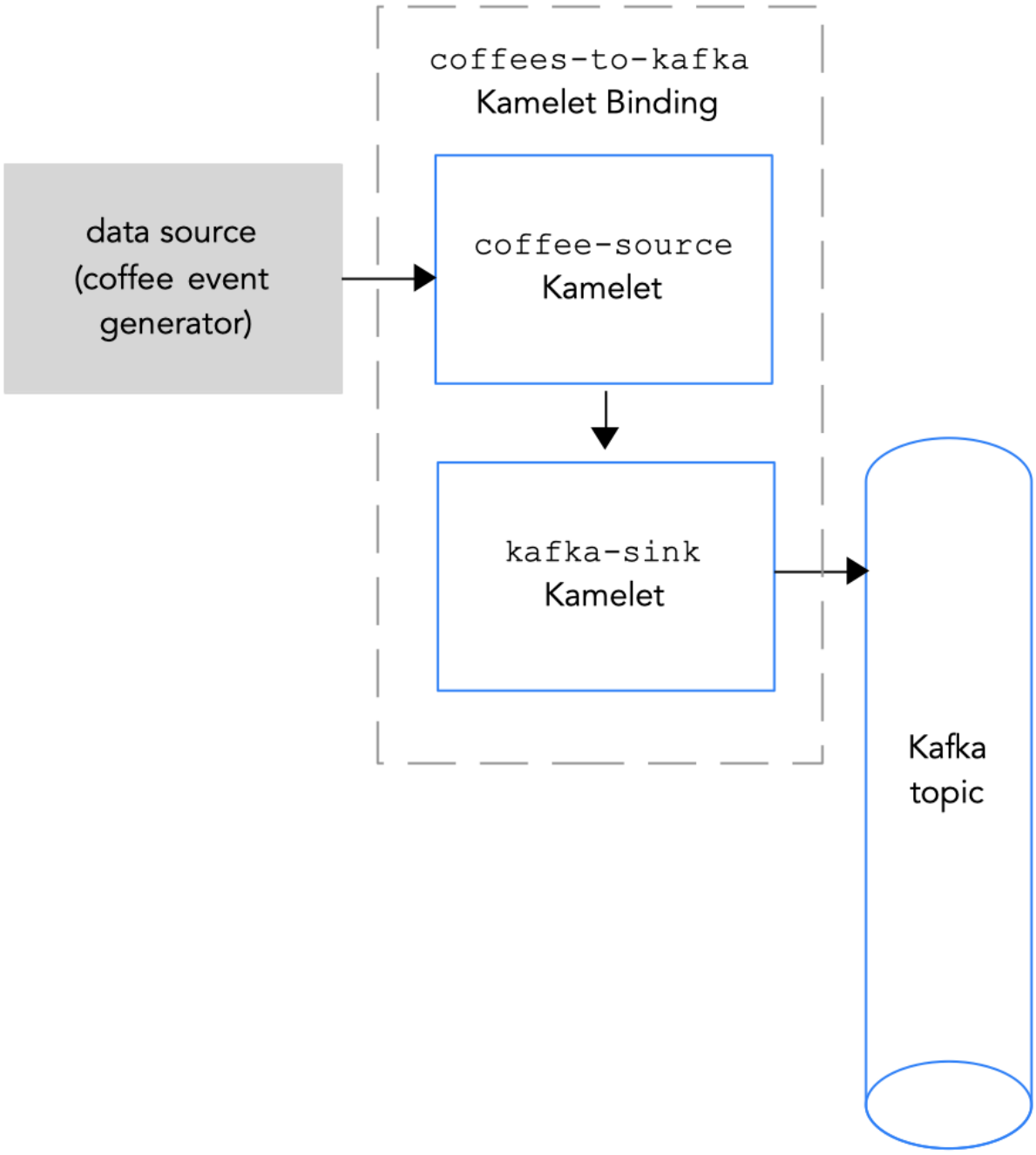


图 2.2 将数据源连接到 Kafka 主题

先决条件

- 您知道要将事件发送到的 Kafka 主题的名称。
  此流程中的示例使用 **test-topic** 接收事件。
- 您知道 Kafka 实例的以下参数值：
  - **bootstrapServers** - 以逗号分隔的 Kafka Broker URL 列表。
  - **Password** - 向 Kafka 进行身份验证的密码。对于 OpenShift Streams，这是 **credentials.json** 文件中的 **密码**。对于 AMQ Streams 上的未经身份验证的 kafka 实例，您可以指定任何非空字符串。

- **User** - 向 Kafka 进行身份验证的用户名。对于 OpenShift Streams，这是 **credentials.json** 文件中的 **clientID**。对于 AMQ Streams 上的未经身份验证的 kafka 实例，您可以指定任何非空字符串。
  有关如何使用 OpenShift Streams 时如何获取这些值的详情，请参考 获取 Kafka 凭证。

- **securityProtocol** - 您知道与 Kafka 代理通信的安全协议。对于 OpenShift Streams 上的 Kafka 集群，它是 **SASL_SSL** （默认）。对于 AMQ 流上的 Kafka 集群，它是 **PLAINTEXT**。

- 您知道您要添加到 Camel K 集成中的 Kamelets 和所需的实例参数。
  此流程的 Kamelets 示例包括：

  - **coffee-source** Kamelet - 它有一个可选参数 **周期**，用于指定发送每个事件的频率。您可以将代码从 Example source Kamelet 复制到名为 **coffee-source.kamelet.yaml** 文件的文件，然后运行以下命令将其作为资源添加到命名空间中：
    oc apply -f coffee-source.kamelet.yaml

  - Kamelet Catalog 中提供的 **kafka-sink** Kamelet。您可以使用 **kafka-sink** Kamelet，因为 Kafka 主题在这个绑定中接收数据（这是数据消费者）。

流程

要将数据源连接到 Kafka 主题，请创建一个 Kamelet Binding：

1. 在您选择的编辑器中，使用以下基本结构创建一个 YAML 文件：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name:
spec:
  source:
  sink:
```

2. 为 Kamelet Binding 添加名称。在本例中，名称是 **coffees-to-kafka**，因为绑定将 **coffee-source** Kamelet 连接到 **kafka-sink** Kamelet。

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffees-to-kafka
spec:
  source:
  sink:
```

3. 对于 Kamelet Binding 的源，指定一个数据源 Kamelet （例如，**coffee-source** Kamelet）生成事件，其中包含有关 coffee 的数据，并为 Kamelet 配置任何参数。

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffees-to-kafka
spec:
  source:
    ref:
      kind: Kamelet
```

