Red Hat AMQ 2020.Q4

OpenShift 中 AMQ Streams 1.6 发行注记

用于 OpenShift Container Platform 上的 AMQ Streams

Last Updated: 2022-02-14

Red Hat AMQ 2020.Q4 OpenShift 中 AMQ Streams 1.6 发行注记

用于 OpenShift Container Platform 上的 AMQ Streams

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

这些发行注记包含 AMQ Streams 1.6 发行版中包含的新功能、增强、修复和问题的最新信息。

目录

第 1 章 功能

AMQ Streams 版本 1.6 基于 Strimzi 0.20.x。

本发行版本中添加的功能，以及之前 AMQ Streams 版本中没有的功能，如下所述。

注意

要查看此版本中已解决的所有增强和错误，请查看 AMQ Streams Jira 项目 。

1.1. AMQ STREAMS 1.6.X 中的 KAFKA 支持（OCP 3.11 的长期支持）

本节论述了 AMQ Streams 1.6 以及后续补丁版本中支持的 Kafka 和 ZooKeeper 版本。

AMQ Streams 1.6.x 是与 OCP 3.11 一起使用的 Long Term Support 版本，且仅在支持 OpenShift Container Platform 3.11 的情况下被支持。

注意

仅 OCP 3.11 支持 AMQ Streams 1.6.4 及更新的版本。如果使用 OCP 4.x，则需要升级到 AMQ Streams 1.7.x 或更高版本。

有关 AMQ LTS 版本支持日期的信息，请参阅红帽知识库解决方案 AMQ LTS 版本的支持时间？

仅支持由红帽构建的 Kafka 发行版本。对于升级目的，AMQ Streams 1.6.x 支持早期版本的 Kafka。

有关支持的 Kafka 版本的更多信息，请参阅 客户门户网站中的 Red Hat AMQ 7 组件详情页 。

1.1.1. Kafka 支持 AMQ Streams 1.6.6 和 1.6.7

AMQ Streams 1.6.6 和 1.6.7 发行版本支持 Apache Kafka 版本 2.6.3。

在将代理和客户端应用程序升级到 Kafka 2.6.3 前，您必须升级 Cluster Operator。有关升级说明，请参阅 AMQ Streams 和 Kafka 升级 。

Kafka 2.6.3 需要 ZooKeeper 版本 3.5.9。因此，当从 AMQ Streams 1.6.4 / 1.6.5 升级时，Cluster Operator *不会执行* ZooKeeper 升级。

如需了解更多信息，请参阅 Kafka 2.6.3 发行注记。

1.1.2. AMQ Streams 1.6.4 和 1.6.5 中的 Kafka 支持

AMQ Streams 1.6.4 和 1.6.5 发行版本支持 Apache Kafka 版本 2.6.2 和 ZooKeeper 3.5.9。

您必须升级 Cluster Operator，然后才能将代理和客户端应用程序升级到 Kafka 2.6.2。有关升级说明，请参阅 AMQ Streams 和 Kafka 升级 。

Kafka 2.6.2 需要 ZooKeeper 版本 3.5.9。因此，当从 AMQ Streams 1.6.2 升级时，Cluster Operator 将执行 ZooKeeper 升级。

如需更多信息，请参阅 Kafka 2.6.2 发行注记。

1.1.3. AMQ Streams 1.6.0 和 1.6.2 中的 Kafka 支持

AMQ Streams 1.6.0 和 1.6.2 支持 Apache Kafka 版本 2.6.0。

在将代理和客户端应用程序升级到 Kafka 2.6.0 之前，您必须升级 Cluster Operator。有关升级说明，请参阅 AMQ Streams 和 Kafka 升级 。

如需更多信息，请参阅 Kafka 2.5.0 和 Kafka 2.6.0 发行注记。

Kafka 2.6.0 需要与 Kafka 2.5.x 相同的 ZooKeeper 版本（ZooKeeper 版本 3.5.7 / 3.5.8）。因此，当从 AMQ Streams 1.5 升级时，Cluster Operator *不会执行* ZooKeeper 升级。

1.2. 容器镜像移到 JAVA 11

AMQ Streams Streams 容器镜像作为 Java 运行时环境(JRE)移到 Java 11。镜像中的 JRE 版本从 OpenJDK 8 改为 OpenJDK 11。

1.3. CLUSTER OPERATOR 日志

Cluster Operator 日志记录现在使用部署 Cluster Operator 时自动创建的 ConfigMap 进行配置。以下新的 YAML 文件中描述了 ConfigMap：

```
install/cluster-operator/050-ConfigMap-strimzi-cluster-operator.yaml
```

配置 Cluster Operator 日志：

1. 在 **050-ConfigMap-strimzi-cluster-operator.yaml** 文件中，编辑 **data.log4j2.properties** 字段：

