9.4

コンテナー内の *IBM MQ*

注記

本書および本書で紹介する製品をご使用になる前に、299 ページの『特記事項』に記載されている情報をお読みください。

本書は、 IBM® MQ バージョン 9 リリース 4、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様が IBM に情報を送信する場合、お客様は IBM に対し、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で情報を使用または配布する非独占的な権利を付与します。

© Copyright International Business Machines Corporation 2007 年, 2025.

目次

コンテナー内の IBM MQ と IBM Cloud Pak for Integration.....5
製品情報.....5
IBM MQ Operator のリリース履歴.....5
キュー・マネージャ・コンテナ・イメージのリリース履歴.....18
計画.....27
コンテナー内の IBM MQ の使用方法の選択.....28
コンテナーでの IBM MQ のサポート.....28
コンテナーでの IBM MQ のライセンス交付の計画.....37
IBM MQ Operator のストレージの計画.....38
コンテナー内の IBM MQ の高可用性の計画.....40
コンテナー内の IBM MQ の災害復旧.....46
コンテナー内の IBM MQ のセキュリティーの計画.....49
コンテナー内の IBM MQ のスケーラビリティーとパフォーマンスの計画.....55
準備、インストール、およびアップグレード.....55
IBM MQ Operator のインストールおよびアップグレード.....56
プルシークレットを作成して「Kubernetes」を使用する準備.....86
をインストールして Kubernetes を使う準備をする。 IBM License Service.....87
Podman または「Docker 使用するために、ビルド済みのコンテナイメージを取り込む.....87
独自のコンテナー・イメージのビルドによる IBM MQ の準備.....87
デプロイおよび構成.....97
IBM MQ Operator を使用したキュー・マネージャーのデプロイおよび構成.....97
Deploying and configuring queue managers on Kubernetes.....173
Podman にキュー・マネージャーをデプロイする。 188
IBM MQ Operator へのマイグレーション.....193
必要な機能を利用できることの確認.....194
キュー・マネージャー構成の抽出.....195
オプション: キュー・マネージャーの鍵と証明書の抽出および取得.....195
オプション: LDAP の構成.....197
オプション: IBM MQ 構成内の IP アドレスとホスト名の変更.....205
コンテナー環境用のキュー・マネージャー構成の更新.....206
コンテナーで実行されている IBM MQ のためのターゲット HA アーキテクチャーの選択.....209
キュー・マネージャー用のリソースの作成.....210
Red Hat OpenShift での新しいキュー・マネージャーの作成.....211
新規コンテナー・デプロイメントの検証.....215
運行中.....217
IBM MQ Operator を使用した IBM MQ.....217
ネイティブ HA キュー・マネージャーの状況の表示.....230
ネイティブ HA CRR の切り替えとフェイルオーバー.....236
ネイティブ HA キュー・マネージャーの終了.....240
トラブルシューティング.....241
コンテナー内の IBM MQ の計画外の再始動のトラブルシューティング.....241
IBM MQ Operator に関する問題のトラブルシューティング.....243
に配置されたキュー・マネージャのトラブルシューティング情報を収集する Kubernetes.....248
参照.....249
IBM MQ Operator のライセンスと API リファレンス.....249
IBM MQ コンテナライセンス注釈.....281
IBM MQ によって適用されるコンテナライセンス注釈。 IBM MQ Operator.....286
IBM MQ Advanced コンテナー・イメージ.....287
IBM MQ ライセンスコンテナイメージ.....291
IBM MQ Advanced for Developers コンテナー・イメージ.....291
IBM MQ MFT Agent for Developers コンテナー・イメージ.....293

IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 295

特記事項..... 299

プログラミング・インターフェース情報..... 300

商標..... 300

コンテナー内の IBM MQ と IBM Cloud Pak for Integration

コンテナーを使用すると、IBM MQ キュー・マネージャーや IBM MQ クライアント・アプリケーションをすべての依存関係とともに、ソフトウェア開発のために標準化された単位でパッケージ化できます。

Red Hat® OpenShift®で IBM MQ Operator を使用して IBM MQ を実行できます。 このテストは、IBM Cloud Pak® for Integration、IBM MQ Advanced、または IBM MQ Advanced for Developers を使用して行うことができます。

また、IBM MQ をお客様が作成した独自のコンテナーで実行することもできます。

IBM MQ Operator について詳しくは、以下のリンクを参照してください。

コンテナー内の IBM MQ について

コンテナー内の IBM MQ の使用を開始するにあたって役に立つ情報を紹介します。

コンテナーは、ランタイム環境でコードのパッケージ化と分離を可能にするテクノロジーであり、同じインフラストラクチャー上の他のソフトウェアから分離された方法で実行できます。 これにより、キュー・マネージャーまたはアプリケーションを環境 (開発、テスト、実動など) 間で簡単に移動できます。 最新のコンテナー・オーケストレーター (Red Hat OpenShift Container Platform や Kubernetes など) は、同じマシン上で多くのタイプのコンテナーを実行できます。各コンテナーは、リソース、セキュリティー、および障害に関して互いに分離されています。

IBM MQ キュー・マネージャーまたは IBM MQ アプリケーションをコンテナー内で実行できます。

関連情報
コンテナーとは

##

IBM MQ Operator のリリース履歴

注:

- 以前の IBM MQ オペレーターについては、 IBM MQ 9.3 資料の IBM MQ Operator のリリース履歴 を参照してください。
- 今後の IBM MQ の更新については、「IBM MQ 推奨フィックスおよび計画メンテナンス・リリース日」ページ全体を参照してください。

IBM MQ Operator 3.6.0

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.2

オペレーター・チャネル
v3.6

.spec.version に許可される値
9.4.3.0-r1

マイグレーション中に許可される .spec.version の値

9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1,

© Copyright IBM Corp. 2007 年, 2025

9.4.0.10-r1, 9.4.0.10-r2, 9.4.0.11-r1, 9.4.0.11-r2, 9.4.0.11-r3, 9.4.1.0-r1, 9.4.1.0-r2, 9.4.1.1-r1, 9.4.2.0-r1, 9.4.2.0-r2, 9.4.2.1-r1, 9.4.2.1-r2

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

新機能

- アクセスロギングの監査要件への準拠を促進するために、ウェブフックアクセスの検証のロギングを改善しました。 詳細については、150 ページの『コンテナ内の IBM MQ に対する監査ロギングの設定』を参照してください。
- IBM MQ Operator は、 IPv6 デュアルスタック環境向けに検証されており、米国連邦政府の最新コンプライアンス要件に適合している。
- これで、 IBM MQ キュー・マネージャ・サービスの `ipFamilyPolicy` と `ipFamilies` の設定を行うことができます。 設定は `spec.service` で可能です。 この設定はすべてのキュー・マネージャ・サービスに適用されます。
- IBM MQ ライセンスアノテーションを使用して、IBM MQ Operator でデプロイされたキューマネージャを構成できるようになりました。 詳しくは、161 ページの『を使用して、 IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。 このアップデートは、すべての IBM MQ 9.4 キュー・マネージャに適用されます
- 新しい IBM MQ Native HA および Cross-Region Replication アドオンライセンスを追加することで、IBM MQ ライセンスのキューマネージャで Native HA および Cross-Region Replication 機能を使用できるようになりました。 詳しくは、116 ページの『IBM MQ 、キューマネージャのネイティブ HA とクロスリージョンレプリケーションを設定するには、次のようにします。 IBM MQ Operator』を参照してください。 このアップデートは、 IBM MQ 9.4.2 キュー・マネージャーにも適用される。

IBM MQ Operator 3.5.3

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.1

オペレーター・チャネル
v3.5

`.spec.version` に許可される値
9.4.2.1-r2

マイグレーション中に許可される `.spec.version` の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2, 9.4.0.11-r1, 9.4.1.0-r1, 9.4.1.0-r2, 9.4.1.1-r1, 9.4.2.0-r2, 9.4.2.1-r1

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点

- 対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7234178

IBM MQ Operator 3.5.2

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.1

オペレーター・チャネル
v3.5

.spec.version に許可される値
9.4.2.1-r1

マイグレーション中に許可される .spec.version の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2, 9.4.0.11-r1, 9.4.1.0-r1, 9.4.1.0-r2, 9.4.1.1-r1, 9.4.2.0-r2

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点

- IBM MQ OpenID Connect (OIDC)のクライアント ID とクライアントシークレットが環境変数ではなくボリュームに保存されるようになりました。
- 対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7232272

IBM MQ Operator 3.5.1

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.1

オペレーター・チャネル
v3.5

.spec.version に許可される値
9.4.2.0-r2

マイグレーション中に許可される .spec.version の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2, 9.4.1.0-r1, 9.4.1.0-r2, 9.4.1.1-r1, 9.4.2.0-r1

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点

- 特定の条件下で、 IntegrationKeycloakClient リソースが不正なクライアント ID で更新される問題を修正しました。
- 対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7228945

IBM MQ Operator 3.5.0

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.1

オペレーター・チャネル
v3.5

.spec.version に許可される値
9.4.2.0-r1

マイグレーション中に許可される .spec.version の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.1.0-r1, 9.4.1.0-r2, 9.4.1.1-r1

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

新機能

- キューマネージャーがデプロイされる際に、追加のファイルを供給できるようになりました。 これらのファイルは、指定された場所にある IBM MQ コンテナにマウントされます。 詳しくは、102 ページの『例：キューマネージャーに追加のファイルを供給する』を参照してください。
- キューマネージャーをデプロイする際に、追加の環境変数を指定できるようになりました。 詳細については、103 ページの『例：キューマネージャー用の追加環境変数の提供』を参照してください。
- IBM MQ 9.4.2.0-r1 のキューマネージャーは、 によって設定され、 によって作成された証明書を使用して、 経由でメトリクスを提供します。 IBM MQ Operator OpenShift HTTPS 詳しくは、 218 ページの『IBM MQ Operator 使用時のモニター』を参照してください。
- IBM MQ CASE 3.5.0 以降は、最新のキューマネージャーのバージョンだけが含まれます。 例えば、IBM MQ CASE 3.5.0 にはキューマネージャーのバージョン 9.4.2.0-r1 のみが含まれています。 エアギャップインストールを行うと、イメージミラーリングのステップが高速化されますが、最新ではないキューマネージャをインストールまたはアップグレードする場合には、追加のイメージミラーリングステップが必要になる場合があります。 詳細については、295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照してください。

変更点
対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7184453

IBM MQ Operator 3.4.1

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.1

オペレーター・チャネル
v3.4

.spec.version に許可される値
9.4.1.1-r1

マイグレーション中に許可される .spec.version の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.1.0-r1, 9.4.1.0-r2

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点
対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7182196

IBM MQ Operator 3.4.0

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.1

オペレーター・チャネル
v3.4

.spec.version に許可される値
9.4.1.0-r2

マイグレーション中に許可される .spec.version の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.1.0-r1

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点

- サポートが切れた多くの古いオペレーターが CASE から削除された。
- 対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7178065

IBM MQ Operator 3.3.0

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
v3.3

.spec.version に許可される値
9.4.1.0-r1

マイグレーション中に許可される .spec.version の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点

- IBM MQ Advanced ライセンスは、Development 使用では設定できなくなりました。
- キューマネージャーに正しいライセンスアノテーションが付けられるようになりました。 より詳細な情報、または以前のキュー・マネージャーを更新する手順については、'172 ページの『配備されたキュー・マネージャーのライセンス注釈の修正』 を参照のこと。
- **spec.license.metric** 値に関するルールが、ウェブフックによって強制されるようになりました。既存のキューマネージャは、9.4.1.0-r1 およびそれ以降のバージョンにアップグレードする場合、これらの規則に従う必要があります。
- IBM MQ Advanced for Non-Production Environment ライセンスのサポートを追加。
- IBM MQ Advanced ライセンスと '**Use** を Development に設定してデプロイする機能を削除。
- 大規模クラスタにおける'IBM MQ Operator のメモリフットプリントを削減する。
- OCP4.16 で実行した場合の'ServiceAccount 照合に関する問題を修正。
- 対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7174358

IBM MQ Operator 3.2.13

CP4I-SC2

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
v3.2-sc2

.spec.version に許可される値
9.4.0.11-r3

マイグレーション中に許可される .spec.version の値

9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3,

9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2, 9.4.0.11-r1, 9.4.0.11-r2

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

IBM MQ Operator 3.2.12

CP4I-SC2

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
`v3.2-sc2`

`.spec.version` に許可される値
9.4.0.11-r2

マイグレーション中に許可される `.spec.version` の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2, 9.4.0.11-r1

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点

- 対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7234178

IBM MQ Operator 3.2.11

CP4I-SC2

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
`v3.2-sc2`

`.spec.version` に許可される値
9.4.0.11-r1

マイグレーション中に許可される `.spec.version` の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1,

9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1, 9.4.0.10-r2

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点

- IBM MQ OpenID Connect (OIDC)のクライアント ID とクライアントシークレットが環境変数ではなくボリュームに保存されるようになりました。
- Kubernetes の警告を防ぐため、未使用の sa-<namespace> image pull secret を削除した。
- 対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7232272

IBM MQ Operator 3.2.10

CP4I-SC2

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
v3.2-sc2

.spec.version に許可される値
9.4.0.10-r2

マイグレーション中に許可される .spec.version の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1 ,9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.0.25-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1, 9.4.0.10-r1

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点

- 特定の条件下で、 IntegrationKeycloakClient リソースが不正なクライアント ID で更新される問題を修正しました。
- 対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7228945

IBM MQ Operator 3.2.9

CP4I-SC2

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
v3.2-sc2

.spec.version に許可される値
9.4.0.10-r1

マイグレーション中に許可される .spec.version の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2, 9.4.0.7-r1

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点
対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7184453

IBM MQ Operator 3.2.8

CP4I-SC2

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
v3.2-sc2

.spec.version に許可される値
9.4.0.7-r1

マイグレーション中に許可される .spec.version の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1, 9.4.0.6-r2

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点
対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7182196

IBM MQ Operator 3.2.7

CP4I-SC2

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
v3.2-sc2

.spec.version に許可される値
9.4.0.6-r2

マイグレーション中に許可される .spec.version の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1,

9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2, 9.4.0.6-r1

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点
対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7178065

IBM MQ Operator 3.2.6

CP4I-SC2

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
`v3.2-sc2`

`.spec.version` に許可される値
9.4.0.6-r1

マイグレーション中に許可される `.spec.version` の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.0.21-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1, 9.4.0.5-r2

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点
対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7174358

IBM MQ Operator 3.2.5

CP4I-SC2 CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
`v3.2-sc2`

`.spec.version` に許可される値
9.4.0.5-r2

マイグレーション中に許可される `.spec.version` の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.0.21-r2, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2,

9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3, 9.4.0.5-r1

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点
対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7171536

IBM MQ Operator 3.2.4

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
`v3.2-sc2`

`.spec.version` に許可される値
9.4.0.5-r1

マイグレーション中に許可される `.spec.version` の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.0.21-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2, 9.4.0.0-r3

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点
対処された脆弱性については、このセキュリティ・ブリーチに詳述されている：https://www.ibm.com/support/pages/node/7167732

IBM MQ Operator 3.2.3

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
`v3.2-sc2`

`.spec.version` に許可される値
9.4.0.0-r3

マイグレーション中に許可される `.spec.version` の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.0.20-r2, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1, 9.4.0.0-r2

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点
対処された脆弱性は、これらのセキュリティ・ブレットに詳述されている：

- https://www.ibm.com/support/pages/node/7162095
- https://www.ibm.com/support/pages/node/7162272

IBM MQ Operator 3.2.2

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
`v3.2-sc2`

`.spec.version` に許可される値
9.4.0.0-r2

マイグレーション中に許可される `.spec.version` の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3, 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.0.20-r1, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2, 9.4.0.0-r1

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点

- 対処された脆弱性の詳細については、このセキュリティー情報を参照してください。

IBM MQ Operator 3.2.1

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
`v3.2-sc2`

`.spec.version` に許可される値
9.4.0.0-r1

マイグレーション中に許可される `.spec.version` の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3,9.3.3.2-r3 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.2-r3, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

変更点

- 問題を解決する OpenShift Container Platform4.12 アップグレードする場合 v3.2-sc2 チャネルが予期しない動作を引き起こす可能性があります IBM Cloud Pak for Integration ユーザー。 詳細については、アップグレード 2023.4 の中に IBM Cloud Pak for Integration ドキュメンテーション。

IBM MQ Operator 3.2.0

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 16.1.0

オペレーター・チャネル
`v3.2-sc2`

`.spec.version` に許可される値
9.4.0.0-r1

マイグレーション中に許可される `.spec.version` の値
9.3.0.0-r1, 9.3.0.0-r2, 9.3.0.0-r3,9.3.3.2-r3 9.3.0.1-r1, 9.3.0.1-r2, 9.3.0.1-r3, 9.3.0.1-r4, 9.3.0.3-r1, 9.3.0.4-r1, 9.3.0.4-r2, 9.3.0.5-r1, 9.3.0.5-r2, 9.3.0.5-r3, 9.3.0.6-r1, 9.3.0.10-r1, 9.3.0.10-r2, 9.3.0.11-r1, 9.3.0.11-r2, 9.3.0.15-r1, 9.3.0.16-r1, 9.3.0.16-r2, 9.3.0.17-r1, 9.3.0.17-r2, 9.3.0.17-r3, 9.3.1.0-r1, 9.3.1.0-r2, 9.3.1.0-r3, 9.3.1.1-r1, 9.3.2.0-r1, 9.3.2.0-r2, 9.3.2.1-r1, 9.3.2.1-r2, 9.3.3.0-r1, 9.3.3.0-r2, 9.3.3.1-r1, 9.3.3.1-r2, 9.3.3.2-r1, 9.3.3.2-r2, 9.3.3.2-r3, 9.3.3.3-r1, 9.3.3.3-r2, 9.3.4.0-r1, 9.3.4.1-r1, 9.3.5.0-r1, 9.3.5.0-r2, 9.3.5.1-r1, 9.3.5.1-r2

Red Hat OpenShift Container Platform のバージョン
OpenShift Container Platform 4.12 以上。

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services バージョン 4.6 のみ。

新機能

- 168 ページの『永続ボリュームの拡張』 がサポートされるようになりました。
- `mq.ibm.com/stop` アノテーションを追加して `true` に設定することで、キュー・マネージャーを停止できるようになりました。 229 ページの『キュー・マネージャーの停止 (mq.ibm.com/stop)』 を参照してください。

  注:

  - 停止したキュー・マネージャーの `StatefulSet` 内の *.replicas* フィールドは、0 に設定されています。
  - IBM MQ Operator は現在、 `StatefulSet` 内の *.replicas* フィールドをアクティブに管理しているため、このフィールドを変更すると、オペレーターによって即時に元に戻されます。
  - 古いバージョンの IBM MQ では、 *.replicas* フィールドを変更した場合、「Failed」状態になりますが、変更した値は保持されます。 既存の操作手順がこの動作に依存している場合は、 IBM MQ 9.4 以降、 `mq.ibm.com/stop` アノテーションを使用する必要があります。

変更点

- Red Hat OpenShift Container Platform の奇数番号のリリースがサポートされるようになりました。
- IBM MQ カタログ・イメージが SQLite データベース・フォーマットからファイル・ベースのカタログ・フォーマットに移動されました。
- Red Hat ユニバーサル・ベース・イメージ 9.4-949.1716471857 に基づく。 **注:** UBI 9 は、 FIPS 140-3 認証待ちです。 UBI 9 は Power® 8 アーキテクチャではサポートされていません。
- 対処された脆弱性の詳細については、このセキュリティー情報を参照してください。

OpenShift CP4I キュー・マネージャ・コンテナ・イメージのリリース履歴

注 : 以前のキューマネージャコンテナイメージについては、'IBM MQ 9.3 ドキュメントの ' IBM MQ Operator のリリース履歴を参照してください。

画像はスタンドアローンでも、'IBM MQ Operator サポートされる。

9.4.3.0-r1

CD

対応オペレーターバージョン
スタンドアロン、または IBM MQ Operator 以上でサポート。 3.6.0 またはそれ以上。

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.3.0-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.3.0-r1` (IBM MQ Advanced)

  注 : また、適切なライセンス条項を遵守し、キューマネージャに注釈を付ければ、このイメージを使用して IBM MQ ライセンスのキューマネージャを配備することもできます。 詳しくは、161 ページの『を使用して、 IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』 および 281 ページの『IBM MQ コンテナライセンス注釈』 を参照してください。
- `icr.io/ibm-messaging/mq:9.4.3.0-r1` (IBM MQ Advanced for Developers)

新機能

- Prometheus メトリクス・エンドポイントへのアクセスの監査ログを有効にできるようになりました。 詳細については、150 ページの『コンテナ内の IBM MQ に対する監査ロギングの設定』を参照してください。
- IBM MQ ライセンスアノテーションを使用して、IBM MQ Operator でデプロイされたキューマネージャを構成できるようになりました。 詳しくは、161 ページの『を使用して、 IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。 このアップデートは、すべての IBM MQ 9.4 キュー・マネージャに適用されます
- 新しい IBM MQ Native HA および Cross-Region Replication アドオンライセンスを追加することで、IBM MQ ライセンスのキューマネージャで Native HA および Cross-Region Replication 機能を使用できるようになりました。 詳細については、 116 ページの『IBM MQ 、キューマネージャのネイティブ HA とクロスリージョンレプリケーションを設定するには、次のようにします。 IBM MQ Operator』 または 186 ページの『IBM MQ のキューマネージャでネイティブ HA とクロスリージョンレプリケーションを設定する。 Kubernetes』 を参照してください。 このアップデートは、 IBM MQ 9.4.2 キュー・マネージャーにも適用される。

変更点

- 何が変わったのか IBM MQ 9.4.3。
- 対処された脆弱性については、このセキュリティー情報に詳述されている： https://www.ibm.com/support/pages/node/7236608

9.4.2.1-r2

CD

対応オペレーターバージョン
スタンドアロン、または IBM MQ Operator 以上でサポート。 3.5.3 またはそれ以上。

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.2.1-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.2.1-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.2.1-r2` (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.2」の新機能 - マルチプラットフォーム - ベースとアドバンスド・エンタイトルメント。

変更点

- 何が変わったのか IBM MQ 9.4.2。
- 対処された脆弱性については、このセキュリティー情報に詳述されている： https://www.ibm.com/support/pages/node/7234178

9.4.2.1-r1

対応オペレーターバージョン
スタンドアロン、または IBM MQ Operator 以上でサポート。 3.5.2 またはそれ以上。

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.2.1-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.2.1-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.2.1-r1` (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.2」の新機能 - マルチプラットフォーム - ベースとアドバンスド・エンタイトルメント。

変更点

- 何が変わったのか IBM MQ 9.4.2。
- 対処された脆弱性については、このセキュリティー情報に詳述されている： https://www.ibm.com/support/pages/node/7232272

9.4.2.0-r2

対応オペレーターバージョン
スタンドアロン、または IBM MQ Operator 以上でサポート。 3.5.1 またはそれ以上。

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.2.0-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.2.0-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.2.0-r2` (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.2」の新機能 - マルチプラットフォーム - ベースとアドバンスド・エンタイトルメント。

変更点

- 何が変わったのか IBM MQ 9.4.2。
- 対処された脆弱性については、このセキュリティー情報に詳述されている： https://www.ibm.com/support/pages/node/7228945

9.4.2.0-r1

CD

対応オペレーターバージョン
スタンドアロン版、または IBM MQ Operator 3.5.0 またはそれ以上。

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.2.0-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.2.0-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.2.0-r1` (IBM MQ Advanced for Developers)

新機能

- キューマネージャーのメトリクスは、 `/etc/mqm/metrics/pki/keys` に TLS キーを登録することで、 HTTPS から取得できます。 詳しくは、289 ページの『ファイルシステムとマウントポイント』を参照してください。
- IBM MQ 9.4.2」の新機能 - マルチプラットフォーム - ベースとアドバンスド・エンタイトルメント。

変更点

- 何が変わったのか IBM MQ 9.4.2。
- 対処された脆弱性については、このセキュリティー情報に詳述されている： https://www.ibm.com/support/pages/node/7184453

9.4.1.1-r1

CD

対応オペレーターバージョン
スタンドアロンでサポート、または IBM MQ Operator 3.4.1 またはそれ以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.1.1-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.1.1-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.1.1-r1` (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.1」の新機能 - マルチプラットフォーム - ベースとアドバンスド・エンタイトルメント。

変更点

- 何が変わったのか IBM MQ 9.4.1。
- 対処された脆弱性については、このセキュリティー情報に詳述されている： https://www.ibm.com/support/pages/node/7182196

9.4.1.0-r2

CD

対応オペレーターバージョン
スタンドアロン、または「IBM MQ Operator3.4.0 以上でサポート

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.1.0-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.1.0-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.1.0-r2` (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.1」の新機能 - マルチプラットフォーム - ベースとアドバンスド・エンタイトルメント。

変更点

- 何が変わったのか IBM MQ 9.4.1。
- 対処された脆弱性については、このセキュリティー情報に詳述されている： https://www.ibm.com/support/pages/node/7178065

9.4.1.0-r1

CD

対応オペレーターバージョン
スタンドアロン、または「IBM MQ Operator3.3.0 以上でサポート

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.1.0-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.1.0-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.1.0-r1` (IBM MQ Advanced for Developers)

新機能

- IBM MQ Advanced for Non-Production Environment ライセンスが 9.4.1.0-r1 以降のキューマネージャ用に設定可能になりました。 ライセンス情報については、'249 ページの『IBM MQ Operator のライセンスと API リファレンス』 を参照のこと。
- IBM MQ 9.4.1 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- APARIT46430 の修正。
- 中間証明書の処理に関する問題を解決するための修正。
- FIPS が有効になっていない場合、 **runmqakm** キーストア生成の FIPS モードを無効にするよう修正。
- ファイルパスをサニタイズするように修正。
- 画像のサイズを小さくするための更新。
- IBM MQ キューマネージャコンテナイメージ 9.4.1.0-r1 には、'IBM MQ トレースユーザエグジットのバージョン 3.1.112024.3.0 が含まれています。 IBM Instana®ドキュメントの'IBM MQ トレースを参照のこと。
- ネイティブ HA 設定をテンプレートから生成するのではなく、外部から提供できるようにする。

  - ネイティブ HA の環境変数の設定を廃止（ただし、'MQ_NATIVE_HA=true は必須）。
  - 現在では非推奨となっている環境変数の設定について、'IBM Documentation 動作を明確にした。
- 何が変わったのか IBM MQ 9.4.1。
- 対処された脆弱性については、このセキュリティー情報に詳述されている： https://www.ibm.com/support/pages/node/7174358

IBM MQ Operator 使用可能な画像

9.4.0.11-r3

CP4I-SC2

必要なオペレーター・バージョン
10 ページの『IBM MQ Operator 3.2.13』 またはそれ以降

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.11-r3 (IBM Cloud Pak for Integration)
- cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.11-r3 (IBM MQ Advanced)

  注：また、適切なライセンス条項を遵守し、キューマネージャに注釈を付ければ、このイメージを使用して IBM MQ ライセンスのキューマネージャを配備することもできます。 詳しくは、161 ページの『を使用して、 IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』 および 281 ページの『IBM MQ コンテナライセンス注釈』 を参照してください。
- icr.io/ibm-messaging/mq:9.4.0.11-r3 (IBM MQ Advanced for Developers)

新機能

- IBM MQ ライセンスアノテーションを使用して、IBM MQ Operator でデプロイされたキューマネージャを構成できるようになりました。 詳しくは、161 ページの『を使用して、 IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。 このアップデートは、すべての IBM MQ 9.4 キュー・マネージャに適用されます

変更点

- 何が変わったのか IBM MQ 9.4.0。
- このセキュリティ情報では、対処された脆弱性について詳しく説明されています。 https://www.ibm.com/support/pages/node/7236608

9.4.0.11-r2

CP4I-SC2

必要なオペレーター・バージョン
11 ページの『IBM MQ Operator 3.2.12』 以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.11-r2 (IBM Cloud Pak for Integration)
- cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.11-r2 (IBM MQ Advanced)
- icr.io/ibm-messaging/mq:9.4.0.11-r2 (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.0 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- 何が変わったのか IBM MQ 9.4.0。
- このセキュリティ情報では、対処された脆弱性について詳しく説明されています。 https://www.ibm.com/support/pages/node/7234178

9.4.0.11-r1

CP4I-SC2

必要なオペレーター・バージョン
3.2.11 以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.11-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.11-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.11-r1` (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.0 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- 何が変わったのか IBM MQ 9.4.0。

9.4.0.10-r2

CP4I-SC2

必要なオペレーター・バージョン
3.2.10 以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.10-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.10-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.10-r2` (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.0 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- 何が変わったのか IBM MQ 9.4.0。

9.4.0.10-r1

CP4I-SC2

必要なオペレーター・バージョン
3.2.9 以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.10-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.10-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.10-r1` (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.0 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- 何が変わったのか IBM MQ 9.4.0。

9.4.0.7-r1

CP4I-SC2

必要なオペレーター・バージョン
3.2.8 以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.7-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.7-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.7-r1` (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.0 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- 何が変わったのか IBM MQ 9.4.0。

9.4.0.6-r2

CP4I-SC2

必要なオペレーター・バージョン
3.3.2.7 以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.6-r2` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.6-r2` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.6-r2` (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.0 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- 何が変わったのか IBM MQ 9.4.0。

9.4.0.6-r1

CP4I-SC2

必要なオペレーター・バージョン
3.3.2.6 以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.6-r1 (IBM Cloud Pak for Integration)
- cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.6-r1 (IBM MQ Advanced)
- icr.io/ibm-messaging/mq:9.4.0.6-r1 (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.0 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- 何が変わったのか IBM MQ 9.4.0。
- Mac 上で Rosetta を使用して「**endmqm** コマンドを実行するとコンテナがハングする問題を修正しました。

9.4.0.5-r2

CP4I-SC2 CD

必要なオペレーター・バージョン
3.2.5 またはそれ以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.5-r2 (IBM Cloud Pak for Integration)
- cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.5-r2 (IBM MQ Advanced)
- icr.io/ibm-messaging/mq:9.4.0.5-r2 (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.0 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- 何が変わったのか IBM MQ 9.4.0。
- Red Hat Universal Base Image 9.4-1227.1725849298 に基づいています。

9.4.0.5-r1

CP4I-SC2 CD

必要なオペレーター・バージョン
3.2.4 またはそれ以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.5-r1 (IBM Cloud Pak for Integration)

- cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.5-r1 (IBM MQ Advanced)
- icr.io/ibm-messaging/mq:9.4.0.5-r1 (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.0 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- 何が変わったのか IBM MQ 9.4.0。
- Red Hat Universal Base Image 9.4-1194 に基づいています。

9.4.0.0-r3

CP4I-SC2 CD

必要なオペレーター・バージョン
3.2.3 またはそれ以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.0-r3 (IBM Cloud Pak for Integration)
- cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.0-r3 (IBM MQ Advanced)
- icr.io/ibm-messaging/mq:9.4.0.0-r3 (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.0 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- 何が変わったのか IBM MQ 9.4.0。
- に基づく Red Hat ユニバーサルベースイメージ 9.4-1134 。
- OpenShift Container Platform (OCP) 4.16 でのみ見られる、和解する ServiceAccount が画像のプルシークレットの不定のループを作る問題を解決しました。 OCP 4.16 は、OCP が .metadata.annotation を ServiceAccount に注入するという動作の変更を導入しました。

9.4.0.0-r2

CP4I-SC2 CD

必要なオペレーター・バージョン
3.2.2 以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.0-r2 (IBM Cloud Pak for Integration)
- cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.0-r2 (IBM MQ Advanced)
- icr.io/ibm-messaging/mq:9.4.0.0-r2 (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.0 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- 何が変わったのか IBM MQ 9.4.0。

- に基づく Red Hat ユニバーサルベースイメージ 9.4-1134 。

9.4.0.0-r1

CP4I-SC2 CD

必要なオペレーター・バージョン
3.2.0 以上

サポートされているアーキテクチャー
amd64, s390x, ppc64le

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.0.0-r1` (IBM Cloud Pak for Integration)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.4.0.0-r1` (IBM MQ Advanced)
- `icr.io/ibm-messaging/mq:9.4.0.0-r1` (IBM MQ Advanced for Developers)

新機能

- IBM MQ 9.4.0 for Multiplatforms の新機能-基本ライセンスと Advanced ライセンス

変更点

- IBM MQ 9.4.0 の変更内容
- Deprecated IBM MQ Advanced for Developers を使用する場合、環境変数を使用して `admin` ユーザーおよび app ユーザーのパスワードを設定することは推奨されません。 代わりにシークレットを使用してください。
- 環境変数 *MQ_LOGGING_CONSOLE_SOURCE* に新しいオプション値 `mqsc` が追加されました。 このオプションは、コンテナー・ログ内の `autocfgmqsc.LOG` の内容を反映するために使用できます。
- Red Hat ユニバーサル・ベース・イメージ 9.4-949.1716471857 に基づく。 **注：** UBI 9 は、 FIPS 140-3 認証待ちです。 UBI 9 は Power 8 アーキテクチャではサポートされていません。

コンテナー内の IBM MQ の計画

コンテナー内の IBM MQ の計画を立てるときには、高可用性の実現方法、キュー・マネージャーの保護方法など、さまざまなアーキテクチャー・オプションのために IBM MQ が提供しているサポートについて考慮してください。

このタスクについて

コンテナー・アーキテクチャーで IBM MQ を計画する前に、基本的な IBM MQ 概念 (IBM MQ 技術の概要を参照) 、および基本的な Kubernetes/Red Hat OpenShift の概念を理解しておく必要があります (OpenShift Container Platform のアーキテクチャーを参照してください) 。

手順

- 28 ページの『コンテナー内の IBM MQ の使用方法の選択』.
- 28 ページの『コンテナーでの IBM MQ のサポート』.
- 38 ページの『IBM MQ Operator のストレージの計画』.
- 40 ページの『コンテナー内の IBM MQ の高可用性の計画』.
- 46 ページの『コンテナー内の IBM MQ の災害復旧』.
- 49 ページの『コンテナー内の IBM MQ でのユーザーの認証と許可』.

コンテナー内の IBM MQ の使用方法の選択

コンテナー内の IBM MQ の使用方法としては、いくつもの選択肢があります。プリパッケージされているコンテナー・イメージを使用する IBM MQ Operator を使用することも、独自のイメージとデプロイメント・コードをビルドすることもできます。

IBM MQ Operator の使用

Red Hat OpenShift Container Platform にデプロイする場合は、おそらく、IBM MQ Operator を使用したいはずです。

IBM MQ Operator は、 Red Hat OpenShift Container Platform API を拡張して、新しい `QueueManager` カスタム・リソースを追加します。 オペレーターは、新しいキュー・マネージャー定義を監視し、それを `StatefulSet` リソースや `Service` リソースなどの必要な下位リソースに変換します。 Native HA の場合、オペレーターはキュー・マネージャー・インスタンスの複雑なローリング更新を実行することもできます。 以下を参照してください。 44 ページの『ネイティブ HA キュー・マネージャーの独自ローリング更新を実行する場合の考慮事項』

IBM MQ の機能の中には、IBM MQ Operator を使用する場合にはサポートされないものもあります。 29 ページの『IBM MQ Operator のサポート』 を使ったときに何がサポートされるかについては、IBM MQ Operator を参照してください。

組み込み済みの IBM MQ Advanced 画像を使う

IBM Container Registry から利用できる組み込み済みの画像を使うことができます。 詳細は 30 ページの『IBM MQ Advanced コンテナ画像のサポート』 を参照してください。

IBM MQ

From IBM MQ 9.4.1, you can use the prebuilt images that are available from the IBM Container Registry. 詳しくは、31 ページの『IBM MQ ライセンスのコンテナ・イメージのサポート』を参照してください。

独自のイメージおよびデプロイメント・コードのビルド

これは最も柔軟なコンテナソリューションですが、コンテナの設定に強いスキルが必要であり、結果としてのコンテナを"所有する"必要があります。

独自のイメージをビルドするためのサンプルが用意されています。 87 ページの『独自のコンテナー・イメージのビルドによる IBM MQ の準備』 を参照してください。

独自のイメージとデプロイメントコードを構築するときにサポートされるものの詳細については、33 ページの『独自の IBM MQ コンテナ・イメージの構築のサポート』を参照してください。

関連資料

28 ページの『コンテナーでの IBM MQ のサポート』
コンテナーでは、すべての IBM MQ 機能が同じ方法で使用可能およびサポートされているわけではありません。

コンテナーでの IBM MQ のサポート

コンテナーでは、すべての IBM MQ 機能が同じ方法で使用可能およびサポートされているわけではありません。

関連概念

IBM MQ FAQ for Long Term Support and Continuous Delivery リリース

関連資料

IBM Cloud Pak for Integration ソフトウェア・サポート・ライフサイクルの補遺

関連情報
SELinux および AppArmor に対する IBM MQ のサポート

IBM MQ Operator のサポート

IBM MQ Operator は IBM MQ Advanced と IBM Cloud Pak for Integration の一部として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

IBM MQ Operator は Red Hat OpenShift Container Platform にのみ対応しています。 Red Hat OpenShift Container Platform のリリースは、Red Hat がサポートを終了すると、IBM MQ ではサポートされなくなります。 詳細は 35 ページの『IBM MQ Operator のバージョン・サポート』 および Red Hat OpenShift Container Platform のライフサイクル日付 を参照してください

対応 CPU アーキテクチャ

IBM MQ Operator は amd64 と s390x z/Linux で、ppc64le Power Systems version 9 以上でサポートされています。 Red Hat OpenShift Container Platform クラスタ内のすべてのノードが同じ CPU アーキテクチャを使用しなければならないことに注意してください。

サポート期間

IBM は、IBM MQ Operator と関連するキューマネージャイメージの不具合と使用法のサポートを提供します。 サポート期間は、使用するライセンスによって異なります：

- **IBM Cloud Pak for Integration** ライセンスを使用している場合、IBM MQ Operator は対応する IBM Cloud Pak for Integration バージョンと同じ期間サポートされます。 いくつかのバージョンは IBM Cloud Pak for Integration - Support Cycle 2 (formerly Long Term Support) であり、他のバージョンは Continuous Delivery (CD) です。 CD オペレータとキューマネージャは、次の IBM Cloud Pak for Integration CD または CP4I-LTS リリースまでサポートされます CP4I-LTS オペレータと IBM Cloud Pak for Integration ライセンスでデプロイされたキューマネージャは、次の CP4I-LTS リリースまでサポートされ、さらにアップグレードのための猶予期間があります。

  例えば、IBM Cloud Pak for Integration ライセンス “L-JTPV-KYG8TF” を使用してキューマネージャを作成した場合、これは CP4I-LTS リリースと関連付けられます。 この場合、IBM MQ Advanced 9.4.0 がデプロイされ、その IBM Cloud Pak for Integration バージョンに合わせてサポートされます (2 年間)。
- または **IBM MQ Advanced** または **IBM MQ** ライセンスを使用する場合、 IBM MQ Continuous Delivery ストリームのみがサポートされます。 IBM MQ Advanced または IBM MQ のライセンスで展開される各 IBM MQ Operator およびキュー・マネージャーは、そのキュー・マネージャー・バージョンをサポートする最初のオペレーターから 1 年間サポートされます。 リリースは、次の Continuous Delivery リリースまで、積極的にセキュリティフィックスを受ける。 例えば、 IBM MQ Advanced のライセンス “L-EHXT-MQCRN9” を使用してキューマネージャを作成した場合、 Continuous Delivery のリリースとして IBM MQ Advanced 9.4.0 を使用することになり、1 年間サポートされ、 IBM MQ 9.4.1 のリリースまでセキュリティフィックスが公開されます。 バージョン 9.4.1 のリリース後、ご要望があれば IBM は、以前の CD リリースレベルでの修正を提供する合理的な努力の実現可能性を評価します

詳細は、IBM MQ 長期サポートおよび継続的デリバリ リリースに関する FAQ を参照してください。

可用性を修正する

キューマネージャの定期的および中間的な修正は IBM Container Registry から入手可能です。 IBM MQ Operator の中間修正は利用できません。

利用できない機能

IBM MQ Advanced の以下の機能は、IBM MQ Operator で運用されるキューマネージャでは利用できません：

- 複製データ キュー マネージャー (RDQM) は使用できません。この機能は Linux カーネルと緊密に結合しているため、コンテナーではサポートされません
- キュー・マネージャのリニア・ロギングは利用できません
- **crtmqdir**、**crtmqm**、**strmqm**、**endmqm** のカスタムコマンドラインオプションの指定は利用できません - ほとんどのオプションは INI ファイルを使って設定できますが、リニアロギングの使用など、設定できないものもあります。
- オペレーティング・システム（OS）ユーザーは利用できない
- AMQP がインストールされていない
- MQTT がインストールされていない
- マネージドファイル転送エージェントは使用できません。 しかし、IBM MQ Operator を使用して、1 つ以上の Coordination、Command、Agent キューマネージャを提供することができます。
- 別の鍵ストアを使用した MCA 傍受による高度なメッセージ・セキュリティは利用できない。 MCA の傍受は、メイン・キュー・マネージャのキー・ストアを使用するように設定することができる。

IBM が MQ Operator とともに提供するキュー・マネージャ・コンテナ・イメージのみを使用することができます。 例外は、IBM が提供するイメージにコンテナ・イメージ層を追加して、設定ファイルや終了を追加する場合です。

関連概念

IBM MQ FAQ for Long Term Support and Continuous Delivery リリース

関連資料

IBM Cloud Pak for Integration ソフトウェア・サポート・ライフサイクル補遺

関連情報

コンテナ内の IBM MQ のライフサイクル日付

MQ Adv. Linux CD V 9.4.1 IBM MQ Advanced コンテナ画像のサポート

IBM MQ Advanced バージョン 9.4.1 以降では、IBM MQ Advanced コンテナイメージを IBM MQ Operator なしで使うことができます

ビルド済みの IBM MQ Advanced コンテナイメージは、いくつかの論理層で構成されています：

- IBM MQ コンテナ有効化コード（`runmqserver`、`chkmqhealthy`、`chkmqready`、`chkmqstarted`）
- IBM MQ Advanced それ自体
- Red Hat のパッケージは Red Hat Universal Base Image から

サポートされているプラットフォーム

ビルド済みの IBM MQ Advanced イメージは、containerd, cri-o, Podman あるいは Docker が提供するコンテナ実行環境にデプロイできます。 詳しくはシステム要件 IBM MQ をご覧ください。 IBM、IBM MQ コンテナ有効化コードと「IBM MQ Advanced 自体の不具合と使用法のサポートを提供するが、コンテナ環境やデプロイメント・パイプラインに関する問題の設定やデバッグを支援するものではない。

重要 : Red Hat は、'Red Hat Enterprise Linux®または 'Red Hat Enterprise Linux CoreOS ホスト上で動作するコンテナに対してのみ、'Red Hat パッケージのサポートを提供する。 詳細は Red HatContainer Support Policy を参照してください。

Licensing

IBM License Service で Kubernetes を使用する場合、構築済みの IBM MQ Advanced イメージはコンテナーライセンスの対象となります。 Kubernetes オーケストレーションを採用していない場合 (たとえば、Podman や Docker を使用している場合)、この最適化されたライセンス モデルを利用することはできません。 詳しくは、コンテナ ライセンスに関する FAQ をご覧ください。

対応 CPU アーキテクチャ

ビルド済みの IBM MQ Advanced イメージは amd64 と s390x z/Linux で、ppc64le Power Systems バージョン 9 以上でサポートされています。 同じキューマネージャの一部であるすべてのコンテナ（例えば、Native HA レプリカ）は、同じ CPU アーキテクチャを使用しなければならないことに注意。

サポート期間

IBM MQ Advanced Continuous Delivery ストリームのみがサポートされています 各キュー・マネージャーのバージョンは、通常のサポート期間（1 年間）に従ってサポートされます。

リリースは、次の Continuous Delivery リリースまで、積極的にセキュリティ修正が行われます。 詳細は、IBM MQ 長期サポートおよび継続的デリバリ リリースに関する FAQ を参照してください。

可用性を修正する

定期的および中間的な修正は IBM Container Registry から入手可能です。

利用できない機能

IBM MQ Advanced の以下の機能は、ビルド済みコンテナ・イメージでは利用できません：

- 複製データ キュー マネージャー (RDQM) は使用できません。この機能は Linux カーネルと緊密に結合しているため、コンテナーではサポートされません
- キュー・マネージャのリニア・ロギングは利用できません
- **crtmqdir**、**crtmqm**、**strmqm**、**endmqm** のカスタムコマンドラインオプションの指定は利用できません - ほとんどのオプションは INI ファイルを使って設定できますが、リニアロギングの使用など、設定できないものもあります。
- オペレーティング・システム（OS）ユーザーは利用できない
- AMQP がインストールされていない
- MQTT がインストールされていない
- マネージドファイル転送エージェントは使用できません。 しかし、構築済みのコンテナ・イメージを使用して、1 つまたは複数の Coordination、Command、Agent キュー・マネージャを提供することができる。

これらの機能を使いたい場合は、独自のコンテナ・イメージを構築することができる。

関連概念

IBM MQ FAQ for Long Term Support and Continuous Delivery リリース

Red Hat コンテナ・サポート・ポリシー

関連情報

コンテナ内の IBM MQ のライフサイクル日付

V 9.4.1 IBM MQ ライセンスのコンテナ・イメージのサポート

IBM MQ 9.4.1 以降は、IBM MQ ライセンス付きコンテナ・イメージを IBM MQ Operator なしで使用できます。

事前に構築された IBM MQ コンテナ・イメージは、いくつかの論理レイヤーで構成されている：

- IBM MQ コンテナ・イネーブルメント・コード (,, and)`runmqserver` `chkmqhealthy` `chkmqready` `chkmqstarted`
- IBM MQ それ自体
- Red Hat パッケージの Red Hat Universal Base Image

注 : IBM MQ ライセンスのコンテナ・イメージは IBM MQ Advanced イメージを使用しますが、 IBM MQ Advanced 固有の機能は適切なライセンスがなければ使用できません。 詳しくは、このトピックで後述する 32 ページの『から利用できない機能 IBM MQ Advanced』 のセクションを参照のこと。

サポートされているプラットフォーム

ビルド済みの IBM MQ イメージは、containerd、cri-o、Podman または Docker によって提供されるコンテナ実行環境にデプロイできる。 詳細については、 IBM MQ のシステム要件を参照してください。 IBM は、-supplied コンテナ有効化コードと 自体の不具合と使用法のサポートを提供しますが、コンテナ環境やデプロイメントパイプラインに関する問題の設定やデバッグを支援するものではありません。 IBM IBM MQ IBM MQ

重要 : Red Hat は、または のホスト上で動作するコンテナに対してのみ、パッケージをサポートする。 Red Hat Enterprise Linux Red Hat Enterprise Linux CoreOS Red Hat 詳細は Red Hat コンテナ・サポート・ポリシーを参照。

Licensing

ビルド済みの IBM MQ イメージは、 Kubernetes 上で IBM License Service とともに使用される場合、コンテナ・ライセンスの対象となります。 Kubernetes オーケストレーションを採用していない場合 (たとえば、Podman または Docker を使用している場合)、この最適化されたライセンスモデルを利用することはできません。 詳細はコンテナライセンスに関する FAQ をご覧ください。

対応 CPU アーキテクチャ

ビルド済みの IBM MQ イメージは、 amd64 と s390x z/ Linux、および ppc64le Power Systems バージョン 9 以上でサポートされています。 同じキューマネージャの一部であるすべてのコンテナ（たとえば Native HA レプリカ）は、同じ CPU アーキテクチャを使用しなければならないことに注意。

サポート期間

IBM MQ 継続的デリバリーのストリームのみがサポートされています。 各キュー・マネージャーのバージョンは、通常のサポート期間（1 年間）に従ってサポートされます。 リリースは、次の Continuous Delivery リリースまで、積極的にセキュリティフィックスを受ける。 詳しくは、 IBM MQ 長期サポートおよび継続的デリバリーリリースに関する FAQ をご覧ください。

可用性を修正する

定期的および暫定的な修正は、 IBM Container Registry から入手可能です。

から利用できない機能 IBM MQ Advanced

IBM MQ Advanced IBM MQ の以下の機能は、 のライセンス付きコンテナ・イメージでは使用できません：

- ネイティブ HA とクロスリージョンレプリケーション。 IBM MQ ライセンスのキュー・マネージャでネイティブ HA およびクロスリージョンレプリケーション機能を使用するには、116 ページの『IBM MQ 、キューマネージャのネイティブ HA とクロスリージョンレプリケーションを設定するには、次のようにします。 IBM MQ Operator』 を参照してください。
- AMS
- Managed File Transfer OpenSSH タイプの SFTP サーバー用秘密鍵のサポート
- ネイティブ HA キューマネージャの損傷オブジェクトの自動回復
- メディア・イメージのスケジューリング - 複製ロギング
- RDQM は Red Hat Enterprise Linux 9
- Kafka 接続

これらの機能を使いたい場合は、 IBM MQ Advanced。 詳しくは、 IBM MQ Advanced コンテナ・イメージのサポートを参照。

Linux 独自の IBM MQ コンテナ・イメージの構築のサポート

IBM MQ コンテナ・サンプルは、コンテナ内で 'IBM MQ 有効にするための、 サポートされていないラッパー・コードのサンプルである。 これは最も柔軟なコンテナソリューションですが、コンテナの設定に強いスキルが必要であり、結果としてのコンテナを"所有する"必要があります。

ダウンロード

IBM MQ コンテナのサンプルはこちら： https://github.com/ibm-messaging/mq-container.

注 : コンテナのサンプルと、あなたが追加または修正したコードは、'IBM ではサポートされていません。

コンテナ・サンプルで使用するには、「IBM MQ 非インストール・イメージをダウンロードする。

- IBM MQ Advanced for Developers ライセンスをお持ちの場合は、'Continuous Delivery(CD) バージョンの 'IBM MQ をお使いください。 これらの非インストール・イメージは、DownloadingIBM MQ 9.4CD から入手できます。
- プロダクション・ライセンスをお持ちの場合は、'Long Term Support(LTS) または 'CD バージョンの 'IBM MQ を使用してください。 これらの非インストールイメージは、Fix Central から入手できます。

サポートされているプラットフォーム

サポートされているコンテナ実行環境と Linux ディストリビューションの詳細については、System Requirements for IBM MQ を参照してください。 IBM MQ は、containerd、cri-o、Podman、または 'Docker によって提供されるコンテナ実行環境で使用することができる。 IBM、「IBM MQ コアソフトウェアの不具合や使用方法のサポートを提供するが、コンテナ環境やデプロイメントパイプラインの設定や問題のデバッグは手助けしない。

コンテナのサンプルでは、Red Hat Universal Base Image をベース画像として使っています。 Red Hat は、Red Hat Enterprise Linux または Red Hat Enterprise Linux CoreOS ホスト上で動作するコンテナに対してのみ Red Hat パッケージのサポートを提供します。 詳細は Red HatContainer Support Policy を参照してください。 例えば、'IBM MQ がサポートする'Linux ディストリビューションにマッチする任意の'Linux ベースイメージ上に、独自のコンテナイメージを構築することができます：

- Red Hat Universal Base Image
- Ubuntu
- SUSE Linux Enterprise Server ベースコンテナ画像

対応 CPU アーキテクチャ

IBM MQ のシステム要件を参照してください。

IBM MQ サポート期間

IBM MQ 非インストールイメージの製品版に対するサポートが提供されます。 LTS 場合、サポート期間は一般提供開始から 5 年間。 CD、各インクリメンタルリリースは一般提供から 1 年間サポートされる。 詳細については、長期サポートおよび継続的デリバリー・リリース IBM MQFAQ に関する を参照してください。

IBM MQ 可用性を修正

製品版の「IBM MQ 非インストールイメージについては、サポート期間中、修正が提供されます。 修正を要求することもできる。 CD アップデートが不具合修正の対象外となった場合は、対象となるバージョンにアップデートし、不具合が再現できる場合は修正を提供します。 利用可能な「LTS」および「CD リリースと最新の修正については、バージョン別 IBM MQ 修正を/span>のを参照してください。

利用できない機能

オペレーティングシステム（OS）ユーザーは、'IBM MQ ノンインストールイメージを使用することはできません。 OS ユーザーを使いたい場合は、RPM インストーラーを使って独自のイメージを構築する必要があるが、これにはサンプルがない。

サポートされるコンポーネントの詳細については、「IBM MQ」のシステム要件を参照してください。 サンプルはいくつか用意されているが、すべてのコンポーネントについて用意されているわけではない。

関連概念

IBM MQ FAQ for Long Term Support and Continuous Delivery リリース

関連タスク

87 ページの『独自のコンテナー・イメージのビルドによる IBM MQ の準備』

自作コンテナーを開発します。 これは最も柔軟なコンテナソリューションですが、コンテナの設定に強いスキルが必要であり、結果としてのコンテナを"所有する"必要があります。

MQ Adv. Linux IBM MQ Advanced for Developers コンテナイメージの有無

IBM MQ Advanced for Developers の画像は IBM Container Registry から入手できます：`icr.io/ibm-messaging/mq` から入手できます。 この画像は無料でダウンロードできるもので、無保証のものです。 **画像はサポートされていません**。

利用可能なプラットフォーム

ビルド済みの IBM MQ Advanced for Developers イメージは、containerd, cri-o, Podman あるいは Docker が提供するコンテナ実行環境にデプロイできます。 詳しくはシステム要件 IBM MQ をご覧ください。

利用可能な CPU アーキテクチャ

ビルド済みの IBM MQ Advanced for Developers イメージは `amd64` と `s390x` z/Linux で、`ppc64le` Power Systems バージョン 9 以上で利用可能です。 同じキューマネージャの一部であるすべてのコンテナ（例えば、Native HA レプリカ）は、同じ CPU アーキテクチャを使用しなければならないことに注意。

利用できない機能

IBM MQ Advanced for Developers の以下の機能は、ビルド済みコンテナ・イメージでは利用できません：

- 複製データ キュー マネージャー (RDQM) は使用できません。この機能は Linux カーネルと緊密に結合しているため、コンテナーではサポートされません
- キュー・マネージャのリニア・ロギングは利用できません
- **`crtmqdir`**、**`crtmqm`**、**`strmqm`**、**`endmqm`** のカスタムコマンドラインオプションの指定は利用できません - ほとんどのオプションは INI ファイルを使って設定できますが、リニアロギングの使用など、設定できないものもあります。
- オペレーティング・システム（OS）ユーザーは利用できない
- AMQP がインストールされていない
- MQTT がインストールされていない
- マネージドファイル転送エージェントは使用できません。 しかし、構築済みのコンテナ・イメージを使用して、1 つまたは複数の Coordination、Command、Agent キュー・マネージャを提供することができる。

関連概念

IBM MQ FAQ for Long Term Support and Continuous Delivery リリース

291 ページの『IBM MQ Advanced for Developers コンテナー・イメージ』

IBM MQ Advanced for Developers には、事前ビルドされたコンテナー・イメージが用意されています。 このイメージは、 IBM Container Registry から入手できます。 このイメージは、 Docker、 Podman、 Kubernetes、その他のコンテナ環境での使用に適しています。

IBM MQ Operator のバージョン・サポート

IBM MQ、OpenShift Container Platform および IBM Cloud Pak for Integration のサポート対象バージョンの対応関係。

- 35 ページの『使用可能な IBM MQ のバージョン』
- 35 ページの『互換性のある Red Hat OpenShift Container Platform のバージョン』
- 36 ページの『IBM Cloud Pak for Integration のバージョン』
- 36 ページの『以前ので使用可能な IBM MQ バージョン』
- 36 ページの『以前のオペレーターの互換性のある OpenShift Container Platform バージョン』

使用可能な IBM MQ のバージョン

オペレーター・チャネル	Operator バージョン	IBM MQ のバージョン									
		9.4.3	9.4.2	9.4.1	9.4.0	9.3.5	9.3.4	9.3.3	9.3.2	9.3.1	9.3.0
v3.6	3.6	CD	DEP	DEP	MIG	DEP	DEP	DEP	DEP	DEP	MIG
v3.5	3.5		CD	DEP	MIG	DEP	DEP	DEP	DEP	DEP	MIG
v3.4	3.4			CD	MIG	DEP	DEP	DEP	DEP	DEP	MIG
v3.3	3.3			CD	MIG	DEP	DEP	DEP	DEP	DEP	MIG
v3.2-sc2	3.2				CD と SC2	DEP	DEP	DEP	DEP	DEP	MIG

キー

- CD: Continuous Delivery サポートが使用可能です。
- SC2: IBM Cloud Pak for Integration - Support Cycle 2 (formerly Long Term Support) が使用可能です。
- MIG: IBM Cloud Pak for Integration - Support Cycle 2 (formerly Long Term Support) オペランドから Continuous Delivery オペランドへのマイグレーション時にのみ使用可能です。
- DEP: Deprecated 非推奨。 IBM MQ リリースがサポートから外れると、それらのリリースは引き続きオペレーターで構成可能になる可能性がありますが、サポートには適格ではなくなり、将来のリリースで除去される可能性があります。

各バージョンの詳細機能、変更点、フィックスなど、各バージョンの全詳細については、5 ページの『IBM MQ Operator のリリース履歴』を参照してください。

互換性のある Red Hat OpenShift Container Platform のバージョン

オペレーター・チャネル	Operator バージョン	OpenShift Container Platform のバージョン [1]					
		4.18	4.17	4.16	4.15	4.14	4.12
v3.6	3.6.0 オンワーズ	CD	CD	CD	CD	CD	CD
v3.5	3.5.1 オンワーズ	CD	CD	CD	CD	CD	CD
	3.5.0 以降		CD	CD	CD	CD	CD

[1] OpenShift Container Platform のバージョンは、それぞれのサポート日に影響を受けます。 詳しくは、OpenShift Container Platform ライフ・サイクル・ポリシー を参照してください。

オペレーター・チャネル	Operator バージョン	OpenShift Container Platform のバージョン [1]					
		4.18	4.17	4.16	4.15	4.14	4.12
v3.4	3.3.4.0 以降		CD	CD	CD	CD	CD
v3.3	3.3.3.0 以降		CD	CD	CD	CD	CD
v3.2-sc2	3.2.10 オンワーズ	SC2	SC2	SC2	SC2	SC2	SC2
	3.3.2.6 以降		SC2	SC2	SC2	SC2	SC2
	3.2.3 以降			SC2	SC2	SC2	SC2
	3.2.0 以降				SC2	SC2	SC2

キー

- CD: Continuous Delivery サポートが使用可能です。
- SC2: IBM Cloud Pak for Integration - Support Cycle 2 (formerly Long Term Support) が使用可能です。
- EOS: サポートされなくなりました。 より新しい OpenShift Container Platform バージョンにマイグレーションしてください。

IBM Cloud Pak for Integration のバージョン

各 IBM MQ Operator バージョンは、 IBM Cloud Pak for Integration バージョン（以下の表に示す）の一部として、または単独で使用するためにサポートされています。

IBM Cloud Pak for Integration バージョン	IBM MQ Operator のバージョン
16.1.2	3.6.x
16.1.1	3.4.x, 3.5.x
16.1.0	3.2.x, 3.3.x

以前ので使用可能な IBM MQ バージョン

IBM MQ 9.3 資料の 使用可能な IBM MQ バージョン を参照してください。

以前のオペレーターの互換性のある OpenShift Container Platform バージョン

IBM MQ 9.3 資料の 互換性のある OpenShift Container Platform バージョン を参照してください。

##

IBM MQ Operator によって作成されたリソースの編集

IBM MQ Operator は、ネイティブ Kubernetes リソースを作成および管理することによって、QueueManager カスタム・リソースを調整します。 これらの管理対象リソースを直接編集しては **なりません**。

[1] OpenShift Container Platform のバージョンは、それぞれのサポート日に影響を受けます。 詳しくは、OpenShift Container Platform ライフ・サイクル・ポリシー を参照してください。

通常、 ownerReferences を調べることによって、あるリソースが別の上位レベルのリソースによって所有されているかどうかを判別できます。 例えば、 StatefulSet から取得された以下のメタデータは、それが QueueManager リソース「qm1」によって所有されていることを示しています。

```
metadata:
  ownerReferences:
    - apiVersion: mq.ibm.com/v1beta1
      kind: QueueManager
      name: qm1
      uid: 60fda34c-9f7c-42d2-a293-78fec4315c62
      controller: true
      blockOwnerDeletion: true
```

すべてのリソースにこのメタデータがあるわけではないことに注意してください。

基礎となるリソース (StatefulSet、 Service 、 Route など) を管理するのは、 IBM MQ Operator の責任です。 これらの基礎となるリソースのいずれかを変更すると、 IBM MQ Operator はそれらを元に戻し、その変更にローリング更新が必要な場合はダウン時間が発生する可能性があります。

キュー・マネージャーの重要な設定のほとんどは、 QueueManager リソースで使用可能です。 ただし、基礎となるリソースを完全に制御する必要がある場合は、いくつかのオプションがあります。

- IBM MQ Operator によって作成されたポッドの設定をオーバーライドする必要がある場合は、 QueueManager YAML の .spec.template セクションにポッド・オーバーライド・テンプレートを追加できます。
- IBM MQ Operator によって作成されたキュー・マネージャー Route の設定をオーバーライドする必要がある場合は、経路全体で .spec.route.enabled 設定を「false」に設定してから、独自の経路を作成する必要があります。
- ラベルやアノテーションなどの設定や、 security Context などの Pod 設定は、すべて QueueManager リソースで設定できます。
- また、完全な制御が必要な場合は、 IBM MQ Operator がお客様のユース・ケースに適していないことがあります。

コンテナーでの IBM MQ のライセンス交付の計画

コンテナー・ライセンスにより、コンテナーが実行されているサーバー全体のライセンス交付を必要とするのではなく、個々の IBM MQ コンテナーの使用可能な容量のみのライセンス交付を受けることができます。 コンテナ ライセンスを利用するには、 IBM License Service を使用してライセンスの使用状況を追跡し、必要なエンタイトルメントを決定する必要があります。

コンテナベースのライセンスは、 Red Hat OpenShift Container Platform または Kubernetes を使用している場合にのみ利用可能であることに注意してください。 詳細については、 コンテナのライセンスに関する FAQ をご覧ください。

IBM License Service を使用する：

- IBM MQ Operator を使用している場合は、 IBM MQ Operator インストール・プロセスの License Service ステップ「Deploy the 」を参照してください。
- Kubernetes にデプロイする場合は、IBM Cloud Pak foundational services ドキュメントの Kubernetes クラスターに License Service をインストールするを参照してください。

Red Hat OpenShift Container Platform または Kubernetes を使用していない場合は、 IBM License Metric Tool を使用する必要があります。 このツールは、サブキャパシティライセンスを利用するために従来の仮想化環境に導入された IBM 製品のライセンス使用量を測定し、レポートするために必要です。 詳しくは、 IBM License Metric Tool ドキュメントを参照。

関連資料

281 ページの『IBM MQ コンテナライセンス注釈』
Kubernetes および Red Hat OpenShift Container Platform 環境では、IBM License Service は Pod アノテーションを使用して、基礎となるマシンではなくコンテナで定義された制限に基づいて使用量を追跡する。 IBM MQ Operator 、これらの Pod アノテーションを設定してくれますが、他の方法でデプロイする場合は、自分で設定する必要があります。

関連情報
IBM コンテナー・ライセンス
コンテナーのライセンス交付に関する FAQ
ライセンス・サービスのインストール
ライセンス使用の表示および追跡

IBM MQ Operator のストレージの計画

IBM MQ Operator は、次の 2 つのストレージ・モードで稼働します。

- **一時ストレージ**は、コンテナーの再始動時にコンテナーのすべての状態情報を破棄してよい場合に使用します。 これは、デモンストレーション用の環境を作成する場合や、スタンドアロンのキュー・マネージャーを使用して開発する場合によく使用されます。
- **永続ストレージ**は IBM MQ の一般的な構成であり、コンテナーが再始動されても、既存の構成、ログ、永続メッセージを再始動後のコンテナーで使用することができます。

IBM MQ Operator は、環境によってかなり異なるものになることがあるストレージ特性と、必要なストレージ・モードをカスタマイズする機能を備えています。

一時ストレージ

IBM MQ はステートフル・アプリケーションであるため、再始動時にリカバリーできるように、自身の状態をストレージに保存します。 一時ストレージを使用している場合は、キュー・マネージャーのすべての状態情報が再始動時に失われます。 これには、以下が含まれます。

- すべてのメッセージ
- すべてのキュー・マネージャー間通信の状態 (チャネルのメッセージ・シーケンス番号)
- キュー・マネージャーの MQ クラスター ID
- すべてのトランザクション状態
- すべてのキュー・マネージャー構成
- ローカルにあるすべての診断データ

このため、実稼働、テスト、または開発のシナリオにとって一時ストレージが適したアプローチであるかどうか検討する必要があります。 例えば、すべてのメッセージが非永続メッセージであると認識され、キュー・マネージャーが MQ クラスターのメンバーでない場合は、 再始動時にすべてのメッセージング状態が廃棄されるだけでなく、キュー・マネージャーの構成も廃棄されます。 完全に一時的であるコンテナーを有効にするには、コンテナー・イメージ自体に IBM MQ 構成を追加する必要があります (詳しくは、159 ページの『Red Hat OpenShift CLI を使用した、カスタム MQSC および INI ファイルを使用したイメージの作成』を参照してください)。 これを行わない場合は、コンテナーが再始動するたびに IBM MQ を構成する必要があります。

例えば、IBM MQ に一時ストレージを構成するには、`QueueManager` のストレージ・タイプに以下を指定する必要があります。

```
queueManager:
  storage:
    queueManager:
      type: ephemeral
```

永続ストレージ

IBM MQ は通常、永続ストレージを使用して実行され、キュー・マネージャーが再始動後も永続メッセージと構成を保持するようにします。 これはデフォルトの動作です。 さまざまなストレージ・プロバイダーがあり、それぞれが異なる機能をサポートしているため、多くの場合、構成のカスタマイズが必要になります。 以下の例は、 v1beta1 API の IBM MQ ストレージ構成をカスタマイズする共通フィールドの概要を示しています。

- **spec.queueManager.availability** は、可用性モードを制御します。 SingleInstance または NativeHA を使用している場合は、 ReadWriteOnce ストレージのみが必要です。 multiInstance には、正しいファイル・ロック特性を持つ ReadWriteMany をサポートするストレージ・クラスが必要です。 IBM MQ は、サポートに関するステートメントとテストに関するステートメントを提示しています。 可用性モードは、永続ボリュームのレイアウトにも影響します。 詳しくは、40 ページの『コンテナー内の IBM MQ の高可用性の計画』を参照してください。
- **spec.queueManager.storage** は、個々のストレージ設定を制御します。 キュー・マネージャーは、1 つから 4 つの永続ボリュームを使用するように構成できます。

次の例は、単一インスタンスのキュー・マネージャーを使用する単純な構成のスニペットを示しています。

```
spec:
  queueManager:
    storage:
      queueManager:
        enabled: true
```

次の例は、マルチインスタンスのキュー・マネージャー構成のスニペットを示しており、デフォルトではないストレージ・クラスを指定し、補助グループを必要とするファイル・ストレージを指定しています。

```
spec:
  queueManager:
    availability:
      type: MultiInstance
    storage:
      queueManager:
         class: ibmc-file-gold-gid
      persistedData:
        enabled: true
        class: ibmc-file-gold-gid
      recoveryLogs:
        enabled: true
        class: ibmc-file-gold-gid
  securityContext:
    supplementalGroups: [65534] # Change to 99 for clusters with RHEL7 or earlier worker nodes
```

ネイティブ HA キュー・マネージャーのストレージに関する考慮事項については、42 ページの『ネイティブ HA』を参照してください。

注: また、単一インスタンス・キュー・マネージャーを使用して、補足グループを構成することもできます。

ストレージ容量

OpenShift CP4I

IBM MQ Operator を使用する場合は、継続的なニーズに十分な大きさのボリュームを要求するようにしてください。 ただし、1 つ以上のボリュームのストレージ容量を増やす必要がある場合、ご使用のストレージ・クラスがボリューム拡張をサポートしていれば、これらのボリュームを拡張することができます。 ボリュームは、オンラインまたはオフラインのいずれかの手順で拡張できます。 オフライン手順では QueueManager ポッドを再始動する必要がありますが、オンライン手順では再始動する必要はありません。 ご使用のストレージ・クラスがボリューム拡張をサポートしているかどうか、およびボリューム拡張がどの手順に従うかを判別するには、ご使用のストレージ・プロバイダーの資料を参照してください。 ストレージ・クラスを選択する際には、この情報を考慮する必要があります。 ボリューム拡張のガイドについては、168 ページの『永続ボリュームの拡張』を参照してください。

暗号化

OpenShift CP4I

IBM MQ は、保存データをアクティブに暗号化しません。 したがって、メッセージを暗号化するには、パッシブに暗号化されたストレージまたは IBM MQ Advanced Message Security、あるいはその両方を使用する必要があります。 IBM Cloud® では、ブロック・ストレージとファイル・ストレージの両方で、保存時にパッシブ暗号化を使用できます。

OpenShift Kubernetes コンテナー内の IBM MQ の高可用性の計画

IBM MQ Operator での高可用性には 3 つの選択肢があります。 **ネイティブ HA キュー・マネージャー** (アクティブなレプリカと 2 つのスタンバイ・レプリカを持つ)、 **複数インスタンス・キュー・マネージャー** (共有ネットワーク・ファイル・システムを使用するアクティブ/スタンバイ・ペア)、または **単一回復力キュー・マネージャー** (ネットワーク・ストレージを使用する HA の単純なアプローチを提供する) です。 後者の 2 つは、リカバリー可能データを確実に使用できるかどうかは、ファイル・システムが鍵を握りますが、ネイティブ HA ではそうではありません。 そのため、ネイティブ HA を使用しない場合、ファイル・システムの可用性はキュー・マネージャーの可用性にとって非常に重要となります。 データ・リカバリーが重要となる場合は、複製を行ってファイル・システムの冗長性を確保する必要があります。

メッセージの可用性と**サービス**の可用性は分けて考える必要があります。 IBM MQ for Multiplatforms を使用する場合、メッセージは厳密に 1 つのキュー・マネージャーに保管されます。 そのため、そのキュー・マネージャーが使用不可になると、その中に保管されているメッセージに一時的にアクセスできなくなります。 メッセージの可用性を高めるためには、できるだけ速やかにキュー・マネージャーを復旧できなければなりません。 サービスの可用性を高めるには、IBM MQ 均一クラスターを使用するなど、クライアント・アプリケーションが使用するキューのインスタンスを複数用意しておくことができます。

キュー・マネージャーは、ディスク上に保管されるデータとそのデータへのアクセスを可能にする実行プロセスの 2 つの部分に分けて考えることができます。 キュー・マネージャーは、同じデータ (Kubernetes Persistent Volumes によって提供されたもの) を保持し、クライアント・アプリケーションによってネットワーク上で引き続きアドレス可能である限り、別の Kubernetes ノードに移動することができます。 Kubernetes では、ネットワークにおける同一性を維持するために 1 つのサービスが一貫して使用されます。

IBM MQ は、永続ボリュームのデータの可用性に依存しています。 このため、IBM MQ の可用性は使用するストレージの可用性を上回ることができないので、永続ボリュームを提供するストレージの可用性はキュー・マネージャーの可用性にとって非常に重要となります。 可用性ゾーン全体の障害を許容する場合は、ディスク書き込みを別のゾーンに複製するボリューム・プロバイダーを使用することが必要です。

ネイティブ HA キュー・マネージャー

MQ Adv.

