Red Hat AMQ 7.7

AMQ Streams 1.5 on OpenShift リリースノート

AMQ Streams on OpenShift Container Platform の使用

Last Updated: 2023-02-03

Red Hat AMQ 7.7 AMQ Streams 1.5 on OpenShift リリースノート

AMQ Streams on OpenShift Container Platform の使用

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.

法律上の通知

概要

本リリースノートには、AMQ Streams 1.5 リリースに含まれる新機能、改良された機能、修正、および問題に関する最新情報が含まれています。

目次

第1章 機能

AMQ Streams バージョン 1.5 は Strimzi 0.18.x をベースにしています。

このリリースで追加され、これまでの AMQ Streams リリースにはなかった機能は次のとおりです。

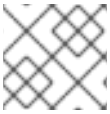
1.1. KAFKA 2.5.0 のサポート

AMQ Streams は Apache Kafka バージョン 2.5.0 に対応するようになりました。

AMQ Streams は Kafka 2.5.0 を使用します。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がサポートされます。

ブローカーおよびクライアントアプリケーションを Kafka 2.5.0 にアップグレードする前に、Cluster Operator を AMQ Streams バージョン 1.5 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 および Kafka のアップグレード を参照してください。

詳細は、Kafka 2.4.0 および Kafka 2.5.0 のリリースノートを参照してください。



注記

Kafka 2.4.x は、アップグレードの目的のみで AMQ Streams 1.5 でサポートされます。

サポートされるバージョンの詳細は、カスタマーポータルの Red Hat AMQ 7 Component Details Page を参照してください。

1.2. ZOOKEEPER 3.5.8 のサポート

Kafka バージョン 2.5.0 では、ZooKeeper バージョン 3.5.8 が必要です。

手作業で ZooKeeper 3.5.8 にアップグレードする必要は**ありません**。Kafka ブローカーがアップグレードされる ときに、Cluster Operator によって ZooKeeper のアップグレードが実行されます。ただし、この手順の実行中に追加のローリング更新が実行されることがあります。

1.3. OPENSHIFT 4.X の非接続インストール

OpenShift 4.x での **非接続インストール** のサポートは、テクノロジープレビューから AMQ Streams の一般提供コンポーネントに移行されます。

OpenShift クラスターが制限されたネットワークで**非接続クラスター** として使用されている場合、AMQ Streams の非接続インストールを実行できます。

非接続インストールでは、必要なイメージを取得し、それらをローカルでコンテナーレジストリーにプッシュします。Operator Lifecycle Manager (OLM) を使用している場合、OperatorHub によって使用されるデフォルトのソースを無効にし、ローカルミラーを作成してローカルソースから AMQ Streams をインストールすることになります。

ネットワークが制限された環境での Operator Lifecycle Manager の使用 を参照してください。

1.4. MIRRORMAKER 2.0

MirrorMaker 2.0 のサポートは、テクノロジープレビューから AMQ Streams の一般提供コンポーネントに移行されます。

MirrorMaker 2.0 は Kafka Connect フレームワークをベースとし、**コネクター**によってクラスター間のデータ転送が管理されます。

MirrorMaker 2.0 は以下を使用します。

- ソースクラスターからデータを消費するソースクラスターの設定
- データをターゲットクラスターに出力するターゲットクラスターの設定。

MirrorMaker 2.0 では、クラスターでデータをレプリケートする全く新しい方法が導入されました。MirrorMaker 2.0 の使用を選択した場合、現在、レガシーサポートがないため、リソースを手作業で新しい形式に変換する必要があります。

AMQ Streams の MirrorMaker 2.0 との使用 を参照してください。

1.5. 変更データキャプチャー統合の DEBEZIUM

Red Hat Debezium は分散型の変更データキャプチャー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ーして、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 をコネクター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 AMQ Streams on OpenShift 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。

- MySQL
- PostgreSQL
- SQL Server
- MongoDB

AMQ Streams で Debezium をデプロイするための詳細は、製品ドキュメント を参照してください。

1.6. 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ライズおよびデシリアライズ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。

Service Registry を使用したスキーマの管理 を参照してください。

第2章 改良された機能

このリリースで改良された機能は次のとおりです。

2.1. KAFKA 2.5.0 で改良された機能

Kafka 2.5.0 に導入された改良機能の概要は Kafka 2.5.0 Release Notes を参照してください。

2.2. ZOOKEEPER の改良

ZooKeeper 3.5.* バージョンへのサポートの移行により、ZooKeeper Pod で TLS サイドカーを使う必要がなくなりました。サイドカーは削除され、TLS のネイティブ ZooKeeper サポートが使用されるようになりました。