```
    apiVersion: camel.apache.org/v1alpha1
    name: coffee-source
  properties:
    period: 5000
sink:
```

4. 对于 Kamelet Binding 的 sink，请指定 **kafka-sink** Kamelet 及其所需属性。
例如，当 Kafka 集群位于 OpenShift Streams 中时：

   - 对于 **user** 属性，指定 **clientID**，例如： **srvc-acct-eb575691-b94a-41f1-ab97-50ade0cd1094**
   - 对于 **password** 属性，指定密码，例如： **facf3df1-3c8d-4253-aa87-8c95ca5e1225**
   - 您不需要设置 **securityProtocol** 属性。

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffees-to-kafka
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: coffee-source
    properties:
      period: 5000
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-sink
    properties:
      bootstrapServers: "my-kafka--ptdfzrhmlkwqw-a-ykm-mawgdh.kafka.devshift.org:443"
      password: "facf3df1-3c8d-4253-aa87-8c95ca5e1225"
      topic: "test-topic"
      user: "srvc-acct-eb575691-b94a-41f1-ab97-50ade0cd1094"
```

对于另一个示例，当 Kafka 集群位于 AMQ Streams 时，将 **securityProtocol** 属性设置为 **"PLAINTEXT"**：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffees-to-kafka
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: coffee-source
    properties:
      period: 5000
  sink:
```

```
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-sink
    properties:
      bootstrapServers: "broker.url:9092"
      password: "testpassword"
      topic: "test-topic"
      user: "testuser"
      securityProtocol: "PLAINTEXT"
```

5. 保存 YAML 文件（例如，**coffees-to-kafka.yaml**）。
6. 登录您的 OpenShift 项目。
7. 将 Kamelet Binding 作为资源添加到 OpenShift 命名空间中：
   oc apply -f <kamelet binding filename>

   例如：

   oc apply -f coffees-to-kafka.yaml

   Camel K operator 使用 **KameletBinding** 资源生成并运行 Camel K 集成。构建可能需要几分钟时间。
8. 查看 **KameletBinding** 资源的状态：
   oc get kameletbindings
9. 查看其集成的状态：
   oc get integrations
10. 查看集成的日志：
    kamel logs <integration> -n <project>

    例如：

    kamel logs coffees-to-kafka -n my-camel-k-kafka

另请参阅

- 将操作应用到 Kafka 连接中的数据
- 处理连接中的错误
- 将 Kafka 主题连接到 Kamelet Binding 中的数据接收器

2.4. 将 KAFKA 主题连接到 KAMELET BINDING 中的数据接收器

要将 Kafka 主题连接到数据 sink，您可以创建一个 Kamelet Binding，*如图 2.3 所示*。

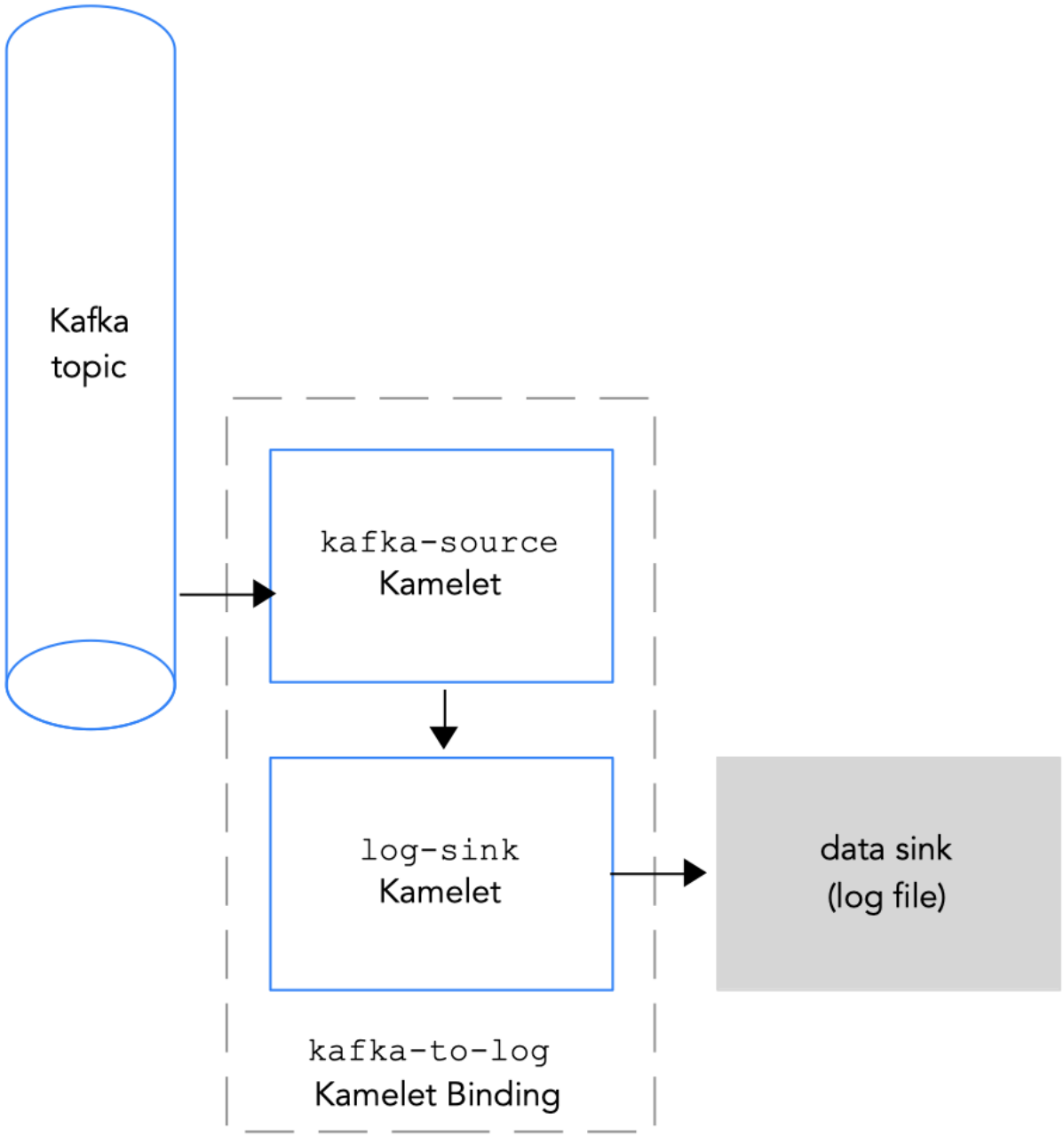


图 2.3 将 Kafka 主题连接到数据接收器

先决条件

- 您知道要从中发送事件的 Kafka 主题的名称。此流程中的示例使用 **test-topic** 发送事件。这与您在 Kamelet Binding 中将数据源连接到 Kafka 主题中的 coffee 源接收事件的主题相同。
- 您知道 Kafka 实例的以下参数值：
  - **bootstrapServers** - 以逗号分隔的 Kafka Broker URL 列表。
  - **Password** - 向 Kafka 进行身份验证的密码。
  - **User** - 向 Kafka 进行身份验证的用户名。
    有关如何使用 OpenShift Streams 时如何获取这些值的详情，请参考 获取 Kafka 凭证。
- 您知道与 Kafka 代理通信的安全协议。对于 OpenShift Streams 上的 Kafka 集群，它是 **SASL_SSL** （默认）。对于 AMQ 流上的 Kafka 集群，它是 **PLAINTEXT**。

- 您知道您要添加到 Camel K 集成中的 Kamelets 和所需的实例参数。此流程的 Kamelets 示例在 Kamelet Catalog 中提供：
  - **kafka-source** Kamelet - 使用 **kafka-source** Kamelet，因为 Kafka 主题在这个绑定中发送数据（这是数据制作者）。所需的参数的值示例为：
    - bootstrapServers - **"broker.url:9092"**
    - password - **"testpassword"**
    - user - **"testuser"**
    - topic - **"test-topic"**
    - **securityProtocol** - 对于 OpenShift Streams 上的 Kafka 集群，您不需要设置此参数，因为 **SASL_SSL** 是默认值。对于 AMQ 流上的 Kafka 集群，此参数值为 **"PLAINTEXT** "。
  - **log-sink** Kamelet - 使用 **log-sink** 来记录它从 **kafka-source** Kamelet 接收的数据。（可选）指定 **showStreams** 参数来显示数据的消息正文。**log-sink** Kamelet 用于调试目的。

流程

要将 Kafka 主题连接到数据 sink，请创建一个 Kamelet Binding：

1. 在您选择的编辑器中，使用以下基本结构创建一个 YAML 文件：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name:
spec:
  source:
  sink:
```

