Red Hat Streams for Apache Kafka 2.7

在 Apache Kafka Bridge 的 Streams 中使用 3scale API 管理

使用 3scale 提供的功能和功能

Last Updated: 2024-05-29

Red Hat Streams for Apache Kafka 2.7 在 Apache Kafka Bridge 的 Streams 中使用 3scale API 管理

使用 3scale 提供的功能和功能

法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

在 OpenShift Container Platform 中部署 AMQ Streams Kafka Bridge，并集成 Red Hat 3scale API Management。

目录

前言

对红帽文档提供反馈

我们感谢您对我们文档的反馈。

要改进，创建一个 JIRA 问题并描述您推荐的更改。提供尽可能多的详细信息，以便我们快速解决您的请求。

前提条件

- 您有红帽客户门户网站帐户。此帐户可让您登录到 Red Hat Jira Software 实例。
  如果您没有帐户，系统会提示您创建一个帐户。

流程

1. 点以下内容： Create issue。
2. 在 **Summary** 文本框中输入问题的简短描述。
3. 在 **Description** 文本框中提供以下信息：
   - 找到此问题的页面的 URL。
   - 有关此问题的详细描述。
     您可以将信息保留在任何其他字段中的默认值。
4. 添加 reporter 名称。
5. 点 **Create** 将 JIRA 问题提交到文档团队。

感谢您花时间来提供反馈。

第 1 章 3SCALE API 管理

如果在 OpenShift Container Platform 上部署了 Kafka Bridge，您可以在 3scale 中使用它。

对于 Kafka Bridge 的普通部署，没有置备身份验证或授权，且不支持外部客户端的 TLS 加密连接。3scale API 管理可以使用 TLS 保护 Kafka Bridge，并提供身份验证和授权。与 3scale 集成还意味着提供了指标、速率限制和计费等其他功能。

使用 3scale，您可以对希望访问 Apache Kafka 的外部客户端使用不同类型的身份验证。3scale 支持以下验证类型：

标准 API 密钥

单一随机字符串或哈希充当标识符和机密令牌。

应用程序标识符和密钥对

不可变标识符和可变 secret key 字符串。

OpenID Connect

委派的身份验证的协议。

1.1. KAFKA BRIDGE 服务发现

3scale 使用服务发现集成，这需要 3scale 部署到与 Apache Kafka 和 Kafka Bridge 相同的 OpenShift 集群。

Apache Kafka Cluster Operator 部署的流必须设置以下环境变量：

- STRIMZI_CUSTOM_KAFKA_BRIDGE_SERVICE_LABELS
- STRIMZI_CUSTOM_KAFKA_BRIDGE_SERVICE_ANNOTATIONS

当部署了 Kafka Bridge 时，用于公开 Kafka Bridge 的 REST 接口的服务使用注解和标签以供 3scale 发现。

- 3scale 使用 **discovery.3scale.net=true** 标签来查找服务。
- 注解提供有关该服务的信息。

您可以在 OpenShift 控制台中通过导航到 Kafka Bridge 实例的 **Services** 来检查您的配置。在 **Annotations** 下，您会看到 Kafka Bridge 的 OpenAPI 规格的端点。

1.2. 3SCALE APICAST 网关策略

3scale 与 3scale APIcast 结合使用，3scale APIcast 是一个与 3scale 一起部署的 API 网关，为 Kafka Bridge 提供单一入口点。

APIcast 策略提供了一种机制来定制网关的运行方式。3scale 为网关配置提供一组标准策略。您还可以创建自己的策略。

如需有关 APIcast 策略的更多信息，请参阅 Red Hat 3scale 文档。

Kafka Bridge 的 APIcast 策略

3scale 与 Kafka Bridge 集成的示例策略配置随 **policies_config.json** 文件提供，该文件定义：

- 匿名访问

- 标头修改
- 路由
- URL rewriting

网关策略通过此文件启用或禁用。

您可以使用此示例来定义您自己的策略。

匿名访问

匿名访问策略在没有身份验证的情况下公开服务，在 HTTP 客户端不提供它们时提供默认凭证（匿名访问）。该策略不是强制性的，如果始终需要身份验证，则可以禁用或删除。

