Red Hat AMQ 2020.Q4

AMQ Streams 1.6 on OpenShift リリースノート

AMQ Streams on OpenShift Container Platform の使用

Last Updated: 2023-03-30

Red Hat AMQ 2020.Q4 AMQ Streams 1.6 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 1.6 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 1.6 は Strimzi 0.20.x をベースにしています。

本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。

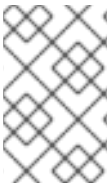
注記

本リリースで解決された改良機能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 1.6.X の KAFKA サポート（OCP 3.11 での長期サポート）

ここでは、AMQ Streams 1.6 および後続のパッチリリースでサポートされる Kafka および ZooKeeper のバージョンについて説明します。

AMQ Streams 1.6.x は、OCP 3.11 で使用する長期サポートリリースであり、OpenShift Container Platform 3.11 がサポートされる限りのみサポートされます。



注記

AMQ Streams 1.6.4 以降のパッチリリースは OCP 3.11 でのみサポートされます。OCP 4.x を使用している場合は、AMQ Streams 1.7.x 以降にアップグレードする必要があります。

AMQ LTS バージョンのサポート日付の詳細は、Red Hat ナレッジベースソリューションHow long are AMQ LTS releases supported?を参照してください。

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。以前のバージョンの Kafka は、アップグレードの目的のみで AMQ Streams 1.6.x でサポートされます。

サポートされる Kafka 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」 ページを参照してください。

1.1.1. AMQ Streams 1.6.6 および 1.6.7 での Kafka サポート

AMQ Streams 1.6.6 および 1.6.7 リリースは Apache Kafka バージョン 2.6.3 をサポートします。

ブローカーおよびクライアントアプリケーションを Kafka 2.6.3 にアップグレードする前に、Cluster Operator を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 and Kafka upgrades」を参照してください。

Kafka 2.6.3 には ZooKeeper バージョン 3.5.9 が必要です。そのため、AMQ Streams 1.6.4 / 1.6.5 からアップグレードする場合、Cluster Operator は ZooKeeper のアップグレードを実行し **ません**。

詳細は、Kafka 2.6.3 リリースノートを参照してください。

1.1.2. AMQ Streams 1.6.4 および 1.6.5 での Kafka サポート

AMQ Streams 1.6.4 および 1.6.5 リリースは、Apache Kafka バージョン 2.6.2 および ZooKeeper バージョン 3.5.9 をサポートします。

ブローカーおよびクライアントアプリケーションを Kafka 2.6.2 にアップグレードする前に、Cluster Operator を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 and Kafka upgrades」を参照してください。

Kafka 2.6.2 には ZooKeeper バージョン 3.5.9 が必要です。そのため、AMQ Streams 1.6.2 からアップグレードする際に、Cluster Operator は ZooKeeper のアップグレードを実行します。

詳細は、Kafka 2.6.2 Release Notes を参照してください。

1.1.3. AMQ Streams 1.6.0 および 1.6.2 での Kafka のサポート

AMQ Streams 1.6.0 および 1.6.2 は Apache Kafka バージョン 2.6.0 をサポートします。

ブローカーおよびクライアントアプリケーションを Kafka 2.6.0 にアップグレードする前に、Cluster Operator を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 and Kafka upgrades」を参照してください。

詳細は、Kafka 2.5.0 および Kafka 2.6.0 のリリースノートを参照してください。

Kafka 2.6.0 には、Kafka 2.5.x と同じ ZooKeeper バージョン (ZooKeeper バージョン 3.5.7 / 3.5.8) が必要です。そのため、AMQ Streams 1.5 からアップグレードする場合、Cluster Operator は ZooKeeper のアップグレードを実行し **ません**。

1.2. コンテナーイメージの JAVA 11 への移行

AMQ Streams コンテナーイメージは、Java ランタイム環境 (JRE) として Java 11 に移行されます。イメージの JRE バージョンは OpenJDK 8 から OpenJDK 11 に変更されます。

1.3. CLUSTER OPERATOR のロギング

Cluster Operator のロギングは、Cluster Operator がデプロイされる際に自動作成される ConfigMap を使用して設定されるようになりました。ConfigMap は以下の新規 YAML ファイルに記述されます。