2.3. OAUTH 2.0 認証設定オプションの拡張

新しい設定オプションによって、より多くの承認サーバーとの統合が可能になりました。

OAuth 2.0 認証の適用方法や、承認サーバーのタイプによっては、追加 (任意) の設定を使用できます。

Kafka ブローカーの追加設定オプション

```
# ...
authentication:
  type: oauth
  # ...
  checkIssuer: false (1)
  fallbackUserNameClaim: client_id (2)
  fallbackUserNamePrefix: client-account- (3)
  validTokenType: bearer (4)
  userInfoEndpointUri: https://AUTH-SERVER-ADDRESS/auth/realms/external/protocol/openid-connect/userinfo (5)
```

(1) 承認サーバーが **iss** クレームを提供しない場合は、発行者チェックを行うことができません。このような場合、**checkIssuer** を **false** に設定し、**validIssuerUri** を指定しないようにします。デフォルトは **true** です。

(2) 承認サーバーは、通常ユーザーとクライアントの両方を識別する単一の属性を提供しない場合があります。独自の名前を認証するクライアントによって **クライアント ID** が提供される場合があります。しかし、更新トークンまたはアクセストークンの取得に、ユーザー名とパスワードを使用して認証されるユーザーは、クライアント ID の他に **ユーザー名** を提供する場合があります。プライマリーユーザー ID 属性が使用できない場合は、このフォールバックオプションで、使用するユーザー名クレーム (属性) を指定します。

(3) **fallbackUserNameClaim** が適用される場合、ユーザー名クレームの値とフォールバックユーザー名クレームの値が競合しないようにする必要もあることがあります。**producer** というクライアントが存在し、**producer** という通常ユーザーも存在する場合について考えてみましょう。この 2 つを区別するには、このプロパティーを使用してクライアントのユーザー ID に接頭辞を追加します。

(4) (イントロスペクションエンドポイント URI を使用する場合のみ該当): 使用している認証サーバーによっては、イントロスペクションエンドポイントによって**トークンタイプ**属性が返されるかどうかは分からず、異なる値が含まれることがあります。イントロスペクションエンドポイントからの

応答に含まれなければならない有効なトークンタイプ値を指定できます。

5 (イントロスペクションエンドポイント URI を使用する場合のみ該当): イントロスペクションエンドポイントの応答に識別可能な情報が含まれないように、承認サーバーが設定または実装されることがあります。ユーザー ID を取得するには、**userinfo** エンドポイントの URI をフォールバックとして設定します。**userNameClaim**、**fallbackUserNameClaim**、および **fallbackUserNamePrefix** の設定が **userinfo** エンドポイントの応答に適用されます。

Kafka コンポーネントの追加設定オプション

```
# ...
spec:
  # ...
  authentication:
    # ...
    disableTlsHostnameVerification: true
    checkAccessTokenType: false
    accessTokenIsJwt: false
    scope: any 1
```

1 トークンエンドポイントからトークンを要求するための **scope**。認証サーバーでは、クライアントによるスコープの指定が必要になることがあります。

Kafka ブローカーの OAuth 2.0 サポートの設定 および Kafka コンポーネントの OAuth 2.0 の設定 を参照してください。

2.4. CLUSTER OPERATOR、TOPIC OPERATOR、および USER OPERATOR のメトリクス

Prometheus サーバーは、AMQ Streams ディストリビューションの一部としてサポートされません。しかし、メトリクスを公開するために使用される Prometheus エンドポイントと JMX エクスポーターはサポートされます。

Kafka コンポーネントを監視する他に、メトリクス設定を追加して Cluster Operator、Topic Operator、および User Operator を監視できます。

Grafana ダッシュボードのサンプルが提供され、Prometheus によって公開される以下の operator メトリクスを表示できます。

- 処理されたカスタムリソースの数
- 処理された調整の数
- JVM メトリクス

Kafka へのメトリクスの導入 を参照してください。

2.5. SSL 設定オプション

TLS バージョンの特定の暗号に外部リスナーを実行することができるようになりました。

AMQ Streams で稼働している Kafka コンポーネントでは、許可される 3 つの **ssl** 設定オプションを使用して、TLS バージョンの固有の暗号スイートで外部リスナーを実行できます。暗号スイートでは、セキュアな接続とデータ転送のためのアルゴリズムが組み合わされます。

以下の例は、Kafka リソースの例を示しています。

```
apiVersion: kafka.strimzi.io/v1beta1
kind: Kafka
metadata:
  name: my-cluster
spec:
  # ...
  config: (1)
    # ...
    ssl.cipher.suites: "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384" (2)
    ssl.enabled.protocols: "TLSv1.2" (3)
    ssl.protocol: "TLSv1.2" (4)
```