标头修改

标头修改策略允许修改现有的 HTTP 标头，或向通过网关的请求或响应添加新的标头。对于 3scale 集成，策略会向从 HTTP 客户端通过网关传递给 Kafka 网桥的每个请求添加标头。
当 Kafka Bridge 收到创建一个新消费者的请求时，它会返回一个 JSON 有效负载，其中包含一个带有消费者所有后续请求必须使用的 URI 的 **base_uri**。例如：

```
{
  "instance_id": "consumer-1",
  "base_uri":"http://my-bridge:8080/consumers/my-group/instances/consumer1"
}
```

使用 APIcast 时，客户端会将所有后续请求发送到网关，而不是直接发送到 Kafka Bridge。因此，URI 需要网关主机名，而不是网关后面的 Kafka 网桥的地址。

使用标头修改策略，将标头添加到来自 HTTP 客户端的请求，以便 Kafka Bridge 使用网关主机名。

例如，通过应用 **Forwarded: host=my-gateway:80;proto=http** 标头，Kafka Bridge 会为使用者提供以下内容。

```
{
    "instance_id": "consumer-1",
    "base_uri":"http://my-gateway:80/consumers/my-group/instances/consumer1"
}
```

X-Forwarded-Path 标头将来自客户端的请求中包含的原始路径传输到网关。当网关支持多个 Kafka Bridge 实例时，这个标头严格与应用的路由策略相关。

路由

当存在多个 Kafka Bridge 实例时，会应用路由策略。请求必须发送到初始创建使用者的同一 Kafka Bridge 实例，因此请求必须为网关指定一个路由，以将请求转发到适当的 Kafka Bridge 实例。
路由策略会命名每个网桥实例，并使用名称执行路由。在部署 **Kafka Bridge** 时，您可以在 KafkaBridge 自定义资源中指定名称。

例如，每个请求（使用 **X-Forwarded-Path**）从消费者到：

http://my-gateway:80/my-bridge-1/consumers/my-group/instances/consumer1

转发到：

http://my-bridge-1-bridge-service:8080/consumers/my-group/instances/consumer1

URL 重写策略会删除网桥名称，因为它在将请求从网关转发到 Kafka Bridge 时不使用它。

URL rewriting

在将请求从网关转发到 Kafka Bridge 时，URL rewiring 策略确保从客户端到特定 Kafka Bridge 的请求不包含网桥名称。
网桥名称不在网桥公开的端点中使用。

1.3. 3SCALE APICAST 用于 TLS 验证

您可以为 TLS 验证设置 APIcast，它需要使用模板自管理 APIcast。**apicast** 服务作为路由公开。

您还可以在 Kafka Bridge API 中应用 TLS 策略。

1.4. 使用现有的 3SCALE 部署

如果您已将 3scale 部署到 OpenShift，并且希望将其与 Kafka Bridge 搭配使用，请确保您以完成了K为 Kafka Bridge 部署 3scale中介绍的设置。

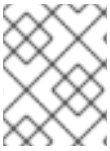
第 2 章 为 KAFKA BRIDGE 部署 3SCALE

要将 3scale 与 Kafka Bridge 搭配使用，您首先部署它，然后将其配置为发现 Kafka Bridge API。

在这种情况下，Apache Kafka、Kafka、Kafka Bridge 和 3scale API 管理组件的 Streams 在相同的 OpenShift 集群中运行。

以下 3scale 组件可帮助发现 Kafka Bridge：

- 3scale APIcast 为 HTTP 客户端提供基于 NGINX 的 API 网关，以连接到 Kafka Bridge API 服务。
- 3scale toolbox 用于将 Kafka Bridge 服务的 OpenAPI 规格导入到 3scale。



注意

如果 3scale 已部署到与 Kafka Bridge 相同的集群中，您可以跳过部署步骤并使用您的当前部署。

先决条件

- 部署需要了解 3scale 组件。
- Apache Kafka 和 Kafka 的流正在运行。
- Kafka Bridge 已部署。

对于 3scale 部署：

- 检查 Red Hat 3scale API Management 支持的配置。
- 安装需要具有 **cluster-admin** 角色的用户，如 **system:admin**。
- 您需要访问描述以下内容的 JSON 文件：
  - Kafka Bridge OpenAPI specification (**openapiv2.json**)
  - Kafka Bridge 的标头修改和路由策略(**policies_config.json**)
    在 GitHub 上查找 JSON 文件。

