



## Red Hat AMQ Streams 2.2

# AMQ Streams 2.2 on OpenShift のリリースノート

OpenShift Container Platform 上の本 AMQ Streams リリースにおける新機能と変更  
内容のハイライト



# Red Hat AMQ Streams 2.2 AMQ Streams 2.2 on OpenShift のリリース ノート

---

OpenShift Container Platform 上の本 AMQ Streams リリースにおける新機能と変更内容のハイライト

## 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<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

Node.js<sup>®</sup> 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<sup>®</sup> 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

本リリースノートでは、AMQ Streams 2.2 リリースで導入された新機能、改良された機能、および修正についてまとめています。

## 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 機能 .....                                   | 4  |
| 1.1. AMQ STREAMS 2.2.X (長期サポート)                | 4  |
| 1.2. OPENSIFT CONTAINER PLATFORM のサポート         | 4  |
| 1.3. KAFKA 3.2.3 のサポート                         | 4  |
| 1.4. VIBETA2 API バージョンのサポート                    | 4  |
| 1.5. IBM Z および LINUXONE アーキテクチャーのサポート          | 5  |
| 1.6. IBM POWER アーキテクチャーのサポート                   | 6  |
| 1.7. USESTRIMZIPODSETS フィーチャーゲート (テクノロジープレビュー) | 6  |
| 1.8. USEKRAFT フィーチャーゲート (開発プレビュー)              | 6  |
| 1.9. CRUISE CONTROL の一般提供                      | 7  |
| 1.10. CRUISE CONTROL のスケーリングとリバランスモード          | 8  |
| 1.11. DEBEZIUM FOR CHANGE DATA CAPTURE の統合     | 8  |
| 1.12. SERVICE REGISTRY                         | 9  |
| 第2章 機能拡張 .....                                 | 10 |
| 2.1. KAFKA 3.2.3 で改良された機能                      | 10 |
| 2.2. コマンドラインでの CRUISE CONTROL ステータスの追跡         | 10 |
| 2.3. メンテナンス期間中のユーザー証明書の更新                      | 10 |
| 2.4. CRUISE CONTROL トピックの名前変更                  | 10 |
| 2.5. ZOOKEEPER なしでの CRUISE CONTROL の使用         | 11 |
| 2.6. MIRRORMAKER 2.0 のラックアウェアネスの設定             | 11 |
| 2.7. 使用可能なメモリーの割合に基づいたヒープサイズの設定                | 12 |
| 第3章 テクノロジープレビュー .....                          | 13 |
| 3.1. 新しいフィーチャーゲート                              | 13 |
| 3.2. KAFKA STATIC QUOTA プラグインの設定               | 13 |
| 第4章 KAFKA の重大な変更 .....                         | 14 |
| 4.1. KAFKA のサンプルファイルコネクタの使用                    | 14 |
| 第5章 非推奨の機能 .....                               | 15 |
| 5.1. OPENTRACING                               | 15 |
| 5.2. JAVA 8                                    | 15 |
| 5.3. KAFKA MIRRORMAKER 1                       | 15 |
| 5.4. CRUISE CONTROL TLS サイドカーのプロパティ            | 15 |
| 5.5. ID レプリケーションポリシー                           | 15 |
| 5.6. LISTENERSTATUS タイプのプロパティ                  | 16 |
| 5.7. CRUISE CONTROL の容量設定                      | 16 |
| 第6章 修正された問題 .....                              | 17 |
| 6.1. AMQ STREAMS 2.2.1 で修正された問題                | 17 |
| 6.2. AMQ STREAMS 2.2.0 で修正された問題                | 17 |
| 第7章 既知の問題 .....                                | 19 |
| 7.1. IPV6 クラスターの AMQ STREAMS CLUSTER OPERATOR  | 19 |
| 7.2. CRUISE CONTROL の CPU 使用率予測                | 20 |
| 7.3. ユーザー OPERATOR のスケーラビリティ                   | 22 |
| 第8章 RED HAT 製品へのサポートされる統合 .....                | 23 |
| 第9章 重要なリンク .....                               | 24 |



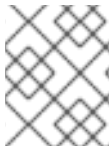
## 多様性を受け入れるオープンソースの強化

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 機能

AMQ Streams 2.2 以降のパッチリリースでは、このセクションで説明する機能が導入されています。

OpenShift 上の AMQ Streams 2.2 は、Kafka 3.2.3 および Strimzi 0.29.x に基づいています。



### 注記

本リリースで解決された機能拡張とバグをすべて確認するには、[AMQ Streams の Jira プロジェクト](#) を参照してください。

### 1.1. AMQ STREAMS 2.2.X (長期サポート)

AMQ Streams 2.2.x は、AMQ Streams の長期サポート (LTS) オファリングです。

最新のパッチリリースは AMQ Streams 2.2.1 です。AMQ Streams の製品イメージがバージョン 2.2.1 に変更されました。サポートされる Kafka のバージョンは 3.2.3 のままです。

LTS の期間および日付については、[AMQ Streams LTS サポートポリシー](#) を参照してください。

### 1.2. OPENSIFT CONTAINER PLATFORM のサポート

AMQ Streams 2.2 は、OpenShift Container Platform 4.8 ~ 4.11 でサポートされています。

サポートされているプラットフォームバージョンの詳細については、[AMQ Streams Supported Configurations](#) を参照してください。