   Cluster Operator 日志配置示例

   ```
   kind: ConfigMap
   apiVersion: v1
   metadata:
     name: strimzi-cluster-operator
     labels:
       app: strimzi
   data:
     log4j2.properties: |
       name = COConfig
       monitorInterval = 30
       appender.console.type = Console
       appender.console.name = STDOUT
       # ...
   ```

2. 更新自定义资源：

   ```
   oc apply -f install/cluster-operator/050-ConfigMap-strimzi-cluster-operator.yaml
   ```

要更改重新加载日志的频率，请在 **monitorInterval** 字段中设置一个以秒为单位的时间（默认的重新加载时间为 30 秒）。

注意

因此，**STRIMZI_LOG_LEVEL** 环境变量已从 **060-Deployment-strimzi-cluster-operator.yaml** 文件中被删除。改为在 ConfigMap 中设置日志级别。

请参阅 Cluster Operator 配置。

1.4. OAUTH 2.0 授权

对 OAuth 2.0 授权的支持从技术预览转变为 AMQ Streams 的通用组件。

如果您使用 OAuth 2.0 进行基于令牌的身份验证，您现在可以使用 OAuth 2.0 授权规则来限制客户端对 Kafka 代理的访问。

AMQ Streams 支持通过 Red Hat Single Sign-On Authorization Services 使用基于 OAuth 2.0 令牌的授权，它允许您集中管理安全策略和权限。

Red Hat Single Sign-On 中定义的安全策略和权限用于授予对 Kafka 代理上资源的访问权限。用户和客户端与允许对 Kafka 代理执行特定操作的策略进行匹配。

请参阅使用 OAuth 2.0 基于令牌的授权。

1.5. 开源策略代理(OPA)集成

开源策略代理(OPA)是一个开源策略引擎。您可以将 OPA 与 AMQ Streams 集成，作为基于策略的授权机制，允许在 Kafka 代理上进行客户端操作。

从客户端发出请求时，OPA 将根据为 Kafka 访问定义的策略评估请求，然后允许或拒绝请求。

您可以为 Kafka 集群、消费者组和主题定义访问控制。例如，您可以定义一个授权策略，允许从制作者客户端写入到特定代理主题的访问。

请参阅 **KafkaAuthorizationOpa** 模式参考

注意

红帽不支持 OPA 服务器。

1.6. 用于更改数据捕获集成的 DEBEZIUM

红帽 Debezium 是一个分布式更改数据捕获平台。它捕获数据库中的行级更改，创建更改事件记录，并将记录流传输到 Kafka 主题。Debezium 基于 Apache Kafka 构建。您可以将 Debezium 与 AMQ Streams 部署和集成。部署 AMQ Streams 后，您可以通过 Kafka Connect 将 Debezium 部署为连接器配置。Debezium 将更改事件记录传递到 OpenShift 上的 AMQ Streams。应用程序可以读取 *这些更改事件流*，并按发生更改事件的顺序访问更改事件。

Debezium 具有多个用途，包括：

- 数据复制
- 更新缓存和搜索索引
- 简化单体式应用程序
- 数据集成

- 启用流查询

Debezium 为以下通用数据库提供连接器（基于 Kafka Connect）：

- MySQL
- PostgreSQL
- SQL Server
- MongoDB

有关使用 AMQ Streams 部署 Debezium 的更多信息，请参阅 产品文档。

1.7. SERVICE REGISTRY

您可以将服务注册表用作数据流的集中式存储。对于 Kafka，您可以使用 Service Registry 来存储 *Apache Avro* 或 JSON 模式。

服务注册表提供 REST API 和 Java REST 客户端，用于通过服务器端端点从客户端应用注册和查询架构。

使用 Service Registry 将管理模式的过程与客户端应用程序配置分离。您可以通过在客户端代码中指定 URL 来启用应用程序从 registry 中使用 schema。

例如，消息序列化和反序列化的架构可以存储在注册表中，后者随后从使用它们的应用程序引用，以确保它们发送和接收的消息与这些模式兼容。

Kafka 客户端应用程序可以在运行时从 Service Registry 中推送或拉取其模式。

请参阅使用 Service Registry 管理模式。

第 2 章 功能增强

下面概述了本发行版本中添加的增强功能。

2.1. KAFKA 增强

有关引入的增强的概述：

- Kafka 2.6.2，请参阅 Kafka 2.6.2 发行注记 （仅适用于 AMQ Streams 1.6.4）
- Kafka 2.6.1，请参阅 Kafka 2.6.1 发行注记 （仅适用于 AMQ Streams 1.6.4）
- kafka 2.6.0，请参阅 Kafka 2.6.0 发行注记

2.2. KAFKA 网桥的改进

此发行版本包括对 AMQ Streams 的 Kafka Bridge 组件的以下改进。

检索分区和元数据

Kafka Bridge 现在支持以下操作：

- 检索给定主题的分区列表：

```
GET /topics/{topicname}/partitions
```

- 检索给定分区的元数据，如分区 ID、领导代理和副本数：

```
GET /topics/{topicname}/partitions/{partitionid}
```

