



Red Hat AMQ Streams 2.3

AMQ Streams Kafka Bridge での 3scale API Management の使用

3scale で利用可能な機能を使用する

Red Hat AMQ Streams 2.3 AMQ Streams Kafka Bridge での 3scale API Management の使用

3scale で利用可能な機能を使用する

法律上の通知

Copyright © 2023 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.

概要

Red Hat 3scale API Management をデプロイし、OpenShift Container Platform に AMQ Streams Kafka Bridge をデプロイして統合します。

目次

多様性を受け入れるオープンソースの強化	3
第1章 3SCALE API MANAGEMENT	4
1.1. KAFKA BRIDGE のサービス検出	4
1.2. 3SCALE APICAST ゲートウェイポリシー	4
1.3. TLS 検証用の 3SCALE APICAST	6
1.4. 既存の 3SCALE デプロイメントを使用する場合	6
第2章 KAFKA BRIDGE を使用するための 3SCALE のデプロイメント	7
付録A サブスクリプションの使用	11
アカウントへのアクセス	11
サブスクリプションのアクティベート	11
Zip および Tar ファイルのダウンロード	11
DNF を使用したパッケージのインストール	11

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#) をご覧ください。

第1章 3SCALE API MANAGEMENT

Kafka Bridge を OpenShift Container Platform にデプロイした場合は、3scale で使用できます。

Kafka Bridge のプレーンデプロイメントでは、認証または承認のプロビジョニングがなく、TLS 暗号化による外部クライアントへの接続はサポートされません。3scale API Management は、TLS を使用して Kafka Bridge を保護し、認証および認可を提供できます。また、3scale との統合により、メトリクス、流量制御、請求などの追加機能も利用できるようになります。

3scale では、AMQ Streams へのアクセスを希望する外部クライアントからのリクエストに対して、各種タイプの認証を使用できます。3scale では、以下のタイプの認証がサポートされます。

標準 API キー

識別子およびシークレットトークンとして機能する、ランダムな単一文字列またはハッシュ。

アプリケーション ID とキーのペア

イミュータブルな識別子およびミュータブルなシークレットキー文字列。

OpenID Connect

委譲された認証のプロトコル。

1.1. KAFKA BRIDGE のサービス検出

3scale は、サービス検出を使用して統合されますが、これには 3scale が AMQ Streams および Kafka Bridge と同じ OpenShift クラスターにデプロイされている必要があります。

AMQ Streams Cluster Operator デプロイメントには、以下の環境変数が設定されている必要があります。

- STRIMZI_CUSTOM_KAFKA_BRIDGE_SERVICE_LABELS
- STRIMZI_CUSTOM_KAFKA_BRIDGE_SERVICE_ANNOTATIONS

Kafka Bridge をデプロイすると、Kafka Bridge の REST インターフェイスを公開するサービスは、3scale による検出にアノテーションとラベルを使用します。

- 3scale によって **discovery.3scale.net=true** ラベルが使用され、サービスが検出されます。
- アノテーションによってサービスに関する情報が提供されます。

OpenShift コンソールで設定を確認するには、Kafka Bridge インスタンスの **Services** に移動します。Annotations に、Kafka Bridge の OpenAPI 仕様へのエンドポイントが表示されます。

1.2. 3SCALE APICAST ゲートウェイポリシー

3scale は 3scale APIcast と組み合わせて使用されます。これは、3scale と共にデプロイされた API ゲートウェイであり、Kafka Bridge への単一のエントリーポイントを提供します。

APIcast ポリシーは、ゲートウェイの動作をカスタマイズするメカニズムを提供します。3scale には、ゲートウェイ設定のための標準ポリシーのセットが含まれています。また、独自のポリシーを作成することもできます。

APIcast ポリシーに関する詳細は、[Red Hat 3scale のドキュメント](#) を参照してください。

Kafka Bridge の APIcast ポリシー

3scale と Kafka Bridge との統合のポリシー設定例は `policies_config.json` ファイルに含まれており、このファイルでは以下を定義します。

- Anonymous Access (匿名アクセス)
- Header Modification (ヘッダー変更)
- Routing (ルーティング)
- URL Rewriting (URL の書き換え)