### 1.3. KAFKA 3.2.3 のサポート

AMQ Streams は Apache Kafka バージョン 3.2.3 に対応するようになりました。

AMQ Streams は Kafka 3.2.3 を使用します。サポート対象は、Red Hat によってビルドされた Kafka ディストリビューションのみです。

ブローカーおよびクライアントアプリケーションを Kafka 3.2.3 にアップグレードする前に、Cluster Operator を AMQ Streams バージョン 2.2 にアップグレードする必要があります。アップグレードの手順は、[AMQ Streams のアップグレード](#) を参照してください。

詳細は、[Kafka 3.1.0](#)、[Kafka 3.2.0](#)、[Kafka 3.2.1](#)、および [Kafka 3.2.3](#) のリリースノートを参照してください。



### 注記

Kafka 3.1.x は、AMQ Streams 2.2 にアップグレードする目的でのみサポートされます。

サポート対象バージョンの詳細については、[AMQ Streams Component Details](#) を参照してください。

Kafka 3.2.3 は、Kafka 3.1.0 と同じバージョンの ZooKeeper バージョン 3.6.3 を使用します。

### 1.4. V1BETA2 API バージョンのサポート



すべてのカスタムリソースの **v1beta2** API バージョンが AMQ Streams 1.7 で導入されました。AMQ Streams 1.8 では、**KafkaTopic** および **KafkaUser** を除くすべての AMQ Streams カスタムリソースから **v1alpha1** および **v1beta1** API バージョンが削除されました。

カスタムリソースを **v1beta2** にアップグレードすると、Kubernetes v1.22 に必要な Kubernetes CRD **v1** へ移行する準備ができます。

バージョン 1.7 より前の AMQ Streams バージョンからアップグレードする場合は、以下を行います。

1. AMQ Streams 1.7 へのアップグレード
2. カスタムリソースの **v1beta2** への変換
3. AMQ Streams 1.8 へのアップグレード



### 重要

AMQ Streams バージョン 2.2 にアップグレードする前に、API バージョン **v1beta2** を使用するようにカスタムリソースをアップグレードする必要があります。

[Deploying and upgrading AMQ Streams](#) を参照してください。

#### 1.4.1. カスタムリソースの **v1beta2** へのアップグレード

カスタムリソースの **v1beta2** へのアップグレードをサポートするため、AMQ Streams では **API 変換ツール** が提供されます。これは [AMQ Streams ソフトウェアのダウンロードページ](#) からダウンロードできます。

カスタムリソースのアップグレードは、2つのステップで実行します。

##### ステップ 1: カスタムリソースの形式への変換

API 変換ツールを使用して、以下のいずれかの方法でカスタムリソースの形式を **v1beta2** に適用可能な形式に変換できます。

- AMQ Streams カスタムリソースの設定を記述する YAML ファイルの変換
- クラスターでの AMQ Streams カスタムリソースの直接変換

各カスタムリソースを、**v1beta2** に適用可能な形式に手動で変換することもできます。カスタムリソースを手動で変換する手順は、ドキュメントを参照してください。

##### ステップ 2: CRD の **v1beta2** へのアップグレード

次に、**crd-upgrade** コマンドで API 変換ツールを使用して、CRD の **ストレージ API バージョン** として **v1beta2** を設定する必要があります。この手順は手動で行うことはできません。

完全な手順については、[Upgrading AMQ Streams](#) を参照してください。

## 1.5. IBM Z および LINUXONE アーキテクチャーのサポート

AMQ Streams 2.2 は IBM Z および LinuxONE s390x アーキテクチャーで稼働するように有効になっています。

IBM Z と LinuxONE のサポートは、OpenShift Container Platform 4.10 以降で Kafka で実行している AMQ Streams に適用されます。

### 1.5.1. IBM Z および LinuxONE の要件

- OpenShift Container Platform 4.10 以降

### 1.5.2. IBM Z および LinuxONE でのサポート対象外

- 非接続 OpenShift Container Platform 環境の AMQ Streams
- AMQ Streams OPA の統合

## 1.6. IBM POWER アーキテクチャーのサポート

AMQ Streams 2.2 は、BM Power ppc64le アーキテクチャーでの実行が可能になりました。

IBM Power のサポートは、OpenShift Container Platform 4.9 以降において Kafka で実行している AMQ ストリームに適用されます。

### 1.6.1. IBM Power の要件

- OpenShift Container Platform 4.9 以降

### 1.6.2. IBM Power でサポート対象外

- 非接続 OpenShift Container Platform 環境の AMQ Streams

## 1.7. USESTRIMZIPODSETS フィーチャーゲート (テクノロジープレビュー)

**UseStrimziPodSets** フィーチャーゲートは、**StrimziPodSet** と呼ばれる Pod の管理用のリソースを制御します。フィーチャーゲートが有効な場合には、StatefulSets の代わりにこのリソースが使用されます。AMQ Streams は、OpenShift ではなく Pod の作成および管理を処理します。StatefulSets の代わりに StrimziPodSets を使用すると、機能の制御が強化されます。