ネイティブ HA キュー・マネージャーには、1 つの **アクティブ** と 2 つの **レプリカ** Kubernetes ポッドが含まれます。これらのポッドは、 Kubernetes StatefulSet の一部として実行され、それぞれに独自の Kubernetes 永続ボリューム・セットを持つ正確に 3 つのレプリカが含まれます。 ネイティブ HA キュー・マネージャーを使用する場合、IBM MQ での共有ファイル・システムの要件も適用されますが (リース・ベースのロックを除く)、共有ファイル・システムを使用する必要はありません。 上部に適切なファイル・システムを配置することで、ブロック・ストレージを使用できます。 例えば、*xfs* や *ext4* を配置します。 ネイティブ HA キュー・マネージャーが復旧に要する時間は、以下の要因によって左右されます。

1. アクティブ・インスタンスに障害が発生したことをレプリカ・インスタンスが検出するのにどれほどの時間がかかるか。 これは構成可能です。
2. 作動可能コンテナーが変更されてネットワーク・トラフィックがリダイレクトされたことを Kubernetes ポッドの Readiness Probe が検出するまでにかかる時間。 これは構成可能です。
3. IBM MQ クライアントが再接続するまでにかかる時間。

詳細については、42 ページの『ネイティブ HA』を参照してください。

ネイティブ HA CRR（クロスリージョンレプリケーション）

V 9.4.2 MQ Adv.

別々の地域に配置された 2 つのネイティブ HA グループで構成される災害復旧ソリューションを実装できます。 一方のネイティブ HA グループは**ライブ**の役割を持ち、もう一方のグループは**リカバリー**の役割を持つ。 ライブグループがあるリージョンで計画的または計画外の障害が発生した場合、ユーザーアクションによって、リモートリージョンにあるグループの役割をリカバリからライブに切り替えることができる。 その後、このグループは、元のキュー・マネージャー・インスタンスの作業を引き継ぎ、アプリケーション接続などを受け入れます。

詳細については、46 ページの『ネイティブ HA クロスリージョンレプリケーション』を参照してください。

複数インスタンス・キュー・マネージャー

複数インスタンス・キュー・マネージャーには**アクティブ**で**スタンバイ状態の** Kubernetes ポッドが必要で、これらは厳密に 2 つのレプリカと Kubernetes 永続ボリューム一式と共に Kubernetes ステートフル・セットの一部として稼働します。 キュー・マネージャーのトランザクション・ログとトランザクション・データは、共用ファイル・システムを使用して、2 つの永続ボリュームに保管されます。

複数インスタンス・キュー・マネージャーには、永続ボリュームへの同時アクセスを可能にするために、**アクティブな**ポッドと**スタンバイ状態の**ポッドの両方が必要です。 これを構成するには、'KubernetesPersistent Volumes を使用し、'**access mode** を ReadWriteMany に設定する。 これらのボリュームは、IBM MQ の共有ファイル・システムの要件も満たしていなければなりません。IBM MQ がキュー・マネージャー・フェイルオーバーの実施をファイル・ロックの自動解除に依存しているからです。IBM MQ はテスト対象ファイル・システムのリストを作成します。

複数インスタンス・キュー・マネージャーが復旧に要する時間は、以下の要因によって左右されます。

1. 障害が発生した後、もともとアクティブ・インスタンスによって実行されたロックを共用ファイル・システムが解除するためにかかる時間。
2. スタンバイ状態のインスタンスがロックを取得してから起動するまでにかかる時間。
3. 作動可能コンテナーが変更されてネットワーク・トラフィックがリダイレクトされたことを Kubernetes ポッドの Readiness Probe が検出するまでにかかる時間。 これは構成可能です。
4. IBM MQ クライアントが再接続するまでにかかる時間。

シングル・レジリエント・キュー・マネージャー

シングル・レジリエント・キュー・マネージャーは、1 つの Kubernetes ポッド内で実行されるキュー・マネージャーの単一のインスタンスのことで、ここで Kubernetes はキュー・マネージャーをモニタリングし、必要に応じてこのポッドを置き換えます。

シングル・レジリエント・キュー・マネージャーを使用する場合、IBM MQ での共有ファイル・システムの要件も適用されますが (リース・ベースのロックを除く)、共有ファイル・システムを使用する必要はありません。 上部に適切なファイル・システムを配置することで、ブロック・ストレージを使用できます。 例えば、 *xfs* や *ext4* を配置します。

シングル・レジリエント・キュー・マネージャーが復旧に要する時間は、以下の要因によって左右されます。

1. Liveness プローブの実行にかかる時間、および Liveness プローブが許容する失敗の数。 これは構成可能です。
2. Kubernetes スケジューラーが失敗したポッドを新規ノードに再スケジュールするためにかかる時間。
3. コンテナー・イメージを新規ノードにダウンロードするためにかかる時間。 `IfNotPresent` に **imagePullPolicy** 値を使用している場合は、すでにそのノードで画像が利用可能になっている可能性があります。
4. 新規キュー・マネージャー・インスタンスが起動するのにかかる時間。
5. コンテナーが作動可能であることを Kubernetes ポッドの Readiness Probe が検出するまでにかかる時間。 これは構成可能です。
6. IBM MQ クライアントが再接続するまでにかかる時間。

重要 :

シングル・レジリエント・キュー・マネージャー・パターンにはいくつかの利点がありますが、ノードの障害に関連した制限がある状態で、目標とする可用性に達するかどうかについて十分に理解しておく必要があります。

Kubernetes では、障害が発生したポッドは通常迅速に復旧しますが、ノード全体で障害が発生した場合は対応が異なります。 IBM MQ ようなステートフルなワークロードを「Kubernetes StatefulSet,で使用する場

合、「Kubernetes マスターノードがワーカーノードとの接触を失った場合、そのノードが故障したのか、単にネットワーク接続が失われたのかを判断できない。 そのため、このような場合、次のいずれかのイベントが発生するまで、Kubernetes は**何も対応しません**。

1. ワーカー・ノードが Kubernetes マスター・ノードと通信できる状態に復旧する。
2. 管理上の処置として、Kubernetes マスター・ノード上のポッドを明示的に削除する。 この場合、必ずしもポッドの実行を停止するわけではなく、単に Kubernetes ストアから削除します。 したがって、この管理上の処置は慎重に行う必要があります。

注 : IBM MQ Operator を使用してキュー・マネージャーを作成する場合、 IBM MQ キュー・マネージャーの StatefulSet の詳細 (レプリカの数を含む) の変更はサポートされません。

関連概念

高可用性の構成

関連タスク

111 ページの『IBM MQ Operator を使用したキュー・マネージャーの高可用性の構成』

キューマネージャの高可用性を設定するには、ネイティブ HA ソリューションとマルチインスタンス・キューマネージャ・ソリューションのいずれかを使用します。

CP4I MQ Adv. Kubernetes ネイティブ HA

ネイティブ HA は、IBM MQ 用のネイティブ (組み込み) 高可用性ソリューションであり、クラウド・ブロック・ストレージで使用するときに適しています。

ネイティブ HA 構成によって、高可用性キュー・マネージャーを実行できます。このキュー・マネージャーは、リカバリー可能な MQ データ (メッセージなど) のレプリケーションをストレージの複数セットにまたがって行うことにより、ストレージの障害によるデータの損失を防ぎます。 このキュー・マネージャーには実行中のインスタンスが複数あり、そのうちの 1 つがリーダーになります。他のキュー・マネージャーは、障害発生時にすぐにテークオーバーできるように準備を整えているので、キュー・マネージャーとメッセージへのアクセスが最大化されます。

ネイティブ HA 構成は、3 つの Kubernetes ポッドから成り、その個々にキュー・マネージャーのインスタンスがあります。 1 つのインスタンスがアクティブ・キュー・マネージャーとして機能し、メッセージはそこで処理されてリカバリー・ログに書き込まれます。 そのリカバリー・ログへの書き込みが行われると、アクティブ・キュー・マネージャーはレプリカと呼ばれる他の 2 つのインスタンスにデータを送信します。各レプリカは、個別に所有するリカバリー・ログへの書き込みを行い、データを認知し、複製されたリカバリー・ログからキュー・データをそれぞれ更新します。 アクティブ・キュー・マネージャーを実行しているポッドに障害が発生すると、キュー・マネージャーのレプリカ・インスタンスの 1 つがアクティブの役割を引き継ぎ、このインスタンスにある最新データで運用が行われます。

ログ・タイプは「複製ログ」と呼ばれます。 複製されたログは基本的にリニア・ログであり、自動ログ管理と自動メディア・イメージが有効になっています。 ログのタイプを参照してください。 リニア・ログの管理に使用するものと同じ手法を使用して、複製されたログを管理します。

ネットワーク・トラフィックを扱う準備ができている唯一のポッドであると識別された現行アクティブ・インスタンスに TCP/IP クライアント接続をルーティングするために、Kubernetes サービスが使用されます。 この処理を行うために、クライアント・アプリケーションがさまざまなインスタンスを認識しておく必要はありません。

3 つのポッドを使用すると、スプリット・ブレーンという状態になる可能性が大幅に減少します。 2 ポッドの高可用性システムでは、2 つのポッド間の接続が切断されるとスプリット・ブレーンが発生する可能性があります。 接続のない状態だと、両方のポッドでキュー・マネージャーが同時に実行され、別々のデータが累積される可能性があります。 そうなると、接続が復元された時に 2 つの別々のバージョンのデータ (「スプリット・ブレーン」) が存在することになるので、保持するデータ・セットと破棄するデータ・セットを決定するために手操作による介入が必要になります。

ネイティブ HA では、スプリット・ブレーン状態を回避するために、クォーラムを設定した 3 ポッド・システムを使用します。 1 つのポッドが他の 1 つ以上のポッドと通信できる場合は、その通信可能なポッド同士がクォーラムを形成します。 キュー・マネージャーは、クォーラムを形成するポッド上にある場合にのみ、アクティブ・インスタンスになることができます。 キュー・マネージャーは、他の 1 つ以上のポッド

と接続していないポッド上ではアクティブになることはできないので、2 つのアクティブ・インスタンスが同時に存在することは絶対にありません。

- 1 つのポッドで障害が発生しても、他の 2 つのポッドのいずれかにあるキュー・マネージャーが引き継ぐことができます。 2 つのポッドで障害が発生すると、残りの 1 つのポッドにあるキュー・マネージャーはアクティブ・インスタンスになることができません。そのポッドはクォーラムを形成していないからです (その残りのポッドは、他の 2 つのポッドで障害が発生したのか、それとも自分の接続が失われていて他の 2 つのポッドはまだ実行されているのかの違いを判別できません)。
- 1 つのポッドが接続を失うと、そのポッドにあるキュー・マネージャーはアクティブになることができません。そのポッドはクォーラムを形成していないからです。 残りの 2 つのポッドでは、いずれか 1 つのキュー・マネージャーが引き継ぎを実行できます。クォーラムを形成しているからです。 すべてのポッドが接続を失うと、どのポッドのキュー・マネージャーもアクティブになることができません。どのポッドもクォーラムを形成していないからです。

アクティブ・ポッドに障害が発生し、その後復旧すると、レプリカの役割でグループに再参加することができます。

パフォーマンスと信頼性のために、RWOReadWriteOnce）永続ストレージを Native HA 構成で使用することを推奨する。 すべてのストレージ・プロバイダーからの RWO ボリュームは、以下の条件を満たしている場合にサポートされます。

- ブロック・ストレージ・プロバイダーから取得されます。
- ext4 または XFS としてフォーマットされます (これにより、 POSIX 準拠が保証されます)。
- ダイナミック・ボリューム・プロビジョニングと"volumeBindingMode: WaitForFirstConsumer"」をサポート。

以下のプロバイダーは、明示的に禁止されています。

- NFS
- GlusterFS
- その他の非ブロック・プロバイダー。

注 : V 9.4.2 ネイティブ HA のクロスリージョンレプリケーションを設定する場合は、ネイティブ HA の 2 倍以上のディスク容量が必要です。 これは、リベース中にログバックアップディレクトリが必要になるためです（46 ページの『ネイティブ HA クロスリージョンレプリケーション』 を参照）

以下の図に、1 つのキュー・マネージャーの 3 つのインスタンスが 3 つのコンテナー内にデプロイされている、標準的なデプロイメントを示します。

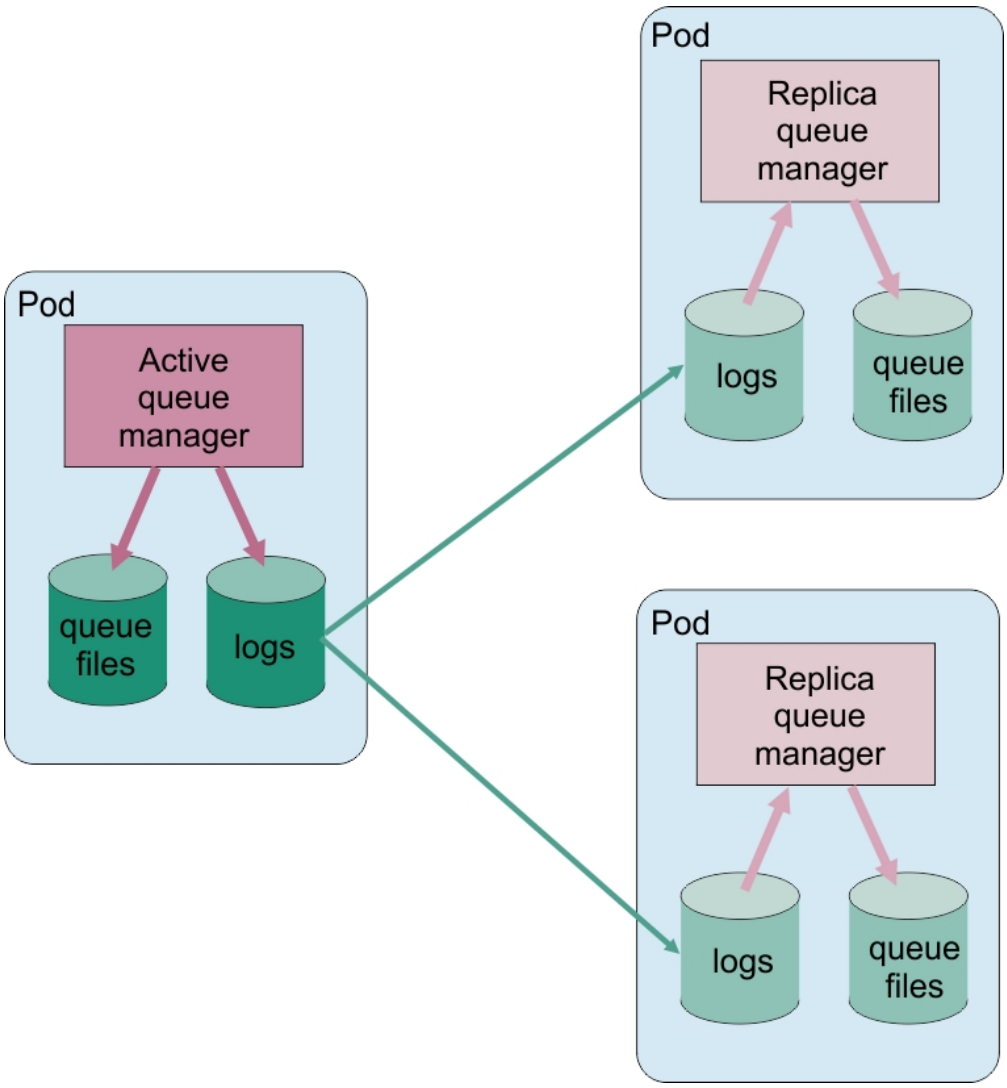


図 *1.* ネイティブ *HA* 構成の例

MQ Adv. Kubernetes ネイティブ HA キュー・マネージャーの独自ローリング更新を実行する場合の考慮事項

ネイティブ HA キュー・マネージャーの IBM MQ バージョンまたはポッド仕様を更新する場合は、キュー・マネージャー・インスタンスのローリング更新を実行する必要があります。 これは IBM MQ Operator によって自動的に処理されますが、独自のデプロイメント・コードを作成する場合は、いくつかの重要な考慮事項があります。

注 : サンプル Helm チャートには、ローリング更新を実行するためのシェル・スクリプトが含まれていますが、このスクリプトは、このトピックにおける考慮事項には対応していないため、実動での使用には適し**ていません**。

Kubernetes Kubernetes では、`StatefulSet` リソースは、順序付けられた始動およびローリング更新を管理するために使用されます。 始動手順の一環として、各ポッドを個別に開始し、そのポッドが作動可能になるまで待ってから、次のポッドに移動します。 すべてのポッドを開始してリーダー選出を実行できるようにする必要があるため、これはネイティブ HA では機能しません。 したがって、`.spec.podManagementPolicy` フィールド (`StatefulSet` 上) を `Parallel` に設定する必要があります。 これはすべてのポッドが並行して更新されることも意味しますが、これは特に望ましくありません。 このため、 `StatefulSet` は `OnDelete` 更新方針も使用する必要があります。

`StatefulSet` ローリング更新コードを使用できないため、カスタム・ローリング更新コードが必要になります。以下の点を考慮してください。

- 一般的なローリング更新手順
- 最適な順序でポッドを更新することによるダウン時間の最小化
- クラスター状態の変更を処理
- エラーの処理
- タイミングの問題を処理

一般的なローリング更新手順

ローリング更新コードは、各インスタンスが REPLICA の状況を dspmq から示すまで待ちます。 つまり、インスタンスは何らかのレベルの始動を行った (例えば、コンテナーが開始され、MQ プロセスが実行されている) が、必ずしもその他のインスタンスと対話するようにはなっていません。 例えば、ポッド A が再始動され、 REPLICA 状態になるとすぐにポッド B が再始動されます。 ポッド B は新しい構成で開始されると、ポッド A と対話できるようになり、クォーラムを形成できます。A または B のいずれかが新しいアクティブ・インスタンスになります。

その一環として、各ポッドが REPLICA 状態になった後に遅延を設定して、そのポッドがピアに接続してクォーラムを確立できるようにすると便利です。

最適な順序でポッドを更新することによるダウン時間の最小化

ローリング更新コードは、既知のエラー状態にあるポッドから、正常に開始されなかったポッドまで、一度に 1 つずつポッドを削除する必要があります。 通常、アクティブなキュー・マネージャー・ポッドは最後に更新する必要があります。

また、前回の更新でポッドが既知のエラー状態になった場合は、ポッドの削除を一時停止することも重要です。 これにより、すべてのポッドにわたって、中断された更新がロールアウトされなくなります。 例えば、これが行われる可能性があるのは、アクセスできない (または誤植が含まれる) 新しいコンテナー・イメージを使用するようにポッドが更新される場合です。

クラスター状態の変更を処理

ローリング更新コードは、クラスター状態のリアルタイム変更に適切に対応する必要があります。 例えば、キュー・マネージャーのポッドの 1 つが、ノード・リブートまたはノード圧力が原因で排除されることがあります。 排除されたポッドは、クラスターがビジーだと即時に再スケジュールされない可能性があります。 この場合、ローリング更新コードは、他のすべてのポッドを再始動する前に適切に待機する必要があります。

エラーの処理

ローリング更新コードは、Kubernetes API の呼び出し時や他の予期しないクラスター動作時の障害に対して堅牢でなければなりません。

さらに、ローリング更新コード自体は、再始動を許容できなければなりません。 ローリング更新は長時間実行される可能性があり、コードの再始動が必要になる可能性があります。

タイミングの問題を処理

ローリング更新コードは、ポッドが再始動したことを確認できるように、ポッドの更新改訂を検査する必要があります。 これにより、ポッドが「開始済み」であることを示していても実際はまだ終了していないというタイミングの問題が回避されます。

関連概念

28 ページの『コンテナー内の IBM MQ の使用方法の選択』

コンテナー内の IBM MQ の使用方法としては、いくつもの選択肢があります。プリパッケージされているコンテナー・イメージを使用する IBM MQ Operator を使用することも、独自のイメージとデプロイメント・コードをビルドすることもできます。

OpenShift CP4I Kubernetes コンテナー内の IBM MQ の災害復旧

どのような種類の災害に備えるのかを検討する必要があります。 クラウド環境では、複数のアベイラビリティー・ゾーンを使用すると、災害に対して一定レベルの耐障害性を確保できる上、利用方法も簡単です。奇数個のデータ・センター (クォーラムの場合) があり、ネットワーク・リンクの待ち時間が短い場合は、単一の Red Hat OpenShift Container Platform クラスターまたは Kubernetes クラスターを、別々の物理的な場所にある複数のアベイラビリティー・ゾーンで実行することもできます。 このトピックでは、これらの基準を満たすことができない場合 (つまり、データ・センターが偶数個の場合やネットワーク・リンクの待ち時間が長い場合) の災害復旧に関する考慮事項を説明します。

災害復旧では、以下の点を考慮する必要があります。

- 災害復旧ロケーションへの IBM MQ データ (1 つ以上の `PersistentVolume` リソースで保持されている) の複製
- 複製されたデータを使用してキュー・マネージャーを再作成すること
- IBM MQ クライアント・アプリケーションおよびその他のキュー・マネージャーに表示されるキュー・マネージャーのネットワーク ID。 例えば、この ID は DNS 項目になります。

永続データを災害復旧サイトに同期的または非同期的に複製する必要があります。 通常、これはストレージ・プロバイダーに固有ですが、`VolumeSnapshot` を使用して行うこともできます。 ボリュームのスナップショットについて詳しくは、CSI volume snapshots を参照してください。

災害から復旧する場合には、複製されたデータを使用して、新しい Kubernetes クラスター上でキュー・マネージャー・インスタンスを再作成する必要があります。 IBM MQ Operator を使用している場合は、`QueueManager` YAML が必要なほか、`ConfigMap` や `Secret` など、他のサポート・リソース用の YAML も必要になります。

関連情報

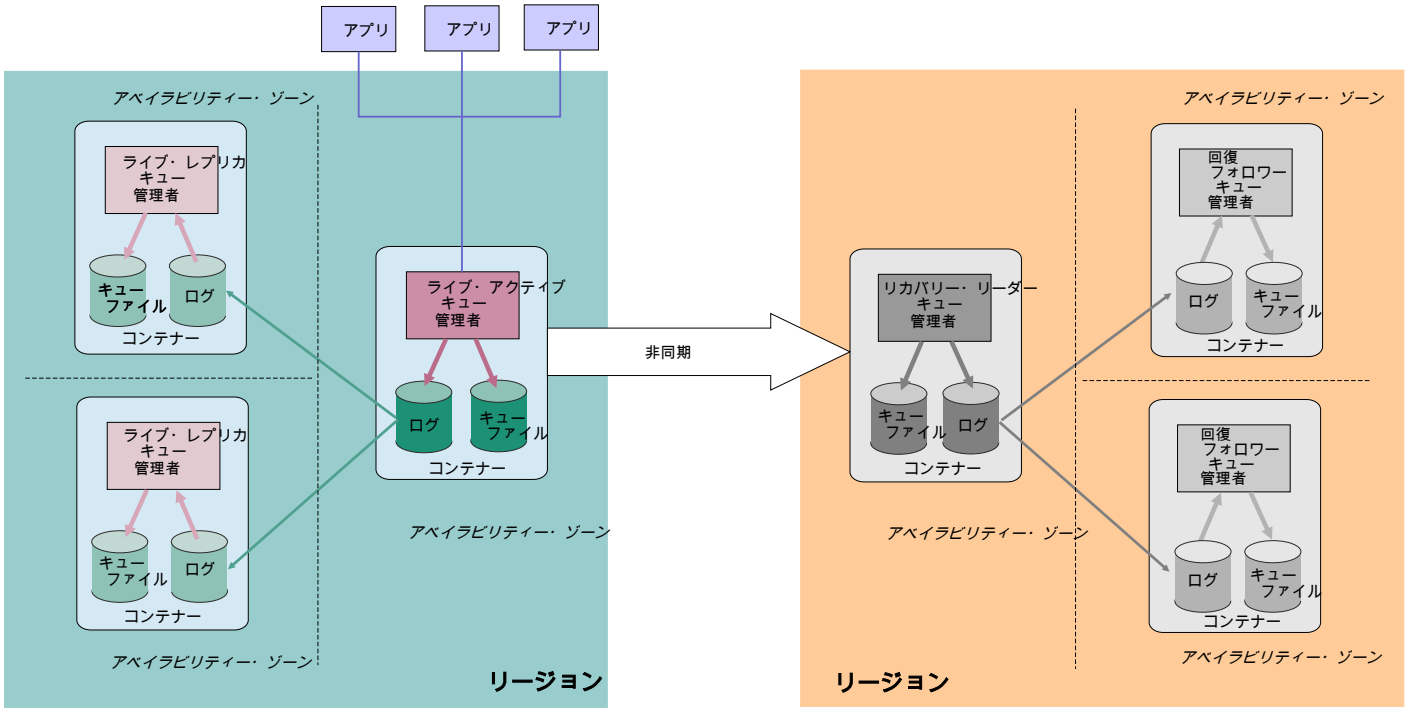
ha_for_ctr.dita

V 9.4.2 ネイティブ HA クロスリージョンレプリケーション

ネイティブ HA ソリューションをベースに、ネイティブ HA クロスリージョンレプリケーション（CRR）構成を実装することができます。 CRR 構成は、災害復旧を提供します。

ネイティブ HA ソリューションの概要については、42 ページの『ネイティブ HA』 をご覧ください。

Native HA は、3 つのキューマネージャーインスタンスのグループを使用して、ログデータの複製されたコピーを維持します。 グループは、データの最良のコピーを比較・選択することで、アクティブなインスタンスを自動的に選択します。 CRR は、地理的に離れた可能性のあるグループ（地域）も含むようにログレプリケーションを拡張し、高い可用性を実現します。 本サイトで計画または計画外の停止が発生した場合、復旧サイトのグループが業務を引き継ぐことができます。 グループの役割を最初に定義し、計画された切り替えを完了させるために、または計画外の停止が発生した場合に、必要に応じてグループの役割を手動で切り替えます。 ログデータは非同期でリカバリーグループにレプリケートされるため、計画外の停電に伴う切り替え時にデータ損失が発生する可能性がある。



希望する役割と効果的な役割

要求された役割は、 qm.ini の NativeHALocalInstance スタンザで指定されたネイティブ HA インスタンスの GroupRole 構成設定です。 設定では、インスタンスがグループ内の他のインスタンスと協力して役割を果たすことが求められます。 インスタンスの大多数が同じリクエストされた役割を共有すると、グループのリーダーを選ぶための選挙を行うことができます。 選挙に勝利すると、要求されたロールが、そのインスタンスが最後に使用したロールと比較され、有効なロールが導き出されます。 qm.ini ファイルの NativeHAInstance スタンザと qm.ini ファイルの NativeHALocalInstance スタンザを参照してください。

生きている役割

ライブグループは、3 つのインスタンスで構成される通常の Native HA キューマネージャです。 1 つのインスタンスがアクティブなものとして選ばれ、このインスタンスがアプリケーションからの新しい作業を受け入れます。 既存のネイティブ HA キューマネージャーをネイティブ CRR 構成に移行することができます。

qm.ini で明示的に指定されていない場合、デフォルトは「ライブ」とみなされます。 ライブグループの選出されたリーダーは、アクティブキューマネージャとも呼ばれ、このインスタンスは、新しいメッセージング作業の実行やキューマネージャオブジェクトの管理のために、アプリケーションやツールの接続を受け入れることができます。

回復の役割

リカバリグループも 3 つのインスタンスで構成され、そのうちの 1 つがリーダーとして選出されます。 リーダーは代理としてのみ機能し、ペアになっている Live グループから送られてきた作業を受け入れます。 リカバリグループへのアプリケーション接続は許可されていませんが、キューマネージャーインスタンスの終了などの操作を完了できるように、一部の制御コマンドは許可されています。

qm.ini 構成ファイルの NativeHARecoveryGroup 節は、Recovery グループのネットワークアドレスを特定します。

保留中のロール移行

現在のライブグループを新しいリカバリグループとし、現在のリカバリグループを新しいライブグループとする計画的な切り替えを行う場合、グループ間の調整が必要です。 この調整により、ログが同一であることが保証され、RPO（目標復旧時点）がゼロになります。

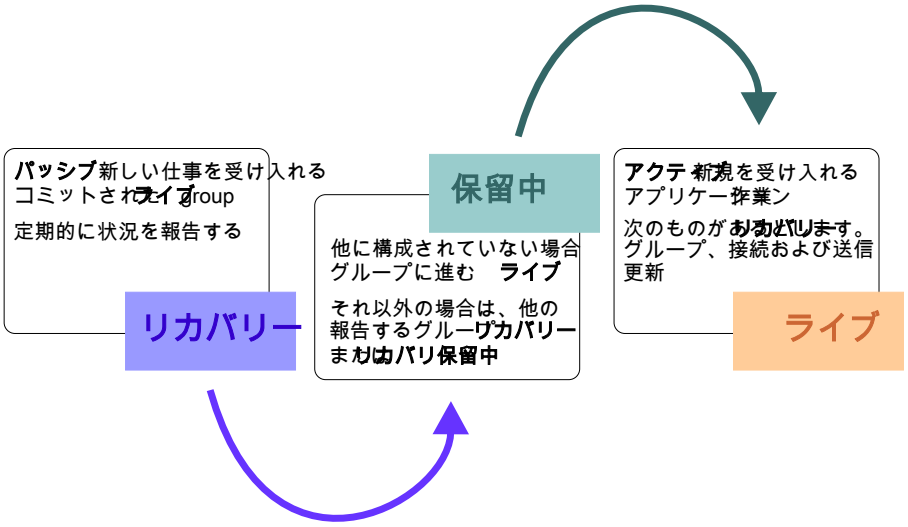
2 つのグループが Recovery と Live 間の切り替えを調整している場合、両グループで一時的に有効な保留中の役割が採用されます。

リカバリロールからライブロール（ライブ保留中）

以前にリカバリの役割で運用していたグループがライブの役割に変更する場合は、別のグループが設定されているかどうかを基に、その役割を使用するタイミングを決定します。

他に設定されたグループがない場合は、即座にライブロールに移行します。

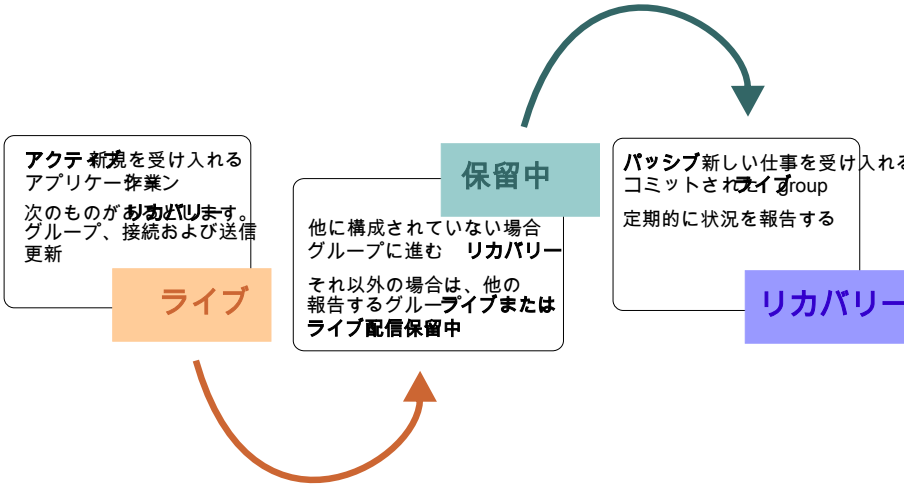
別のグループが構成され、有効になっている場合、そのグループがリカバリまたはリカバリ保留のロールを引き受けるまで、また両方のグループが同期するまでは、ライブロールへの移行はブロックされます。



ライブロールからリカバリロール（リカバリ保留中）

以前はライブロールで運用されていたグループがリカバリロールに変更する場合は、別のグループが構成されているかどうかに基づいて、リカバリロールを使用するタイミングを決定します。

別のグループが構成され、有効になっている場合、そのグループがライブロールまたは保留中ライブロールに変更され、両方のグループが同期されるまで、リカバリロールへの移行はブロックされます。



複製と再ベース化

Native HA CRR 構成の 2 つのグループが接続されている間、ログの更新はリカバリグループのリーダーに渡され、他のリカバリインスタンスに転送されます。 これを**複製**といいます。

2 つのグループ間のネットワーク接続が失われた場合、レプリケートされたログを使用してもリカバリグループが追いつけないことがあります（例えば、ライブグループが追いつくために必要なログエクステントを再利用した場合など）。 この場合、リカバリリーダーは**リベース**を実行します。 リベースとは、受信したログデータを破棄し、ライブグループから送信されたログデータの完全なセットからキューマネージャ

を再構築するプロセスです。 リカバリリーダーがリベースを完了すると、ログがリカバリインスタンスにレプリケートされます。

リベース操作中に問題が発生した場合、キューマネージャーがリカバリーグループのメンバーのいずれでも起動できなくなる可能性があります。 この可能性を避けるため、リベース操作を開始する前にログのバックアップが取得されます。 再ベース化に失敗した場合は、バックアップログデータを使用してキューマネージャーを再起動することができます。 バックアップログは、リベースが成功した後に削除されます。

バックアップログが必要なため、Native HA CRR 構成では、Native HA 構成の少なくとも 2 倍のストレージ容量が必要です。

OpenShift CP4I コンテナー内の IBM MQ のセキュリティーの計画

コンテナー構成で IBM MQ を計画する際のセキュリティーに関する考慮事項。

手順

- 49 ページの『コンテナー内の IBM MQ でのユーザーの認証と許可』
  - 50 ページの『コンテナーでのオペレーティング・システム・ユーザーの使用に関するセキュリティー制約』
- 50 ページの『コンテナー内の IBM MQ へのネットワーク・トラフィックを制限する際の考慮事項』

コンテナー内の IBM MQ でのユーザーの認証と許可

コンテナー内の IBM MQ は、LDAP、相互 TLS、またはカスタム MQ プラグインを使用してユーザーを認証するように構成できます。

IBM MQ Operator では、コンテナー・イメージ内でのオペレーティング・システムのユーザーおよびグループの使用は許可されないことに注意してください。 詳細については、50 ページの『コンテナーでのオペレーティング・システム・ユーザーの使用に関するセキュリティー制約』を参照してください。

LDAP

LDAP ユーザー・リポジトリーを使用するように IBM MQ を構成する方法については、 接続認証: ユーザー・リポジトリー および LDAP 許可を参照してください。

相互 TLS

TLS 証明書 (相互 TLS) を必要とするようにキュー・マネージャーへの着信接続を構成する場合は、証明書の識別名をユーザー名にマップできます。 以下の 2 つのことを行う必要があります。

- SSLPEER を使用して、ユーザー名へのマッピングを作成するようにチャネル認証レコードを構成します。 詳しくは、 MCAUSER ユーザー ID への SSL または TLS 識別名のマッピングを参照してください。
- キュー・マネージャーを構成して、システムに認識されていないユーザー名の権限レコードを定義できるようにします。 詳しくは、 qm.ini ファイルの Service スタンザを参照してください。

JSON Web トークン

JSON Web トークン (JWT) を使用するように IBM MQ を構成する方法については、 認証トークンの処理を参照してください。

カスタム MQ プラグイン

これは高度な手法であり、さらに多くの作業を必要とします。 詳しくは、 カスタム許可サービスの使用を参照してください。

関連タスク

106 ページの『例: 相互 TLS 認証を使用したキュー・マネージャーの構成』

この例では、 IBM MQ Operator を使用して、キュー・マネージャーを OpenShift Container Platform にデプロイします。 相互 TLS は、認証に使用され、TLS 証明書からキュー・マネージャー内の ID にマップされます。

コンテナーでのオペレーティング・システム・ユーザーの使用に関するセキュリティー制約

コンテナーでのオペレーティング・システム・ユーザーの使用は推奨されておらず、 IBM MQ Operator では禁止されています。

マルチテナントのコンテナー環境では、潜在的なセキュリティー問題を防ぐために、通常は以下の例のようなセキュリティー制約が適用されます。

- **コンテナー内での「root」ユーザーの使用の防止**
- **ランダム UID の使用の強制**。 例えば Red Hat OpenShift Container Platform では、`SecurityContextConstraints` のデフォルト (つまり `restricted`) を使用すると、コンテナーごとにランダムなユーザー ID が使用されます。
- **特権エスカレーションの使用の防止**。 IBM MQ on Linux は、特権エスカレーションを使用してユーザーのパスワードを検査します。つまり、「root」ユーザーになるために「setuid」プログラムを使用します。

OpenShift CP4I これらのセキュリティー手段に確実に準拠するために、 IBM MQ Operator は、コンテナー内のオペレーティング・システム・ライブラリーで定義されている ID の使用を許可しません。コンテナー内に `mqm` というユーザー ID やグループは定義されていません。

コンテナー内の IBM MQ へのネットワーク・トラフィックを制限する際の考慮事項

OpenShift Container Platform および Kubernetes で、クラスター内のポッドへのトラフィックを制限するためのネットワーク・ポリシーを定義できます。 このトピックでは、ネットワーク・ポリシーを IBM MQ に適用する方法に関するいくつかの考慮事項について説明します。

キュー・マネージャーへのネットワーク入口の場合、考慮すべきポートがいくつかあります。

- キュー・マネージャー・トラフィック用のポート 1414
- ネイティブ HA 用のポート 9414
- メトリック用のポート 9157
- Web コンソールおよび REST API 用のポート 9443

ネットワーク出口はより複雑です。 考慮すべきネットワーク Egress の例を以下に示します。

- DNS-DNS 名を使用するチャネルまたはその他の構成がある場合
- 他のキュー・マネージャー
- Online Certificate Status Protocol (OCSP) および証明書失効リスト (CRL)-証明書プロバイダーによって決定されます。
- 認証プロバイダー:
  - LDAP
  - IBM MQ Web サーバー用の Open ID Connect またはその他の構成済みログイン・プロバイダー。 これには、 IBM Cloud Pak Keycloak が含まれます。
- トレース・プロバイダー:
  - IBM Instana

  注 : 以前のバージョンの IBM MQ では、 IBM Cloud Pak for Integration Operations Dashboard もトレース・プロバイダーとして使用できました。 ただし、Operations Dashboard は IBM MQ 9.3.3 CD および IBM MQ 9.4.0 LTS で削除されました。

Ingress NetworkPolicy の例

Red Hat OpenShift Container Platform で使用するために、「myqm」というキュー・マネージャーの Ingress を制御するネットワーク・ポリシーの例を以下に示します。

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: myqm
spec:
  podSelector:
    matchLabels:
      app.kubernetes.io/instance: myqm
      app.kubernetes.io/name: ibm-mq
  ingress:
    # Allow access to queue manager listener from anywhere
    - ports:
        - protocol: TCP
          port: 1414
    # Allow access to Native HA port from other instances of the same queue manager
    - from:
        - podSelector:
            matchLabels:
              app.kubernetes.io/instance: myqm
              app.kubernetes.io/name: ibm-mq
      ports:
        - protocol: TCP
          port: 9414
    # Allow access to metrics from monitoring project
    - from:
        - namespaceSelector:
            matchLabels:
              network.openshift.io/policy-group: monitoring
      ports:
        - protocol: TCP
          port: 9157
    # Allow access to web server via Route
    - from:
        - namespaceSelector:
            matchLabels:
              network.openshift.io/policy-group: ingress
      ports:
        - protocol: TCP
          port: 9443
```

FIPS IBM MQ 、コンテナ内のコンプライアンス

起動時に、コンテナ内の IBM MQ は、コンテナを起動するオペレーティング・システムが FIPS に準拠しているかどうかを検出し、（準拠している場合は）自動的に FIPS サポートを設定する。 ここでは、要件と制限について説明します。

Federal Information Processing Standards

米国政府は、データ暗号化など、IT システムおよびセキュリティーに関する技術的助言を行っています。 米国連邦情報・技術局 (NIST) は、IT システムとセキュリティーに関する政府機関です。 NIST は、Federal Information Processing Standards (FIPS)を含む勧告や規格を作成している。

重要な FIPS 標準は FIPS 140-2 、強力な暗号アルゴリズムの使用を要求している。 FIPS 140-2 また、パケットを転送中の改ざんから保護するために使用されるハッシュアルゴリズムの要件も規定している。

IBM MQ が設定されている場合は、 をサポートします。 FIPS 140-2

注 : AIX®, Linux, and Windows では、 IBM MQ は、 IBM Crypto for C (ICC) 暗号モジュールを通して、 FIPS 140-2 コンプライアンスを提供する。 このモジュールの証明書は「履歴」ステータスに移動されました。 顧客は、 IBM Crypto for C (ICC) の証明書を閲覧し、NIST が提供する助言に注意する必要がある。

V 9.4.2 その FIPS 140-3 暗号モジュール内 IBM Semeru Runtime 2024 年 8 月に NIST によって承認されました。 IBM MQ 9.4.2 Java 8 および の で、TLS を使用した および クライアント接続の処理をサポートする。 IBM Semeru Runtime 11+ FIPS 140-3 IBM MQ classes for JMS IBM MQ classes for Java FIPS 140-3 モジュールに関連する NIST 認証は https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755 で見ることができる。 FIPS 140-2 プロバイダーはデフォルトのプロ

ファイルのままです。 IBM MQ 9.4.2 はデフォルトの動作を変更しないが、 を使って接続を設定することができる。 FIPS 140-3

コンテナの IBM MQ 、 IBM MQ Operator 3.2.0 とキュー・マネージャー・コンテナ・イメージ 9.4.0.0 以降は UBI 9 をベースにしている。 FIPS 140-3 IBM MQ in Containers への準拠は現在保留中である。

要件

クラスタのセットアップやその他の考慮事項に関する要件については、 FIPS Wall を参照のこと： FIPS コンプライアンスに対する現在の IBM アプローチ。

IBM MQ コンテナ内では、 コンプライアンス・モードで実行できる。 FIPS 140-2 起動中、コンテナ内の IBM MQ は、コンテナを起動するホスト・オペレーティング・システムが FIPS に準拠しているかどうかを検出する。 ホストのオペレーティングシステムが FIPS に準拠しており、プライベートキーと証明書が提供されている場合、 IBM MQ コンテナは、キューマネージャ、 IBM MQ Web サーバ、および Native High Availability 配置のノード間のデータ転送を、 FIPS 準拠モードで実行するように構成します。

IBM MQ Operator を使用してキュー・マネージャーをデプロイする場合、オペレーターは終了タイプ **Passthrough** の経路を作成します。 これは、ルーターが TLS 終端を提供することなく、トラフィックが宛先に直接送信されることを意味します。 この場合、 IBM MQ キュー・マネージャーと IBM MQ ウェブ・サーバーが宛先となり、すでに FIPS 準拠のセキュアな通信を提供している。

主な要件:

1. キュー・マネージャーおよび Web サーバーにシークレットで提供される秘密鍵および証明書。これにより、外部クライアントはキュー・マネージャーおよび Web サーバーに安全に接続できます。
2. ネイティブ高可用性構成内の異なるノード間でのデータ転送用の秘密鍵および証明書。

制限

FIPS 、 IBM MQ をコンテナで配備する場合、以下を考慮する：

- コンテナー内の IBM MQ は、メトリックを収集するためのエンドポイントを提供します。 現在、このエンドポイントは HTTP。 メトリクスのエンドポイントをオフにすることで、 IBM MQ FIPS の残りの部分を準拠させることができます。
- コンテナー内の IBM MQ は、カスタム・イメージのオーバーライドを許可します。 つまり、 IBM MQ コンテナー・イメージをベース・イメージとして使用してカスタム・イメージをビルドできます。 FIPS このようなカスタマイズされた画像には、コンプライアンスが適用されない場合があります。
- IBM Instana を使ったメッセージ追跡の場合、 IBM MQ と IBM Instana の間の通信は、 HTTP または HTTPS で、 FIPS には準拠しない。
- IBM MQ Operator IBM ID およびアクセス管理（IAM）/Zen サービスへのアクセスは、 に準拠していません。 FIPS

FIPS への準拠が検出され、 *FIPS* サポートが自動的に設定される方法

コンテナを起動するオペレーティング・システムが FIPS に準拠している場合、 FIPS サポートが自動的に設定される。

注 : AIX, Linux, and Windows では、 IBM MQ は、 IBM Crypto for C (ICC) 暗号モジュールを通して、 FIPS 140-2 コンプライアンスを提供する。 このモジュールの証明書は「履歴」ステータスに移動されました。 顧客は、 IBM Crypto for C (ICC) の証明書を閲覧し、NIST が提供する助言に注意する必要がある。

V 9.4.2 その FIPS 140-3 暗号モジュール内 IBM Semeru Runtime 2024 年 8 月に NIST によって承認されました。 IBM MQ 9.4.2 Java 8 および の で、TLS を使用した および クライアント接続の処理をサポートする。 IBM Semeru Runtime 11+ FIPS 140-3 IBM MQ classes for JMS IBM MQ classes for Java FIPS 140-3 モジュールに関連する NIST 認証は https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755 で見ることができる。 FIPS 140-2 プロバイダーはデフォルトのプロファイルのままです。 IBM MQ 9.4.2 はデフォルトの動作を変更しないが、 を使って接続を設定することができる。 FIPS 140-3

コンテナの IBM MQ 、 IBM MQ Operator 3.2.0 とキュー・マネージャー・コンテナ・イメージ 9.4.0.0 以降は UBI 9 をベースにしている。 FIPS 140-3 IBM MQ in Containers への準拠は現在保留中である。

起動中、コンテナ内の IBM MQ は、コンテナを起動するオペレーティング・システムが FIPS に準拠しているかどうかを検出する。 その場合、以下のアクションが自動的に実行されます。

キュー・マネージャー
ホストオペレーティングシステムが FIPS に準拠しており、秘密鍵と証明書が提供されている場合、キューマネージャ属性 **SSLFIPS** は YES に設定される。 そうでない場合、 **SSLFIPS** 属性は NO に設定される。

IBM MQ Web サーバー
IBM MQ ウェブサーバーは、 IBM MQ を管理するための HTTP / HTTPS インターフェースを提供します。 ホスト・オペレーティング・システムが FIPS に準拠している場合、ウェブ・サーバーが FIPS に準拠した暗号を使用するように JVM オプションが更新されます。 FIPS を使用できるようにするには、コンテナーの開始時に秘密鍵と証明書を指定する必要があります。

ネイティブ HA
ノード間で複製されるデータのセキュリティーは、qm.ini ファイルの **NativeHALocalInstance** スタンザによって制御されます。 以下に例を示します。

```
NativeHALocalInstance:
    KeyRepository=/run/runmqserver/ha/tls/key.kdb
    CertificateLabel=NHAQM
    CipherSpec=ECDHE_RSA_AES_256_GCM_SHA384
```

FIPS が有効な場合、 **SSLFipsRequired** 属性がスタンザに追加され、値は Yes に設定される：

```
NativeHALocalInstance:
    KeyRepository=/run/runmqserver/ha/tls/key.kdb
    CertificateLabel=NHAQM
    CipherSpec=ECDHE_RSA_AES_256_GCM_SHA384
    SSLFipsRequired=Yes
```

コンテナが FIPS サポートのない OpenShift クラスタで実行されている場合、キュー・マネージャ、 IBM MQ Web サーバー、および Native HA コンポーネントは、 FIPS サポートが自動的に有効にならない。 現在、FIPS の OpenShift プラットフォームでサポートされているのは、x86-64 アーキテクチャのみである。 Power および Linux on IBM Z® アーキテクチャの場合、 OpenShift は FIPS をサポートしない。 これらのアーキテクチャ用の IBM MQ コンポーネントで FIPS サポートを明示的に有効にするには、キューマネージャの YAML で *MQ_ENABLE_FIPS* 環境変数を true に設定する。 以下の YAML スニペットは、*MQ_ENABLE_FIPS* 環境変数の使用法について記述している：

```
template:
    pod:
      containers:
        - env:
            - name: MQ_ENABLE_FIPS
              value: "true"
          name: qmgr
```

コンテナ内の *IBM MQ* の自動 *FIPS* モードのオーバーライド

環境変数 *MQ_ENABLE_FIPS* を使用して、コンテナ内の IBM MQ コンポーネントの FIPS モードを明示的に有効または無効にする。

始める前に

注 : AIX, Linux, and Windows では、 IBM MQ は、 IBM Crypto for C (ICC) 暗号モジュールを通して、 FIPS 140-2 コンプライアンスを提供する。 このモジュールの証明書は「履歴」ステータスに移動されました。 顧客は、 IBM Crypto for C (ICC) の証明書を閲覧し、NIST が提供する助言に注意する必要がある。

V 9.4.2 その FIPS 140-3 暗号モジュール内 IBM Semeru Runtime 2024 年 8 月に NIST によって承認されました。 IBM MQ 9.4.2 Java 8 および の で、TLS を使用した および クライアント接続の処理をサポートする。 IBM Semeru Runtime 11+ FIPS 140-3 IBM MQ classes for JMS IBM MQ classes for Java FIPS 140-3 モジュールに関連する NIST 認証は https://csrc.nist.gov/projects/cryptographic-module-

validation-program/certificate/4755 で見ることができる。 FIPS 140-2 プロバイダーはデフォルトのプロファイルのままです。 IBM MQ 9.4.2 はデフォルトの動作を変更しないが、 を使って接続を設定することができる。 FIPS 140-3

コンテナの IBM MQ 、 IBM MQ Operator 3.2.0 とキュー・マネージャー・コンテナ・イメージ 9.4.0.0 以降は UBI 9 をベースにしている。 FIPS 140-3 IBM MQ in Containers への準拠は現在保留中である。

このタスクについて

MQ_ENABLE_FIPS は、以下の 3 つの値をサポートします。

auto
これがデフォルト値です。

ホスト OS が FIPS 、すべてのコンポーネント（キューマネージャ、 IBM MQ Web サーバ、Native HA）が FIPS 。

ホスト OS が FIPS 、すべてのコンポーネントが FIPS 。

true
この値は、コンテナ内の選択されたコンポーネントの FIPS をオンにする。

コンテナ内の IBM MQ が、 FIPS に準拠していないホストオペレーティングシステム上で実行されている場合でも、キューマネージャ属性 **SSLFIPS** は YES に設定される。 つまり、 IBM MQ キュー・マネージャー、ウェブ・サーバー、およびネイティブ HA が FIPS に準拠しているが、コンテナのオペレーティング・システムが準拠していない場合である。

false
この値は、 FIPS コンプライアンスをオフにする。

コンテナ内の IBM MQ が FIPS 準拠のホストマシン上で実行されている場合でも、キューマネージャ属性 **SSLFIPS** は NO に設定される。 しかし、 IBM MQ 、秘密鍵と証明書が提供されれば、接続は保護される。

IBM MQ Web サーバーの JVM オプションは更新されません。 しかし、IBM MQ Web サーバーは、秘密鍵と証明書が提供された場合でも、 HTTPS エンドポイントを実行する。

ネイティブ HA のデータレプリケーションは、 FIPS 暗号を使用しない。

例

キューマネージャコンポーネントの TLS と FIPS の有効化について記述した、キューマネージャ YAML のサンプルです。

注 : この例では、IBM MQ Advanced ライセンスを使用していますが、IBM MQ ライセンスを使用することもできます。 詳しくは、161 ページの『を使用して、 IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  namespace: ibm-mq-fips
  name: exampleqm
spec:
  license:
    accept: true
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    storage:
      queueManager:
        type: ephemeral
  template:
    pod:
      containers:
        - env:
            - name: MQ_ENABLE_FIPS
              value: "true"
          name: qmgr
  version: 9.4.3.0-r1
  web:
```

```
    enabled: false
  pki:
    keys:
      - name: ibm-mq-tls-certs
        secret:
          secretName: ibm-mq-tls-secret
          items:
          - tls.key
          - tls.crt
```

コンテナー内の IBM MQ のスケーラビリティーとパフォーマンスの計画

ほとんどの場合、コンテナー内の IBM MQ のスケーリングとパフォーマンスは、IBM MQ for Multiplatforms と同じです。 ただし、コンテナー・プラットフォームによって課される可能性がある追加の制限がいくつかあります。

このタスクについて

コンテナー内の IBM MQ のスケーラビリティーとパフォーマンスを計画する場合は、以下のオプションを考慮してください。

手順

- **スレッドおよびプロセスの数を制限します。**

  IBM MQ はスレッドを使用して並行性を管理します。 Linux では、スレッドはプロセスとして実装されるため、コンテナー・プラットフォームまたはオペレーティング・システムによって課される、プロセスの最大数に関する制限を検出することができます。 Red Hat OpenShift Container Platform 4.11 以降では、コンテナーごとに 4096 プロセスというデフォルトの制限があります。 これは大多数のシナリオに適していますが、キュー・マネージャーのクライアント接続の数に影響を与える可能性があります。

  Kubernetes のプロセス制限は、クラスター管理者が kubelet 構成設定 **podPidsLimit** を使用して構成できます。 Kubernetes 資料の プロセス ID の制限と予約 を参照してください。 Red Hat OpenShift Container Platform では、 **ContainerRuntimeConfig** カスタム・リソースを作成して CRI-O パラメーターを編集することもできます。

  IBM MQ 構成では、キュー・マネージャーのクライアント接続の最大数を設定することもできます。 個々のサーバー接続チャネルに制限を適用する場合は サーバー接続チャネルの制限 を、キュー・マネージャー全体に制限を適用する場合は MAXCHANNELS INI 属性 を参照してください。

- **ボリューム数を制限します。**

  クラウド・システムおよびコンテナー・システムでは、Network Attached Storage ボリュームが一般的に使用されます。 Linux ノードに接続できるボリュームの数には制限があります。 例えば、 AWS EC2 は、VM 当たりのボリューム数を 30 以下に制限します。 Red Hat OpenShift Container Platform 類似した制限がある (Microsoft Azure および Google Cloud Platform と同様)。

  ネイティブ HA キュー・マネージャーは、3 つのインスタンスごとに 1 つのボリュームを必要とし、インスタンスをノード間に分散させます。 ただし、インスタンスごとに 3 つのボリューム (キュー・マネージャー・データ、リカバリー・ログ、および永続データ) を使用するようにキュー・マネージャーを構成できます。

- **IBM MQ のスケーリング手法を使用します。**

  少数の大規模なキュー・マネージャーではなく、IBM MQ 均一クラスターなどの IBM MQ スケーリング手法を使用して、同じ構成で複数のキュー・マネージャーを実行することをお勧めします。 これにより、単一コンテナーの再始動 (例えば、コンテナー・プラットフォームの保守の一環として) の影響が軽減されるという利点が追加されます。

コンテナー内の IBM MQ 用の環境の準備、インストール、およびアップグレード

IBM MQ 用に環境を準備するために、さまざまなタスクを実行します。

このタスクについて

IBM MQ Operator を使用している場合は、オペレーターをインストールして Red Hat OpenShift Container Platform クラスターを準備します。 56 ページの『IBM MQ Operator のインストールおよびアップグレード』 を参照してください。

そうでない場合は、独自のコンテナー・イメージをビルドしてコンテナー環境を準備します。 87 ページの『独自のコンテナー・イメージのビルドによる IBM MQ の準備』を参照

IBM MQ Operator のインストールおよびアップグレード

IBM MQ Operator をインストール、アンインストール、およびアップグレードするには、さまざまなタスクを実行します。

このタスクについて

Red Hat OpenShift Container Platform での IBM MQ Operator のインストールおよびアップグレードを開始するには、以下のトピックを参照してください。

手順

- 56 ページの『IBM MQ Operator の依存関係』
- 56 ページの『IBM MQ Operator に必要なクラスター・スコープ許可』
- 57 ページの『イメージ署名の検証』
- 57 ページの『IBM MQ Operator のインストール』
- 69 ページの『IBM MQ Operator とキュー・マネージャーのアップグレード』
- 84 ページの『IBM MQ Operator のアンインストール』

OpenShift CP4I IBM MQ Operator の依存関係

IBM MQ Operator のインストール時に他のオペレーターが自動的にインストールされることはありません。

IBM Licensing Operator は、ライセンス使用量を追跡するために別個にインストールする必要があります。 IBM Cloud Pak for Integration 資料の License Service を参照してください。

IBM Cloud Pak for Integration ライセンスを使用して `QueueManager` を作成する場合、 Keycloak の IBM Cloud Pak for Integration インスタンスでシングル・サインオンを使用するかどうかを選択できます。 IBM Cloud Pak for Integration ライセンスでは、 Keycloak の使用はデフォルトで有効になっていますが、インストールされていない場合、 `QueueManager` は、正しい依存関係がインストールされるまで「ブロック」状態になります。 依存関係について詳しくは、 57 ページの『IBM MQ Operator のインストール』 を参照してください。

OpenShift CP4I IBM MQ Operator に必要なクラスター・スコープ許可

IBM MQ Operator 、ストレージ・クラスとクラスタのバージョン情報を読み取るには、常にクラスタ・スコープのパーミッションが必要です。

IBM MQ Operator は 2 つの異なるスコープに取り付けることができる：

1. **クラスタ・スコープとは**、`ClusterRole` および `ClusterRoleBinding` リソースを使用するパーミッションがオペレータに割り当てられることを意味します。 これらによって、オペレータはすべての名前空間にわたって行動できるようになる。
2. **名前空間スコープとは**、 `Role` と `RoleBinding`、指定された名前空間内でのみパーミッションが与えられることを意味します。 また、以下の読み取り専用パーミッションについては、 `ClusterRole` 、 `ClusterRoleBinding` ：

- クラスター・バージョンを読み取る権限。 オペレーターがクラスター環境に関する問題をフィードバックすることが可能になります。

  - API グループ: **config.openshift.io**
  - リソース: **clusterversions**
  - verbs: **get, list, watch**
- クラスター上のストレージ・クラスを読み取る権限。 これによりオペレーターは、コンテナ内の選択されたストレージクラスに関する問題をフィードバックすることができます。
  - API グループ: **storage.k8s.io**
  - リソース: **storageclasses**
  - verbs: **get, list**