ゲートウェイポリシーは、このファイルを使用して有効または無効に設定します。

この例をひな形として使用し、独自のポリシーを定義できます。

Anonymous Access (匿名アクセス)

Anonymous Access ポリシーでは、認証をせずにサービスが公開され、HTTP クライアントがデフォルトのクレデンシャル (匿名アクセス用) を提供しない場合に、このポリシーによって提供されません。このポリシーは必須ではなく、認証が常に必要であれば無効または削除できます。

Header Modification (ヘッダー変更)

Header Modification ポリシーを使用すると、既存の HTTP ヘッダーを変更したり、ゲートウェイを通過するリクエストまたはレスポンスへ新規ヘッダーを追加したりすることができます。3scale の統合では、このポリシーによって、HTTP クライアントから Kafka Bridge までゲートウェイを通過するすべてのリクエストにヘッダーが追加されます。

Kafka Bridge は、新規コンシューマー作成のリクエストを受信すると、URI のある `base_uri` フィールドが含まれる JSON ペイロードを返します。コンシューマーは後続のすべてのリクエストにこの URI を使用する必要があります。以下に例を示します。

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

APIcast を使用する場合、クライアントは以降のリクエストをすべてゲートウェイに送信し、Kafka Bridge には直接送信しません。そのため URI には、ゲートウェイの背後にある Kafka Bridge のアドレスではなく、ゲートウェイのホスト名が必要です。

Header Modification ポリシーを使用すると、ヘッダーが HTTP クライアントからリクエストに追加されるので、Kafka Bridge はゲートウェイホスト名を使用します。

たとえば、**Forwarded: host=my-gateway:80;proto=http** ヘッダーを適用すると、Kafka Bridge は以下をコンシューマーに提供します。

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

X-Forwarded-Path ヘッダーには、クライアントからゲートウェイへのリクエストに含まれる元のパスが含まれています。このヘッダーは、ゲートウェイが複数の Kafka Bridge インスタンスをサポートする場合に適用される Routing ポリシーに密接に関連します。

Routing (ルーティング)

Routing ポリシーは、複数の Kafka Bridge インスタンスがある場合に適用されます。コンシュー

マーが最初に作成された Kafka Bridge インスタンスにリクエストを送信する必要があるため、適切な Kafka Bridge インスタンスにリクエストを転送するようゲートウェイのルートを一括的に指定する必要があります。

Routing ポリシーは各ブリッジインスタンスに名前を付け、ルーティングはその名前を使用して実行されます。Kafka Bridge のデプロイ時に、**KafkaBridge** カスタムリソースで名前を指定します。

たとえば、コンシューマーから以下への各リクエスト (**X-Forwarded-Path** を使用) について考えてみましょう。

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

この場合、各リクエストは以下に転送されます。

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

URL Rewriting ポリシーはブリッジ名を削除しますが、これは、リクエストをゲートウェイから Kafka Bridge に転送するときにこのポリシーが使用されないからです。

URL Rewriting (URL の書き換え)

URL Rewriting ポリシーは、ゲートウェイから Kafka Bridge にリクエストが転送される時、クライアントから特定の Kafka Bridge インスタンスへのリクエストにブリッジ名が含まれないようにします。

ブリッジ名は、ブリッジが公開するエンドポイントで使用されません。

1.3. TLS 検証用の 3SCALE APICAST

TLS の検証用に APICast を設定できます。これにはテンプレートを使用した APICast の自己管理によるデプロイメントが必要になります。**apicast** サービスがルートとして公開されます。

TLS ポリシーを Kafka Bridge API に適用することもできます。

1.4. 既存の 3SCALE デプロイメントを使用する場合

3scale がすでに OpenShift にデプロイされており、Kafka Bridge と併用する場合は、[Kafka Bridge を使用するための 3scale のデプロイメント](#) に記載されているとおりに設定されていることを確認してください。

第2章 KAFKA BRIDGE を使用するための 3SCALE のデプロイメント

3scale を Kafka Bridge で使用するには、まず 3scale をデプロイし、次に Kafka Bridge API の検出を設定します。

また、3scale APIcast および 3scale toolbox も使用します。

- APIcast は、HTTP クライアントが Kafka Bridge API サービスに接続するための NGINX ベースの API ゲートウェイとして、3scale により提供されます。
- 3scale toolbox は設定ツールで、Kafka Bridge サービスの OpenAPI 仕様を 3scale にインポートするために使用されます。