フィーチャーゲートの成熟度はアルファレベルにあるため、テクノロジープレビューとして扱う必要があります。

プレビューは、**StrimziPodSet** リソースをテストする機会を提供します。この機能は、リリース 2.3 でデフォルトで有効になります。

フィーチャーゲートを有効にするには、Cluster Operator 設定の **STRIMZI\_FEATURE\_GATES** 環境変数として **+UseStrimziPodSets** を指定します。

### UseStrimziPodSets フィーチャーゲートの有効化

```
env:  
  - name: STRIMZI_FEATURE_GATES  
    value: +UseStrimziPodSets
```

[UseStrimziPodSets 機能ゲート](#) および [機能ゲートリリース](#) を参照してください。

## 1.8. USEKRAFT フィーチャーゲート (開発プレビュー)

Kafka クラスター管理者は、Cluster Operator デプロイメント設定でフィーチャーゲートを使用して、機能のサブセットのオンとオフを切り替えることができます。

Apache Kafka は、ZooKeeper の必要性を段階的に取り除く過程にあります。新しい **UseKRaft** 機能ゲートを有効にすると、ZooKeeper なしで KRaft (Kafka Raft メタデータ) モードで Kafka クラスターのデプロイを試すことができます。

このフィーチャーゲートの成熟度はアルファレベルですが、開発プレビューとして扱う必要があります。

## 注意

このフィーチャーゲートは実験的なものであり、開発とテスト **のみ** を目的としており、実稼働環境では有効にしないでください。

**UseKRaft** 機能ゲートを有効にするには、Cluster Operator 設定で **STRIMZI\_FEATURE\_GATES** 環境変数の値として **+UseKRaft** および **+UseStrimziPodSets** を指定します。**UseKRaft** フィーチャーゲートは、**UseStrimziPodSets** フィーチャーゲートに依存します。

## UseKRaft フィーチャーゲートの有効化

```
env:
- name: STRIMZI_FEATURE_GATES
  value: +UseKRaft, +UseStrimziPodSets
```

現在、AMQ Streams の KRaft モードには、次の主要な制限があります。

- ZooKeeper を使用する Kafka クラスターから KRaft クラスターへの移動、またはその逆の移動はサポートされていません。
- Apache Kafka バージョンまたは AMQ Streams Operator のアップグレードとダウングレードはサポートされていません。ユーザーは、クラスターを削除し、Operator をアップグレードして、新しい Kafka クラスターをデプロイする必要がある場合があります。
- Entity Operator (User Operator と Topic Operator を含む) はサポートされていません。**spec.entityOperator** プロパティは、**Kafka** カスタムリソースから **削除する必要があります**。
- **simple** 認証はサポートされていません。
- SCRAM-SHA-512 認証はサポートされていません。
- JBOD ストレージはサポートされていません。**type: jbod** ストレージを使用できますが、JBOD アレイに含めることができるディスクは1つだけです。
- Liveness および Readiness プローブは無効になっています。
- すべての Kafka ノードには、**controller** と **broker** の両方の KRaft ロールがあります。個別の **controller** と **broker** ノードを持つ Kafka クラスターはサポートされていません。

[UseKRaft feature gate](#) および [Feature gate releases](#) を参照してください。

## 1.9. CRUISE CONTROL の一般提供

Cruise Control は、テクノロジープレビューから一般提供 (GA) に移行します。[Cruise Control](#) をデプロイし、これを使用して **最適化ゴール** (CPU、ディスク、ネットワーク負荷などに定義された制約) を使用し、Kafka をリバランスできます。バランス調整された Kafka クラスターでは、ワークロードがブローカー Pod 全体に均等に分散されます。

Cruise Control は **Kafka** リソースの一部として設定され、デプロイされます。デフォルトの最適化ゴールを使用するか、要件に合わせて変更できます。Cruise Control の YAML 設定ファイルのサンプルは、[examples/cruise-control/](#)にあります。

Cruise Control がデプロイされると、**KafkaRebalance** カスタムリソースを作成して以下を行うことができます。

- 複数の最適化のゴールから、最適化プロポーザルを生成します。
- 最適化プロポーザルを基にして Kafka クラスタを再分散します。

異常検出、通知、独自ゴールの作成、トピックレプリケーション係数の変更などの、その他の Cruise Control の機能は現在サポートされていません。

[Cruise Control によるクラスタのリバランス](#) を参照してください。

## 1.10. CRUISE CONTROL のスケーリングとリバランスモード

以下のいずれかのモードで、リバランス操作の最適化プロポーザルを生成できるようになりました。

- **full**
- **add-brokers**
- **remove-brokers**

以前は、プロポーザルは **full** モードで生成されていました。このモードでは、レプリカがクラスタ内のすべてのブローカー間を移動する可能性があります。ここで、**add-brokers** と **remove-brokers** モードを使用して、スケールアップおよびスケールダウン操作を考慮に入れます。

スケールアップした後は、**add-brokers** モードを使用します。新しいブローカーを指定すると、リバランス操作により、レプリカが既存のブローカーから新しく追加されたブローカーに移動されます。これは、クラスタ全体をリバランスするよりも迅速なオプションになります。

スケールダウンする前に、**remove-brokers** モードを使用してください。削除するブローカーを指定します。これは、それらのブローカー上のレプリカがリバランス操作で移動されることを意味します。