2. TLS の暗号スイートは、**ECDHE** 鍵交換メカニズム、**RSA** 認証アルゴリズム、**AES** 一括暗号化アルゴリズム、および **SHA384** MAC アルゴリズムの組み合わせを使用します。
3. SSl プロトコル **TLSv1.2** は有効になります。
4. **TLSv1.2** プロトコルを指定し、SSL コンテキスト生成します。

カスタムリソース API のリファレンス を参照してください。

2.6. KAFKA BRIDGE の CORS (CROSS-ORIGIN RESOURCE SHARING)

CORS (Cross-Origin Resource Sharing) より、Kafka Bridge のアクセス制御を有効化および定義できるようになりました。CORS は、複数のオリジンから指定のリソースにブラウザーでアクセスできるようにする HTTP メカニズムです。CORS を設定するには、許可されるリソースオリジンのリストと、それらにアクセスする HTTP メソッドを定義します。リクエストの追加の HTTP ヘッダーには Kafka クラスターへのアクセスが許可されるオリジンが記述 されています。

Kafka Bridge の HTTP 設定

```
apiVersion: kafka.strimzi.io/v1alpha1
kind: KafkaBridge
metadata:
  name: my-bridge
spec:
  # ...
  http:
    port: 8080
    cors:
      allowedOrigins: "https://strimzi.io" (1)
      allowedMethods: "GET,POST,PUT,DELETE,OPTIONS,PATCH" (2)
  # ...
```

1. 許可される CORS オリジンのコンマ区切りリスト。URL または Java 正規表現を使用できます。
2. CORS で許可される HTTP メソッドのコンマ区切りリスト。

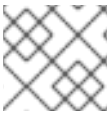
Kafka Bridge HTTP の設定 を参照してください。

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の機能は、Red Hat の実稼働環境のサービスレベルアグリーメント (SLA) ではサポートされず、機能的に完全ではないことがあるため、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. CRUISE CONTROL によるクラスターの再分散



注記

これはテクノロジープレビューの機能です。

Cruise Control を AMQ Streams クラスターにデプロイし、Kafka クラスターの再分散に使用できるようになりました。Cruise Control を使用すると、分散された Kafka クラスターを効率的に実行するための時間および労力を削減できます。

Cruise Control は、**Kafka** リソースの設定の一部としてデプロイされます。Cruise Control の YAML 設定ファイルのサンプルは、**examples/cruise-control/** にあります。

各 Kafka クラスターに Cruise Control のインスタンスをデプロイすると、以下を実行できます。

- 複数の最適化のゴールから、最適化の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 クラスターを再分散します。

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 によるクラスターのリバランス を参照してください。

ユーザー提供の最適化ゴールの設定に関する既知の問題が存在します。6章*既知の問題*を参照してください。

3.2. OAUTH 2.0 承認

注記

これはテクノロジープレビューの機能です。

トークンベースの認証に OAuth 2.0 を使用している場合、OAuth 2.0 承認ルールを使用して、クライアントの Kafka ブローカーへのアクセスを制限できるようになりました。

AMQ Streams は、Red Hat Single Sign-On の 承認サービス による OAuth 2.0 トークンベースの承認をサポートします。これにより、セキュリティーポリシーとパーミッションの一元的な管理が可能になります。

Red Hat Single Sign-On で定義されたセキュリティーポリシーおよびパーミッションは、Kafka ブローカーのリソースへのアクセスを付与するために使用されます。ユーザーとクライアントは、Kafka ブローカーで特定のアクションを実行するためのアクセスを許可するポリシーに対して照合されます。

OAuth 2.0 トークンベース承認の使用 を参照してください。

第4章 非推奨の機能

このリリースで非推奨となり、これまでの AMQ Streams リリースではサポートされていた機能は次のとおりです。

4.1. API バージョン

AMQ Streams 1.2 リリースでは、**v1alpha1** バージョンの以下のリリースは非推奨となり、 **v1beta1** に置き換えられました。

- **Kafka**
- **KafkaConnect**
- **KafkaConnectS2I**
- **KafkaMirrorMaker**
- **KafkaTopic**
- **KafkaUser**

次回のリリースでは、**v1alpha1** バージョンのこれらのリソースは削除される予定です。リソースのアップグレードに関する情報は AMQ Streams リソースのアップグレード を参照してください。

第5章 修正された問題

以下のセクションには、AMQ Streams 1.5.1 および 1.5 で修正された問題が記載されています。

5.1. AMQ STREAMS 1.5.1 で修正された問題

AMQ Streams 1.5.1 パッチリリースを使用できます。このマイクロリリースは、OpenShift OperatorHub ユーザーインターフェイスから AMQ Streams 1.5.0 をインストールするために使用される Operator メタデータを更新します。AMQ Streams の製品イメージは変更されておらず、バージョン 1.5.0 のままになります。

OpenShift Container Platform 4.5 を使用している場合、Red Hat は AMQ Streams 1.5.1 にアップグレードすることを推奨します。