请参阅 Kafka Bridge API 参考。

支持 Kafka 消息标头

使用 Kafka Bridge 发送的消息现在可以包括 Kafka 消息标头。

在发送到 **/topics 端点** 的 POST 请求中，您可以选择在消息有效负载中指定标头，该标头包含在请求正文中。消息标头值必须采用二进制格式，并以 Base64 格式编码。

带有 Kafka 消息标头的请求示例

```
curl -X POST \
  http://localhost:8080/topics/my-topic \
  -H 'content-type: application/vnd.kafka.json.v2+json' \
  -d '{
    "records": [
        {
            "key": "my-key",
            "value": "sales-lead-0001"
            "partition": 2
            "headers": [
              {
                "key": "key1",
                "value": "QXBhY2hlIEthZmthIGlzIHRoZSBib21iIQ=="
              }
```

```
        ]
      },
    ]
}'
```

请参阅 对 Kafka Bridge 的请求

2.3. OAUTH 2.0 身份验证和授权

此发行版本包括对基于 OAuth 2.0 令牌的身份验证和授权的以下增强。

会话重新身份验证
AMQ Streams 中的 OAuth 2.0 身份验证现在支持 Kafka 代理的 *会话重新身份验证*。这定义了 Kafka 客户端和 Kafka 代理之间经过身份验证的 OAuth 2.0 会话的最长持续时间。会话重新身份验证支持两种类型的令牌验证：快速本地 JWT 和内省端点。

要配置会话重新身份验证，请使用 Kafka 代理的 OAuth 2.0 配置中新的 **maxSecondsWithoutReauthentication** 选项。

对于特定的监听程序，**maxSecondsWithoutReauthentication** 允许您：

- 启用会话重新身份验证
- 设置 Kafka 客户端和 Kafka 代理之间经过身份验证的会话的最大持续时间（以秒为单位）

会话重新验证在 1 小时后的配置示例

```
apiVersion: kafka.strimzi.io/v1beta1
kind: Kafka
spec:
  kafka:
    listeners:
      #...
      - name: tls
        port: 9093
        type: internal
        tls: true
        authentication:
          type: oauth
          maxSecondsWithoutReauthentication: 3600
          # ...
```

如果经过身份验证的会话超过配置的 **maxSecondsWithoutReauthentication**，或者达到访问令牌到期时间，则会关闭它。然后，客户端必须再次登录到授权服务器，获取新的访问令牌，然后重新身份验证到 Kafka 代理。这将在现有连接上创建一个新的经过身份验证的会话。

当下次需要重新身份验证时，客户端尝试的任何操作（不包括重新身份验证）将导致代理终止连接。

请参阅：Kafka 代理的 会话重新身份验证以及 配置 Kafka 代理的 OAuth 2.0 支持。

JWKS 密钥刷新间隔
将 Kafka 代理配置为使用快速本地 JWT 令牌验证时，您现在可以在外部监听程序配置中设置 **jwksMinRefreshPauseSeconds** 选项。这会定义代理尝试刷新授权服务器发布的 JSON Web 密钥集 (JWKS)公钥尝试之间的最小间隔。

在这个版本中，如果 Kafka 代理检测到未知签名密钥，它会尝试立即刷新 JWKS 密钥，而无需等待常规刷新计划。

尝试刷新 JWKS 密钥之间暂停 2 分钟的示例

```
  listeners:
    #...
    - name: external2
      port: 9095
      type: loadbalancer
      tls: false
      authentication:
        type: oauth
        validIssuerUri: <https://<auth-server-address>/auth/realms/external>
        jwksEndpointUri: <https://<auth-server-address>/auth/realms/external/protocol/openid-
connect/certs>
        userNameClaim: preferred_username
        tlsTrustedCertificates:
        - secretName: oauth-server-cert
          certificate: ca.crt
        disableTlsHostnameVerification: true
        jwksExpirySeconds: 360
        jwksRefreshSeconds: 300
        jwksMinRefreshPauseSeconds: 120
        enableECDSA: "true"
```

JWKS 密钥刷新计划在 **jwksRefreshSeconds** 选项中设置。当遇到未知的签名密钥时，会在刷新计划外调度 JWKS 密钥刷新。刷新将持续到自上次刷新以来的时间到达 **jwksMinRefreshPauseSeconds** 中指定的间隔。

jwksMinRefreshPauseSeconds 的默认值为 **1**。

请参阅 为 Kafka 代理配置 OAuth 2.0 支持 。

从 Red Hat Single Sign-On 刷新授权

通过红帽单点登录，为基于 OAuth 2.0 令牌的授权添加了新的配置选项。在配置 Kafka 代理时，现在可以定义以下与从 Red Hat SSO 授权服务刷新授权相关的选项：

- **grantRefreshPeriodSeconds**:连续两次的时间允许刷新运行。默认值为 **60**。如果设置为 **0** 或以下，则禁用刷新授权。
- **grantsRefreshPoolSize**:可以获取的线程数并行为活动会话授予。默认值为 **5**。