OpenShift CP4I イメージ署名の検証

IBM MQ Operator および IBM MQ のキュー・マネージャー・コンテナー・イメージはデジタル署名されています。

このタスクについて

デジタル署名により、コンテンツの利用者は、ダウンロードしたものが本物 (期待されるソースからのもの) であり、整合性があること (期待されるもの) を確認することができます。

手順

- IBM MQ Operator および IBM MQ キュー・マネージャー・コンテナー・イメージの署名を検査します。
  - 3.4.0 以降の 'IBM MQ Operator または 9.4.1.0-r2 以降の 'IBM MQ キューマネージャコンテナイメージについては、 'IBM Cloud Pak for Integration(CP4I)16.1.1 ドキュメントの「イメージ署名の検証」を参照してください。
  - 3.2.x または 3.3.x の 'IBM MQ Operator、あるいはバージョン 9.4.0.0-r1 から 9.4.1.0-r1 までの 'IBM MQ キューマネージャコンテナイメージについては、 'IBM Cloud Pak for Integration(CP4I)16.1.0 ドキュメントの「イメージ署名の検証」を参照してください。

OpenShift CP4I IBM MQ Operator のインストール

IBM MQ Operator は、 OpenShift コンソールまたはコマンド・ライン・インターフェース (CLI) を使用して Red Hat OpenShift にインストールできます。

始める前に

重要 :

- このトピックでは、スタンドアロンで使用するための IBM MQ Operator のインストールについて **のみ**説明します。 1 つ以上のキュー・マネージャーで IBM Cloud Pak for Integration または Keycloak SSO を使用する場合は、 65 ページの『CP4I で使用するための IBM MQ Operator のインストール』を参照してください。
- IBM MQ Operator をインストールする前に、 デプロイメントの構造化 に関するガイダンスを確認してください。

インストールができるだけスムーズに行われるようにするには、インストールを開始する前に、すべての前提条件と要件を理解しておく必要があります。 27 ページの『コンテナー内の IBM MQ の計画』 を参照してください。

このタスクについて

以下のステップは、 IBM MQ Operator をインストールするための標準的な作業フローを示しています。

1. Red Hat OpenShift Container Platform をインストールします。
2. ストレージを構成する。

3. ミラー・イメージ (エア・ギャップのみ)。
4. IBM MQ Operator カタログを追加します。
5. IBM MQ Operator をインストールします。
6. ライセンス・キー・シークレットを作成します (オンライン・インストールのみ)。
7. License Service をデプロイします。
8. キュー・マネージャーをデプロイします。

手順

1. Red Hat OpenShift Container Platform をインストールします。

   OpenShift をインストールする詳細なステップについては、 Red Hat ソフトウェア 4.6 以降のインストールを参照してください。

   重要 : サポートされているバージョンの OpenShift Container Platform がインストールされていることを確認してください。 例えば、 IBM MQ Operator 3.6 以降を使用するには、 OpenShift Container Platform 4.12 以降をインストールする必要があります。 詳細については、IBM Cloud Pak および Red Hat OpenShift Container Platform 互換性 を参照してください。

   Red Hat OpenShift Container Platform CLI を使用するすべてのステップで、 `oc login` を使用して OpenShift クラスターにログインする必要があります。 CLI をインストールするには、OpenShift CLI の概要を参照してください。

   OpenShift をインストールした後、 ライセンス・キー・シークレットの作成で作成した IBM ライセンス・キーを使用して、コンテナー・ソフトウェアを検証し、アクセス権限を取得できます。
2. ストレージを構成します。

   Red Hat OpenShift Container Platform でストレージ・クラスを定義し、サイジング要件を満たすようにストレージ構成を設定する必要があります。

   重要 : IBM MQ 単一インスタンスおよびネイティブ HA キュー・マネージャーは RWO アクセス・モードを使用できますが、複数インスタンス・キュー・マネージャーは、 38 ページの『IBM MQ Operator のストレージの計画』で説明されているように RWX を必要とします。 IBM MQ 複数インスタンス・キュー・マネージャーには、特定のファイル・システム特性が必要です。これは、 IBM MQ の共有ファイル・システムのテストの手順を使用して検証できます。

   既知の準拠ファイル・システムと非準拠ファイル・システムのリスト、およびその他の制限や制約事項については、 IBM MQ ファイル・システムのテスト・ステートメントを参照してください。

   推奨されるストレージ・プロバイダーについては、 CP4I 「ストレージに関する考慮事項」 ページを参照してください。
3. イメージをミラーリングします (エア・ギャップのみ)。

   クラスタが制限された（エアギャップ）ネットワーク環境にある場合は、 IBM MQ のイメージをミラーリングする必要があります。

   重要 : バージョン 3.5.0 より、IBM MQ CASE が変更され、単一のキューマネージャーバージョンだけを含むようになりました。 イメージのミラーリングプロセスはかなり高速に実行されますが、複数回実行する必要があるかもしれません。インストールするキューマネージャの各バージョンごとに 1 回ずつ実行する必要があります。

   詳細については、295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照してください。

   必要なオペレータとキューマネージャのバージョンが含まれている IBM MQ CASE のバージョンを調べ、それぞれに対してエアギャップミラープロセスを繰り返す必要があります。 IBM MQ CASE ルックアップテーブルで、バージョン対応表をご確認ください。

例えば、下記にリンクされた指示に従ってイメージをミラーリングする場合、最新のオペレータと対応するキュー・マネージャの単一バージョンを、以下の値を使用してインストールします

```
export OPERATOR_PACKAGE_NAME=ibm-mq
export OPERATOR_VERSION=3.6.0
```

エアギャップのインストールとアップグレードに使用される IBM MQ CASE の内容については、'295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照のこと。

ミラー・イメージを作成するには、 エアギャップ・クラスターのミラーリング・イメージを参照してください。

4. IBM MQ Operator カタログ・ソースを追加します。

   オペレーターをクラスターで使用できるようにするカタログ・ソースを追加します。 60 ページの『IBM MQ Operator カタログ・ソースの追加』 を参照してください。

5. IBM MQ Operator のインストール

   以下の 2 つのオプションのいずれかを選択します (コンソールを使用するか、CLI を使用します)。

   - オプション 1: OpenShift コンソールを使用して IBM MQ Operator をインストールします。
   - オプション 2: OpenShift CLI を使用して IBM MQ Operator をインストールします。

6. 使用権キー・シークレットを作成します (オンライン・インストールのみ)。

   IBM MQ Operator は、ライセンス資格検査を実行するコンテナー・レジストリーからプルされたキュー・マネージャー・イメージをデプロイします。 この検査には、 `docker-registry` プル・シークレットに保管されているライセンス・キーが必要です。 キュー・マネージャーをインストールする名前空間にまだライセンス・キーがない場合は、以下の手順に従ってライセンス・キーを取得し、プル・シークレットを作成します。

   注 : IBM MQ Advanced for Developers (非警告) キュー・マネージャーのみをデプロイする場合は、ライセンス・キーは必要ありません。

   OpenShift コンソールまたは CLI を使用して、使用権キー・シークレットを作成できます。 以下の例では、CLI を使用します。

   a. IBM ID に割り当てられているライセンス・キーを取得します。 ライセンスが付与されているソフトウェアに関連付けられた IBM ID とパスワードを使用して、 MyIBM Container Software Library にログインします。

   b.「**使用権キー (Entitlement keys)**」セクションで「**キーのコピー (Copy key)**」を選択して、使用権キーをクリップボードにコピーします。

   c. OpenShift CLI から以下のコマンドを実行して、 `ibm-entitlement-key` という名前のイメージ・プル・シークレットを作成します。

   ```
   oc create secret docker-registry ibm-entitlement-key \
   --docker-server=cp.icr.io \
   --docker-username=cp \
   --docker-password=entitlement_key \
   --docker-email=user_email
   \--namespace=namespace
   ```

   ここで、 *entitlement_key* はステップ b でコピーしたライセンス・キー、 *user_email* はライセンスが付与されているソフトウェアに関連付けられた IBM ID、 *namespace* は IBM MQ Operator をインストールした名前空間です。

7. License Service をデプロイします。

   これは、キュー・マネージャーのライセンス使用状況をモニターするために必要です。 License Service の手順に従います。

8. キュー・マネージャーをデプロイします。

   サンプルの "クイック・スタート" ・キュー・マネージャーのデプロイ手順については、 98 ページの『IBM MQ Operator を使用した単純なキュー・マネージャーのデプロイ』 を参照してください。

関連タスク

84 ページの『IBM MQ Operator のアンインストール』
Red Hat OpenShift コンソールまたは CLI を使用して、Red Hat OpenShift から IBM MQ Operator をアンインストールできます。

OpenShift *IBM MQ Operator カタログ・ソースの追加*

IBM MQ Operator カタログ・ソースを OpenShift クラスターに追加して、 IBM MQ Operator をインストールできるようにします。 このタスクは、アップグレードを完了する前にカタログ・ソースのフィックスパックを適用する場合にも必要です。

このタスクについて

オペレーター・カタログは、 Red Hat OpenShift Container Platform クラスターの API を拡張して IBM ソフトウェア製品を使用可能にするために使用できるオペレーターの索引です。

以下のカタログ・ソースが使用可能です。

オプション 1: IBM MQ Operator の特定のカタログ・ソース。

特定の IBM MQ Operator カタログ・ソースを使用することにより、クラスター上でのソフトウェアのバージョン管理と、アップグレードがいつ行われるかを完全に制御できます。 新しい IBM MQ Operator バージョンが OpenShift クラスターで使用可能になるのは、カタログ・ソースを更新した後 **のみ** です。 このプロセスにより、アップグレードを手動で制御できるため、オペレーターの **Update approval** 設定に 「手動」 オプションを使用する必要はありません。 「**手動**」 オプションを選択すると、可能なすべてのアップグレードが同時に強制的に実行され、アップグレードがブロックされる可能性があるため、 「**自動**」 オプションのみを使用してください。 詳しくは、 Red Hat OpenShift コンソールを使用したオペレーターのインストールの「承認戦略による自動更新の制限」セクションを参照してください。

アップグレードを実行していて、新しいバージョンの IBM MQ Operator カタログ・ソースを追加する必要がある場合は、このオプションを選択します。

このオプションを使用するには、「 オプション 1: IBM MQ Operator の特定のカタログ・ソースの追加」にスキップします。

オプション 2: IBM オペレーター・カタログ。

このオプションを使用すると、新しいオペレーター・バージョンが使用可能になり、ユーザーの介入を **必要とせずに** 適用されます。 そのため、このオプションは、 IBM MQ Operator の **自動** アップグレードが必要で、確定的インストールが必要ないオンライン・インストールに **のみ** 使用してください。

注 : このオプションは PoC (概念検証) 環境には役立ちますが、 **実稼働環境には適していません。**

このオプションを使用するには、「 オプション 2: IBM オペレーター・カタログの追加」にスキップします。

手順

- **オプション 1: IBM MQ Operator の特定のカタログ・ソースを追加します。**

  このタスクでは、 57 ページの『IBM MQ Operator のインストール』の最初の 3 つのステップが完了していることを前提としています。

  このタスクは、クラスター管理者が実行する必要があり、CLI を使用して実行する必要があります。

  a) アップグレードのみ: アップグレード前にカタログ・ソースのフィックスパックを適用する場合は、以下の手順を実行します。

  - オペレーターが正しく実行されていることを確認します。
  - 手動による承認が必要な保留中の IBM MQ Operator 更新がある場合は、この手順を開始する前にそれらを承認してください。 詳しくは、 Red Hat OpenShift コンソールを使用したオペレーターのインストールの「承認戦略による自動更新の制限」を参照してください。

b) まだインストールしていない場合、または更新が必要な場合は、GitHub から IBM カタログ管理プラグイン (バージョン 1.6.0 以降) をダウンロードします。

このプラグインを使用すると、クラスターに対して **oc ibm-pak** コマンドを実行できます。

c) **oc login** コマンドとユーザー資格情報を使用して、クラスターにログインします。

```
oc login openshift_url -u username -p password -n namespace
```

d) IBM MQ Operator の以下の環境変数をエクスポートします。

```
export OPERATOR_PACKAGE_NAME=ibm-mq
export OPERATOR_VERSION=CHOSEN_OPERATOR_VERSION
export ARCH=ARCHITECTURE
```

説明:

- 建築 is the architecture of the system on which you are deploying the IBM MQ Operator, and has a value of amd64, ppc64le, or s390x.
- *CHOSEN_OPERATOR_VERSION* は、IBM MQ Operator の必須バージョンです。 The current CD version is 3.6.0. The current SC2 version is 3.2.13.

重要 : IBM オペレーター・カタログから IBM MQ Operator の特定のカタログ・ソースに移動する場合は、 OPERATOR_VERSION を IBM MQ Operator のデプロイメントのバージョンに設定します。

e) IBM MQ Operator 用のファイルをダウンロードします。

注 : エアギャップ ・インストールを実行する場合は、 「IBM MQ Operator のインストール」の「ミラー・イメージ」ステップの完了後に必要なファイルが既にあるはずです。この場合は、ステップ 61 ページの『8』 「IBM MQ Operator カタログ・ソースをクラスターに適用する」にスキップできます。

```
oc ibm-pak get ${OPERATOR_PACKAGE_NAME} --version ${OPERATOR_VERSION}
```

f) IBM MQ Operator に必要なカタログ・ソースを生成します。

```
oc ibm-pak generate mirror-manifests ${OPERATOR_PACKAGE_NAME} icr.io --version $
{OPERATOR_VERSION}
```

g) オプション: カタログ・ソースを生成し、別のディレクトリーに保存します。

a. カタログ・ソースを取得します。

```
cat ~/.ibm-pak/data/mirror/${OPERATOR_PACKAGE_NAME}/${OPERATOR_VERSION}/catalog-
sources.yaml
```

b. (オプション) ファイル・ブラウザーでディレクトリーにナビゲートして、これらの成果物を再使用またはパイプライン用に保持できるファイルにコピーします。

h) IBM MQ Operator カタログ・ソースをクラスターに適用します。

```
oc apply -f ~/.ibm-pak/data/mirror/${OPERATOR_PACKAGE_NAME}/${OPERATOR_VERSION}/catalog-
sources.yaml
```

i) openshift-marketplace 名前空間に IBM MQ Operator カタログ・ソースが作成されたことを確認します。

```
oc get catalogsource -n openshift-marketplace
```

出力例:

```
oc get catalogsource -n openshift-marketplace

NAME                           DISPLAY               TYPE   PUBLISHER   AGE
ibmmq-operator-catalogsource   ibm-mq-3.1.3          grpc   IBM         23h
```

これで、 IBM MQ Operator のインストールのステップ 5 を実行する準備ができました。

- **オプション 2: IBM オペレーター・カタログを追加します。**

重要 : IBM オペレーター・カタログ **は、** IBM MQ Operator の **自動** アップグレードが必要で、確定的インストールが不要なオンライン・インストールにのみ使用してください。 このオプションは PoC (概念検証) 環境には役立ちますが、 **実稼働環境には適していません**。

IBM オペレーター・カタログは、 Red Hat OpenShift Container Platform クラスターの API を拡張して IBM ソフトウェア製品を使用可能にするために使用できるオペレーターの索引です。 カタログ・ソースを OpenShift クラスターに追加すると、インストール可能なオペレーターのリストに IBM オペレーターが追加されます。

このタスクでは、 57 ページの『IBM MQ Operator のインストール』の最初の 3 つのステップが完了していることを前提としています。

このタスクは、CLI または OpenShift Web コンソールを使用して実行できます。

CLI の使用

1. IBM オペレーター用の以下のリソース定義を、ご使用のコンピューター上のローカル・ファイルにコピーします。

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: ibm-operator-catalog
  namespace: openshift-marketplace
spec:
  displayName: IBM Operator Catalog
  publisher: IBM
  sourceType: grpc
  image: icr.io/cpopen/ibm-operator-catalog:latest
  updateStrategy:
    registryPoll:
      interval: 45m
```

2. 次のコマンドを実行します。 *filename*.yaml を、前のステップで作成したファイルの名前に置き換えます。

```
oc apply -f filename.yaml
```

OpenShift Web コンソールの使用

1. OpenShift クラスター管理者の資格情報を使用して OpenShift Web コンソールにログインします。
2. バナーで、プラス記号 (「+」) をクリックします。 「**YAML のインポート**」 ダイアログ・ボックスを開くためのアイコン。

   注 : 「**プロジェクト**」の値を選択する必要はありません。 次のステップの YAML コードには、 `metadata:namespace` の正しい値が既に含まれています。 これにより、カタログ・ソースが正しいプロジェクト (名前空間) にインストールされます。
3. 以下のリソース定義をダイアログ・ボックスに貼り付けます。

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: ibm-operator-catalog
  namespace: openshift-marketplace
spec:
  displayName: IBM Operator Catalog
  image: 'icr.io/cpopen/ibm-operator-catalog:latest'
  publisher: IBM
  sourceType: grpc
  updateStrategy:
    registryPoll:
      interval: 45m
```

4. 「**作成**」 をクリックします。

これで、 IBM MQ Operator のインストールのステップ 5 を実行する準備ができました。

OpenShift CP4I *OpenShift コンソールを使用した IBM MQ Operator のインストール*

IBM MQ Operator は、 OperatorHub を使用して Red Hat OpenShift にインストールできます。

始める前に

このタスクは、 57 ページの『IBM MQ Operator のインストール』のステップ 1 から 4 を完了していることを前提としています。

手順

1. Red Hat OpenShift クラスター・コンソールにログインします。
2. ナビゲーション・ペインで、「**オペレーター (Operators)**」 > 「**OperatorHub**」をクリックします。
   「OperatorHub」ページが表示されます。
3. 「**All Items**」フィールドで、「IBM MQ」と入力します。
   IBM MQ カタログ・エントリーが表示されます。
4. 「**IBM MQ**」を選択します。
   「IBM MQ」ウィンドウが表示されます。
5. 「**インストール**」をクリックします。
   「Install Operator」ページが表示されます。
6. 以下の値を入力します。
   a) 「**チャネル**」 を、選択したバージョンに設定します。

      35 ページの『IBM MQ Operator のバージョン・サポート』を確認して、選択するオペレーター・チャネルを判別します。

   b) 「**インストール・モード**」 を「クラスター上の特定の名前空間」(次のステップで作成できます) またはクラスター全体の有効範囲のいずれかに設定します。

      異なるバージョンのオペレーターを異なる名前空間にインストールすると問題が発生する可能性があるため、クラスター全体の有効範囲を選択することをお勧めします。 オペレーターは、コントロール・プレーンの拡張機能となるように設計されています。

   c) オプション: 「クラスター上の特定の名前空間」を選択した場合は、 「**名前空間**」 を、オペレーターのインストール先のプロジェクト (名前空間) の値に設定します。

      注 : コンソールを使用してオペレーターをインストールする場合は、既存の名前空間を使用するか、オペレーターによって提供されるデフォルトの名前空間を使用するか、新しい名前空間を作成することができます。 このフォームから新しい名前空間を作成する場合は、以下のようにします。ナビゲーション・ペインで **ホーム** > 「**プロジェクト**」をクリックし、 「**プロジェクトの作成**」を選択し、作成するプロジェクトの **名前** (名前空間) を指定して、 「**作成**」をクリックします。

   d) 「**承認戦略**」 を「自動」に設定します。
7. 「**インストール**」 をクリックし、オペレーターがインストールするまで待ちます。

   インストールが完了すると、確認が表示されます。

   インストールを検証するには、 **オペレーター** > 「**インストール済みオペレーター (Installed Operators)**」にナビゲートし、 「**プロジェクト (Projects)**」 ドロップダウン・リストからプロジェクトを選択します。 インストールが完了すると、オペレーターの状況が「成功」に変わります。

次のタスク

これで、 ライセンス・キー・シークレットを作成 する準備ができました (57 ページの『IBM MQ Operator のインストール』のステップ 6)。

OpenShift CP4I *Red Hat OpenShift CLI を使用した IBM MQ Operator のインストール*

IBM MQ Operator は、コマンド・ライン・インターフェース (CLI) を使用して Red Hat OpenShift にインストールできます。

始める前に

このタスクは、57 ページの『IBM MQ Operator のインストール』のステップ 1 から 4 を完了していることを前提としています。

手順

1. **oc login** を使用して、Red Hat OpenShift コマンド・ライン・インターフェース (CLI) にログインします。
2. オプション: IBM MQ Operator に使用する名前空間を作成します。

   IBM MQ Operator は、単一の名前空間またはすべての名前空間を範囲としてインストールできます。 このステップは、まだ存在しない特定の名前空間にインストールする場合にのみ必要です。

   CLI で新しい名前空間を作成するには、以下のコマンドを実行します。

   ```
   oc create namespace namespace_name
   ```

   ここで、 *namespace_name* は、作成する名前空間の名前です。
3. OperatorHub:からクラスタで利用可能なオペレータのリストを表示する

   ```
   oc get packagemanifests -n openshift-marketplace
   ```
4. IBM MQ Operator を調べて、サポートされている **InstallModes** かつ使用可能な **Channels** ことを確認します。

   ```
   oc describe packagemanifests ibm-mq -n openshift-marketplace
   ```
5. オプション: **OperatorGroup** を作成します。

   OperatorGroup は、**OperatorGroup** と同じ名前空間におけるすべてのオペレーターに必要となる RBAC アクセス権限を生成するターゲット名前空間を選択する OLM リソースです。

   オペレーターのサブスクライブ先となる名前空間には **OperatorGroup** が必要です。これは、オペレーターの **InstallMode** (`AllNamespaces` モードまたは `SingleNamespace` モードのいずれか) に適合するものです。

   インストールするオペレーターが `AllNamespaces` モードを使用する場合は、 `openshift-operators` 名前空間に適切な **OperatorGroup** が既に設定されているため、このステップをスキップできます。

   オペレーターが `SingleNamespace` モードを使用し、適切な **OperatorGroup** がまだ設定されていない場合は、以下のコマンドを実行して作成します。

   ```
   cat << EOF | oc apply -f -
   apiVersion: operators.coreos.com/v1
   kind: OperatorGroup
   metadata:
     name: operatorgroup_name
     namespace: namespace_name
   spec:
     targetNamespaces:
     - namespace_name
   EOF
   ```
6. 35 ページの『IBM MQ Operator のバージョン・サポート』を確認して、選択するオペレーター・チャネルを判別します。
7. オペレーターをインストールします。

以下のコマンドを使用し、以下の値を変更する：

- *ibm_mq_operator_channel* を、インストールする IBM MQ Operator バージョンのチャネルに一致するように変更する。
- "AllNamespaces"モードを使用している場合は *namespace_name* を **openshift-operators** に、"SingleNamespace"モードを使用している場合は IBM MQ Operator をデプロイしたい名前空間に変更します。
- *catalog_source_name* を、前のインストール・ステップ 60 ページの『IBM MQ Operator カタログ・ソースの追加』で追加された、IBM MQ Operator インストールに使用されるカタログ・ソースの名前に変更する。

```
cat << EOF | oc apply -f -
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-mq
  namespace: namespace_name
spec:
  channel: ibm_mq_operator_channel
  installPlanApproval: Automatic
  name: ibm-mq
  source: catalog_source_name
  sourceNamespace: openshift-marketplace
EOF
```

8. 数分後に、オペレーターがインストールされます。 以下のコマンドを実行して、すべてのコンポーネントが「成功」状態であることを確認します。

```
oc get csv -n namespace_name | grep ibm-mq
```

ここで *namespace_name* は、"AllNamespaces"モードを使用している場合は **openshift-operators**、"SingleNamespace"モードを使用している場合はプロジェクト（ネームスペース）名です。

次のタスク

これで、 ライセンス・キー・シークレットを作成 する準備ができました (57 ページの『IBM MQ Operator のインストール』のステップ 6)。

OpenShift CP4I CP4I で使用するための IBM MQ Operator のインストール

IBM Cloud Pak for Integration (CP4I) で使用する場合、 OpenShift コンソールまたはコマンド・ライン・インターフェース (CLI) を使用して、 IBM MQ Operator を Red Hat OpenShift にインストールできます。

始める前に

重要：

- このトピックでは、 CP4I で使用するために IBM MQ Operator をインストールする場合、または CP4I ライセンス **のみ**を使用して少なくとも 1 つのキュー・マネージャーをデプロイする場合について説明します。 スタンドアロンで使用するための IBM MQ Operator のインストール手順については、 57 ページの『IBM MQ Operator のインストール』を参照してください。
- IBM MQ Operator をインストールする前に、 デプロイメントの構造化 に関するガイダンスを確認してください。

インストールが可能な限り円滑に行われるようにするには、インストールを開始する前に、すべての前提条件および要件を理解しておく必要があります。 27 ページの『コンテナー内の IBM MQ の計画』 を参照してください。

このタスクについて

以下のステップは、IBM MQ Operator をインストールするための標準的なタスク・フローを示しています。

1. Red Hat OpenShift Container Platform をインストールします。

2. ストレージを構成する。
3. ミラー・イメージ (エア・ギャップのみ)。
4. IBM MQ Operator カタログを追加 し、クラスターを準備します。
5. IBM MQ Operator をインストールします。
6. ライセンス・キー・シークレットを作成します (オンライン・インストールのみ)。
7. オプション: IBM Cloud Pak for Integration (CP4I) とその依存関係をインストールします。
8. License Service をデプロイします。
9. キュー・マネージャーをデプロイします。

手順

1. Red Hat OpenShift Container Platform をインストールします。

   OpenShift をインストールする詳しい手順については、 Red Hat ソフトウェアのインストール 4.6 以降を参照してください。

   重要 : サポートされているバージョンの OpenShift Container Platform がインストールされていることを確認してください。 例えば、 IBM MQ Operator 3.6 以降を使用するには、 OpenShift Container Platform 4.12 以降をインストールする必要があります。 詳細については、IBM Cloud Pak と Red Hat OpenShift Container Platform の互換性を参照してください。

   Red Hat OpenShift Container Platform CLI を使用するすべてのステップで、 `oc login` を使用して OpenShift クラスターにログインする必要があります。 CLI をインストールするには、OpenShift CLI の概要を参照してください。

   OpenShift をインストールした後、使用権キーの秘密の作成で作成した IBM 使用権キーを使用して、コンテナー・ソフトウェアを検証し、アクセス権限を取得することができます。
2. ストレージを構成します。

   Red Hat OpenShift Container Platform でストレージ・クラスを定義し、サイズ設定要件を満たすようにストレージ構成を設定する必要があります。

   重要 : IBM MQ の単一インスタンス・キュー・マネージャーおよびネイティブ HA キュー・マネージャーは RWO アクセス・モードを使用できますが、複数インスタンス・キュー・マネージャーは 38 ページの『IBM MQ Operator のストレージの計画』で説明されているように RWX を必要とします。 IBM MQ 複数インスタンス・キュー・マネージャーには、特定のファイル・システム特性が必要です。この特性は、IBM MQ の共有ファイル・システムのテストの手順を使用して確認できます。

   既知の準拠および非準拠のファイル・システムのリスト、およびその他の制限または制約事項に関する注記は、 Testing statement for IBM MQ file systems に記載されています。

   推奨されるストレージ・プロバイダーについては、 CP4I Storage considerations ページを参照してください。
3. イメージをミラーリングします (エア・ギャップのみ)。

   クラスターが制限付き (エアギャップ) ネットワーク環境にある場合は、IBM MQ イメージをミラーリングする必要があります。 構成によっては、いくつかの追加コンポーネントのミラーリングが必要になる場合もあります。 以下の情報を読み、必要に応じてイメージをミラーリングします。

   - IBM MQ イメージをミラーリングする必要があります。

     重要 : バージョン 3.5.0 より、 IBM MQ CASE が変更され、単一のキューマネージャーバージョンだけを含むようになりました。 イメージのミラーリングプロセスはかなり高速に実行されますが、複数回実行する必要があるかもしれません。インストールするキューマネージャの各バージョンごとに 1 回ずつ実行する必要があります。

     詳細については、295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照してください。

必要なオペレータとキューマネージャのバージョンが含まれている IBM MQ CASE のバージョンを調べ、それぞれに対してエアギャップミラープロセスを繰り返す必要があります。 IBM MQ CASE ルックアップテーブルで、バージョン対応表をご確認ください。

例えば、下記にリンクされた指示に従ってイメージをミラーリングする場合、最新のオペレータと対応するキュー・マネージャの単一バージョンを、以下の値を使用してインストールします

```
export OPERATOR_PACKAGE_NAME=ibm-mq
export OPERATOR_VERSION=3.6.0
```

エアギャップのインストールとアップグレードに使用される IBM MQ CASE の内容については、'295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照のこと。

- また、以下の **すべて** の記述が当てはまるキュー・マネージャーを少なくとも 1 つデプロイする予定の場合は、いくつかの追加の必須コンポーネントもミラーリングする必要があります。
  - CP4I ライセンスを使用している。
  - IBM MQ Console が使用可能になります。
  - IBM MQ Console シングル・サインオン (SSO) 認証および許可に IBM Cloud Pak for Integration Keycloak サービスを使用している (デフォルト)。

  上記の記述がすべて当てはまる場合、SSO は Keycloak によって提供されます。 したがって、IBM MQ Operator の場合と同様に、これらの追加の必須コンポーネントについても、それぞれの手順を繰り返す必要があります
  - IBM Cloud Pak foundational services
  - IBM Cloud Pak for Integration
  - Keycloak (Red Hat OpenShift 演算子)

ミラー・イメージを作成するには、 エアギャップ・クラスターのミラーリング・イメージを参照してください。

4. IBM MQ Operator カタログ・ソースを追加します。

   以下の値を使用して、IBM MQ Operator をクラスターで使用できるようにするカタログ・ソースを追加します。

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=CHOSEN_OPERATOR_VERSION
   export ARCH=ARCHITECTURE
   ```

   説明:

   - 建築 is the architecture of the system on which you are deploying the IBM MQ Operator, and has a value of `amd64`, `ppc64le`, or `s390x`.
   - *CHOSEN_OPERATOR_VERSION* は、IBM MQ Operator の必須バージョンです。 The current CD version is 3.6.0. The current SC2 version is 3.2.13.

   少なくとも 1 つのキュー・マネージャーをデプロイするときに、以下の **すべて** の記述が当てはまる場合は、いくつかの追加の必須コンポーネントがあります。

   - CP4I ライセンスを使用している。
   - IBM MQ Console が使用可能になります。
   - IBM MQ Console シングル・サインオン (SSO) 認証および許可に IBM Cloud Pak for Integration Keycloak サービスを使用している (デフォルト)。

   上記の記述がすべて当てはまる場合、SSO は Keycloak によって提供されます。 したがって、 IBM MQ Operator カタログ・ソースの場合と同様に、以下の追加の必須コンポーネントのそれぞれについてもステップを繰り返す必要があります。

   - IBM Cloud Pak foundational services
   - IBM Cloud Pak for Integration

- Keycloak (Red Hat OpenShift 演算子)

クラスターへのカタログ・ソースの追加に記載されている、必要なカタログ・ソースの手順に従います。

5. IBM MQ Operator のインストール

以下の 2 つのオプションのいずれかを選択します (コンソールを使用するか、CLI を使用します)。

- オプション 1: OpenShift コンソールを使用して IBM MQ Operator をインストールします。
- オプション 2: OpenShift CLI を使用して IBM MQ Operator をインストールします。

6. 使用権キー・シークレットを作成します (オンライン・インストールのみ)。

IBM MQ Operator は、ライセンス資格検査を実行するコンテナー・レジストリーからプルされるキュー・マネージャー・イメージをデプロイします。 このチェックには、 `docker-registry` プル・シークレットに保管されている使用権キーが必要です。 キュー・マネージャーをインストールする名前空間にまだ使用権キーがない場合は、以下の手順に従って使用権キーを取得し、プル・シークレットを作成します。

注 : IBM MQ Advanced for Developers (非保証) キュー・マネージャーのみをデプロイする場合、ライセンス・キーは必要ありません。

OpenShift コンソールまたは CLI を使用して、使用権キー・シークレットを作成できます。 以下の例では、CLI を使用します。

a. IBM ID に割り当てられているライセンス・キーを取得します。 ライセンス契約済みソフトウェアに関連付けられている IBM ID とパスワードを使用して、 MyIBM Container Software Library にログインします。

b.「**使用権キー (Entitlement keys)**」セクションで「**キーのコピー (Copy key)**」を選択して、使用権キーをクリップボードにコピーします。

c. OpenShift CLI から、以下のコマンドを実行して、`ibm-entitlement-key` という名前のイメージ・プル・シークレットを作成します。

```
oc create secret docker-registry ibm-entitlement-key \
--docker-server=cp.icr.io \
--docker-username=cp \
--docker-password=entitlement_key \
--docker-email=user_email
\--namespace=namespace
```

ここで、 *entitlement_key* はステップ b でコピーした使用権キー、 *user_email>* は使用権付きソフトウェアに関連付けられた IBM ID、 *namespace* は IBM MQ Operator をインストールした名前空間です。

7. オプション: CP4I とその依存関係をインストールします。

少なくとも 1 つのキュー・マネージャーをデプロイするときに、以下の **すべて** の記述が当てはまる場合は、いくつかの追加の必須コンポーネントがあります。

- CP4I ライセンスを使用している。
- IBM MQ Console が使用可能になります。
- IBM MQ Console シングル・サインオン (SSO) 認証および許可に CP4I Keycloak サービスを使用している (デフォルト)。

上記の記述がすべて当てはまる場合、SSO は Keycloak によって提供されるため、以下の追加ステップを実行する必要があります。

- CP4I Operator と同じインストール・モードで IBM Cloud Pak foundational services Operator をインストールします。 サポートされるバージョンについては、 このリリースのオペレーター・チャネルのバージョンを参照してください。
- CP4I Operator をインストールします。
- オプション: プラットフォーム UI をデプロイします。

a. `ibm-common-services` 名前空間を作成します。 CLI を使用して OpenShift クラスターにログインしたら、以下のコマンドを実行します。

```
oc new-project ibm-common-services
```

b. プラットフォーム UI をデプロイします。

8. License Service をデプロイします。

   これは、キュー・マネージャーのライセンス使用状況をモニターするために必要です。 License Service の手順に従います。

9. キュー・マネージャーをデプロイします。

   サンプルの "クイック・スタート" ・キュー・マネージャーのデプロイ手順については、 98 ページの『IBM MQ Operator を使用した単純なキュー・マネージャーのデプロイ』を参照してください。

関連タスク

84 ページの『IBM MQ Operator のアンインストール』
Red Hat OpenShift コンソールまたは CLI を使用して、Red Hat OpenShift から IBM MQ Operator をアンインストールできます。

OpenShift CP4I IBM MQ Operator とキュー・マネージャーのアップグレード

IBM MQ Operator のユーザーには、 IBM MQ ライセンスを使用するか、 IBM Cloud Pak for Integration (CP4I) ライセンスを使用するかに応じて、さまざまなアップグレード・プロセスがあります。 ご使用のデプロイメント・タイプのアップグレード・ステップを実行します。

このタスクについて

IBM MQ Operator およびキュー・マネージャーをアップグレードするには、以下のいずれかのステップを実行します。

手順

- オプション 1: **現在のオペレーター・チャネルでデプロイメントを最新バージョンにアップグレードします。**

  現在のオペレーター・チャネルで IBM MQ Operator のデプロイメントを最新バージョンにアップグレードするには、 70 ページの『IBM MQ Operator チャネルの最新セキュリティー・リリースへのアップグレード』を参照してください。

- オプション 2: **IBM MQ Operator for IBM MQ ライセンスをアップグレードします。**

  IBM MQ ライセンス **のみ** を使用する IBM MQ Operator のデプロイメントをアップグレードするには、 69 ページの『IBM MQ Operator のアップグレード』を参照してください。

- オプション 3: **CP4I ユーザーのために IBM MQ Operator をアップグレードします。**

  IBM Cloud Pak for Integration のユーザーのために IBM MQ Operator のデプロイメントをアップグレードします。 これには、 CP4I ライセンスの下に少なくとも 1 つのキュー・マネージャーをデプロイした場合が含まれます。 79 ページの『CP4I ユーザーのための IBM MQ Operator のアップグレード』を参照してください。

OpenShift *IBM MQ Operator のアップグレード*

IBM MQ ライセンス **のみ** が使用されている IBM MQ Operator のデプロイメントをアップグレードします。

始める前に

重要 : このタスクは、 IBM MQ Operator および **のみ** の IBM MQ ライセンスのユーザーを対象としています。 IBM Cloud Pak for Integration (CP4I) ユーザーである場合、または CP4I ライセンスを使用して少なくとも 1 つのキュー・マネージャーをデプロイした場合は、 79 ページの『CP4I ユーザーのための IBM MQ Operator のアップグレード』を参照してください。

このタスクについて

以下のいずれかのステップを、必要なアップグレードに合わせて実行します。

注 :

- Version 3.6.0 of the IBM MQ Operator is the latest CD release.
- Version 3.2.13 of the IBM MQ Operator is the latest SC2 release.

手順

- オプション 1:70 ページの『IBM MQ Operator チャネルの最新セキュリティー・リリースへのアップグレード』
- オプション 2:72 ページの『Upgrading to the 3.2.x SC2 channel of the IBM MQ Operator』
- オプション 3:75 ページの『Upgrading to the current CD channel of the IBM MQ Operator』

IBM MQ Operator チャネルの最新セキュリティー・リリースへのアップグレード

IBM MQ Operator をアップグレードすると、キュー・マネージャーをアップグレードできます。

始める前に

重要 : このトピックでは、 IBM MQ Operator のデプロイメントを、デプロイメントのチャネル上の最新のセキュリティー・リリースにアップグレードします。 これがご使用のデプロイメントに該当しない場合は、69 ページの『IBM MQ Operator とキュー・マネージャーのアップグレード』で説明されている代替アップグレード・パスを参照してください。

このタスクについて

アップグレードされる IBM MQ Operator の展開に使用されるカタログ・ソースによって、異なる手順がある。

オプション 1: IBM MQ Operator の特定のカタログ・ソース

新しい'IBM MQ Operator バージョンは、カタログソースを更新した**後**にのみ OpenShift クラスターで利用可能になります。 このプロセスにより、アップグレードを手動で制御できるため、オペレーターの **Update approval** 設定に「**手動**」 オプションを使用する必要はありません。 「**手動**」 オプションを選択すると、可能なすべてのアップグレードが同時に強制的に実行され、アップグレードがブロックされる可能性があるため、 「**自動**」 オプションのみを使用してください。

このオプションを使用するには、 IBM MQ Operator の特定のカタログ・ソースを使用したアップグレードにスキップします。

オプション 2: IBM オペレーター・カタログ

このオプションを使用すると、新しいオペレーター・バージョンが使用可能になり、ユーザーの介入を**必要とせずに** 適用されます。 そのため、このオプションは、 IBM MQ Operator の **自動** アップグレードが必要で、確定的インストールが必要ないオンライン・インストールに **のみ** 使用してください。 このオプションは PoC (概念検証) 環境には役立ちますが、 **実稼働環境には適していません**。

このオプションを使用するには、 IBM オペレーター・カタログを使用したアップグレードにスキップしてください。

IBM オペレーター・カタログの使用から、アップグレードをより詳細に制御できる IBM MQ Operator の特定のカタログ・ソースの使用に移行するには、 78 ページの『IBM MQ Operator の特定のカタログ・ソースへの移動』を参照してください。

手順

- **IBM MQ Operator の特定のカタログ・ソースを使用したアップグレード**

  a) 鏡像（エアギャップのみ）。

クラスタが制限された（エアギャップ）ネットワーク環境にある場合は、 IBM MQ のイメージをミラーリングする必要があります。

重要 : バージョン 3.5.0 より、 IBM MQ CASE が変更され、単一のキューマネージャーバージョンだけを含むようになりました。 イメージのミラーリングプロセスはかなり高速に実行されますが、複数回実行する必要があるかもしれません。インストールするキューマネージャの各バージョンごとに 1 回ずつ実行する必要があります。

詳細については、295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照してください。

必要なオペレータとキューマネージャのバージョンが含まれている IBM MQ CASE のバージョンを調べ、それぞれに対してエアギャップミラープロセスを繰り返す必要があります。 IBM MQ CASE ルックアップテーブルで、バージョン対応表をご確認ください。

例えば、下記にリンクされている指示に従ってイメージをミラーリングする場合、以下の値を使用して、最新のオペレータと対応するキュー・マネージャの単一バージョンをインストールします

```
export OPERATOR_PACKAGE_NAME=ibm-mq
export OPERATOR_VERSION=3.6.0
```

エアギャップのインストールとアップグレードに使用される IBM MQ CASE の内容については、'295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照のこと。

ミラーイメージを作成するには、エアギャップクラスターのミラーイメージを参照してください。

b) 最新のカタログ・ソースを適用します。

「 IBM MQ Operator カタログ・ソースの追加」の「 IBM MQ Operator の特定のカタログ・ソースの追加」の説明に従ってください。

c) IBM MQ Operator をアップグレードします。

IBM MQ Operator **更新承認**ステータスが「**自動**」に設定されている場合、オペレータはアップグレードします。 そうでない場合は、手動で IBM MQ Operator をアップグレードする：

a. ナビゲーション・ペインで、「**オペレーター (Operators)**」 > 「**インストール済みのオペレーター (Installed Operators)**」をクリックします。

指定したプロジェクトにインストールされているすべてのオペレーターが表示されます。

b.「**IBM MQ Operator**」を選択します。

c.「**サブスクリプション**」タブにナビゲートします。

d.「**アップグレード可能**」 をクリックします。

e.「**プレビュー**」 **InstallPlan をクリックします。**

f. アップグレードを完了するには、「**承認**」をクリックします

オペレーターが新しいバージョンにアップグレードします。

d) すべての IBM MQ キュー・マネージャーをアップグレードします。

UpgradeIBM MQqueue managers の指示に従ってください。

- **IBM オペレーター・カタログを使用したアップグレード**

a) IBM MQ Operator を新しいバージョンにアップグレードします。

自動アップグレードが設定されている場合、新しいセキュリティー・リリースがリリースされると、IBM MQ Operator はアップグレードを完了します。 そうでない場合は、手動でアップ IBM MQ Operator を承認してください：

– 使用可能なアップグレードがある場合、 **Upgrade Status** は「Upgrade available」の可能性があります。

– この場合、 IBM MQ Operator をアップグレードする **InstallPlan** を承認するために使用できる制御が存在する可能性があります。

b) すべての IBM MQ キュー・マネージャーのアップグレード

UpgradeIBM MQqueue managers の指示に従ってください。

- **IBM MQ キュー・マネージャーをアップグレードします。**

  IBM MQ Operator のアップグレード後に、すべての IBM MQ キュー・マネージャーを新しいバージョンにアップグレードする必要があります。 以下の表では、各アクティブ Operator チャネルの IBM MQ キュー・マネージャの最新バージョンについて説明しています。 該当するバージョンを使用して、83 ページの『Red Hat OpenShift を使用した IBM MQ キュー・マネージャーのアップグレード』の手順に従ってください。

オペレーター・チャネル	最新の IBM MQ キュー・マネージャー
3.2SC2)	9.4.0.11-r3
3.6CD)	9.4.3.0-r1

OpenShift CP4I *Upgrading to the 3.2.x SC2 channel of the IBM MQ Operator*
Upgrade deployments of the IBM MQ Operator where **のみ** IBM MQ licenses are used.

始める前に

重要：

- このタスクは、IBM MQ Operator IBM MQ のユーザーのための**もの**です。 もしあなたが IBM Cloud Pak for IntegrationCP4I) ユーザであるか、または CP4I ライセンスを使用してキュー・マネージャの少なくとも 1 つをデプロイしている場合は、79 ページの『CP4I ユーザーのための IBM MQ Operator のアップグレード』を参照してください。
- This task is for upgrading deployments of the IBM MQ Operator to the Support Cycle 2 (SC2) channel of IBM MQ Operator 3.2.x **のみ**. これは、最新の CD リリース(3.6.0*)にアップグレードする際にも、中間的なステップになる可能性があることに注意してください。 これがあなたの配備に当てはまらない場合は、69 ページの『IBM MQ Operator とキュー・マネージャーのアップグレード』 で説明されている別のアップグレードパスを参照してください。

このタスクについて

3.2.x SC2 チャネルにアップグレードする IBM MQ Operator 配備に応じて、以下のオプションのいずれかを完了します：

手順

- **オプション 1**:2.0.x Long Term SupportLTS)IBM MQ Operator のデプロイメントをアップグレードする。

  72 ページの『2.0.x LTS IBM MQ Operator から 3.2.x SC2 チャンネルへのアップグレード』のステップを実行します。
- **オプション 2**:IBM MQ Operator CD のデプロイメントをアップグレードする。

  73 ページの『Upgrading a CD IBM MQ Operator to the v3.2.x SC2 channel』のステップを実行します。

OpenShift CP4I-SC2 V 9.4.0 *2.0.x LTS IBM MQ Operator から 3.2.x SC2 チャンネルへのアップグレード*
IBM MQ Operator をアップグレードすると、キュー・マネージャーをアップグレードできます。

始める前に

重要：

- このタスクは、 IBM MQ Operator および **のみ** の IBM MQ ライセンスのユーザーを対象としています。 IBM Cloud Pak for Integration (CP4I) ユーザーである場合、または CP4I ライセンスを使用して少なくとも 1 つのキュー・マネージャーをデプロイした場合は、 79 ページの『CP4I ユーザーのための IBM MQ Operator のアップグレード』を参照してください。

- このトピックは、 2.0.x Long Term Support (LTS) IBM MQ Operator を IBM MQ Operator 3.2.x **のみ**の Support Cycle 2 (SC2) チャネルにアップグレードするためのものです。 これがご使用のデプロイメントに該当しない場合は、 69 ページの『IBM MQ Operator とキュー・マネージャーのアップグレード』で説明されている代替アップグレード・パスを参照してください。

IBM MQ Operator 3.2.13 にアップグレードするには、 Red Hat OpenShift Container Platform 4.12 以降を実行している必要があります。 各 IBM MQ Operator チャネルの互換性のあるバージョンを確認するには、35 ページの『互換性のある Red Hat OpenShift Container Platform のバージョン』を参照してください。 プラットフォームをアップグレードするには、 Red Hat OpenShift のアップグレードを参照してください。

手順

1. **ミラー・イメージ (エア・ギャップのみ)**。

   クラスタが制限された（エアギャップ）ネットワーク環境にある場合は、 IBM MQ のイメージをミラーリングする必要があります。

   重要 : バージョン 3.5.0 より、IBM MQ CASE が変更され、単一のキューマネージャーバージョンだけを含むようになりました。 イメージのミラーリングプロセスはかなり高速に実行されますが、複数回実行する必要があるかもしれません。インストールするキューマネージャの各バージョンごとに 1 回ずつ実行する必要があります。

   詳細については、295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照してください。

   必要なオペレータとキューマネージャのバージョンが含まれている IBM MQ CASE のバージョンを調べ、それぞれに対してエアギャップミラープロセスを繰り返す必要があります。 IBM MQ CASE ルックアップテーブルで、バージョン対応表をご確認ください。

   例えば、下記にリンクされた指示に従ってイメージをミラーリングする場合、最新のオペレータと対応するキュー・マネージャの単一バージョンを、以下の値を使用してインストールします

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.6.0
   ```

   以前のインストールまたはアップグレード時にイメージ・レジストリーへの接続がセットアップされている必要があるため、セクション 3.5 「クラスターの構成」を省略する必要があります。

   リンク: エアギャップ・クラスター用のミラーリング・イメージ。
2. **IBM MQ Operator を 3.2.13 にアップグレードします**。

   81 ページの『Red Hat OpenShift を使用した IBM MQ Operator のアップグレード』 を参照してください。
3. **インスタンスをアップグレードします**。

   最新の機能とセキュリティー・フィックスを入手するには、IBM MQ Operand (キュー・マネージャー・コンテナー・イメージ) を最新の SC2 バージョン (9.4.0.11) にアップグレードします。 83 ページの『Red Hat OpenShift を使用した IBM MQ キュー・マネージャーのアップグレード』を参照してください。

OpenShift CP4I V9.4.0 *Upgrading a CD IBM MQ Operator to the v3.2.x SC2 channel*

IBM MQ Operator をアップグレードすると、キュー・マネージャーをアップグレードできます。

始める前に

重要 :

- このタスクは、 IBM MQ Operator および **のみ** の IBM MQ ライセンスのユーザーを対象としています。 IBM Cloud Pak for Integration (CP4I) ユーザーである場合、または CP4I ライセンスを使用して少なくとも 1 つのキュー・マネージャーをデプロイした場合は、 79 ページの『CP4I ユーザーのための IBM MQ Operator のアップグレード』を参照してください。

- このタスクは、バージョン'3.2.0 以前の'IBM MQ Operator の'Continuous Delivery(CD)デプロイメントをバージョン'3.2.13'**のみ**にアップグレードするためのものである。 これは、最新の「CD リリース (v3.6.0) にアップグレードする際にも、中間的なステップになる可能性があることに注意してください。 これがあなたの配備に当てはまらない場合は、「69 ページの『IBM MQ Operator とキュー・マネージャーのアップグレード』」で説明されている代替のアップグレードパスを参照してください。

IBM MQ Operator 3.2.13 にアップグレードするには、 Red Hat OpenShift Container Platform 4.12 以降を実行している必要があります。 各 IBM MQ Operator チャネルの互換性のあるバージョンを確認するには、35 ページの『互換性のある Red Hat OpenShift Container Platform のバージョン』を参照してください。 プラットフォームをアップグレードするには、 Red Hat OpenShift のアップグレードを参照してください。

手順

1. オプション: **3.0.0 より前の CD バージョンである IBM MQ Operator をアップグレードします。**

   ご使用の IBM MQ Operator が現在 3.0.0 より前の CD バージョンである場合は、IBM MQ Operator (IBM MQ 9.3 資料) の現行 CD チャネルへのマイグレーションの関連ステップに従ってから、ここに戻って最新の CD バージョンにアップグレードしてください。 これは、バージョン 3.2.13 にアップグレードする前の必須の前提条件ステップであることに注意してください。
2. **現在のチャンネルを最新のオペレータにアップグレードする**

   70 ページの『IBM MQ Operator チャネルの最新セキュリティー・リリースへのアップグレード』のステップを実行します。
3. **ミラー・イメージ (エア・ギャップのみ)。**

   クラスタが制限された（エアギャップ）ネットワーク環境にある場合は、 IBM MQ のイメージをミラーリングする必要があります。

   重要 : バージョン 3.5.0 より、IBM MQ CASE が変更され、単一のキューマネージャーバージョンだけを含むようになりました。 イメージのミラーリングプロセスはかなり高速に実行されますが、複数回実行する必要があるかもしれません。インストールするキューマネージャの各バージョンごとに 1 回ずつ実行する必要があります。

   詳細については、295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照してください。

   必要なオペレータとキューマネージャのバージョンが含まれている IBM MQ CASE のバージョンを調べ、それぞれに対してエアギャップミラープロセスを繰り返す必要があります。 IBM MQ CASE ルックアップテーブルで、バージョン対応表をご確認ください。

   例えば、下記にリンクされた指示に従ってイメージをミラーリングする場合、最新のオペレータと対応するキュー・マネージャの単一バージョンを、以下の値を使用してインストールします

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.6.0
   ```

   以前のインストールまたはアップグレード時にイメージ・レジストリーへの接続がセットアップされている必要があるため、セクション 3.5 「クラスターの構成」を省略する必要があります。

   リンク: エアギャップ・クラスター用のミラーリング・イメージ。
4. **IBM MQ Operator を 3.2.13 にアップグレードします。**

   81 ページの『Red Hat OpenShift を使用した IBM MQ Operator のアップグレード』 を参照してください。
5. **インスタンスをアップグレードします。**

   最新の機能とセキュリティー・フィックスを入手するには、IBM MQ Operand (キュー・マネージャー・コンテナー・イメージ) を最新の SC2 バージョン (9.4.0.11) にアップグレードします。 83 ページの『Red Hat OpenShift を使用した IBM MQ キュー・マネージャーのアップグレード』を参照してください。

OpenShift *Upgrading to the current CD channel of the IBM MQ Operator*
IBM MQ **ライセンス**のみが使用されている「IBM MQ Operator デプロイメントをアップグレードする。

始める前に

重要：

- このタスクは、IBM MQ Operator IBM MQ のユーザーのための**もの**です。 もしあなたが IBM Cloud Pak for IntegrationCP4I) ユーザであるか、または CP4I ライセンスを使用してキュー・マネージャの少なくとも 1 つをデプロイしている場合は、79 ページの『CP4I ユーザーのための IBM MQ Operator のアップグレード』を参照してください。
- This topic is for upgrading to the Continuous Delivery (CD) channel of the IBM MQ Operator (v3.6) **のみ**. これがあなたの配備に当てはまらない場合は、69 ページの『IBM MQ Operator とキュー・マネージャーのアップグレード』 で説明されている別のアップグレードパスを参照してください。

このタスクについて

3.6CD チャンネルにアップグレードする'IBM MQ Operator のデプロイメントに応じて、以下のオプションのいずれかを完了します：

手順

- **オプション 1：** 3.2.x'SC2'IBM MQ Operator のデプロイメントをアップグレードする。

  75 ページの『3.2.x の'SC2'IBM MQ Operator を現在の'CD チャンネルにアップグレードする』のステップを実行します。
- **オプション 2：**は'IBM MQ Operator の'CD 展開をアップグレードする。

  76 ページの『CD「IBM MQ Operator」を現在の「CD チャンネルにアップグレードする』のステップを実行します。

OpenShift *3.2.x の'SC2'IBM MQ Operator を現在の'CD チャンネルにアップグレードする*
IBM MQ **ライセンス**のみが使用されている 'IBM MQ Operator 3.2.x'SC2 デプロイメントをアップグレードします。

始める前に

重要：

- このタスクは'IBM MQ Operator と'IBM MQ ライセンスのユーザーのためのもの**です**。 IBM Cloud Pak for Integration(CP4I) ユーザ、または 'CP4I ライセンスを使用してキューマネージャの少なくとも 1 つをデプロイしている場合は、'79 ページの『CP4I ユーザーのための IBM MQ Operator のアップグレード』を参照してください。
- このタスクは、3.2.x の'SC2 バージョンの 'IBM MQ Operator を、現在の 'Continuous Delivery(CD) チャネルの 'IBM MQ Operator(v3.6) にアップグレードするためのもの**で**ある。 これがあなたの配備に当てはまらない場合は、「69 ページの『IBM MQ Operator とキュー・マネージャーのアップグレード』」で説明されている別のアップグレードパスを参照してください。

IBM MQ Operator'3.6.0 にアップグレードするには、'Red Hat OpenShift Container Platform'4.12 以降を実行している必要があります。 各'IBM MQ Operator チャンネルの互換バージョンを確認するには、'35 ページの『互換性のある Red Hat OpenShift Container Platform のバージョン』を参照のこと。 プラットフォームをアップグレードするには、「IBM Cloud Pak for Integration ドキュメントの「 Red Hat OpenShift アップグレードを参照してください。

手順

1. オプション: **現在 3.2.0 より前のバージョンの'IBM MQ Operator をアップグレードする。**

あなたの「IBM MQ Operator 現在 3.2.0 より前のバージョンである場合、「72 ページの『Upgrading to the 3.2.x SC2 channel of the IBM MQ Operator』」の関連する手順に従って、最新の「CD バージョンにアップグレードしてください。 Note that if you are already on a v3.2.x version of the IBM MQ Operator you can skip this step, however upgrading to the v3.2 channel of the IBM MQ Operator is a mandatory prerequisite step before upgrading to version 3.6.0.

2. **現在のチャンネルで最新のオペレータにアップグレードする**

   70 ページの『IBM MQ Operator チャネルの最新セキュリティー・リリースへのアップグレード』のステップを実行します。

3. **鏡像（エアギャップのみ）。**

   クラスタが制限された（エアギャップ）ネットワーク環境にある場合は、 IBM MQ のイメージをミラーリングする必要があります。

   重要 : バージョン 3.5.0 より、IBM MQ CASE が変更され、単一のキューマネージャーバージョンだけを含むようになりました。 イメージのミラーリングプロセスはかなり高速に実行されますが、複数回実行する必要があるかもしれません。インストールするキューマネージャの各バージョンごとに 1 回ずつ実行する必要があります。

   詳細については、295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照してください。

   必要なオペレータとキューマネージャのバージョンが含まれている IBM MQ CASE のバージョンを調べ、それぞれに対してエアギャップミラープロセスを繰り返す必要があります。 IBM MQ CASE ルックアップテーブルで、バージョン対応表をご確認ください。

   例えば、下記にリンクされた指示に従ってイメージをミラーリングする場合、最新のオペレータと対応するキュー・マネージャの単一バージョンを、以下の値を使用してインストールします

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.6.0
   ```

   セクション 3.5「クラスタの設定」は省略する必要があります。なぜなら、イメージレジストリへの接続は以前のインストールまたはアップグレード時に設定されているはずだからです。

   エアギャップのインストールとアップグレードに使用される IBM MQ CASE の内容については、'295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照のこと。

   リンク エアギャップクラスターのミラーリングイメージ。

4. **IBM MQ Operator を'3.6.0 にアップグレードする。**

   81 ページの『Red Hat OpenShift を使用した IBM MQ Operator のアップグレード』 の指示に従ってください。

5. **インスタンスをアップグレードする。**

   最新の機能とセキュリティ修正を受けるには、IBM MQ オペランド（キューマネージャーコンテナーイメージ）を最新の CD バージョン（9.4.3.0-r1 ）にアップグレードしてください。83 ページの『Red Hat OpenShift を使用した IBM MQ キュー・マネージャーのアップグレード』 を参照してください。

OpenShift *CD*「*IBM MQ Operator*」を現在の「*CD* チャンネルにアップグレードする

IBM MQ **ライセンス**のみが使用されている「IBM MQ Operator「CD デプロイメントをアップグレードします。

始める前に

重要 :

- このタスクは'IBM MQ Operator と'IBM MQ ライセンスのユーザーのためのもの**です**。 IBM Cloud Pak for Integration(CP4I) ユーザ、または 'CP4I ライセンスを使用してキューマネージャの少なくとも 1 つをデプロイしている場合は、'79 ページの『CP4I ユーザーのための IBM MQ Operator のアップグレード』を参照してください。

- このタスクは、「IBM MQ Operator」の「Continuous Delivery(CD)バージョンを、「IBM MQ Operator」の現在の「CD チャンネル (v3.6)にアップグレードするためのものである。 これがあなたの配備に当てはまらない場合は、「69 ページの『IBM MQ Operator とキュー・マネージャーのアップグレード』」で説明されている別のアップグレードパスを参照してください。

IBM MQ Operator'3.6.0 にアップグレードするには、'Red Hat OpenShift Container Platform'4.12 以降を実行している必要があります。 各'IBM MQ Operator チャンネルの互換バージョンを確認するには、'35 ページの『互換性のある Red Hat OpenShift Container Platform のバージョン』を参照のこと。 プラットフォームをアップグレードするには、「IBM Cloud Pak for Integration ドキュメントの「 Red Hat OpenShift アップグレードを参照してください。

手順

1. オプション: **現在 3.2.0 より前のバージョンの'IBM MQ Operator をアップグレードする。**

   あなたの'IBM MQ Operator が現在 3.2.0 より前のバージョンである場合、'72 ページの『Upgrading to the 3.2.x SC2 channel of the IBM MQ Operator』の関連する手順に従ってください、そして、最新の'CD バージョンにアップグレードするためにステップ'77 ページの『2』に従ってここに戻ってください。 Note that if you are already on a 3.2.x version of the IBM MQ Operator you can skip this step, however upgrading to the v3.2 channel of the IBM MQ Operator is a mandatory prerequisite step before upgrading to version 3.6.0.
2. オプション: **現在バージョン'3.2.x の'IBM MQ Operator をアップグレードする。**

   あなたの'IBM MQ Operator が現在バージョン'3.2.x の場合、'75 ページの『3.2.x の'SC2'IBM MQ Operator を現在の'CD チャンネルにアップグレードする』の関連するステップに従う。
3. **現在バージョン 3.3.0 以上の IBM MQ Operator をアップグレードします。**

   IBM MQ Operator が現在、バージョン 3.3.0 またはそれ以上である場合は、以下のサブステップを完了して、 IBM MQ Operator 3.6.0 にアップグレードしてください。

   a) **現在のチャンネルで最新のオペレータにアップグレードする**

   70 ページの『IBM MQ Operator チャネルの最新セキュリティー・リリースへのアップグレード』のステップを実行します。

   b) **次のオペレータチャネルに切り替えます。**

      i) 鏡像（エアギャップのみ）。

      クラスタが制限された（エアギャップ）ネットワーク環境にある場合は、 IBM MQ のイメージをミラーリングする必要があります。 これらの値のみを使用して、以下のリンクの手順を完了します：

      ```
      export OPERATOR_PACKAGE_NAME=ibm-mq
      export OPERATOR_VERSION=[operator_version]
      ```

      [operator_version] は、アップグレード先の IBM MQ Operator のバージョンです。 アップグレード先のチャネルにおける最新の IBM MQ Operator を決定するには、 5 ページの『IBM MQ Operator のリリース履歴』 を参照してください。

      セクション 3.5「クラスタの設定」は省略する必要があります。なぜなら、イメージレジストリへの接続は以前のインストールまたはアップグレード時に設定されているはずだからです。

      エアギャップのインストールとアップグレードに使用される IBM MQ CASE の内容については、'295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照のこと。

      リンク エアギャップクラスターのミラーリングイメージ。

      ii) IBM MQ Operator をアップグレードしてください。

      81 ページの『Red Hat OpenShift を使用した IBM MQ Operator のアップグレード』 の指示に従ってください。

      iii) インスタンスをアップグレードします。

最新の機能とセキュリティ修正プログラムを受け取るには、 IBM MQ オペランド（キューマネージャーのコンテナイメージ）をアップグレードしてください。 83 ページの『Red Hat OpenShift を使用した IBM MQ キュー・マネージャーのアップグレード』を参照してください。

c) 最新の CD チャンネル（v3.6.0 ）に到達するまで、前のサブステップ（a）と（b）を繰り返します。

OpenShift *IBM MQ Operator* の特定のカタログ・ソースへの移動

以前のリリースの IBM MQ Operator がインストールされていて、IBM オペレーター・カタログを使用している場合、特定のカタログ・ソースを適用することが、クラスター上のソフトウェアのバージョン管理を完全に制御するための最も効果的な方法です。

始める前に

重要 : このタスクは、クラスター管理者が実行する必要があります。 OpenShift の役割と権限を参照してください。

CLI を使用して、以下のステップを実行します。

このタスクについて

IBM オペレーター・カタログは、 Red Hat OpenShift Container Platform クラスターの API を拡張して IBM ソフトウェア製品を使用可能にするために使用できるオペレーターの索引です。

この手順では、 IBM オペレーター・カタログから IBM MQ Operator のインストールを移動して、 IBM MQ Operator の特定のカタログ・ソースを使用できるようにします。

手順

1. IBM MQ Operator カタログを追加します。

   「IBM MQ Operator カタログ・ソースの追加」の「IBM MQ Operator の特定のカタログ・ソースの追加」の説明に従ってください。
2. IBM MQ Operator カタログ・ソースが `openshift-marketplace` 名前空間に作成されたことを確認します。

   以下のコマンドを実行します。

   ```
   oc get catalogsource -n openshift-marketplace
   ```

   出力例:

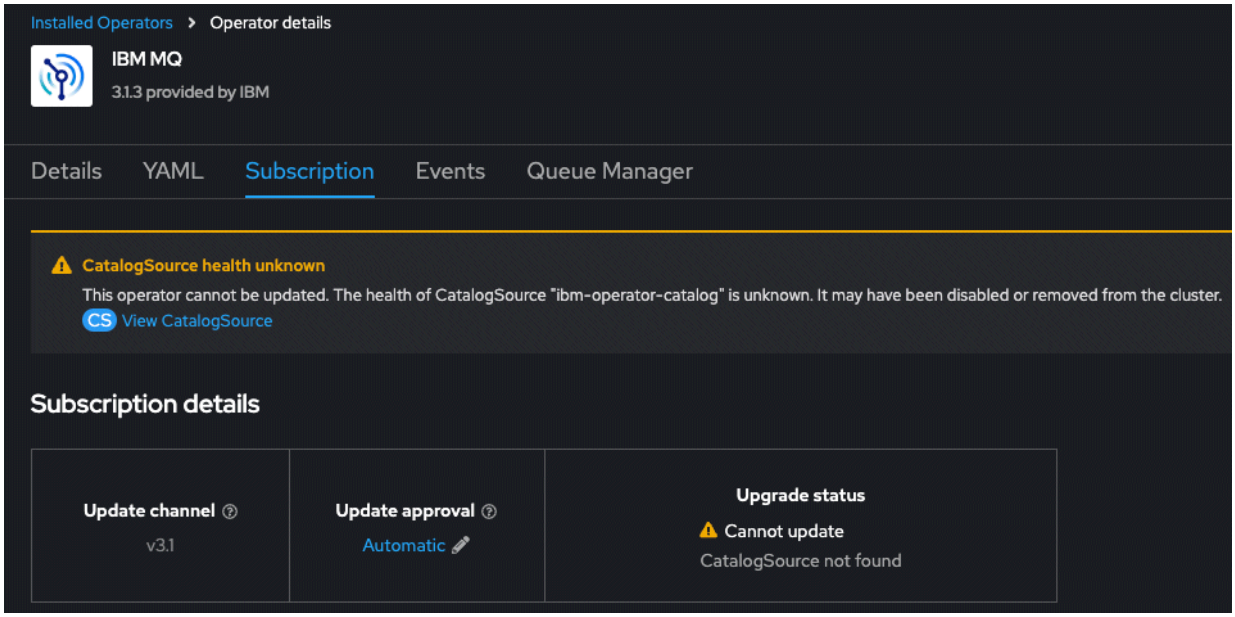
   ```
   oc get catalogsource -n openshift-marketplace

   NAME                           DISPLAY                TYPE   PUBLISHER   AGE
   ibm-operator-catalog           IBM Operator Catalog   grpc   IBM         23h
   ibmmq-operator-catalogsource   ibm-mq-3.1.3           grpc   IBM         23h
   ```
3. オプション: IBM Operator カタログ・ソースを削除します。