AMQ Streams 1.5.1 で解決された問題の詳細は、AMQ Streams 1.5.x Resolved Issues を参照してください。

5.2. AMQ STREAMS 1.5 で修正された問題

課題番号	説明
ENTMQST-622	Admin Client API を利用するための **SimpleAclOperator** のリファクタリング。
ENTMQST-643	CA 鍵の削除時に新しいクラスターまたはクライアントシークレット/鍵が生成されるように修正。
ENTMQST-1337	StorageDiff によるノードのスケールアップおよびダウンの処理が必要。
ENTMQST-1556	KafkaRoller のエクスポネンシャルバックオフに 20 分以上かかるときがある。
ENTMQST-1557	KafkaRoller クラスのロギングの改善。
ENTMQST-1691	リソースの編集後に KafkaConnectS2I によってローリング更新が実行されない。
ENTMQST-1740	ZooKeeper のスケーリングの自動化。
ENTMQST-1835	テンプレートからすべてのラベルを伝播。
ENTMQST-1839	KafkaConnect が 0 にスケーリングされると、調整が常にタイムアウトする。
ENTMQST-1859	インストール YAML から Cluster Operator のメモリー制限を増やす。
ENTMQST-1877	パーティションの減少後にトピックの CR によって誤った状態が返される。

課題番号	説明
ENTMQST-1888	**min.insync.replicas** の値を変更した後に、トピックの CR によって誤った状態が返される。
ENTMQST-1889	Kafka Entity Operator のログにプレーンテキストのパスワードが含まれる。
ENTMQST-1922	RF=1 および MIN-ISR=2 のトピックが存在すると、KafkaRoller によってローリング更新がブロックされる。
ENTMQST-1927	OCP で Cluster Operator が起動しない。
ENTMQST-1980	禁止された設定オプションとそれらの例外の解析を修正。
ENTMQST-2019	Cluster Operator をインストールしてすべての namespace を監視するための RBAC ファイルの namespace を修正。
ENTMQST-2022	1.3 から 1.4 にアップグレードした後、Kafka クラスターが重複した kafkaTopic となる。

第6章 既知の問題

ここでは、AMQ Streams 1.5 の既知の問題について説明します。

課題番号

ENTMQST-2060 - Cruise Control default **hard.goals** still include unsupported goals (Cruise Control のデフォルトの hard.goals にサポートされないゴールが含まれる)

説明および回避策

goals フィールドに、1 つ以上の**サポートされる**最適化ゴールが含まれる **KafkaRebalance** カスタムリソースを作成すると、Cruise Control によって以下のエラーが返されます。

```
Missing hard goals [NetworkInboundCapacityGoal, DiskCapacityGoal, RackAwareGoal,
NetworkOutboundCapacityGoal, CpuCapacityGoal, ReplicaCapacityGoal] in the provided goals...
```

この問題を回避するには、以下の 1 つを行います。

- **skipHardGoalCheck: true** を **KafkaRebalance** カスタムリソースに追加します。

```
apiVersion: kafka.strimzi.io/v1alpha1
kind: KafkaRebalance
metadata:
  name: my-rebalance
  labels:
    strimzi.io/cluster: my-cluster
spec:
  goals:
    - NetworkInboundCapacityGoal
    - DiskCapacityGoal
    - RackAwareGoal
    - NetworkOutboundCapacityGoal
    - ReplicaCapacityGoal
  skipHardGoalCheck: true
```

- **Kafka** リソースの **cruiseControl** プロパティーに、以下のハードゴールを指定します。

```
apiVersion: kafka.strimzi.io/v1beta1
kind: Kafka
metadata:
  name: my-cluster
spec:
  cruiseControl:
  config:
    hard.goals: >
      com.linkedin.kafka.cruisecontrol.analyzer.goals.RackAwareGoal,
      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
```

第7章 サポート対象のインテグレーション製品

AMQ Streams 1.5 は、以下の Red Hat 製品との統合をサポートします。

- OAuth 2.0 認証および OAuth 2.0 承認 (テクノロジーテクノロジープレビューとして) 用の **Red Hat Single Sign-On 7.4 以上**。
- Kafka ブリッジをセキュアにし、追加の API 管理機能を提供する **Red Hat 3scale API Management 2.6 以上**。
- データベースを監視し、イベントストリームを作成する **Red Hat Debezium 1.0 以上**。
- 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の **Service Registry 2020-04 以上**。

これらの製品によって AMQ Streams デプロイメントに導入可能な機能の詳細は、AMQ Streams 1.5 のドキュメントを参照してください。

第8章 重要なリンク

- Red Hat AMQ 7 でサポートされる設定
- Red Hat AMQ 7 コンポーネントの詳細

改訂日時：2023-01-28 11:55:05 +1000