请参阅 使用基于 OAuth 2.0 令牌的授权和 配置 OAuth 2.0 授权支持。

在 Red Hat Single Sign-On 中检测权限更改

在这个版本中，**密钥克隆(** Red Hat SSO)授权会定期检查活动会话的权限更改。现在，可以实时检测用户更改和权限管理更改。

2.4. KAFKA 网桥和 CRUISE CONTROL 的指标

现在，您可以在 Kafka Bridge 和 Cruise Control 中添加指标配置。

Kafka Bridge 和 Cruise Control 的指标文件示例随 AMQ Streams 提供，包括：

- 带有指标配置的自定义资源 YAML 文件

- Grafana 仪表板 JSON 文件

部署指标配置以及 Prometheus 和 Grafana 设置后，您可以使用示例 Grafana 仪表板来监控。

AMQ Streams 提供的指标文件示例

```
metrics
├── grafana-dashboards
│   ├── strimzi-cruise-control.json
│   ├── strimzi-kafka-bridge.json
│   ├── strimzi-kafka-connect.json
│   ├── strimzi-kafka-exporter.json
│   ├── strimzi-kafka-mirror-maker-2.json
│   ├── strimzi-kafka.json
│   ├── strimzi-operators.json
│   └── strimzi-zookeeper.json
├── grafana-install
│   └── grafana.yaml
├── prometheus-additional-properties
│   └── prometheus-additional.yaml
├── prometheus-alertmanager-config
│   └── alert-manager-config.yaml
├── prometheus-install
│   ├── alert-manager.yaml
│   ├── prometheus-rules.yaml
│   ├── prometheus.yaml
│   └── strimzi-pod-monitor.yaml
├── kafka-bridge-metrics.yaml
├── kafka-connect-metrics.yaml
├── kafka-cruise-control-metrics.yaml
├── kafka-metrics.yaml
└── kafka-mirror-maker-2-metrics.yaml
```

表 2.1. 带有指标配置的自定义资源示例

组件	自定义资源	YAML 文件示例
Kafka 和 ZooKeeper	**kafka**	**kafka-metrics.yaml**
Kafka Connect	**KafkaConnect** 和 **KafkaConnectS2I**	**kafka-connect-metrics.yaml**
Kafka MirrorMaker 2.0	**KafkaMirrorMaker2**	**kafka-mirror-maker-2-metrics.yaml**
Kafka Bridge	**KafkaBridge**	**kafka-bridge-metrics.yaml**
Sything Control	**kafka**	**kafka-cruise-control-metrics.yaml**

请参阅 向 Kafka 引入指标。

注意

Prometheus 服务器作为 AMQ Streams 分发的一部分不受支持。但是，支持用于公开指标的 Prometheus 端点和 JMX 导出器。

2.5. 日志更改的动态更新

在这个版本中，更改大多数自定义资源的内联和外部的日志级别不再会触发对 Kafka 集群的滚动更新。现在，日志记录更改会被动态应用（无需重启）。

这个增强功能适用于以下资源：

- Kafka 集群
- Kafka Connect 和 Kafka Connect S2I
- Kafka Mirror Maker 2.0
- Kafka Bridge

它不适用于Mirror Maker 或 Cruise Control。

如果您通过 ConfigMap 使用外部日志记录，在更改日志记录附加程序时仍会触发滚动更新。例如：

```
log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
```

请参阅《在 *OpenShift 中使用 AMQ Streams》的" 外部日志记录 "和* "部署配置" 一章。

2.6. 用于指标提取的 PODMONITOR

在这个发行版本中，Prometheus 指标从 pod 中提取的方法（用于 Kafka、ZooKeeper、Kafka Connect 等）已改变。

现在，只有 **PodM** nitors 才会从 pod 中提取指标，该指标在 **strimzi-pod-monitor.yaml** 中定义。在以前的版本中，这由 **ServiceMonitor 和 PodMonitor** 执行。在这个发行版本中，Service **Monitor** 已从 AMQ Streams **中删除**。

您需要升级监控堆栈以使用 **PodMonitor**，如 升级监控堆栈以使用下面的 PodMonitor 所述。

因此，已从与 Kafka 和 ZooKeeper 相关的服务 **中删除了** 以下元素：

- **tcp-prometheus 监控** 端口（端口 9404）
- Prometheus 注解

这个更改适用于以下服务：

- **cluster-name-zookeeper-client**
- **cluster-name-kafka-brokers**

要添加 Prometheus 注解，您现在应使用相关 AMQ Streams 自定义资源中的 **template** 属性，如 自定义 OpenShift 资源 中所述。

升级监控堆栈以使用 PodMonitor

为了避免中断对 Kafka 集群的监控，请在升级到 AMQ Streams 1.6 **前** 执行以下步骤。

1. 使用新的 **AMQ Streams 1.6**安装工件，将 **strimzi-pod-monitor.yaml** 文件应用到您的 AMQ Streams 1.5 集群：

```
oc apply -f examples/metrics/prometheus-install/strimzi-pod-monitor.yaml
```