2. 为 Kamelet Binding 添加名称。在本例中，名称是 **kafka-to-log**，因为绑定将 **kafka-source** Kamelet 连接到 **log-sink** Kamelet。

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: kafka-to-log
spec:
  source:
  sink:
```

3. 对于 Kamelet Binding 的源，请指定 **kafka-source** Kamelet 并配置其参数。
例如，当 Kafka 集群位于 OpenShift Streams 上时（您不需要设置 **securityProtocol** 参数）：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: kafka-to-log
spec:
  source:
    ref:
      kind: Kamelet
```

```
    apiVersion: camel.apache.org/v1alpha1
    name: kafka-source
  properties:
    bootstrapServers: "broker.url:9092"
    password: "testpassword"
    topic: "test-topic"
    user: "testuser"
sink:
```

例如，当 Kafka 集群位于 AMQ Streams 中时，您必须将 **securityProtocol** 参数设置为 **"PLAINTEXT"**：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: kafka-to-log
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-source
    properties:
      bootstrapServers: "broker.url:9092"
      password: "testpassword"
      topic: "test-topic"
      user: "testuser"
      securityProtocol: "PLAINTEXT"
  sink:
```

4. 对于 Kamelet Binding 的 sink，指定数据消费者 Kamelet （例如，**log-sink** Kamelet）并为 Kamelet 配置任何参数，例如：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: kafka-to-log
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-source
    properties:
      bootstrapServers: "broker.url:9092"
      password: "testpassword"
      topic: "test-topic"
      user: "testuser"
      securityProtocol: "PLAINTEXT" // only for AMQ streams
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
```

```
    name: log-sink
  properties:
    showStreams: true
```

5. 保存 YAML 文件（如 **kafka-to-log.yaml**）。
6. 登录您的 OpenShift 项目。
7. 将 Kamelet Binding 作为资源添加到 OpenShift 命名空间中：
   oc apply -f <kamelet binding filename>

   例如：

   oc apply -f kafka-to-log.yaml

   Camel K operator 使用 **KameletBinding** 资源生成并运行 Camel K 集成。构建可能需要几分钟时间。
8. 查看 **KameletBinding** 资源的状态：
   oc get kameletbindings
9. 查看其集成的状态：
   oc get integrations
10. 查看集成的日志：
    kamel logs <integration> -n <project>

    例如：

    kamel logs kafka-to-log -n my-camel-k-kafka

    在输出中，您应该看到 coffee 事件，例如：

```
INFO  [log-sink-E80C5C904418150-0000000000000001] (Camel (camel-1) thread #0 - timer://tick) {"id":7259,"uid":"a4ecb7c2-05b8-4a49-b0d2-d1e8db5bc5e2","blend_name":"Postmodern Symphony","origin":"Huila, Colombia","variety":"Kona","notes":"delicate, chewy, black currant, red apple, star fruit","intensifier":"balanced"}
```

11. 要停止正在运行的集成，请删除关联的 Kamelet Binding 资源：
    oc delete kameletbindings/<kameletbinding-name>

    例如：

    oc delete kameletbindings/kafka-to-log

另请参阅

- 将操作应用到 Kafka 连接中的数据
- 在 Kamelet Binding 中添加错误处理器策略

2.5. 将操作应用到 KAFKA 连接中的数据

如果要对 Kamelet 和 Kafka 主题之间传递的数据执行操作，请使用 action Kamelets 作为 Kamelet Binding 中的中间步骤。

- 将操作应用到连接中的数据
- 将事件数据路由到不同的目标主题
- 过滤特定 Kafka 主题的事件数据

2.5.1. 将事件数据路由到不同的目标主题

当您配置到 Kafka 实例时，您可以选择将主题信息从事件数据转换，以便事件路由到不同的 Kafka 主题。使用以下转换操作 Kamelets 之一：

- **regex Router** - 使用正则表达式和替换字符串修改消息的主题。例如，如果要删除主题前缀、添加前缀或删除主题名称的一部分。配置 Regex Router Action Kamelet (**regex-router-action**)。
- **timestamp** - 根据原始主题和消息的时间戳修改消息的主题。例如，在使用需要写入不同表或基于时间戳的索引的接收器时。例如，当您要将事件从 Kafka 写入 Elasticsearch 时，但每个事件都需要根据事件本身的信息进入不同的索引。配置 Timestamp Router Action Kamelet (**timestamp-router-action**)。
- message **TimeStamp - 根据原始主题值修改消息** 的主题，以及来自消息值字段的 timestamp 字段。配置 Message Timestamp Router Action Kamelet (**message-timestamp-router-action**)。
- predicate - 通过配置 **Predicate** Filter Action Kamelet (**predicate-filter-action**)，根据给定的 JSON 路径表达式过滤事件。

