Red Hat AMQ Streams 2.4

Using 3scale API Management with the AMQ Streams Kafka Bridge

Use the features and functions available with 3scale

Last Updated: 2023-05-19

Red Hat AMQ Streams 2.4 Using 3scale API Management with the AMQ Streams Kafka Bridge

Use the features and functions available with 3scale

Legal Notice

Abstract

Deploy and integrate Red Hat 3scale API Management with a deployment of AMQ Streams Kafka Bridge on OpenShift Container Platform.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE 3

CHAPTER 1. 3SCALE API MANAGEMENT 4
1.1. KAFKA BRIDGE SERVICE DISCOVERY 4
1.2. 3SCALE APICAST GATEWAY POLICIES 4
1.3. 3SCALE APICAST FOR TLS VALIDATION 6
1.4. USING AN EXISTING 3SCALE DEPLOYMENT 6

CHAPTER 2. DEPLOYING 3SCALE FOR THE KAFKA BRIDGE 7

APPENDIX A. USING YOUR SUBSCRIPTION 11
Accessing Your Account 11
Activating a Subscription 11
Downloading Zip and Tar Files 11
Installing packages with DNF 11

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

CHAPTER 1. 3SCALE API MANAGEMENT

If you deployed the Kafka Bridge on OpenShift Container Platform, you can use it with 3scale.

With a plain deployment of the Kafka Bridge, there is no provision for authentication or authorization, and no support for a TLS encrypted connection to external clients. 3scale API Management can secure the Kafka Bridge with TLS, and provide authentication and authorization. Integration with 3scale also means that additional features like metrics, rate limiting and billing are available.

With 3scale, you can use different types of authentication for requests from external clients wishing to access AMQ Streams. 3scale supports the following types of authentication:

Standard API Keys

Single randomized strings or hashes acting as an identifier and a secret token.

Application Identifier and Key pairs

Immutable identifier and mutable secret key strings.

OpenID Connect

Protocol for delegated authentication.

1.1. KAFKA BRIDGE SERVICE DISCOVERY

3scale is integrated using service discovery, which requires that 3scale is deployed to the same OpenShift cluster as AMQ Streams and the Kafka Bridge.

Your AMQ Streams Cluster Operator deployment must have the following environment variables set:

- STRIMZI_CUSTOM_KAFKA_BRIDGE_SERVICE_LABELS
- STRIMZI_CUSTOM_KAFKA_BRIDGE_SERVICE_ANNOTATIONS

When the Kafka Bridge is deployed, the service that exposes the REST interface of the Kafka Bridge uses the annotations and labels for discovery by 3scale.

- A **discovery.3scale.net=true** label is used by 3scale to find a service.
- Annotations provide information about the service.

You can check your configuration in the OpenShift console by navigating to **Services** for the Kafka Bridge instance. Under **Annotations** you will see the endpoint to the OpenAPI specification for the Kafka Bridge.

1.2. 3SCALE APICAST GATEWAY POLICIES

3scale is used in conjunction with 3scale APIcast, an API gateway deployed with 3scale that provides a single point of entry for the Kafka Bridge.

APIcast policies provide a mechanism to customize how the gateway operates. 3scale provides a set of standard policies for gateway configuration. You can also create your own policies.

For more information on APIcast policies, see the Red Hat 3scale documentation.

APIcast policies for the Kafka Bridge

A sample policy configuration for 3scale integration with the Kafka Bridge is provided with the **policies_config.json** file, which defines:

- Anonymous access
- Header modification
- Routing
- URL rewriting

Gateway policies are enabled or disabled through this file.

You can use this sample as a starting point for defining your own policies.

Anonymous access

The anonymous access policy exposes a service without authentication, providing default credentials (for anonymous access) when a HTTP client does not provide them. The policy is not mandatory and can be disabled or removed if authentication is always needed.

Header modification

The header modification policy allows existing HTTP headers to be modified, or new headers added to requests or responses passing through the gateway. For 3scale integration, the policy adds headers to every request passing through the gateway from a HTTP client to the Kafka Bridge.
When the Kafka Bridge receives a request for creating a new consumer, it returns a JSON payload containing a **base_uri** field with the URI that the consumer must use for all the subsequent requests. For example:

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

When using APIcast, clients send all subsequent requests to the gateway and not to the Kafka Bridge directly. So the URI requires the gateway hostname, not the address of the Kafka Bridge behind the gateway.