このシナリオでは、AMQ Streams、Kafka、Kafka Bridge、および 3scale/APIcast を、同じ OpenShift クラスターで実行します。



注記

3scale がすでに Kafka Bridge と同じクラスターにデプロイされている場合は、デプロイメントの手順を省略して、現在のデプロイメントを使用できます。

前提条件

- 3scale について理解している
- [AMQ Streams](#) および [Kafka](#) が稼働している
- [Kafka Bridge](#) がデプロイされている

3scale デプロイメントの場合:

- [Red Hat 3scale API Management Supported Configurations](#) を確認する
- インストール用に **cluster-admin** ロール (**system:admin** など) を持つユーザーを用意する
- 以下が記述されている JSON ファイルにアクセスできる
 - Kafka Bridge OpenAPI 仕様 (**openAPIV2.json**)
 - Kafka Bridge のヘッダー変更および Routing ポリシー (**policies_config.json**)
[GitHub](#) で JSON ファイルを探します。

詳細については、[Red Hat 3scale のドキュメント](#) を参照してください。

手順

1. 3scale API Management を OpenShift クラスターにデプロイします。
 - a. 新規プロジェクトを作成するか、または既存プロジェクトを使用します。

```
oc new-project my-project \  
  --description="description" --display-name="display_name"
```

- b. 3scale をデプロイします。

[Red Hat 3scale のドキュメント](#) では、テンプレートまたは operator を使用して OpenShift に 3scale をデプロイする方法について説明します。

どの方法を使用する場合も、**WILDCARD_DOMAIN** パラメーターが OpenShift クラスターのドメインに設定されていることを確認してください。

3scale 管理ポータルにアクセスするために表示される URL およびクレデンシャルを書き留めておきます。

2. 3scale が Kafka Bridge サービスを検出するように承認を付与します。

```
oc adm policy add-cluster-role-to-user view system:serviceaccount:my-project:amp
```

3. 3scale が OpenShift コンソールまたは CLI から Openshift クラスターに正常にデプロイされたことを確認します。
以下に例を示します。

```
oc get deployment 3scale-operator
```

4. 3scale toolbox を設定します。

- a. [Red Hat 3scale のドキュメント](#) に記載されている情報を使用して、3scale toolbox をインストールします。
- b. 3scale と対話できるように環境変数を設定します。

```
export REMOTE_NAME=stirzmi-kafka-bridge ❶
export SYSTEM_NAME=stirzmi_http_bridge_for_apache_kafka ❷
export TENANT=stirzmi-kafka-bridge-admin ❸
export PORTAL_ENDPOINT=$TENANT.3scale.net ❹
export TOKEN=3scale access token ❺
```

- ❶ **REMOTE_NAME** は、3scale 管理ポータルのリモートアドレスに割り当てられた名前です。
- ❷ **SYSTEM_NAME** は、3scale toolbox で OpenAPI 仕様をインポートして作成される 3scale サービス/API の名前です。
- ❸ **TENANT** は、3scale 管理ポータルのテナント名です ([https://\\$TENANT.3scale.net](https://$TENANT.3scale.net))。
- ❹ **PORTAL_ENDPOINT** は、3scale 管理ポータルを実行するエンドポイントです。
- ❺ **TOKEN** は、3scale toolbox または HTTP リクエストを介して対話するために 3scale 管理ポータルによって提供されるアクセストークンです。

- c. 3scale toolbox のリモート Web アドレスを設定します。

```
3scale remote add $REMOTE_NAME https://$TOKEN@$PORTAL_ENDPOINT/
```

これで、toolbox を実行するたびに、3scale 管理ポータルのエンドポイントアドレスを指定する必要がなくなりました。

5. Cluster Operator デプロイメントに、3scale が Kafka Bridge サービスを検出するために必要なラベルプロパティおよびアノテーションプロパティがあることを確認します。

```
#...
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
#...
```

これらのプロパティーがない場合は、OpenShift コンソールからプロパティーを追加するか、[Cluster Operator](#) および [Kafka Bridge](#) を再デプロイします。