先决条件

- 您已创建了 Kamelet Binding，其中接收器是一个 **kafka-sink** Kamelet，如 将数据源连接到 Kamelet Binding 中的 Kafka 主题 中所述。
- 您知道您要添加到 Kamelet Binding 中的转换类型。

流程

要转换目的地主题，请使用一个转换操作 Kamelets 作为 Kamelet Binding 中的中间步骤。

有关如何将操作 Kamelet 添加到 Kamelet Binding 的详情，请参阅 Adding a operation to a Kamelet Binding。

2.5.2. 过滤特定 Kafka 主题的事件数据

如果您使用为许多不同的 Kafka 主题生成记录的源 Kamelet，而您想要将记录过滤为一个 Kafka 主题，请将 **topic-name-matches-filter-action** Kamelet 添加为 Kamelet Binding 中的中间步骤。

先决条件

- 您已在 YAML 文件中创建了一个 Kamelet Binding。
- 您知道要过滤事件数据的 Kafka 主题的名称。

流程

1. 编辑 Kamelet Binding，使其包含 **topic-name-matches-filter-action** Kamelet 作为 source 和 sink Kamelets 之间的中间步骤。
通常，您可以使用 **kafka-source** Kamelet 作为源 Kamelet，并提供一个主题作为所需 **主题参数的值**。

在以下 Kamelet Binding 示例中，**kafka-source** Kamelet 指定 **test-topic、test-topic-2 和 test-topic-3** Kafka 主题，以及 **topic-name-matches-filter-action** Kamelet 指定过滤来自 **topic-test** 主题的事件数据：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: kafka-to-log-by-topic
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-source
    properties:
      bootstrapServers: "broker.url:9092"
      password: "testpassword"
      topic: "test-topic, test-topic-2, test-topic-3"
      user: "testuser"
      securityProtocol: "PLAINTEXT" // only for AMQ streams
steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: topic-name-matches-filter-action
    properties:
      regex:  "test-topic"
sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: log-sink
    properties:
        showStreams: true
```

如果要过滤来自 **kafka-source** Kamelet 以外的源 Kamelet 的主题，您必须提供 Kafka 主题信息。您可以使用 **insert-header-action** Kamelet 将 Kafka 主题字段添加为中间步骤，在 Kamelet Binding 中的 **topic-name-matches-filter-action** 步骤前，如下例所示：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffee-to-log-by-topic
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: coffee-source
    properties:
      period: 5000
```

```
steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name:  "KAFKA.topic"
      value:  "test-topic"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: topic-name-matches-filter-action
    properties:
      regex:  "test-topic"
sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: log-sink
    properties:
      showStreams: true
```

2. 保存 Kamelet Binding YAML 文件。

第 3 章 使用 KAMELETS 连接到 KNATIVE

您可以将 Kamelets 连接到 Knative 目的地（通道或代理）。Red Hat OpenShift Serverless 基于开源 Knative 项目，通过启用企业级无服务器平台在混合和多云环境中提供可移植性和一致性。OpenShift Serverless 包含对 Knative Eventing 和 Knative Serving 组件的支持。

Red Hat OpenShift Serverless、Knative Eventing 和 Knative Serving 可让您使用带有无服务器应用程序的 事件驱动的架构，使用 publish-subscribe 或 event-streaming 模型来分离事件制作者和消费者之间的关系。Knative Eventing 使用标准 HTTP POST 请求来发送和接收事件创建者和用户之间的事件。这些事件符合 CloudEvents 规范，它允许在任何编程语言中创建、解析、发送和接收事件。

您可以使用 Kamelets 将 CloudEvents 发送到 Knative，并将它们从 Knative 发送到事件消费者。kamelets 可将消息转换为 CloudEvents，您可以使用它们应用 CloudEvents 中数据的任何预处理和后处理。

3.1. 使用 KAMELETS 连接到 KNATIVE 概述

如果使用 Knative 流处理框架，您可以使用 Kamelets 将服务和应用程序连接到 Knative 目标（频道或代理）。

图 3.1 演示了将源和接收器 Kamelets 连接到 Knative 目的地的流程。

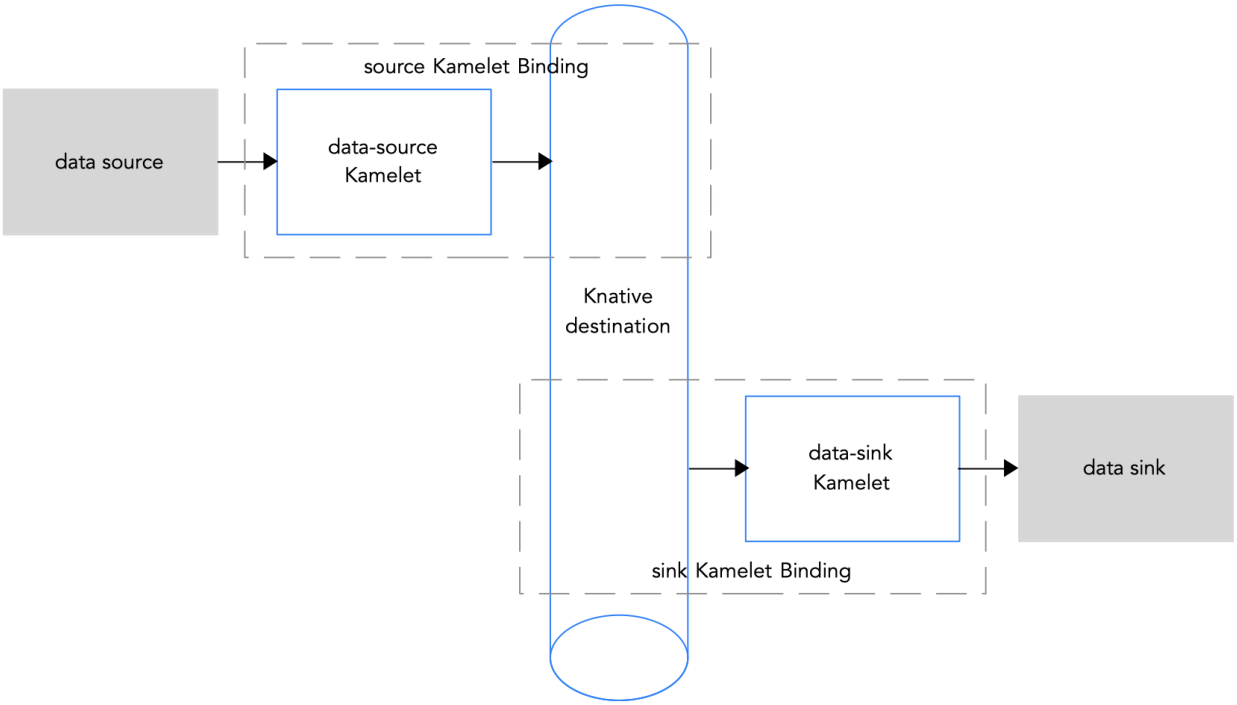


图 3.1：带有 Kamelets 和 Knative 频道的数据流

以下是使用 Kamelets 和 Kamelet Bindings 将应用程序和服务连接到 Knative 目的地的基本步骤概述：

1. 设置 Knative：
   a. 通过安装 Camel K 和 OpenShift Serverless operator 准备 OpenShift 集群。
   b. 安装所需的 Knative Serving 和 Eventing 组件。
   c. 创建 Knative 频道或代理。
2. 确定您要连接到 Knative 频道或代理的服务或应用程序。