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

Cluster Operator のロギングを設定するには、以下を実行します。

1. **050-ConfigMap-strimzi-cluster-operator.yaml** ファイルで **data.log4j2.properties** フィールドを編集します。

   Cluster Operator のロギングの設定例

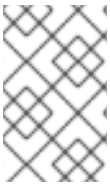
   ```
   kind: ConfigMap
   apiVersion: v1
   metadata:
     name: strimzi-cluster-operator
     labels:
       app: strimzi
   data:
     log4j2.properties: |
       name = COConfig
       monitorInterval = 30
   ```

```
appender.console.type = Console
appender.console.name = STDOUT
# ...
```

2. カスタムリソースを編集します。

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

ログのリロード頻度を変更するには、**monitorInterval** フィールドに、秒単位で時間を設定します (デフォルトのリロード時間は 30 秒 です)。



注記

この変更により、**STRIMZI_LOG_LEVEL** 環境変数が **060-Deployment-strimzi-cluster-operator.yaml** ファイルから削除されました。代わりに ConfigMap にログレベルを設定します。

「Cluster Operator の設定」を参照してください。

1.4. OAUTH 2.0 承認

OAuth 2.0 承認のサポートは、テクノロジープレビューから AMQ Streams の一般利用可能なコンポーネントに移行されます。

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。

1.5. OPEN POLICY AGENT (OPA) の統合

Open Policy Agent (OPA) は、オープンソースのポリシーエンジンです。OPA と AMQ Streams を統合して、Kafka ブローカーでのクライアント操作を許可するポリシーベースの承認メカニズムとして機能します。

クライアントからリクエストが実行されると、OPA は Kafka アクセスに定義されたポリシーに対してリクエストを評価し、リクエストを許可または拒否します。

Kafka クラスター、コンシューマーグループ、およびトピックのアクセス制御を定義できます。たとえば、プロデューサークライアントから特定のブローカートピックへの書き込みアクセスを許可する承認ポリシーを定義できます。

KafkaAuthorizationOpa スキーマ参照

注記

Red Hat は OPA サーバーをサポートしません。

1.6. 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 トピックへ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 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。

- 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。

1.7. 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。保存後、スキーマを使用するアプリケーションから参照され、アプリケーションが送受信するメッセージがこれらのスキーマと互換性を維持するようにします。

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 の改良された機能

以下で紹介されている機能拡張の概要を説明します。

- Kafka 2.6.2 は Kafka 2.6.2 Release Notes を参照してください（AMQ Streams 1.6.4 のみに適用）。
- Kafka 2.6.1 は Kafka 2.6.1 リリースノートを参照してください (AMQ Streams 1.6.4 のみに適用)。
- Kafka 2.6.0。Kafka 2.6.0 リリースノート を参照してください。

2.2. KAFKA BRIDGE の改良された機能

本リリースには、AMQ Streams の Kafka Bridge コンポーネントに対して改良された以下の機能が含まれています。

パーティションおよびメタデータの取得

Kafka Bridge は以下の操作をサポートするようになりました。

- 特定のトピックのパーティションリストを取得します。

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

- パーティション ID、リーダーブローカー、レプリカ数などの指定のパーティションのメタデータを取得します。

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

『Kafka Bridge API reference』を参照してください。

Kafka メッセージヘッダーのサポート

Kafka Bridge を使用して送信されたメッセージに Kafka メッセージヘッダーが含まれるようになりました。

/topics エンドポイントへの POST リクエストでは、リクエストボディーに含まれるメッセージペイロードに任意でヘッダーを指定できます。メッセージヘッダーの値はバイナリー形式である必要があり、Base64 としてエンコードされる必要があります。

Kafka メッセージヘッダーのあるリクエストの例

```
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=="
            }
          ]
        },
    ]
}'
```