6. 3scale で Kafka Bridge API サービスを検出します。
 - a. 3scale をデプロイしたときに提供されたクレデンシャルを使用して、3scale 管理ポータルにログインします。
 - b. 管理ポータルダッシュボードの **APIs** から、**Create Product** をクリックします。
 - c. **Import from OpenShift** をクリックします。
 - d. Kafka Bridge サービスを選択します。
 - e. **Create Product** をクリックします。

ページを更新して Kafka Bridge サービスを表示することが必要な場合もあります。

ここで、サービスの設定をインポートする必要があります。エディターからインポートしますが、ポータルを開いたまま正常にインポートされたことを確認します。

7. OpenAPI 仕様 (JSON ファイル) の **Host** フィールドを編集して、Kafka Bridge サービスのベース URL を使用します。
以下に例を示します。

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

host URL に以下が正しく含まれることを確認します。

- Kafka Bridge 名 (**my-bridge**)
- プロジェクト名 (**my-project**)
- Kafka Bridge のポート (**8080**)

8. 3scale toolbox を使用して、更新された OpenAPI 仕様をインポートします。

```
3scale import openapi -k -d $REMOTE_NAME openapiv2.json -t myproject-my-bridge-bridge-service
```

9. サービスの Header Modification および Routing ポリシー (JSON ファイル) をインポートします。
 - a. 3scale で作成したサービスの ID を特定します。

ここでは、`jq` ユーティリティーを使用します。

```
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")
```

ポリシーをインポートするときにこの ID が必要です。

- b. ポリシーをインポートします。

```
curl -k -X PUT
"https://$PORTAL_ENDPOINT/admin/api/services/$SERVICE_ID/proxy/policies.json" --
data "access_token=$TOKEN" --data-urlencode policies_config@policies_config.json
```

10. 3scale 管理ポータルから、**Integration** → **Configuration** に移動し、Kafka Bridge サービスのエンドポイントとポリシーが読み込まれていることを確認します。
11. アプリケーションプランを作成するために、**Applications** → **Create Application Plan** に移動します。
12. アプリケーションを作成するために、**Audience** → **Developer** → **Applications** → **Create Application** に移動します。
認証のユーザーキーを取得するためにアプリケーションが必要になります。
13. 実稼働環境用の手順: 実稼働環境のゲートウェイで API を利用可能にするには、設定をプロモートします。

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

14. API テストツールを使用して、コンシューマーの作成に呼び出しを使用する APIcast ゲートウェイと、アプリケーションに作成されたユーザーキーで、Kafka Bridge にアクセスできることを検証します。
以下に例を示します。

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

Kafka Bridge からペイロードが返されれば、コンシューマーが正常に作成されています。

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

ベース URI は、クライアントが以降のリクエストで使用するアドレスです。

付録A サブスクリプションの使用

AMQ Streams は、ソフトウェアサブスクリプションから提供されます。サブスクリプションを管理するには、Red Hat カスタマーポータルでアカウントにアクセスします。

アカウントへのアクセス

1. access.redhat.com に移動します。
2. アカウントがない場合は、作成します。
3. アカウントにログインします。

サブスクリプションのアクティベート

1. access.redhat.com に移動します。
2. **My Subscriptions** に移動します。
3. **Activate a subscription** に移動し、16 桁のアクティベーション番号を入力します。

Zip および Tar ファイルのダウンロード

zip または tar ファイルにアクセスするには、カスタマーポータルを使用して、ダウンロードする関連ファイルを検索します。RPM パッケージを使用している場合は、この手順は必要ありません。

1. ブラウザーを開き、access.redhat.com/downloads で Red Hat カスタマーポータルの **製品のダウンロード** ページにログインします。
2. **INTEGRATION AND AUTOMATION** カテゴリで、**AMQ Streams for Apache Kafka** エントリーを見つけます。
3. 必要な AMQ Streams 製品を選択します。 **Software Downloads** ページが開きます。
4. コンポーネントの **Download** リンクをクリックします。

DNF を使用したパッケージのインストール

パッケージとすべてのパッケージ依存関係をインストールするには、以下を使用します。

```
dnf install <package_name>
```

ローカルディレクトリーからダウンロード済みのパッケージをインストールするには、以下を使用します。

```
dnf install <path_to_download_package>
```

改訂日時: 2023-04-06