3. 查看 Kamelet Catalog，以查找您要添加到集成的源和接收器组件的 Kamelets。另外，确定您要使用的每个 Kamelet 所需的配置参数。

4. 创建 Kamelet Bindings：

   - 创建一个 Kamelet Binding，将源 Kamelet 连接到 Knative 频道（或代理）。

   - 创建一个 Kamelet Binding，将 Knative 频道（或代理）连接到接收器 Kamelet。

5. （可选）通过在 Kamelet Binding 中添加一个或多个操作 Kamelets 作为 Intermediary 步骤来操作在 Knative 频道（或代理）和数据源或 sink 之间传递的数据。

6. （可选）定义如何在 Kamelet Binding 中处理错误。

7. 将 Kamelet Bindings 作为资源应用到项目。

Camel K operator 为每个 Kamelet Binding 生成一个单独的 Camel 集成。

当您将 Kamelet Binding 配置为使用 Knative 频道或代理作为事件源时，Camel K operator 会将对应的集成作为 Knative Serving 服务来利用 Knative 提供的自动扩展功能。

3.2. 设置 KNATIVE

设置 Knative 涉及安装所需的 OpenShift operator 并创建 Knative 频道。

3.2.1. 准备 OpenShift 集群

要使用 Kamelets 和 OpenShift Serverless，请安装以下 operator、组件和 CLI 工具：

- **Red Hat Integration - Camel K**operator 和 CLI 工具 - Operator 安装和管理 Camel K - 在 OpenShift 上原生运行的轻量级集成框架。**kamel** CLI 工具允许您访问所有 Camel K 功能。请参阅 安装 Camel K 中的安装说明。

- **OpenShift Serverless** operator - 提供一系列 API，使容器、微服务和功能能够运行"无服务器"。无服务器应用程序可按需扩展和缩减（到零），并由多个事件源触发。安装 OpenShift Serverless Operator 时，它会自动创建 **knative-serving** 命名空间（用于安装 Knative Serving 组件）和 **knative-eventing** 命名空间（安装 Knative Eventing 组件是必需的）。

- **Knative Eventing** 组件

- **Knative Serving** 组件

- **Knative CLI** 工具(**kn**)- 允许您从命令行或 Shell 脚本中创建 Knative 资源。

3.2.1.1. 安装 OpenShift Serverless

您可以从 OperatorHub 在 OpenShift 集群上安装 OpenShift Serverless Operator。OperatorHub 由 OpenShift Container Platform Web 控制台获得，为集群管理员提供了一个界面，以发现和安装 Operator。

OpenShift Serverless Operator 支持 Knative Serving 和 Knative Eventing 功能。如需了解更多详细信息，请参阅安装 OpenShift Serverless Operator。

先决条件

- 具有集群管理员访问安装了 Camel K Operator 的 OpenShift 项目。

- 已安装 OpenShift CLI 工具(**oc**)，以便可以在命令行中与 OpenShift 集群交互。有关如何安装 OpenShift CLI 的详情，请参阅安装 OpenShift CLI。

流程

1. 在 OpenShift Container Platform Web 控制台中，使用具有集群管理员特权的帐户登录。
2. 在左侧导航菜单中点 **Operators** > **OperatorHub**。
3. 在 **Filter by keyword** 文本框中，输入 **Serverless** 以查找 **OpenShift Serverless Operator**。
4. 阅读 Operator 的信息，然后点 **Install** 显示 Operator 订阅页面。
5. 选择默认订阅设置：
   - **Update Channel** > 选择与 OpenShift 版本匹配的频道，如 **4.14**
   - **Installation Mode** > **All namespaces on the cluster**
   - **Approval Strategy** > **Automatic**

   注意

   如果您的环境需要，也可以使用 **Approval Strategy** > **Manual** 设置。
6. 点 **Install**，等待片刻，直到 Operator 准备就绪为止。
7. 使用 OpenShift 文档中的步骤安装所需的 Knative 组件：
   - 安装 Knative Serving
   - 安装 Knative Eventing
8. （可选）下载并安装 OpenShift Serverless CLI 工具：
   a. 在 OpenShift Web 控制台顶部的帮助菜单(?)中，选择 **Command line tools**。
   b. 向下滚动到 **kn - OpenShift Serverless - Command Line Interface**部分。
   c. 单击链接以下载本地操作系统的二进制文件(Linux、Mac、Windows)
   d. 在您的系统路径中解压并安装 CLI。
   e. 要验证是否可以访问 **kn** CLI，请打开一个命令窗口，然后输入以下内容：
   kn --help

   此命令显示有关 OpenShift Serverless CLI 命令的信息。

   如需了解更多详细信息，请参阅 OpenShift Serverless CLI 文档。

其他资源

- 在 OpenShift 文档中安装 OpenShift Serverless

3.2.2. 创建 Knative 频道

Knative 频道是转发事件的自定义资源。事件源或生成程序将事件发送到频道后，可使用订阅将这些事件发送到多个 Knative 服务或其他 sink。

本例使用 **InMemoryChannel** 频道，用于 OpenShift Serverless 进行开发目的。请注意，**InMemoryChannel** 类型频道有以下限制：

- 事件没有持久性。如果 Pod 停机，则 Pod 上的事件将会丢失。
- **InMemoryChannel** 频道没有实现事件排序，因此同时接收到的两个事件可能会以任何顺序传送给订阅者。
- 如果订阅者拒绝某个事件，则不会默认重新发送尝试。您可以通过修改 Subscription 对象中的 delivery 规格来配置重新发送尝试。

先决条件

- OpenShift Serverless Operator、Knative Eventing 和 Knative Serving 组件安装在 OpenShift Container Platform 集群中。
- 已安装 OpenShift Serverless CLI (**kn**)。
- 您已创建了一个项目，或者具有适当的角色和权限访问项目，以便在 OpenShift Container Platform 中创建应用程序和其他工作负载。