流程

1. 设置 3scale API 管理，如 Red Hat 3scale 文档 中所述。
   a. 使用 3scale API 管理操作器安装 3scale API Manager 和 APIcast。
   在部署 API Manager 前，将 **API Manager** 自定义资源的 **wildcardDomain** 属性更新为托管 OpenShift 集群的域。

   域在 URL 中使用用于访问 3scale 管理门户(**http[s]://<authentication_token>@3scale-admin.<cluster_domain&**gt;)。

   b. 通过检查 **API Manager** 自定义资源的状态，验证 3scale 是否已成功部署。
2. 为 3scale API Manager 授予 3scale API Manager 授权来发现 Kafka Bridge 服务：

```
oc adm policy add-cluster-role-to-user view system:serviceaccount:
<my_bridge_namespace>:amp
```

该命令向指定命名空间中的 Kafka Bridge 资源(<**my_bridge_namespace**>)授予 API Manager (**amp**)读取访问权限(查看)。

3. 设置 3scale API 管理 toolbox。

   a. 按照 Red Hat 3scale 文档所述安装 3scale toolbox。

   b. 设置环境变量以便与 3scale 交互：

   Kafka Bridge 配置示例

```
export REMOTE_NAME=strimzi-kafka-bridge 1
export SYSTEM_NAME=strimzi_http_bridge_for_apache_kafka 2
export TENANT=strimzi-kafka-bridge-admin 3
export PORTAL_ENDPOINT=$TENANT.3scale.net 4
export TOKEN=<3scale_authentication_token> 5
```

   1. **REMOTE_NAME** 是分配给 3scale 管理门户的远程地址的名称。
   2. **SYSTEM_NAME** 是通过 3scale toolbox 导入 OpenAPI 规格创建的 3scale 服务/API 的名称。
   3. **TENANT** 是 3scale 管理门户的租户名称(**https://$TENANT.3scale.net**)。
   4. **PORTAL_ENDPOINT** 是运行 3scale 管理门户的端点。
   5. **TOKEN** 是 3scale 管理门户提供的身份验证令牌，用于通过 3scale toolbox 或 HTTP 请求进行交互。

   c. 配置 3scale toolbox 的远程 web 地址：

```
podman run -v /path/to/openapiv2.json:/tmp/oas/openapiv2.json registry.redhat.io/3scale-amp2/toolbox-rhel8:3scale2.14 3scale import openapi -d <admin_portal_url> /tmp/oas/openapiv2.json
```

   为 3scale toolbox 指定容器镜像。3scale 的容器镜像在 红帽生态系统目录 中提供。

   将 **<admin_portal_url** > 替换为 3scale 管理门户端点的路径 (**https://$TOKEN@$PORTAL_ENDPOINT**/)。现在，每次运行 toolbox 时不需要指定 3scale 管理门户的端点地址。

4. 检查您的 Cluster Operator 部署是否有 3scale 发现 Kafka Bridge 服务所需的标签和注解属性。

```
#...
env:
- name: STRIMZI_CUSTOM_KAFKA_BRIDGE_SERVICE_LABELS
    value: |
    discovery.3scale.net=true
- name: STRIMZI_CUSTOM_KAFKA_BRIDGE_SERVICE_ANNOTATIONS
    value: |
    discovery.3scale.net/scheme=http
    discovery.3scale.net/port=8080
```

```
    discovery.3scale.net/path=/
    discovery.3scale.net/description-path=/openapi
#...
```

如果没有，请通过 OpenShift 控制台添加属性，或尝试重新部署 Cluster Operator 和 Kafka Bridge。

5. 在 3scale 管理门户中，从 OpenShift 导入 Kafka Bridge API 服务，如 Red Hat 3scale 文档所述。

6. 编辑 OpenAPI 规格(JSON 文件)中的 **Host** 字段，以使用 Kafka Bridge 服务的基本 URL：
例如：

```
"host": "my-bridge-bridge-service.my-project.svc.cluster.local:8080"
```