Using header modification policies, headers are added to requests from the HTTP client so that the Kafka Bridge uses the gateway hostname.

For example, by applying a **Forwarded: host=my-gateway:80;proto=http** header, the Kafka Bridge delivers the following to the consumer.

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

An **X-Forwarded-Path** header carries the original path contained in a request from the client to the gateway. This header is strictly related to the routing policy applied when a gateway supports more than one Kafka Bridge instance.

Routing

A routing policy is applied when there is more than one Kafka Bridge instance. Requests must be sent to the same Kafka Bridge instance where the consumer was initially created, so a request must specify a route for the gateway to forward a request to the appropriate Kafka Bridge instance.
A routing policy names each bridge instance, and routing is performed using the name. You specify the name in the **KafkaBridge** custom resource when you deploy the Kafka Bridge.

For example, each request (using **X-Forwarded-Path**) from a consumer to:

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

is forwarded to:

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

URL rewriting policy removes the bridge name, as it is not used when forwarding the request from the gateway to the Kafka Bridge.

URL rewriting

The URL rewiring policy ensures that a request to a specific Kafka Bridge instance from a client does not contain the bridge name when forwarding the request from the gateway to the Kafka Bridge.
The bridge name is not used in the endpoints exposed by the bridge.

1.3. 3SCALE APICAST FOR TLS VALIDATION

You can set up APIcast for TLS validation, which requires a self-managed deployment of APIcast using a template. The **apicast** service is exposed as a route.

You can also apply a TLS policy to the Kafka Bridge API.

1.4. USING AN EXISTING 3SCALE DEPLOYMENT

If you already have 3scale deployed to OpenShift and you wish to use it with the Kafka Bridge, ensure that you have the setup described in Deploying 3scale for the Kafka Bridge .

CHAPTER 2. DEPLOYING 3SCALE FOR THE KAFKA BRIDGE

In order to use 3scale with the Kafka Bridge, you first deploy it and then configure it to discover the Kafka Bridge API.

You will also use 3scale APIcast and 3scale toolbox.

- APIcast is provided by 3scale as an NGINX-based API gateway for HTTP clients to connect to the Kafka Bridge API service.
- 3scale toolbox is a configuration tool that is used to import the OpenAPI specification for the Kafka Bridge service to 3scale.

In this scenario, you run AMQ Streams, Kafka, the Kafka Bridge, and 3scale/APIcast in the same OpenShift cluster.

NOTE

If you already have 3scale deployed in the same cluster as the Kafka Bridge, you can skip the deployment steps and use your current deployment.

Prerequisites

- An understanding of 3scale
- AMQ Streams and Kafka is running
- The Kafka Bridge is deployed

For the 3scale deployment:

- Check the Red Hat 3scale API Management supported configurations .
- Installation requires a user with **cluster-admin** role, such as **system:admin**.
- You need access to the JSON files describing the:
    - Kafka Bridge OpenAPI specification (**openapiv2.json**)
    - Header modification and routing policies for the Kafka Bridge (**policies_config.json**)
      Find the JSON files on GitHub.

For more information, see the Red Hat 3scale documentation .

Procedure

1. Deploy 3scale API Management to the OpenShift cluster.
    a. Create a new project or use an existing project.

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

    b. Deploy 3scale.
    The Red Hat 3scale documentation describes how to deploy 3scale on OpenShift using a template or operator.

Whichever approach you use, make sure that you set the *WILDCARD_DOMAIN* parameter to the domain of your OpenShift cluster.

Make a note of the URLS and credentials presented for accessing the 3scale Admin Portal.

2. Grant authorization for 3scale to discover the Kafka Bridge service:

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

3. Verify that 3scale was successfully deployed to the Openshift cluster from the OpenShift console or CLI.
For example:

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

4. Set up 3scale toolbox.

   a. Use the information provided in the Red Hat 3scale documentation to install 3scale toolbox.

   b. Set environment variables to be able to interact with 3scale:

```
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 access token (5)
```

(1) **REMOTE_NAME** is the name assigned to the remote address of the 3scale Admin Portal.