流程

1. 登录您的 OpenShift 集群。
2. 打开您要在其中创建集成应用程序的项目。例如：
   oc project camel-k-knative
3. 使用 Knative (**kn**) CLI 命令创建频道
   kn channel create <channel_name> --type <channel_type>

   例如，创建名为 **mychannel** 的频道：

   kn channel create mychannel --type messaging.knative.dev:v1:InMemoryChannel
4. 要确认频道现在存在，请输入以下命令列出所有现有频道：
   kn channel list

   您应该在列表中看到您的频道。

后续步骤

- 将数据源连接到 Kamelet Binding 中的 Knative 目的地
- 将 Knative 目的地连接到 Kamelet Binding 中的数据接收器

3.2.3. 创建 Knative 代理

Knative 代理是一个自定义资源，用于定义用于收集 CloudEvent 池的事件网格。OpenShift Serverless 提供了一个 default Knative 代理，您可以使用 **kn** CLI 创建该代理。

您可以在 Kamelet Binding 中使用代理，例如，当应用程序处理多个事件类型且您不想为每个事件类型创建频道时。

先决条件

- OpenShift Serverless Operator、Knative Eventing 和 Knative Serving 组件安装在 OpenShift Container Platform 集群中。
- 已安装 OpenShift Serverless CLI (**kn**)。
- 您已创建了一个项目，或者具有适当的角色和权限访问项目，以便在 OpenShift Container Platform 中创建应用程序和其他工作负载。

流程

1. 登录您的 OpenShift 集群。
2. 打开您要在其中创建集成应用程序的项目。例如：
   oc project camel-k-knative
3. 使用此 Knative (**kn**) CLI 命令创建代理：
   kn broker create default
4. 要确认代理现在存在，请输入以下命令列出所有现有代理：
   kn broker list

您应该在列表中看到 default 代理。

后续步骤

- 将数据源连接到 Kamelet Binding 中的 Knative 目的地
- 将 Knative 目的地连接到 Kamelet Binding 中的数据接收器

3.3. 将数据源连接到 KAMELET BINDING 中的 KNATIVE 目的地

要将数据源连接到 Knative 目标（频道或代理），您可以创建一个 Kamelet Binding，*如图 3.2 所示*。

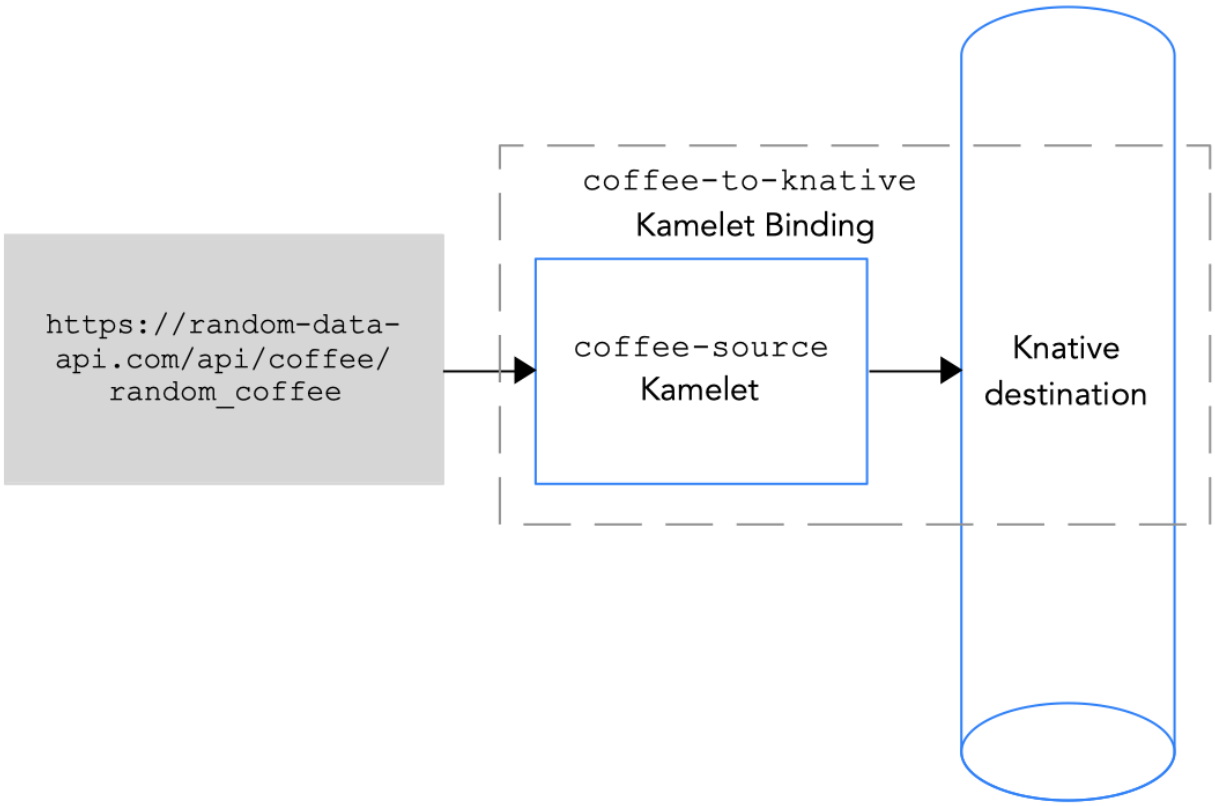


图 3.2 将数据源连接到 Knative 目的地

Knative 目的地可以是 Knative 频道或 Knative 代理。

当您向频道发送数据时，该频道只有一个事件类型。您不需要在 Kamelet Binding 中为频道指定任何属性值。

将数据发送到代理时，因为代理可以处理多个事件类型，您必须在 Kamelet Binding 中引用代理时为 type 属性指定一个值。

先决条件

- 您知道您要将事件发送到的 Knative 频道或代理的名称和类型。
  此流程中的示例使用名为 **mychannel** 的 **InMemoryChannel** 频道或名为 **default** 的代理。对于代理示例，对于 coffee 事件，**type** 属性值为 **coffee**。
- 您知道您要添加到 Camel 集成中的 Kamelet 和所需的实例参数。
  此流程的 Kamelet 示例是 **coffee-source** Kamelet。它有一个可选参数， 用于指定发送每个事件的频率。您可以将代码从 Example source Kamelet 复制到名为 **coffee-source.kamelet.yaml** 文件的文件，然后运行以下命令将其作为资源添加到命名空间中：

  oc apply -f coffee-source.kamelet.yaml

流程

要将数据源连接到 Knative 目的地，请创建一个 Kamelet Binding：

1. 在您选择的编辑器中，使用以下基本结构创建一个 YAML 文件：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name:
spec:
  source:
  sink:
```

2. 为 Kamelet Binding 添加名称。在本例中，名称是 **coffees-to-knative**，因为绑定将 **coffee-source** Kamelet 连接到 Knative 目的地。

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffees-to-knative
spec:
  source:
  sink:
```