[Rebalancing modes](#)、[Generating optimization proposals](#)、および [Approving optimization proposals](#) を参照してください。

## 1.11. DEBEZIUM FOR CHANGE DATA CAPTURE の統合

Red Hat Debezium は分散型の変更データキャプチャプラットフォームです。データベースの行レベルの変更をキャプチャして、変更イベントレコードを作成し、Kafka トピックにレコードをストリーミングします。Debezium は Apache Kafka に構築されます。AMQ Streams で Debezium をデプロイおよび統合できます。AMQ Streams のデプロイ後に、Kafka Connect で Debezium をコネクタ設定としてデプロイします。Debezium は変更イベントレコードを OpenShift 上の AMQ Streams に渡します。アプリケーションは **変更イベントストリーム** を読み取りでき、変更イベントが発生した順にアクセスできます。

Debezium には、以下を含む複数の用途があります。

- データレプリケーション
- キャッシュの更新およびインデックスの検索

- モノリシックアプリケーションの簡素化
- データ統合
- ストリーミングクエリーの有効化

Debezium は、以下の共通データベースのコネクター (Kafka Connect をベースとする) を提供します。

- Db2
- MongoDB
- MySQL
- PostgreSQL
- SQL Server

AMQ Streams での Debezium のデプロイについて、詳しくは [製品ドキュメント](#) を参照してください。

## 1.12. SERVICE REGISTRY

Service Registry は、データストリーミングのサービススキーマの集中型ストアとして使用できます。Kafka では、Service Registry を使用して **Apache Avro** または JSON スキーマを格納できます。

Service Registry は、REST API および Java REST クライアントを提供し、サーバー側のエンドポイントを介してクライアントアプリケーションからスキーマを登録およびクエリーします。

Service Registry を使用すると、クライアントアプリケーションの設定からスキーマ管理のプロセスが分離されます。クライアントコードに URL を指定して、アプリケーションがレジストリーからスキーマを使用できるようにします。

たとえば、メッセージをシリアル化およびデシリアル化するスキーマをレジストリーに保存できます。アプリケーションは保存されたスキーマを参照し、それらを使用して送受信するメッセージとスキーマとの互換性を維持します。

Kafka クライアントアプリケーションは、実行時にスキーマを Service Registry からプッシュまたはプルできます。

AMQ Streams での Service Registry の使用について、詳しくは [Service Registry documentation](#) を参照してください。

## 第2章 機能拡張

AMQ Streams 2.2 では、多くの機能拡張が追加されました。

### 2.1. KAFKA 3.2.3 で改良された機能

Kafka 3.2.0 および 3.2.1 で導入された拡張機能の概要は、[Kafka 3.2.0 リリースノート](#) [Kafka 3.2.1 リリースノート](#)、および [Kafka 3.2.3 リリースノート](#) を参照してください。

### 2.2. コマンドラインでの CRUISE CONTROL ステータスの追跡

最適化プロポーザルのステータスの確認が簡単になりました。リソース設定 YAML を検査する代わりに、コマンドラインでステータスを確認できます。

プロポーザルを実行するときは、次のコマンドを実行して、最適化プロポーザルのステータスが **ProposalReady** になるのを待ちます。

```
oc get kafkarebalance -o wide -w -n <namespace>
```

#### PendingProposal

**PendingProposal** ステータスは、最適化プロポーザルの準備できているかどうかを確認するために、リバランス Operator が Cruise Control API をポーリングしていることを意味します。

#### ProposalReady

**ProposalReady** ステータスは、最適化プロポーザルのレビューおよび承認の準備ができていることを意味します。

ステータスが **ProposalReady** になると、最適化プロポーザルを承認する準備が整います。

[Generating optimization proposals](#) および [Approving optimization proposals](#) を参照してください。

### 2.3. メンテナンス期間中のユーザー証明書の更新

メンテナンス時間枠によって、Kafka および ZooKeeper クラスターの特定のローリング更新が便利な時間に開始されるようにスケジュールできます。User Operator でメンテナンス期間がサポートされるようになりました。スタンドアロン Operator としてではなく、Cluster Operator を使用して User Operator をデプロイした場合、ユーザー証明書の自動更新がスケジュールに含まれます。

スタンドアロンの User Operator をデプロイしている場合は、有効期限が切れるユーザー証明書が更新されるメンテナンス期間を設定できます。期間は、**STRIMZI\_MAINTENANCE\_TIME\_WINDOWS** 環境変数の Cron 式として指定します。

[Deploying the standalone User Operator](#) を参照してください。

### 2.4. CRUISE CONTROL トピックの名前変更

Cruise Control によって自動的に作成されるメトリクスに関連するトピックの名前を変更できるようになりました。次の Cruise Control 設定プロパティを使用して、名前を変更できます。