2. 从 AMQ Streams 1.5 集群中删除现有 **ServiceMonitor 资源**。
3. 删除名为 **additional-scrape-configs** 的 **Secret**。
4. 从 **AMQ Streams 1.6**安装工件中提供的 **prometheus -additional.yaml 文件中创建一个名为 additional- scrape-configs** 的新 **Secret**。
5. 检查 Prometheus 用户界面的 Prometheus 目标是否已启动并再次运行。
6. 从升级 Cluster Operator 开始，继续升级到 AMQ Streams 1.6。

完成 AMQ Streams 1.6 的升级后，您可以加载 AMQ Streams 1.6 的 Grafana 仪表板示例。

请参阅 向 Kafka 引入指标。

2.7. 通用监听程序配置

本发行版本中引入了 A **GenericKafkaListener** 模式。

模式用于在 Kafka **资源** 中配置 Kafka 侦听程序，并替换 **KafkaListeners** 模式，该模式已弃用。

使用 **GenericKafkaListener** 模式时，您可以根据需要配置多个监听器，只要它们的名称和端口是唯一的。**监听器** 配置被定义为数组，但已弃用的格式也受到支持。

请参阅 **GenericKafkaListener** 模式参考

将监听程序更新到新配置
KafkaListeners 模式将子属性用于 **纯文本、tls 和外部** 监听程序，以及每个节点的固定端口。升级 Kafka 后，您可以将使用 **KafkaListeners** 模式配置的监听程序转换为 **GenericKafkaListener** 模式的格式。

例如，如果您当前在 **Kafka** 配置中使用以下配置：

旧监听程序配置

```
listeners:
  plain:
    # ...
  tls:
    # ...
  external:
    type: loadbalancer
    # ...
```

使用以下方法将监听程序转换为新格式：

新的监听程序配置

```
listeners:
```

```
#...
- name: plain
  port: 9092
  type: internal
  tls: false (1)
- name: tls
  port: 9093
  type: internal
  tls: true
- name: external
  port: 9094
  type: EXTERNAL-LISTENER-TYPE (2)
  tls: true
```

1. 现在，所有监听器都需要 TLS 属性。
2. 选项： **入口、负载均衡器、 节点端口、路由。**

确保使用显示 **的确切** 名称和端口号。

对于任何其他 **配置** 或 **覆盖** 使用旧格式的属性，您需要将它们更新为新格式。

对监听程序 **配置** 进行了更改：

- **覆盖** 与 **configuration** 部分合并
- **dnsAnnotations** 已重命名为 **注解**
- **preferredAddressType** 被重命名为 **preferredNodePortAddressType**
- **地址** 已被重命名为替代名称
- **LoadBalancerSourceRanges 和 external TrafficPolicy 移至现在已弃用 的模板中的监听程序配置**

现在，所有监听器都支持配置公告的主机名和端口。

例如，这个配置：

旧的其他监听程序配置

```
listeners:
  external:
    type: loadbalancer
    authentication:
      type: tls
```

```
    overrides:
      bootstrap:
        dnsAnnotations:
          #...
```

更改：

新的附加监听程序配置

```
listeners:
    #...
  - name: external
    port: 9094
    type:loadbalancer
    tls: true
    authentication:
      type: tls
    configuration:
      bootstrap:
        annotations:
          #...
```

重要

新监听器配置中显示的名称和端口号 必须 用于向后兼容。使用任何其他值将导致对 Kafka 侦听器和 Kubernetes 服务进行重命名。

2.8. MIRRORMAKER 2.0 主题重命名更新

MirrorMaker 2.0 架构通过自动重命名远程主题来代表源集群来支持双向复制。原始集群的名称前面是主题名称的前面。

现在，您可以通过在源连接器配置中添加 IdentityReplicationPolicy 来覆盖自动重命名。应用此配置后，主题会保留其原始名称。

```
apiVersion: kafka.strimzi.io/v1alpha1
```

```
kind: KafkaMirrorMaker2
metadata:
  name: my-mirror-maker2
spec:
  #...
  mirrors:
  - sourceCluster: "my-cluster-source"
    targetCluster: "my-cluster-target"
    sourceConnector:
      config:
        replication.factor: 1
        offset-syncs.topic.replication.factor: 1
        sync.topic.acls.enabled: "false"
        replication.policy.separator: ""
        replication.policy.class: "io.strimzi.kafka.connect.mirror.IdentityReplicationPolicy" (1)
  #...
```