「Kafka Bridge へのリクエスト」を参照してください。

2.3. OAUTH 2.0 認証および承認

本リリースには、OAuth 2.0 トークンベースの認証および承認に対して改良された以下の機能が含まれています。

セッションの再認証
AMQ Streams の OAuth 2.0 認証は Kafka ブローカーの **セッションの再認証** をサポートするようになりました。これは、Kafka クライアントと Kafka ブローカー間の認証された OAuth 2.0 セッションの最大期間を定義します。セッションの再認証は、高速のローカル JWT とイントロスペクションエンドポイントの両方のタイプでサポートされます。

セッションの再認証を設定するには、Kafka ブローカーの OAuth 2.0 設定に新しい **maxSecondsWithoutReauthentication** オプションを使用します。

特定のリスナーでは、**maxSecondsWithoutReauthentication** では以下を行うことができます。

- セッションの再認証を有効にすることができます。
- Kafka クライアントと Kafka ブローカー間の認証されたセッションの最大期間を秒単位で設定できます。

1時間後にセッション再認証する設定の例

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

認証されたセッションは、設定された**maxSecondsWithoutReauthentication** を超えた場合や、アクセストークンの有効期限に達した場合に閉じられます。その後、クライアントは再度承認サーバーにログインし、新しいアクセストークンを取得して、Kafka ブローカーへ再認証する必要があります。これにより、既存の接続上で新しい認証セッションが確立されます。

次に再認証が必要な場合、クライアントによって試行された操作 (再認証以外) によって、ブローカーは接続を終了します。

以下を参照してください。「Kafka ブローカーのセッション再認証」および「Kafka ブローカーの OAuth 2.0 サポートの設定」

JWKS キーの更新間隔

高速のローカル JWT トークン検証を使用するように Kafka ブローカーを設定する場合に、外部リスナー設定に **jwksMinRefreshPauseSeconds** オプションを設定できるようになりました。これは、承認サーバーによって発行される JSON Web Key Set (JWKS) 公開鍵の更新をブローカーが試行する最小間隔を定義します。

本リリースでは、Kafka ブローカーは不明な署名キーを検出した場合に、通常の更新スケジュールを待たずに JWKS キーの更新を即座に試行します。

JWKS キーの更新を実行する間隔が 2 分間である設定の例

```
    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 キーの更新スケジュールは、**jwksRefreshSeconds** オプションに設定されます。不明な署名キーが検出されると、JWKS キーの更新は更新スケジュールとは別にスケジュールされます。最後の更新からの経過時間が **jwksMinRefreshPauseSeconds** で指定される間隔に達するまで、更新は開始されません。

jwksMinRefreshPauseSeconds のデフォルト値は **1** です。

「Kafka ブローカーの OAuth 2.0 サポートの設定」を参照してください。

Red Hat Single Sign-On からの付与 (Grants) の更新

Red Hat Single Sign-On により、OAuth 2.0 のトークンベースの承認に新たな設定オプションが追加されました。Kafka ブローカーの設定時に、Red Hat SSO Authorization Service からの付与 (Grants) の更新に関連する以下のオプションを定義できます。

- **grantsRefreshPeriodSeconds**:連続する付与 (Grants) 更新実行の間隔。デフォルト値は **60** です。**0** 以下に設定されている場合、付与 (Grants) の更新は無効になります。

- **grantsRefreshPoolSize**:アクティブなセッションの付与 (Grants) を並行して取得できるスレッドの数。デフォルト値は **5** です。

「OAuth 2.0 トークンベース承認の使用」および「OAuth 2.0 承認サポートの設定」を参照してください。

Red Hat Single Sign-On でのパーミッション変更の検出
本リリースでは、**keycloak** (Red Hat SSO) 承認によってアクティブなセッションのパーミッションの変更が定期的にチェックされるようになりました。ユーザーの変更とパーミッション管理の変更がリアルタイムで検出されるようになりました。

2.4. KAFKA BRIDGE および CRUISE CONTROL のメトリクス

メトリクス設定を Kafka Bridge および Cruise Control に追加できるようになりました。

Kafka Bridge および Cruise Control のサンプルメトリクスファイルは、以下を含む AMQ Streams で提供されます。

- メトリクス設定を含むカスタムリソース YAML ファイル
- Grafana ダッシュボード JSON ファイル

メトリクス設定がデプロイされ、Prometheus および Grafana が設定されると、監視に Grafana ダッシュボードのサンプルを使用できます。

AMQ Streams で提供されるサンプルメトリクスファイル