3. 对于 Kamelet Binding 的源，指定一个数据源 Kamelet （例如，**coffee-source** Kamelet）生成事件，其中包含有关 coffee 的数据，并为 Kamelet 配置任何参数。

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffees-to-knative
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: coffee-source
    properties:
      period: 5000
  sink:
```

4. 对于 Kamelet Binding 的 sink，请指定 Knative 频道或代理以及所需的参数。
这个示例将 Knative 频道指定为 sink：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffees-to-knative
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: coffee-source
    properties:
      period: 5000
  sink:
    ref:
```

```
      apiVersion: messaging.knative.dev/v1
      kind: InMemoryChannel
      name: mychannel
```

本例将 Knative 代理指定为接收器：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: coffees-to-knative
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: coffee-source
    properties:
      period: 5000
  sink:
      ref:
        kind: Broker
        apiVersion: eventing.knative.dev/v1
        name: default
      properties:
        type: coffee
```

5. 保存 YAML 文件（例如，**coffees-to-knative.yaml**）。
6. 登录您的 OpenShift 项目。
7. 将 Kamelet Binding 作为资源添加到 OpenShift 命名空间中：
   oc apply -f <kamelet binding filename>

   例如：

   oc apply -f coffees-to-knative.yaml

   Camel K operator 使用 **KameletBinding** 资源生成并运行 Camel K 集成。构建可能需要几分钟时间。
8. 查看 **KameletBinding** 的状态：
   oc get kameletbindings
9. 查看其集成的状态：
   oc get integrations
10. 查看集成的日志：
    kamel logs <integration> -n <project>

    例如：

    kamel logs coffees-to-knative -n my-camel-knative

后续步骤

- 将 Knative 目的地连接到 Kamelet Binding 中的数据接收器

另请参阅

- 将操作应用到连接中的数据
- 处理连接中的错误

3.4. 将 KNATIVE 目的地连接到 KAMELET BINDING 中的数据接收器

要将 Knative 目的地连接到数据 sink，您可以创建一个 Kamelet Binding，*如图 3.3 所示*。

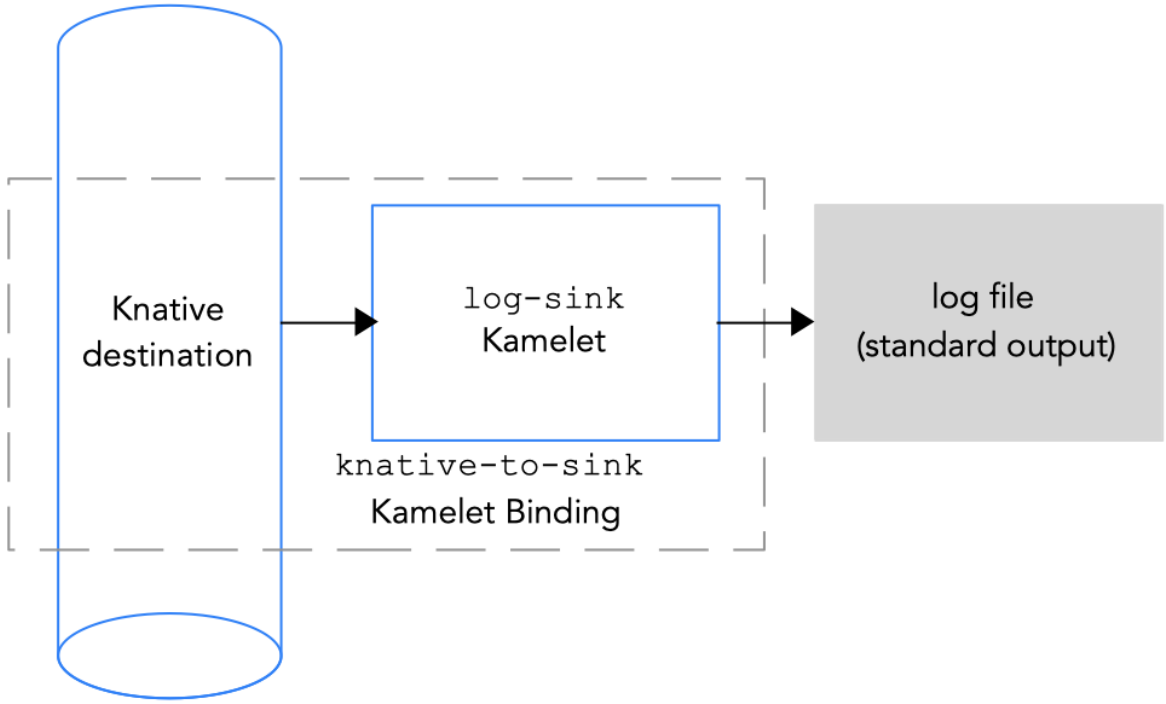


图 3.3 将 Knative 目的地连接到数据接收器

Knative 目的地可以是 Knative 频道或 Knative 代理。

当您从频道发送数据时，该频道只有一个事件类型。您不需要在 Kamelet Binding 中为频道指定任何属性值。

当您从代理发送数据时，因为代理可以处理多个事件类型，您必须在 Kamelet Binding 中引用代理时为 type 属性指定一个值。

先决条件

- 您知道 Knative 频道的名称和类型，或您要从中接收事件的代理名称。对于代理，您还知道要接收的事件类型。
  此流程中的示例使用名为 mychannel 的 InMemoryChannel 频道或名为 mybroker 和 coffee 事件（用于 type 属性）的代理。这些 与目的地相同的示例目的地，用于从将数据源连接到 Kamelet Binding 中的 Knative 频道的 coffee 源接收事件。
- 您知道您要添加到 Camel 集成中的 Kamelet 和所需的实例参数。
  此流程的 Kamelet 示例是在 Kamelet Catalog 中提供的 **log-sink** Kamelet，可用于测试和调试。指定的 **showStreams** 参数来显示数据的消息正文。

流程

要将 Knative 频道连接到数据 sink，请创建一个 Kamelet Binding：

1. 在您选择的编辑器中，使用以下基本结构创建一个 YAML 文件：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name:
spec:
  source:
  sink:
```

2. 为 Kamelet Binding 添加名称。在本例中，名称是 **knative-to-log**，因为绑定将 Knative 目的地连接到 **log-sink** Kamelet。