1

添加可覆盖远程主题自动重命名的策略。该主题不会用源集群的名称来附加名称，而是保留其原始名称。

例如，覆盖在您要备份或将数据迁移到另一个集群的 *主动/被动* 集群配置中非常有用。在这两种情况下，您可能不希望自动重命名远程主题。

请参阅 Kafka MirrorMaker 2.0 配置

2.9. 对 HOSTALIASES的支持

现在，在自定义 Kubernetes 模板和 pod 部署时，可以配置 hostAliases。

hostAliases 配置示例

```
apiVersion: kafka.strimzi.io/v1beta1
kind: KafkaConnect
#...
spec:
  # ...
  template:
    pod:
      hostAliases:
      - ip: "192.168.1.86"
        hostnames:
```

```
      - "my-host-1"
      - "my-host-2"
    #...
```

如果指定了主机和 IP 列表，它们将注入到 pod 的 /etc/hosts 文件中。当用户请求集群外的连接时，这对 Kafka Connect 或 MirrorMaker 特别有用。

请参阅 PodTemplate 模式参考

2.10. 协调的资源指标

新的 Operator 指标提供有关指定资源状态的信息，即是否成功协调。

协调的资源指标定义

```
strimzi_resource_state{kind="Kafka", name="my-cluster", resource-namespace="my-kafka-namespace"}
```

2.11. KAFKAUSER的 SECRET 元数据

现在，您可以使用由 User Operator 创建的 Secret 的模板属性。使用 KafkaUserTemplate，您可以使用标签和注解来配置元数据，以定义如何为 KafkaUser 资源生成 Secret。

显示 KafkaUserTemplate的示例

```
apiVersion: kafka.strimzi.io/v1beta1
kind: KafkaUser
metadata:
  name: my-user
  labels:
    strimzi.io/cluster: my-cluster
spec:
  authentication:
```

```
  type: tls
template:
  secret:
    metadata:
      labels:
        label1: value1
      annotations:
        anno1: value1
# ...
```

请参阅 KafkaUserTemplate 模式参考

2.12. 容器镜像中的其他工具

在 AMQ Streams 容器镜像中添加了以下工具：

- jstack
- jcmd
- jmap
- netstat (net-tools)
- lsof

2.13. 删除 KAFKA EXPORTER 服务

Kafka Exporter 服务已从 AMQ Streams 中删除。这个服务已不再需要，因为 Prometheus 现在通过 PodMonitor 声明直接从 Kafka Exporter pod 中提取 Kafka Exporter 指标。

请参阅 向 Kafka 引入指标。

2.14. 在 KAFKA 管理工具中弃用 ZOOKEEPER 选项

以下 Kafka 管理工具中已弃用 --zookeeper 选项：

- bin/kafka-configs.sh
- bin/kafka-leader-election.sh
- bin/kafka-topics.sh

在使用这些工具时，现在应使用 --bootstrap-server 选项指定要连接的 Kafka 代理。例如：

```
kubectl exec BROKER-POD -c kafka -it -- \
  /bin/kafka-topics.sh --bootstrap-server localhost:9092 --list
```

虽然 --zookeeper 选项仍然可以正常工作，但将在以后的 Kafka 发行版本中的所有管理工具中删除。这是 Apache Kafka 项目中持续工作的一部分，以消除 Kafka 对 ZooKeeper 的依赖。

第 3 章 技术预览

重要

技术预览功能不被红帽产品服务级别协议(SLA)支持，且可能无法完成。因此，红帽不推荐在生产环境中实施任何技术预览功能。此技术预览功能为您提供对即将推出的产品创新的早期访问，允许您在开发过程中测试并提供反馈。如需有关支持范围的更多信息，请参阅 技术预览功能支持范围。

3.1. 使用 CRUISE CONTROL 进行集群重新平衡

在本版本中，True Control 仍处于技术预览阶段，但有一些新的 增强。

现在，您可以将 Cruise Control 部署到 AMQ Streams 集群，并使用 *优化目标*（CPU、磁盘和网络负载的预定义限制）来重新平衡 Kafka 集群。在均衡 Kafka 集群中，工作负载更加均匀地分布在代理 pod 中。

作为 Kafka 资源的一部分，Tithise Control 被配置和部署。Cruise Control 的 YAML 配置文件示例在 example /cruise-control/ 中提供。

部署 Cruise Control 后，您可以使用 KafkaRebalance 自定义资源：

- **从多个优化目标生成优化效果**
- **基于优化建议重新平衡 Kafka 集群**

目前不支持其他 Cruise 控制功能，包括异常检测、通知、写入目标以及更改主题复制因素。

请参阅 Cruise Control for cluster 重新平衡。

3.1.1. 技术预览的改进

在使用 Cruise Control 进行集群重新平衡的初始技术预览中添加了以下增强。

重新平衡性能调优

通过五个新的 *性能调优选项*，您可以控制集群重新平衡的方式，并降低其性能影响。

对于组成集群重新平衡的每个批处理 *分区分配命令*，您可以配置以下内容：

- 每个代理的最大并发分区移动数（默认值为 5）
- 最大并发分区移动数（默认为 2）
- 最大并发领导移动数（默认为 1000）
- 分配给分区分配的带宽（默认值不是限制）
- 副本移动策略（默认为 BaseReplicaMovementStrategy）