   警告 : このステップは、 IBM オペレーター・カタログを使用しているオペレーターが他にないことが確実な場合にのみ実行してください。

   以下のコマンドを実行します。

   ```
   oc delete catalogsource ibm-operator-catalog -n openshift-marketplace
   ```

   IBM MQ Operator の状況が `CatalogSource not found` に変わります。 これは、予期されていることです。



4. IBM MQ Operator のサブスクリプションを、新しい特定の IBM MQ Operator カタログ・ソースを指すように変更します。

   a) サブスクリプションを編集します。

   以下のコマンドを実行します。 *OPERATOR-NAMESPACE* は、 IBM MQ Operator のクラスター全体のインストール済み環境の場合は `openshift-operators` に、 IBM MQ Operator がデプロイされている特定の名前空間に置き換えてください。

   ```
   oc edit subscription ibm-mq -n OPERATOR-NAMESPACE
   ```

   b) `spec.source` 値を `ibm-operator-catalog` から、ステップ 78 ページの『1』 で作成したカタログ・ソースの名前に変更します。

   以下に例を示します。

   ```
   spec:
     channel: v3.1
     installPlanApproval: Automatic
     name: ibm-mq
     source: ibm-operator-catalog # CHANGE --> ibmmq-operator-catalogsource
     sourceNamespace: openshift-marketplace
   ```

   c) 変更を保存します。

   これで、 IBM MQ Operator のインストール済み環境が IBM MQ Operator カタログ・ソースを指すようになりました。 OperatorIBM を削除した場合、ステータスは"CatalogSourcenot found」から「Succeeded」に戻ります。

タスクの結果

これで、 IBM MQ Operator のインストール済み環境は、 IBM MQ Operator の特定のカタログ・ソースを指すようになります。 これにより、オペレーターのアップグレードを完全に制御できます。

OpenShift CP4I *CP4I ユーザーのための IBM MQ Operator のアップグレード*

IBM Cloud Pak for Integration (CP4I) ライセンスが使用されている IBM MQ Operator のデプロイメントをアップグレードします。

始める前に

重要 : このタスクは、 CP4I ユーザーを対象としています。 これには、 CP4I ライセンスの下に少なくとも 1 つのキュー・マネージャーをデプロイした場合が含まれます。 これが該当しない場合は、 69 ページの『IBM MQ Operator のアップグレード』 を参照してください。

このタスクについて

IBM MQ Operator CD バージョン（バージョン 3.6.0 にアップグレードする場合は、以下の説明に従って、まず v3.2 SC2 チャンネルにアップグレードする必要があります。

IBM MQ Operator のバージョンが 3.3.0 またはそれ以上の場合、 Upgrade to the CD Channel の手順の(c)のサブステップに進みます。

手順

1. オプション: **v3.2 SC2 チャンネルにアップグレードしてください。**

   IBM MQ Operator IBM MQ Operator の Support Cycle 2SC2)チャンネルにアップグレードするには、以下のオプションの 1 つ以上を完了してください：

   - **オプション 1:** 2.0.x Long Term SupportLTS)IBM MQ Operator のデプロイメントをアップグレードする。

     アップグレード計画の生成による 2022.2 からのアップグレードの手順に従います。
   - **オプション 2:** 3.0.x または 3.1.x の IBM MQ Operator をアップグレードする。

     アップグレード計画の生成による 2023.4 からのアップグレードの手順に従います。
   - **オプション 3:** IBM MQ Operator の他のデプロイメントをアップグレードする。

     Follow the relevant steps in 現在の CD チャンネルに移行する IBM MQ OperatorIBM MQ 9.3 ドキュメント), then return here and proceed with **オプション 2**. これは必須の前提条件ステップであることに注意してください。
2. **チャンネ CD(3.6*)にアップグレードする。**

   a) オプション: **3.2.x の展開にアップグレードする。 IBM MQ Operator**

   IBM MQ Operator を 3.2.x のデプロイメントにアップグレードするには、前のメインステップに従ってください。

   b) オプション: **IBM MQ Operator の 3.2.x デプロイメントをアップグレードする。**

   16.1.0 からのアップグレードの手順に従って、アップグレードプランを作成してください。

   c) オプション: **の 3.3.x 展開をアップグレードする。 IBM MQ Operator**

   お使いの IBM MQ Operator が現在バージョン 3.3.x の場合、3.6.0 へのアップグレードを続行する前に、以下の手順を実行して、オペレータを 3.4.x に更新してください。

   i) 鏡像（エアギャップのみ）。

   クラスタが制限された（エアギャップ）ネットワーク環境にある場合は、 IBM MQ のイメージをミラーリングする必要があります。 以下のリンク先のページで、これらの値のみを使用して手順を完了させてください

   ```
   export OPERATOR_PACKAGE_NAME=ibm-mq
   export OPERATOR_VERSION=3.4.1
   ```

   セクション 3.5 「クラスタの設定」は省略してください。イメージレジストリへの接続は、以前のインストールまたはアップグレード時に設定されているはずです。

   エアギャップインストールおよびアップグレードに使用される IBM MQ CASE の内容については、 295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。』 をご覧ください。

   リンク ： エアギャップクラスタのミラーリングイメージ。

   ii) IBM MQ Operator をアップグレードしてください。

   81 ページの『Red Hat OpenShift を使用した IBM MQ Operator のアップグレード』 の指示に従ってください。

   iii) インスタンスをアップグレードします。

最新の機能とセキュリティ修正プログラムを受け取るには、 IBM MQ オペランド（キューマネージャーのコンテナイメージ）をアップグレードしてください。 83 ページの『Red Hat OpenShift を使用した IBM MQ キュー・マネージャーのアップグレード』を参照してください。

d) **の 3.4.0 またはそれ以上のデプロイメントをアップグレードします。 IBM MQ Operator**

i) 鏡像（エアギャップのみ）。

クラスタが制限された（エアギャップ）ネットワーク環境にある場合は、 IBM MQ のイメージをミラーリングする必要があります。 以下のリンク先のページで、これらの値のみを使用して手順を完了させてください

```
export OPERATOR_PACKAGE_NAME=ibm-mq
export OPERATOR_VERSION=[operator_version]
```

[operator_version] は、アップグレード先の IBM MQ Operator のバージョンです。 アップグレード先のチャネルにおける最新の IBM MQ Operator を決定するには、 5 ページの『IBM MQ Operator のリリース履歴』 を参照してください。

セクション 3.5 「クラスタの設定」は省略してください。イメージレジストリへの接続は、以前のインストールまたはアップグレード時に設定されているはずです。

エアギャップインストールおよびアップグレードに使用される IBM MQ CASE の内容については、 295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』 をご覧ください。

リンク ： エアギャップクラスタのミラーリングイメージ。

ii) 現在のチャンネルで最新のオペレーターにアップグレードする。

70 ページの『IBM MQ Operator チャネルの最新セキュリティー・リリースへのアップグレード』のステップを実行します。

iii) IBM MQ Operator デプロイメントをアップグレードします。

アップグレードプランを生成して、 16.1.1 からのアップグレードの手順に従ってください。

OpenShift CP4I *Red Hat OpenShift を使用した IBM MQ Operator のアップグレード*

IBM MQ Operator」は「Red Hat OpenShift コンソールやコマンドラインからアップグレードできる。

始める前に

注 :

- IBM MQ Operator の最新版 CD は 3.6.0。
- IBM MQ Operator の最新版 SC2 は 3.2.13。

最新の IBM MQ Operator リリースノートについては、 IBM MQ Operator のリリース履歴をご覧ください。

手順

- **オプション 1 : Red Hat OpenShift コンソールを使用して IBM MQ Operator をアップグレードする。**

  a) Red Hat OpenShift クラスタコンソールにログインします。

  b) 鏡像（エアギャップのみ）。

  クラスタが制限された（エアギャップされた）ネットワーク環境にある場合は、以下の値を使用して「IBM MQ イメージをミラーリングする必要があります：

```
export OPERATOR_PACKAGE_NAME=ibm-mq
export OPERATOR_VERSION=3.6.0
```

エアギャップのインストールとアップグレードに使用される IBM MQ CASE の内容については、'295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照のこと。

ミラーイメージを作成するには、エアギャップクラスターのミラーイメージを参照してください。

c) 最新のカタログソースを適用する。

`ibm-operator-catalog`「IBM MQ Operator 用の特定のカタログソースを使用している場合は、新しい「IBM MQ バージョン用のカタログソースを適用する必要があります。

IBM オペレーターカタログの使用から、'IBM MQ Operator の特定のカタログソースの使用に移行し、アップグレードをより詳細に制御するには、次のステップを完了するために戻る前に、'78 ページの『IBM MQ Operator の特定のカタログ・ソースへの移動』のステップを参照してください。

IBMOperator カタログを使用している場合（一部のオンラインインストールのみ）、次のステップに進みます。

そうでない場合は、「60 ページの『IBM MQ Operator カタログ・ソースの追加』指示に従い、次のステップに進む。

d) IBM MQ Operator をアップグレードします。 IBM MQ Operator の新規のメジャー/マイナーのバージョンは、新しいサブスクリプション・チャネルを介して配信されます。 オペレータを新しいメジャー/マイナーバージョンにアップグレードするには、'IBM MQ Operator サブスクリプションで選択したチャネルを更新する必要があります。

a. ナビゲーション・ペインで、「**オペレーター (Operators)**」 > 「**インストール済みのオペレーター (Installed Operators)**」をクリックします。

指定したプロジェクトにインストールされているすべてのオペレーターが表示されます。

b.「**IBM MQ Operator**」を選択します。

c.「**Subscription**」タブにナビゲートします。

d.「**チャンネル**」をクリックします。

「**サブスクリプション更新チャネルの変更 (Change Subscription Update Channel)**」ウィンドウが表示されます。

e. 希望のチャンネルを選択し、「**保存**」をクリックします。

オペレーターは、新しいチャンネルで利用可能な最新バージョンにアップグレードする。 35 ページの『IBM MQ Operator のバージョン・サポート』 を参照してください。

- **オプション 2：'Red Hat OpenShift コマンドラインを使って'IBM MQ Operator アップグレードする。**

a) **`oc login`** を使用してクラスターにログインします。

b) 鏡像（エアギャップのみ）。

クラスタが制限された（エアギャップされた）ネットワーク環境にある場合は、以下の値を使用して「IBM MQ イメージをミラーリングする必要があります：

```
export OPERATOR_PACKAGE_NAME=ibm-mq
export OPERATOR_VERSION=3.6.0
```

エアギャップのインストールとアップグレードに使用される IBM MQ CASE の内容については、'295 ページの『IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。 』を参照のこと。

ミラーイメージを作成するには、エアギャップクラスターのミラーイメージを参照してください。

c) 最新のカタログソースを適用する。

`ibm-operator-catalog`「IBM MQ Operator 用の特定のカタログソースを使用している場合は、新しい「IBM MQ バージョン用のカタログソースを適用する必要があります。

IBM オペレーターカタログの使用から、'IBM MQ Operator の特定のカタログソースの使用に移行し、アップグレードをより詳細に制御するには、次のステップを完了するために戻る前に、'78 ページの『IBM MQ Operator の特定のカタログ・ソースへの移動』のステップを参照してください。

IBMOperator カタログを使用している場合（一部のオンラインインストールのみ）、次のステップに進みます。

そうでない場合は、「60 ページの『IBM MQ Operator カタログ・ソースの追加』指示に従い、次のステップに進む。

d) IBM MQ Operator をアップグレードします。 新規メジャー/マイナー IBM MQ Operator バージョンは、新規サブスクリプション・チャネルを使用して提供されます。 オペレータを新しいメジャーバージョンまたはマイナーバージョンにアップグレードするには、以下の手順を完了し、'IBM MQ Operator サブスクリプションで選択したチャネルを更新します。

a. 必要な IBM MQ Operator アップグレード・チャネルが使用可能であることを確認します。

```
oc get packagemanifest ibm-mq -o=jsonpath='{.status.channels[*].name}'
```

b. Subscription にパッチを適用して目的の更新チャネルに移動します (*vX.Y* は前のステップで識別された目的の更新チャネルです)。

```
oc patch subscription ibm-mq --patch '{"spec":{"channel":"vX.Y"}}' --type=merge
```

OpenShift CP4I *Red Hat OpenShift を使用した IBM MQ キュー・マネージャーのアップグレード*

IBM MQ Operator 使用してデプロイされた IBM MQ キュー・マネージャーは、以下のインターフェースのいずれかを使用して Red Hat OpenShift アップグレードできます：オペレーター・ハブ; コマンドライン;IBM Cloud Pak for Integration Platform UI.

始める前に

IBM MQ キュー・マネージャーをアップグレードするプロセスの一環として、 IBM Cloud Pak for Integration の資料からこのトピックに移動した可能性があります。

注 :

- The latest CD version of the IBM MQ queue manager is 9.4.3.0-r1.
- The latest SC2 version of the IBM MQ queue manager is 9.4.0.11-r3.

For the latest IBM MQ queue manager release notes, see 18 ページの『キュー・マネージャ・コンテナ・イメージのリリース履歴』.

あなたの IBM MQ Operator が必要なバージョンであることを確認してください。 IBM MQ Operator アップグレードするには、69 ページの『IBM MQ Operator のアップグレード』を参照してください。

エアギャップ環境でキュー・マネージャをアップグレードする前に、IBM MQ Operator のアップグレード中に完了するエアギャップ特有の手順で、最新の IBM Cloud Pak for Integration イメージをミラーリングする必要があります。 これらの手順の例については、75 ページの『Upgrading to the current CD channel of the IBM MQ Operator』を参照のこと。

このタスクについて

このタスクは、OpenShift・マネージャーをアップグレードするための 3 つのオプションを提供します IBM MQ

1. Upgrading an IBM MQ queue manager using the Red Hat OpenShift console.
2. Red Hat OpenShift を使用したキュー・IBM MQ のアップグレード
3. Upgrading an IBM MQ queue manager in Red Hat OpenShift using the Platform UI

手順

- **オプション 1**: Upgrading an IBM MQ queue manager using the Red Hat OpenShift console.
  a) Red Hat OpenShift クラスターのウェブコンソールにログインします。

b) ナビゲーション・ペインで、「**オペレーター (Operators)**」 > 「**インストール済みのオペレーター (Installed Operators)**」をクリックします。
指定したプロジェクトにインストールされているすべてのオペレーターが表示されます。

c) 「**IBM MQ Operator**」を選択します。
「**IBM MQ Operator**」ウィンドウが表示されます。

d) 「**キュー・マネージャー**」タブにナビゲートします。
「**キュー・マネージャーの詳細**」ウィンドウが表示されます。

e) アップグレードするキュー・マネージャーを選択します。

f) YAML タブに移動します。

g) 以下のフィールドを更新します。必要に応じて、対象の IBM MQ キュー・マネージャーのバージョン・アップグレードに合わせてください。

- spec.version
- spec.license.licence

バージョンとキュー・マネージャ IBM MQ コンテナ・イメージのマッピングについては 18 ページの『キュー・マネージャ・コンテナ・イメージのリリース履歴』を参照のこと IBM MQ Operator

h) 更新したキュー・マネージャー YAML を保存します。

- **オプション 2**: CLIRed Hat OpenShift を使用したキュー IBM MQ マネージャのアップグレード

a) Log in to the Red Hat OpenShift command line interface (CLI) using `oc login`.

b) **QueueManager** リソースを編集して以下のフィールドを更新します。必要に応じて、対象の IBM MQ キュー・マネージャーのバージョン・アップグレードに合わせてください。

- spec.version
- spec.license.licence

チャンネルと IBM MQ Operator バージョンおよび IBM MQ キュー・マネージャのバージョンの対応付けについては 35 ページの『IBM MQ Operator のバージョン・サポート』を参照のこと。

以下のコマンドを使用します。

```
oc edit queuemanager my_qmgr
```

ここで、*my_qmgr* は、アップグレードする QueueManager リソースの名前です。

- **オプション 3**: Upgrading an IBM MQ queue manager in Red Hat OpenShift using the Platform UI.

IBM Cloud Pak for Integration の「Managing versions and upgrades by using Platform UI」の説明に従ってください。

OpenShift CP4I IBM MQ Operator のアンインストール

Red Hat OpenShift コンソールまたは CLI を使用して、Red Hat OpenShift から IBM MQ Operator をアンインストールできます。

手順

- オプション 1: OpenShift コンソールを使用して IBM MQ Operator をアンインストールします。

注 : クラスター上のすべてのプロジェクト/名前空間に IBM MQ Operator がインストールされている場合は、キュー・マネージャーを削除するプロジェクトごとに、以下の手順のステップ 2 から 6 を繰り返します。

a) Red Hat OpenShift Container Platform クラスター管理者資格情報を使用して、 Red Hat OpenShift Container Platform Web コンソールにログインします。

b) 「**プロジェクト**」 を、 IBM MQ Operator をアンインストールする名前空間に変更します。 「**プロジェクト**」 ドロップダウン・リストから名前空間を選択します。

c) ナビゲーション・ペインで、「**オペレーター**」 > 「**インストール済みオペレーター**」をクリックします。

d)「**IBM MQ**」オペレーターをクリックします。

e)「**キュー・マネージャー**」タブをクリックして、この IBM MQ Operator によって管理されているキュー・マネージャーを表示します。

f) 1 つ以上のキュー・マネージャーを削除します。

これらのキュー・マネージャーは引き続き稼働しますが、IBM MQ Operator がない状態では正常に機能しない場合があることに注意してください。

g) オプション: 必要に応じて、キュー・マネージャーを削除するプロジェクトごとにステップ 2 から 6 を繰り返します。

h)「**オペレーター (Operators)**」 > 「**インストール済みのオペレーター (Installed Operators)**」に戻ります。

i)「**IBM MQ**」オペレーターの横にある 3 点メニューをクリックし、「**オペレーターのアンインストール (Uninstall Operator)**」を選択します。

- オプション 2: OpenShift CLI を使用して IBM MQ Operator をアンインストールする

a) `oc login` を使用して Red Hat OpenShift クラスターにログインします。

b) IBM MQ Operator が 1 つの名前空間にインストールされている場合は、以下のサブステップを実行します。

a. アンインストールする IBM MQ Operator が含まれているプロジェクトにいることを確認します。

```
oc project project_name
```

b. プロジェクトにインストールされているキュー・マネージャーを表示します。

```
oc get qmgr
```

c. 1 つ以上のキュー・マネージャーを削除します。

```
oc delete qmgr qmgr_name
```

これらのキュー・マネージャーは引き続き稼働しますが、IBM MQ Operator がない状態では正常に機能しない場合があることに注意してください。

d. **ClusterServiceVersion** インスタンスを表示します。

```
oc get csv
```

e. IBM MQ **ClusterServiceVersion** を削除します。

```
oc delete csv ibm_mq_csv_name
```

f. サブスクリプションを表示します。

```
oc get subscription
```

g. すべてのサブスクリプションを削除します。

```
oc delete subscription ibm_mq_subscription_name
```

h. 共通サービスを使用しているものがほかにない場合は、共通サービス・オペレーターをアンインストールし、オペレーター・グループを削除できます。

i) IBM Cloud Pak foundational services 製品資料の「Uninstalling foundational services」の説明に従って、共通サービス・オペレーターをアンインストールします。

ii) オペレーター・グループを表示します。

```
oc get operatorgroup
```

iii) オペレーター・グループを削除します。

```
oc delete OperatorGroup operator_group_name
```

c) IBM MQ Operator がクラスター上のすべての名前空間にインストールされて使用可能な状態になっている場合は、以下のサブステップを実行します。

a. インストールされているすべてのキュー・マネージャーを表示します。

```
oc get qmgr -A
```

b. 1 つ以上のキュー・マネージャーを削除します。

```
oc delete qmgr qmgr_name -n namespace_name
```

これらのキュー・マネージャーは引き続き稼働しますが、IBM MQ Operator がない状態では正常に機能しない場合があることに注意してください。

c. **ClusterServiceVersion** インスタンスを表示します。

```
oc get csv -A
```

d. クラスターから IBM MQ **ClusterServiceVersion** を削除します。

```
oc delete csv ibm_mq_csv_name -n openshift-operators
```

e. サブスクリプションを表示します。

```
oc get subscription -n openshift-operators
```

f. サブスクリプションを削除します。

```
oc delete subscription ibm_mq_subscription_name -n openshift-operators
```

g. オプション: 共通サービスを使用しているものが他にない場合は、共通サービス・オペレーターをアンインストールすることができます。 これを行うには、 IBM Cloud Pak foundational services 製品資料の「 基本サービスのアンインストール 」の説明に従ってください。

Kubernetes V 9.4.1 プルシークレットを作成して「Kubernetes」を使用する準備

ビルド済みの「IBM MQ Advanced イメージは「IBM Container Registry」から引き出すことができるが、レジストリはライセンスのエンタイトルメントチェックを実行する。 このチェックには、「docker-registry プルシークレットに格納されているエンタイトルメント・キーが必要である。

始める前に

kubectl コマンドラインツールをインストールし、'Kubernetes クラスタにログインする。

手順

1. IBMID に割り当てられたエンタイトルメント・キーを取得する。
   a) MyIBMContainer Software Library に、権利のあるソフトウェアに関連付けられている「IBMID とパスワードでログインします。
   b)「**使用権キー (Entitlement keys)**」セクションで「**キーのコピー (Copy key)**」を選択して、使用権キーをクリップボードにコピーします。
2. エンタイトルメントキーのシークレットを作成する。
   a. OpenShiftCLI から以下のコマンドを実行し、'ibm-entitlement-key というイメージのプルシークレットを作成する。

```
kubectl create secret docker-registry ibm-entitlement-key \
--docker-server=cp.icr.io \
--docker-username=cp \
--docker-password=entitlement_key \
```

```
--docker-email=user_email \
--namespace=namespace
```

ここで、*entitlement_key* はステップ b でコピーしたエンタイトルメント・キー、*user_email* は権利を持つソフトウェアに関連付けられた「IBMID、*namespace* はキュー・マネージャに使用する予定の名前空間です。

Kubernetes をインストールして Kubernetes を使う準備をする。 IBM License Service

コンテナライセンシングでは、コンテナが実行されているサーバー全体のライセンスを必要とするのではなく、個々の IBM MQ コンテナの利用可能な容量のみをライセンスすることができます。 コンテナ ライセンスを利用するには、 IBM License Service を使用してライセンスの使用状況を追跡し、必要なエンタイトルメントを決定する必要があります。

手順

IBM Cloud Pak foundational services ドキュメントの「 Installing License Service on Kubernetes cluster 」の説明に従って、 IBM License Service をインストールします。

Linux V 9.4.1 Podman または「Docker 使用するために、ビルド済みのコンテナイメージを取り込む

Podman や'Docker などのコマンドラインツールを使って、ビルド済みの'IBM MQ Advanced イメージを'IBM Container Registry から引き出すことができる。

始める前に

Podman または'Docker をインストールする。

注 : 以下のコマンド例では「Podman 使用しているが、同じコマンドパラメーターで「Podman または「Docker」を使用することができる。

手順

1. IBM ID に割り当てられているエンタイトルメントキーを取得します。
   a) MyIBM コンテナ ソフトウェア ライブラリに、権利を持つソフトウェアに関連付けられている IBM ID とパスワードでログインします。
   b)「**使用権キー (Entitlement keys)**」セクションで「**キーのコピー (Copy key)**」を選択して、使用権キーをクリップボードにコピーします。
2. コンテナ・レジストリにログインする

   例えば、'Podman ：

   ```
   podman login --username cp --password ... cp.icr.io
   ```

   どこに「...」があるのか は、前のステップでコピーしたエンタイトルメント・キーに置き換えられる。
3. 画像を引き出す

   例えば、'Podman ：

   ```
   podman pull cp.icr.io/cp/ibm-mqadvanced-server:9.4.3.0-r1
   ```

独自のコンテナー・イメージのビルドによる IBM MQ の準備

自作コンテナーを開発します。 これは最も柔軟なコンテナソリューションですが、コンテナの設定に強いスキルが必要であり、結果としてのコンテナを"所有する"必要があります。

始める前に

独自のコンテナーを開発する前に、代わりに IBM MQ Operator を使用できるかどうかを検討してください。 28 ページの『コンテナー内の IBM MQ の使用方法の選択』 を参照してください。

このタスクについて

手順

- 88 ページの『独自のキュー・マネージャー・イメージを作成する際の一般的な考慮事項』
- 88 ページの『サンプルの IBM MQ キュー・マネージャーのコンテナー・イメージのビルド』
- 91 ページの『別々のコンテナーでのローカル・バインディング・アプリケーションの実行』
- IBM MQ サンプルの Helm チャートを確認します。

独自のキュー・マネージャー・イメージを作成する際の一般的な考慮事項

コンテナーで IBM MQ キュー・マネージャーを実行するには、考慮すべき要件がいくつかあります。 サンプルのコンテナー・イメージは、これらの要件に対応できるようになっていますが、独自のイメージを使用する場合は、これらの要件に対応する方法を検討する必要があります。

プロセス監視

コンテナーを実行することは、基本的に単一のプロセス (コンテナー内部の PID 1) を実行することになります。 この単一のプロセスは、後で子プロセスを spawn することができます。

メインプロセスが終了すると、コンテナー・ランタイムがコンテナーを停止します。 IBM MQ キュー・マネージャーでは、複数のプロセスをバックグラウンドで実行する必要があります。

このため、キュー・マネージャーの実行中は、メインプロセスがアクティブな状態であることを確認する必要があります。 グッド・プラクティスとして、例えば管理照会などを実行して、キュー・マネージャーがアクティブであることをこのプロセスから確認してください。

/var/mqm への移植

コンテナーは、 `/var/mqm` をボリュームとして構成する必要があります。

これを行うと、コンテナーが最初に始動したときに、このボリュームのディレクトリーは空の状態です。 通常、このディレクトリーにはインストール時に内容が取り込まれますが、コンテナーを使用する場合は、インストールとランタイムが別々の環境になります。

これを解決するために、コンテナーが始動して初めて実行されるときに、**crtmqdir** コマンドを使用して、`/var/mqm` にデータを取り込みます。

コンテナーのセキュリティー

ランタイムのセキュリティー要件を最小限にするために、サンプルのコンテナー・イメージは、unzip 可能な IBM MQ インストールを使用してインストールされます。 これにより、`setuid` ビットが設定されなくなり、コンテナーは特権エスカレーションを使用する必要がなくなります。 コンテナー・システムの中には、使用できるユーザー ID を定義したものもありますが、この unzip 可能なインストールでは、使用可能なオペレーティング・システム・ユーザーについて一切想定されていません。

サンプルの IBM MQ キュー・マネージャーのコンテナー・イメージのビルド

コンテナーで IBM MQ キュー・マネージャーを実行するためにサンプルのコンテナー・イメージをビルドするには、この情報を活用してください。

このタスクについて

まず、Red Hat Universal Base Image ファイル・システムと IBM MQ のクリーン・インストールが含まれているベース・イメージをビルドします。

次に、そのベースの上に、ユーザー ID とパスワードによる基本的なセキュリティーを可能にする IBM MQ 構成を追加するための別のコンテナー・イメージ層をビルドします。

最後に、このイメージをファイル・システムとして使用し、ホスト・ファイル・システム上のコンテナー固有のボリュームによって提供される `/var/mqm` の内容を使用してコンテナーを実行します。

手順

- コンテナーで IBM MQ キュー・マネージャーを実行するためにサンプルのコンテナー・イメージをビルドする方法については、以下のサブトピックを参照してください。
  - 89 ページの『サンプルの IBM MQ キュー・マネージャーのベース・イメージのビルド』
  - 89 ページの『サンプルの構成済みの IBM MQ キュー・マネージャーのイメージのビルド』

サンプルの *IBM MQ* キュー・マネージャーのベース・イメージのビルド

独自のコンテナー・イメージの IBM MQ を使用するためには、まず、IBM MQ クリーン・インストールを含むベース・イメージをビルドする必要があります。 以下の手順は、GitHub でホストされているサンプル・コードを使用してサンプルのベース・イメージをビルドする方法を示しています。

手順

- mq-container GitHub リポジトリーで提供されている Make ファイルを使用して実稼働用のコンテナー・イメージをビルドします。

  GitHub の コンテナー・イメージの作成の指示に従います。
- オプション: Red Hat OpenShift Container Platform "restricted" Security Context Constraint (SCC) を使用してセキュア・アクセスを構成する予定の場合は、 IBM MQ 非インストール・イメージの 1 つを使用してください。

  これらのイメージをダウンロードするためのリンクは、 IBM MQ ダウンロードの「コンテナー」セクションにあります。

タスクの結果

IBM MQ がインストールされたベース・コンテナー・イメージができました。

これで、構成済みのサンプル IBM MQ キュー・マネージャー・イメージを作成する準備ができました。

サンプルの構成済みの *IBM MQ* キュー・マネージャーのイメージのビルド

汎用的なベースの IBM MQ コンテナー・イメージをビルドしたら、セキュアなアクセスを可能にするために独自の構成を適用する必要があります。 これを行うには、汎用イメージを親として使用して、独自のコンテナー・イメージ層を作成します。

始める前に

このタスクは、サンプルの基本 IBM MQ キュー・マネージャー・イメージの作成時に "インストールなし" の IBM MQ パッケージを使用したことが前提になっています。 そうでない場合は、Red Hat OpenShift Container Platform の "制限付き" のセキュリティー・コンテキスト制約 (SCC) を使用してセキュア・アクセスを構成することができなくなります。 デフォルトで使用される "制限付き" の SCC では、ランダムのユーザー ID が使用され、別のユーザーへの変更による特権のエスカレーションが不可になります。 IBM MQ の従来の RPM ベースのインストーラーは、`mqm` のユーザーとグループに依存していて、実行可能プログラムで `setuid` ビットを使用します。 現行バージョンの IBM MQ では、 "No-Install" IBM MQ パッケージを使用すると、 `mqm` ユーザーも `mqm` グループも存在しなくなります。

手順

1. 新しいディレクトリーを作成し、以下を内容とする config.mqsc というファイルを追加します。

```
DEFINE QLOCAL(EXAMPLE.QUEUE.1) REPLACE
```

   上記の例では、ユーザー ID とパスワードによる単純な認証が使用されることに注意してください。 ただし、企業が必要とする任意のセキュリティー構成を適用できます。

2. 以下を内容とする Dockerfile というファイルを作成します。

```
FROM mq
COPY config.mqsc /etc/mqm/
```

3. 次のコマンドを使用して、カスタム・コンテナー・イメージをビルドします。

```
docker build -t mymq .
```

   「.」は、先ほど作成した 2 つのファイルが含まれているディレクトリーです。

   Docker はそのイメージを使用して一時コンテナーを作成し、残りのコマンドを実行します。

   注 : Red Hat Enterprise Linux (RHEL) では、コマンド **docker** (RHEL V7) または **podman** (RHEL V7 または RHEL V8) を使用します。 Linux では、コマンドの先頭に **sudo** を指定して **docker** コマンドを実行し、追加の特権を取得する必要があります。

4. カスタマイズした新しいイメージを実行して、先ほど作成したディスク・イメージを含む新しいコンテナーを作成します。

   新しいイメージ層では、実行する特定のコマンドを指定しなかったので、親イメージから継承されています。 親のエントリー・ポイント (このコードは GitHub で提供されています):

   - キュー・マネージャーを作成する
   - キュー・マネージャーを開始する
   - デフォルトのリスナーを作成する
   - 次に、 /etc/mqm/config.mqsc.から MQSC コマンドを実行します

   以下のコマンドを発行して、カスタマイズした新しいイメージを実行します。

```
docker run \
  --env LICENSE=accept \
  --env MQ_QMGR_NAME=QM1 \
  --volume /var/example:/var/mqm \
  --publish 1414:1414 \
  --detach \
  mymq
```

   説明:

   最初の env パラメーター
   IBM IBM WebSphere® MQ のライセンスに同意したことを知らせる環境変数をコンテナーに渡しています。 LICENSE 変数を view に設定してライセンスを表示することもできます。

   IBM MQ ライセンスについて詳しくは、 IBM MQ ライセンス情報 を参照してください。

   2 番目の env パラメーター
   使用するキュー・マネージャー名を設定しています。

   volume パラメーター
   MQ が /var/mqm 上で実際に /var/example に書き込む必要があることをコンテナーに指示します。

   このオプションは、後で永続データを保持したままコンテナーを簡単に削除できることを意味します。 このオプションにより、ログ・ファイルの表示も簡単になります。

publish パラメーター
ホスト・システムのポートをコンテナーのポートにマップしています。 デフォルトでは、コンテナーはコンテナー自身の内部 IP アドレスを使用して実行されます。つまり、ポートを公開するには、具体的にポートをマップする必要があります。

この例では、ホストのポート 1414 をコンテナーのポート 1414 にマップしています。

detach パラメーター
バックグラウンドでコンテナーを実行します。

タスクの結果

構成したコンテナー・イメージをビルドしたので、**docker ps** コマンドを使用して、実行中のコンテナーを表示できます。 **docker top** コマンドを使用して、コンテナーで実行されている IBM MQ プロセスを表示できます。

重要 :

docker logs ${CONTAINER_ID} コマンドを使用して、コンテナーのログを表示できます。

次のタスク

- **docker ps** コマンドを使用してもコンテナーが表示されない場合は、コンテナーが失敗している可能性があります。 **docker ps -a** コマンドを使用して、失敗したコンテナーを表示できます。
- **docker ps -a** コマンドを使用すると、コンテナー ID が表示されます。 この ID は、**docker run** コマンドを発行したときにも表示されたものです。
- **docker logs ${CONTAINER_ID}** コマンドを使用して、コンテナーのログを表示できます。

別々のコンテナーでのローカル・バインディング・アプリケーションの実行

コンテナー間でプロセス名前空間を共有することにより、 IBM MQ キュー・マネージャーとは別のコンテナーにある IBM MQ へのローカル・バインディング接続を必要とするアプリケーションを実行できます。

このタスクについて

以下の制約事項に従う必要があります。

- `--pid` 引数を使用して、各コンテナーの PID 名前空間を共有する必要があります。
- `--ipc` 引数を使用して、各コンテナーの IPC 名前空間を共有する必要があります。
- 以下のいずれかが必要です。
  1. `--uts` 引数を使用して、各コンテナーの UTS 名前空間をホストと共有する。
  2. `-h` 引数または `--hostname` 引数を使用して、各コンテナーを同じホスト名にする。
- IBM MQ データディレクトリは、 `/var/mqm` ディレクトリの下にあるすべてのコンテナが利用可能なボリュームにマウントする必要があります。

以下の例では、サンプルの IBM MQ コンテナー・イメージを使用しています。 このイメージの詳細については、Github を参照してください。

手順

1. 次のコマンドを発行して、ボリュームとして機能する一時ディレクトリーを作成します。

```
mkdir /tmp/dockerVolume
```

2. 次のコマンドを発行して、1 つのコンテナー内にキュー・マネージャー (`QM1`) を作成し、名前 `sharedNamespace` を指定します。

```
docker run -d -e LICENSE=accept -e MQ_QMGR_NAME=QM1 --volume /tmp/dockerVol:/mnt/mqm
--uts host --name sharedNamespace ibmcom/mq
```

3. 次のコマンドを発行して、ibmcom/mq をベースとする secondaryContainer という 2 つ目のコンテナーを始動します。ただし、キュー・マネージャーは作成しません。

```
docker run --entrypoint /bin/bash --volumes-from sharedNamespace --pid
container:sharedNamespace --ipc container:sharedNamespace --uts host --name
secondaryContainer -it --detach ibmcom/mq
```

4. 次のコマンドを発行して、2 つ目のコンテナーに対して **dspmq** コマンドを実行し、両方のキュー・マネージャーの状況を調べます。

```
docker exec secondaryContainer dspmq
```

5. 次のコマンドを実行して、もう一方のコンテナーで実行されているキュー・マネージャーに対する MQSC コマンドを処理します。

```
docker exec -it secondaryContainer runmqsc QM1
```

タスクの結果

これで、別々のコンテナーでローカル・アプリケーションが実行されるようになりました。**dspmq**、**amqsput**、**amqsget**、**runmqsc** などのコマンドを、QM1 キュー・マネージャーへのローカル・バインディングとして、2 つ目のコンテナーから正常に実行することができます。

予期した結果にならない場合は、92 ページの『名前空間アプリケーションのトラブルシューティング』で詳細を参照してください。

名前空間アプリケーションのトラブルシューティング

共有の名前空間を使用する場合は、すべての名前空間 (IPC、PID、UTS/ホスト名) およびマウント・ボリュームを共有する必要があります。そうしないと、アプリケーションは機能しません。

従う必要がある制約事項のリストについては、91 ページの『別々のコンテナーでのローカル・バインディング・アプリケーションの実行』を参照してください。

リストされているすべての制約事項をアプリケーションが満たしていないと、コンテナーが始動しても、予期したように機能しないという問題が発生する可能性があります。

以下のリストに、一般的な原因と、制約事項のいずれかを満たすことを忘れた場合に見られる動作をまとめています。

- コンテナーの名前空間 (UTS/PID/IPC)、またはホスト名のいずれかを共有することを忘れた状態でボリュームをマウントした場合、コンテナーは、キュー・マネージャーを認識できますが、キュー・マネージャーと対話することができません。
  - **dspmq** コマンドの場合は、以下のようになります。

```
docker exec container dspmq

QMNAME(QM1)                          STATUS(Status not available)
```

  - **runmqsc** コマンドなど、キュー・マネージャーに接続しようとするコマンドの場合は、AMQ8146 エラー・メッセージを受け取るはずです。

```
docker exec -it container runmqsc QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2025.
Starting MQSC for queue manager QM1.
AMQ8146: IBM MQ queue manager not available
```

- すべての必要な名前空間を共有しているが、共有ボリュームを /var/mqm ディレクトリーにマウントせず、有効な IBM MQ データ・パスを持っている場合は、コマンドも アンキ 8146 のエラー・メッセージを受け取ります。

しかし、**dspmq** は、キュー・マネージャーをまったく認識できないので、代わりに空の応答を返します。

```
docker exec container dspmq
```

- 必要なすべての名前空間を共有しているが、共有ボリュームを /var/mqm ディレクトリーにマウントせず、有効な IBM MQ データ・パス (または IBM MQ データ・パスなし) を持っていない場合は、データ・パスが IBM MQ インストールのキー・コンポーネントであるため、さまざまなエラーが表示されます。データ・パスがなければ、IBM MQ は動作できません。

  以下のコマンドのいずれかを実行したときに、これらの例に示すような応答が表示される場合は、ディレクトリーをマウントしたこと、または IBM MQ データ・ディレクトリーを作成したことを確認する必要があります。

```
docker exec container dspmq
'No such file or directory' from /var/mqm/mqs.ini
AMQ6090: IBM MQ was unable to display an error message FFFFFFFF.
AMQffff

docker exec container dspmqver
AMQ7047: An unexpected error was encountered by a command. Reason code is 0.

docker exec container mqrc
<file path>/mqrc.c[1152]
lpiObtainQMDetails --> 545261715

docker exec container crtmqm QM1
AMQ8101: IBM MQ error (893) has occurred.

docker exec container strmqm QM1
AMQ6239: Permission denied attempting to access filesystem location '/var/mqm'.
AMQ7002: An error occurred manipulating a file.

docker exec container endmqm QM1
AMQ8101: IBM MQ error (893) has occurred.

docker exec container dltmqm QM1
AMQ7002: An error occurred manipulating a file.

docker exec container strmqweb
<file path>/mqrc.c[1152]
lpiObtainQMDetails --> 545261715
```

MQ Adv. Kubernetes 独自のコンテナーを作成する場合のネイティブ HA グループの作成

ネイティブ HA グループを作成するには、3 つのキュー・マネージャーを作成、構成、および開始する必要があります。

このタスクについて

ネイティブ HA ソリューションを作成するために推奨される方法は、 IBM MQ 演算子を使用することです (ネイティブ HA を参照)。 あるいは、独自のコンテナーを作成する場合は、以下の手順に従うことができます。

ネイティブ HA グループを作成するには、ログ・タイプを `log replication` に設定して、3 つのノードに 3 つのキュー・マネージャーを作成します。 次に、各キュー・マネージャーの `qm.ini` ファイルを編集して、3 つのノードのそれぞれの接続の詳細を追加し、ログ・データを相互に複製できるようにします。

その後、3 つすべてのキュー・マネージャーを開始して、3 つすべてのインスタンスが相互に通信できることを確認し、どのインスタンスがアクティブ・インスタンスになり、どのインスタンスがレプリカになるかを判別できるようにする必要があります。

注 : この方法で独自のコンテナーにネイティブ HA グループを作成できるのは、 Kubernetes または Red Hat OpenShift を実行している場合のみです。

手順

1. 3 つのノードのそれぞれで、ログ・レプリカのログ・タイプを指定し、各ログ・インスタンスに固有の名前を指定して、キュー・マネージャーを作成します。 各キュー・マネージャーの名前は同じです。

```
crtmqm -lr instance_name qmname
```

以下に例を示します。

```
node 1> crtmqm -lr qm1_inst1 qm1

node 2> crtmqm -lr qm1_inst2 qm1

node 3> crtmqm -lr qm1_inst3 qm1
```

2. 各キュー・マネージャーが正常に作成されると、 NativeHALocalInstance という名前のスタンザがキュー・マネージャー構成ファイル qm.ini に追加されます。 指定されたインスタンス名を指定する Name 属性がスタンザに追加されます。

   オプションで、以下の属性を qm.ini ファイルの NativeHALocalInstance スタンザに追加できます。

   KeyRepository
   ログ複製トラフィックの保護のために、使用するデジタル証明書を保持する鍵リポジトリーのロケーション。 位置は語幹形式で指定されます。つまり、拡張子なしの絶対パスとファイル名が含まれます。 KeyRepository スタンザ属性を省略すると、インスタンス間でログ複製データがプレーン・テキストで交換されます。

   CertificateLabel
   ログ複製トラフィックの保護に使用するデジタル証明書を識別する証明書ラベル。
   KeyRepository が指定されているが、 CertificateLabel が省略されている場合は、デフォルト値 ibmwebspheremq*queue_manager* が使用されます。

   CipherSpec
   ログ複製トラフィックを保護するために使用する MQCipherSpec。 このスタンザ属性を指定する場合は、 KeyRepository も指定する必要があります。 KeyRepository が指定されているが、 CipherSpec が省略されている場合は、デフォルト値 ANY が使用されます。

   LocalAddress
   ログ複製トラフィックを受け入れるローカル・ネットワーク・インターフェース・アドレス。 このスタンザ属性が指定されている場合は、「[addr] [(port)]」の形式を使用してローカル・ネットワーク・インターフェースまたはポート (あるいはその両方) を識別します。 ネットワーク・アドレスは、ホスト名、 IPv4 小数点付き 10 進数、または IPv6 16 進形式で指定できます。 この属性を省略すると、キュー・マネージャーはすべてのネットワーク・インターフェースへのバインドを試行し、ローカル・インスタンス名と一致する NativeHAInstances スタンザの ReplicationAddress に指定されているポートを使用します。

   HeartbeatInterval
   ハートビート間隔は、ネイティブ HA キュー・マネージャーのアクティブ・インスタンスがネットワーク・ハートビートを送信する頻度をミリ秒単位で定義します。 ハートビート間隔値の有効範囲は 500 (0.5 秒) から 60000 (1 分) で、この範囲外の値を使用するとキュー・マネージャーの開始が失敗します。 この属性を省略すると、デフォルト値の 5000 (5 秒) が使用されます。 各インスタンスで同じハートビート間隔を使用する必要があります。

   HeartbeatTimeout
   ハートビート・タイムアウトは、ネイティブ HA キュー・マネージャーのレプリカ・インスタンスが、アクティブ・インスタンスが応答しないと判断するまで待機する時間の長さを定義します。 ハートビート間隔タイムアウト値の有効範囲は 500 (0.5 秒) から 120000 (2 分) です。 ハートビート・タイムアウトの値は、ハートビート間隔以上でなければなりません。

   無効な値を指定すると、キュー・マネージャーの開始が失敗します。 この属性が省略されると、レプリカは、新しいアクティブ・インスタンスを選択するプロセスを開始する前に 2 x HeartbeatInterval 待機します。 各インスタンスで同じハートビート・タイムアウトを使用する必要があります。

RetryInterval
再試行間隔は、障害が発生した複製リンクがネイティブ HA キュー・マネージャーで再試行される頻度をミリ秒単位で定義します。 再試行間隔の有効範囲は 500 (0.5 秒) から 120000 (2 分) です。 この属性が省略されると、レプリカは、障害が発生した複製リンクを再試行する前に 2 x HeartbeatInterval 待機します。

3. 各キュー・マネージャーの qm.ini ファイルを編集し、接続の詳細を追加します。 ネイティブ HA グループ内のキュー・マネージャー・インスタンス (ローカル・インスタンスを含む) ごとに 1 つずつ、合計 3 つの NativeHAInstance スタンザを追加します。 以下の属性を追加します。

   名前
   キュー・マネージャー・インスタンスの作成時に使用したインスタンス名を指定します。

   ReplicationAddress
   インスタンスのホスト名、IPv4 ドット 10 進または IPv6 16 進形式のアドレスを指定します。 アドレスは、ホスト名、 IPv4 小数点付き 10 進数、または IPv6 16 進形式のアドレスとして指定できます。 複製アドレスは、グループ内の各インスタンスから解決可能かつルーティング可能でなければなりません。 ログ複製に使用するポート番号は、大括弧で囲んで指定する必要があります。以下に例を示します。

   ```
   ReplicationAddress=host1.example.com(4444)
   ```

   注: NativeHAInstance スタンザはすべてのインスタンスで同じであり、自動構成 (**crtmqm -ii**) を使用して提供できます。

4. 以下の 3 つのインスタンスのそれぞれを開始します。

   ```
   strmqm QMgrName
   ```

   インスタンスが開始されると、3 つのインスタンスがすべて実行中であることを確認するために通信が行われます。次に、3 つのインスタンスのうちどちらがアクティブ・インスタンスであるかを決定し、残りの 2 つのインスタンスは引き続きレプリカとして実行されます。

例

以下の例は、3 つのインスタンスのいずれかに必要なネイティブ HA の詳細を指定する qm.ini ファイルのセクションを示しています。

```
NativeHALocalInstance:
  LocalName=node-1

NativeHAInstance:
  Name=node-1
  ReplicationAddress=host1.example.com(4444)
NativeHAInstance:
  Name=node-2
  ReplicationAddress=host2.example.com(4444)
NativeHAInstance:
  Name=node-3
  ReplicationAddress=host3.example.com(4444)
```

V 9.4.2 MQ Adv. Kubernetes 独自のコンテナを作成する場合のネイティブ HA CRR 構成の作成

ネイティブ HA グループを作成するには、3 つのキューマネージャを作成、設定、起動する必要があります。

このタスクについて

Native HA CRR ソリューションを作成する推奨される方法は、 IBM MQ オペレーターを使用することです (115 ページの『ネイティブ HA CRR の設定に IBM MQ Operator』 を参照)。 あるいは、独自のコンテナを作成する場合は、こちらの指示に従ってください。

ネイティブ HA CRR 構成を作成するには、キューマネージャに対して 2 つのネイティブ HA グループを作成する必要があります。1 つは「ライブ」サイトに、もう 1 つは「リカバリ」サイトに作成します。 次に、

キューマネージャーがそれらの間でクロスリージョンレプリケーションを設定できるように、 qm.ini ファイルを編集します。

注 : Kubernetes または Red Hat OpenShift を実行している場合のみ、この方法で自身のコンテナ内にネイティブ HA CRR 構成を作成できます。

手順

1. ライブサイトでネイティブ HA グループを作成します。 93 ページの『独自のコンテナーを作成する場合のネイティブ HA グループの作成』 を参照してください。
2. リカバリサイトで、2 つ目のネイティブ HA グループを作成します。 93 ページの『独自のコンテナーを作成する場合のネイティブ HA グループの作成』 を参照してください。
3. キューマネージャの qm.ini ファイルに以下の変更を加えることで、ライブサイトとリカバリサイトの 2 つのグループがネイティブ HA CRR 構成で連携して動作するように指定します。

   リカバリグループでは、 NativeHALocalInstance のスタンザで以下の値を指定します

   GroupName
   リカバリグループの名前を指定します。例えば、「NHACRR_recovery」などです。

   GroupRole
   グループの現在の役割を指定します。 この場合、「リカバリ」に設定します。

   NativeHARecoveryGroup のスタンザに以下の値を指定します

   GroupName
   ライブサイト上のグループ名、例えば「NHACRR_live」など。

   ReplicationAddress

   ネイティブ HA のクロスリージョンレプリケーション構成で、もう一方のグループのレプリケーショントラフィックポートに対応するホスト名とポートを指定します。 アドレスは、単一のサービスアドレスとポート、またはカンマ区切りのリストで指定できます。

   ライブグループでは、 NativeHALocalInstance のスタンザで以下の値を指定します

   GroupName
   ライブグループの名前を指定します。例えば、「NHACRR_live」など。

   GroupRole
   グループの現在の役割を指定します。 この場合、「ライブ」に設定します。

   NativeHARecoveryGroup のスタンザに以下の値を指定します

   GroupName
   リカバリサイト上のグループ名、例えば「NHACRR_recovery」など。

   ReplicationAddress

   ネイティブ HA のクロスリージョンレプリケーション構成で、もう一方のグループのレプリケーショントラフィックポートに対応するホスト名とポートを指定します。 アドレスは、単一のサービスアドレスとポート、またはカンマ区切りのリストで指定できます。
4. レプリケーションに使用する接続は、TLS を使用して保護する必要があります。 2 つのグループを作成した際に内部レプリケーションのセキュリティを指定した場合は、同じ設定を使用して外部レプリケーションを保護することができます。 あるいは、リカバリグループとライブグループの NativeHALocalInstance スタンザに以下の値を指定することもできます。

   GroupCipherSpec
   CipherSpec で指定された Cipher Spec を上書きします。

   GroupCertificateLabel
   CertificateLabel で指定された証明書ラベルを上書きします。

   これらの値を外部レプリケーションに使用し、内部レプリケーションにはプレーンテキストを使用する場合は、ライブグループとリカバリグループにこれらの 2 つの値を設定し、CipherSpec を NULL に設定します。

例

次の例は、ライブグループ上の 3 つのインスタンスのうちの 1 つに必要なネイティブ HA CRR の詳細を指定する qm.ini ファイルのセクションを示しています

```
NativeHALocalInstance:
  ...
  GroupName=NHACRR_live
  GroupRole=live

NativeHARecoveryGroup:
  GroupName=NHACRR_recovery
  ReplicationAddress=recovery-cluster.fyre.ibm.com(4445)
```

関連情報

qm.ini ファイルの NativeHALococalInstance スタンザ
NativeHARecoveryGroup qm.ini ファイルの 1 つの節
qm.ini ファイルの NativeHAInstance スタンザ

コンテナーでのキュー・マネージャーのデプロイおよび構成

IBM MQ キュー・マネージャーをデプロイおよび構成するには、さまざまなタスクを実行します。

このタスクについて

キュー・マネージャーのデプロイおよび構成を開始するには、以下のトピックを参照してください。

手順

- 97 ページの『IBM MQ Operator を使用したキュー・マネージャーのデプロイおよび構成』
- 188 ページの『サンプルの Helm チャートを使用したキューマネージャの展開と設定』

OpenShift IBM MQ Operator を使用したキュー・マネージャーのデプロイおよび構成

構成例: HA の構成、 OpenShift クラスターの外部からの接続、 CP4i ダッシュボードとの統合、Instana トレースとの統合、カスタム MQSC ファイルおよび INI ファイルを使用したイメージの作成、カスタム・アノテーションおよびラベルの追加。

このタスクについて

手順

- 100 ページの『キュー・マネージャーの構成例』.
- 111 ページの『IBM MQ Operator を使用したキュー・マネージャーの高可用性の構成』.
- 148 ページの『Red Hat OpenShift クラスターの外部からキュー・マネージャーに接続するためのルートの構成』.
- 152 ページの『IBM MQ と IBM Instana トレースの統合』.
- 159 ページの『Red Hat OpenShift CLI を使用した、カスタム MQSC および INI ファイルを使用したイメージの作成』.
- 163 ページの『キュー・マネージャー・リソースへのカスタム・アノテーションとカスタム・ラベルの追加』.
- 164 ページの『実行時 Webhook チェックの無効化』.
- 165 ページの『キュー・マネージャー仕様に対するデフォルト値更新の無効化』.

OpenShift CP4I IBM MQ Operator を使用した単純なキュー・マネージャーのデプロイ

この例では、一時 (非永続) ストレージを使用する「クイック・スタート」キュー・マネージャーをデプロイし、 IBM MQ セキュリティーをオフにします。 メッセージは、キュー・マネージャーの再始動後は保持されません。 構成を調整することで、キュー・マネージャーのさまざまな設定を変更できます。

このタスクについて

このタスクには、キュー・マネージャーを OpenShift にデプロイするための 3 つのオプションが用意されています。

1. OpenShift コンソールを使用してキュー・マネージャーをデプロイします。
2. OpenShift CLI を使用してキュー・マネージャーをデプロイします。
3. IBM Cloud Pak for Integration Platform UI を使用してキュー・マネージャーをデプロイします。

手順

- **オプション 1: OpenShift コンソールを使用してキュー・マネージャーをデプロイする。**

  a) キュー・マネージャーをデプロイします。

  a. Red Hat OpenShift Container Platform クラスター管理者資格情報を使用して、 OpenShift コンソールにログインします。

  b.「**プロジェクト**」 を、 IBM MQ Operator をインストールした名前空間に変更します。 「**プロジェクト**」 ドロップダウン・リストから名前空間を選択します。

  c. ナビゲーション・ペインで、 「**オペレーター**」 >「**インストール済みオペレーター**」をクリックします。

  d.「インストール済みオペレーター (Installed Operators)」パネルのリストで、 **IBM MQ** を見つけてクリックします。

  e.「**キュー・マネージャー (Queue Manager)**」タブをクリックします。

  f.「**QueueManager の作成 (Create QueueManager)**」ボタンをクリックします。 インスタンス作成パネルが表示され、リソースを構成するための 2 つの方法 (**フォーム・ビュー** と **YAML ビュー**) が示されます。 デフォルトでは、 「**フォーム・ビュー**」 が選択されています。

  b) キュー・マネージャーを構成します。

  ステップ 2 オプション 1: **フォーム・ビュー**で構成します。

  「**フォーム・ビュー**」 は、リソース構成を表示または変更するために使用できるフォームを開きます。

  a.「**ライセンス**」の横にある矢印をクリックして、ライセンス承諾セクションを展開します。

  b. ご使用条件に同意する場合は、 「**ライセンスの受諾**」 を **true** に設定します。

  c. 矢印をクリックしてドロップダウン・リストを開き、ライセンスを選択します。 IBM MQ は、複数の異なるライセンスで提供されています。 有効なライセンスについて詳しくは、249 ページの『mq.ibm.com/v1beta1: 現在のライセンスバージョン』を参照してください。 キュー・マネージャーをデプロイするには、ライセンスに同意する必要があります

  d.「**作成**」 をクリックします。 現在のプロジェクト (名前空間) 内のキュー・マネージャーのリストが表示されます。 新しい `QueueManager` が `Pending` の状態になっているはずです。

  ステップ 2 オプション 2: **YAML ビュー**で構成します。

  YAML ビュー は、`QueueManager` の YAML ファイルの例を含むエディターを開きます。 以下の手順に従って、ファイル内の値を更新します。

  a. `metadata.namespace` をプロジェクト (名前空間) 名に変更します。

b. spec.license.license の値を、要件に一致するライセンス・ストリングに変更します。 ライセンスの詳細については、 249 ページの『mq.ibm.com/v1beta1: 現在のライセンスバージョン』を参照してください。

c. ご使用条件に同意する場合は、 spec.license.accept を true に変更します。

d.「**作成**」 をクリックします。 現在のプロジェクト (名前空間) 内のキュー・マネージャーのリストが表示されます。 新しい QueueManager が Pending の状態になっているはずです。

c) キュー・マネージャーの作成を確認してください。

キュー・マネージャーが作成されたことを確認するには、以下の手順を実行します。

a. IBM MQ Operator を作成した名前空間内にいることを確認します。

b.「**ホーム**」 画面で、 **オペレーター** >「**インストール済みオペレーター**」をクリックし、キュー・マネージャーを作成した対象のインストール済み IBM MQ Operator を選択します。

c.「**キュー・マネージャー (Queue Manager)**」タブをクリックします。 QueueManager の状況が Running になったら、作成は完了です。

- **オプション 2: OpenShift CLI を使用してキュー・マネージャーをデプロイする。**

a) QueueManager YAML ファイルの作成

例えば、IBM Cloud Pak for Integration に基本的なキュー・マネージャーをインストールするには、以下を内容とするファイル「mq-quickstart.yaml」を作成します。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
spec:
  version: 9.4.3.0-r1
  license:
    accept: false
    license: L-CYPF-CRPF3H
    use: NonProduction
  web:
    enabled: true
  queueManager:
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
```

重要 : ご使用条件に同意する場合は、accept: false を accept: true に変更します。 ライセンスについて詳しくは、249 ページの『mq.ibm.com/v1beta1: 現在のライセンスバージョン』を参照してください。

この例には、キュー・マネージャーとともにデプロイされ、 IBM Cloud Pak for Integration 内でシングル・サインオンを使用して Web コンソールが使用可能になっている Web サーバーも含まれています。 シングル・サインオンを機能させるには、最初に他の IBM Cloud Pak for Integration コンポーネントをインストールする必要があります。 65 ページの『CP4I で使用するための IBM MQ Operator のインストール』を参照してください。

IBM Cloud Pak for Integration とは別に、基本的なキュー・マネージャーをインストールするには、以下を内容とするファイル「mq-quickstart.yaml」を作成します。

注 : この例では、 IBM MQ Advanced ライセンスを使用していますが、 IBM MQ ライセンスを使用することもできます。 詳しくは、161 ページの『を使用して、 IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart
spec:
  version: 9.4.3.0-r1
  license:
    accept: false
    license: L-NUUP-23NH8Y
  web:
```

```
    enabled: true
  queueManager:
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
```

重要: MQ のご使用条件に同意する場合は、 `accept: false` を `accept: true` に変更してください。 ライセンスについて詳しくは、249 ページの『mq.ibm.com/v1beta1: 現在のライセンスバージョン』を参照してください。

b) QueueManager オブジェクトを作成します。

```
oc apply -f mq-quickstart.yaml
```

c) キュー・マネージャーの作成を確認してください。
以下の手順を実行して、キュー・マネージャーが作成されたことを確認します。

a. デプロイメントを検証します。

```
oc describe queuemanager Queue_Manager_Resource_Name
```

b. 状況を確認します。

```
oc describe queuemanager quickstart
```

- **オプション 3: IBM Cloud Pak for Integration Platform UI を使用してキュー・マネージャーをデプロイする。**

  プラットフォーム UI を使用したインスタンスのデプロイの手順に従います。

関連タスク

148 ページの『Red Hat OpenShift クラスターの外部からキュー・マネージャーに接続するためのルートの構成』
Red Hat OpenShift クラスターの外部から IBM MQ キュー・マネージャーにアプリケーションを接続するには、 Red Hat OpenShift 経路が必要です。 IBM MQ キュー・マネージャーおよびクライアント・アプリケーションで TLS を有効にする必要があります。SNI は、TLS 1.2 以上のプロトコルが使用されている場合にのみ TLS プロトコルで使用できるためです。 Red Hat OpenShift Container Platform Router では、IBM MQ キュー・マネージャーへの要求のルーティングに SNI が使用されます。

218 ページの『IBM MQ Console クラスターにデプロイされた Red Hat OpenShift への接続』
Red Hat OpenShift Container Platform クラスターにデプロイされているキュー・マネージャーの IBM MQ Console に接続する方法について説明します。

100 ページの『キュー・マネージャーの構成例』
QueueManager カスタム・リソースの内容を調整することによってキュー・マネージャーを構成できます。

OpenShift CP4I キュー・マネージャーの構成例

QueueManager カスタム・リソースの内容を調整することによってキュー・マネージャーを構成できます。

このタスクについて

QueueManager YAML ファイルを使用してキュー・マネージャーを構成するのに役立つ例を以下に挙げます。

手順

- 101 ページの『例: MQSC ファイルと INI ファイルの提供』
- 106 ページの『例: 相互 TLS 認証を使用したキュー・マネージャーの構成』

OpenShift CP4I 例: *MQSC* ファイルと *INI* ファイルの提供

この例では、2 つの MQSC ファイルと 1 つの INI ファイルを組み込んだ Kubernetes ConfigMap を作成します。 その後、それらの MQSC ファイルと INI ファイルを処理するキュー・マネージャーをデプロイします。

このタスクについて

MQSC ファイルと INI ファイルは、キュー・マネージャーのデプロイ時に提供できます。 MQSC と INI のデータは、1 つ以上の Kubernetes ConfigMap とシークレットで定義する必要があります。 これらは、キューマネージャをデプロイするネームスペース（プロジェクト）で作成する必要があります。

注 : MQSC ファイルや INI ファイルに機密データを組み込む場合は、Kubernetes シークレットを使用してください。

例

2 つの MQSC ファイルと 1 つの INI ファイルを組み込んだ Kubernetes ConfigMap を作成する例を以下に示します。 その後、それらの MQSC ファイルと INI ファイルを処理するキュー・マネージャーをデプロイします。

例 ConfigMap: 以下の YAML をクラスタに適用します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mqsc-ini-example
data:
  example1.mqsc: |
    DEFINE QLOCAL('DEV.QUEUE.1') REPLACE
    DEFINE QLOCAL('DEV.QUEUE.2') REPLACE
  example2.mqsc: |
    DEFINE QLOCAL('DEV.DEAD.LETTER.QUEUE') REPLACE
  example.ini: |
    Channels:
       MQIBindType=FASTPATH
```

例 QueueManager: キューマネージャを以下の設定で展開するには、コマンドラインまたは Red Hat OpenShift Container Platform ウェブコンソールを使用します。

注 : この例では、IBM MQ Advanced ライセンスを使用していますが、IBM MQ ライセンスを使用することもできます。 詳しくは、161 ページの『を使用して、 IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: mqsc-ini-qm
spec:
  version: 9.4.3.0-r1
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  web:
    enabled: true
  queueManager:
    name: "MQSCINI"
    mqsc:
    - configMap:
        name: mqsc-ini-example
        items:
        - example1.mqsc
        - example2.mqsc
    ini:
    - configMap:
        name: mqsc-ini-example
        items:
        - example.ini
    storage:
      queueManager:
        type: ephemeral
```

重要 : IBM MQ Advanced のご使用条件に同意する場合は、`accept: false` を `accept: true` に変更します。 ライセンスについて詳しくは、249 ページの『mq.ibm.com/v1beta1: 現在のライセンスバージョン』を参照してください。

追加情報:

- キュー・マネージャーは、単一の Kubernetes ConfigMap またはシークレット (この例を参照) を使用するように構成することも、複数の ConfigMaps およびシークレットを使用するように構成することもできます。
- この例のように 1 つの Kubernetes ConfigMap またはシークレットから MQSC と INI のすべてのデータを使用することも、キュー・マネージャーごとに使用可能なファイルのサブセットだけを使用するように構成することも可能です。
- MQSC ファイルと INI ファイルは、それぞれのキーに基づいてアルファベット順に処理されます。 したがって、キューマネージャーの設定で表示される順序に関係なく、 `example1.mqsc` は常に `example2.mqsc` より先に処理されます。
- 複数の Kubernetes ConfigMap またはシークレットにまたがる複数の MQSC ファイルや INI ファイルが同じキーを持っている場合、その一連のファイルはそれぞれがキュー・マネージャー構成で定義されている順序に基づいて処理されます。
- キュー・マネージャー・ポッドの実行中は、 Kubernetes ConfigMap に対する変更は反映されません。これは、 IBM MQ Operator が変更を認識しないためです。 ConfigMap, に変更を加えた場合、例えば MQSC コマンドや INI ファイルに変更を加えた場合、それらの変更を取り込むためにキューマネージャを手動で再起動する必要があります。 単一インスタンス・キュー・マネージャーの場合は、ポッドを削除して、必要な再始動をトリガーします。 ネイティブ HA デプロイメントの場合は、まずスタンバイ・ポッドを削除して再始動します。 再度実行状態になったら、アクティブ・ポッドを削除して再始動します。 この再始動の順序により、キュー・マネージャーのダウン時間を最小限に抑えることができます。

OpenShift CP4I 例：キューマネージャーに追加のファイルを供給する

この例では、2 つのファイルを含む Kubernetes ConfigMap を作成します。 次にキューマネージャーが展開され、これらのファイルが指定された場所にある IBM MQ コンテナにマウントされます。

このタスクについて

キューマネージャーが展開される際に、追加のファイルを供給することができます。 これらのファイルは、1 つ以上の Kubernetes ConfigMaps および「秘密」 に定義する必要があります。 これらは、キューマネージャをデプロイするネームスペース（プロジェクト）で作成する必要があります。 Kubernetes 機密情報は、ファイルに機密データが含まれている場合に利用すべきです。

注 : この機能は、 IBM MQ Operator 3.5.0 以降でのみサポートされています。

例

次の例では、2 つのファイルを含む Kubernetes ConfigMap を作成します。 次にキューマネージャーが展開され、これらのファイルが指定された場所にある IBM MQ コンテナにマウントされます。

例 ConfigMap: 以下の YAML をクラスタに適用します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: files-example
data:
  file1.txt: |
    Test File 1
  file2.txt: |
    Test File 2
```

例 QueueManager: キューマネージャを以下の設定で展開するには、コマンドラインまたは Red Hat OpenShift Container Platform ウェブコンソールを使用します。

注 : この例では、IBM MQ Advanced ライセンスを使用していますが、IBM MQ ライセンスを使用することもできます。 詳しくは、161 ページの『を使用して、 IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: files-qm
spec:
  version: 9.4.3.0-r1
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    files:
    - configMap:
        name: files-example
        defaultMountPath: "/etc/mqm/myfiles"
        items:
        - item: file1.txt
        - item: file2.txt
          mountPath: "/etc/mqm/otherfiles"
    storage:
      queueManager:
        type: ephemeral
```

重要 : IBM MQ Advanced のご使用条件に同意する場合は、`accept: false` を `accept: true` に変更します。 ライセンスについて詳しくは、249 ページの『mq.ibm.com/v1beta1: 現在のライセンスバージョン』を参照してください。

ファイルの場所:

- ファイル `file1.txt` は場所 `/etc/mqm/myfiles/file1.txt` にマウントされており、これはこの ConfigMap に指定された **defaultMountPath** です。
- ファイル `file2.txt` は場所 `/etc/mqm/otherfiles/file2.txt` にマウントされており、これはこの特定のファイルに対する **mountPath** のオーバーライドです。

追加情報:

- **defaultMountPath** が未設定（ ConfigMap または Secret）で、 **mountPath** のオーバーライドが未指定の場合、ファイルはデフォルトで次の場所にマウントされます。 `/etc/mqm/customfiles`
- ConfigMap または Secret 内のいずれかのファイルの内容を変更した場合は、キュー・マネージャを手動で再起動して、変更を反映させる必要があります。 シングルインスタンスのキューマネージャーの場合、必要な再起動を実行するには、ポッドを削除します。 ネイティブ HA の展開では、まずスタンバイポッドを削除して再起動します。 再び実行状態になったら、アクティブなポッドを削除して再起動します。 この再起動の順序により、キューマネージャのダウンタイムを最小限に抑えることができます。

OpenShift CP4I 例：キューマネージャー用の追加環境変数の提供

この例では、2 つの追加環境変数とともにキューマネージャを展開します。 これらの環境変数は、IBM MQ コンテナ上に設定されます。

このタスクについて

キューマネージャーがデプロイされる際に、追加の環境変数を指定することができます。 これにより、IBM MQ Operator によってまだ公開されていない環境変数の設定が可能になります。

注 : この機能は、 IBM MQ Operator 3.5.0 以降でのみサポートされています。

例

次の例では、2 つの追加環境変数とともにキューマネージャを展開します。 これらの環境変数は、IBM MQ コンテナ上に設定されます。

例 QueueManager: キューマネージャを以下の設定で展開するには、コマンドラインまたは Red Hat OpenShift Container Platform ウェブコンソールを使用します。

注 : この例では、IBM MQ Advanced ライセンスを使用していますが、IBM MQ ライセンスを使用することもできます。 詳しくは、161 ページの『を使用して、 IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: env-qm
spec:
  version: 9.4.3.0-r1
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    env:
    - name: TEST_ENV_1
      value: "true"
    - name: TEST_ENV_2
      value: "false"
    storage:
      queueManager:
        type: ephemeral
```

重要 : IBM MQ Advanced のご使用条件に同意する場合は、`accept: false` を `accept: true` に変更します。 ライセンスについて詳しくは、249 ページの『mq.ibm.com/v1beta1: 現在のライセンスバージョン』を参照してください。

追加情報:

- IBM MQ Operator によって設定されるはずの環境変数を指定した場合、予期せぬ動作につながる可能性があります。 このシナリオでは、 QueueManager に警告ステータス条件が追加されます。

OpenShift CP4I *OpenSSL を使用した自己署名 PKI の作成*

IBM MQ では、認証に相互 TLS を使用できます。この場合、接続の両端で証明書が提供され、証明書の詳細を使用してキュー・マネージャーとの ID が確立されます。 このトピックでは、 OpenSSL コマンド行ツールを使用して Public Key Infrastructure (PKI) の例を作成し、他の例で使用できる 2 つの証明書を作成する方法について説明します。

始める前に

OpenSSL コマンド・ライン・ツールがインストールされていることを確認します。

IBM MQ client をインストールし、`samp/bin` と `bin` を パスに追加します。 **runmqicred** コマンドが必要だが、これは「IBM MQ client 一部として以下のようにインストールできる：

- Windows Linux Windows および Linux の場合: https://ibm.biz/mq94redistclients から、ご使用のオペレーティング・システム用の IBM MQ 再配布可能クライアントをインストールします。
- macOS Mac の場合: IBM MQ MacOS Toolkit をダウンロードしてセットアップします。 https://developer.ibm.com/tutorials/mq-macos-dev/

このタスクについて

重要 : ここで説明する例は、実稼働環境には適しておらず、すぐに使用できるようにするための例としてのみ意図されています。 証明書管理は、上級者向けの複雑なサブジェクトです。 実動環境では、ローテーション、取り消し、鍵の長さ、災害復旧などを考慮する必要があります。

これらのステップは、 OpenSSL 3.1.4 を使用してテストされています。

手順

1. 内部認証局に使用する秘密鍵を作成します

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out ca.key
```

内部認証局の秘密鍵は、 *ca.key* というファイルに作成されます。 このファイルは、安全に秘密にしておく必要があります。内部認証局の証明書に署名するために使用されます。

2. 内部認証局の自己署名証明書の発行

```
openssl req -x509 -new -nodes -key ca.key -sha512 -days 30 -subj "/CN=example-selfsigned-ca"
-out ca.crt
```

-days は、ルート CA 証明書が有効である日数を指定します。

ca.crt というファイルに証明書が作成されます。 この証明書には、内部認証局に関する公開情報が含まれており、自由に共有できます。

3. キュー・マネージャーの秘密鍵と証明書の作成

a) キュー・マネージャーの秘密鍵および証明書署名要求の作成

```
openssl req -new -nodes -out example-qm.csr -newkey rsa:4096 -keyout example-qm.key -subj
'/CN=example-qm'
```

example-qm.key という名前のファイルに秘密鍵が作成され、 *example-qm.csr* という名前のファイルに証明書署名要求が作成されます。

b) 内部認証局を使用してキュー・マネージャー鍵に署名します。

```
openssl x509 -req -in example-qm.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out
example-qm.crt -days 7 -sha512
```

-days は、証明書が有効になる日数を指定します。

example-qm.crt というファイルに署名証明書が作成されます。

c) キュー・マネージャーの鍵と証明書を使用して Kubernetes 秘密を作成します。

```
kubectl create secret generic example-qm-tls --type="kubernetes.io/tls" --from-
file=tls.key=example-qm.key --from-file=tls.crt=example-qm.crt --from-file=ca.crt
```

example-qm-tls という Kubernetes シークレットが作成されます。 この秘密には、キュー・マネージャーの秘密鍵、公開証明書、および CA 証明書が含まれています。

4. アプリケーションの秘密鍵と証明書の作成

a) アプリケーションの秘密鍵および証明書署名要求の作成

```
openssl req -new -nodes -out example-app1.csr -newkey rsa:4096 -keyout example-app1.key
-subj '/CN=example-app1'
```

example-app1.key というファイルに秘密鍵が作成され、*example-app1.csr* というファイルに証明書署名要求が作成されます。

b) 内部認証局を使用してキュー・マネージャー鍵に署名します。

```
openssl x509 -req -in example-app1.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out
example-app1.crt -days 7 -sha512
```

-days は、証明書が有効になる日数を指定します。

署名付き証明書が *example-app1.crt* というファイルに作成されます。

c) アプリケーションの鍵と証明書を使用して PKCS#12 鍵ストアを作成します。

IBM MQ は、個々の鍵ファイルではなく、鍵データベースを使用します。 コンテナー化されたキュー・マネージャーは、シークレットからキュー・マネージャーの鍵データベースを作成しますが、クライアント・アプリケーションの場合は、鍵データベースを手動で作成する必要があります。

```
openssl pkcs12 -export -in "example-app1.crt" -name "example-app1" -certfile "ca.crt"
-inkey "example-app1.key" -out "example-app1.p12" -passout pass:PASSWORD
```

ここで、 *PASSWORD* はユーザーが選択したパスワードです。

example-app1.p12 という名前のファイルに鍵ストアが作成されます。 アプリケーションの鍵と証明書は、CA 証明書だけでなく、「example-app1」の「label」または「friendly name」という名前で内部に保管されます。

d) arm64 Apple Mac を使用している場合は、アプリケーション証明書と CA 証明書を組み合わせた追加ファイルを構成する必要があります。
以下に例を示します。

```
cat example-app1.crt ca.crt > example-app1-chain.crt
```

関連タスク

106 ページの『例: 相互 TLS 認証を使用したキュー・マネージャーの構成』
この例では、 IBM MQ Operator を使用して、キュー・マネージャーを OpenShift Container Platform にデプロイします。 相互 TLS は、認証に使用され、TLS 証明書からキュー・マネージャー内の ID にマップされます。

112 ページの『例: IBM MQ Operator を使用したネイティブ HA の構成』
この例では、 IBM MQ Operator を使用して、ネイティブ高可用性フィーチャーを使用するキュー・マネージャーを OpenShift Container Platform にデプロイします。 相互 TLS は、認証に使用され、TLS 証明書からキュー・マネージャー内の ID にマップされます。

146 ページの『IBM MQ Operator を使用した複数インスタンス・キュー・マネージャーの構成』
この例では、 IBM MQ Operator を使用して、複数インスタンス・キュー・マネージャーを OpenShift Container Platform にデプロイします。 相互 TLS は、認証に使用され、TLS 証明書からキュー・マネージャー内の ID にマップされます。

OpenShift CP4I Linux 例: 相互 *TLS* 認証を使用したキュー・マネージャーの構成

この例では、 IBM MQ Operator を使用して、キュー・マネージャーを OpenShift Container Platform にデプロイします。 相互 TLS は、認証に使用され、TLS 証明書からキュー・マネージャー内の ID にマップされます。

始める前に

この例を完了するには、まず以下の前提条件を満たしておく必要があります。

- この例のための OpenShift Container Platform (OCP) プロジェクト / 名前空間を作成します。
- コマンド・ラインで OCP クラスターにログインし、上記の名前空間に切り替えます。
- 上記の名前空間に IBM MQ Operator がインストールされ、使用可能な状態になっていることを確認します。

このタスクについて

この例では、OpenShift Container Platform にデプロイするキュー・マネージャーを定義したカスタム・リソース YAML を提供しています。 また、TLS を有効にしてキュー・マネージャーをデプロイするために必要な追加のステップも詳しく説明しています。

手順

1. 104 ページの『OpenSSL を使用した自己署名 PKI の作成』の説明に従って、証明書のペアを作成します。
2. MQSC コマンドと INI ファイルを含む構成マップを作成します。

   MQSC コマンドを含む Kubernetes ConfigMap を作成して、新しいキューと SVRCONN チャネルを作成し、チャネルへのアクセスを許可するチャネル認証レコードを追加します。

   前に作成した名前空間 (始める前にを参照) にいることを確認してから、OCP Web コンソールで、またはコマンド・ラインを使用して、以下の YAML を入力します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-tls-configmap
data:
  example-tls.mqsc: |
    DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
```

```
SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(BROWSE,PUT,GET,INQ)
  example-tls.ini: |
    Service:
        Name=AuthorizationService
        EntryPoints=14
        SecurityPolicy=UserExternal
```

MQSC は、*MTLS.SVRCONN* というチャネルと、*EXAMPLE.QUEUE*。 チャネルは、「共通名」が *example-app1* の証明書を提示するクライアントにのみアクセスを許可するように構成されています。 これは、ステップ 106 ページの『1』で作成した証明書のいずれかで使用される共通名です。 この共通名を持つこのチャネル上の接続は、 *app1* というユーザー ID にマップされます。このユーザー ID は、キュー・マネージャーに接続し、サンプル・キューにアクセスすることを許可されています。 INI ファイルはセキュリティー・ポリシーを使用可能にします。これは、*app1* ユーザー ID が外部ユーザー・レジストリーに存在する必要がないことを意味します。この ID は、この構成に名前としてのみ存在します。

3. キュー・マネージャーのデプロイ

   以下のカスタム・リソース YAML を使用して、新しいキュー・マネージャーを作成します。 このタスクを開始する前に作成した名前空間にいることを確認してから、OCP Web コンソールで、またはコマンド・ラインを使用して、以下の YAML を入力します。 正しいライセンスが指定されていることを確認し、`false` を `true` に変更してライセンスを受け入れます。

   注 : この例では、 IBM MQ Advanced ライセンスを使用していますが、 IBM MQ ライセンスを使用することもできます。 詳しくは、161 ページの『を使用して、IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    mqsc:
    - configMap:
        name: example-tls-configmap
        items:
        - example-tls.mqsc
    ini:
    - configMap:
        name: example-tls-configmap
        items:
        - example-tls.ini
    storage:
      queueManager:
        type: ephemeral
  version: 9.4.3.0-r1
  pki:
    keys:
      - name: default
        secret:
          secretName: example-qm-tls
          items:
            - tls.key
            - tls.crt
            - ca.crt
```

シークレット *example-qm-tls* がステップ 106 ページの『1』で作成され、 ConfigMap *example-tls-configmap* がステップ 106 ページの『2』 で作成されたことに注意してください。

4. キュー・マネージャーが稼働していることの確認

キュー・マネージャーがデプロイされます。 続行する前に、Running 状態であることを確認してください。 以下に例を示します。

```
oc get qmgr exampleqm
```

5. キュー・マネージャーへの接続のテスト
   キュー・マネージャーが相互 TLS 通信用に構成されていることを確認するには、108 ページの『ラップトップからキュー・マネージャーへの相互 TLS 接続のテスト』の手順に従います。

タスクの結果

これで、TLS が有効になっているキュー・マネージャーが正常にデプロイされ、TLS 証明書で提供された詳細を使用してキュー・マネージャーで認証を行い、ID を指定することができました。

OpenShift CP4I Linux ラップトップからキュー・マネージャーへの相互 *TLS* 接続のテスト

IBM MQ Operator を使用してキュー・マネージャーを作成した後、そのキュー・マネージャーに接続し、メッセージの書き込みと読み取りを行うことによって、そのキュー・マネージャーが機能しているかどうかをテストできます。 このタスクでは、ラップトップなどの Kubernetes クラスター外部のマシン上で IBM MQ サンプル・プログラムを実行することにより、それらのプログラムを使用して接続する方法について説明します。

始める前に

この例を完了するには、まず以下の前提条件を満たしておく必要があります。

- IBM MQ client のインストール **amqsputc** コマンドと **amqsgetc** コマンドが必要です。これらのコマンドは、以下のように IBM MQ client の一部としてインストールできます。
  - Windows Linux Windows および Linux の場合: https://ibm.biz/mq94redistclients から、ご使用のオペレーティング・システム用の IBM MQ 再配布可能クライアントをインストールします。
  - macOS Mac の場合: IBM MQ MacOS Toolkit をダウンロードしてセットアップします。 https://developer.ibm.com/tutorials/mq-macos-dev/
- 必要な鍵ファイルと証明書ファイルがマシン上のディレクトリーにダウンロードされていること、および鍵ストアのパスワードがわかっていることを確認してください。 例えば、以下のファイルが 104 ページの『OpenSSL を使用した自己署名 PKI の作成』に作成されます。
  - `example-app1.p12`
  - `example-app1-chain.crt` (arm64 Apple Mac を使用している場合のみ)
- TLS で構成されたキュー・マネージャーを OCP クラスターにデプロイします。例えば、 106 ページの『例: 相互 TLS 認証を使用したキュー・マネージャーの構成』 の手順に従います。

このタスクについて

この例では、ラップトップなどの Kubernetes クラスター外部のマシン上で実行されている IBM MQ サンプル・プログラムを使用して、TLS で構成された QueueManager に接続し、メッセージの書き込みと取得を行います。

手順

1. キュー・マネージャーが稼働していることの確認
   キュー・マネージャーがデプロイされます。 続行する前に、Running 状態であることを確認してください。 以下に例を示します。

```
oc get qmgr exampleqm
```

2. キュー・マネージャーのホスト名の検索

自動的に作成される経路を使用して、OCP クラスターの外部からキュー・マネージャーの完全修飾ホスト名を検索するには、次のコマンドを使用します。 `exampleqm-ibm-mq-qm`

```
oc get route exampleqm-ibm-mq-qm --template="{{.spec.host}}"
```

3. IBM MQ クライアント・チャネル定義テーブル (CCDT) の作成

   以下の内容を含む `ccdt.json` というファイルを作成します。

```
{
    "channel":
    [
        {
            "name": "MTLS.SVRCONN",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "hostname from previous step",
                        "port": 443
                    }
                ],
                "queueManager": "EXAMPLEQM"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ANY_TLS13",
              "certificateLabel": "example-app1"
            },
            "type": "clientConnection"
        }
   ]
}
```

   この接続では、Red Hat OpenShift Container Platform ルーターが listen しているポートであるため、ポート 443 が使用されます。 トラフィックは、ポート 1414 でキュー・マネージャーに転送されます。

   別のチャネル名を使用した場合は、それも調整する必要があります。 相互 TLS の例では、*MTLS.SVRCONN* という名前のチャネルを使用します。

   詳しくは、 JSON 形式の CCDT の構成 を参照してください。

4. 接続の詳細を構成するためのクライアント INI ファイルの作成

   現行ディレクトリーに `mqclient.ini` という名前のファイルを作成します。 このファイルは、**amqsputc** および **amqsgetc** によって読み取られます。

```
Channels:
  ChannelDefinitionDirectory=.
  ChannelDefinitionFile=ccdt.json
SSL:
  OutboundSNI=HOSTNAME
  SSLKeyRepository=example-app1.p12
  SSLKeyRepositoryPassword=password you used when creating the p12 file
```

   必ず、 *SSLKeyRepository*「パスワード」 を、 PKCS#12 ファイルの作成時に選択したパスワードに更新してください。 暗号化されたパスワードを使用するなど、鍵ストアのパスワードを設定する方法は他にもあります。 詳しくは、 AIX, Linux, and Windows での IBM MQ MQI client の鍵リポジトリー・パスワードの提供 を参照してください。

   Red Hat OpenShift Container Platform Router では、IBM MQ キュー・マネージャーへの要求のルーティングに SNI が使用されます。 *OutboundSNI= HOSTNAME* 属性は、ルーターが IBM MQ Operator によって構成されたデフォルト経路を処理するために必要な情報が IBM MQ クライアントに確実に含まれるようにします。 詳しくは、148 ページの『Red Hat OpenShift クラスターの外部からキュー・マネージャーに接続するためのルートの構成』を参照してください。

5. arm64 Apple Mac を使用している場合は、追加の環境変数を構成する必要があります。

```
export MQSSLTRUSTSTORE=example-app1-chain.crt
```

   このファイルには、アプリケーションおよび CA 証明書を含む完全な証明書チェーンが含まれています。

6. キューへのメッセージの書き込み
   以下のコマンドを実行します。

   ```
   /opt/mqm/samp/bin/amqsputc EXAMPLE.QUEUE EXAMPLEQM
   ```

   キュー・マネージャーへの接続が成功すると、以下の応答が出力されます。

   ```
   target queue is EXAMPLE.QUEUE
   ```

   任意のテキストを入力してから **Enter** を押す操作を何回か繰り返すことで、キューに複数のメッセージを書き込みます。

   書き込みを終了するには、**Enter** を 2 回押します。
7. キューからのメッセージの取得
   以下のコマンドを実行します。

   ```
   /opt/mqm/samp/bin/amqsgetc EXAMPLE.QUEUE EXAMPLEQM
   ```

   前のステップで追加したメッセージがコンシュームされ、出力されます。 数秒後にコマンドが終了します。

タスクの結果

これで、TLS が有効になっているキュー・マネージャーの接続のテストが正常に完了し、クライアントからキュー・マネージャーにメッセージを安全に書き込んだり、取得したりできることが示されました。

OpenShift CP4I 例：ライセンス・サービスの注釈のカスタマイズ

IBM MQ Operator は、デプロイ済みリソースに IBM License Service アノテーションを自動的に追加します。 これらは IBM License Service によってモニターされ、必要な資格に対応するレポートが生成されます。

このタスクについて

IBM MQ Operator によって追加される注釈は、標準シチュエーションで予期される注釈であり、キュー・マネージャーのデプロイメント中に選択されたライセンス値に基づいています。

例

License が `L-RJON-BZFQU2 (IBM Cloud Pak for Integration 2021.2.1)` に設定されていて、**Use** が `NonProduction` に設定されている場合は、次のアノテーションが適用されます:

- cloudpakId: c8b82d189e7545f0892db9ef2731b90d
- cloudpakName: IBM Cloud Pak for Integration
- productChargedContainers: qmgr
- productCloudpakRatio: '4:1'
- productID: 21dfe9a0f00f444f888756d835334909
- 実動用の製品名: IBM MQ Advanced 非実動用
- 製品メトリック：VIRTUAL_PROCESSOR_CORE
- productVersion: 9.2.3.0

IBM Cloud Pak for Integration 内で、 IBM App Connect Enterprise のデプロイメントには、IBM MQ に対する制限付き使用権が含まれます。 このような状況では、これらのアノテーションをオーバーライドして、IBM License Service が正しい使用法をキャプチャーすることを確認する必要があります。 そのためには、163 ページの『キュー・マネージャー・リソースへのカスタム・アノテーションとカスタム・ラベルの追加』で説明されている方法を使用してください

例えば、IBM MQ が IBM App Connect Enterprise ライセンスの下にデプロイされている場合は、以下のコード・フラグメントに示されている方法を使用してください。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: mq4ace
  namespace: cp4i
spec:
  annotations:
    productMetric: FREE
```

ライセンスアノテーションの変更が必要となる理由は、他に 2 つあります。

1. IBM MQ Advanced は、別の IBM 製品の使用権に含まれています。
   - この状態では、IBM App Connect Enterprise について前に説明したアプローチを使用してください。
2. IBM MQ は IBM Cloud Pak for Integration ライセンスの下にデプロイされます。
   - IBM Cloud Pak for Integration ライセンスがある場合は、IBM MQ または IBM MQ Advanced の比率の下にキュー・マネージャーをデプロイすることを決定できます。 IBM MQ の比率の下にデプロイする場合は、ネイティブ HA や Advanced Message Security などの高度な機能を使用しないようにする必要があります。
   - このような場合は、以下のアノテーションを使用して制作してください。

     ```
     apiVersion: mq.ibm.com/v1beta1
     kind: QueueManager
     metadata:
       name: mq4ace
       namespace: cp4i
     spec:
       annotations:
         productID: c661609261d5471fb4ff8970a36bccea
         productCloudpakRatio: '4:1'
         productName: IBM MQ for Production
         productMetric: VIRTUAL_PROCESSOR_CORE
     ```

   - 非実動使用には以下の注釈を使用します。

     ```
     apiVersion: mq.ibm.com/v1beta1
     kind: QueueManager
     metadata:
       name: mq4ace
       namespace: cp4i
     spec:
       annotations:
         productID: 151bec68564a4a47a14e6fa99266deff
         productCloudpakRatio: '8:1'
         productName: IBM MQ for Non-Production
         productMetric: VIRTUAL_PROCESSOR_CORE
     ```

OpenShift MQ Adv. IBM MQ Operator を使用したキュー・マネージャーの高可用性の構成

キューマネージャの高可用性を設定するには、ネイティブ HA ソリューションとマルチインスタンス・キューマネージャ・ソリューションのいずれかを使用します。

このタスクについて

選択した高可用性ソリューションの手順に従ってください。

手順

- 42 ページの『ネイティブ HA』.
- 112 ページの『例: IBM MQ Operator を使用したネイティブ HA の構成』.
- 146 ページの『IBM MQ Operator を使用した複数インスタンス・キュー・マネージャーの構成』.

IBM MQ Operator を使用したネイティブ HA の構成

ネイティブ HA は QueueManager API を使用して構成され、拡張オプションは INI ファイルを使用して利用できます。

ネイティブ HA は、.spec.queueManager.availability (QueueManager API) を使用して構成されます。以下に例を示します。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
name: nativeha-example
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    availability:
      type: NativeHA
  version: 9.4.3.0-r1
```

注 : この例では、 IBM MQ Advanced ライセンスを使用しています。 9.4.2 以降、 IBM MQ ライセンスを取得したキュー・マネージャーも、キュー・マネージャーにライセンスを追加することで、ネイティブ HA とクロスリージョンレプリケーション機能を使用するように設定することができます。 詳しくは、116 ページの『IBM MQ 、キューマネージャのネイティブ HA とクロスリージョンレプリケーションを設定するには、次のようにします。 IBM MQ Operator』を参照してください。

.spec.queueManager.availability.type フィールドは NativeHA に設定する必要があります。

.spec.queueManager.availability では、複製時にキュー・マネージャー・インスタンス間で使用する TLS シークレットと TLS 暗号を構成することもできます。 この構成を行うことを強くお勧めします。 112 ページの『例: IBM MQ Operator を使用したネイティブ HA の構成』には、ステップバイステップ形式のガイドが用意されています。

関連タスク

112 ページの『例: IBM MQ Operator を使用したネイティブ HA の構成』
この例では、 IBM MQ Operator を使用して、ネイティブ高可用性フィーチャーを使用するキュー・マネージャーを OpenShift Container Platform にデプロイします。 相互 TLS は、認証に使用され、TLS 証明書からキュー・マネージャー内の ID にマップされます。

例: IBM MQ Operator を使用したネイティブ HA の構成

この例では、 IBM MQ Operator を使用して、ネイティブ高可用性フィーチャーを使用するキュー・マネージャーを OpenShift Container Platform にデプロイします。 相互 TLS は、認証に使用され、TLS 証明書からキュー・マネージャー内の ID にマップされます。

始める前に

この例を完了するには、まず以下の前提条件を満たしておく必要があります。

- この例のための OpenShift Container Platform (OCP) プロジェクト / 名前空間を作成します。
- コマンド・ラインで OCP クラスターにログインし、上記の名前空間に切り替えます。
- 上記の名前空間に IBM MQ Operator がインストールされ、使用可能な状態になっていることを確認します。

このタスクについて

この例では、OpenShift Container Platform にデプロイするキュー・マネージャーを定義したカスタム・リソース YAML を提供しています。 また、TLS を有効にしてキュー・マネージャーをデプロイするために必要な追加のステップも詳しく説明しています。

手順

1. 104 ページの『OpenSSL を使用した自己署名 PKI の作成』の説明に従って、証明書のペアを作成します。
2. MQSC コマンドと INI ファイルを含む構成マップを作成します。

   MQSC コマンドを含む Kubernetes ConfigMap を作成して、新しいキューと SVRCONN チャネルを作成し、チャネルへのアクセスを許可するチャネル認証レコードを追加します。

   前に作成した名前空間 (始める前にを参照) にいることを確認してから、OCP Web コンソールで、またはコマンド・ラインを使用して、以下の YAML を入力します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-nativeha-configmap
data:
  example-tls.mqsc: |
    DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(BROWSE,PUT,GET,INQ)
  example-tls.ini: |
    Service:
        Name=AuthorizationService
        EntryPoints=14
        SecurityPolicy=UserExternal
```

   MQSC は、*MTLS.SVRCONN* というチャネルと、*EXAMPLE.QUEUE*。 チャネルは、「共通名」が *example-app1* の証明書を提示するクライアントにのみアクセスを許可するように構成されています。 これは、ステップ 113 ページの『1』で作成した証明書のいずれかで使用される共通名です。 この共通名を持つこのチャネル上の接続は、 *app1* というユーザー ID にマップされます。このユーザー ID は、キュー・マネージャーに接続し、サンプル・キューにアクセスすることを許可されています。 INI ファイルはセキュリティー・ポリシーを使用可能にします。これは、 *app1* ユーザー ID が外部ユーザー・レジストリーに存在する必要がないことを意味します。この ID は、この構成に名前としてのみ存在します。
3. キュー・マネージャーのデプロイ

   以下のカスタム・リソース YAML を使用して、新しいキュー・マネージャーを作成します。 このタスクを開始する前に作成した名前空間にいることを確認してから、OCP Web コンソールで、またはコマンド・ラインを使用して、以下の YAML を入力します。 正しいライセンスが指定されていることを確認し、`false` を `true` に変更してライセンスを受け入れます。

   注 :

   この例では、 IBM MQ Advanced ライセンスを使用しています。 IBM MQ 9.4.2 より、 IBM MQ のライセンスを使用するキューマネージャーでは、ネイティブ HA およびクロスリージョンレプリケーション機能が利用可能です。 詳しくは、 116 ページの『IBM MQ 、キューマネージャのネイティブ HA とクロスリージョンレプリケーションを設定するには、次のようにします。 IBM MQ Operator』を参照してください。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    availability:
      type: NativeHA
      tls:
        secretName: example-qm-tls
    mqsc:
```

```
    - configMap:
        name: example-nativeha-configmap
        items:
        - example-tls.mqsc
    ini:
    - configMap:
        name: example-nativeha-configmap
        items:
        - example-tls.ini
    storage:
      queueManager:
        type: persistent-claim
  version: 9.4.2.0-r2
  pki:
    keys:
      - name: default
        secret:
          secretName: example-qm-tls
          items:
            - tls.key
            - tls.crt
            - ca.crt
```

シークレット *example-qm-tls* がステップ 113 ページの『1』で作成され、ConfigMap *example-nativeha-configmap* がステップ 113 ページの『2』 で作成されたことに注意してください。

可用性タイプは *NativeHA* に設定され、永続ストレージが選択されている。 Kubernetes クラスターで構成されているデフォルトのストレージ・クラスが使用されます。 ストレージ・クラスがデフォルトとして構成されていない場合、または別のストレージ・クラスを使用したい場合は、`defaultClass: storage_class_name` を `spec.queueManager.storage` の下に追加します。

ネイティブ HA キュー・マネージャー内の 3 つのポッドは、ネットワークを介してデータを複製します。このリンクはデフォルトでは暗号化されませんが、この例ではトラフィックの暗号化にキュー・マネージャーの証明書を使用します。 セキュリティーを強化するために、別の証明書を指定することができます。 ネイティブ HA TLS シークレットは、特定の構造を持つ Kubernetes TLS シークレットでなければなりません (例えば、秘密鍵を *tls.key* という名前にする必要があります)。

4. キュー・マネージャーが稼働していることの確認

   キュー・マネージャーがデプロイされます。 続行する前に、`Running` 状態であることを確認してください。 以下に例を示します。

   ```
   oc get qmgr exampleqm
   ```

5. キュー・マネージャーへの接続のテスト

   キュー・マネージャーが構成済みで使用可能であることを確認するには、108 ページの『ラップトップからキュー・マネージャーへの相互 TLS 接続のテスト』の手順に従います。

6. アクティブ・ポッドの障害の強制試行

   キュー・マネージャーの自動復旧を検証するには、ポッドの障害をシミュレートします。

   a) アクティブ・ポッドとスタンバイ・ポッドの表示

   以下のコマンドを実行します。

   ```
   oc get pods --selector app.kubernetes.io/instance=exampleqm
   ```

   READY フィールドでは、アクティブ・ポッドは値 `1/1` を返し、レプリカ・ポッドは値 `0/1` を返すことに注意してください。

   b) アクティブ・ポッドの削除

   アクティブ・ポッドの絶対パス名を指定して次コマンドを実行します。

   ```
   oc delete pod exampleqm-ibm-mq-value
   ```

   c) ポッドの状況の再表示

   以下のコマンドを実行します。

   ```
   oc get pods --selector app.kubernetes.io/instance=exampleqm
   ```

d) キュー・マネージャーの状況の表示

<active-pod>のフルネームを指定して、以下のコマンドを実行する。アクティブポッドはレディ状態になっているはずで、ステップ c のコマンドで示されたレディ 1/1 のポッドである：

```
oc exec -t <active-pod> -- dspmq -o nativeha -x -m EXAMPLEQM
```

アクティブ・インスタンスが変更されたことを示す状況が表示されるはずです。以下に例を示します。

```
QMNAME(EXAMPLEQM) ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
INSTANCE(inst1) ROLE(Active) REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

e) キュー・マネージャーへの接続を再度テストします。

キュー・マネージャーが回復したことを確認するには、108 ページの『ラップトップからキュー・マネージャーへの相互 TLS 接続のテスト』の手順に従います。

タスクの結果

これで、ネイティブの高可用性と相互 TLS 認証を使用してキュー・マネージャーが正常にデプロイされ、アクティブなポッドで障害が発生したときに自動的に復旧することが確認されました。

OpenShift V 9.4.2 MQ Adv. ネイティブ *HA CRR* の設定に *IBM MQ Operator*

ネイティブ HA クロスリージョンレプリケーション（CRR）は、QueueManager API を使用して設定され、INI ファイルを使用して高度なオプションが利用可能です。

注 : この例では、 IBM MQ Advanced ライセンスを使用しています。 IBM MQ 9.4.2, IBM MQ から、キューマネージャにライセンスを追加することで、Native HA CRR の機能を使用するように設定することもできます。 詳しくは、116 ページの『IBM MQ 、キューマネージャのネイティブ HA とクロスリージョンレプリケーションを設定するには、次のようにします。 IBM MQ Operator』を参照してください。

ネイティブ HA CRR は以下を使用して設定されます。 .spec.queueManager.availability の QueueManager API。 .spec.queueManager.availability.type フィールドを NativeHA に設定する必要があります。 .spec.queueManager.availability.nativeHAGroups.local.role フィールドは、ローカルグループの役割を指定するために、 Live または Recovery に設定されています。

リモートグループの詳細は、.spec.queueManager.availability.nativeHAGroups.remotes セクションの構成で指定します。 リモートグループで使用される OpenShift ルートのホスト名とポート番号を指定します。 ライブグループには、最低でも remotes セクションを含める必要があり、それによりリカバリグループへのレプリケーションが可能になります。 リカバリグループには、 remotes セクションを含める必要があります。そうすれば、いずれ本番稼働した際に、他のサイトにレプリケートすることができます。 （236 ページの『ネイティブ HA CRR の切り替えとフェイルオーバー』 を参照）

.spec.queueManager.availability では、複製時にキュー・マネージャー・インスタンス間で使用する TLS シークレットと TLS 暗号を構成することもできます。 この構成を行うことを強くお勧めします。112 ページの『例: IBM MQ Operator を使用したネイティブ HA の構成』には、ステップバイステップ形式のガイドが用意されています。

レプリケーションを行う際には、ライブグループとリカバリグループ間で使用する TLS シークレットと暗号を設定する必要があります。 インスタンス間のレプリケーションで指定したものと同じシークレットを使用することもできますし、この目的のために別のシークレットを指定することもできます。

関連タスク

112 ページの『例: IBM MQ Operator を使用したネイティブ HA の構成』
この例では、 IBM MQ Operator を使用して、ネイティブ高可用性フィーチャーを使用するキュー・マネージャーを OpenShift Container Platform にデプロイします。 相互 TLS は、認証に使用され、TLS 証明書からキュー・マネージャー内の ID にマップされます。

OpenShift CP4I V 9.4.2 **IBM MQ 、キューマネージャのネイティブ *HA* とクロスリージョンレプリケーションを設定するには、次のようにします。 *IBM MQ Operator***

IBM MQ 9.4.2, IBM MQ から、キューマネージャにライセンスを追加することで、ネイティブ HA クロスリージョンレプリケーション（CRR）機能を使用するようにキューマネージャを構成することができます。

始める前に

注 : これらの説明は、 IBM MQ Operator を使用する場合にのみ適用されます。 キュー・マネージャーを Kubernetes に配備している場合は、 186 ページの『IBM MQ のキューマネージャでネイティブ HA とクロスリージョンレプリケーションを設定する。 Kubernetes』 を参照してください。

IBM MQ ライセンスのキューマネージャでネイティブ HA CRR 機能を使用するには、 IBM License Service が必要です。 IBM MQ Operator インストール・プロセスの「 License Service デプロイ 」ステップを参照してください。

重要 : このタスクはキューマネージャの YAML を更新することを含み、ポッドの再起動をトリガします。

手順

1. `IBMLicensingDefinition` CR は、本番使用か非本番使用かによって、どちらか一方、または両方を追加する。 これらの CR は，キューマネージャに Native HA CRR の機能に関する追加のライセンスアノテーションを追加します。

   この手順を、Native HA CRR 機能の追加ライセンスを使用する各ネームスペースで繰り返します。

   - **オプション 1:** OpenShift コンソールを使って CR を追加する：
     - Red Hat OpenShift Container Platform クラスタ管理者の認証情報を使用して OpenShift コンソールにログインします。
     - **Project**] ドロップダウン・リストからネームスペースを選択して、キュー・マネージャが配備されているネームスペースに [**Project**] を変更します。
     - 画面右上の**クイック作成** （プラス記号）アイコンをクリックして、 **YAML のインポートページを**開きます。
     - 関連する `IBMLicensingDefinition` CR YAML を追加する（このステップの最後にある例を参照）。
     - 「**作成**」 をクリックします。
     - 必要であれば、この手順を繰り返して、他の `IBMLicensingDefinition` CR を追加する。
   - **オプション 2：** コマンドラインを使って CR を追加する：
     - `oc login` コマンドを使用して、 Red Hat OpenShift コマンドラインインターフェイス（CLI）にログインする。
     - `project` (名前空間) を、キュー・マネージャが配置されているものに設定します：

       ```
       oc project <QUEUE-MANAGER-NAMESPACE>
       ```

     - `IBMLicensingDefinition` YAML ファイルを作成する（このステップの最後にある例を参照）。
     - `IBMLicensingDefinition` YAML ：

       ```
       oc apply -f <IBMLicensingDefinition-UNIQUE-NAME>.yaml
       ```

     - 必要であれば、この手順を繰り返して、他の `IBMLicensingDefinition` CR を追加する。

   以下の例の `IBMLicensingDefinition` CR YAML ファイルでは、`<QUEUE-MANAGER-NAMESPACE>` の値をキュー・マネージャがデプロイされているプロジェクトに変更してください。

   - 生産用：

     ```
     apiVersion: operator.ibm.com/v1
     kind: IBMLicensingDefinition
     metadata:
     ```

```
  name: ibmlicensingdefinition-mq-nha-prod
  namespace: <QUEUE-MANAGER-NAMESPACE>
spec:
  action: cloneModify
  condition:
    metadata:
      annotations:
        additionalLicense: L-EZBK-Y86SSH
        additionalLicenseUse: Production
  scope: cluster
  set:
    productID: 46a57a4a66ee4b904e9b5544b53a2ba2
    productName: IBM MQ Native HA and Cross-Region Replication Add-on
```

- 非生産用：

```
apiVersion: operator.ibm.com/v1
kind: IBMLicensingDefinition
metadata:
  name: ibmlicensingdefinition-mq-nha-non-prod
  namespace: <QUEUE-MANAGER-NAMESPACE>
spec:
  action: cloneModify
  condition:
    metadata:
      annotations:
        additionalLicense: L-EZBK-Y86SSH
        additionalLicenseUse: NonProduction
  scope: cluster
  set:
    productID: 525b1331a74d4a209d9f6e9c908e374e
    productName: IBM MQ Native HA and Cross-Region Replication Add-on for Non-Production
Env
```

2. 関連する注釈をキュー・マネージャの spec.annotations に追加する。

   重要 : キュー・マネージャの YAML を更新すると、ポッドの再起動がトリガされます。

   162 ページの『を使用してキュー・マネージャーにアノテーションを追加する。 IBM MQ Operator』 の指示に従って、必要に応じて以下の注釈のいずれかを追加する：

   - IBM MQ 、 IBM MQ Native HA および Cross-Region Replication アドオンの権限について：

     ```
     spec:
       annotations:
         productID: c661609261d5471fb4ff8970a36bccea
         productName: IBM MQ
         additionalLicense: L-EZBK-Y86SSH
         additionalLicenseUse: Production
     ```

   - IBM MQ for Non-Production Environment 、IBM MQ Native HA および Cross-Region Replication Add-on for Non-Production Entititlements の場合

     ```
     spec:
       annotations:
         productID: 151bec68564a4a47a14e6fa99266deff
         productName: IBM MQ for Non-Production Environment
         additionalLicense: L-EZBK-Y86SSH
         additionalLicenseUse: NonProduction
     ```

   必要に応じて、他のキュー・マネージャについてもこの手順を繰り返す。

OpenShift MQ Adv. Kubernetes 例 *IBM MQ Operator* を使用したシンプルなネイティブ *HA CRR* 構成のデプロイ

この例では、シンプルなネイティブ HA クロスリージョンレプリケーション(CRR)構成を使用したキューマネージャを、 IBM MQ Operator を使用して OpenShift® Container Platform。

始める前に

この例を完成させるには、まず以下のステップを完了させる必要がある：

- コマンド・ラインで、使用している OpenShift Container Platform （OCP）クラスタにログインする。

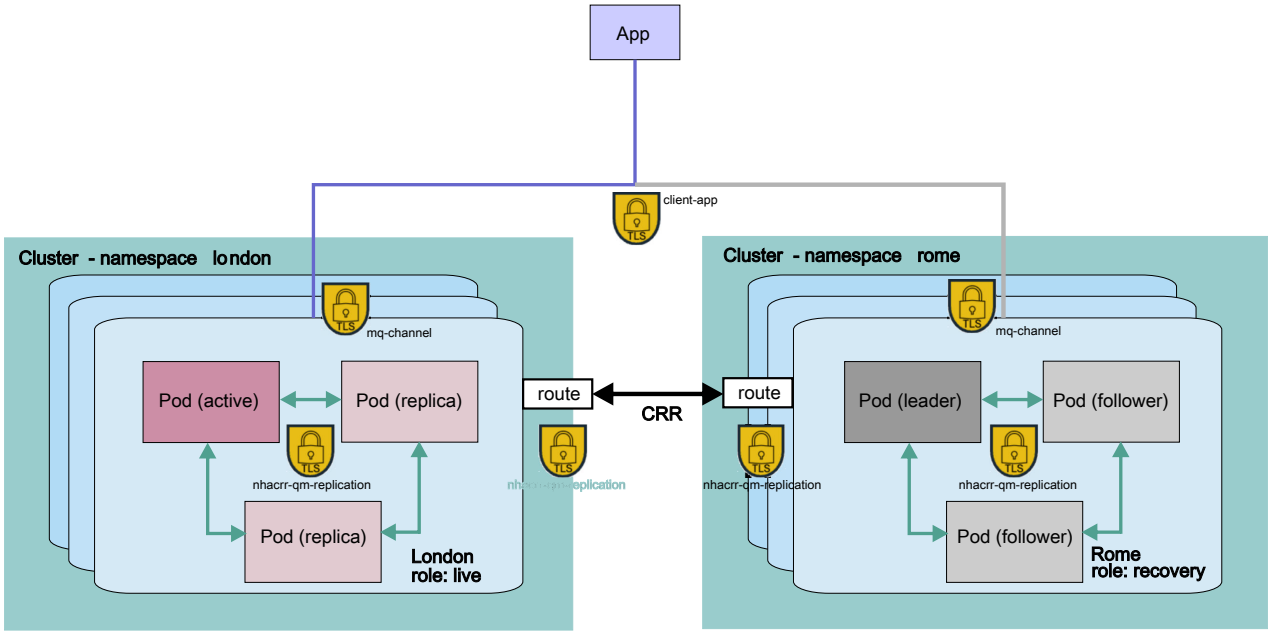
- この例では、2 つの OCP プロジェクト/ネームスペースを作成します。 この例では、ロンドンとローマのサイトを参照する。 これらは、単一のクラスタ内のロンドンとローマの 2 つのネームスペース、または各サイトの各クラスタ内のネームスペースのいずれかを指します。
- IBM MQ Operator 3.5.0 またはそれ以上がクラスタにインストールされ、ネームスペースで利用可能であることを確認します。
- OpenSSL コマンドラインツールがインストールされていることを確認する。
- 86 ページの『プルシークレットを作成して「Kubernetes」を使用する準備』 タスクを完了する。
- 98 ページの『IBM MQ Operator を使用した単純なキュー・マネージャーのデプロイ』 を読み、 IBM MQ 演算子の使い方に慣れておくこと。

このタスクについて

ネイティブ HA CRR 構成では、2 つのネイティブ HA グループを定義し、これらを合わせて高可用 キュー・マネージャを構成する。 これらのグループは通常、異なる場所にある 2 つの OCP クラスタに配備されますが、この例では、必要に応じて両方のグループを同じクラスタに配備することができます。 ネイティブ HA CRR の概要については、「 ネイティブ HA クロスリージョンレプリケーション 」を参照してください。

簡単にするために、この例ではキュー・マネージャのレプリケーション・トラフィックをすべて保護するために単一の証明書を使用します。

Native HA CRR における証明書の使用方法、およびグループとインスタンスのトラフィックを保護するための証明書の使い分けについては、 130 ページの『例：ネイティブ HA CRR で複数の TLS 証明書を使用する場合』 を参照してください。



重要 : これらの例は、本番環境には適していません。 証明書の管理は、上級ユーザーにとっては複雑なテーマである。 本番環境では、ローテーション、失効、鍵の長さ、ディザスタリカバリなど、さまざまなことを考慮しなければならない。

この例では、クロスリージョンレプリケーションを提供する 2 つの Native HA グループを定義するためのカスタムリソース YAML を提供します。

手順

1. キュー・マネージャのレプリケーション・トラフィックに使用する証明書を作成する。

以下のコマンドを使用して秘密鍵を作成し、それを使用して内部認証局用の自己署名証明書を発行する。 次に、内部認証局で署名されたキューマネージャ用の秘密鍵と証明書を作成します：

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out nhacrr-ca.key
openssl req -x509 -new -nodes -key "nhacrr-ca.key" -sha512 -days 30 -subj "/CN=nhacrr-
selfsigned-ca" -out nhacrr-ca.crt
openssl req -new -nodes -out nhacrr-qm-replication.csr -newkey rsa:4096 -keyout nhacrr-qm-
replication.key -subj "/CN=nhacrr-qm-replication"
openssl x509 -req -in nhacrr-qm-replication.csr -CA nhacrr-ca.crt -CAkey nhacrr-ca.key
-CAcreateserial -out nhacrr-qm-replication.crt -days 7 -sha512
```

これらのコマンドの詳細については、104 ページの『OpenSSL を使用した自己署名 PKI の作成』を参照のこと。

2. キューマネージャの鍵と証明書を含む Kubernetes シークレットを 2 つ作成する。 ローマとロンドンの両サイトに nhacrr-qm-replication という名前のシークレットを作成し、キューマネージャインスタンスの鍵と証明書を格納します。

   シークレットを作成するには以下のコマンドを使用する。 コマンドを実行する前に、それぞれのネームスペース（ローマ、ロンドン）に変更する。

   ```
   kubectl create secret generic nhacrr-qm-replication --type="kubernetes.io/tls" --from-
   file=tls.key=nhacrr-qm-replication.key --from-file=tls.crt=nhacrr-qm-replication.crt --from-
   file=nhacrr-ca.crt
   ```

3. 104 ページの『OpenSSL を使用した自己署名 PKI の作成』 の説明に従って、キュー・マネージャのトラフィック用に証明書のペアを作成する。 ローマとロンドンの両サイトでシークレットを作成する（図では mq-channel と client-app という証明書）。
4. ローマとロンドン用の MQSC コマンドと INI ファイルを含むコンフィグマップを作成する。 新しい永続メッセージキューと SVRCONN チャネルを作成し、チャネルへのアクセスを許可するチャネル認証レコードを追加するための MQSC コマンドを含む ConfigMap リソースを作成する。 Rome サイトで、OCP ウェブ・コンソールに以下の YAML を入力し、 ConfigMaps を選択し、 ConfigMap, を作成するか、 **oc apply** コマンドを使用します。 これをロンドンのサイトでも繰り返す：

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: nhacrr-configmap
   data:
     nhacrr.mqsc: |
       DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
   SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
       SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
   ACTION(REPLACE)
       SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
   MCAUSER('app1') ACTION(REPLACE)
       SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
       DEFINE QLOCAL('EXAMPLE.QUEUE') DEFPSIST(YES) REPLACE
       SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
   AUTHADD(BROWSE,PUT,GET,INQ)
     nhacrr.ini: |
       Service:
           Name=AuthorizationService
           EntryPoints=14
           SecurityPolicy=UserExternal
   ```

   MQSC は、 MTLS.SVRCONN というチャンネルと、 EXAMPLE.QUEUE というキューを定義します。 チャネルは、ステップ 3 で作成したような CN が example-app1 の証明書を提示するクライアントにのみアクセスを許可するように設定されている。
5. Native HA Recovery グループを展開します。 ローマサイトに新しい回復グループを作成する。 rome ネームスペースにいることを確認してから、OCP ウェブ・コンソールで以下の YAML を入力して QueueManager, を作成するか、**oc apply** コマンドを使用してください。 正しいライセンスが指定されていることを確認し、`false` を `true` に変更してライセンスに同意します。

   注：ライブグループとリカバリグループの両方で同じキューマネージャ名を使用する必要があります。 デフォルトでは、 `.spec.queuemanager.name` を指定しない場合、 `.metadata.name` がデフォルトとなる。 以下のいずれかの方法で、名前の一致を実現できる：

- キュー・マネージャ・リソースの名前は各クラスタで同じ名前にし（つまり、 .metadata.name に一致させ）、名前を上書きしないでください
- .metadata.name でキュー・マネージャに個別の名前を付け、 .spec.queuemanager.name でその名前を上書きして、両方のサイトで名前が一致するようにします。

この例では、 .metadata.name を用いて、キューマネージャ名を exampleqm に設定している。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
  labels:
    role: Recovery
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    availability:
      type: NativeHA
      tls:
        secretName: nhacrr-qm-replication
      nativeHAGroups:
        local:
          name: "rome"
          role: "Recovery"
    mqsc:
    - configMap:
        name: nhacrr-configmap
        items:
        - nhacrr.mqsc
    ini:
    - configMap:
        name: nhacrr-configmap
        items:
        - nhacrr.ini
    storage:
      queueManager:
        type: persistent-claim
  version:  9.4.2.0-r1
  pki:
    keys:
      - name: default
        secret:
          secretName: example-qm-tls
          items:
            - tls.key
            - tls.crt
            - ca.crt
```

6. リカバリグループが正しく配置されていることを確認する：

a) キューマネージャリソースを表示するために oc get qmgr exampleqm コマンドを発行し、キューマネージャが実行されるのを待つ。 次のようなメッセージが表示されるはずです。

```
NAME             PHASE
exampleqm   Running
```

b) キューマネージャが実行されているときに、 oc get pods コマンドを使用してポッドの状態を確認します。 1 つのポッドはレディ状態でなければならない：

```
NAME                 READY   STATUS    RESTARTS   AGE
exampleqm-ibm-mq-0   1/1     Running   0          2m45s
exampleqm-ibm-mq-1   0/1     Running   0          2m45s
exampleqm-ibm-mq-2   0/1     Running   0          2m45s
```

c) oc exec -t <leader-pod> -- dspmq -o nativeha -g 。 <leader-pod> はレディ 1/1 ポッドの名前（例では exampleqm-ibm-mq-0 ）である。

```
QMNAME(EXAMPLEQM)                                         ROLE(Leader)
INSTANCE(exampleqm-ibm-mq-0) INSYNC(yes) QUORUM(1/3) GRPLSN() GRPNAME(rome)
GRPROLE(Recovery)
 GRPNAME(rome) GRPROLE(Recovery) GRPADDR(Unknown) GRPVER(9.4.2.0) GRSTATUS(Waiting for
```

```
connection) RCOVLSN() RCOVTIME() BACKLOG(Unknown) INSYNC(Unknown) SYNCTIME()
ALTDATE(2025-04-25) ALTTIME(13.48.15)
```

このステータスの主要部分は以下の通りである：

- Role(Leader)
- QUORUM(1/3) - これは、ライブグループとリカバリーグループが接続するまで 1/3 のままである
- GRSTATUS(Waiting for connection) - ライブグループが接続すると更新されます

7. Native HA Live グループをデプロイする。 ライブ・グループはロンドン・サイトに配備されている：
   ロンドン・サイトに新しいライブ・グループを作成するには、正しいネームスペースにいることを確認し、OCP ウェブ・コンソール、またはコマンド・ラインを使って以下の YAML を入力します。 正しいライセンスが指定されていることを確認し、false を true に変更してライセンスを受け入れる。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
  labels:
    role: Live
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    availability:
      type: NativeHA
      tls:
        secretName: nhacrr-qm-replication
      nativeHAGroups:
        local:
          name: "london"
          role: "Live"
        remotes:
          - name: "rome"
            addresses:
              - "<rome group hostname>:443"
    mqsc:
    - configMap:
        name: nhacrr-configmap
        items:
        - nhacrr.mqsc
    ini:
    - configMap:
        name: nhacrr-configmap
        items:
        - nhacrr.ini
    storage:
      queueManager:
        type: persistent-claim
  version: 9.4.2.0-r1
  pki:
    keys:
      - name: default
        secret:
          secretName: example-qm-tls
          items:
            - tls.key
            - tls.crt
            - ca.crt
```

ライブ・グループがリカバリ・グループに接続するために使用するアドレスを指定する YAML のアドレスを更新する（例では"<rome group hostname>:443 "）。 Rome サイトの exampleqm-ibm-mq-nhacrr ルートの host 属性に置き換えてください。 この属性を取得するには、Rome サイトにいることを確認し、以下のコマンドを使用して nhacrr ルートホスト名を取得します：

```
oc get route exampleqm-ibm-mq-nhacrr --template="{{.spec.host}}"
```

注：

CRR グループが配備されると、2 つのルートが作成される。1 つは qm で終わり、もう 1 つは nhacrr で終わる。 qm のルートはキュー・マネージャーへのメッセージ送信に使われ、nhacrr のルートはグループ間通信に使われる。 YAML のグループのリモートアドレスとして、 nhacrr ルートのホストアドレスを常に追加する。

8. Live グループと Cross Region Replication を確認する。 ステップ 6 の操作を繰り返す。
   a) キューマネージャリソースを表示するために oc get qmgr exampleqm コマンドを発行し、キューマネージャが実行されるのを待つ。 次のようなメッセージが表示されるはずです。

   ```
   NAME            PHASE
   exampleqm   Running
   ```

   b) キューマネージャが実行されているときに、 oc get pods コマンドを使用してポッドの状態を確認します。 1 つのポッドはレディ状態でなければならない：

   ```
   NAME                 READY   STATUS    RESTARTS   AGE
   exampleqm-ibm-mq-0   1/1     Running   0          2m45s
   exampleqm-ibm-mq-1   0/1     Running   0          2m45s
   exampleqm-ibm-mq-2   0/1     Running   0          2m45s
   ```

   c) oc exec -t <active-pod> -- dspmq -o nativeha -g 。 <active-pod> はレディ 1/1 ポッドの名前である（例では exampleqm-ibm-mq-0 ）。

   ```
   QMNAME(EXAMPLEQM)                                         ROLE(Active)
   INSTANCE(exampleqm-ibm-mq-0) INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:87:58322>)
   GRPNAME(london) GRPROLE(Live)
    GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.2.0) GRSTATUS(Normal)
   RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) INITLSN(<0:0:9:4696>)
   INITTIME(2025-05-12T16:58:13.582274Z) LIVETIME(2025-05-12T16:58:16.319844Z)
   ALTDATE(2025-05-16) ALTTIME(14.27.10)
    GRPNAME(rome) GRPROLE(Recovery) GRPADDR(exampleqm-ibm-mq-nhacrr-
   <namespace>.apps.<cluster>) GRPVER(9.4.2.0) CONNGRP(yes) GRSTATUS(Normal)
   RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) BACKLOG(0) INSYNC(yes)
   SYNCTIME(2025-05-16T14:28:38.084565Z) ALTDATE(2025-05-16) ALTTIME(14.28.31)
   ```

   このステータスの主要部分は以下の通りである：

   - Role(Active)
   - QUORUM(3/3) - ライブグループの定足数が満たされている
   - GRSTATUS(Normal)

   d) Rome サイトで oc exec -t <leader-pod> -- dspmq -o nativeha -g コマンドを実行して、Recovery グループのステータスを再度確認してください。

   ```
   QMNAME(EXAMPLEQM)                                         ROLE(Leader)
   INSTANCE(exampleqm-ibm-mq-0) INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:87:58322>)
   GRPNAME(rome) GRPROLE(Recovery)
    GRPNAME(rome) GRPROLE(Recovery) GRPADDR(exampleqm-ibm-mq-nhacrr-
   <namespace>.apps.<cluster>) GRPVER(9.4.3.0) GRSTATUS(Normal) RCOVLSN(<0:0:87:58322>)
   RCOVTIME(2025-05-16T14:27:09.051799Z) BACKLOG(0) INSYNC(yes)
   SYNCTIME(2025-05-16T14:30:40.134627Z) ALTDATE(2025-05-16) ALTTIME(14.27.17)
    GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.3.0) CONNGRP(yes)
   GRSTATUS(Normal) RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z)
   INITLSN(<0:0:9:4696>) INITTIME(2025-05-12T16:58:13.582274Z)
   LIVETIME(2025-05-12T16:58:16.319844Z) ALTDATE(2025-05-16) ALTTIME(14.30.41)
   ```

   QUORUM(3/3)、 GRSTATUS(Normal) 。

9. キュー・マネージャへの接続をテストする。 キュー・マネージャが構成され、利用可能であることを確認するために、108 ページの『ラップトップからキュー・マネージャーへの相互 TLS 接続のテスト』のステップに従い、ステップ 2 で決定されたキュー・マネージャのホスト名が、ロンドン・サイトのルートのホスト名であることを確認してください。
10. Native HA Recovery グループにリモートアドレスを追加する。 ローマがライブグループになったとき、回復グループとしてロンドンに接続できるように、ロンドンのリモートアドレスを知る必要がある。 これを追加するには、Rome サイトの spec.queueManager.availability.nativeHAGroups 、ローカル・スタンザの下に以下のように追加して更新する：

```
          remotes:
            - name: "london"
              addresses:
              - "<london group hostname>:443"
```

アドレスをロンドンサイトの exampleqm-ibm-mq-nhacrr ルートのホスト属性に置き換える。 ロンドン・サイトにいることを確認し、以下のコマンドを使用して nhacrr ルートのホスト名を取得する：