检查**主机** URL 包括以下内容：

- Kafka Bridge name (**my-bridge**)
- 项目名称(**my-project**)
- Kafka Bridge 的端口(**8080**)

7. 从本地文件将更新的 OpenAPI 规格导入到 3scale toolbox：

```
podman run -v /path/to/openapiv2.json:/tmp/oas/openapiv2.json registry.redhat.io/3scale-amp2/toolbox-rhel8:3scale2.14 3scale import openapi [opts] -d=<admin_portal_url> -t 3scale-kafka-bridge /tmp/oas/openapiv2.json
```

在这里，我们将系统名称指定为 **3scale-kafka-bridge**，而不是从 OpenAPI 规格生成名称。将 **/path/to/openapiv2.json** 替换为 OpenAPI 规格文件的路径，将 **<admin_portal_url** > 替换为 3scale 管理门户的端点的路径。

8. 为服务导入标头修改和路由策略 (JSON 文件)。

   a. 找到您在 3scale 中创建的服务的 ID，这是导入策略时所需的：

```
export SERVICE_ID=$(curl -k -s -X GET "https://$PORTAL_ENDPOINT/admin/api/services.json?access_token=$TOKEN" | jq ".services[] | select(.service.system_name | contains(\"$SYSTEM_NAME\")) | .service.id")
```

   在这里，我们在请求中使用 **jq** 命令行 JSON 解析器工具。

   b. 导入策略：

```
3scale policies import -f /path/to/policies_config.json -d=<admin_portal_url> 3scale-kafka-bridge
```

   将 **/path/to/policies_config.json** 替换为策略配置文件的路径，将 & **lt;admin_portal_url** > 替换为 3scale 管理门户的端点的路径。

9. 在 3scale 管理门户中，检查 Kafka Bridge 服务的端点和策略是否已加载。

10. 在 3scale Toolbox 中创建应用程序计划和应用程序。
需要应用程序才能获取用于身份验证的用户密钥。

11. （生产环境步骤）要在生产环境的网关中使用 API，请提升配置：

```
3scale proxy-config promote $REMOTE_NAME $SERVICE_ID
```

12. 使用 API 测试工具来验证您可以通过 APIcast 网关访问 Kafka 网桥，方法是使用调用来创建消费者，以及为应用程序创建的用户密钥。
例如：

```
https//my-project-my-bridge-bridge-service-3scale-apicast-
staging.example.com:443/consumers/my-group?
user_key=3dfc188650101010ecd7fdc56098ce95
```

如果从 Kafka Bridge 返回有效负载，则已成功创建使用者。

```
{
  "instance_id": "consumer1",
  "base uri": "https//my-project-my-bridge-bridge-service-3scale-apicast-
staging.example.com:443/consumers/my-group/instances/consumer1"
}
```

Base URI 是客户端在后续请求中使用的地址。

附录 A. 使用您的订阅

Apache Kafka 的流通过软件订阅提供。要管理您的订阅，请访问红帽客户门户中的帐户。

访问您的帐户

1. 转至 access.redhat.com。
2. 如果您还没有帐户，请创建一个帐户。
3. 登录到您的帐户。

激活订阅

1. 转至 access.redhat.com。
2. 导航到 **My Subscriptions**。
3. 导航到 **激活订阅** 并输入您的 16 位激活号。

下载 Zip 和 Tar 文件

要访问 zip 或 tar 文件，请使用客户门户网站查找下载的相关文件。如果您使用 RPM 软件包，则不需要这一步。

1. 打开浏览器并登录红帽客户门户网站 **产品下载页面**，网址为 access.redhat.com/downloads。
2. 在 **INTEGRATION AND AUTOMATION** 目录中找到 **Apache Kafka for Apache Kafka**的流。
3. 选择 Apache Kafka 产品所需的流。此时会打开 **Software Downloads** 页面。
4. 单击组件的 **Download** 链接。

使用 DNF 安装软件包

要安装软件包以及所有软件包的依赖软件包，请使用：

```
dnf install <package_name>
```

要从本地目录中安装之前下载的软件包，请使用：

```
dnf install <path_to_download_package>
```

更新于 2024-04-30