在以前的版本中，AMQ Streams 从 Cruise Control 中继承这些选项，因此无法调整其默认值。

您可以为 Cruise Control 服务器和/或单独重新平衡设置性能调优选项。

- 对于 Cruise Control 服务器，在 spec.cruiseControl.config 下设置 Kafka 自定义资源中的选项。
- 对于集群重新平衡，在 KafkaRebalance 自定义资源的 spec 属性中设置选项。

请参阅 重新平衡性能调优概述。

从优化调整中排除主题

现在，您可以在优化建议中排除一个或多个主题。这些主题不包括在计算分区副本和分区领导移动中。

要排除主题，请在 spec.excludedTopics Regex 属性中指定与 KafkaRe balance 自定义资源中主题名称匹配的正则表达式。

查看生成的优化建议时，excludeTopics 属性会显示被排除的主题。

请参阅 重新平衡性能调优概述。

CPU 容量目标支持

现在，通过以下配置支持基于 CPU 容量重新平衡 Kafka 集群：

- CpuCapacityGoal 优化目标
- cpuUtilization 容量限制

CPU 容量目标可防止每个代理的 CPU 使用率超过最大百分比阈值。默认阈值设置为每个代理的 CPU 容量 100%。

要降低百分比阈值，请在 Kafka 自定义资源中配置 cpuUtilization 容量限制。容量限制适用于所有代理。

CPU 容量被预先设置为 Cruise Control 中的一项困难目标。因此，除非您覆盖 Kafka.spec.cruiseControl.config 中的 hard.goals 属性中预先设置的硬链接，否则它会继承自 Cruise Control。

KafkaRebalance 自定义资源中的 CPU 容量目标配置示例

```
apiVersion: kafka.strimzi.io/v1alpha1
kind: KafkaRebalance
metadata:
  name: my-rebalance
  labels:
    strimzi.io/cluster: amq-streams-cluster
spec:
  goals:
```

```
    - CpuCapacityGoal
    - DiskCapacityGoal
  #...
```

CPU 使用率限值配置示例

```
apiVersion: kafka.strimzi.io/v1beta1
kind: Kafka
metadata:
  name: amq-streams-cluster
spec:
  # ...
  cruiseControl:
    # ...
    brokerCapacity:
      cpuUtilization: 85
      disk: 100Gi
    # ...
```

请参阅 优化目标和 容量配置。

第 4 章 已弃用的功能

本发行版本中弃用的功能，以及之前的 AMQ Streams 版本中所支持的功能如下所示。

4.1. KAFKALISTENERS 模式

Kafka. KafkaSpec.KafkaClusterSpec 资源中的 Kafka Listeners 模式已弃用。

如需更多信息，请参阅 Enhancements 部分中的 Generic 侦听器配置。

第 5 章 修复的问题

以下小节列出了 AMQ Streams 1.6.x 中已解决的问题。如果您在 OpenShift Container Platform 3.11 中使用 AMQ Streams 1.6.x，红帽建议您升级到最新的补丁版本。

有关修复的问题的详情：

- Kafka 2.6.3，请参阅 Kafka 2.6.3 发行注记
- Kafka 2.6.2，请参阅 Kafka 2.6.2 发行注记
- Kafka 2.6.1，请参阅 Kafka 2.6.1 发行注记
- kafka 2.6.0，请参阅 Kafka 2.6.0 发行注记

5.1. 修复了 AMQ STREAMS 1.6.7 的问题

AMQ Streams 1.6.7 补丁发行版本(Long Term Support)现已正式发布。

AMQ Streams 1.6.7 是唯一与 OpenShift Container Platform 3.11 一起使用的最新 Long Term Support 版本，且仅在 OpenShift Container Platform 3.11 支持时才受支持。

请注意，仅 OCP 3.11 支持 AMQ Streams 1.6.7。

AMQ Streams 产品镜像已升级到 1.6.7 版本。

有关 AMQ Streams 1.6.7 中问题的详情，请参阅 AMQ Streams 1.6.x 解决问题。

Log4j 漏洞

AMQ Streams 包括 log4j 1.2.17。发行版本修复了多个 log4j 漏洞。

有关本版本中解决漏洞的更多信息，请参阅以下 CVE 文章：

- **CVE-2022-23307**
- **CVE-2022-23305**
- **CVE-2022-23302**
- **CVE-2021-4104**
- **CVE-2020-9488**
- **CVE-2019-17571**
- **CVE-2017-5645**