```
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 メトリクス設定を含むカスタムリソースの例

コンポーネント	カスタムリソース	サンプル YAML ファイル
Kafka および ZooKeeper	**Kafka**	**kafka-metrics.yaml**
Kafka Connect	**KafkaConnect** および **KafkaConnectS2I**	**kafka-connect-metrics.yaml**
Kafka MirrorMaker 2.0	**KafkaMirrorMaker2**	**kafka-mirror-maker-2-metrics.yaml**
Kafka Bridge	**KafkaBridge**	**kafka-bridge-metrics.yaml**
Cruise Control	**Kafka**	**kafka-cruise-control-metrics.yaml**

「Kafka へのメトリクスの導入」を参照してください。

注記

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。

2.5. ロギング変更の動的更新

本リリースでは、ほとんどのカスタムリソースのロギングレベル (インラインと外部の両方) を変更しても、Kafka クラスターへのローリングアップデートがトリガーされなくなりました。再起動せずに、ロギングの変更が動的に適用されるようになりました。

この機能改良は、以下のリソースに適用されます。

- Kafka クラスター
- Kafka Connect および Kafka Connect S2I
- Kafka Mirror Maker 2.0
- Kafka Bridge

Mirror Maker や Cruise Control には適用されません。

ConfigMap 経由で外部ロギングを使用する場合、ロギングアペンダーの変更時にローリングアップデートがトリガーされます。以下に例を示します。

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

『**AMQ Streams on OpenShift の使用**』の「外部ロギング」および「デプロイメント設定」の章を参照してください。

2.6. メトリクスのスクレイピングに使用される PODMONITORS

本リリースでは、Prometheus メトリクスが Pod (Kafka、ZooKeeper、Kafka Connect など) からスクレイピングされる方法が変更になりました。

メトリクスは **strimzi-pod-monitor.yaml** で定義される **PodMonitors** によってのみ Pod からスクレイピングされるようになりました。以前のバージョンでは、これは **ServiceMonitors** および **PodMonitors** によって実行されていました。**ServiceMonitors** は本リリースで AMQ Streams から **削除** されました。

「PodMonitors を使用するためのモニタリングスタックのアップグレード」の説明にしたがって、**PodMonitors** を使用するようにモニタリングスタックをアップグレードする必要があります。

この変更により、以下の要素が Kafka および ZooKeeper に関連するサービスから **削除** されました。

- **tcp-prometheus** 監視ポート (ポート 9404)
- Prometheus アノテーション

この変更は、以下のサービスに適用されます。

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

Prometheus アノテーションを追加するには、「OpenShift リソースのカスタマイズ」の説明どおりに、関連する AMQ Streams カスタムリソースで **template** プロパティーを使用する必要があります。

PodMonitors を使用するためのモニタリングスタックのアップグレード

Kafka クラスターの監視が中断されないようにするには、AMQ Streams 1.6 にアップグレードする **前** に以下の手順を実行します。

1. 新しい **AMQ Streams 1.6** インストールアーティファクトを使用して、**strimzi-pod-monitor.yaml** ファイルを AMQ Streams 1.5 クラスターに適用します。

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

2. AMQ Streams 1.5 クラスターから既存の **ServiceMonitor** リソースを削除します。
3. **additional-scrape-configs** という名前の **Secret** を削除します。
4. **AMQ Streams 1.6** インストールアーティファクトで提供される **prometheus-additional.yaml** ファイルから **additional-scrape-configs** という名前の新しい **Secret** を作成します。
5. Prometheus ユーザーインターフェースの Prometheus ターゲットが稼働していることを確認します。
6. 「Cluster Operator のアップグレード」から開始して AMQ Streams 1.6 へのアップグレードを行います。

AMQ Streams 1.6 へのアップグレードが完了したら、AMQ Streams 1.6 の Grafana ダッシュボードのサンプルをロードできます。

「Kafka へのメトリクスの導入」を参照してください。

2.7. 汎用リスナーの設定

本リリースに **GenericKafkaListener** スキーマが導入されました。

スキーマは、**Kafka** リソースの Kafka リスナーを設定するもので、非推奨である **KafkaListeners** スキーマの代わりに使用されます。