#### Cruise Control トピックの名前変更の例

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
```



```

metadata:
  name: my-cluster
spec:
  # ...
  cruiseControl:
    # ...
    config: ❶
    # ...
    metric.reporter.topic: cruise-control-metrics-reporter-topic-name
    partition.metric.sample.store.topic: cruise-control-partitions-metrics-name
    broker.metric.sample.store.topic: cruise-control-broker-metrics
    # ...
  # ...

```

既存のデプロイメントでトピックの名前を変更する場合は、古い名前のトピックを手動で削除する必要があります。

[Configuring and deploying Cruise Control with Kafka](#) を参照してください。

## 2.5. ZOOKEEPER なしでの CRUISE CONTROL の使用

Cruise Control は、Kafka API を代わりに使用することで、ZooKeeper なしで実行されるようになりました。ZooKeeper との安全な通信に使用されていた TLS サイドカーは削除されました。

**Kafka** リソースの Cruise Control 用の TLS サイドカー設定は不要になりました。このため、**.spec.cruiseControl.tlsSidecar** および **.spec.cruiseControl.template.tlsSidecar** プロパティは非推奨になりました。

[Configuring and deploying Cruise Control with Kafka](#) を参照してください。

## 2.6. MIRRORMAKER 2.0 のラックアウェアネスの設定

MirrorMaker 2.0 リソース設定で、ラックアウェアネスを有効化できるようになりました。これは、リージョン間ではなく、同じロケーション内でのデプロイメントを目的とした **特殊なオプション** です。このオプションは、コネクタがリーダーレプリカではなく、最も近いレプリカから消費する場合に使用できます。

**rack** 設定の **topologyKey** は、ラック ID を含むノードラベルと一致する必要があります。次の例では、標準の **topology.kubernetes.io/zone** ラベルが指定されています。

### MirrorMaker 2.0 のラック設定

```

apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaMirrorMaker2
metadata:
  name: my-mirror-maker2
spec:
  version: 3.2.3
  # ...
  rack:
    topologyKey: topology.kubernetes.io/zone

```

最も近いレプリカから消費するには、Kafka ブローカー設定で **RackAwareReplicaSelector** も有効にする必要があります。

## レプリカ対応セクターを有効にした rack 設定例

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    rack:
      topologyKey: topology.kubernetes.io/zone
    config:
      # ...
      replica.selector.class: org.apache.kafka.common.replica.RackAwareReplicaSelector
    # ...
```

[Configuring Kafka MirrorMaker 2.0](#) および [Rack schema reference](#) を参照してください。

## 2.7. 使用可能なメモリーの割合に基づいたヒープサイズの設定

設定でヒープサイズを指定しない場合、Cluster Operator はデフォルトのヒープサイズを自動的に指定します。Cluster Operator は、メモリーリソース設定の割合に基づいて、デフォルトの最大および最小ヒープ値を設定します。デフォルトで割り当てられている使用可能なメモリーは、次の表に示すレベルに設定されています。

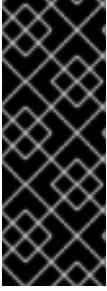
表2.1 コンポーネントのデフォルトのヒープ設定

| コンポーネント         | ヒープに割り当てられた<br>使用可能なメモリーの割合 | 上限    |
|-----------------|-----------------------------|-------|
| Kafka           | 50%                         | 5 GB  |
| ZooKeeper       | 75%                         | 2 GB  |
| Kafka Connect   | 75%                         | None  |
| MirrorMaker 2.0 | 75%                         | None  |
| MirrorMaker     | 75%                         | None  |
| Cruise Control  | 75%                         | None  |
| Kafka Bridge    | 50%                         | 31 Gi |

[jvmOptions](#) を参照してください。



## 第3章 テクノロジープレビュー



### 重要

テクノロジープレビュー機能は、Red Hat の実稼働環境のサービスレベルアグリーメント (SLA) ではサポートされません。また、機能的に完全ではない可能性があるため、Red Hat はテクノロジープレビュー機能を実稼働環境に実装することは推奨しません。テクノロジープレビューの機能は、最新の技術をいち早く提供して、開発段階で機能のテストやフィードバックの収集を可能にするために提供されます。サポート範囲の詳細は、[テクノロジープレビュー機能のサポート範囲](#) を参照してください。

### 3.1. 新しいフィーチャーゲート

**UseKRaft** および **UseStrimziPodSets** 機能ゲートのプレビューが利用可能になりました。

[1章 機能](#) を参照してください。

### 3.2. KAFKA STATIC QUOTA プラグインの設定

**Kafka Static Quota** プラグインを使用して、Kafka クラスターのブローカーにスループットおよびストレージの制限を設定します。**Kafka** リソースを設定して、プラグインを有効にし、制限を設定します。バイトレートのしきい値およびストレージクォータを設定して、ブローカーと対話するクライアントに制限を設けることができます。