5.2. 修复了 AMQ STREAMS 1.6.6 的问题

有关 AMQ Streams 1.6.6 中解决的问题的详情，请参阅 AMQ Streams 1.6.x 解决问题。

Log4j2 漏洞

AMQ Streams 包括 log4j2 2.17.1。此发行版本修复了多个 log4j2 漏洞。

有关本版本中解决漏洞的更多信息，请参阅以下 CVE 文章：

- CVE-2021-45046
- CVE-2021-45105
- CVE-2021-44832
- CVE-2021-44228

5.3. 修复了 AMQ STREAMS 1.6.5 的问题

有关 AMQ Streams 1.6.5 中解决问题的更多详情，请参阅 AMQ Streams 1.6.x 解析的问题。

Log4j2 漏洞

1.6.5 发行版本修复了使用 log4j2 的 AMQ Streams 组件的远程代码执行漏洞。如果系统从未授权来源记录字符串值，则该漏洞允许在服务器上执行远程代码。这会影响 2.0 和 2.14.1 之间的 log4j 版本。

如需更多信息，请参阅 CVE-2021-44228。

5.4. 修复了 AMQ STREAMS 1.6.4 的问题

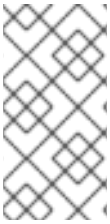
有关 AMQ Streams 1.6.4 中解决问题的更多详情，请参阅 AMQ Streams 1.6.x 解析的问题。

5.5. 修复了 AMQ STREAMS 1.6.2 的问题

AMQ Streams 1.6.2 补丁版本现已正式发布。此发行版本包含与 Kafka Connect 相关的几个修复。

AMQ Streams 产品镜像没有改变，并保留在 1.6 版本中。

有关 AMQ Streams 1.6.2 中解决问题的更多详细信息，请参阅 AMQ Streams 1.6.2 解决的问题。



注意

在 CVE 更新后，由 Operator Lifecycle Manager(OLM)管理的 AMQ Streams 版本改为 1.6.1。为避免混淆，AMQ Streams 1.6 的补丁发行版本被提供了版本号为 1.6.2。

5.6. 修复了 AMQ STREAMS 1.6.0 的问题

问题号	描述
ENTMQST-2049	Kafka 网桥：Kafka consumer 应该使用 group-consumerid 键进行跟踪
ENTMQST-2289	允许带有比降级版本旧的消息版本降级
ENTMQST-2292	在对 Diff PodDisruptionBudgets 进行补丁前，不要在每次协调时重新创建它们
ENTMQST-2146	OCP 上的 MirrorMaker 2 没有使用标头正确镜像消息
ENTMQST-2147	MirrorMaker 2 没有在连接器中正确配置 Jaeger 追踪
ENTMQST-2099	当容限 Kafka 集群设置为空白时会重复滚动更新
ENTMQST-2084	docs 上的 zookeeper 版本与 AMQ Streams 1.5 中的版本不匹配
ENTMQST-2340	使用 KafkaConnect API 时连接 Leak
ENTMQST-2338	修复将点挂载到 Connect 的 Secret 或 ConfigMap
ENTMQST-2294	olm install - yaml 包含 'authentication' 的拼写错误

表 5.1. 修复了常见漏洞和风险(CVE)

问题号	描述
ENTMQST-2332	CVE-2020-13956 httpclient: apache-httpclient: 在请求 URIs [amq-st-1] 中错误地处理错误的授权组件

第 6 章 已知问题

本节列出了 AMQ Streams 的已知问题。

问题号

ENTMQST-2386 - 添加或删除 JBOD 卷无法正常工作

描述和解决方法

AMQ Streams 无法为较新的 Kubernetes 版本添加或删除 JBOD 卷。改进了 StatefulSet Controller 中的验证，这意味着 pod 只能按顺序删除和重新创建（pod-0，然后是 pod-1 等）。目前，AMQ Streams 滚动更新过程不会按顺序触发 pod 删除。

当前的解决方法是按顺序手动删除所有 pod。在 pod 被重新创建后，StatefulSet Controller 不会失败，并如预期运行。

问题

S2I 无法在 SHA 摘要指定的环境中下载源镜像

描述和解决方法

由于 Openshift 镜像流的限制，Kafka Connect S2I 无法在断开连接的集群中构建新的连接器插件。如果通过指定 ImageContentSourcePolicy 更改镜像 registry 的路径，它将被忽略。镜像流将尝试从源存储库下载。

当前的解决方法是手动部署 Kafka Connect。

第 7 章 支持的集成产品

AMQ Streams 1.6 支持与以下红帽产品集成：

- Red Hat Single Sign-On 7.4 及更新的版本 用于 OAuth 2.0 身份验证和 OAuth 2.0 授权
- Red Hat 3scale API Management 2.6 及更新的版本 来保护 Kafka Bridge 并提供额外的 API 管理功能
- Red Hat Debezium 1.0 及之后的版本 用于监控数据库和创建事件流
- Service Registry 2020-Q4 及更新的版本 作为数据流的服务模式的集中存储

有关这些产品可引入到您的 AMQ Streams 部署的功能信息，请参阅 AMQ Streams 1.6 文档。

第 8 章 重要链接

- Red Hat AMQ 7 支持的配置
- Red Hat AMQ 7 组件详情

修订到 2022 年 2 月 13 日 15:27:25 +1000