```
oc get route exampleqm-ibm-mq-nhacrr --template="{{.spec.host}}"
```

アップデートが完了したら、ポッドが再起動し、1/1 ポッドになるのを待つ。

11. ローマサイトのホスト名を ccdt.json ファイルに追加します。 ローマがライブサイトとなる切り替え後、クライアントはローマのグループに接続する必要がある。 ローマのホスト名を ccdt.json ファイルに追加すると、この再接続が容易になる。

    Rome サイトにいることを確認し、以下のコマンドを使用して、 qm ルートのホスト名を取得します：

    ```
    oc get route exampleqm-ibm-mq-qm --template="{{.spec.host}}"
    ```

    注：この場合、使用すべきはメッセージ・トラフィック用のルートである。 ccdt.json ファイルに、qm ルートのホストアドレスを追加してください。

    ccdt.json 、2 つ目のホスト名とポートのペアを接続リストに追加し、接続リストは以下のようになる：

    ```
    "connection":
                    [
                        {
                            "host": "<hostname from London qm route>",
                            "port": 443
                        },
                        {
                            "host": "<hostname from Rome qm route>",
                            "port": 443
                        }
                    ],
    ```

12. キューに永続的なメッセージを追加する。 ConfigMap で指定された mqsc ファイルで定義されたキューは、 DEFPSIST(YES) という属性を持っていたので、キューに入れられたすべてのメッセージは永続化され、回復グループに複製されます。 以下のコマンドを使ってメッセージをキューに入れる：

    ```
    /opt/mqm/samp/bin/amqsputc EXAMPLE.QUEUE EXAMPLEQM
    ```

    キュー・マネージャーへの接続が成功すると、以下の応答が出力されます。

    ```
    target queue is EXAMPLE.QUEUE
    ```

    任意のテキストを入力してから **Enter** を押す操作を何回か繰り返すことで、キューに複数のメッセージを書き込みます。 書き込みを終了するには、**Enter** を 2 回押します。

13. ロンドンの Live グループからローマの Recovery グループへのマネージドスイッチオーバーを実行し、Native HA - CRR のセットアップが正しいことを証明する。 236 ページの『ネイティブ HA CRR 構成における計画された切り替えを完了する』 の指示に従ってください。 ライブグループとリカバリグループのロールを更新するとき、変更を反映するために YAML の metadata.labels.role 。 切り替えが完了したら、新しいライブグループからメッセージをローマで取り出すことができます。

    フェイルオーバーとスイッチオーバーの詳細については、 236 ページの『ネイティブ HA CRR の切り替えとフェイルオーバー』 を参照してください。

14. 以下のコマンドを実行して、切り替え前に送信されたメッセージを取得する：

    ```
    /opt/mqm/samp/bin/amqsgetc EXAMPLE.QUEUE EXAMPLEQM
    ```

    ステップ 11 で追加したメッセージが出力されます。 数秒後にコマンドが終了します。

次のタスク

メンテナンスが完了したら、ステップ 12 と 13 のプロセスを繰り返すことで、元のライブグループに切り替えることができます。

また、 237 ページの『ネイティブ HA CRR 構成で予定外のフェイルオーバーを完了する』 にある手順を使用して、計画外のフェイルオーバーを試すこともできます。

異なる証明書を使用して CRR 構成の異なる部分を保護する方法の詳細については、 130 ページの『例：ネイティブ HA CRR で複数の TLS 証明書を使用する場合』 を参照してください。

例 *IBM MQ Operator* を使用して、既存の *Native HA* 構成にリカバリグループを追加します

この例では、IBM MQ オペレータを使用して、OpenShift® Container Platform にシンプルなネイティブ HA 構成を使用したキューマネージャをデプロイする。 この例では、リカバリグループを追加し、Native HA キューマネージャをライブグループに更新して、Native HA Cross-Region Replication (CRR) 構成に拡張しています。

始める前に

この例を完成させるには、まず以下のステップを完了させる必要がある：

- コマンド・ラインで、使用している OpenShift Container Platform （OCP）クラスタにログインする。
- この例では、2 つの OCP プロジェクト/ネームスペースを作成します。 この例では、ロンドンとローマのサイトを参照する。 これらは、単一のクラスタ内のロンドンとローマの 2 つのネームスペース、または各サイトの各クラスタ内のネームスペースのいずれかを指します。
- IBM MQ Operator 3.5.0 またはそれ以上がクラスタにインストールされ、ネームスペースで利用可能であることを確認します。
- OpenSSL コマンドラインツールがインストールされていることを確認する。
- 86 ページの『プルシークレットを作成して「Kubernetes」を使用する準備』 タスクを完了する。
- 98 ページの『IBM MQ Operator を使用した単純なキュー・マネージャーのデプロイ』 を読み、 IBM MQ 演算子の使い方に慣れておくこと。

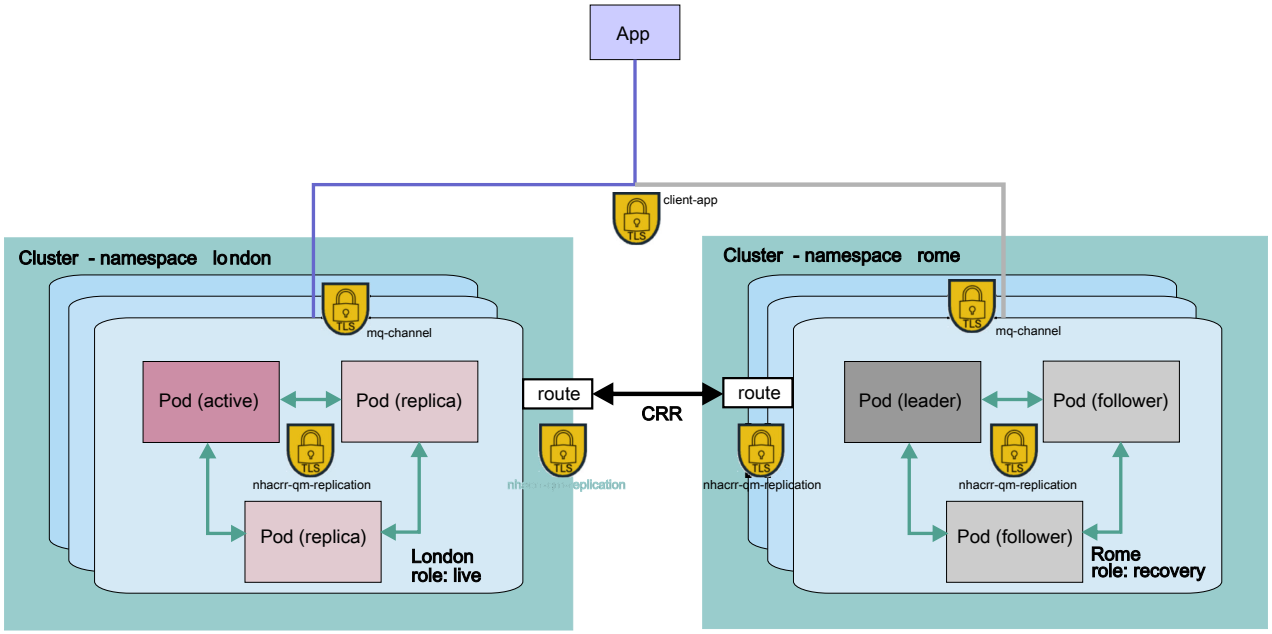
このタスクについて

CRR グループは通常、異なる場所にある 2 つの OCP クラスタに配置されますが、この例では、必要に応じて両方のグループを同じクラスタに配置することができます。 ネイティブ HA CRR の概要については、「 ネイティブ HA クロスリージョンレプリケーション 」を参照してください。

簡単にするために、この例ではキュー・マネージャのレプリケーション・トラフィックをすべて保護するために単一の証明書を使用します。

Native HA CRR で証明書がどのように使用されるかを理解し、グループとインスタンスのトラフィックを保護するために異なる証明書を使用するには、 130 ページの『例：ネイティブ HA CRR で複数の TLS 証明書を使用する場合』 「 Deploying Native HA CRR with multiple certificates 」を参照してください。

重要 : これらの例は、本番環境には適していません。 証明書の管理は、上級ユーザーにとっては複雑なテーマである。 本番環境では、ローテーション、失効、鍵の長さ、ディザスタリカバリなど、さまざまなことを考慮しなければならない。



この例では、クロスリージョンレプリケーションで使用される 2 つの Native HA キューマネージャとリカバリグループを定義するためのカスタムリソース YAML を提供します。

手順

1. ロンドンサイトに Native HA キューマネージャを作成する。 正しいクラスタの London ネームスペースにいることを確認し、 112 ページの『例: IBM MQ Operator を使用したネイティブ HA の構成』 の指示に従って Native HA 設定を作成します。 (このステップを完了するときに、図の中で client-app と mq-channel という 2 つの証明書を作成する)
2. CRR 構成でキュー・マネージャのレプリケーション・トラフィックに使用する証明書を作成します。 以下のコマンドを使用して秘密鍵を作成し、それを使用して内部認証局用の自己署名証明書を発行する。 次に、内部認証局で署名されたキューマネージャ用の秘密鍵と証明書を作成します。

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out nhacrr-ca.key
openssl req -x509 -new -nodes -key "nhacrr-ca.key" -sha512 -days 30 -subj "/CN=nhacrr-
selfsigned-ca" -out nhacrr-ca.crt
openssl req -new -nodes -out nhacrr-qm-replication.csr -newkey rsa:4096 -keyout nhacrr-qm-
replication.key -subj "/CN=nhacrr-qm-replication"
openssl x509 -req -in nhacrr-qm-replication.csr -CA nhacrr-ca.crt -CAkey nhacrr-ca.key
-CAcreateserial -out nhacrr-qm-replication.crt -days 7 -sha512
```

   これらのコマンドの詳細については、 104 ページの『OpenSSL を使用した自己署名 PKI の作成』 を参照のこと。
3. キューマネージャの鍵と証明書を含む Kubernetes シークレットを 2 つ作成する。 ローマとロンドンの両サイトに nhacrr-qm-replication というシークレットを作成し、キューマネージャインスタンスの鍵と証明書を格納します。 以下のコマンドを使用してシークレットを作成し、コマンドを実行する前に各ネームスペース（ロンドンよりローマ）に変更する。

```
kubectl create secret generic nhacrr-qm-replication --type="kubernetes.io/tls" --from-
file=tls.key=nhacrr-qm-replication.key --from-file=tls.crt=nhacrr-qm-replication.crt --from-
file=nhacrr-ca.crt
```

4. レプリケートされたトラフィックに新しいシークレットを使用するように、ネイティブ HA キューマネージャを更新します。 内部グループ・トラフィックに使用される証明書を更新し、その後グループ間トラフィックに使用される証明書を更新するには、 London ネームスペースにいることを確認し、OCP ウェブ・コンソールの QueueManager YAML
   の `.spec.queuemanager.availability.tls.secretname` 属性を以下のように変更します：

```
      tls:
        secretName: nhacrr-qm-replication
```

5. 112 ページの『例: IBM MQ Operator を使用したネイティブ HA の構成』 のステップ 1 で生成した証明書と鍵を含むシークレットを、Rome サイトに作成する。 クラスターにログオンしてシークレットを作成し、Rome ネームスペースに移動して、証明書ファイルが作成されたディレクトリで以下のコマンドを実行します：

```
   kubectl create secret generic example-qm-tls --type="kubernetes.io/tls" --from-
file=tls.key=example-qm.key --from-file=tls.crt=example-qm.crt --from-file=ca.crt
```

6. Rome で MQSC コマンドと INI ファイルを含むコンフィグマップを作成する。 ロンドンのサイトで作成した設定マップをローマで複製する。 Rome ネームスペースにいることを確認し、OCP ウェブ・コンソールで以下の YAML を入力します。 ConfigMaps を選択し、 Create ConfigMap を選択するか、 oc apply コマンドを使用します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-nativeha-configmap
data:
  example-tls.mqsc: |
    DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(BROWSE,PUT,GET,INQ)
  example-tls.ini: |
    Service:
        Name=AuthorizationService
        EntryPoints=14
        SecurityPolicy=UserExternal
```

MQSC は、 MTLS.SVRCONN というチャンネルと、 EXAMPLE.QUEUE というキューを定義する。 チャネルは、ステップ 4 で作成した example-qm-tls に格納されているような、CN が example-app1 の証明書を提示するクライアントにのみアクセスを許可するように設定されている。

7. Native HA Recovery グループを展開します。 以下のカスタムリソース YAML を使用して、Rome サイトに新しい回復グループを作成します。 Rome ネームスペースにいることを確認してから、OCP ウェブコンソールで以下の YAML を入力してキューマネージャを作成するか、 oc apply コマンドを使用します。 正しいライセンスが指定されていることを確認し、false を true に変更してライセンスを受け入れる。

注 : ライブグループとリカバリグループの両方で同じキューマネージャ名を使用する必要があります。 デフォルトでは、 .spec.queuemanager.name を指定しない場合、 .metadata.name がデフォルトとなる。 以下のいずれかの方法で、名前の一致を実現できる：

- キュー・マネージャ・リソースの名前は各クラスタで同じ名前にし（つまり、 .metadata.name に一致させ)、名前を上書きしないでください。
- .metadata.name でキュー・マネージャに個別の名前を付け、 .spec.queuemanager.name でその名前を上書きして、両方のサイトで名前が一致するようにします。

この例では、 .metadata.name を用いて、キューマネージャ名を exampleqm に設定している。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
  labels:
    role: Recovery
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    availability:
      type: NativeHA
```

```
      tls:
        secretName: nhacrr-qm-replication
      nativeHAGroups:
        local:
          name: "rome"
          role: "Recovery"
  mqsc:
  - configMap:
      name: example-nativeha-configmap
      items:
      - example-tls.mqsc
  ini:
  - configMap:
      name: example-nativeha-configmap
      items:
      - example-tls.ini
  storage:
    queueManager:
      type: persistent-claim
version: 9.4.2.0-r1
pki:
  keys:
    - name: default
      secret:
        secretName: example-qm-tls
        items:
          - tls.key
          - tls.crt
          - ca.crt
```

8. グループが正しくデプロイされたことを確認します（このコマンドは Rome サイトで実行する必要があります)。

   a) キューマネージャリソースを表示するために `oc get qmgr exampleqm` コマンドを発行し、キューマネージャが実行されるのを待つ：

   ```
   NAME            PHASE
   exampleqm   Running
   ```

   b) キュー・マネージャが実行されているとき、 `oc get pods` を使ってポッドの状態をチェックすることができる：

   ```
   NAME                  READY   STATUS    RESTARTS   AGE
   exampleqm-ibm-mq-0   1/1     Running   0          2m45s
   exampleqm-ibm-mq-1   0/1     Running   0          2m45s
   exampleqm-ibm-mq-2   0/1     Running   0          2m45s
   ```

   c) 以下のコマンドを実行して、回復グループのステータスを確認する：

   ```
   oc exec -t <leader-pod> -- dspmq -o nativeha -g
   ```

   ここで、 `<leader-pod>` はレディ 1/1 ポッドの名前である（例では `exampleqm-ibm-mq-0` ）：

   ```
   QMNAME(EXAMPLEQM)                                         ROLE(Leader)
   INSTANCE(exampleqm-ibm-mq-0) INSYNC(yes) QUORUM(1/3) GRPLSN() GRPNAME(rome)
   GRPROLE(Recovery)
    GRPNAME(rome) GRPROLE(Recovery) GRPADDR(Unknown) GRPVER(9.4.2.0) GRSTATUS(Waiting for
   connection) RCOVLSN() RCOVTIME() BACKLOG(Unknown) INSYNC(Unknown) SYNCTIME()
   ALTDATE(2025-04-25) ALTTIME(13.48.15)
   ```

   このステータスの主要部分は以下の通りである：

   - `Role(Leader)`
   - `QUORUM(1/3)` - これは、ライブグループとリカバリーグループが接続するまで 1/3 のままである
   - `GRSTATUS(Waiting for connection)` - ライブグループが接続すると更新されます

9. Live グループを使用するように、ネイティブ HA キューマネージャをアップデートします。

   a) Rome サイトで以下のコマンドを使用して回復グループアドレスを取得します：

   ```
   oc get route exampleqm-ibm-mq-nhacrr --template="{{.spec.host}}"
   ```

注: CRR グループが配備されると、2 つのルートが作成される。1 つは qm で終わり、もう 1 つは nhacrr で終わる。 qm のルートはキュー・マネージャーへのメッセージ送信に使われ、nhacrr のルートはグループ間通信に使われる。 以下の YAML に nhacrr ルートのホストアドレスをグループのリモートアドレスとして追加していることを確認してください。

b) 以下の YAML のアドレスは、コマンドによって返されるホスト名で置き換えてください。これは、ライブグループが回復グループに接続するために使用するアドレスを指定します。 OCP ウェブ・コンソールの London QueueManager YAML に、type: NativeHA 属性の直下に以下の YAML を追加してください：

```
      nativeHAGroups:
        local:
          name: "london"
          role: "Live"
        remotes:
          - name: "rome"
            addresses:
              - "<rome group hostname>:443"
```

YAML を保存する前に、メタデータスタンザの先頭に Live role ラベルを追加します：

```
metadata:
  labels:
    role: Live
```

10. Live グループと Cross Region Replication を確認する。

a) ロンドン・サイト上で、 oc get qmgr exampleqm コマンドを発行してキュー・マネージャ・リソースを表示し、キュー・マネージャが実行されるのを待つ。 次のようなメッセージが表示されるはずです。

```
NAME             PHASE
exampleqm   Running
```

b) キューマネージャが実行されているときに、 oc get pods コマンドを使用してポッドの状態を確認します。 1 つのポッドはレディ状態でなければならない：

```
NAME                 READY   STATUS    RESTARTS   AGE
exampleqm-ibm-mq-0   1/1     Running   0          2m45s
exampleqm-ibm-mq-1   0/1     Running   0          2m45s
exampleqm-ibm-mq-2   0/1     Running   0          2m45s
```

c) 以下のコマンドを実行して、ライブ・グループのステータスをチェックする：

```
oc exec -t <active-pod> -- dspmq -o nativeha -g
```

ここで、 <active-pod> はレディ 1/1 ポッドの名前である（例では exampleqm-ibm-mq-0 ）：

```
QMNAME(EXAMPLEQM)                                         ROLE(Active)
INSTANCE(exampleqm-ibm-mq-0) INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:13:21327>)
GRPNAME(london) GRPROLE(Live)
GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.3.0) GRSTATUS(Normal)
RCOVLSN(<0:0:13:21327>) RCOVTIME(2025-06-04T16:22:54.414645Z) INITLSN(<0:0:9:5720>)
INITTIME(2025-06-04T16:12:09.629132Z) LIVETIME(2025-06-04T16:12:11.825065Z)
ALTDATE(2025-06-04) ALTTIME(16.23.01)
GRPNAME(rome) GRPROLE(Recovery) GRPADDR(exampleqm-ibm-mq-nhacrr-
<namespace>.apps.<cluster>) GRPVER(9.4.3.0) CONNGRP(yes) GRSTATUS(Normal)
RCOVLSN(<0:0:13:21327>) RCOVTIME(2025-06-04T16:22:54.414645Z) BACKLOG(0) INSYNC(yes)
SYNCTIME(2025-06-04T16:29:17.206996Z) ALTDATE(2025-06-04) ALTTIME(16.29.12)
```

このステータスの主要部分は以下の通りである：

- Role(Active)
- QUORUM(3/3) - ライブグループの定足数が満たされている
- GRSTATUS(Normal)

d) ローマのサイトで同じコマンドを実行し、QUORUM(3/3) と GRSTATUS(Normal) も更新されていることを確認する：

```
QMNAME(EXAMPLEQM)                                         ROLE(Leader)
INSTANCE(exampleqm-ibm-mq-0) INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:87:58322>)
GRPNAME(rome) GRPROLE(Recovery)
 GRPNAME(rome) GRPROLE(Recovery) GRPADDR(exampleqm-ibm-mq-nhacrr-
rome.apps.destructive-1.cp.fyre.ibm.com) GRPVER(9.4.3.0) GRSTATUS(Normal)
RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) BACKLOG(0) INSYNC(yes)
SYNCTIME(2025-05-16T14:30:40.134627Z) ALTDATE(2025-05-16) ALTTIME(14.27.17)
 GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.3.0) CONNGRP(yes)
GRSTATUS(Normal) RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z)
INITLSN(<0:0:9:4696>) INITTIME(2025-05-12T16:58:13.582274Z)
LIVETIME(2025-05-12T16:58:16.319844Z) ALTDATE(2025-05-16) ALTTIME(14.30.41)
```

11. Native HA Recovery グループにリモートアドレスを追加する。 ローマがライブグループになったとき、回復グループとしてロンドンに接続できるように、ロンドンのリモートアドレスを知る必要がある。 これを追加するには、Rome サイトの `spec.queueManager.availability.nativeHAGroups` 、ローカル・スタンザの下に以下を追加して更新する：

```
        remotes:
          - name: "london"
            addresses:
              - "<london group hostname>:443"
```

<london group hostname>をロンドンサイトの `exampleqm-ibm-mq-nhacrr` ルートの host 属性に置き換えてください。 ロンドン・サイトにいることを確認し、以下のコマンドを使用して nhacrr ルートのホスト名を取得する：

```
oc get route exampleqm-ibm-mq-nhacrr --template="{{.spec.host}}"
```

アップデートが完了したら、ポッドが再起動し、1/1 ポッドになるのを待つ。

12. ローマサイトのホスト名を `ccdt.json` ファイルに追加します。 ローマがライブサイトとなる切り替え後、クライアントはローマのグループに接続する必要があり、ローマのホスト名を `ccdt.json` ファイルに追加することでこれが可能になる。

    a) Rome サイトにいることを確認し、以下のコマンドを使用して qm ルートのホスト名を取得します：

    `oc get route exampleqm-ibm-mq-qm --template="{{.spec.host}}"`

    注： この場合、メッセージ・トラフィック用のルートを使用する。 qm ルートのホストアドレスを `ccdt.json` に追加する。

    b) `ccdt.json` 、2 つ目のホスト名とポートのペアを接続リストに追加し、接続リストは以下のようになる：

```
                "connection":
                [
                    {
                        "host": "<hostname from London qm route>",
                        "port": 443
                    },
                    {
                        "host": "<hostname from Rome qm route>",
                        "port": 443
                    }
                ],
```

13. ロンドンからローマへのマネージドスイッチオーバーを実行し、ネイティブ HA - CRR のセットアップが正しいことを証明します。次に、ロンドンのライブグループからローマのリカバリグループへのマネージドスイッチオーバーを実行します。 236 ページの『ネイティブ HA CRR 構成における計画された切り替えを完了する』 の指示に従ってください。 ライブグループとリカバリーグループのロールを更新するとき、YAML の `metadata.labels.role` を更新して変更を反映させると便利かもしれませんが、これは切り替えのために必要なことではありません。 切り替えが完了したら、新しいライブグループからメッセージをローマで取り出すことができます。

    フェイルオーバーとスイッチオーバーの詳細については、 236 ページの『ネイティブ HA CRR の切り替えとフェイルオーバー』 を参照。

次のタスク

メンテナンスが完了したら、ステップ 12 と 13 のプロセスを繰り返すことで、元の Live グループに切り替えることができます。

また、 237 ページの『ネイティブ HA CRR 構成で予定外のフェイルオーバーを完了する』 にある手順を使用して、計画外のフェイルオーバーを試すこともできます。

最後に、異なる証明書を使用して CRR 構成の異なる部分をセキュアにする方法については、Native HA CRR の TLS を設定するを参照してください。

例：ネイティブ *HA CRR* で複数の *TLS* 証明書を使用する場合

始める前に

この例を開始する前に、まず 117 ページの『例 IBM MQ Operator を使用したシンプルなネイティブ HA CRR 構成のデプロイ』（この例では、配置が存在し、London グループがライブグループであることを前提としています）を完了する必要があります。

その後、以下のステップを完了する：

- 使用している OpenShift Container Platform （OCP）クラスタまたはクラスタにログインできることを確認し、関連するプロジェクト/ネームスペースを選択します。 (117 ページの『例 IBM MQ Operator を使用したシンプルなネイティブ HA CRR 構成のデプロイ』 で説明した例では、 OpenShift Container Platform (OCP) プロジェクト/ネームスペースを 2 つ作成し、1 つのクラスタにロンドンとローマの 2 つのネームスペースを作成するか、各クラスタに各サイトのネームスペースを作成しました。 この例では、これらをロンドンとローマのサイトとも呼ぶ)
- IBM MQ Operator 3.5.0 またはそれ以上がクラスタにインストールされ、ネームスペースで利用可能であることを確認します。
- OpenSSL コマンドラインツールがインストールされていることを確認する。
- 86 ページの『プルシークレットを作成して「Kubernetes」を使用する準備』 タスクを完了する。
- 98 ページの『IBM MQ Operator を使用した単純なキュー・マネージャーのデプロイ』 を読み、 IBM MQ 演算子の使い方に慣れておくこと。

このタスクについて

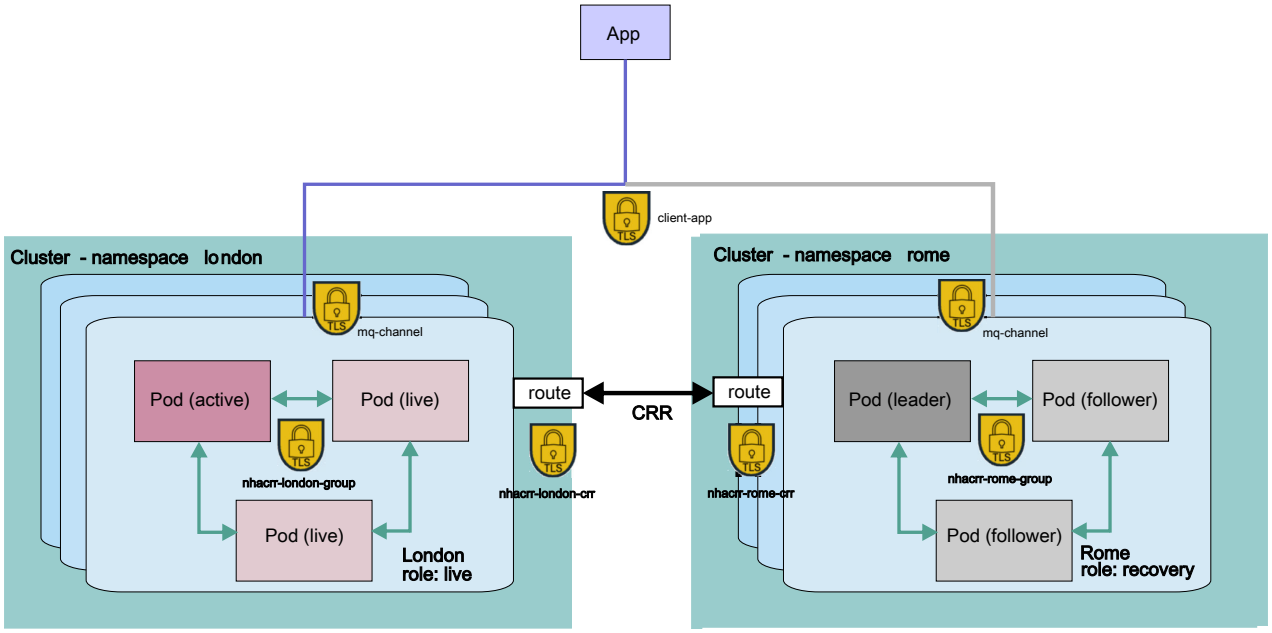
この例では、ネイティブ HA CRR の構成で TLS と証明書がどのように使用されるかの概要から始まり、`nhacrr-london-group`、 `nhacrr-rome-group`、 `nhacrr-london-crr` 、 `nhacrr-rome-crr` のトラフィックがどのように保護されるかを簡単な手順で説明する。

以下の手順のセクションでは、 117 ページの『例 IBM MQ Operator を使用したシンプルなネイティブ HA CRR 構成のデプロイ』 のデプロイメントを変更して、トラフィックのタイプごとに異なる証明書を使用し、CipherSpec を制御する手順を説明します。

ネイティブ HA CRR の TLS 設定

下図は、Native HA CRR 構成のどこで証明書が使用されるかを示している。 証明書は以下のように使用される：

1. キュー・マネージャーとクライアント・アプリケーションの間の通信を保護するために、 `client-app` 。 `mq-channel`
2. キュー・マネージャーのレプリケーション・トラフィックを保護するために、 `nhacrr-london-group`、 `nhacrr-rome-group`、 `nhacrr-london-crr` 、および `nhacrr-rome-crr`



キュー・マネージャーとアプリケーション間のセキュアな通信

図に示す 2 つの Native HA グループは、一緒になって可用性の高いキューマネージャを形成し、クライアントアプリケーションはライブグループを経由して接続することができる。 切り替えが発生すると、アプリケーションは切断され、新しいライブグループに再接続する必要がある。 (両グループの接続詳細をアプリケーション接続リストに追加することで、再接続が容易になる)

ルート経由で接続するアプリケーションが、どちらか、あるいは両方のグループの外部、あるいは別のクラスタにある場合、秘密鍵と証明書を使用して TLS で通信する必要がある。 この図では、クライアントの証明書と鍵は `client-app` と表示され、キュー・マネージャのものは `mq-channel` と表示されている。 `mq-channel` 証明書と秘密鍵は、グループごとに 1 つずつ、2 つのシークレットとして追加する必要がある。

117 ページの『例 IBM MQ Operator を使用したシンプルなネイティブ HA CRR 構成のデプロイ』 は、これらの証明書、鍵、秘密の作成方法を説明します。 以下の手順では、キュー・マネージャのレプリケーショントラフィック用の証明書に焦点を当て、 `mq-channel` または `client-app` の証明書は変更しません。

キュー・マネージャーのレプリケーション・トラフィックを保護

トラフィックはグループ内でもグループ間でも複製される。 2 つのグループ間のネットワーク・トラフィックは常にセキュアでなければならないが、グループ内のトラフィックは一般的にクラスタ内のトラフィックにアクセスできないため、プレーンテキストでも構わない。

1. `nhacrr-london-group` と は、各グループ内のトラフィックを保護するために使用される証明書と秘密鍵を表す。 `nhacrr-rome-group`
2. `nhacrr-london-crr` と は、グループ横断トラフィックの各エンドを保護するために使用される証明書と秘密鍵を表す。 `nhacrr-rome-crr`

例 IBM MQ オペレータを使用したシンプルな Native HA CRR 構成の展開では、各ネームスペースの `nhacrr-qm-replication` シークレットに格納された同じ証明書と秘密鍵が、 `nhacrr-london-group`、 `nhacrr-rome-group`、 `nhacrr-london-crr` 、 `nhacrr-rome-crr` のトラフィックを保護するために使用される。

重要 : これらの例は、本番環境には適していません。 証明書の管理は、上級ユーザーにとっては複雑なテーマである。 本番環境では、ローテーション、失効、鍵の長さ、ディザスタリカバリなど、さまざまなことを考慮しなければならない。

この例では、トラフィックの種類ごとに異なる証明書を使用する方法を示している。

手順

1. キュー・マネージャのレプリケーショントラフィックに使用する 4 セットの鍵と証明書を作成します。 この例でこれらを作成するには、以下のコマンドを実行する。 各コマンド・セットは、秘密鍵を作成し、それを使用して内部認証局用の自己署名証明書を発行し、次に内部認証局で署名された秘密鍵と証明書を作成する。 この演習を繰り返して、 nhacrr-london-group、 nhacrr-rome-group、 nhacrr-london-crr 、 nhacrr-rome-crr トラフィックの証明書を作成する。

   コマンドの詳細については、 104 ページの『OpenSSL を使用した自己署名 PKI の作成』 を参照のこと。

   注 : この例では、証明書チェーンごとに別々の CA を作成していますが、これは、必要に応じてグループ間の信頼の設定を示すためです。 同じ組織内の証明書は、共通の CA を使用することがよくあります。この場合、CA はすでに信頼されているので、キューマネージャ YAML のトラストスタンザは必要ありません。

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out nhacrr-london-group-
ca.key openssl req -x509 -new -nodes -key
"nhacrr-london-group-ca.key" -sha512 -days 30 -subj "/CN=nhacrr-london-group-selfsigned-ca"
-out nhacrr-london-group-ca.crt openssl
req -new -nodes -out nhacrr-london-group.csr -newkey rsa:4096 -keyout nhacrr-london-
group.key -subj "/CN=nhacrr-london-group"
openssl x509 -req -in nhacrr-london-group.csr -CA nhacrr-london-group-ca.crt -CAkey nhacrr-
london-group-ca.key -CAcreateserial -out
nhacrr-london-group.crt -days 7 -sha512
```

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out nhacrr-rome-group-ca.key
openssl req -x509 -new -nodes -key
"nhacrr-rome-group-ca.key" -sha512 -days 30 -subj "/CN=nhacrr-rome-group-selfsigned-ca"
-out nhacrr-rome-group-ca.crt openssl req -new
-nodes -out nhacrr-rome-group.csr -newkey rsa:4096 -keyout nhacrr-rome-group.key -subj "/
CN=nhacrr-rome-group" openssl x509
-req -in nhacrr-rome-group.csr -CA nhacrr-rome-group-ca.crt -CAkey nhacrr-rome-group-ca.key
-CAcreateserial -out nhacrr-rome-group.crt
-days 7 -sha512
```

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out nhacrr-london-crr-ca.key
openssl req -x509 -new -nodes -key
"nhacrr-london-crr-ca.key" -sha512 -days 30 -subj "/CN=nhacrr-london-crr-selfsigned-ca"
-out nhacrr-london-crr-ca.crt openssl req -new -nodes
-out nhacrr-london-crr.csr -newkey rsa:4096 -keyout nhacrr-london-crr.key -subj "/CN=nhacrr-
london-crr" openssl x509 -req -in
nhacrr-london-crr.csr -CA nhacrr-london-crr-ca.crt -CAkey nhacrr-london-crr-ca.key
-CAcreateserial -out nhacrr-london-crr.crt -days 7 -sha512
```

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out nhacrr-rome-crr-ca.key
openssl req -x509 -new -nodes -key
"nhacrr-rome-crr-ca.key" -sha512 -days 30 -subj "/CN=nhacrr-rome-crr-selfsigned-ca" -out
nhacrr-rome-crr-ca.crt openssl req -new -nodes -out
nhacrr-rome-crr.csr -newkey rsa:4096 -keyout nhacrr-rome-crr.key -subj "/CN=nhacrr-rome-
crr" openssl x509 -req -in nhacrr-rome-crr.csr
-CA nhacrr-rome-crr-ca.crt -CAkey nhacrr-rome-crr-ca.key -CAcreateserial -out nhacrr-rome-
crr.crt -days 7 -sha512
```

2. Rome 用の鍵と証明書を含む Kubernetes シークレットを作成する。 以下のコマンドを使用して、Rome サイトに 2 つの TLS シークレットを作成します：

   a. 内部 Rome グループのトラフィックを保護するための 1 つの証明書。

   b. ローマ・グループからのグループ横断トラフィックを保護するための証明書。

```
kubectl create secret generic nhacrr-rome-group --type="kubernetes.io/tls" --from-
file=tls.key=nhacrr-rome-group.key --from-file=tls.crt=nhacrr-rome-group.crt --from-
file=nhacrr-rome-group-ca.crt
kubectl create secret generic nhacrr-rome-crr --type="kubernetes.io/tls" --from-
file=tls.key=nhacrr-rome-crr.key --from-file=tls.crt=nhacrr-rome-crr.crt --from-file=nhacrr-
rome-crr-ca.crt
```

次に、以下のコマンドを使用して、ロンドン・グループが信頼するローマ証明書を含む 2 つのシークレットを作成する：

```
kubectl create secret generic nhacrr-london-group-trust --from-file=tls.crt=nhacrr-london-group.crt --from-file=ca.crt=nhacrr-london-group-ca.crt --type=opaque
kubectl create secret generic nhacrr-london-crr-trust --from-file=tls.crt=nhacrr-london-crr.crt  --from-file=ca.crt=nhacrr-london-crr-ca.crt --type=opaque
```

3. ロンドン用の鍵と証明書を含む Kubernetes シークレットを作成する。 以下のコマンドを使用して、ロンドン・サイトに 2 つの TLS シークレットを作成する：

   a. ロンドン・グループ内部のトラフィックを保護するための 1 つの証明書。

   b. ロンドン・グループからのグループ横断トラフィックを保護するための証明書。

```
kubectl create secret generic nhacrr-london-group --type="kubernetes.io/tls" --from-file=tls.key=nhacrr-london-group.key --from-file=tls.crt=nhacrr-london-group.crt --from-file=nhacrr-london-group-ca.crt
kubectl create secret generic nhacrr-london-crr --type="kubernetes.io/tls" --from-file=tls.key=nhacrr-london-crr.key --from-file=tls.crt=nhacrr-london-crr.crt --from-file=nhacrr-london-crr-ca.crt
```

   以下のコマンドを使用して、Rome グループが信頼するロンドン証明書を含む 2 つのシークレットを作成する：

```
kubectl create secret generic nhacrr-rome-group-trust --from-file=tls.crt=nhacrr-rome-group.crt --from-file=ca.crt=nhacrr-rome-group-ca.crt --type=opaque
kubectl create secret generic nhacrr-rome-crr-trust --from-file=tls.crt=nhacrr-rome-crr.crt  --from-file=ca.crt=nhacrr-rome-crr-ca.crt --type=opaque
```

4. ローマとロンドンのサイトでは、異なる証明書でトラフィックを保護する。 117 ページの『例 IBM MQ Operator を使用したシンプルなネイティブ HA CRR 構成のデプロイ』 で作成した配置は、すべてのレプリケーション トラフィックを nhacrr-qm-replication シークレットで保護しました。 このステップでは、各グループ内のトラフィックをそれぞれ nhacrr-rome-group 、 nhacrr-london-group シークレットで保護するように設定を変更する。

   .spec.queuemanager.availability.tls.secretName 属性は、サイト・グループ内のトラフィックを保護するために使用されるシークレットを指定します。 他の TLS 属性が定義されていない場合、秘密鍵はグループ間のトラフィックを保護するためにも使われる。

   a) Rome サイトで、OCP ウェブ・コンソールの QueueManager YAML の .spec.queuemanager.availability.tls.secretName 属性を nhacrr-rome-group に変更する：

```
      tls:
        secretName: nhacrr-rome-group
```

   b) YAML に spec.queueManager.availability.NativeHAGroups.remotes.trust stanza を追加し、信頼するものとして nhacrr-london-group-trust secret を追加する：

```
...
      nativeHAGroups:
        local:
          name: "rome"
          role: "Recovery"
        remotes:
          - name: "london"
            addresses:
              - "exampleqm-ibm-mq-nhacrr-london.apps.trouble.cp.fyre.ibm.com:443"
            trust:
              - secret:
                  items:
                    - tls.crt
                    - ca.crt
                  secretName: nhacrr-london-group-trust
...
```

   nhacrr-london-group-trust シークレットには、ロンドン・サイトの nhacrr-london-group シークレットに格納されている証明書と CA が含まれる。 nhacrr-london-group のシークレッ

トは、ロンドン・グループからのトラフィックをセキュアにするために使われるため、ローマ・グループはこのシークレットを使って、提供された証明書が信頼できるものであることを確認する。

c) ロンドンのサイトで同等の手順を行う。 `.spec.queuemanager.availability.tls.secretName` ：

```
      tls:
        secretName: nhacrr-london-group
```

d) `spec.queueManager.availability.NativeHAGroups.remotes.trust` スタンザに `nhacrr-rome-group-trust` シークレットを追加する：

```
...
      nativeHAGroups:
        local:
          name: "london"
          role: "Live"
        remotes:
          - name: "rome"
            addresses:
              - "<rome group hostname>:443"
            trust:
              - secret:
                  items:
                    - tls.crt
                    - ca.crt
                  secretName: nhacrr-rome-group-trust
...
```

5. 新しい証明書でクロスリージョンレプリケーションを検証する。 ロンドン・サイトでは、 `oc get qmgr exampleqm` コマンドを発行してキュー・マネージャ・リソースを表示し、キュー・マネージャが実行されるのを待つ。 それから、 `oc get pods` コマンドを使ってポッドの状態をチェックする。 1 つのポッドがレディ状態になっているはずで、これがアクティブポッドである。

   次に、`oc exec -t <active-pod> -- dspmq -o nativeha -g` コマンドを実行して、ライブ・グループのステータスをチェックすることができる。 `<active-pod>` を、`oc get pods` コマンドによって返されたレディ 1/1 ポッド名に置き換える。

```
QMNAME(EXAMPLEQM)                                         ROLE(Active) INSTANCE(exampleqm-
ibm-mq-0) INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:87:58322>) GRPNAME(london) GRPROLE(Live)
 GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.3.0) GRSTATUS(Normal)
RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) INITLSN(<0:0:9:4696>)
INITTIME(2025-05-12T16:58:13.582274Z) LIVETIME(2025-05-12T16:58:16.319844Z)
ALTDATE(2025-05-16) ALTTIME(14.27.10)
 GRPNAME(rome) GRPROLE(Recovery) GRPADDR(exampleqm-ibm-mq-nhacrr-
<namespace>.apps.<cluster>) GRPVER(9.4.3.0) CONNGRP(yes) GRSTATUS(Normal)
RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) BACKLOG(0) INSYNC(yes)
SYNCTIME(2025-05-16T14:28:38.084565Z) ALTDATE(2025-05-16) ALTTIME(14.28.31)
```

   このステータスの主要部分は以下の通りである：

   - `Role(Active)`
   - `QUORUM(3/3)` - ライブグループの定足数が満たされている
   - `GRSTATUS(Normal)`

6. グループ間のトラフィックを別々の証明書で保護する。 グループ内のトラフィックとグループ間のトラフィックに別々の証明書を使用することができる。 このステップでは、 `nhacrr-rome-crr` と `nhacrr-london-crr` の秘密鍵に含まれる証明書はグループ間のトラフィックを保護するために使用され、 `nhacrr-rome-group` と `nhacrr-london-group` の秘密鍵は各グループ内のトラフィックを保護するために残される。 1 つはグループ間のトラフィックのシークレットを指定するため、もう 1 つは使用する新しい証明書に合わせてトラストを変更するためである。

   注: `.spec.queuemanager.availability.NativeHAGroups.local.tls.secretName` と `.spec.queuemanager.availability.tls.secretName` 属性に使用されるシークレットに同じ証明書が含まれている場合、衝突が発生し、キューマネージャインスタンスを起動することができません。 単一の証明書を使用する場合は、 `.spec.queuemanager.availability.tls.secretName` のみを設定する。

a) Rome サイトで、OCP ウェブ・コンソールの QueueManager YAML に spec.queuemanager.availability.NativeHAGroups.local.tls.key.secretName 属性を追加し、 nhacrr-rome-crr に設定します。 そして、 .spec.queueManager.availability.NativeHAGroups.remotes.trust.secret.secretName を nhacrr-london-crr-trust に変更する：

```
...
      nativeHAGroups:
        local:
          name: "rome"
          role: "Recovery"
          tls:
            key:
              secretName: nhacrr-rome-crr
        remotes:
          - name: "london"
            addresses:
              - "<london group hostname>:443"
            trust:
              - secret:
                  items:
                    - tls.crt
                    - ca.crt
                  secretName: nhacrr-london-crr-trust
...
```

b) ロンドンのサイトで同等の手順を行う。 OCP ウェブ・コンソールの QueueManager YAML に .spec.queuemanager.availability.NativeHAGroups.local.tls.key.secretName 属性を追加し、 nhacrr-london-crr に設定してください。 そして、.spec.queueManager.availability.NativeHAGroups.remotes.trust.secret.secretName を nhacrr-rome-crr-trust に変更する。

```
...
      nativeHAGroups:
        local:
          name: "london"
          role: "Live"
          tls:
            key:
              secretName: nhacrr-london-crr
        remotes:
          - name: "london"
            addresses:
              - "<london group hostname>:443"
            trust:
              - secret:
                  items:
                    - tls.crt
                    - ca.crt
                  secretName: nhacrr-rome-crr-trust
...
```

7. 新しい証明書でクロスリージョンレプリケーションを検証する。 ロンドン・サイトでは、 oc get qmgr exampleqm コマンドを発行してキュー・マネージャ・リソースを表示し、キュー・マネージャが実行されるのを待つ。 次に、oc get pods を使ってポッドの状態をチェックします。この場合も、1 つのポッドが準備完了状態になっているはずで、これがアクティブなポッドです。

   oc exec -t <active-pod> -- dspmq -o nativeha -g コマンドを発行して、ライブグループのステータスをチェックする。 <active-pod> を、oc get pods コマンドが返すレディ 1/1 ポッド名に置き換える。

```
QMNAME(EXAMPLEQM)                                         ROLE(Active) INSTANCE(exampleqm-
ibm-mq-0) INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:87:58322>) GRPNAME(london) GRPROLE(Live)
 GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.3.0) GRSTATUS(Normal)
RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) INITLSN(<0:0:9:4696>)
INITTIME(2025-05-12T16:58:13.582274Z) LIVETIME(2025-05-12T16:58:16.319844Z)
ALTDATE(2025-05-16) ALTTIME(14.27.10)
 GRPNAME(rome) GRPROLE(Recovery) GRPADDR(exampleqm-ibm-mq-nhacrr-
<namespace>.apps.<cluster>) GRPVER(9.4.3.0) CONNGRP(yes) GRSTATUS(Normal)
RCOVLSN(<0:0:87:58322>) RCOVTIME(2025-05-16T14:27:09.051799Z) BACKLOG(0) INSYNC(yes)
SYNCTIME(2025-05-16T14:28:38.084565Z) ALTDATE(2025-05-16) ALTTIME(14.28.31)
```

このステータスの主要部分は以下の通りである：

- Role(Active)
- QUORUM(3/3) - ライブグループの定足数が満たされている
- GRSTATUS(Normal)

最後に、 kubectl logs <active-pod> コマンドを発行すると、ログに以下のメッセージが表示され、異なるタイプのトラフィックに正しい証明書が使用されていることが確認できるはずです。 グループ間のトラフィックについては、次のようになるはずだ：

```
AMQ3307I: The Native HA secure connection between the local group and 'rome' is using
certificate DN 'CN=nhacrr-london-crr'
```

そして、内部グループのトラフィックについては、以下のメッセージが表示されるはずである。 <replica-pod> は、レディ 0/1 のステータスを持つポッドのいずれかの名前である：

```
AMQ3306I: The Native HA secure connection between the local instance and '<replica-pod>'
accepted remote certificate DN 'CN=nhacrr-london-group'
```

8. レプリケートされたトラフィック用に特定の CipherSpecs。

   レプリケートされたトラフィックに対して特定の CipherSpecs が定義されていない場合は、 ANY_TLS13_OR_HIGHER が使用され、プロトコルと暗号化はインスタンス間でネゴシエートされる。 CipherSpecs の詳細については、 IBM MQ CipherSpecs 読み取り有効化をサポートします。

   グループ内とグループ間のトラフィックの両方に CipherSpec を指定するには、 QueueManager YAML に .spec.queuemanager.availability.tls.cipherSpec 属性を追加する。

   ローマとロンドンの両サイトの OCP ウェブ・コンソールの QueueManager YAML に以下の YAML を追加して、すべてのレプリケーション・トラフィックに TLS_AES_256_GCM_SHA384 CipherSpec を使用する：

   ```
         tls:
           secretName: nhacrr-rome-group
           cipherSpec: "TLS_AES_256_GCM_SHA384"
   ```

   グループ間のトラフィックに別々の CipherSpec を指定する必要がある場合、 QueueManager YAML に .spec.queuemanager.availability.NativeHAGroups.local.tls.cipherSpec 属性を追加することができる。

   以下の YAML をローマとロンドンの両サイトの OCP ウェブ・コンソールの QueueManager YAML に追加して、グループ間のトラフィックに TLS_AES_128_GCM_SHA256 CipherSpec を使用し、グループ内のトラフィックに TLS_AES_256_GCM_SHA384 CipherSpec を使用するようにしてください：

   ```
         nativeHAGroups:
           local:
          ...
             tls:
               key:
                 secretName: nhacrr-rome-crr
               cipherSpec: "TLS_AES_128_GCM_SHA256"
   ```

   インスタンスの再起動後、ロンドン・サイトで以下のコマンドを実行する：

   ```
   kubectl logs
           <active-pod>
   ```

   以下の例のようなメッセージがログに表示されるはずだ。 この例では、グループ・インスタンス間の通信に CipherSpec を使用している。 TLS_AES_256_GCM_SHA384:

   ```
   [CommentInsert1(exampleqm-ibm-mq-2), CommentInsert2(exampleqm-ibm-mq-replica-2(9414)),
           CommentInsert3(TLS_AES_256_GCM_SHA384)]
   ```

   この例では、Rome グループへの接続に CipherSpec。 TLS_AES_128_GCM_SHA256:

```
AMQ3253I: A Native HA recovery group connection to 'rome' has
        been established using the 'TLS_AES_128_GCM_SHA256' CipherSpec.
```

9. グループ内でプレーンテキストのトラフィックを許可する。 グループ間のトラフィックは常に安全かつ暗号化されていなければならないが、グループ内のトラフィックは暗号化されていなくても構わない。 ロンドン・サイトの `.spec.queuemanager.availability.tls.cipherSpec` を空白に設定し、ロンドン・グループの内部トラフィックの暗号化を解除する：

```
      tls:
        cipherSpec: ''
        secretName: nhacrr-london-group
```

インスタンスの再起動後、ロンドン・サイトで以下のコマンドを実行する：

```
kubectl logs
        <active-pod>
```

ログには次のようなメッセージが表示される：

`AMQ9722W: Plain text communication is enabled.`

10. グループ内のトラフィックのセキュリティを無効にする。 `.spec.queuemanager.availability.tls` スタンザを削除すると、グループの内部トラフィックのセキュリティが完全に無効になる。

注 : グループ間のトラフィックは安全でなければならないので、 `.spec.queuemanager.availability.tls.secretName` が削除された場合は、 `.spec.queuemanager.availability.NativeHAGroups.local.tls.key.secretName` を指定しなければならない。

Rome サイトの OCP ウェブ・コンソールの QueueManager YAML から `.spec.queuemanager.availability.tls` スタンザを削除して、Rome グループのセキュ リティを無効にします：

```
      tls:
        secretName: nhacrr-rome-group
        cipherSpec: "TLS_AES_256_GCM_SHA384"
```

インスタンスの再起動後、Rome サイトで `kubectl logs <active-pod>` コマンドを実行すると、ログに以下のようなメッセージが表示されます：

```
AMQ3213I: Native HA inbound connection accepted from 'exampleqm-ibm-mq-2'.
[CommentInsert1(exampleqm-ibm-mq-2), CommentInsert2(<ip>)]
AMQ3211I: Native HA outbound connection established to 'exampleqm-ibm-mq-2'.
[CommentInsert1(exampleqm-ibm-mq-2), CommentInsert2(<replica>(9414))]
```

以下の代わりに

```
AMQ3214I: Native HA inbound secure connection accepted from 'exampleqm-ibm-mq-0'.
[CommentInsert1(exampleqm-ibm-mq-0), CommentInsert2(<ip>), CommentInsert3(<cipherspec>)]
AMQ3212I: Native HA outbound secure connection established to 'exampleqm-ibm-mq-0'.
[CommentInsert1(exampleqm-ibm-mq-0), CommentInsert2(<replica>(9414)),
CommentInsert3(<cipherspec)
```

OpenShift MQ Adv. *IBM MQ* コンテナーのネイティブ *HA* キュー・マネージャーの状況の表示

IBM MQ コンテナーの場合、実行中のいずれかのポッド内で **dspmq** コマンドを実行することで、ネイティブ HA インスタンスの状況を表示できます。

このタスクについて

実行中のポッドのうち 1 つで **dspmq** コマンドを使用すると、キュー・マネージャー・インスタンスの運用状況を表示できます。 返される情報は、インスタンスがアクティブとレプリカのどちらであるかに応じて

異なります。 アクティブ・インスタンスから提供される情報は確定的であり、レプリカ・インスタンスからの情報は古い可能性がある。

V 9.4.2 ネイティブ HA クロスリージョンレプリケーション（CRR）構成の場合、提供される情報も、インスタンスがライブグループまたはリカバリグループのいずれに属しているかによって異なります。 ライブグループからの情報はより確実ですが、リカバリーグループが定期的にしか情報を送り返さないため、まだ古い情報である可能性があります。

以下のアクションを実行できます。

- 現行ノード上のキュー・マネージャー・インスタンスがアクティブかレプリカを表示します。
- 現行ノード上のインスタンスのネイティブ HA の運用状況を表示します。
- ネイティブ HA 構成に属する 3 つのインスタンスすべての運用状況を表示します。
- V 9.4.2 そのインスタンスが所属するグループ、または接続されているグループのステータスを表示します。

以下のステータスフィールドは、キューマネージャインスタンスのネイティブ HA およびネイティブ HA クロスリージョンレプリケーション（CRR）の構成ステータスを報告するために使用されます。 これらのフィールドは常に表示される：

ROLE
インスタンスの現在の役割を指定します。 `Active`、 `Replica`、 `Unknown`、(リカバリグループ用) `Leader`、または `Not Configured` のいずれかです。

INSTANCE
このキュー・マネージャー・インスタンスの作成時に **crtmqm** コマンドの **-lr** オプションを使用してこのキュー・マネージャー・インスタンスに対して指定された名前。

INSYNC
必要な場合にインスタンスがアクティブ・インスタンスとしてテークオーバーできるかどうかを示します。

QUORUM
クォーラムの状況を *number_of_instances_in-sync/number_of_instances_configured* という形式でレポートします。

V 9.4.2 **GRPLSN**
HA グループ内のインスタンスの過半数間で複製されたログシーケンス番号（LSN）を *nnnnn:nnnnn:nnnnn:nnnnn* 形式で指定します。 LSN が不明な場合は空白。

V 9.4.2 **GRPNAME**
HA グループの名前を指定します。 名前が設定されていない場合は空白。

V 9.4.2 **GRPROLE**
HA グループの現在の役割を指定する：

- `Live` - ライブグループ
- `Recovery` - 回復グループ
- `Pending live` - グループになるのを待っている人々 `Live`
- `Pending recovery` - グループになるのを待っている人々 `Recovery`
- `Not configured` - ネイティブ HA は構成されていません
- `Unknown` - 現在の役割を特定できません

これらのフィールドは、`-x` パラメータを使ってすべてのインスタンスに対して拡張情報を要求したときに表示されます：

REPLADDR
キュー・マネージャー・インスタンスの複製アドレス。

CONNACTV
インスタンスがアクティブ・インスタンスに接続されているかどうかを示します。

BACKLOG
アクティブなインスタンスに対して、インスタンスが遅れている KB 数を示します。

CONNINST
指定されたインスタンスがこのインスタンスに接続されているかどうかを示します。

ALTDATE
この情報が最後に更新された日付を示します。 更新されたことがない場合は空白にしてください。

ALTTIME
この情報が最後に更新された時刻を示す。 更新されたことがない場合は空白にしてください。

V 9.4.2 **ACKLSN**
インスタンスが回復ログに書き込まれたことを認めた LSN を *nnnnn:nnnnn:nnnnn:nnnnn* の形式で示します。 LSN 値が不明な場合は空白。

V 9.4.2 **HAStatus**

このインスタンスの動作ステータスを示し、以下の値のいずれかである：

- `Normal` - インスタンスは正常に動作しています。
- `Checking` - インスタンスは、リカバリログの一貫性を確保するためにチェックされています。
- `Synchronizing` - インスタンスにデータを送信して同期させる。
- `Rebasing` - 新しいリカバリログがインスタンスに送信されています。
- `Disk full` - インスタンスのディスクがいっぱいになっている。
- `Disconnected` - インスタンスと通信できません。
- `Unknown` - インスタンスの状態が不明です。

V 9.4.2 **SYNCTIME**
このインスタンスが最後にアクティブなキューマネージャと同期した時刻を ISO 8601 形式で表示します。 時間が不明の場合は空白にしてください。

V 9.4.2 これらのフィールドは、`-g` パラメータを使ってグループ情報を要求したときに表示されます。 これらのフィールドは、グループが Native HA CRR 構成の一部である場合のステータスを表示するのに便利です

V 9.4.2 **GRPNAME**
HA グループの名前を指定します。 名前が設定されていない場合は空白。

V 9.4.2 **GRPROLE**
HA グループの現在の役割を指定する：

- `Live` - ライブグループ
- `Recovery` - 回復グループ
- `Pending live` - グループになるのを待っている人々 `Live`
- `Pending recovery` - グループになるのを待っている人々 `Recovery`
- `Unknown` - 現在の役割を特定できません
- `Not configured` - ネイティブ HA は構成されていません

V 9.4.2 **GRPADDR**
グループに接続するために使用される IP アドレス。 グループへの接続が発生していないため、アドレスが不明な場合は、 Unknown に設定されます。

V 9.4.2 **グループバー**
`V.R.M.F` 形式で現在のグループリーダーのバージョンを示します。 `V.R.M.F` は、バージョン、リリース、修正、および修正パックの番号です。 値が不明な場合は、これが設定されます。 `?.?.?.?`

V 9.4.2 **CONNGRP**

グループがローカルグループとつながっているかどうかを示す：

- yes - グループはつながっている。
- no - グループは切断されています。
- unknown - 接続状態が不明です。
- suspended - グループは接続されていますが、グループ間の構成上の不整合が解決されるまでは、グループはデータを複製できません。

このフィールドは、ローカルグループ以外のグループに対してのみ表示されます。

V 9.4.2 **INSYNC**

これは、ロールが Recovery または Pending live のグループに対してのみ表示されます。 フェイルオーバーが発生した場合に、グループがデータ損失なしに Live になる可能性があるかどうかを示します。 （Recovery グループと Live グループは、ログが非同期で複製されるため、常に同期しているとは限らないことに注意してください。） RCOVLSN と RCOVTIME の値を使用して、予期せぬフェールオーバーでデータ損失が発生する可能性があるかどうかを示します。

V 9.4.2 **SYNCTIME**

これは、ロールが Recovery または Pending live のグループに対してのみ表示されます。 これは、このグループが Live グループと最後に同期した日時を ISO 8601 形式で示しています（不明の場合は空白）。 （Recovery グループと Live グループは、ログが非同期で複製されるため、常に同期しているとは限らないことに注意してください。） RCOVLSN と RCOVTIME の値を使用して、予期せぬフェールオーバーでデータ損失が発生する可能性があるかどうかを示します。

V 9.4.2 **BACKLOG**

これは、ロールが Recovery または Pending live のグループに対してのみ表示されます。 これは、そのグループがライブグループより遅れている KB 数を示しています。

V 9.4.2 **GRSTATUS**

このグループの動作状態を示し、以下の値のいずれかである：

- Normal - グループは通常通り運営されています。
- Checking - グループは、リカバリログの一貫性を確保するために確認されています。
- Synchronizing - グループに同期させるためのデータが送信されています。
- Rebasing - 新しいリカバリログがインスタンスに送信されています。
- Waiting for connection - グループは、Live または Pending Recovery の役割を持つグループからの接続を待っています。
- Partitioned - 2 つのグループが、自分たちはライブグループであると信じている（つまり、「スプリットブレイン」の状態）。
- Unknown - グループの状態が不明です。

V 9.4.2 **RCOVLSN**

nnnnn:nnnnn:nnnnn:nnnnn フォーマットで、グループが回復できるログシーケンス番号(LSN)。 この時点以降にリカバリログに書き込まれたデータは、予期せぬフェイルオーバーが発生した場合に失われる可能性があります。 この値は、**dmpmqlog** コマンドの -s オプションに渡すことができる。

V 9.4.2 **時間**

ISO 8601 形式で、グループが回復できる時間。 この時間以降にリカバリログに書き込まれたデータは、予期せぬフェールオーバーが発生した場合に失われる可能性があります。 この値は、 **dmpmqlog** コマンドの -t または -u オプションに渡すことができます。

V 9.4.2 **イニットエルエスエヌ**

これは、ロールが Live または Pending recovery のグループに対してのみ表示されます。 これは、ネイティブ HA グループが最初にライブになった際に回復された最後のログレコードのログシーケン

ス番号（LSN）を示します。 *nnnnn:nnnnn:nnnnn:nnnnn* という形式で表示され、グループがライブになっていない場合は空白になります。

V 9.4.2 **initTime**
これは、ロールが Live または Pending recovery のグループに対してのみ表示されます。 ネイティブ HA グループが最初に稼働した際に、最後に回復されたログレコードの時刻を ISO 8601 形式で表示します。 グループがライブでない場合は空白。

V 9.4.2 **ライブタイム**
これは、ロールが Live または Pending recovery のグループに対してのみ表示されます。 ISO 8601 形式で、グループがライブグループになった最初の日時を示します。 グループがライブでない場合は空白。

V 9.4.2 **ALTDATE**
この情報が最後に更新された日付を示します。

V 9.4.2 **ALTTIME**
この情報が最後に更新された時刻を示す。

手順

- キュー・マネージャーの一部であるポッドを見つけます。

```
oc get pod --selector app.kubernetes.io/instance=nativeha-qm
```

- いずれかのポッドで dspmq を実行します。

```
oc exec -t Pod dspmq
```

```
oc rsh Pod
```

対話式シェルの場合は、dspmq を直接実行できる場所です。

- キュー・マネージャー・インスタンスがアクティブ・インスタンスとして実行されているか、それともレプリカとして実行されているか判別するには、次のようにします

```
oc exec -t Pod dspmq -o status -m QMgrName
```

BOB という名前のキュー・マネージャーのアクティブ・インスタンスからは、次の状況が報告されます

```
QMNAME(BOB)                STATUS(Running)
```

V 9.4.2 ネイティブ HA CRR 構成のリカバリグループリーダーである BOB のインスタンスは、以下のステータスを報告する：

```
QMNAME(BOB)                STATUS(Recovery group leader)
```

BOB という名前のキュー・マネージャーのレプリカ・インスタンスからは、次の状況が報告されます

```
QMNAME(BOB)                STATUS(Replica)
```

非アクティブ・インスタンスからは、次の状況が報告されます

```
QMNAME(BOB)                STATUS(Ended Immediately)
```

- 指定されたポッド内のインスタンスのネイティブ HA 運用状況を判別するには、次のようにします

```
oc exec -t Pod dspmq -o nativeha -m QMgrName
```

BOB という名前のキュー・マネージャーのアクティブ・インスタンスからは、次のような状況が報告されます

```
QMNAME(BOB)         ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
GRPLSN(<0:25:365:1000>) GRPNAME() GRPROLE(Live)
```

V 9.4.2

BOB という名前のキュー・マネージャーのレプリカ・インスタンスからは、次のような状況が報告されます

```
QMNAME(BOB)         ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
GRPLSN(<0:25:365:1000>) GRPNAME() GRPROLE(Live)
```

V 9.4.2

BOB という名前のキュー・マネージャーの非アクティブ・インスタンスからは、次のような状況が報告されます

```
QMNAME(BOB)         ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3)
GRPLSN() GRPNAME() GRPROLE(Live)
```

V 9.4.2

V 9.4.2 Native HA CRR 構成の Live グループに属する BOB という名前のキューマネージャのアクティブなインスタンスは、以下のステータスを報告するかもしれません：

```
QMNAME(BOB)         ROLE(Active) INSTANCE(inst1) INSYNC(yes) QUORUM(3/3)
GRPLSN(<0:25:365:1000>) GRPNAME(alpha) GRPROLE(Live)
```

V 9.4.2 グループが回復グループになるのを待っているとき、同じインスタンスが次のようなステータスを報告するかもしれない：

```
QMNAME(BOB)         ROLE(Active) INSTANCE(inst1) INSYNC(yes) QUORUM(3/3)
GRPLSN(<0:25:365:1000>) GRPNAME(alpha) GRPROLE(Pending recovery)
```

V 9.4.2 Native HA または Native HA CRR 構成のインスタンスではない CAROL という名前のキューマネージャは、以下のステータスを報告します：

```
QMNAME(CAROL)       ROLE(Not configured) INSTANCE() INSYNC() QUORUM() GRPLSN() GRPNAME()
GRPROLE(Not configured)
```

- ネイティブ HA 構成内のすべてのインスタンスのネイティブ HA 運用状況を判別するには、次のようにします

```
oc exec -t Pod dspmq -o nativeha -x -m QMgrName
```

例えば、キューマネージャー BOB のアクティブなインスタンスが実行されているノードでこのコマンドを発行した場合、すべてのインスタンスが同期されているため、以下のステータスが返される可能性があります

```
QMNAME(BOB)         ROLE(Active) INSTANCE(inst1) INSYNC(yes) QUORUM(2/3)
GRPLSN(<0:25:365:1000>) GRPROLE(Live) GRPNAME()
 INSTANCE(inst1) ROLE(Active)  REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN(<0:26:365:1000>)  HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN (<0:24:365:1000>) HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN (<0:25:365:1000>) HASTATUS(Synchronizing)
SYNCTIME(2023-12-20T17:13:22.123321Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
```

別の例として、遅れをとっているキューマネージャー BOB のレプリカインスタンスを実行しているノードでこのコマンドを発行すると、以下のステータスが返される場合があります

```
QMNAME(BOB)         ROLE(Replica) INSTANCE(inst2) INSYNC(no) QUORUM(2/3)
GRPLSN(<0:25:365:1000>) GRPROLE(Live) GRPNAME()
 INSTANCE(inst1) ROLE(Active)  REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN(<0:26:365:1000>)  HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN (<0:24:365:1000>) HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
```

```
 INSTANCE(inst2) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(no) BACKLOG(435)
CONNINST(yes) ACKLSN (<0:25:365:1000>) HASTATUS(Synchronizing)
SYNCTIME(2023-12-20T17:13:22.123321Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
```

キュー・マネージャー BOB の非アクティブ・インスタンスを実行しているノード上でこのコマンドを発行すると、以下のような状況が表示されます

```
QMNAME(BOB)            ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3) GRPLSN()
GRPROLE() GRPNAME()
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ACKLSN()  HASTATUS(Unknown) SYNCTIME() ALTDATE() ALTTIME()
ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ACKLSN()  HASTATUS(Unknown) SYNCTIME() ALTDATE() ALTTIME()
ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No)      INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ACKLSN()  HASTATUS(Unknown) SYNCTIME() ALTDATE() ALTTIME()
ALTDATE() ALTTIME()
```

どのインスタンスがアクティブでどれがレプリカになるかをまだネゴシエーションしている間にコマンドを発行すると、次の状況が表示されます

```
QMNAME(BOB)              STATUS(Negotiating)
```

- V 9.4.2
  ネイティブ HA CRR 構成内のすべてのネイティブ HA グループの稼動状況を確認する：

```
oc exec -t Pod dspmq -o nativeha -g -m QMgrName
```

Live グループのメンバーで alpha という名前のユーザーにこのコマンドを発行すると、alpha と Recovery グループの beta のステータスが次のようになっている可能性があります

```
GRPNAME(alpha) GRPROLE(Live) GRPADDR(9.20.135.129) GRPVER(09040200)
GRSTATUS(Normal) RCOVLSN(<0:0:104:65163>) RCOVTIME(2023-12-20T17:37:27.114787Z)
INITLSN(<0:0:101:352>) INITTIME(2023-12-19T09:13:27.512294)
LIVETIME(2023-12-19T06:14:27.512294) ALTDATE(2023-12-20) ALTTIME(17.13.27)
```

```
GRPNAME(beta) GRPROLE(Recovery) GRPADDR(9.20.135.130) GRPVER(09040300)
CONNGRP(yes) GRSTATUS(Synchronizing) RCOVLSN(<0:0:19:16723>)
RCOVTIME(2023-12-20T17:36:23.463209Z) BACKLOG(5619) INSYNC(no) SYNCTIME()
ALTDATE(2023-12-20) ALTTIME(17.13.27)
```

関連タスク

112 ページの『例: IBM MQ Operator を使用したネイティブ HA の構成』
この例では、 IBM MQ Operator を使用して、ネイティブ高可用性フィーチャーを使用するキュー・マネージャーを OpenShift Container Platform にデプロイします。 相互 TLS は、認証に使用され、TLS 証明書からキュー・マネージャー内の ID にマップされます。

関連資料

dspmq (キュー・マネージャーの表示) コマンド

OpenShift MQ Adv. ネイティブ *HA* の詳細チューニング

タイミングと間隔のチューニングに関する詳細設定。 デフォルトがシステムの要件に適合しないことが判明している場合を除いて、これらの設定を使用する必要はないはずです。

ネイティブ HA を構成するための基本的なオプションは、QueueManager API を使用して処理されます。 IBM MQ Operator はこの API を使用して、基礎となるキュー・マネージャー INI ファイルを自動的に構成します。 NativeHALocal インスタンス・スタンザの下には、INI ファイルを使用してのみ構成可能な拡張オプションがいくつかあります。 INI ファイルの構成方法について詳しくは、101 ページの『例: MQSC ファイルと INI ファイルの提供』 も参照してください。

HeartbeatInterval

ハートビート間隔は、ネイティブ HA キュー・マネージャーのアクティブ・インスタンスがネットワーク・ハートビートを送信する頻度をミリ秒単位で定義します。 ハートビート間隔値の有効範囲は 500 (0.5 秒) から 60000 (1 分) で、この範囲外の値を使用するとキュー・マネージャーの開始が失敗しま