GenericKafkaListener スキーマを使用すると、名前とポートが一意であれば、必要なリスナーをいくつでも設定できます。**リスナー設定** は配列として定義されますが、非推奨の形式もサポートされます。

GenericKafkaListener スキーマ参照

新しい設定へのリスナーの更新

KafkaListeners スキーマは、**plain**、**tls** および **external** リスナーのサブプロパティーを使用し、それぞれに固定ポートを使用していました。アップグレードプロセスのいずれかの段階で、**KafkaListeners** スキーマを使用して設定されたリスナーを **GenericKafkaListener** スキーマの形式に変換する必要があります。

たとえば、現在 **Kafka** 設定で以下の設定を使用しているとします。

これまでのリスナー設定

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

以下を使用して、リスナーを新しい形式に変換します。

新しいリスナー設定

```
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. すべてのリスナーに TLS プロパティーが必要になります。
2. オプション： **ingress**、**loadbalancer**、**nodeport**、**route**。

必ず **正確な** 名前とポート番号を使用してください。

これまでの形式で使用された追加の **configuration** または **overrides** プロパティーは、新しい形式に更新する必要があります。

リスナー **設定** に追加された変更：

- **overrides** が **configuration** セクションとマージされる。
- **dnsAnnotations** の名前が **アノテーション**になった。
- **preferredAddressType** の名前が **preferredNodePortAddressType** に変更された。
- **address** の名前が **alternativenames** に変更された。
- **loadBalancerSourceRanges** および **externalTrafficPolicy** が、現在の非推奨の **テンプレート**から**リスナー設定に移動する。**

すべてのリスナーが、アドバタイズされたホスト名およびポートの設定をサポートするようになりました。

例として、以下の設定を見てみましょう。

従来の追加リスナー設定

```
listeners:
  external:
    type: loadbalancer
    authentication:
      type: tls
    overrides:
      bootstrap:
        dnsAnnotations:
          #...
```

これを以下に変更します。

新しい追加リスナー設定

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

重要

後方互換性を維持するため、新しいリスナー設定にある名前およびポート番号を使用する **必要** があります。他の値を使用すると、Kafka リスナーおよび Kubernetes サービスの名前が変更されます。

2.8. MIRRORMAKER 2.0 トピック名変更の更新

MirrorMaker 2.0 アーキテクチャーは、リモートトピックの名前を自動変更してソースクラスターを表すことで、双方向レプリケーションをサポートします。元のクラスターの名前の先頭には、トピックの名前が追加されます。

必要に応じて、**IdentityReplicationPolicy** をソースコネクター設定に追加することで、名前の自動変更をオーバーライドできるようになりました。この設定が適用されると、トピックには元の名前が保持されます。

```
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 リモートトピック名の自動変更をオーバーライドするポリシーを追加します。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。

オーバーライドは、**アクティブ/パッシブ**クラスター設定でバックアップを作成したり、データを別のクラスターに移行する場合などに便利です。いずれの場合でも、リモートトピックの名前を自動的に変更したくないことがあります。

「Kafka MirrorMaker 2.0 の設定」を参照してください。

2.9. HOSTALIASES のサポート

Kubernetes テンプレートおよび Pod のデプロイメントをカスタマイズする場合に、**hostAliases** を設定できるようになりました。

hostAliasesの設定例

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

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

ホストおよび IP の一覧が指定された場合、Pod の **/etc/hosts** ファイルに注入されます。これは特に、クラスター外部の接続がユーザーによっても要求される場合に Kafka Connect または MirrorMaker で役立ちます。

PodTemplate スキーマ参照

2.10. 調整されたリソースメトリクス

新規 Operator メトリクスは、指定されたリソースの状態に関する情報 (つまり、正常に調整されたかどうか) を提供します。

調整されたリソースメトリクス定義

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

2.11. KAFKAUSER のシークレットメタデータ

User Operator によって作成される **Secret** のテンプレートプロパティーを使用できるようになりました。**KafkaUserTemplate** を使用すると、ラベルとアノテーションを使用して、**KafkaUser** リソースに対して **Secret** が生成される方法を定義するメタデータを設定できます。