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: knative-to-log
spec:
  source:
  sink:
```

3. 对于 Kamelet Binding 的源，请指定 Knative 频道或代理以及所需的参数。
这个示例将 Knative 频道指定为源：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: knative-to-log
spec:
  source:
    ref:
      apiVersion: messaging.knative.dev/v1
      kind: InMemoryChannel
      name: mychannel
  sink:
```

这个示例将 Knative 代理指定为源：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: knative-to-log
spec:
  source:
    ref:
      kind: Broker
      apiVersion: eventing.knative.dev/v1
      name: default
    properties:
      type: coffee
sink:
```

4. 对于 Kamelet Binding 的 sink，指定数据消费者 Kamelet （例如，**log-sink** Kamelet）并为 Kamelet 配置任何参数，例如：

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: knative-to-log
spec:
  source:
    ref:
      apiVersion: messaging.knative.dev/v1
      kind: InMemoryChannel
      name: mychannel
  sink:
    ref:
      apiVersion: camel.apache.org/v1alpha1
      kind: Kamelet
      name: log-sink
    properties:
      showStreams: true
```

5. 保存 YAML 文件（如 **knative-to-log.yaml**）。

6. 登录您的 OpenShift 项目。

7. 将 Kamelet Binding 作为资源添加到 OpenShift 命名空间中： **oc apply -f <kamelet binding filename>**
例如：

   oc apply -f knative-to-log.yaml

   Camel K operator 使用 **KameletBinding** 资源生成并运行 Camel K 集成。构建可能需要几分钟时间。

8. 查看 **KameletBinding** 的状态：
oc get kameletbindings

9. 查看集成的状态：
oc get integrations

10. 查看集成的日志：
kamel logs <integration> -n <project>

    例如：

    kamel logs knative-to-log -n my-camel-knative

    在输出中，您应该看到 coffee 事件，例如：

```
[1] INFO [sink] (vert.x-worker-thread-1) {"id":254,"uid":"8e180ef7-8924-4fc7-ab81-
d6058618cc42","blend_name":"Good-morning Star","origin":"Santander,
Colombia","variety":"Kaffa","notes":"delicate, creamy, lemongrass, granola,
soil","intensifier":"sharp"}
[1] INFO [sink] (vert.x-worker-thread-2) {"id":8169,"uid":"3733c3a5-4ad9-43a3-9acc-
```

d4cd43de6f3d","blend_name":"Caf? Java","origin":"Nayarit, Mexico","variety":"Red Bourbon","notes":"unbalanced, full, granola, bittersweet chocolate, nougat","intensifier":"delicate"}

11. 要停止正在运行的集成，请删除关联的 Kamelet Binding 资源：
oc delete kameletbindings/<kameletbinding-name>

    例如：

    oc delete kameletbindings/knative-to-log

另请参阅

- 将操作应用到连接中的数据
- 处理连接中的错误

第 4 章 KAMELETS 参考

4.1. KAMELET 结构

Kamelet 通常以 YAML 域特定语言进行编码。文件名前缀是 Kamelet 的名称。例如，名称 **FTP sink** 的 Kamelet 具有文件名 **ftp-sink.kamelet.yaml**。

请注意，在 OpenShift 中，Kamelet 是一个显示 Kamelet 的名称的资源（而不是文件名）。

在高级别上，Kamelet 资源描述了：

- 包含 Kamelet 的 ID 和其他信息的 metadata 部分，如 Kamelet 类型(**源**、**sink** 或 **action**)。
- 包含一组参数的定义(JSON-schema 规格)，可用于配置 Kamelet。
- 一个可选的 **type** 部分，其中包含 Kamelet 预期的输入和输出的信息。
- 定义 Kamelet 实施的 YAML DSL 中的 Camel 模板。

下图显示了 Kamelet 及其部分的示例。

Kamelet 结构示例

```
telegram-text-source.kamelet.yaml
apiVersion: camel.apache.org/v1alpha1
kind: Kamelet
metadata:
  name: telegram-source (1)
  annotations: (2)
    camel.apache.org/catalog.version: "master-SNAPSHOT"
    camel.apache.org/kamelet.icon: "data:image/..."
    camel.apache.org/provider: "Red Hat"
    camel.apache.org/kamelet.group: "Telegram"
  labels: (3)
    camel.apache.org/kamelet.type: "source"
spec:
  definition: (4)
    title: "Telegram Source"
    description: |-
      Receive all messages that people send to your telegram bot.
      To create a bot, contact the @botfather account using the
      Telegram app.
      The source attaches the following headers to the messages:
      - chat-id / ce-chatid: the ID of the chat where the
      message comes from
    required:
      - authorizationToken
    type: object
    properties:
      authorizationToken:
        title: Token
        description: The token to access your bot on Telegram, that you
              can obtain from the Telegram "Bot Father".
        type: string
        format: password
```

```
      x-descriptors:
      - urn:alm:descriptor:com.tectonic.ui:password
  types: 5
    out:
      mediaType: application/json
  dependencies:
  - "camel:jackson"
  - "camel:kamelet"
  - "camel:telegram"
  template: 6
    from:
      uri: telegram:bots
      parameters:
        authorizationToken: "{{authorizationToken}}"
      steps:
      - set-header:
        name: chat-id
        simple: "${header[CamelTelegramChatId]}"
      - set-header:
        name: ce-chatid
        simple: "${header[CamelTelegramChatId]}"
      - marshal:
        json: {}
      - to: "kamelet:sink"
```

1. Kamelet ID - 当您要引用 Kamelet 时，在 Camel K 集成中使用此 ID。
2. 注解（如 icon）为 Kamelet 提供显示功能。
3. 标签允许用户查询 Kamelets （例如 kind: "source", "sink", 或 "action"）
4. JSON-schema 规格格式的 Kamelet 和参数的描述。
5. 输出的介质类型（可以包含 schema）。
6. 定义 Kamelet 行为的路由模板。

4.2. 源 KAMELET 示例

以下是示例 **coffee-source** Kamelet 的内容：

```
apiVersion: camel.apache.org/v1alpha1
kind: Kamelet
metadata:
  name: coffee-source
  labels:
    camel.apache.org/kamelet.type: "source"
spec:
  definition:
    title: "Coffee Source"
    description: "Retrieve a random coffee from a catalog of coffees"
    properties:
      period:
        title: Period
        description: The interval between two events in seconds
```

```
      type: integer
      default: 1000
types:
  out:
    mediaType: application/json
template:
  from:
    uri: timer:tick
    parameters:
      period: "{{period}}"
    steps:
    - to: "https://random-data-api.com/api/coffee/random_coffee"
    - to: "kamelet:sink"
```