#### Kafka Static Quota プラグインの設定例

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    config:
      client.quota.callback.class: io.strimzi.kafka.quotas.StaticQuotaCallback
      client.quota.callback.static.produce: 1000000
      client.quota.callback.static.fetch: 1000000
      client.quota.callback.static.storage.soft: 4000000000000
      client.quota.callback.static.storage.hard: 5000000000000
      client.quota.callback.static.storage.check-interval: 5
```

[Kafka Static Quota プラグインを使用したブローカーへの制限の設定](#) を参照してください。

## 第4章 KAFKA の重大な変更

このセクションでは、機能を継続するために AMQ Streams への対応する変更が必要な Kafka への変更について説明します。

### 4.1. KAFKA のサンプルファイルコネクターの使用

Kafka は、デフォルトで、サンプルファイルコネクター **FileStreamSourceConnector** と **FileStreamSinkConnector** を **CLASSPATH** と **plugin.path** に含めなくなりました。AMQ Streams が更新され、これらのサンプルコネクターを引き続き使用できるようになりました。サンプルは、他のコネクターと同様にプラグインパスに追加する必要があります。

コネクター設定ファイルの 2 つのサンプルが提供されています。

- **examples/connect/kafka-connect-build.yaml** は、Kafka Connect **build** 設定を提供します。これをデプロイして、ファイルコネクターを使用して新しい KafkaConnect イメージをビルドすることができます。
- **examples/connect/source-connector.yaml** は、ファイルコネクターを **KafkaConnector** リソースとしてデプロイするために必要な設定を提供します。

[Deploying example KafkaConnector resources](#) および [Extending Kafka Connect with connector plugins](#) を参照してください。

## 第5章 非推奨の機能

以下の機能は、これまでの AMQ Streams リリースではサポート対象でしたが、このリリースで非推奨となりました。

### 5.1. OPENTRACING

OpenTracing のサポートは非推奨となりました。

Jaeger クライアントは廃止され、OpenTracing プロジェクトはアーカイブされました。そのため、将来の Kafka バージョンのサポートを保証することはできません。OpenTelemetry プロジェクトに基づく新しいトレース実装を導入しています。

### 5.2. JAVA 8

Java 8 のサポートは、Kafka 3.0.0 および AMQ Streams 2.0 で非推奨になりました。Java 8 は、将来、クライアントを含むすべての AMQ Streams コンポーネントでサポート対象外となります。

AMQ Streams は Java 11 をサポートします。新しいアプリケーションを開発する場合は、Java 11 を使用してください。また、現在 Java 8 を使用しているアプリケーションの Java 11 への移行も計画してください。

### 5.3. KAFKA MIRRORMAKER 1

Kafka MirrorMaker は、データセンター内またはデータセンター全体の 2 台以上の Kafka クラスター間でデータをレプリケーションします。Kafka MirrorMaker 1 は Kafka 3.0.0 で非推奨となり、Kafka 4.0.0 で削除されます。MirrorMaker 2.0 のみが利用可能なバージョンになります。MirrorMaker 2.0 は Kafka Connect フレームワークをベースとし、コネクタによってクラスター間のデータ転送が管理されます。

そのため、Kafka MirrorMaker 1 のデプロイに使用される AMQ Streams **KafkaMirrorMaker** カスタムリソースが非推奨になりました。Kafka 4.0.0 が導入されると、**KafkaMirrorMaker** リソースは AMQ Streams から削除されます。

MirrorMaker 1 (AMQ Streams ドキュメントで **MirrorMaker** と呼ばれる) を使用している場合は、**IdentityReplicationPolicy** と **KafkaMirrorMaker2** のカスタムリソースを使用します。MirrorMaker 2.0 では、ターゲットクラスターにレプリケートされたトピックの名前が変更されます。**IdentityReplicationPolicy** 設定は、名前の自動変更を上書きします。これを使用して、MirrorMaker 1 と同じアクティブ/パッシブの一方方向レプリケーションを作成します。

[Kafka MirrorMaker 2.0 クラスターの設定](#) を参照してください。

### 5.4. CRUISE CONTROL TLS サイドカーのプロパティ

このリリースでは、[Cruise Control TLS サイドカーが削除されました](#)。その結果、**.spec.cruiseControl.tlsSidecar** および **.spec.cruiseControl.template.tlsSidecar** プロパティは非推奨になりました。プロパティは無視され、将来削除されます。

### 5.5. ID レプリケーションポリシー

ID レプリケーションポリシーは MirrorMaker 2.0 で使用され、リモートトピックの自動名前変更をオーバーライドします。その名前の前にソースクラスターの名前を追加する代わりに、トピックが元の名前を保持します。このオプションの設定は、active/passive バックアップおよびデータ移行に役立ちま

す。

現在、AMQ Streams Identity Replication Policy class (**io.strimzi.kafka.connect.mirror.IdentityReplicationPolicy**) は非推奨であり、将来削除される予定です。Kafka 独自の ID レプリケーションポリシー (**class org.apache.kafka.connect.mirror.IdentityReplicationPolicy**) に更新できます。