KafkaUserTemplate を示す例

```
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
  # ...
```

KafkaUserTemplate スキーマ参照

2.12. コンテナーイメージの追加ツール

AMQ Streams コンテナーイメージに以下のツールが追加されました。

- **jstack**

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

2.13. KAFKA EXPORTER サービスの削除

Kafka Exporter サービスは AMQ Streams から **削除** されました。Prometheus は **PodMonitor** 宣言を使用して Kafka Exporter Pod から直接 Kafka Exporter メトリクスをスクレイピングするようになったため、このサービスは不要になりました。

「Kafka へのメトリクスの導入」を参照してください。

2.14. KAFKA 管理ツールで非推奨になった ZOOKEEPER オプション

--zookeeper オプションは、以下の Kafka 管理ツールで非推奨となりました。

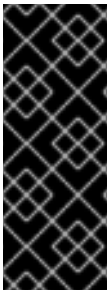
- **bin/kafka-configs.sh**
- **bin/kafka-leader-election.sh**
- **bin/kafka-topics.sh**

これらのツールを使用する場合は、**--bootstrap-server** オプションを使用して、接続する Kafka ブローカーを指定する必要があります。以下は例になります。

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

--zookeeper オプションは引き続き動作しますが、今後の Kafka リリースのすべての管理ツールから削除されます。これは、Kafka の ZooKeeper への依存関係を削除するために Apache Kafka プロジェクトが進めている作業の一部です。

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リバランス

Cruise Control は本リリースでもテクノロジープレビューの機能であり、新たな改良 が加えられました。

Cruise Control を AMQ Streams クラスターにデプロイし、これを使用して **最適化ゴール** (CPU、ディスク、およびネットワークの負荷に対して事前定義された制約) を使用した Kafka クラスターのリバランスを行えるようになりました。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。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使用できます。

- 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。
- 最適化の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。

3.1.1. テクノロジープレビューの改良

テクノロジープレビューである Cruise Control を使用したクラスターのリバランスに、以下の改良が加えられました。

リバランスパフォーマンスチューニング

新しい 5 つの **パフォーマンスチューニングオプション** を使用すると、クラスターのリバランスの実行方法を制御し、パフォーマンスへの影響を軽減することができます。

クラスターのリバランスを構成する **パーティション再割り当てコマンド** のバッチごとに、以下を設定できます。

- ブローカー毎のパーティション同時移動の最大数 (デフォルトは 5)。
- ブローカー内でのパーティション同時実行の最大数 (デフォルトは 2)。
- リーダーの同時移動の最大数 (デフォルトは 1000)。
- パーティションの再割り当てに割り当てるバイト/秒単位の帯域幅 (デフォルトは制限なし)。

- レプリカの移動ストラテジー (デフォルトは **BaseReplicaMovementStrategy** です)

以前のバージョンでは、AMQ Streams はこれらのオプションを Cruise Control から継承するため、デフォルト値を調整できませんでした。

Cruise Control サーバー、個別のリバランス、またはその両方のパフォーマンスチューニングオプションを設定できます。

- Cruise Controlサーバーの場合は、**Kafka** のカスタムリソースの **spec.cruiseControl.config** にオプションを設定します。
- クラスターリバランスの場合は、**KafkaRebalance** カスタムリソースの **spec** プロパティーにオプションを設定します。

「リバランスパフォーマンスチューニングの概要」を参照してください。

最適化プロポーザルからトピックを除外

最適化プロポーザルから 1 つ以上のトピックを除外できるようになりました。これらのトピックは、クラスターリバランスのパーティションレプリカおよびパーティションリーダーシップの移動の計算に含まれていません。

トピックを除外するには、**KafkaRebalance** カスタムリソースのトピック名と一致する正規表現を、**spec.excludedTopicsRegex** に指定します。