す。 この属性を省略すると、デフォルト値の 5000 (5 秒) が使用されます。 各インスタンスで同じハートビート間隔を使用する必要があります。

HeartbeatTimeout
ハートビート・タイムアウトは、ネイティブ HA キュー・マネージャーのレプリカ・インスタンスが、アクティブ・インスタンスが応答しないと判断するまで待機する時間の長さを定義します。 ハートビート間隔タイムアウト値の有効範囲は 500 (0.5 秒) から 120000 (2 分) です。 ハートビート・タイムアウトの値は、ハートビート間隔以上でなければなりません。

無効な値を指定すると、キュー・マネージャーの開始が失敗します。 この属性が省略されると、レプリカは、新しいアクティブ・インスタンスを選択するプロセスを開始する前に 2 x `HeartbeatInterval` 待機します。 各インスタンスで同じハートビート・タイムアウトを使用する必要があります。

RetryInterval
再試行間隔は、障害が発生した複製リンクがネイティブ HA キュー・マネージャーで再試行される頻度をミリ秒単位で定義します。 再試行間隔の有効範囲は 500 (0.5 秒) から 120000 (2 分) です。 この属性が省略されると、レプリカは、障害が発生した複製リンクを再試行する前に 2 x `HeartbeatInterval` 待機します。

OpenShift V 9.4.2 MQ Adv. *Native HA CRR の高度なチューニング*

タイミングと間隔のチューニングに関する詳細設定。 デフォルトがシステムの要件に適合しないことが判明している場合を除いて、これらの設定を使用する必要はないはずです。

Native HA CRR の基本的な設定オプションは、 `QueueManager` API を使用して処理されます。 IBM MQ Operator は、この API を使用して、基盤となるキューマネージャーの INI ファイルを設定します。 qm.ini ファイルの NativeHARecoveryGroup 節にある INI ファイルを使用してのみ設定可能な、より高度なオプションがいくつかあります。 INI ファイルの設定方法の詳細については、101 ページの『例: MQSC ファイルと INI ファイルの提供』 もご覧ください。

ShortRetryTimer
リカバリグループへの接続を試みる間の待機間隔を、短時間再試行フェーズで指定します。 ミリ秒単位で指定し、デフォルトは 2500（ 2.5 秒）です。

ShortRetryCount
短い再試行フェーズ中にリカバリグループへの接続を試みる回数を指定します。

LongRetryTimer
長時間再試行フェーズでリカバリグループへの接続を試みる間の待機間隔を指定します。 接続が確立されるまで、または構成が変更されて `ReplicationAddress` がコメントアウトまたは削除されるまで、長い再試行フェーズは無期限に続きます。 ミリ秒単位で指定し、デフォルトは 60000（60 秒）です。

RetryReporting
`Phase` または `All` に設定して、各再試行の試行が報告されるか、フェーズ（長または短）のみが報告されるかを指定します。 指定がない場合は、 `Phase` がデフォルトとなります。

StatusInterval
リカバリグループがリカバリログステータスを送信する前に待機する最長時間を指定します。 ミリ秒単位で指定し、デフォルトは 10000（10 秒）です。

StatusCommitThreshold
リカバリログステータスを送信する前に、リカバリグループがコミットする最大バイト数を指定します。 バイト単位で指定され、デフォルトは 0 です。これは、 `StatusInterval` のみが、リカバリログの状態が送信されるタイミングを決定するために使用されることを意味します。

OpenShift MQ Adv. ネイティブ *HA* キュー・マネージャーの終了

コンテナ内の IBM MQ では、**endmqm** コマンドを使用して、Native HA グループの一部であるアクティブまたはレプリカのキュー・マネージャを終了できます。

始める前に

注 : この情報はコンテナ環境のみに適用される。

このタスクについて

Native HA グループの一部であるキュー・マネージャを停止する手順は、アクティブ・インスタンスかレプリカ・インスタンスかによって異なります。 どちらのタイプのインスタンスを終了する場合も、インスタンスを終了してもネイティブ HA グループのクォーラムが壊れないようにチェックが行われます。 もしクォーラムが壊れてしまうなら、**endmqm** コマンドは失敗する。

endmqm コマンドを発行すると、グループ内の他のインスタンスは、接続が切れたときにエラーを報告しないように、これが起こっていることを警告されます。

あまりにも多くのレプリカ・インスタンスが終了したり切断されたりしてアクティブ・インスタンスがクォーラムを失った場合、アクティブ・インスタンスは完全に終了する前に設定可能な時間だけ待機します。これにより、アプリケーションの接続が切断されるだけでなく、処理を優雅にシャットダウンする期間ができる。 このタイムアウト値は、`qm.ini` ファイルの `NativeHALocalInstance` スタンザ内の `QuorumConnectivityTimeout` 属性で指定できます。 デフォルト値は 0 秒です。

手順

- キュー・マネージャのアクティブ・インスタンスを終了するには、アクティブ・インスタンスが実行されているノードで以下のコマンドを実行します：

  ```
  endmqm -s QMgrName
  ```

  - `-r` オプションを指定して、クライアント・アプリケーションが別のインスタンスに再接続できるようにします。
  - このインスタンスが Native HA グループのアクティブ・インスタンスでない場合、コマンドは失敗します。
  - このアクティブ・インスタンスを終了してグループ・クォーラムが失敗した場合、コマンドは失敗します。 (このコマンドの実行と同時に他のインスタンスが終了するか、または使用できなくなった場合、クォーラム・チェックでこれが検出されない可能性があります。その場合、Native HA グループは終了し、十分な数のインスタンスが使用可能な場合にのみ再起動できます。)

  アクティブなキュー・マネージャが終了すると、レプリカ・インスタンスの 1 つがアクティブな役割を引き継ぎます。 どのレプリカが引き継ぐかはグループ内の交渉によって決定され、どちらが最新のトランザクションログを持っているかに依存します。
- キュー・マネージャのレプリカ・インスタンスを終了するには、以下のコマンドを実行する：

  ```
  endmqm -x QMgrName
  ```

  - このインスタンスがアクティブ・インスタンスである場合、コマンドは失敗する。
  - このレプリカ・インスタンスを終了してグループ・クォーラムが失敗した場合、コマンドは失敗します。 (このコマンドの実行と同時に他のインスタンスが終了するか、または使用できなくなった場合、クォーラム・チェックでこれが検出されない可能性があります。その場合、Native HA グループは終了し、十分な数のインスタンスが使用可能な場合にのみ再起動できます。)

  注 : また、-c、-i、-p、または-w スイッチは、Native HA インスタンスがどのロールに属しているかに関係なく、**endmqm** コマンドで使用できます。 キューマネージャインスタンスは，グループクォーラムへの影響を無視して終了します。 しかし,情報はグループ内の他のインスタンスと共有される。 これらのスイッチは、アクティブ・インスタンスの-s と一緒に使うことができる。 これらのスイッチを-x スイッチと一緒にレプリカ・インスタンスに使用することはできません。

IBM MQ Operator を使用した複数インスタンス・キュー・マネージャーの構成

この例では、 IBM MQ Operator を使用して、複数インスタンス・キュー・マネージャーを OpenShift Container Platform にデプロイします。 相互 TLS は、認証に使用され、TLS 証明書からキュー・マネージャー内の ID にマップされます。

始める前に

この例を完了するには、まず以下の前提条件を満たしておく必要があります。

- この例のための OpenShift Container Platform (OCP) プロジェクト / 名前空間を作成します。
- コマンド・ラインで OCP クラスターにログインし、上記の名前空間に切り替えます。
- 上記の名前空間に IBM MQ Operator がインストールされ、使用可能な状態になっていることを確認します。

このタスクについて

この例では、OpenShift Container Platform にデプロイするキュー・マネージャーを定義したカスタム・リソース YAML を提供しています。 また、TLS を有効にしてキュー・マネージャーをデプロイするために必要な追加のステップも詳しく説明しています。

手順

1. 適切なストレージ・クラスの判別

   Kubernetes クラスター内のストレージには、複数の 永続ボリューム・アクセス・モードを使用してアクセスできます。 複数インスタンス・キュー・マネージャーは、複数の永続ボリューム (キュー・マネージャーごとに 1 つ、少なくとも 1 つの共有ボリューム) を作成します。 複数インスタンス・キュー・マネージャーの共有ボリュームでは、 `ReadWriteMany` ストレージ・クラスを使用する必要があります。 Kubernetes クラスターのデフォルトのストレージ・クラスは、通常、 `ReadWriteOnce` ストレージ・クラス (ブロック・ストレージ) 用です。 例えば、 Red Hat OpenShift Data Foundation を使用している場合、ストレージ・クラス *ocs-storagecluster-cephfs* は適切な共有ファイル・システムを提供します。 すべての共有ファイル・システムが同じ方法でファイル・ロックを処理するわけではないため、ファイル・システムの選択は非常に重要です。 Multiplatforms でのファイル・システム・サポートの計画および IBM MQ 複数インスタンス・キュー・マネージャー・ファイル・システムの テスト・ステートメントを参照してください。
2. 104 ページの『OpenSSL を使用した自己署名 PKI の作成』の説明に従って、証明書のペアを作成します。
3. MQSC コマンドと INI ファイルを含む構成マップを作成します。

   MQSC コマンドを含む Kubernetes ConfigMap を作成して、新しいキューと SVRCONN チャネルを作成し、チャネルへのアクセスを許可するチャネル認証レコードを追加します。

   前に作成した名前空間 (始める前にを参照) にいることを確認してから、OCP Web コンソールで、またはコマンド・ラインを使用して、以下の YAML を入力します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-miqm-configmap
data:
  example-tls.mqsc: |
    DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(BROWSE,PUT,GET,INQ)
  example-tls.ini: |
    Service:
```

```
        Name=AuthorizationService
        EntryPoints=14
        SecurityPolicy=UserExternal
```

MQSC は、*MTLS.SVRCONN* というチャネルと、*EXAMPLE.QUEUE*。 チャネルは、「共通名」が *example-app1* の証明書を提示するクライアントにのみアクセスを許可するように構成されています。 これは、ステップ 146 ページの『2』で作成した証明書のいずれかで使用される共通名です。 この共通名を持つこのチャネル上の接続は、 *app1* というユーザー ID にマップされます。このユーザー ID は、キュー・マネージャーに接続し、サンプル・キューにアクセスすることを許可されています。 INI ファイルはセキュリティー・ポリシーを使用可能にします。これは、 *app1* ユーザー ID が外部ユーザー・レジストリーに存在する必要がないことを意味します。この ID は、この構成に名前としてのみ存在します。

4. キュー・マネージャーのデプロイ

   以下のカスタム・リソース YAML を使用して、新しいキュー・マネージャーを作成します。 このタスクを開始する前に作成したネームスペースにいることを確認してから、OCP ウェブ・コンソールで、またはコマンド・ラインを使って、以下の YAML を入力します。 正しいライセンスが指定されていることを確認し、`false` を `true` に変更してライセンスを受け入れます。

   注 : この例では、 IBM MQ Advanced ライセンスを使用していますが、 IBM MQ ライセンスを使用することもできます。 詳しくは、161 ページの『を使用して、IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: exampleqm
spec:
  license:
    accept: false
    license: L-NUUP-23NH8Y
    use: Production
  queueManager:
    name: EXAMPLEQM
    availability:
      type: MultiInstance
    mqsc:
    - configMap:
        name: example-miqm-configmap
        items:
        - example-tls.mqsc
    ini:
    - configMap:
        name: example-miqm-configmap
        items:
        - example-tls.ini
    storage:
      defaultClass: STORAGE_CLASS
  version: 9.4.3.0-r1
  pki:
    keys:
      - name: default
        secret:
          secretName: example-qm-tls
          items:
            - tls.key
            - tls.crt
            - ca.crt
```

STORAGE_CLASS を、ステップ 146 ページの『1』で識別したストレージ・クラスに変更します。

シークレット *example-qm-tls* がステップ 146 ページの『2』で作成され、 ConfigMap *example-miqm-configmap* がステップ 146 ページの『3』 で作成されたことに注意してください。

可用性タイプは *MultiInstance* に設定されており、自動的に永続ストレージが選択される。

5. キュー・マネージャーが稼働していることの確認

   キュー・マネージャーがデプロイされます。 続行する前に、Running 状態であることを確認してください。 以下に例を示します。

```
oc get qmgr exampleqm
```

6. キュー・マネージャーへの接続のテスト

キュー・マネージャーが構成済みで使用可能であることを確認するには、108 ページの『ラップトップからキュー・マネージャーへの相互 TLS 接続のテスト』の手順に従います。

7. アクティブ・ポッドの障害の強制試行

   キュー・マネージャーの自動復旧を検証するには、ポッドの障害をシミュレートします。

   a) アクティブ・ポッドとスタンバイ・ポッドの表示

      以下のコマンドを実行します。

      ```
      oc get pods --selector app.kubernetes.io/instance=exampleqm
      ```

      READY フィールドでは、アクティブ・ポッドが値 1/1 を返し、スタンバイ・ポッドが値 0/1 を返すことに注意してください。

   b) アクティブ・ポッドの削除

      アクティブ・ポッドの絶対パス名を指定して次コマンドを実行します。

      ```
      oc delete pod exampleqm-ibm-mq-value
      ```

   c) ポッドの状況の再表示

      以下のコマンドを実行します。

      ```
      oc get pods --selector app.kubernetes.io/instance=exampleqm
      ```

   d) キュー・マネージャーの状況の表示

      他のポッドのフルネームを指定して、以下のコマンドを実行します。

      ```
      oc exec -t Pod -- dspmq -x
      ```

      アクティブ・インスタンスが変更されたことを示す状況が表示されるはずです。以下に例を示します。

      ```
      QMNAME(EXAMPLEQM)                                         STATUS(Running as standby)
          INSTANCE(exampleqm-ibm-mq-1) MODE(Active)
          INSTANCE(exampleqm-ibm-mq-0) MODE(Standby)
      ```

   e) キュー・マネージャーへの接続を再度テストします。

      キュー・マネージャーが回復したことを確認するには、108 ページの『ラップトップからキュー・マネージャーへの相互 TLS 接続のテスト』の手順に従います。

タスクの結果

これで、相互 TLS 認証を使用して複数インスタンス・キュー・マネージャーが正常にデプロイされ、アクティブなポッドで障害が発生したときに自動的に復旧することが確認されました。

OpenShift CP4I CD Red Hat OpenShift クラスターの外部からキュー・マネージャーに接続するためのルートの構成

Red Hat OpenShift クラスターの外部から IBM MQ キュー・マネージャーにアプリケーションを接続するには、Red Hat OpenShift 経路が必要です。 IBM MQ キュー・マネージャーおよびクライアント・アプリケーションで TLS を有効にする必要があります。SNI は、TLS 1.2 以上のプロトコルが使用されている場合にのみ TLS プロトコルで使用できるためです。 Red Hat OpenShift Container Platform Router では、IBM MQ キュー・マネージャーへの要求のルーティングに SNI が使用されます。

このタスクについて

Red Hat OpenShift Route の必要な構成は、クライアント・アプリケーションの Server Name Indication (SNI) の動作によって異なります。 IBM MQ では、構成とクライアントのタイプに応じて 2 種類の SNI ヘッダー設定がサポートされています。 SNI ヘッダーは、クライアントの宛先のホスト名に設定されるか、または IBM MQ チャネル名に設定されます。 IBM MQ でチャネル名がどのようにホスト名にマップされるかについては、IBM MQ で複数の証明書の機能を提供する方法を参照してください。

SNI ヘッダーを IBM MQ チャネル名に設定するか、ホスト名に設定するかは、 **OutboundSNI** 属性を使用して制御します。 可能な値は、 OutboundSNI=CHANNEL (デフォルト値) または OutboundSNI=HOSTNAME です。 詳しくは、クライアント構成ファイルの SSL スタンザを参照してください。 CHANNEL および HOSTNAME は、使用する正確な値です。これらは、実際のチャネル名またはホスト名に置き換える変数名ではありません。

OutboundSNI 設定が異なるクライアントの動作

OutboundSNI が HOSTNAME に設定されていて、接続名でホスト名が指定されていると、以下のクライアントではホスト名の SNI が設定されます。

- C クライアント
- 非管理対象モードの .NET クライアント
- Java/JMS クライアント

OutboundSNI が HOSTNAME に設定されていて、接続名で IP アドレスが使用されていると、以下のクライアントではブランクの SNI ヘッダーが送信されます。

- C クライアント
- 非管理対象モードの .NET クライアント
- Java/JMS クライアント (ホスト名の逆引き DNS ルックアップを実行できない)

OutboundSNI が CHANNEL に設定されている場合や、何も設定されていない場合は、ホスト名と IP アドレスのどちらの接続名が使用されていても、IBM MQ チャネル名が代わりに使用されて常に送信されます。

以下のクライアント・タイプでは、SNI ヘッダーを IBM MQ チャネル名に設定できないので、 **OutboundSNI** の設定に関係なく常に SNI ヘッダーをホスト名に設定しようとします。

- AMQP クライアント
- XR クライアント

IBM MQ 管理対象 .NET クライアントは、 **OutboundSNI** プロパティーが HOSTNAME に設定されている場合、SERVERNAME をそれぞれのホスト名に設定します。これにより、IBM MQ 管理対象 .NET クライアントは、 Red Hat OpenShift 経路を使用してキュー・マネージャーに接続できます。

クライアント・アプリケーションが IBM MQ Internet Pass-Thru (MQIPT) を介して Red Hat OpenShift クラスターにデプロイされたキュー・マネージャーに接続する場合、 MQIPT は、ルート定義内の SSLClientOutboundSNI プロパティーを使用して、 SNI をホスト名に設定するように構成することができます。

OutboundSNI、複数の証明書、および Red Hat OpenShift 経路

IBM MQ は、SNI ヘッダーを使用して複数の証明書機能を提供します。 アプリケーションが、CERTLABL フィールドを介して別の証明書を使用するように構成されている IBM MQ チャネルに接続する場合、アプリケーションは CHANNEL の **OutboundSNI** 設定を使用して接続する必要があります。

Red Hat OpenShift 経路構成に HOSTNAME SNI が必要な場合は、 IBM MQ の複数の証明書機能を使用できず、 IBM MQ チャネル・オブジェクトに CERTLABL 設定を設定できません。

OutboundSNI に CHANNEL 以外の設定を持つアプリケーションが、証明書ラベルが構成されたチャネルに接続すると、そのアプリケーションは MQRC_SSL_INITIALIZATION_ERROR で拒否され、キュー・マネージャーのエラー・ログに AMQ9673 メッセージが出力されます。

IBM MQ が複数の証明書機能を提供する方法について詳しくは、IBM MQ が複数の証明書機能を提供する方法 を参照してください。

例

SNI を MQ チャネルに設定するクライアント・アプリケーションには、接続先のチャネルごとに新しい Red Hat OpenShift ルートが作成されている必要があります。 また、適切なキュー・マネージャーにルーティン

グできるようにするには、Red Hat OpenShift Container Platform クラスターで一意のチャネル名を使用する必要があります。

IBM MQ がチャネル名を SNI ヘッダーにマップする方法により、 MQ チャネル名の末尾が小文字にならないことが重要です。

それぞれの新規 Red Hat OpenShift ルートに必要なホスト名を判別するには、各チャネル名を SNI アドレスにマップする必要があります。 詳しくは、IBM MQ で複数の証明書の機能を提供する方法を参照してください。

次に、クラスターに以下の yaml を適用して、チャネルごとに新しい Red Hat OpenShift 経路を作成する必要があります。

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: unique_name_for_the_route
  namespace: namespace_of_your_MQ_deployment
spec:
  host: SNI_address_mapping_for_the_channel
  to:
    kind: Service
    name: name_of_Kubernetes_Service_for_your_MQ_deployment (for example "queue_manager_name-
ibm-mq")
  port:
    targetPort: 1414
  tls:
    termination: passthrough
```

クライアント・アプリケーション接続の詳細の構成

以下のコマンドを実行すると、クライアント接続で使用するホスト名を判別できます。

```
oc get route Name of hostname based Route (for example "queue_manager_name-ibm-mq-qm")>
-n namespace of your MQ deployment -o jsonpath="{.spec.host}"
```

クライアント接続用のポートは、Red Hat OpenShift Container Platform ルーターが使用するポート (通常は 443) に設定する必要があります。

関連タスク

218 ページの『IBM MQ Console クラスターにデプロイされた Red Hat OpenShift への接続』
Red Hat OpenShift Container Platform クラスターにデプロイされているキュー・マネージャーの IBM MQ Console に接続する方法について説明します。

コンテナ内の IBM MQ に対する監査ロギングの設定

IBM MQ コンテナでは、さまざまなイベントの監査ロギングのためのさまざまな機能が提供される。

あなたの環境の監査ロギングの要件に応じて、あなたは、イベントログを収集し、保持するための適切なメカニズムだけでなく、これらの機能の 1 つ以上を構成する必要があるかもしれません。 以下のオプションから選択します。

IBM MQ Operator Validating Webhook

バージョン 3.6.0 以降、IBM MQ Operator はデフォルトで、公開されたエンドポイントへのアクセス監査イベントをポッドのコンテナ・ログに出力します。 コンフィギュレーションは必要ない。

IBM MQ イベント

IBM MQ 、監査用のイベント・メッセージを生成するように設定できる。 詳細は監査を参照。

V 9.4.3 **MQ コンテナメトリクス**

IBM MQ コンテナは、 Prometheus メトリクス・エンドポイントを公開します（詳細は 218 ページの『IBM MQ Operator 使用時のモニター』 を参照)。 IBM MQ 9.4.3 から、環境変数 `MQ_LOGGING_METRICS_AUDIT_ENABLED` を `true` に設定することで、このエンドポイントへのアク

セスのロギングを有効にできます。 IBM MQ Operator を使用している場合、この変数は queueManager コンフィギュレーションの env セクションで設定できる：

```
[...]
spec:
  queueManager:
    env:
    - name: MQ_LOGGING_METRICS_AUDIT_ENABLED
      value: "true"
[...]
```

ロギングを有効にすると、 /var/mqm/errors ディレクトリにログファイルが生成されます。 外部ストレージを /mnt/mqm にマウントすることで、このディレクトリにあるデータを永続化することができる（ コンテナ・イメージのリファレンスを参照）。 IBM MQ Operator を使用している場合は、代わりに 267 ページの『.spec.queueManager.storage.persistedData』 フィールドを使用して外部ストレージをマッピングすることができる。 メトリクスアクセスログファイルは JSON 形式で記述される。 ファイル名は metricaudit<nn>.json、nn は数字である。 metricaudit01.json ファイルには最新のイベントが含まれている。

組み込み IBM WebSphere Application Server Liberty ウェブサーバー

組み込みの WebSphere Liberty ウェブサーバーを設定して、監査イベントを生成することができます。 audit-1.0 と audit-2.0 機能がインストールされており、要件に応じて設定可能です。 詳しくは、WebSphere Liberty ドキュメントの Audit 1.0 および Audit 2.0 を参照。

コンフィギュレーション・ファイルを使って、この機能を有効にし、設定することができます。 IBM MQ Operator を使用している場合は、 275 ページの『.spec.web.manualConfig』 マウントを使用してファイルを読み込むことができます。 例えば、以下のコードは Audit 2.0 機能のデフォルト設定を有効にします。

audit-2.0 機能を有効にするために、 mqwebuser.xml という設定ファイルを作成する：

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: example-enable-audit
data:
  mqwebuser.xml: |
    <?xml version="1.0" encoding="UTF-8"?>
    <server>
      <featureManager>
        <feature>audit-2.0</feature>
      </featureManager>
    </server>
```

spec.web.manualConfig マウントを使用して、 queueManager のコンフィギュレーションに前のコードを追加する：

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
[...]
spec:
  [...]
  web:
    enabled: true
    manualConfig:
      configMap:
        name: example-enable-audit
```

注 : デフォルトでは、監査ログは基本（プレーンテキスト）形式でコンテナログにミラーリングされる。 このフォーマットには、縮小された監査情報が含まれている。 より多くの情報を得るには、 IBM MQ コンテナ・ログを JSON 形式に変更するか、 web ログのミラーリングを無効にして、 WebSphere Liberty JSON 監査ログ・ファイルを直接収集する。 MQ_LOGGING_CONSOLE_FORMAT と MQ_LOGGING_CONSOLE_SOURCE 環境変数を使用することで、ログフォーマットとログミラーリングソースを制御することができます。 有効な値については、 287 ページの『サポートされる環境変数』 を参照のこと。

OpenShift CP4I IBM MQ と IBM Instana トレースの統合

IBM Instana を使用して、 IBM Cloud Pak for Integration 内のトランザクションをトレースできます。

始める前に

本書では、 IBM Instana トレースについて説明します。これは、システムを介してメッセージをトレースするプロセスです。 このトピックでは、 IBM MQ キュー・マネージャーの状態に関する詳細を取得する IBM Instana モニターについては説明しません。 IBM Instana による IBM MQ のモニターについては、 Monitoring IBM MQ を参照してください。 認証済みモニターについて詳しくは、 153 ページの『TLS を使用した認証済み IBM Instana モニターの構成』を参照してください。

注 :

- この機能は、 IBM MQ バージョン 9.3.1.0-r2 以降のオペランドでのみサポートされています。
- IBM MQ Operator とキュー・マネージャの旧バージョンでは、 IBM Instana トレースを実行できますが、ネイティブでは実行できません。 IBM Instana ドキュメントの「 Configuring IBM MQ Tracing 」を参照。
- この機能は、 IPv6 ネットワーキングを使用している場合はサポートされません。

IBM MQ Operator を使用して IBM Instana トレースを実行する前に、 IBM Instana バックエンド・エージェントと IBM Instana エージェントの両方をデプロイする必要があります。 デフォルトでは、 IBM MQ キュー・マネージャーは、キュー・マネージャー・ポッドと同じノードにデプロイされた IBM Instana エージェントと通信します。

このタスクについて

IBM Instana との統合を有効にすると、 IBM MQ API 出口がキュー・マネージャーにインストールされます。 API 出口は、キュー・マネージャーを流れるメッセージに関するトレース・データを IBM Instana エージェントに送信します。

API 出口は、各メッセージに RFH2 ヘッダーを追加します。 これらのヘッダーにはトレース情報が含まれます。

IBM Instana エージェントは、 IBM Instana バックエンドへのトレース・データの送信を担当します。

IBM Instana バックエンドおよび IBM Instana エージェントのデプロイについては、 IBM Instana 資料の「 プラットフォーム UI での Instana モニター・リンクの有効化 」を参照してください。

手順

標準デプロイメント

- IBM Instana トレースを有効にしてキュー・マネージャーをデプロイします。

  デフォルトでは、 IBM Instana トレースは無効になっています。

  IBM Cloud Pak for Integration Platform UI または OpenShift Web コンソールを使用している場合:

  1. **演算子 > インストールされている演算子 > IBM MQ > 詳細 > インスタンスの作成**をクリックします。 キュー・マネージャーに必要な情報をすべて入力してください。
  2. 同じ場所で、**Telemetry > Tracing > Instana** をクリックします。
  3. 「**Instana トレースを有効にする (Enable Instana tracing)**」 トグルを `true` に設定します。

  YAML を使用してデプロイする場合は、以下のスニペットを使用します。

```
spec:
  telemetry:
    tracing:
      instana:
        enabled: true
```

拡張デプロイメント

- https 経由で IBM Instana エージェントと通信します。

デフォルトでは、 IBM MQ の IBM Instana 出口は HTTP を介して IBM Instana エージェントと通信します。 エージェントのホスト・アドレスは、キュー・マネージャーが実行されているノードの IP アドレスに設定されます。 これは、 IBM Instana 資料の「IBM Instana モニターの使用可能化」で説明されている構成と一致します。ここで、 IBM Instana エージェントは、 IBM Instana Agent Operator によってデーモン・セットとしてデプロイされます。

現在、 IBM MQ の IBM Instana 出口と IBM Instana エージェントの間の通信では、http プロトコルまたは https プロトコルがサポートされています。 https を使用するには、まず TLS 暗号化を使用するように IBM Instana エージェントを構成する必要があります。 IBM Instana 資料の エージェント・エンドポイントの TLS 暗号化のセットアップ を参照してください。 その後、以下のようにプロトコルを https に設定できます。

OpenShift Web コンソールを使用している場合:

1. **演算子 > インストールされている演算子 > IBM MQ > 詳細 > インスタンスの作成**をクリックします。 キュー・マネージャーに必要な情報をすべて入力してください。
2. 同じ場所で、**Telemetry** > **Instana** をクリックします。
3. 「**拡張構成**」 ドロップダウン・リストを展開します。
4. 「**Instana エージェント通信プロトコル**」 を `https` に設定します。

YAML を使用してデプロイする場合は、以下のスニペットを使用します。

```
spec:
  telemetry:
    instana:
      enabled: true
      protocol: https
```

- **agentHost** を設定します。

  IBM Instana エージェントが、キュー・マネージャーが実行されている OpenShift クラスターにデーモンセットとしてデプロイされていない場合は、 **agentHost** 値を、 IBM Instana エージェントが実行されているホスト名または IP アドレスに設定する必要があります。 **agentHost** 値にプロトコルまたはポートを含めることはできません。

  OpenShift Web コンソールを使用している場合:

  1. **演算子 > インストールされている演算子 > IBM MQ > 詳細 > インスタンスの作成**をクリックします。 キュー・マネージャーに必要な情報をすべて入力してください。
  2. 同じ場所で、**Telemetry** > **Instana** をクリックします。
  3. 「**拡張構成**」 ドロップダウン・リストを展開します。
  4. 「**Instana エージェント・ホスト (Instana agent host)**」 テキスト・ボックスにホスト名を入力します。

  YAML を使用してデプロイする場合は、以下のスニペットを使用します。

```
spec:
  telemetry:
    instana:
      enabled: true
      agentHost: 9.9.9.9
```

次のタスク

98 ページの『IBM MQ Operator を使用した単純なキュー・マネージャーのデプロイ』も参照してください。

OpenShift CP4I Operator2.2.0 TLS を使用した認証済み IBM Instana モニターの構成

IBM Instana エージェントを介してキュー・マネージャーをモニターできるようにするには、エージェントとキュー・マネージャーの両方を構成する必要があります。

始める前に

IBM Instana 資料の 「IBM MQ のモニター」の「構成」セクション には、 IBM Instana モニター構成に関する一般情報が記載されています。 ただし、キュー・マネージャーの構成に関する詳細は含まれていません。

IBM MQ Operator で IBM Instana トレースを実行する前に、 IBM Instana バックエンドと IBM Instana エージェントの両方をデプロイする必要があります。 これを行うには、 IBM Instana 資料の CP4I Platform UI での IBM Instana モニタリングの有効化 を参照してください。

手順

1. 証明書を生成します。
2. IBM Instana エージェントを構成します。
3. キュー・マネージャーを構成します。
4. 検証およびデバッグ。

関連タスク

152 ページの『IBM MQ と IBM Instana トレースの統合』
IBM Instana を使用して、 IBM Cloud Pak for Integration 内のトランザクションをトレースできます。

OpenShift CP4I Operator2.2.0 *IBM Instana* エージェントおよびキュー・マネージャーの証明書と鍵を生成します。

IBM Instana エージェントとキュー・マネージャーの間の TLS 通信の場合、両方に証明書とそれに対応する秘密鍵が必要です。

始める前に

これは、 TLS を使用して認証済み IBM Instana モニターを構成する 4 つのタスクのうちの最初のタスクです。

注 : これらの証明書の生成に使用される値は、デモンストレーション用です。 実稼働環境にデプロイする場合は、証明書のサブジェクトと有効期限が適切であることを確認してください。

手順

IBM MQ キュー・マネージャー

TLS を介して IBM Instana エージェントと通信するには、キュー・マネージャーに証明書とそれに対応する秘密鍵が必要です。 これらが既にある場合は、このセクションをスキップしてください。

1. キュー・マネージャーの証明書と秘密鍵を生成します。

   以下のコマンドを実行します。

```
openssl req \
       -newkey rsa:2048 -nodes -keyout server.key \
       -subj "/CN=mq queuemanager/OU=ibm mq" \
       -x509 -days 3650 -out server.crt
```

IBM Instana エージェント

エージェントが IBM MQ キュー・マネージャーとの TLS 通信を実行するには、エージェントに証明書とそれに対応する秘密鍵が必要です。 使用する JKS 鍵ストア内に秘密鍵と証明書が既にある場合は、このセクションをスキップしてください。

2. IBM Instana エージェントの証明書と秘密鍵を生成します。

   以下のコマンドを実行します。

```
openssl req \
       -newkey rsa:2048 -nodes -keyout application.key \
       -subj "/CN=instana-agent/OU=app team1" \
       -x509 -days 3650 -out application.crt
```

3. 証明書と秘密鍵を PKCS12 鍵ストアに保管します。

   以下のコマンドを実行します。 *your_password* は、鍵ストアを保護するために使用するパスワードに置き換えてください。 この置換は、後続のすべてのステップで実行します。

   ```
   openssl pkcs12 -export -out application.p12 -inkey application.key -in application.crt
   -passout pass:your_password
   ```

4. PKCS12 鍵ストアを JKS 鍵ストアに変換します。

   以下のコマンドを実行します。

   ```
   keytool -importkeystore \
           -srckeystore application.p12 \
           -srcstoretype pkcs12 \
           -destkeystore application.jks \
           -deststoretype JKS \
           -srcstorepass your_password \
           -deststorepass your_password \
           -noprompt
   ```

5. 証明書にラベルを付けます。

   以下のコマンドを実行します。

   ```
   keytool -changealias -alias "1" -destalias "instana" -keypass your_password -keystore
   application.jks -storepass your_password -noprompt
   ```

6. キュー・マネージャー証明書を鍵ストアにインポートします。

   以下のコマンドを実行します。

   ```
   keytool -importcert -file server.crt -keystore application.jks -storepass your_password
   -alias myca -noprompt
   ```

次のタスク

これで、 IBM Instana モニター用にエージェントを構成する準備ができました。

OpenShift CP4I Operator2.2.0

Instana モニター: エージェントの構成

鍵ストアを IBM Instana エージェントにマウントしてから、特定のキュー・マネージャーのモニターを構成します。

始める前に

このタスクは、 IBM Instana エージェントおよびキュー・マネージャーの証明書と鍵を生成していることを前提としています。

手順

IBM Instana エージェントへの鍵ストアのマウント

1. IBM Instana エージェント名前空間内の JKS 鍵ストアから秘密を作成します。

   以下のコマンドを実行します。 *keystore_secret_name* は、使用する名前に置き換えてください。 この置換は、後続のすべてのステップで実行します。

   ```
   oc create secret generic keystore_secret_name --from-file=./application.jks -n instana-agent
   ```

2. 秘密をボリュームとして Instana エージェントポッドにマウントします。

   エージェントの `CustomResource` または `values.yaml` Helm チャートでボリュームとボリュームマウントを指定することで、これを実行します

   ```
   agent:
     pod:
       volumeMounts:
       - mountPath: /opt/instana/agent/etc/jks-file-name.jks
   ```

```
          name: mq-key-jks-name
          subPath: jks-file-name.jks
        volumes:
        - name: mq-key-jks-name
          secret:
            secretName: keystore_secret_name
```

この編集を、Instana Agent Operator によってデプロイされた Instana エージェントに対して行うには、次の例のようなコマンドを実行します

```
oc edit agent [instana-agent-name] -n [instana-agent-namespace]
```

Instana エージェントポッドへのシークレットのマウントに関する詳細は、Instana エージェント Helm チャートドキュメントの「ボリューム」および「volumeMounts」を参照してください。

特定のキュー・マネージャーのモニターの構成

3. instana-agent 名前空間で、 `oc edit configmap instana-agent` コマンドを使用して instana-agent 構成マップを編集します。
4. `configuration.yaml: |`の下に以下のセクションを追加します。 このセクションを既に定義している場合は、新しいキュー・マネージャーをリストに追加するだけです。

```
    com.instana.plugin.ibmmq:
      enabled: true
      poll_rate: 60
      queueManagers:
         QUEUE_MANAGER_NAME:
            channel: 'INSTANA.A.SVRCONN'
            keystorePassword: 'your_password'
            keystore: '/opt/instana/agent/etc/application.jks'
            cipherSuite: 'TLS_RSA_WITH_AES_256_CBC_SHA256'
```

説明:

- *your_password* は JKS 鍵ストアのパスワードです。
- *QUEUE_MANAGER_NAME* は、キュー・マネージャー・オペランドの名前ではなく、デプロイする基礎となる IBM MQ キュー・マネージャーの名前です。

注: *QUEUE_MANAGER_NAME* が基礎となるキュー・マネージャー名に設定されておらず、代わりにオペランドに設定されている場合、モニターは機能しません。 基礎となる名前は、キュー・マネージャー・オペランドの `spec.queuemanager.name` で定義されます。

5. instana-agent 名前空間内の instana-agent ポッドを削除します。 これにより、それらのユーザーは再始動し、新しい設定でモニターを開始します。

次のタスク

これで、 IBM Instana モニター用にキュー・マネージャーを構成する準備ができました。

OpenShift CP4I Operator2.2.0 ***Instana* モニター: キュー・マネージャーの構成**

TLS を使用して IBM Instana エージェントと通信するキュー・マネージャーをセットアップします。この接続の認証は、 SSLPEERMAP を使用して行われます。

始める前に

このタスクは、 IBM Instana モニター用にエージェントを構成していることを前提としています。

手順

1. MQSC と INI の両方を使用してキュー・マネージャーを構成します。

   MQSC は、新しい TLS 対応チャネルをセットアップするために使用されます。その後、必須フィールドが指定された証明書がある場合に、接続先の IBM Instana エージェントを認証するようにそのチャネルを構成します。 この場合、 `CN=instana-agent,OU=app team1` フィールドを含む証明書を持つ接続

クライアントをユーザー app1 にマップします。 次に、MQSC は、 IBM Instana モニターに必要な操作を実行する権限をユーザー app1 に付与します。

INI ファイルは、外部ユーザー app1 に許可を付与するために使用されます。

以下の構成マップには、必須の MQSC 設定と INI 設定が含まれています。 キュー・マネージャーの名前空間にデプロイします。

```
apiVersion: v1
data:
  channel.mqsc: |-
    DEFINE CHANNEL('INSTANA.A.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS12_OR_HIGHER')
    ALTER QMGR CONNAUTH(' ')
    REFRESH SECURITY
    SET CHLAUTH('INSTANA.A.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
    SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS) ACTION(REPLACE)
    SET CHLAUTH('INSTANA.A.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=instana-agent,OU=app
team1') USERSRC(MAP) MCAUSER('app1')
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(ALL)
    SET AUTHREC PROFILE('SYSTEM.ADMIN.COMMAND.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(PUT,INQ,DSP,CHG)
    SET AUTHREC PROFILE('SYSTEM.**') PRINCIPAL('app1') OBJTYPE(TOPIC) AUTHADD(DSP)
    SET AUTHREC PROFILE('*') PRINCIPAL('app1') OBJTYPE(TOPIC) AUTHADD(DSP)
    SET AUTHREC PROFILE('SYSTEM.**') PRINCIPAL('app1') OBJTYPE(QUEUE) AUTHADD(DSP, CHG, GET)
    SET AUTHREC PROFILE('SYSTEM.**') PRINCIPAL('app1') OBJTYPE(LISTENER) AUTHADD(DSP)
    SET AUTHREC PROFILE('AMQ.*') PRINCIPAL('app1') OBJTYPE(QUEUE) AUTHADD(DSP, CHG)
    REFRESH SECURITY TYPE(CONNAUTH)
  auth.ini: |-
    Service:
      Name=AuthorizationService
      EntryPoints=14
      SecurityPolicy=UserExternal
kind: ConfigMap
metadata:
  namespace: your-queue-manager-namespace
  name: qmgr-monitoring-config
```

ここで、 *your-queue-manager-namespace* は、キュー・マネージャーがデプロイされる名前空間です。

注 : ユーザー定義キューをモニターする場合は、構成マップ MQSC に追加の行を追加して、DSP、CHG、および GET 許可をそれらのキューに付与する必要があります。 以下に例を示します。

```
SET AUTHREC PROFILE('MYQUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE) AUTHADD(DSP, CHG, GET).
```

この例では、MQSC データと INI データに configmap を使用しますが、追加した内容が機密情報である場合は、シークレットを使用できます。 MQSC および INI を使用したデプロイに関する一般情報については、 101 ページの『例: MQSC ファイルと INI ファイルの提供』を参照してください。

2. TLS 接続を行うには、キュー・マネージャーが IBM Instana エージェントの証明書を信頼する必要があります。 これを行うには、 IBM Instana エージェントの証明書のみを含むシークレットを作成します。

```
oc create secret generic instana-certificate-secret --from-file=./application.crt -n your-
queue-manager-namespace
```

3. キュー・マネージャーは、TLS ハンドシェーク用に独自の証明書を提示する必要があり、関連付けられた秘密鍵へのアクセスを必要とします。 以前に作成したか、既に所有している鍵と証明書を含む秘密をデプロイします。

```
oc create secret tls qm-tls-secret --cert server.crt --key server.key -n your-queue-manager-
namespace
```

構成マップとシークレットが作成されると、キュー・マネージャー自体を作成する準備が整います。

4. キュー・マネージャー YAML がキュー・マネージャー・コンテナーで環境変数 **MQSNOAUT** を設定していないことを確認します。

そうしないと、有効にした後に認証メカニズムが機能しなくなります。 デプロイメント後にこの変数を削除しても、メカニズムは再有効化されないため、キュー・マネージャーを再作成する必要があります。

5. キュー・マネージャー定義に以下のセクションを追加します。ここで、 *MYQM* はキュー・マネージャーの名前です。

```
spec:
  queueManager:
    name: MYQM   #(a)
    ini:   #(b)
      - configMap:
          items:
            - auth.ini
          name: qmgr-monitoring-config
    mqsc:   #(c)
      - configMap:
          items:
            - channel.mqsc
          name: qmgr-monitoring-config
  pki:
    keys:   #(d)
        - name: default
          secret:
            items:
            - tls.key
            - tls.crt
            secretName: qm-tls-secret
    trust:   #(e)
      - name: app
        secret:
          items:
            - application.crt
          secretName: instana-certificate-secret
```

この仕様のフラグが立てられたセクションについて、以下で説明します。

a. 基礎となるキュー・マネージャーに固有の名前を指定したことを確認してください。 基礎となるキュー・マネージャーに固有の名前がない場合は、モニターが意図したとおりに機能しない可能性があります。 この名前は、前に編集した IBM Instana エージェント構成マップ内の名前と一致している必要があります。

b. 構成マップに書き込まれた INI 情報がキュー・マネージャーに追加されます。

c. 構成マップに書き込まれた MQSC 情報がキュー・マネージャーに追加されます。

d. キュー・マネージャーの証明書と秘密鍵がキュー・マネージャーの鍵ストアに追加されます。

e. IBM Instana エージェント証明書がキュー・マネージャーのトラストストアに追加されます。

6. オプション: モニター対象キュー・マネージャーで IBM Instana トレースを有効にします。

これを行う場合は、 152 ページの『IBM MQ と IBM Instana トレースの統合』を参照してください。

7. キュー・マネージャーをデプロイします。

次のタスク

これで、 IBM Instana モニターの検証とデバッグを行う準備ができました。

OpenShift CP4I Operator2.2.0 *Instana* モニター: 検証およびデバッグ

IBM Instana エージェントを介してキュー・マネージャーをモニターできるようにするには、エージェントとキュー・マネージャーの両方を構成する必要があります。

始める前に

この作業は、 IBM Instana モニター用にキュー・マネージャーが構成されていることを前提としています。

手順

の検証

1. デプロイメントが正常に完了したことを確認するには、 IBM Instana ダッシュボードでキュー・マネージャーを表示します。

キュー・マネージャーは、アプリケーション・ページのサービス・セクションと「インフラストラクチャー」ビューにも表示されます。

デバッグ

注 : これらのデバッグ手順では、デーモンセットとして実行されている IBM Instana エージェントの Openshift デプロイメントを想定しています。

IBM Instana ダッシュボードにキュー・マネージャーが表示されない場合は、キュー・マネージャーの構成が誤っている可能性があります。 調査するには、以下の手順を使用します。

2. アクティブなキュー・マネージャー・ポッドが実行されているノードを識別します。

   キュー・マネージャーの名前空間で以下のコマンドを実行します。

   ```
   oc get pods -o wide -n your-queue-manager-namespace
   ```

3. キュー・マネージャーと同じノードで実行されている IBM Instana エージェント・ポッドを判別するには、instana-agent 名前空間で同じコマンドを実行します。

   ```
   oc get pods -o wide -n instana-agent-namespace
   ```

4. IBM Instana エージェント側の問題を理解しやすくするために、 IBM Instana エージェント・ポッドのログを取得し、「mq」またはキュー・マネージャーの名前に関連する項目を探します。

   以下のコマンドを実行します。

   ```
   oc logs instana-agent-pod -c instana-agent -n instana-agent
   ```

5. キュー・マネージャーのログを確認してください。

   エージェントがキュー・マネージャーへの接続を試行した場合、キュー・マネージャーのログに、接続が失敗した理由が示されているはずです。 以下のコマンドを実行します。

   ```
   oc logs your-queue-manager-name -n your-queue-manager-namespace
   ```

タスクの結果

これで、 TLS を使用した認証済み IBM Instana モニターの構成のための 4 つのタスクがすべて完了しました。

OpenShift CP4I Red Hat OpenShift CLI を使用した、カスタム MQSC および INI ファイルを使用したイメージの作成

Red Hat OpenShift Container Platform パイプラインを使用して、新規の IBM MQ コンテナー・イメージを作成できます。このイメージを使用するキュー・マネージャーに適用する MQSC ファイルと INI ファイルも指定できます。 このタスクは、プロジェクト管理担当者が実行する必要があります。

始める前に

Red Hat OpenShift Container Platform のコマンド・ライン・インターフェースをインストールする必要があります。

cloudctl login (IBM Cloud Pak for Integration の場合) または **oc login** を使用してクラスターにログインします。

Red Hat OpenShift プロジェクトに IBM Entitled Registry の Red Hat OpenShift シークレットがない場合は、 ライセンス・キー・シークレットの作成の手順に従います。

手順

1. ImageStream の作成

   イメージ・ストリームおよびそのストリームに関連付けられたタグによって、Red Hat OpenShift Container Platform 内からコンテナー・イメージを参照するための抽象化が可能になります。 イメー

ジ・ストリームおよびそのストリームのタグによって、使用可能なイメージを確認できます。また、必要とする特定のイメージがリポジトリー内で変更されたとしても、確実にそのイメージを使用することができます。

```
oc create imagestream mymq
```

2. 新規イメージ用に BuildConfig を作成

   BuildConfig は、新しいイメージのビルドを許可します。これは、IBM 公式イメージに基づきますが、コンテナーの始動時に実行される MQSC ファイルまたは INI ファイルが追加されます。

   a) BuildConfig リソースを定義する YAML ファイルを作成します

   例えば、以下を内容とする「mq-build-config.yaml」というファイルを作成します。

```
apiVersion: build.openshift.io/v1
kind: BuildConfig
metadata:
  name: mymq
spec:
  source:
    dockerfile: |-
      FROM cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.3.0-r1
      RUN printf "DEFINE QLOCAL(foo) REPLACE\n" > /etc/mqm/my.mqsc \
        && printf "Channels:\n\tMQIBindType=FASTPATH\n" > /etc/mqm/my.ini
      LABEL summary "My custom MQ image"
  strategy:
    type: Docker
    dockerStrategy:
      from:
        kind: "DockerImage"
        name: "cp.icr.io/cp/ibm-mqadvanced-server-integration:9.4.3.0-r1"
      pullSecret:
        name: ibm-entitlement-key
  output:
    to:
      kind: ImageStreamTag
      name: 'mymq:latest-amd64'
```

   ベースの IBM MQ が指定されている 2 箇所を、使用するバージョンとフィックスを表す正しいベース・イメージを指すように置き換える必要があります (詳しくは、5 ページの『IBM MQ Operator のリリース履歴』を参照)。 フィックスが適用されたときには、この手順を繰り返してイメージを再ビルドする必要があります。

   この例では、 IBM の公式イメージに基づいて新規イメージを作成し、 "my.mqsc" および "my.ini" というファイルを /etc/mqm ディレクトリーに追加します。 このディレクトリー内にある MQSC ファイルまたは INI ファイルは、始動時にコンテナーによって適用されます。 INI ファイルは、**crtmqm -ii** オプションを使用して適用され、既存の INI ファイルとマージされます。 MQSC ファイルは、アルファベット順に適用されます。

   MQSC コマンドは、キュー・マネージャーが開始されるたびに 実行されるので、反復可能であることが重要です。 通常、これは、REPLACE パラメーターを DEFINE コマンドに追加すること、および IGNSTATE(YES) パラメーターを START コマンドまたは STOP コマンドに追加することを意味します。

   b) BuildConfig をサーバーに適用します。

```
oc apply -f mq-build-config.yaml
```

3. ビルドを実行してイメージを作成します。

   a) ビルドを開始します。

```
oc start-build mymq
```

   次のような出力が表示されます。

```
build.build.openshift.io/mymq-1 started
```

   b) ビルドの状況を確認します。

例えば、前の手順で返されたビルド ID を使用して、次のコマンドを実行します。

```
oc describe build mymq-1
```

4. 新規イメージを使用してキュー・マネージャーをデプロイします。

   98 ページの『IBM MQ Operator を使用した単純なキュー・マネージャーのデプロイ』で説明している手順に従って、新規カスタム・イメージを YAML に追加します。

   以下の YAML スニペットを通常の QueueManager YAML に追加できます。*my-namespace* は使用する Red Hat OpenShift プロジェクト/名前空間です。*image* は前に作成したイメージの名前 (例えば、「mymq:latest-amd64」) です。

```
spec:
  queueManager:
    image: image-registry.openshift-image-registry.svc:5000/my-namespace/my-image
```

関連タスク

98 ページの『IBM MQ Operator を使用した単純なキュー・マネージャーのデプロイ』
この例では、一時 (非永続) ストレージを使用する「クイック・スタート」キュー・マネージャーをデプロイし、 IBM MQ セキュリティーをオフにします。 メッセージは、キュー・マネージャーの再始動後は保持されません。 構成を調整することで、キュー・マネージャーのさまざまな設定を変更できます。

OpenShift CP4I を使用して、 IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator

IBM MQ ライセンスでキューマネージャを構成するには、キューマネージャに注釈を適用する必要があります。

始める前に

注 : これらの説明は、IBM MQ Operator を使用する場合にのみ適用されます。 Kubernetes にキュー・マネージャーをデプロイしている場合は、代わりに IBM MQ コンテナー・ライセン ス・アノテーションを参照のこと。

このタスクについて

キューマネージャにアノテーションを追加するには、キューマネージャを作成する時か、spec.annotations を使って既存のキューマネージャを更新する必要があります。

重要 : キュー・マネージャの YAML を更新すると、ポッドの再起動がトリガされます。

手順

162 ページの『を使用してキュー・マネージャーにアノテーションを追加する。 IBM MQ Operator』 の指示に従って、既存のキュー・マネージャのライセンス注釈を更新してください。

必要なライセンス注釈

IBM MQ

```
spec:
  annotations:
    productID: c661609261d5471fb4ff8970a36bccea
    productName: IBM MQ
```

IBM MQ for Non-Production Environment

```
spec:
  annotations:
    productID: 151bec68564a4a47a14e6fa99266deff
    productName: IBM MQ for Non-Production Environment
```

IBM MQ コンテナ・マルチインスタンス

```
spec:
  annotations:
    productID: 2dea73b866b648b6b4abe2a85eb76964
    productName: IBM MQ Container Multi Instance
```

IBM MQ 非生産環境のコンテナ・マルチインスタンス：

```
spec:
  annotations:
    productID: af11b093f16a4a26806013712b860b60
    productName: IBM MQ Container Multi Instance for Non-Production Environment
```

IBM MQ CP4I 権利付き

```
spec:
  annotations:
    cloudpakId: c8b82d189e7545f0892db9ef2731b90d
    cloudpakName: IBM Cloud Pak for Integration
    productID: c661609261d5471fb4ff8970a36bccea
    productName: IBM MQ
    productMetric: VIRTUAL_PROCESSOR_CORE
    productCloudpakRatio: 4:1
```

IBM MQ 非生産環境用、 エンタイトルメント付き CP4I

```
spec:
  annotations:
    cloudpakId: c8b82d189e7545f0892db9ef2731b90d
    cloudpakName: IBM Cloud Pak for Integration
    productID: 151bec68564a4a47a14e6fa99266deff
    productName: IBM MQ for Non-Production Environment
    productMetric: VIRTUAL_PROCESSOR_CORE
    productCloudpakRatio: 8:1
```

IBM MQ CP4I エンタイトルメント付きコンテナ・マルチインスタンス

のために：

```
spec:
  annotations:
    productName: IBM MQ Container Multi Instance
    productID: 2dea73b866b648b6b4abe2a85eb76964
    productMetric: VIRTUAL_PROCESSOR_CORE
    productCloudpakRatio: 10:3
    cloudpakName: IBM Cloud Pak for Integration
    cloudpakId: c8b82d189e7545f0892db9ef2731b90d
```

IBM MQ CP4I エンタイトルメント付き非生産環境向けコンテナ・マルチインスタンス

```
spec:
  annotations:
    cloudpakId: c8b82d189e7545f0892db9ef2731b90d
    cloudpakName: IBM Cloud Pak for Integration
    productID: af11b093f16a4a26806013712b860b60
    productName: IBM MQ Container Multi Instance for Non-Production Environment
    productMetric: VIRTUAL_PROCESSOR_CORE
    productCloudpakRatio: 20:3
```

OpenShift CP4I を使用してキュー・マネージャーにアノテーションを追加する。 IBM MQ Operator

アノテーションは、キューマネージャの作成時に追加することも、 `spec.annotations` を使用して既存のキューマネージャを更新することによって追加することもできます。

このタスクについて

これらの指示は、以下の記述の両方が真である場合にのみ適用されます：

- あなたは IBM MQ Operator を使っている。 キュー・マネージャーを Kubernetes に配備する場合は、281 ページの『IBM MQ コンテナライセンス注釈』 を参照してください。
- 既存のキュー・マネージャを編集しています。 新しいキュー・マネージャの場合は、代わりに作成時にアノテーションを追加します。

重要 : キュー・マネージャの YAML を更新すると、ポッドの再起動がトリガされます。

手順

- **オプション 1:** OpenShift コンソールを使用して注釈を追加します。
  a) キュー・マネージャーに移動する
    a. Red Hat OpenShift Container Platform クラスタ管理者の認証情報を使用して OpenShift コンソールにログインします。
    b. **プロジェクトを**、 IBM MQ Operator をインストールしたネームスペースに変更します。 **Project** ドロップダウン・リストからネームスペースを選択します。
    c. ナビゲーション・ペインで、 **OperatorsInstalled** > **Operators をクリックします。**
    d. インストールされているオペレーター "パネルのリストで、以下を見つけ、クリックする。 **IBM MQ**.
    e. **キューマネジャータブを**クリックします。
    f. 必要な**キューマネージャー名を**クリックします。
  b) キュー・マネージャに必要なアノテーションを追加します。
    a.「**YAML**」タブをクリックします。
    b. `spec` 。
    c. キュー・マネージャに必要なアノテーションを追加する。 `annotations` フィールドがまだ存在しない場合は作成する。
    d. **保存** をクリックします。

    キュー・マネージャー・ポッドが再起動し、アノテーションが適用される。
  c) 必要に応じて、他のキューマネージャについてもこれらの手順を繰り返してください。
- **オプション 2：** OpenShift コマンドラインインターフェイス（CLI）を使用して注釈を追加します。
  a) `oc login` コマンドを使用して Red Hat OpenShift CLI にログインする。
  b) `project` (名前空間) を、キュー・マネージャが配置されているものに設定します：

```
oc project <QUEUE-MANAGER-NAMESPACE>
```

  c) 更新したいキュー・マネージャの名前を特定してください：

```
oc get queuemanager
```

  d) キューマネージャを編集して、必要な注釈をキューマネージャに追加します：

```
oc edit queuemanager <QUEUE-MANAGER-NAME>
```

  e) 必要に応じて、他のキューマネージャについてもこれらの手順を繰り返してください。

OpenShift CP4I キュー・マネージャー・リソースへのカスタム・アノテーションとカスタム・ラベルの追加

QueueManager メタデータにカスタム・アノテーションとカスタム・ラベルを追加します。

このタスクについて

PVC を除くすべてのリソースにカスタム・アノテーションとカスタム・ラベルを追加できます。 カスタム・アノテーションまたはカスタム・ラベルが既存のキーと一致する場合は、IBM MQ Operator によって設定される値が使用されます。

手順

- カスタム・アノテーションを追加します。

  ポッドなどのキュー・マネージャー・リソースにカスタム・アノテーションを追加するには、`metadata` の下にアノテーションを追加します。 以下に例を示します。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
  annotations:
    annotationKey: "value"
```

- カスタム・ラベルを追加します。

  ポッドなどのキュー・マネージャー・リソースにカスタム・ラベルを追加するには、`metadata` の下にラベルを追加します。 以下に例を示します。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
  labels:
    labelKey: "value"
```

OpenShift CP4I 実行時 Webhook チェックの無効化

実行時 Webhook チェックによって、ストレージ・クラスがキュー・マネージャーで実行可能かどうかを確認します。 パフォーマンスを向上させたい場合や、ご使用の環境で有効でない場合は、このチェックを無効にできます。

このタスクについて

実行時 Webhook チェックは、キュー・マネージャー構成に対して実行します。 選択したキュー・マネージャー・タイプにストレージ・クラスが適しているかを確認します。

キュー・マネージャーの作成にかかる時間を短縮したい場合や、ご使用の特定の環境で有効でない場合は、チェックを無効にできます。

注 : 実行時 Webhook チェックを無効にすると、すべてのストレージ・クラス値が有効になります。 その結果、キュー・マネージャーの失敗につながることもあります。

手順

- 実行時 Webhook チェックを無効にします。

  `metadata` の下に以下のアノテーションを追加します。 以下に例を示します。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
  annotations:
    "com.ibm.cp4i/disable-webhook-runtime-checks" : "true"
```

OpenShift Operator2.1.0 CP4I キュー・マネージャー仕様に対するデフォルト値更新の無効化

IBM MQ Operator は、キュー・マネージャー仕様内の指定されていない値をデフォルト値で更新します。キュー・マネージャー仕様に変更を加えないようにする場合は、この動作を無効にすることができます。キュー・マネージャー状況フィールドは引き続き更新されます。

手順

- キュー・マネージャーのデフォルト値の更新を無効にします。

  metadata の下に以下のアノテーションを追加します。 以下に例を示します。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart-cp4i
  annotations:
    "com.ibm.mq/write-defaults-spec" : "false"
```

  注: クイック・スタート・サンプルには、デフォルトでこのアノテーションが適用されています。

読み取り専用ルート・ファイル・システムを使用した IBM MQ コンテナーの実行

読み取り専用のルート・ファイル・システムで実行するように IBM MQ コンテナーを構成することができます。 これにより、攻撃者はコンテナー内の悪意のあるコードをコピーして実行することができなくなります。

このタスクについて

読み取り専用ルート・ファイル・システムを有効にすると、コンテナー・ファイルが変更不可能になります。 つまり、コンテナー・ファイル・システムでは、ファイルを表示することはできますが、変更することはできず、新規ファイルを作成することはできません。 ファイルは、マウントされたファイル・システム上でのみ変更または作成することができます。

読み取り専用ルート・ファイル・システムが使用可能な場合、2 つの一時ボリューム Scratch および Tmp が作成され、それぞれコンテナー内の /run ディレクトリーおよび /tmp ディレクトリーにマウントされます。

- 「スクラッチ」 ボリュームには、キュー・マネージャーの構成に使用されるファイル、鍵ストア、およびその他のファイルが含まれています。
- Tmp ボリュームには、キュー・マネージャー RAS ファイルなどの診断ファイルが含まれています。

これらのボリュームは一時ボリュームであるため、これらのボリューム上のファイルはポッドの再始動時に失われます。

キュー・マネージャー・データ用に作成されるボリュームのタイプは、ストレージ・タイプによって異なります。 デフォルトでは、永続ボリュームがマウントされます。 あるいは、ストレージ・タイプが ephemeral の場合は、一時ボリュームがマウントされます。 ボリューム内のデータのサイズが **sizeLimit** プロパティーに指定された値を超える場合、 Kubernetes はコンテナーを排出して新しいコンテナーを作成できます。

読み取り専用ルート・ファイル・システムは、デフォルトでは使用可能になっていません。 有効にするには、以下の手順を実行します。

手順

1. spec.securityContext API を使用して、読み取り専用ルート・ファイル・システムを有効にします。

   キュー・マネージャーの場合は、269 ページの『.spec.securityContext』 の **readOnlyRootFilesystem** プロパティーを true に設定します。

IBM MQ Operator は、 Scratch および Tmp という 2 つの一時ボリュームを作成します。

2. オプション: キュー・マネージャーのデータ・ストレージ・タイプを設定または変更します。

   デフォルトでは、永続ボリューム要求は /mnt/mqm にマウントされます。 あるいは、 267 ページの『.spec.queueManager.storage.queueManager』で **type** プロパティーが ephemeral に設定されている場合、一時ボリュームが作成され、マウントされます。

3. 一時ボリュームごとに、データがどの程度増大する可能性があるかを慎重に検討してください。 SI 単位を含め、 **sizeLimit** プロパティーの値を適宜設定します。

   - Scratch 一時ボリュームの場合、 269 ページの『.spec.queueManager.storage.scratch』で **sizeLimit** プロパティーを設定します。 デフォルト値は「100M」です。
   - Tmp 一時ボリュームの場合は、 269 ページの『.spec.queueManager.storage.tmp』で **sizeLimit** プロパティーを設定します。 デフォルト値は「2Gi」です。
   - キュー・マネージャー・ボリュームの **type** が ephemeral に設定されている場合、 267 ページの『.spec.queueManager.storage.queueManager』に **sizeLimit** プロパティーを設定します。 デフォルト値は「2Gi」です。

OpenShift V 9.4.0 IBM MQ Operator を使用した基本レジストリーでの IBM MQ Console の構成

IBM MQ Console にログインするために、キュー・マネージャーに独自の構成を提供することができます。

始める前に

IBM MQ Advanced for Developers ライセンスを使用してキュー・マネージャーをデプロイする場合は、単純な構成が組み込まれています。 292 ページの『admin および app ユーザーのパスワードを指定する方法を説明するキュー・マネージャー YAML の例』を参照してください。 IBM Cloud Pak for Integration ライセンス・キュー・マネージャーをデプロイする場合は、シングル・サインオンを使用して IBM MQ Console にログインするために、 IBM Cloud Pak for Integration Keycloak との統合を有効にすることができます。 218 ページの『IBM MQ Console クラスターにデプロイされた Red Hat OpenShift への接続』を参照してください。

手順

1. **パスワードを作成し、 securityUtility を使用して暗号化します。**

   ConfigMap は、キュー・マネージャーへのアクセスに使用する資格情報を保管するために使用されます。 セキュリティーを強化するために、 securityUtility コマンドを使用してこれらの資格情報をエンコードします。

   あるいは、 Kubernetes 層で資格情報を保護するシークレットを使用することもできます。 ただし、モニター・ツールやトラブルシューティング・ツールを使用すると、基礎となるファイルがセキュアでなくなる可能性があります。

2. オプション: **Red Hat OpenShift コマンド・ライン・インターフェース (CLI) にログインします。**

   OpenShift CLI を使用している場合は、 `oc login` を使用してログインします。

   あるいは、 OpenShift コンソールを使用することもできます。

3. **ご使用の構成で ConfigMap を作成します。**

   XML 構成の作成については、 IBM MQ Console および REST API セキュリティーを参照してください。

   以下の例では、グループ MQWebAdminGroup 内にユーザーを作成します。 MQWebAdminGroup のメンバーには、MQWebAdmin 役割が割り当てられます。 この例のそれぞれの指定の意味は次のとおりです。

   - *USERNAME* および *PASSWORD* を独自の値に置き換える **必要があります** 。 この例では、 *USERNAME* が 2 回使用されていることに注意してください。

     NAMESPACE は、 IBM MQ Operator がデプロイされ、キュー・マネージャーがデプロイされる場所、または既にデプロイされている場所として指定する **必要があります** 。

a) OpenShift コンソールまたはコマンド行を使用して、以下の ConfigMap を作成します。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: mqwebuserconfigmap
  namespace: NAMESPACE
data:
  mqwebuser.xml: |
    <?xml version="1.0" encoding="UTF-8"?>
    <server>
      <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>basicAuthenticationMQ-1.0</feature>
      </featureManager>
      <enterpriseApplication id="com.ibm.mq.console">
        <application-bnd>
          <security-role name="MQWebAdmin">
            <group name="MQWebAdminGroup" realm="defaultRealm"/>
          </security-role>
        </application-bnd>
      </enterpriseApplication>
      <basicRegistry id="basic" realm="defaultRealm">
        <user name="USERNAME" password="PASSWORD"/>
        <group name="MQWebAdminGroup">
          <member name="USERNAME"/>
        </group>
      </basicRegistry>
      <sslDefault sslRef="mqDefaultSSLConfig"/>
    </server>
```

b) オプション: コマンド行を使用する場合は、 ConfigMap を適用します。

```
oc apply -f mqwebuserconfigmap.yaml
```