[Kafka MirrorMaker 2.0 クラスターの設定](#) を参照してください。

## 5.6. LISTENERSTATUS タイプのプロパティ

**ListenerStatus** の **type** プロパティは非推奨になり、将来削除される予定です。**ListenerStatus** は、内部および外部リスナーのアドレスを指定するために使用されます。**type** を使用する代わりに、アドレスが **name** で指定されるようになりました。

[ListenerStatus スキーマ参照](#) を参照してください。

## 5.7. CRUISE CONTROL の容量設定

**disk** および **cpuUtilization** 容量の設定プロパティは非推奨になり、無視され、将来削除される予定です。プロパティは、リソースベースの最適化ゴールの破損を判断するための最適化プロポーザルの容量制限の設定に使用されました。ディスクと CPU の容量制限は、AMQ Streams によって自動的に生成されるようになりました。

[Cruise Control configuration](#) を参照してください。

## 第6章 修正された問題

以下のセクションでは、AMQ Streams 2.2.x で修正された問題をリストします。Red Hat は、最新のパッチリリースにアップグレードすることを推奨します。

Kafka 3.2.0、3.2.1、および 3.2.3 で修正された問題の詳細は、[Kafka 3.2.0 リリースノート](#)、[Kafka 3.2.1 リリースノート](#)、および [Kafka 3.2.3 リリースノート](#) を参照してください。

### 6.1. AMQ STREAMS 2.2.1 で修正された問題

AMQ Streams 2.2.1 パッチリリース (長期サポート) が利用可能になりました。

AMQ Streams 2.2.1 で解決された問題の詳細については、[AMQ Streams 2.2.x で解決された問題](#) を参照してください。

### 6.2. AMQ STREAMS 2.2.0 で修正された問題

表6.1 修正された問題

| 課題番号                         | 説明  |
|------------------------------|---|
| <a href="#">ENTMQST-3757</a> | [KAFKA] MirrorMaker 2.0 negative lag  |
| <a href="#">ENTMQST-3762</a> | Topic Operator の起動中に "VertxException: Thread blocked" エラーが表示されます                          |
| <a href="#">ENTMQST-3775</a> | Bridge では slf4j-api と log4j-api を同時に使用しないでください。   |
| <a href="#">ENTMQST-3862</a> | KafkaRoller でのロギングを改善します  |
| <a href="#">ENTMQST-3867</a> | StrimziPodSet リソースの非カスケード削除を修正します   |
| <a href="#">ENTMQST-3897</a> | KafkaConnector リソースの調整の失敗は、Operator メトリクスにカウントされません                                       |
| <a href="#">ENTMQST-3918</a> | クラスター起動時に、ローリング更新が Pod を強制的にロールします  |
| <a href="#">ENTMQST-3955</a> | ミリバイト単位でのストレージの解析のサポートを追加します  |
| <a href="#">ENTMQST-3956</a> | 無効なストレージユニットが使用されている場合、調整に失敗します   |
| <a href="#">ENTMQST-3958</a> | Cruise Control デプロイメントの不要なローリング更新を回避します   |
| <a href="#">ENTMQST-3972</a> | Pod に ANNO_STRIMZI_IO_CLUSTER_CA_CERT_GENERATION アノテーションがないため、Kafka の調整中に CO ログでエラーが発生します |
| <a href="#">ENTMQST-3997</a> | curl ダウンロードが失敗すると、Kafka Connect ビルドは失敗するはずですが   |
| <a href="#">ENTMQST-4017</a> | KafkaRebalance カスタムリソースのエラーが正しくログに記録されません   |
| <a href="#">ENTMQST-4071</a> | AMQ Streams Drain クリーナーで FIPS モードを処理する  |

| 課題番号                         | 説明   |
|------------------------------|--|
| <a href="#">ENTMQST-4264</a> | [KAFKA] 認証されていないクライアントがブローカーで OutOfMemoryError を引き起こす場合がある |

表6.2 CVE (Common Vulnerabilities and Exposures) の修正

| 課題番号                         | 説明  |
|------------------------------|---|
| <a href="#">ENTMQST-3917</a> | CVE-2020-36518 jackson-databind: ネストされたオブジェクトの深さが大きいサービス拒否                                    |
| <a href="#">ENTMQST-4049</a> | CVE-2022-24823 netty: 機密データを含む誰でも読み取り可能な一時ファイル  |
| <a href="#">ENTMQST-4050</a> | CVE-2022-25647 com.google.code.gson-gson: com.google.code.gson-gson で Untrusted Data のデシリアライズ |

## 第7章 既知の問題

このセクションでは、AMQ Streams 2.2 on OpenShift の既知の問題について説明します。

### 7.1. IPV6 クラスターの AMQ STREAMS CLUSTER OPERATOR

AMQ Streams Cluster Operator は、IPv6 (Internet Protocol version 6) クラスターでは起動しません。

#### 回避策

この問題を回避する方法は2つあります。

#### 回避方法 1: KUBERNETES\_MASTER 環境変数の設定

1. OpenShift Container Platform クラスターの Kubernetes マスターノードのアドレスを表示します。