(2) **SYSTEM_NAME** is the name of the 3scale service/API created by importing the OpenAPI specification through the 3scale toolbox.

(3) **TENANT** is the tenant name of the 3scale Admin Portal (that is, **https://$TENANT.3scale.net**).

(4) **PORTAL_ENDPOINT** is the endpoint running the 3scale Admin Portal.

(5) **TOKEN** is the access token provided by the 3scale Admin Portal for interaction through the 3scale toolbox or HTTP requests.

   c. Configure the remote web address of the 3scale toolbox:

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

   Now the endpoint address of the 3scale Admin Portal does not need to be specified every time you run the toolbox.

5. Check that your Cluster Operator deployment has the labels and annotations properties required for the Kafka Bridge service to be discovered by 3scale.

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

If not, add the properties through the OpenShift console or try redeploying the Cluster Operator and the Kafka Bridge.

6. Discover the Kafka Bridge API service through 3scale.

   a. Log in to the 3scale Admin Portal using the credentials provided when 3scale was deployed.

   b. From **APIs** on the Admin Portal Dashboard, click **Create Product**.

   c. Click **Import from OpenShift**.

   d. Choose the Kafka Bridge service

   e. Click **Create Product**.
   You may need to refresh the page to see the Kafka Bridge service.

   Now you need to import the configuration for the service. You do this from an editor, but keep the portal open to check the imports are successful.

7. Edit the **Host** field in the OpenAPI specification (JSON file) to use the base URL of the Kafka Bridge service:
   For example:

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

   Check the **host** URL includes the correct:

   - Kafka Bridge name (*my-bridge*)
   - Project name (*my-project*)
   - Port for the Kafka Bridge (*8080*)

8. Import the updated OpenAPI specification using the 3scale toolbox:

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

9. Import the header modification and routing policies for the service (JSON file).

   a. Locate the ID for the service you created in 3scale.
   Here we use the `jq` utility:

```
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")
```

You need the ID when importing the policies.

b. Import the policies:

```
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. From the 3scale Admin Portal, navigate to **Integration → Configuration** to check that the endpoints and policies for the Kafka Bridge service have loaded.

11. Navigate to **Applications → Create Application Plan** to create an application plan.

12. Navigate to **Audience → Developer → Applications → Create Application** to create an application.
The application is required in order to obtain a user key for authentication.

13. (Production environment step) To make the API available to the production gateway, promote the configuration:

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

14. Use an API testing tool to verify you can access the Kafka Bridge through the APIcast gateway using a call to create a consumer, and the user key created for the application.
For example:

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

If a payload is returned from the Kafka Bridge, the consumer was created successfully.

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

The base URI is the address that the client will use in subsequent requests.

APPENDIX A. USING YOUR SUBSCRIPTION

AMQ Streams is provided through a software subscription. To manage your subscriptions, access your account at the Red Hat Customer Portal.

Accessing Your Account

1. Go to access.redhat.com.
2. If you do not already have an account, create one.
3. Log in to your account.

Activating a Subscription

1. Go to access.redhat.com.
2. Navigate to **My Subscriptions**.
3. Navigate to **Activate a subscription** and enter your 16-digit activation number.

Downloading Zip and Tar Files

To access zip or tar files, use the customer portal to find the relevant files for download. If you are using RPM packages, this step is not required.

1. Open a browser and log in to the Red Hat Customer Portal **Product Downloads** page at access.redhat.com/downloads.
2. Locate the **AMQ Streams for Apache Kafka**entries in the **INTEGRATION AND AUTOMATION** category.
3. Select the desired AMQ Streams product. The **Software Downloads** page opens.
4. Click the **Download** link for your component.

Installing packages with DNF

To install a package and all the package dependencies, use:

```
dnf install <package_name>
```

To install a previously-downloaded package from a local directory, use:

```
dnf install <path_to_download_package>
```

Revised on 2023-05-19 08:02:10 UTC