生成された最適化プロポーザルの **excludedTopics** プロパティーに除外されたトピックが表示されます。

「リバランスパフォーマンスチューニングの概要」を参照してください。

CPU 容量ゴールのサポート

CPU 容量に基づいた Kafka クラスターのリバランスが、以下の設定でサポートされるようになりました。

- **CpuCapacityGoal** 最適化の目的
- **cpuUtilization** 容量制限

CPU 容量ゴールは、各ブローカーの CPU 使用率がしきい値の最大割合 (パーセント) を越えないようにします。デフォルトのしきい値は、ブローカーごとに CPU 容量の 100% に設定されます。

しきい値の割合を減らすには、**Kafka** カスタムリソースに **cpuUtilization** 容量制限を設定します。容量制限はすべてのブローカーに適用されます。

CPU 容量は Cruise Control のハードゴールとして事前設定されます。そのため、**Kafka.spec.cruiseControl.config** の **hard.goals** プロパティーで事前設定されたハードゴールをオーバーライドしない限り、Cruise Control からハードゴールとして継承されます。

KafkaRebalance カスタムリソースの CPU 容量ゴールの設定例

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

```
  goals:
    - CpuCapacityGoal
    - DiskCapacityGoal
  #...
```

CPU 使用容量制限の割合の設定例

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

「最適化ゴールの概要」および「容量の設定」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. KAFKALISTENERS スキーマ

Kafka.KafkaSpec.KafkaClusterSpec リソースの **KafkaListeners** スキーマは非推奨になりました。

詳細は、改良された機能の「汎用リスナーの設定」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 1.6.x で修正された問題が記載されています。AMQ Streams 1.6.x を OpenShift Container Platform 3.11 で使用している場合は、Red Hat は最新のパッチリリースへアップグレードすることを推奨します。

修正された問題の詳細は、以下を参照してください。

- Kafka 2.6.3 は、『Kafka 2.6.3 リリースノート』を参照してください。
- Kafka 2.6.2 は『Kafka 2.6.2 Release Notes』を参照してください。
- Kafka 2.6.1 は『Kafka 2.6.1 リリースノート』を参照してください。
- Kafka 2.6.0。Kafka 2.6.0 リリースノートを参照してください。

5.1. AMQ STREAMS 1.6.7 で修正された問題

AMQ Streams 1.6.7 パッチリリース（長期サポート）が利用できるようになりました。

AMQ Streams 1.6.7 は、OpenShift Container Platform 3.11 でのみ使用する最新の長期サポートリリースであり、OpenShift Container Platform 3.11 がサポートされる限りのみサポートされます。

AMQ Streams 1.6.7 は OCP 3.11 でのみサポートされます。

AMQ Streams の製品イメージがバージョン 1.6.7 にアップグレードされました。

AMQ Streams 1.6.7 で解決された問題の詳細は、AMQ Streams 1. 6.x Resolved Issues を参照してください。

Log4j の脆弱性

AMQ Streams には log4j 1.2.17 が含まれています。このリリースでは、log4j の脆弱性が数多く修正されています。

このリリースで対応する脆弱性の詳細は、以下の CVE の記事を参照してください。

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

5.2. AMQ STREAMS 1.6.6 で修正された問題

AMQ Streams 1.6.6 で解決された問題の詳細は、AMQ Streams 1.6 .x Resolved Issues を参照してください。

Log4j2 の脆弱性

AMQ Streams には log4j2 2.17.1 が含まれています。本リリースでは、log4j2 の脆弱性が多数修正されています。

このリリースで対応する脆弱性の詳細は、以下の CVE の記事を参照してください。

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

5.3. AMQ STREAMS 1.6.5 で修正された問題

AMQ Streams 1.6.5 で解決された問題の詳細は、「AMQ Streams 1.6..x Resolved Issues」を参照してください。

Log4j2 の脆弱性

1.6.5 リリースでは、log4j2 を使用する AMQ Streams コンポーネントのリモートコード実行脆弱性が修正されました。この脆弱性により、システムが承認されていないソースから文字列の値をログに記録した場合に、サーバーでのリモートコード実行が可能になる可能性がありました。これは 2.0 から 2.14.1 までの log4j バージョンに影響を与えます。