```
oc cluster-info
Kubernetes master is running at <master_address>
# ...
```

マスターノードのアドレスをコピーします。

2. すべての Operator サブスクリプションを一覧表示します。

```
oc get subs -n <operator_namespace>
```

3. AMQ Streams の **Subscription** リソースを編集します。

```
oc edit sub amq-streams -n <operator_namespace>
```

4. **spec.config.env** で、**KUBERNETES\_MASTER** 環境変数を追加し、Kubernetes マスターノードのアドレスに設定します。以下に例を示します。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: amq-streams
  namespace: <operator_namespace>
spec:
  channel: amq-streams-1.8.x
  installPlanApproval: Automatic
  name: amq-streams
  source: mirror-amq-streams
  sourceNamespace: openshift-marketplace
  config:
    env:
      - name: KUBERNETES_MASTER
        value: MASTER-ADDRESS
```

5. エディターを保存し、終了します。
6. **Subscription** が更新されていることを確認します。

```
oc get sub amq-streams -n <operator_namespace>
```

- Cluster Operator の **Deployment** が、新しい環境変数を使用するように更新されていることを確認します。

```
oc get deployment <cluster_operator_deployment_name>
```

## 回避方法 2: ホスト名検証の無効化

- すべての Operator サブスクリプションを一覧表示します。

```
oc get subs -n <operator_namespace>
```

- AMQ Streams の **Subscription** リソースを編集します。

```
oc edit sub amq-streams -n <operator_namespace>
```

- spec.config.env** で、**true** に設定された **KUBERNETES\_DISABLE\_HOSTNAME\_VERIFICATION** 環境変数を追加します。以下に例を示します。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: amq-streams
  namespace: <operator_namespace>
spec:
  channel: amq-streams-1.8.x
  installPlanApproval: Automatic
  name: amq-streams
  source: mirror-amq-streams
  sourceNamespace: openshift-marketplace
  config:
    env:
      - name: KUBERNETES_DISABLE_HOSTNAME_VERIFICATION
        value: "true"
```

- エディターを保存し、終了します。
- Subscription** が更新されていることを確認します。

```
oc get sub amq-streams -n <operator_namespace>
```

- Cluster Operator の **Deployment** が、新しい環境変数を使用するように更新されていることを確認します。

```
oc get deployment <cluster_operator_deployment_name>
```

## 7.2. CRUISE CONTROL の CPU 使用率予測

AMQ Streams の Cruise Control には、CPU 使用率予測の計算に関連する既知の問題があります。CPU 使用率は、ブローカー Pod の定義容量の割合として計算されます。この問題は、CPU コアが異なる



ノードで Kafka ブローカーを実行する場合に発生します。たとえば、node1 には 2 つの CPU コアがあり、node2 には 4 つの CPU コアが含まれるとします。この場合、Cruise Control はブローカーの CPU 負荷を過大または過少に予測する可能性があります。この問題が原因で、Pod の負荷が大きいときにクラスタのリバランスができない場合があります。

## 回避策

この問題を回避する方法は 2 つあります。

### 回避策 1: CPU 要求と制限を同等レベルにする

**Kafka.spec.kafka.resources** の CPU 制限と同等の CPU 要求を設定できます。これにより、すべての CPU リソースが事前に予約され、常に利用できます。この設定を使用すると、CPU ゴールに基づきリバランスプロポーザルを準備する際に、Cruise Control は CPU 使用率を適切に評価できます。

### 回避策 2: CPU の目標を除外する

Cruise Control 設定に指定されたハードおよびデフォルトのゴールから CPU ゴールを除外できます。

### CPU ゴールのない Cruise Control の設定例

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
  zookeeper:
    # ...
  entityOperator:
    topicOperator: {}
    userOperator: {}
  cruiseControl:
    brokerCapacity:
      inboundNetwork: 10000KB/s
      outboundNetwork: 10000KB/s
    config:
      hard.goals: >
        com.linkedin.kafka.cruisecontrol.analyzer.goals.RackAwareGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.MinTopicLeadersPerBrokerGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaCapacityGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.DiskCapacityGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkInboundCapacityGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkOutboundCapacityGoal
      default.goals: >
        com.linkedin.kafka.cruisecontrol.analyzer.goals.RackAwareGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.MinTopicLeadersPerBrokerGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaCapacityGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.DiskCapacityGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkInboundCapacityGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkOutboundCapacityGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaDistributionGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.PotentialNwOutGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.DiskUsageDistributionGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkInboundUsageDistributionGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkOutboundUsageDistributionGoal,
```

```
com.linkedin.kafka.cruisecontrol.analyzer.goals.TopicReplicaDistributionGoal,  
com.linkedin.kafka.cruisecontrol.analyzer.goals.LeaderReplicaDistributionGoal,  
com.linkedin.kafka.cruisecontrol.analyzer.goals.LeaderBytesInDistributionGoal
```

詳細は、[Insufficient CPU capacity](#) を参照してください。

## 7.3. ユーザー OPERATOR のスケーラビリティ

複数のユーザーを同時に作成すると、User Operator がタイムアウトになることがあります。調整には時間がかかりすぎる場合があります。

### 回避策

この問題が発生した場合は、同時に作成するユーザーの数を減らしてください。さらにユーザーを作成する前に、準備が整うまで待ちます。

## 第8章 RED HAT 製品へのサポートされる統合

AMQ Streams 2.2 は、以下の Red Hat 製品との統合をサポートします。

### Red Hat Single Sign-On

OAuth 2.0 認証と OAuth 2.0 認証を提供します。

### Red Hat 3scale API Management

Kafka Bridge のセキュリティーを保護し、追加の API 管理機能を提供します。

### Red Hat Debezium

データベースを監視し、イベントストリームを作成します。

### Red Hat Service Registry

データストリーミングのサービススキーマの集中型ストアを提供します。

これらの製品を使用することで AMQ Streams デプロイメントに導入できる機能の詳細は、製品ドキュメントを参照してください。

### 関連情報

- [Red Hat Single Sign-On Supported Configurations](#)
- [Red Hat 3scale API Management Supported Configurations](#)
- [Red Hat Debezium Supported Configurations](#)
- [Red Hat Service Registry Supported Configurations](#)

## 第9章 重要なリンク

- [AMQ Streams Supported Configurations](#)
- [AMQ Streams Component Details](#)

改訂日時: 2023-04-06