詳細は、CVE-2021-44228 を参照してください。

5.4. AMQ STREAMS 1.6.4 で修正された問題

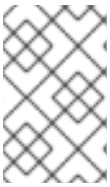
AMQ Streams 1.6.4 で解決された問題の詳細は、「AMQ Streams 1.6..x Resolved Issues」を参照してください。

5.5. AMQ STREAMS 1.6.2 で修正された問題

AMQ Streams 1.6.2 パッチリリースを使用できます。このリリースには、Kafka Connect に関連する修正が複数含まれています。

AMQ Streams の製品イメージは変更されておらず、バージョンは 1.6 のままになります。

AMQ Streams 1.6.2 で解決された問題の詳細は、「AMQ Streams 1.6.2 Resolved Issues」を参照してください。



注記

CVE の更新に従い、Operator Lifecycle Manager (OLM) によって管理される AMQ Streams のバージョンが 1.6.1 に変更されました。混乱を避けるために、AMQ Streams 1.6 のパッチリリースのバージョンには 1.6.2 が割り当てられました。

5.6. AMQ STREAMS 1.6.0 で修正された問題

課題番号	説明
ENTMQST-2049	Kafka Bridge:Kafka コンシューマーは group-consumerid キーで追跡する必要があります。
ENTMQST-2289	ダウングレードバージョンよりも古いメッセージバージョンでダウングレード可能。
ENTMQST-2292	パッチ適用前の Diff PodDisruptionBudgets で調整ごとに再作成されないようにする。
ENTMQST-2146	OCP 上の MirrorMaker 2 がヘッダーでメッセージを適切にミラーリングしない。
ENTMQST-2147	MirrorMaker 2 がコネクターで Jaeger のトレースを適切に設定しない。
ENTMQST-2099	toleration を空の値に設定すると、Kafka クラスターがローリングアップデートを実行を繰り返す。
ENTMQST-2084	ドキュメントの ZooKeeper バージョンが AMQ Streams 1.5 のバージョンと一致しない。
ENTMQST-2340	KafkaConnect API を使用すると Operator の接続リークが発生。
ENTMQST-2338	Connect にマウントされたドットのあるシークレットまたは ConfigMap の修正。
ENTMQST-2294	OLM インストール: yaml の authentication のスペルミス。

表5.1 CVE (Common Vulnerabilities and Exposures) の修正

課題番号	説明
ENTMQST-2332	CVE-2020-13956 httpclient: apache-httpclient: リクエスト URI での不適切な authority コンポーネントへの対処が正しくない [amq-st-1]

第6章 既知の問題

ここでは、AMQ Streams の既知の問題について説明します。

課題番号

ENTMQST-2386: JBOD ボリュームの追加または削除が機能しない

説明および回避策

AMQ Streams では、新しいバージョンの Kubernetes に JBOD ボリュームを追加または削除できません。**StatefulSet コントローラー** の検証が改善されたため、Pod を順番に (pod-0 の次が pod-1 のように) 削除および再作成することのみ可能です。現時点で、AMQ Streams のローリングアップデート手順は Pod の削除を順番にトリガーしません。

すべての Pod を順番に手動で削除することが、現在の回避策となります。Pod が再作成されると、**StatefulSet Controller** は失敗せず、想定どおりに動作します。

問題

SHA ダイジェストで指定された環境では S2I はソースイメージをダウンロードできない

説明および回避策

Openshift イメージストリームの制限により、Kafka Connect S2I は接続されていないクラスターで新しいコネクタープラグインの構築に失敗します。**ImageContentSourcePolicy** を指定してイメージレジストリーへのパスが変更された場合、これは無視されます。代わりに、イメージストリームはソースリポジトリーからダウンロードを試みます。

Kafka Connect を手動でデプロイすることが、現在の回避策となります。

第7章 サポートされる統合製品

AMQ Streams 1.6 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 承認用の **Red Hat Single Sign-On 7.4 以上**。
- Kafka Bridge 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-Q4 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 1.6 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 でサポートされる構成
- Red Hat AMQ 7 コンポーネントの詳細

改訂日時： 2022-07-14 11:36:10 +1000