残りのステップでは、以下のいずれかのオプションを選択します。

- IBM MQ Console にアクセスするための構成を使用して、新しいキュー・マネージャーをデプロイします。
- 既存のキュー・マネージャーに対するアクセス権限を IBM MQ Console に付与する構成を適用します。

4. オプション: **IBM MQ Console にアクセスするための構成を使用して、新規キュー・マネージャーをデプロイします。**

a) キュー・マネージャーを作成します。

以下のいずれかのオプションを使用して、認証および許可プロバイダーを 手動 に設定し、新しく作成した ConfigMap mqwebuserconfigmap を指定します。

- オプション 1: キュー・マネージャー YAML を使用する

  キュー・マネージャー YAML の web セクションの下に以下のコードを追加します。

```
...
web:
    enabled: true
    console:
      authentication:
        provider: manual
      authorization:
        provider: manual
    manualConfig:
      configMap:
        name: mqwebuserconfigmap
```

- オプション 2: OpenShift コンソール・フォーム・ビューを使用して、以下の操作を行います。

  i) OpenShift コンソールで、「**オペレーター**」 >「**インストール済みオペレーター**」を選択します。

  ii) IBM MQ Operator のデプロイメントを選択します。

  iii)「**キュー・マネージャー**」 を選択し、 「**作成**」 **QueueManager** をクリックします。

  iv) キュー・マネージャーに関連するオプションを選択します。

v)「**Web**」 を選択し、 「**Web サーバーを使用可能にする**」 を `true` に設定します。

vi)「**拡張構成**」 リスト・ボックスを開きます。

vii)「**コンソール**」 リスト・ボックスで、 「**認証**」 と 「**許可**」 の両方の **プロバイダー** を `manual` に設定します。

viii)「 **構成** 」 リスト・ボックスを開きます。

ix) **ConfigMap** リスト・ボックスを開き、ステップ 166 ページの『3』 で作成した ConfigMap `mqwebuserconfigmap` を選択します。

x)「**作成**」 をクリックします。

これで、ステップ 166 ページの『3』 で作成した `ConfigMap` で指定した資格情報を使用して、新しいキュー・マネージャーの IBM MQ Console にアクセスできるようになりました。

5. オプション: **既存のキュー・マネージャーに対して IBM MQ Console を有効にする構成を適用します。**

   IBM MQ Console を有効にするキュー・マネージャーの YAML を編集します。

   a. OpenShift コンソールで、 「**オペレーター**」 > 「**インストール済みオペレーター**」を選択します。

   b. IBM MQ Operator のデプロイメントを選択します。

   c.「**キュー・マネージャー**」 を選択し、キュー・マネージャーの名前を選択します。

   d.「**YAML**」を選択します。

   e. キュー・マネージャー YAML の既存の web セクションを以下のコードに置き換えます。

```
...
web:
    enabled: true
    console:
      authentication:
        provider: manual
      authorization:
        provider: manual
    manualConfig:
      configMap:
        name: mqwebuserconfigmap
```

   f. **保存** をクリックします。

   これで、ステップ 166 ページの『3』 で作成した `ConfigMap` で指定した資格情報を使用して、既存のキュー・マネージャーの IBM MQ Console にアクセスできるようになりました。

OpenShift V 9.4.0 V 9.4.0 永続ボリュームの拡張

ご使用のストレージ・プロバイダーがボリューム拡張をサポートしている場合は、このタスクを使用して永続ボリュームを拡張します。 ストレージ・プロバイダーによっては、拡張がオンラインまたはオフラインで行われる場合があります。

始める前に

ボリュームが正常に拡張されるには、ストレージ・プロバイダーが拡張要求を満たす必要があります。 オンライン・サイズ変更がサポートされているかどうか、およびオフライン・サイズ変更手順については、ご使用のストレージ・プロバイダーの資料を参照してください。

ストレージ・プロバイダーが拡張要求を満たすことができない場合、永続ボリューム要求は警告またはエラーのある状態になる可能性があります。 拡張が失敗した場合、 OpenShift 管理者は永続ボリューム要求の状態を手動でリカバリーし、拡張を取り消すことができます。 Red Hat OpenShift 資料の「 ボリューム拡張時の障害からのリカバリー 」を参照してください。

このタスクについて

永続ストレージの管理に役立つように、 Kubernetes では以下の 2 つの API リソースが定義されています。

- PersistentVolume (PV)。これは、管理者によってプロビジョンされたか、ストレージ・クラスを使用して動的にプロビジョンされた、クラスター内のストレージの一部です。 これは、静的または動的にプロビジョンできます。
- PersistentVolumeClaim (PVC)。これは、ユーザーによるストレージの要求です。 また、リソースに対するクレーム・チェックとしても機能します。

詳しくは、 Kubernetes 資料の 永続ボリューム を参照してください。

警告 :

- キュー・マネージャー PVC の作成に使用されるストレージ・クラスがオンライン・サイズ変更をサポートしていない場合は、オフライン・サイズ変更が行われます。 ボリューム拡張を完了するには、オフラインでのサイズ変更中にユーザー介入が必要になるため、キュー・マネージャーでダウン時間が発生します。
- 複数インスタンス・キュー・マネージャーの共有ボリュームのオフライン・サイズ変更の場合、ユーザー介入の実行時にアクティブ・ポッドとスタンバイ・ポッドの両方を同時に停止する必要があります。
- OpenShift では、PVC のサイズの縮小はサポートされていません。 永続ボリュームのサイズを削減しようとすると、キュー・マネージャーは「失敗」状態になります。
- この手順は、一時ボリュームには適用されません。

IBM MQ コンテナーによって使用される PV を展開するには、以下の手順を実行します。

手順

1. **ボリュームの拡張の準備**

   a) 拡張するボリュームを決定します。

   b) ボリュームで使用されている 1 つ以上のストレージ・クラスを判別します。

   以下に例を示します。

```
spec:
  queueManager:
    storage:
      persistedData:
        enabled: true
        type: persistent-claim
        class: ocs-storagecluster-cephfs (1)
      queueManager:
        type: persistent-claim
      recoveryLogs:
        enabled: true
        type: persistent-claim
      defaultClass: ocs-storagecluster-ceph-rbd (2)
```

   注:

   - (1) ボリュームが特定のストレージ・クラスを定義している場合、これはこのタイプの PVC によって使用されます。
   - (2) **defaultClass** が設定されている場合、このストレージ・クラスは、特定のストレージ・クラスを持たないすべてのボリュームに使用されます。 **defaultClass** が設定されておらず、ボリューム・タイプにクラスが指定されていない場合は、クラスターのデフォルトのストレージ・クラスが使用されます。

   基礎となる PVC を記述することで、使用中のストレージ・クラスを確認することもできます。 以下に例を示します。

```
oc describe pvc pvc-name
```

   c) ストレージ・クラスがボリューム拡張をサポートしていることを確認します。

   ストレージ・クラスには、プロパティー **.allowVolumeExpansion** が定義されている場合があります。

- このプロパティーが `true` に設定されている場合、ボリューム拡張がサポートされます。
- このプロパティーが `false` に設定されているか、このプロパティーが定義されていない場合、ストレージ・クラスはボリューム拡張を許可しません。 この場合は、ストレージ・プロバイダーの資料を参照して、この機能を有効にできるかどうかを確認してください。

また、ストレージ・クラスを記述して、そのストレージ・クラスがボリューム拡張をサポートしているかどうかを判別することもできます。 以下に例を示します。

```
oc describe sc storage-class-name
```

d) ストレージ・プロバイダーの資料を参照して、ボリューム拡張にオンラインまたはオフラインの手順が使用されているかどうかを確認してください。

オフライン手順では、キュー・マネージャー・ポッドを手動で再始動する必要がありますが、オンライン手順では再始動しません。 オフラインでのサイズ変更手順については、ご使用のストレージ・プロバイダーの資料を参照してください。

e) キュー・マネージャーに「StorageMismatch」という状況条件があるかどうかを確認します。

キュー・マネージャーがこの状況条件になっている場合、ボリューム拡張を使用可能にすると、条件にリストされているボリュームが拡張されます。 これを行わない場合は、キュー・マネージャー定義内の各ボリューム・タイプに関連付けられているサイズ・フィールドを、プロビジョンされた PVC と一致するように変更します。 一致しないすべてのボリュームに対してこれが行われると、状況条件は除去されます。

2. **ボリュームの拡張**

警告 :

- キュー・マネージャー定義のボリューム・サイズ・フィールドのいずれかを以前に変更したことがある場合は、キュー・マネージャー定義で **.allowVolumeExpansion** が `true` に設定されていると、ボリュームの拡張が開始されます。
- ストレージ・プロバイダーによっては、ファイル・システムの制限やローカル・ハードウェアの可用性のために、ボリュームの最大サイズに制限がある場合があります。 障害を回避するには、ボリュームを拡張する前に、ストレージ・プロバイダーの資料でこれらの制限を確認してください。
- PVC サイズの縮小は、OpenShift ではサポートされていません。 ボリュームのサイズを拡張する場合、縮小することはできません。 これを行おうとして失敗した場合、 IBM MQ Operator は PVC を元の状態に戻すことができません。

ボリューム拡張を示すキュー・マネージャー定義の例:

```
spec:
  queueManager:
    storage:
      allowVolumeExpansion: true (A)
      persistedData:
        enabled: true
        type: persistent-claim
        size: 3Gi (B)
      queueManager:
        type: persistent-claim
        size: 4Gi (B)
      recoveryLogs:
        enabled: true
        type: persistent-claim
        size: 3Gi (B)
```

a) キュー・マネージャーのボリューム拡張を許可するには、キュー・マネージャーのフィールド **.spec.queueManager.storage.allowVolumeExpansion** (A) を `true` に設定します。

b) これで、使用可能な任意のボリューム・タイプのサイズ・フィールド (B) を増やすことができます。これらの変更を適用すると、ボリューム拡張が開始されます。

3. **PVC がサイズ変更されたことを確認します。**

注 :

- ボリュームの拡張には時間がかかる場合があります。 検証が初めて成功しなかった場合は、数分待ってから再度検証することを検討してください。
- ボリューム拡張は、オンライン・サイズ変更が実行された場合にのみ、ユーザー・アクションなしで完了します。
- 一部のストレージ・プロバイダーは、要求したストレージ・サイズを切り上げます。 拡張ボリュームのサイズは、要求のサイズ以上でなければなりません。

a) 状況条件については、キュー・マネージャーを確認してください。 条件、説明、および推奨アクションについては、以下の表を参照してください。

表 1. 保管のステータス条件

CONDITION	MESSAGE	EXPLANATION
StorageMismatch	Storage sizes defined in the QueueManager resource do not match the capacity of one or more provisioned PVCs [pvc-list]. AllowVolumeExpansion is set to false in the QueueManager resource so the MQ Operator will not attempt to reconcile these differences.	キュー・マネージャー定義で **.allowVolumeExpansion** が true に設定されていないため、ボリューム拡張は行われません。
StorageExpansionPending	Volume expansion is pending for the following PVCs [pvc-list]	ボリューム拡張はまだ行われています。 この状況状態が長期間続く場合は、オフラインのサイズ変更またはサイズ変更の失敗が発生している可能性があるため、以下の手順に従って詳細情報を収集してください。
Failed	'Failed' 状況条件を作成する可能性のある、ストレージ関連のメッセージが多数あります。 例: 'MQ Queue Manager failed to deploy: persistentvolumeclaims "<pvc>" is forbidden: only dynamically provisioned pvc can be resized and the storageclass the provisions the pvc must support resize.'	キュー・マネージャーに、ストレージを参照するテキストを持つ 'Failed' 状況条件がある場合は、状況条件内のメッセージを参照してください。 ここに示されているメッセージの例は、拡張をサポートしていないストレージ・クラスを使用していることが原因です。

b) 拡張した PVC ごとに、キュー・マネージャー定義で指定された値以上になるように容量が増加していることを確認します。

HA キュー・マネージャーには、各タイプの複数の PVC がある場合があります。 PVC の容量を取得するには、以下のコマンドを実行します。

```
oc get pvc pvc-name -o template --template '{{.status.capacity.storage}}'
```

c) PVC に、サイズ変更の失敗を示す状況条件またはイベントがないことを確認します。

```
oc describe pvc pvc-name
```

- PVC が「Waiting for user to (re-) start a pod to finish file system resize of volume on node」というメッセージとともに状況条件 FileSystemResizePending を持っている可能性があります。 この状況条件は、オンラインとオフラインの両方のサイズ変更で発生します。 オンライン・サイズ変更の場合、この状況条件は、オンライン・サイズ変更の完了後にユーザー処置なしで表示されなくなります。
- サイズ変更が失敗したことを示すイベントまたは状況条件が PVC にある場合は、 Red Hat OpenShift 資料の ボリューム拡張時の障害からのリカバリー を参照してください。

d) キュー・マネージャーのポッドに、サイズ変更の失敗を示す状況条件やイベントがないことを確認します。 HA デプロイメントの場合は、各レプリカを確認します。

```
oc describe pod queue-manager-pod-name
```

- サイズ変更が失敗したことを示すイベントまたは状況条件がポッドにある場合は、 Red Hat OpenShift 資料の ボリューム拡張時の障害からのリカバリー を参照してください。 エラー・テキストは、問題の解決に役立つ場合があります。また、リカバリー後にサイズ変更を再試行すると、同じ問題の発生を防ぐことができます。

4. **オフラインのサイズ変更時にポッドを再始動する**

   ストレージ・プロバイダーがボリュームの拡張時にオフラインのサイズ変更手順を使用する場合、ボリュームの拡張を完了するには、サイズ変更するボリュームをマウントするキュー・マネージャー・ポッドを再始動する必要があります。

   複数インスタンス・キュー・マネージャーの場合、リカバリー・ログと永続データ・ボリュームは、アクティブ・ポッドとスタンバイ・ポッドの両方で共有されます。 これらのボリュームのサイズ変更を完了するには、両方のポッドを同時に停止します。

   オフラインでのサイズ変更手順については、ご使用のストレージ・プロバイダーの資料を参照してください。

OpenShift V 9.4.1 配備されたキュー・マネージャーのライセンス注釈の修正

IBM MQ キューマネージャのバージョン 9.4.1.0-r1 より前のバージョンで、いくつかのキューマネージャが不正なライセンスアノテーションとともに配備されていました。 バージョン 9.4.1.0-r1 以降、すべてのキューマネージャは正しいライセンスアノテーションとともに自動的に配備されます。

このタスクについて

すでにデプロイされている'IBM MQ キューマネージャには、ライセンスアノテーションを修正するための 2 つのオプションがある：

- キューマネージャをバージョン 9.4.1.0-r11 にアップグレードしてください（'IBM MQ Operator33.3.0 以降が必要です）。 69 ページの『IBM MQ Operator とキュー・マネージャーのアップグレード』を参照してください。 キューマネージャの YAML に無効なライセンスアノテーションが含まれている場合、各キューマネージャをアップグレードする際にプロンプトが表示されるので、これを解決する必要があることに注意してください。
- アップグレードされておらず、以下のライセンス使用法を使用してデプロイされている古いキューマネージャについては、キューマネージャの YAML を更新することでライセンスアノテーションを修正することができます。 以下の手順を実行して、指定されたアノテーションをキュー・マネージャ YAML に追加します。 このプロセスは、各キュー・マネージャに対して個別に完了しなければならないことに注意してください。

  重要 : キュー・マネージャの YAML を更新すると、ポッドの再起動がトリガされます。

影響を受けるライセンスの使用

- IBM MQ Advanced for Non-Production Environment と'CP4I エンタイトルメント
- IBM MQ Advanced for Developers

- IBM MQ Advanced Container Multi Instance
- IBM MQ Advanced Container Multi Instance と'CP4I エンタイトルメント
- IBM MQ Advanced Container Multi Instance for Non-Production Environment と'CP4I エンタイトルメント

手順

162 ページの『を使用してキュー・マネージャーにアノテーションを追加する。 IBM MQ Operator』 の指示に従って、既存のキュー・マネージャのライセンス注釈を更新してください。

必要なライセンス注釈

IBM MQ Advanced for Non-Production Environment と'CP4I エンタイトルメント

```
spec:
  annotations:
    productName: IBM MQ Advanced for Non-Production Environment
```

IBM MQ Advanced for Developers

```
spec:
  annotations:
    productName: IBM MQ Advanced for Developers (Non-Warranted)
    productChargedContainers: [container_name]
```

ここで、*container_name* はキューマネージャポッドの YAML にあり、'`spec.containers[0].name` として指定される。

IBM MQ Advanced Container Multi Instance

```
spec:
  annotations:
    productID: bd35bff411bb47c2a3f3a4590f33a8ef
    productName: IBM MQ Advanced Container Multi Instance
```

IBM MQ Advanced Container Multi Instance と'CP4I エンタイトルメント

```
spec:
  annotations:
    productID: bd35bff411bb47c2a3f3a4590f33a8ef
    productName: IBM MQ Advanced Container Multi Instance
    productCloudpakRatio: 5:3
```

IBM MQ Advanced Container Multi Instance for Non-Production Environment と'CP4I エンタイトルメント

```
spec:
  annotations:
    productID: 31f844f7a96b49749130cd0708fdbb17
    productName: IBM MQ Advanced Container Multi Instance for Non-Production Environments
    productCloudpakRatio: 10:3
```

Kubernetes Deploying and configuring queue managers on Kubernetes

ネイティブ・リソースを使ってキュー・マネージャーを Kubernetes に配置する方法。

このタスクについて

MQ Adv. Kubernetes V 9.4.1 例 Kubernetes での単純なキューマネージャの設定

この例では、エフェメラル（非永続的）ストレージを使用し、IBM MQ のセキュリティをオフにした、単純な「クイックスタート」キューマネージャを配備します。 メッセージはキュー・マネージャの再起動に渡って持続されません。 構成を調整することで、キュー・マネージャーのさまざまな設定を変更できます。

始める前に

この例を完了するには、まず以下の前提条件を満たしておく必要があります。

- この例のために 'Kubernetes 名前空間を作成する。
- コマンドラインで「Kubernetes クラスタにログインし、上記のネームスペースに切り替える。
- 上記のネームスペースの'86 ページの『プルシークレットを作成して「Kubernetes」を使用する準備』の手順に従っていることを確認する。

このタスクについて

この例では、'Service、'ServiceAccount、'StatefulSetYAML を提供し、'Kubernetes にデプロイするキューマネージャを定義している。 この例では、最低限'IBM MQ Advanced 9.4.1 が必要である。

手順

1. IBM MQ トラフィックに一貫した IP アドレスを提供するために「Service 作成する。

   先ほど作成したネームスペースにいることを確認し（始める前にを参照）、'Kubernetes コマンドラインかコンソールを使って次の YAML を適用します。

   ```
   apiVersion: v1
   kind: Service
   metadata:
     labels:
       app.kubernetes.io/component: integration
       app.kubernetes.io/instance: quickstart
       app.kubernetes.io/name: ibm-mq
       app.kubernetes.io/version: 9.4.3.0
     name: quickstart-ibm-mq
   spec:
     ports:
     - name: qmgr
       port: 1414
       protocol: TCP
       targetPort: 1414
     - name: metrics
       port: 9157
       protocol: TCP
       targetPort: 9157
     selector:
       app.kubernetes.io/component: integration
       app.kubernetes.io/instance: quickstart
       app.kubernetes.io/name: ibm-mq
     type: ClusterIP
   ```

2. キュー・マネージャーに使用する'ServiceAccount を作成する。

   以下の例の'ServiceAccount は'imagePullSecret を参照し、画像が引き出されたときにエンタイトルメントチェックに使用される。 詳しくは、86 ページの『プルシークレットを作成して「Kubernetes」を使用する準備』 を参照してください。

   ```
   kind: ServiceAccount
   apiVersion: v1
   metadata:
     name: quickstart-ibm-mq
     labels:
       app.kubernetes.io/component: integration
       app.kubernetes.io/instance: quickstart
       app.kubernetes.io/name: ibm-mq
       app.kubernetes.io/version: 9.4.3.0
   ```

```
imagePullSecrets:
  - name: ibm-entitlement-key
```

3. StatefulSet によって管理されるキューマネージャ Pod を作成する。

   以下の YAML では、'namespace 名前は'mq-test であると仮定します。 この例では、'Kubernetes 環境で使用している名前と一致するように変更してください。

   この YAML は、'Kubernetes 環境の指定した名前空間に単一のインスタンスキューマネージャを作成します。 この例はエフェメラル・ストレージのため、このインスタンスには永続ボリュームは作成されない。

   注: この例では、 IBM MQ Advanced アノテーションを使用しています。 IBM MQ ライセンスを使用するには、 spec.template.metadata の注釈を、 281 ページの『IBM MQ コンテナライセンス注釈』 の IBM MQ または IBM MQ for Non-Production Environment で指定されている値に編集する。

```
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: quickstart-ibm-mq
  namespace: mq-test
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: quickstart
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
spec:
  persistentVolumeClaimRetentionPolicy:
    whenDeleted: Retain
    whenScaled: Retain
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app.kubernetes.io/component: integration
      app.kubernetes.io/instance: quickstart
      app.kubernetes.io/name: ibm-mq
  template:
    metadata:
      labels:
        app.kubernetes.io/component: integration
        app.kubernetes.io/instance: quickstart
        app.kubernetes.io/name: ibm-mq
        app.kubernetes.io/version: 9.4.3.0
        statefulSetName: quickstart-ibm-mq
      annotations:
# These annotations are critical for the IBM License Service to charge you correctly
        productChargedContainers: qmgr
        productID: 208423bb063c43288328b1d788745b0c
        productMetric: PROCESSOR_VALUE_UNIT
        productName: IBM MQ Advanced
        productVersion: 9.4.3.0
    spec:
      restartPolicy: Always
      serviceAccountName: quickstart-ibm-mq
      terminationGracePeriodSeconds: 30
      securityContext: {}
      containers:
        - resources:
            limits:
              cpu: '1'
              memory: 1Gi
            requests:
              cpu: '1'
              memory: 1Gi
          readinessProbe:
            exec:
              command:
                - chkmqready
            initialDelaySeconds: 10
            timeoutSeconds: 3
            periodSeconds: 5
            successThreshold: 1
            failureThreshold: 1
          terminationMessagePath: /run/termination-log
          name: qmgr
          livenessProbe:
            exec:
```

```
              command:
                - chkmqhealthy
            initialDelaySeconds: 90
            timeoutSeconds: 5
            periodSeconds: 10
            successThreshold: 1
            failureThreshold: 1
          env:
            - name: LICENSE
              # if you accept the license, change the following value to "accept"
              value: view
            - name: MQ_QMGR_NAME
              value: QUICKSTART
            - name: MQ_MULTI_INSTANCE
              value: 'false'
            - name: MQ_ENABLE_METRICS
              value: 'true'
            - name: MQ_ENABLE_EMBEDDED_WEB_SERVER
              value: 'false'
            - name: MQ_LOGGING_CONSOLE_FORMAT
              value: basic
            - name: DEBUG
              value: 'false'
            - name: MQ_ENABLE_TRACE_STRMQM
              value: 'false'
            - name: MQ_EPHEMERAL_PREFIX
              value: /run/mqm
            - name: MQ_GRACE_PERIOD
              value: '29'
            - name: MQ_NATIVE_HA
              value: 'false'
          securityContext:
            allowPrivilegeEscalation: false
            capabilities:
              drop:
                - ALL
            privileged: false
            runAsNonRoot: true
            readOnlyRootFilesystem: false
          ports:
            - containerPort: 1414
              protocol: TCP
            - containerPort: 9157
              protocol: TCP
          terminationMessagePolicy: File
# The image to pull.  This example uses a prebuilt MQ Advanced image from IBM Container
Registry
          image: cp.icr.io/cp/ibm-mqadvanced-server:9.4.1.0-r1
          imagePullPolicy: IfNotPresent
      serviceAccount: quickstart-ibm-mq
      dnsPolicy: ClusterFirst
  serviceName: qm
  podManagementPolicy: OrderedReady
  updateStrategy:
    type: OnDelete
```

4. キュー・マネージャーが稼働していることの確認

   キュー・マネージャーがデプロイされます。 キュー・マネージャのポッドをリストアップするなどして、「Running 状態であることを確認する：

   ```
   kubectl get pods --selector app.kubernetes.io/instance=quickstart
   ```

   READY フィールドは値'1/1 示さなければならない。

5. キュー・マネージャーの状況の表示

   キュー・マネージャ・ポッドの完全名を指定して、以下のコマンドを実行してください：

   ```
   kubectl exec -t <qmgr_pod_name> -- dspmq -m QUICKSTART
   ```

   すると、次のような応答が返ってくるはずだ：

   ```
   QMNAME(QUICKSTART) STATUS(Running)
   ```

タスクの結果

あなたは単純な'IBM MQ キュー・マネージャ（シングル・インスタンス型）を'Kubernetes プラットフォームにデプロイすることに成功しました。

MQ Adv. Kubernetes V 9.4.1 例 Kubernetes ネイティブ HA の設定

この例では、ネイティブの高可用性機能を使用したキューマネージャを、YAML リソースを直接使用して'Kubernetes に配置します。 相互 TLS は認証に使用され、TLS 証明書からキューマネージャの ID にマッピングされる。

始める前に

注 : この例では、 IBM MQ Advanced ライセンスを使用しています。 IBM MQ 9.4.2 から、キューマネージャにライセンスを追加することで、 IBM MQ ライセンスを持つキューマネージャがネイティブ HA とクロスリージョンレプリケーション機能を使用するように設定することもできます。 詳しくは、186 ページの『IBM MQ のキューマネージャでネイティブ HA とクロスリージョンレプリケーションを設定する。Kubernetes』 を参照してください。

この例を完了するには、まず以下の前提条件を満たしておく必要があります。

- この例のために 'Kubernetes 名前空間を作成する。
- コマンドラインで「Kubernetes クラスタにログインし、上記のネームスペースに切り替える。
- 上記のネームスペースの「86 ページの『プルシークレットを作成して「Kubernetes」を使用する準備』」の手順に従っていることを確認してください。

このタスクについて

この例では、'Service、'ServiceAccount、'StatefulSetYAML を提供し、'Kubernetes にデプロイするキューマネージャを定義している。 また、TLS を有効にしてキュー・マネージャーをデプロイするために必要な追加のステップも詳しく説明しています。 この例では、最低限'IBM MQ Advanced 9.4.1 が必要である。

手順

1. 104 ページの『OpenSSL を使用した自己署名 PKI の作成』」で説明したように、証明書のペアを作成する。
2. MQSC コマンドと INI ファイルを含むコンフィグマップを作成する。

   新しいキューと 'SVRCONN チャネルを作成し、チャネルへのアクセスを許可する チャネル認証レコードを追加するための MQSC コマンドを含む 'Kubernetes'ConfigMap を作成する。

   先ほど作成したネームスペースにいることを確認し（始める前にを参照）、'Kubernetes コマンドラインを使って次の YAML を適用します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-nativeha-configmap
data:
  example.mqsc: |
    DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
    SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
    SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(BROWSE,PUT,GET,INQ)
  example.ini: |
    Service:
        Name=AuthorizationService
        EntryPoints=14
        SecurityPolicy=UserExternal
```

```
    NativeHAInstance:
      Name=exampleqm-ibm-mq-0
      ReplicationAddress=exampleqm-ibm-mq-replica-0(9414)
    NativeHAInstance:
      Name=exampleqm-ibm-mq-1
      ReplicationAddress=exampleqm-ibm-mq-replica-1(9414)
    NativeHAInstance:
      Name=exampleqm-ibm-mq-2
      ReplicationAddress=exampleqm-ibm-mq-replica-2(9414)
```

MQSC は *MTLS.SVRCONN* というチャネルと *EXAMPLE.QUEUE* というキューを定義している。 このチャネルは、*example-app1* という「コモンネーム」を持つ証明書を提示するクライアントにのみアクセスを許可するように設定されている。 これは、ステップ「177 ページの『1』」で作成された証明書の 1 つで使用されるコモン・ネームである。 この共通名を持つこのチャネル上の接続は、*app1* というユーザ ID にマップされます。このユーザ ID は、キューマネージャへの接続と、例のキューへのアクセスを許可されています。 INI ファイルは、*app1* ユーザー ID が外部ユーザーレジストリに存在する必要がないことを意味するセキュリティポリシーを有効にする。 また、ネイティブ HA で使用するレプリケーションアドレスも設定する。

3. IBM MQ トラフィックに一貫した IP アドレスを提供するために「Service 作成する。

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
  name: exampleqm-ibm-mq
spec:
  ports:
  - name: qmgr
    port: 1414
    protocol: TCP
    targetPort: 1414
  - name: metrics
    port: 9157
    protocol: TCP
    targetPort: 9157
  selector:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
  type: ClusterIP
```

4. 各 Native HA レプリカが相互に通信するために使用する内部'Service を作成する。

```
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
  name: exampleqm-ibm-mq-replica-0
spec:
  ports:
  - port: 9414
    protocol: TCP
    targetPort: 9414
  publishNotReadyAddresses: true
  selector:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    statefulset.kubernetes.io/pod-name: exampleqm-ibm-mq-0
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
```

```
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
  name: exampleqm-ibm-mq-replica-1
spec:
  ports:
  - port: 9414
    protocol: TCP
    targetPort: 9414
  publishNotReadyAddresses: true
  selector:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    statefulset.kubernetes.io/pod-name: exampleqm-ibm-mq-1
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
  name: exampleqm-ibm-mq-replica-2
spec:
  ports:
  - port: 9414
    protocol: TCP
    targetPort: 9414
  publishNotReadyAddresses: true
  selector:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    statefulset.kubernetes.io/pod-name: exampleqm-ibm-mq-2
  type: ClusterIP
```

5. キュー・マネージャーに使用する'ServiceAccount を作成する。

   以下の例の「ServiceAccount」は、画像がプルされたときにエンタイトルメントチェックに使用される「imagePullSecret」を参照している。 詳しくは、86 ページの『プルシークレットを作成して「Kubernetes」を使用する準備』 を参照してください。

```
kind: ServiceAccount
apiVersion: v1
metadata:
  name: exampleqm-ibm-mq
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
imagePullSecrets:
  - name: ibm-entitlement-key
```

6. StatefulSet によって管理されるキューマネージャ Pod を作成する。

   以下の YAML を例として、これらの変更を行ってください：

   - この例では、 Kubernetes 環境で使用している名前に合わせて、 mq-test から namespace の名前を変更します。
   - 使用可能なノードが 3 つ未満の Kubernetes クラスタでこの例を試す場合は、 affinity セクションを削除してください。
   - オプションでライセンスを変更する。 この例では、IBM MQ Advanced ライセンスを使用しています。IBM MQ 9.4.2 から、キューマネージャにライセンスを追加することで、IBM MQ ライセンスを持つキューマネージャがネイティブ HA とクロスリージョンレプリケーション機能を使用するように設定することもできます。

     a. spec.template.metadata の注釈を、 281 ページの『IBM MQ コンテナライセンス注釈』 の IBM MQ または IBM MQ for Non-Production Environment で指定されている値に変更する。

b. 186 ページの『IBM MQ のキューマネージャでネイティブ HA とクロスリージョンレプリケーションを設定する。 Kubernetes』 のステップ 187 ページの『3』 を完了させる。 このステップでは、ネイティブ HA およびクロスリージョンレプリケーション機能用の追加ライセンシング注釈をキューマネージャに追加する IBMLicensingDefinition CR を追加します

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  labels:
    app.kubernetes.io/component: integration
    app.kubernetes.io/instance: exampleqm
    app.kubernetes.io/name: ibm-mq
    app.kubernetes.io/version: 9.4.3.0
  name: exampleqm-ibm-mq
spec:
  persistentVolumeClaimRetentionPolicy:
    whenDeleted: Retain
    whenScaled: Retain
  updateStrategy:
    type: OnDelete
  podManagementPolicy: Parallel
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app.kubernetes.io/component: integration
      app.kubernetes.io/instance: exampleqm
      app.kubernetes.io/name: ibm-mq
  serviceName: qm
  template:
    metadata:
      # These annotations are critical for the IBM License Service to charge you correctly
      annotations:
        productChargedContainers: All
        productID: 208423bb063c43288328b1d788745b0c
        productMetric: PROCESSOR_VALUE_UNIT
        productName: IBM MQ Advanced
        productVersion: 9.4.3.0
      labels:
        app.kubernetes.io/component: integration
        app.kubernetes.io/instance: exampleqm
        app.kubernetes.io/name: ibm-mq
        app.kubernetes.io/version: 9.4.3.0
        statefulSetName: exampleqm-ibm-mq
    spec:
      # Create an anti-affinity rule, so the Native HA replicas don't end up on the same Node
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: statefulSetName
                operator: In
                values:
                - exampleqm-ibm-mq
            topologyKey: kubernetes.io/hostname
      containers:
      # Environment variables to supply to MQ container
      - env:
        - name: LICENSE
          # if you accept the license, change the following value to "accept"
          value: view
        - name: MQ_QMGR_NAME
          value: EXAMPLEQM
        - name: MQ_MULTI_INSTANCE
          value: "false"
        - name: MQ_ENABLE_METRICS
          value: "true"
        - name: MQ_ENABLE_EMBEDDED_WEB_SERVER
          value: "false"
        - name: MQ_LOGGING_CONSOLE_FORMAT
          value: basic
        - name: DEBUG
          value: "false"
        - name: MQ_ENABLE_TRACE_STRMQM
          value: "false"
        - name: MQ_EPHEMERAL_PREFIX
          value: /run/mqm
        - name: MQ_GRACE_PERIOD
```

```
          value: "29"
        - name: MQ_NATIVE_HA
          value: "true"
        # The image to pull.  This example uses a prebuilt MQ Advanced image from IBM
Container Registry
        image: cp.icr.io/cp/ibm-mqadvanced-server:9.4.3.0-r1
        imagePullPolicy: IfNotPresent
        livenessProbe:
          exec:
            command:
            - chkmqhealthy
          failureThreshold: 1
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 5
        name: qmgr
        ports:
        - containerPort: 1414
          protocol: TCP
        - containerPort: 9157
          protocol: TCP
        - containerPort: 9414
          protocol: TCP
        readinessProbe:
          exec:
            command:
            - chkmqready
          failureThreshold: 1
          periodSeconds: 5
          successThreshold: 1
          timeoutSeconds: 3
        # CPU and memory resources to allocate
        resources:
          limits:
            cpu: "1"
            memory: 1Gi
          requests:
            cpu: "1"
            memory: 1Gi
        # Container security context.  Disable all root capabilities and privileges.
        securityContext:
          allowPrivilegeEscalation: false
          capabilities:
            drop:
            - ALL
          privileged: false
          readOnlyRootFilesystem: true
          runAsNonRoot: true
        startupProbe:
          exec:
            command:
            - chkmqstarted
          failureThreshold: 24
          periodSeconds: 5
          successThreshold: 1
          timeoutSeconds: 5
        terminationMessagePath: /run/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        # Mount the main queue manager persistent volume.  Sym-linked from /var/mqm
        - mountPath: /mnt/mqm
          name: data
        # Mount space for temporary files
        - mountPath: /run
          name: scratch
        - mountPath: /tmp
          name: tmp
        # Mount the "default" key for TLS
        - mountPath: /etc/mqm/pki/keys/default
          name: default
          readOnly: true
        # Mount the key to use for Native HA replication traffic
        - mountPath: /etc/mqm/ha/pki/keys/ha
          name: ha-tls
          readOnly: true
        # Mount MQSC to be applied when the queue manager starts
        - mountPath: /etc/mqm/example.mqsc
          name: cm-example-nativeha-configmap
          readOnly: true
          subPath: example.mqsc
        # Mount an INI file to be applied when the queue manager starts
        - mountPath: /etc/mqm/example.ini
```

```
          name: cm-example-nativeha-configmap
          readOnly: true
          subPath: example.ini
      restartPolicy: Always
      securityContext: {}
      serviceAccount: exampleqm-ibm-mq
      serviceAccountName: exampleqm-ibm-mq
      automountServiceAccountToken: false
      terminationGracePeriodSeconds: 30
      volumes:
      - emptyDir:
          sizeLimit: 100M
        name: scratch
      # Temporary directory is used by default to store runmqras output, so make it larger
      - emptyDir:
          sizeLimit: 2Gi
        name: tmp
      - name: default
        secret:
          defaultMode: 288
          items:
          - key: tls.key
            path: tls.key
          - key: tls.crt
            path: tls.crt
          - key: ca.crt
            path: ca.crt
          secretName: example-qm-tls
      - name: ha-tls
        secret:
          defaultMode: 288
          secretName: example-qm-tls
      - configMap:
          defaultMode: 420
          items:
          - key: example.mqsc
            path: example.mqsc
          - key: example.ini
            path: example.ini
          name: example-nativeha-configmap
        name: cm-example-nativeha-configmap
  volumeClaimTemplates:
  - apiVersion: v1
    kind: PersistentVolumeClaim
    metadata:
      labels:
        app.kubernetes.io/component: integration
        app.kubernetes.io/instance: exampleqm
        app.kubernetes.io/name: ibm-mq
      name: data
      namespace: mq-test
    spec:
      accessModes:
      - ReadWriteOnce
      resources:
        requests:
          storage: 2Gi
      volumeMode: Filesystem
```

Secret *example-qm-tls* はステップ「177 ページの『1』」で作成され、「ConfigMap *example-nativeha-configmap* はステップ「177 ページの『2』」で作成されたことに注意。

インスタンスごとに永続ボリュームが作成される。 Kubernetes クラスタに設定されているデフォルトのストレージクラスが使用されます。 ストレージ・クラスがデフォルトとして構成されていない場合、または別のストレージ・クラスを使用したい場合は、storageClassName: *storage_class_name* を spec.volumeClaimTemplates[0].spec の下に追加します。

ネイティブ HA キュー・マネージャー内の 3 つのポッドは、ネットワークを介してデータを複製します。このリンクはデフォルトでは暗号化されていませんが、この例ではトラフィックの暗号化にキューマネージャの証明書を使用しています。 さらにセキュリティを高めるために、別の証明書を指定することもできる。 ネイティブ HA の TLS シークレットは、特定の構造を持つ「KubernetesTLS シークレットでなければならない（例えば、秘密鍵は *tls.key* という名前でなければならない）。

updateStrategy が'OnDelete に設定されていることに注意してほしい。 詳しくは、44 ページの『ネイティブ HA キュー・マネージャーの独自ローリング更新を実行する場合の考慮事項』を参照してください。

7. キューマネージャが起動していることを確認する。

   キュー・マネージャーがデプロイされます。 キュー・マネージャのポッドをリストアップするなどして、先に進む前に「Running 状態であることを確認してください：

   ```
   kubectl get pods --selector app.kubernetes.io/instance=exampleqm
   ```

   理想的には、キュー・マネージャーに接続して、それが実行されていることを確認することだ。 しかし、「Kubernetes クラスタの外からキュー・マネージャに接続することは、「Kubernetes ディストリビューションとクラウド環境の間で大きく異なる可能性があるため、これらの命令の範囲外である。
8. アクティブ・ポッドの障害の強制試行

   キュー・マネージャーの自動復旧を検証するには、ポッドの障害をシミュレートします。

   a) アクティブ・ポッドとスタンバイ・ポッドの表示

   前のコマンドを再実行する：

   ```
   kubectl get pods --selector app.kubernetes.io/instance=exampleqm
   ```

   READY フィールドでは、アクティブ・ポッドは値 1/1 を返し、レプリカ・ポッドは値 0/1 を返すことに注意してください。

   b) アクティブ・ポッドの削除

   アクティブ・ポッドの絶対パス名を指定して次コマンドを実行します。

   ```
   kubectl delete pod exampleqm-ibm-mq-<active_pod_value>
   ```

   c) ポッドの状況の再表示

   以下のコマンドを実行します。

   ```
   kubectl get pods --selector app.kubernetes.io/instance=exampleqm
   ```

   d) キュー・マネージャーの状況の表示

   他のいずれかのポッドの絶対パス名を指定して次のコマンドを実行します。

   ```
   kubectl exec -t <qmgr_pod_name> -- dspmq -o nativeha -x -m EXAMPLEQM
   ```

   アクティブ・インスタンスが変更されたことを示す状況が表示されるはずです。以下に例を示します。

   ```
   QMNAME(EXAMPLEQM) ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3)
   INSTANCE(inst1) ROLE(Active) REPLADDR(9.20.123.45) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
   CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
   INSTANCE(inst2) ROLE(Replica) REPLADDR(9.20.123.46) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
   CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
   INSTANCE(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(Yes) BACKLOG(0)
   CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
   ```

タスクの結果

ネイティブの高可用性と相互 TLS 認証を備えたキュー・マネージャのデプロイに成功し、アクティブなポッドに障害が発生したときに自動的に回復することを確認しました。

V 9.4.2 例：ネイティブ HA クロスリージョンレプリケーションの設定 Kubernetes

この例では、ネイティブの高可用性クロスリージョンレプリケーション（Native HA CRR）機能を使用して、 Kubernetes にキューマネージャーを直接 YAML リソースを使用してデプロイします。

始める前に

この例を完了するには、まず以下の前提条件を満たしておく必要があります。

- ライブグループとリカバリグループが展開される各クラスタに、 Kubernetes の名前空間を作成します。
- コマンドラインから、 Kubernetes クラスタのいずれかにログインし、関連するネームスペースに切り替えることができる。
- 上記の名前空間で、 86 ページの『プルシークレットを作成して「Kubernetes」を使用する準備』 の手順に従っていることを確認してください。

このタスクについて

この例では、ネイティブ HA CRR 機能を使用して、 Kubernetes に 2 つのキューマネージャーを直接 YAML リソースを使用して展開します。 Native HA キューマネージャーのうち、1 つは「ライブ」、もう 1 つは「リカバリ」となります。 この例は、 177 ページの『例 Kubernetes ネイティブ HA の設定』 で作成された YAML ファイルを基にしています。

この例では、 NativeHA: ConfigMap, サービス、StatefulSet,、および Route で定義されたものに加えて必要な変更を示しています。

手順

1. 新規のコンフィグマップを作成するか、既存の ConfigMap 内に新規の.ini データ定義を定義して（ネイティブ HA の例の手順 2 を参照)、ライブキューマネージャまたはリカバリキューマネージャの INI ファイルを定義します。

   Kubernetes ConfigMap を作成し、以下のグループを設定するための定義を含めます

   a) Recovery としてネイティブの高可用性グループ：

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: example-nativeha-configmap
   data:
   ...
     example-crr: |
       NativeHALocalInstance:
         GroupName=beta
         GroupRole=Recovery
         GroupLocalAddress=(9415)
         GroupCipherSpec=ANY_TLS12_OR_HIGHER
   ```

   リカバリグループが展開されると、そのグループのレプリケーションアドレスを決定でき、ライブグループの ConfigMap に提供できます。 例えば、デプロイ済みのリカバリクラスタに次のコマンドを発行することで、この設定を行うことができます

   ```
   kubectl get route example-ibm-mq-crr-route -n <recovery-qm-namespace> -o
   jsonpath="{.spec.host}"
   ```

   b) ネイティブ HA CRR グループをライブ（リカバリグループを指す）として：

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: example-nativeha-configmap
   data:
   ...
     example-crr: |
       NativeHALocalInstance:
         GroupName=alpha
         GroupRole=Live
         GroupLocalAddress=(9415)
         GroupCipherSpec=ANY_TLS12_OR_HIGHER
       NativeHARecoveryGroup:
         Enabled=Yes
         GroupName=beta
         ReplicationAddress=<address.to.recovery.cluster>(443)
   ```

2. ネイティブ HA CRR トラフィックが CRR グループに流入するための追加のサービス ClusterIP ポートを開放します。 これは、新しい YAML 定義を作成するか、ネイティブ HA の例のステップ 3 のサービス定義を拡張することで行うことができます

```
apiVersion: v1
kind: Service
metadata:
...
  name: exampleqm-ibm-mq
spec:
  ports:
...
  - name: ha-crr
    port: 9415
    protocol: TCP
    targetPort: 9415
...
  selector:
...
  type: ClusterIP
```

3. クラスタへの新しいルートを定義し、他のネイティブ HA CRR グループからこのネイティブ HA CRR グループへの通信を可能にします。また、手順 2 で定義した新しいサービス ClusterIP ポートへの TLS「パススルー」を有効にします

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: "exampleqm-ibm-mq-crr-route"
spec:
  to:
    kind: Service
    name: exampleqm-ibm-mq
  port:
    targetPort: 9415
  tls:
    termination: passthrough
```

4. Native HA の例の手順 6 で作成した StatefulSet を修正し、以下の要件が満たされていることを確認してください

   a) ネイティブ HA CRR 用の新しい ConfigMap INI 定義は、キューマネージャが起動時にそれを認識できるようにマウントされています

```
...
        volumeMounts:
...
        # Mount the NativeHA CRR INI file to be applied when the queue manager starts
        - mountPath: /etc/mqm/example-crr.ini
          name: cm-example-nativeha-configmap
          readOnly: true
          subPath: example-crr.ini
...
      volumes:
...
      - configMap:
          defaultMode: 420
          items:
...
          - key: example-crr.ini
            path: example-crr.ini
          name: example-nativeha-configmap
        name: cm-example-nativeha-configmap
```

   b) 2 つの Native HA CRR グループ間の通信は暗号化されなければなりません。 したがって、現在の NativeHA CRR グループの TLS 秘密鍵の宣言と、他の NativeHA CRR グループの信頼証明書は、定義され、 `/etc/mqm/groupha` ディレクトリにマウントする必要があります。

   グループ名「アルファ」（Live 用）と「ベータ」（Recovery 用）を使用すると、YAML の StatefulSet の Live グループに対する修正例は次のようになります

```
...
        volumeMounts:
...
          - mountPath: /etc/mqm/groupha/pki/keys/ha-group
            name: groupha-tls
            readOnly: true
          - name: groupha-tls-trust-0
            mountPath: /etc/mqm/groupha/pki/trust/0
```

```
            readOnly: true
...
      volumes:
...
      - name: groupha-tls
        secret:
          secretName: nha-crr-secret-alpha
          defaultMode: 288
      - name: groupha-tls-trust-0
        secret:
          secretName: nha-crr-secret-beta
          defaultMode: 420
          items:
            - key: tls.crt
              path: native-crr-nha-crr-secret-beta-0-tls.crt
```

5. ライブおよびリカバリキューマネージャーインスタンスの Ready and Running ポッドで、次の **dspmq** コマンドを使用して、ネイティブ HA CRR グループが実行中であり、同期され、接続され、定足数に達していることを確認します

```
kubectl exec -t <qmgr_pod_name> -- dspmq -o nativeha -g -m EXAMPLEQM
```

ステータスには、2 つの Native HA CRR グループが接続され、ライブグループとリカバリグループとして動作していることが表示されるはずです。例えば：

```
QMNAME(EXAMPLEQM) ROLE(Active) INSTANCE(testmqqm-ibm-mq-0) INSYNC(yes) QUORUM(3/3)
GRPLSN(<0:0:10:30665>) GRPNAME(alpha) GRPROLE(Live)
 GRPNAME(alpha) GRPROLE(Live) GRPADDR(Unknown) GRPVER(9.4.2.0) GRSTATUS(Normal)
RCOVLSN(<0:0:10:30665>) RCOVTIME(2025-02-17T16:25:16.527089Z) INITLSN(<0:0:9:4684>)
INITTIME(2025-02-17T16:25:01.226650Z) LIVETIME(2025-02-17T16:25:02.974677Z)
ALTDATE(2025-02-17) ALTTIME(16.25.19)
 GRPNAME(beta) GRPROLE(Recovery) GRPADDR(<address.to.recovery.cluster>) GRPVER(9.4.2.0)
CONNGRP(yes) GRSTATUS(Normal) RCOVLSN(<0:0:10:30665>) RCOVTIME(2025-02-17T16:25:16.527089Z)
BACKLOG(0) INSYNC(yes) SYNCTIME(2025-02-17T16:29:38.643916Z) ALTDATE(2025-02-17)
ALTTIME(16.29.29)
```

タスクの結果

ネイティブのハイアベイラビリティと TLS 認証を備えた 2 つのキューマネージャグループを、グループ内およびグループ間で正常に展開しました。 また、ライブグループとリカバリグループが接続され、クロスリージョンレプリケーションを実行していることも確認しました。

V 9.4.2 Kubernetes IBM MQ のキューマネージャでネイティブ HA とクロスリージョンレプリケーションを設定する。 Kubernetes

IBM MQ 9.4.2, IBM MQ から、キューマネージャにライセンスを追加することで、ネイティブ HA クロスリージョンレプリケーション（CRR）機能を使用するようにキューマネージャを構成することができます。

始める前に

注 : これらの手順は、キュー・マネージャーを Kubernetes に配備する場合にのみ適用されます。 IBM MQ Operator、代わりに 116 ページの『IBM MQ 、キューマネージャのネイティブ HA とクロスリージョンレプリケーションを設定するには、次のようにします。 IBM MQ Operator』 。

- IBM MQ ライセンスのキューマネージャでネイティブ HA CRR 機能を使用するには、 IBM License Service が必要です。 詳細については、 IBM Cloud Pak foundational services ドキュメントの Kubernetes クラスタへの License Service のインストールを参照してください。
- キューマネージャポッドを管理する `StatefulSet` では、`metadata.labels.app.kubernetes.io/instance` ラベルを一意な値に設定する必要があります。

手順

1. コマンドラインで Kubernetes クラスタにログインし、キューマネージャが配置されているネームスペースに切り替えます。
2. 関連するアノテーションをキュー・マネージャーのアノテーションに追加します。

この手順の説明は、既存のキュー・マネージャを編集するためのものであることに注意してください。新しいキュー・マネージャの場合は、作成時にアノテーションを追加します。

重要 : キュー・マネージャのアノテーションを更新するには、ポッドの再起動が必要です。

a) 更新したいキュー・マネージャの StatefulSet ：

```
kubectl get statefulset
```

b) キュー・マネージャーの StatefulSet ：

```
kubectl edit statefulset <QUEUE-MANAGER-STATEFUL-SET>
```

c) 課金対象コンポーネントに関連する注釈を StatefulSet に追加する。spec.template.metadata の下に他の注釈があることに注意してください：

- IBM MQ の場合:

```
spec:
  template:
    metadata:
      productID: c661609261d5471fb4ff8970a36bccea
      productName: IBM MQ
```

- IBM MQ for Non-Production Environment の場合:

```
spec:
  template:
    metadata:
      productID: 151bec68564a4a47a14e6fa99266deff
      productName: IBM MQ for Non-Production Environment
```

d) 変更内容を保存します。 コマンドラインには以下のようなメッセージが表示される：

```
statefulset.apps/<QUEUE-MANAGER-STATEFUL-SET> edited
```

e) 編集した StatefulSet によって管理されているすべてのキュー・マネージャ・ポッドを削除します。

f) 必要に応じて、他のキューマネージャについてもこれらの手順を繰り返してください。

3. Native HA CRR 機能のために、キューマネージャに追加のライセンス注釈を追加する関連 IBMLicensingDefinition CR を追加します。

a) IBMLicensingDefinition YAML ファイルを作成する。

以下の例では、これらの値を変更する：

- <IBMLicensingDefinition-UNIQUE-NAME>: IBMLicensingDefinition CR のユニークな名前。 良い習慣としては、この名前にキュー・マネージャーの名前を含めることです。
- <QUEUE-MANAGER-NAMESPACE>:キュー・マネージャが配置される名前空間。
- <QUEUE-MANAGER-NAME>:キュー・マネージャの metadata.labels.app.kubernetes.io/instance の値。
- 本番用の IBMLicensingDefinition CR の YAML の例：

```
apiVersion: operator.ibm.com/v1
kind: IBMLicensingDefinition
metadata:
  name: <IBMLicensingDefinition-UNIQUE-NAME>
  namespace: <QUEUE-MANAGER-NAMESPACE>
spec:
  action: cloneModify
  condition:
    metadata:
      labels:
        app.kubernetes.io/instance: <QUEUE-MANAGER-NAME>
  scope: cluster
  set:
```

```
    productID: 46a57a4a66ee4b904e9b5544b53a2ba2
    productName: IBM MQ Native HA and Cross-Region Replication Add-on
```

- IBMLicensingDefinition CR の YAML 例(非生産用)

```
apiVersion: operator.ibm.com/v1
kind: IBMLicensingDefinition
metadata:
  name: <IBMLicensingDefinition-UNIQUE-NAME>
  namespace: <QUEUE-MANAGER-NAMESPACE>
spec:
  action: cloneModify
  condition:
    metadata:
      labels:
        app.kubernetes.io/instance: <QUEUE-MANAGER-NAME>
  scope: cluster
  set:
    productID: 525b1331a74d4a209d9f6e9c908e374e
    productName: IBM MQ Native HA and Cross-Region Replication Add-on for Non-
Production Env
```

b) IBMLicensingDefinition YAML ファイルを適用する：

```
kubectl apply -f <IBMLicensingDefinition-UNIQUE-NAME>.yaml
```

c) 必要に応じて、他のキューマネージャについてもこれらの手順を繰り返してください。

Kubernetes サンプルの Helm チャートを使用したキューマネージャの展開と設定

サンプルの Helm チャートを使用して、 Kubernetes にキュー・マネージャーをデプロイして構成することができます。

このタスクについて

Red Hat OpenShift Container Platform を使用していない場合、 IBM MQ Operator はサポートされません。サンプルの Helm チャートを使用して、他のタイプの Kubernetes クラスターにデプロイできます。

手順

- Helm 独自の 'IBM MQ コンテナイメージをデプロイする方法については、'サンプル 'IBM MQ'Helm チャート を参照のこと。

関連資料

28 ページの『コンテナーでの IBM MQ のサポート』
コンテナーでは、すべての IBM MQ 機能が同じ方法で使用可能およびサポートされているわけではありません。

Podman Podman にキュー・マネージャーをデプロイする。

この例では、ビルド済みの'IBM MQ Advanced Container イメージを実行し、キュー・マネージャーを'Podman に配備する。 同じ例は 'Docker でも使える。

始める前に

この例の最後にある TLS 設定を検証する：

- IBM MQ クライアントをインストールします。 メッセージの送受信をテストするには、'**amqsputc** と'**amqsgetc** コマンドが必要です。 これらのコマンドは「IBM MQ クライアントの一部としてインストールできる。
  - Windows と'Linux、'https://ibm.biz/mq94redistclients からお使いのオペレーティング・シ ステム用の'IBM MQ 再配布可能クライアントをインストールしてください。
  - Mac」は「https://developer.ibm.com/tutorials/mq-macos-dev/」から「IBM MQ MacOS Toolkit ダウンロードして設定する。

- ARM64 IBM MQ Advanced Container イメージはあらかじめ組み込まれていない。

この例では'Podman 使用しているが、代わりに'Docker を使用することもできる。

このタスクについて

この例では、ビルド済みの'IBM MQ Advanced Container イメージを実行し、キュー・マネージャーを'Podman に配備する。 メッセージング用 TLS の設定に必要な追加手順、および「IBM MQ Console」と「REST API 設定について詳しく説明する。 ニーズに合わせて不要なステップをスキップすることで、これらのうちどれを設定するかを選択できる。

キューマネージャが動作している場合、この例では、「IBM MQ サンプルプログラムを「Podman 外のマシン、例えばラップトップ上で動作させ、サンプルクライアントとキューマネージャの間で相互 TLS を使用してメッセージを送受信することを検証しています。 また、「IBM MQ Console」にログインし、「REST API 管理およびメッセージングすることも検証する。

注 : 代わりに、デフォルトの開発者設定をいくつか追加した、ビルド済みの「IBM MQ Advanced for Developers コンテナイメージを実行することができる。 291 ページの『IBM MQ Advanced for Developers コンテナー・イメージ』を参照してください。

手順

- **クライアントとキュー・マネージャ間のメッセージングに TLS を設定する。**
  a) 104 ページの『OpenSSL を使用した自己署名 PKI の作成』説明に従って証明書を作成し、すべての手順に従うが、「**kubectl** コマンドはスキップする。

     a. キューマネージャ用に生成された鍵と証明書のファイルを含む、例えば 'qm-pki というディレクトリを作成します：

        i) example-qm.key

        ii) example-qm.crt

        iii) ca.crt

        コンテナを実行しているユーザー（uid 1001）は、'example-qm.key への読み取りアクセスが必要である。 デフォルトでは、'OpenSSL は、キーを作成したユーザーがそのキーにアクセスできるように作成する。

     b. 例えば「mq-client」というディレクトリを作成し、クライアント用に生成されたファイルを格納する：

        i) example-app1.p12

        ii) Mac には'example-app1-chain.crt も必要である。

  b) 例えば '**example-tls.mqsc** ように、新しいキューと 'SVRCONN チャネルを作成し、 チャネルへのアクセスを許可するチャネル認証レコードを追加する MQSC コマンドを含むファイルを作成する。

```
DEFINE CHANNEL('MTLS.SVRCONN') CHLTYPE(SVRCONN) SSLCAUTH(REQUIRED)
SSLCIPH('ANY_TLS13_OR_HIGHER') REPLACE
SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=*') USERSRC(NOACCESS)
ACTION(REPLACE)
SET CHLAUTH('MTLS.SVRCONN') TYPE(SSLPEERMAP) SSLPEER('CN=example-app1') USERSRC(MAP)
MCAUSER('app1') ACTION(REPLACE)
SET AUTHREC PRINCIPAL('app1') OBJTYPE(QMGR) AUTHADD(CONNECT,INQ)
DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
SET AUTHREC PROFILE('EXAMPLE.QUEUE') PRINCIPAL('app1') OBJTYPE(QUEUE)
AUTHADD(BROWSE,PUT,GET,INQ)
```

  MQSC コマンドは、「MTLS.SVRCONN チャネルと「EXAMPLE.QUEUE」というキューを定義する。 チャネルは、「common name」が「example-app1 である証明書を提示するクライアントにのみアクセスを許可するように設定されている。 これは、前のステップで作成した証明書の 1 つに使用されているコモン・ネームである。 このコモンネームを持つこのチャネルの接続は、キューマネージャへの 接続とキュー例へのアクセスを許可されたユーザ ID である 'app1 にマップされる。

c) app1 ユーザー ID が外部ユーザーレジストリに存在する必要がなく、この構成では名前としてのみ存在するようなセキュリティポリシーを有効にする INI ファイル、例えば「example-tls.ini」を作成する。

```
Service:
    Name=AuthorizationService
    EntryPoints=14
    SecurityPolicy=UserExternal
```

- **IBM MQ Console」と「REST API 設定する。**

  この例では、'mqwebuser.xml ファイル内に基本的なレジストリを設定する。 XML ファイル中のユーザー名、パスワード、ロールは、'IBM MQ Console と'REST API ユーザーの認証と認可に使用される。 詳細については、' IBM MQ Console と 'REST API の基本レジストリの設定を参照のこと。

  基本的なレジストリを使用することが本番環境に適しているかどうかを検討する必要がある。 IBM MQ Console」と「REST API セキュリティ参照。

  a) Create a file called mqwebuser.xml, to configure the IBM MQ Console.

  この例では、web admin と web user のロールを持つユーザーを含む基本レジストリが作成されます。

  ADMIN_PASSWORD と *APP_PASSWORD* を独自の値に置き換える**必要があります**。 管理者ユーザー名とアプリ・ユーザー名も変更できますが、アプリ・ユーザー名は前の MQSC コマンドで指定したアプリ・ユーザー名と一致させる必要があります。 この例では、ユーザー名がそれぞれ 2 回使われていることに注意。

```
<?xml version="1.0" encoding="UTF-8"?>
<server>
    <featureManager>
    <feature>appSecurity-2.0</feature>
    <feature>basicAuthenticationMQ-1.0</feature>
    </featureManager>
    <enterpriseApplication id="com.ibm.mq.console">
        <application-bnd>
            <security-role name="MQWebAdmin">
                <group name="MQWebAdminGroup" realm="defaultRealm"/>
            </security-role>
        </application-bnd>
    </enterpriseApplication>
    <enterpriseApplication id="com.ibm.mq.rest">
        <application-bnd>
            <security-role name="MQWebAdmin">
                <group name="MQWebAdminGroup" realm="defaultRealm"/>
            </security-role>
            <security-role name="MQWebUser">
                <group name="MQWebMessaging" realm="defaultRealm"/>
            </security-role>
        </application-bnd>
    </enterpriseApplication>
    <basicRegistry id="basic" realm="defaultRealm">
        <user name="admin" password="ADMIN_PASSWORD"/>
        <user name="app1" password="APP_PASSWORD"/>
        <group name="MQWebAdminGroup">
            <member name="admin"/>
        </group>
        <group name="MQWebMessaging">
            <member name="app1"/>
        </group>
    </basicRegistry>
    <sslDefault sslRef="mqDefaultSSLConfig"/>
</server>
```

  b) コンテナ・レジストリにログインする。

  エンタイトルメント・キーを取得し、それを使ってコンテナ・レジストリにログインする。 87 ページの『Podman または「Docker 使用するために、ビルド済みのコンテナイメージを取り込む』 を参照してください。

  c) Podman でキュー・マネージャーを実行する。

  以下のいずれかのオプションを実行します。

– **オプション 1: キュー・マネージャ証明書を含むディレクトリをマウントして、'Podman でキュー・マネージャを実行する。**

以下のコマンドには、TLS メッセージングのコンフィギュレーションと、'IBM MQ Console および 'REST API コンフィギュレーションが含まれている。 先に説明したコンフィギュレーションを、あなたのニーズには必要ないという理由でスキップした場合は、スキップしたセクションに関連するパラメータを削除してください。

コマンドが、先に作成したファイルとディレクトリの正しい場所を指していることを確認してください。

Podman でキュー・マネージャーを実行するには、以下のコマンドを実行する：

```
podman run --env MQ_ENABLE_EMBEDDED_WEB_SERVER=true  \
-v "./mqwebuser.xml:/etc/mqm/web/installations/Installation1/servers/mqweb/
mqwebuser.xml" \
-v "./example-tls.ini:/etc/mqm/example-tls.ini"  \
-v "./example-tls.mqsc:/etc/mqm/example-tls.mqsc"  \
-v "./qm-pki:/etc/mqm/pki/keys/qmlabel" --env LICENSE=accept --env MQ_QMGR_NAME=QM1 \
--publish 1414:1414 --publish 9443:9443 --detach --name QM1 cp.icr.io/cp/ibm-
mqadvanced-server:9.4.1.0-r1
```

コマンドは「IBM MQ Advanced キュー・マネージャー・イメージを取り出し、コンテナを起動する。

– **オプション 2：'Podman のキュー・マネージャーを'Podman シークレットを使って実行する。**

キュー・マネージャ証明書をコンテナに直接マウントするのではなく、'Podman シークレットを使用してマウントすることができる。

注 : Docker シークレットは'Docker スウォームでしか使用できないため、'Docker スウォームを使用しない限り、'Docker 使用する場合はこのオプションは適用されない。 dockerdocs の'ドッカーシークレット作成コマンドを参照。

以下のステップを実行します。

a. キューマネージャの TLS ファイルそれぞれに 'Podman シークレットを作成する。 例えば、「qm-pki ディレクトリを作成する：

```
podman secret create example-qm.key ./example-qm.key
podman secret create example-qm.crt ./example-qm.crt
podman secret create ca.crt ./ca.crt
```

以下のコマンドには、TLS メッセージングのコンフィギュレーションと、'IBM MQ Console および 'REST API コンフィギュレーションが含まれている。 先に説明したコンフィギュレーションを、あなたのニーズには必要ないという理由でスキップした場合は、スキップしたセクションに関連するパラメータを削除してください。

コマンドが、先に作成したファイルとディレクトリの正しい場所を指していることを確認してください。

Podman」でキュー・マネージャーを実行するには、以下のコマンドを実行する：

```
podman run --env MQ_ENABLE_EMBEDDED_WEB_SERVER=true \
-v "./mqwebuser.xml:/etc/mqm/web/installations/Installation1/servers/mqweb/
mqwebuser.xml" \
-v "./example-tls.ini:/etc/mqm/example-tls.ini" \
-v "./example-tls.mqsc:/etc/mqm/example-tls.mqsc" \
--secret example-qm.key,target=/etc/mqm/pki/keys/qmlabel/example-
qm.key,uid=1001,mode=0440 \
--secret example-qm.crt,target=/etc/mqm/pki/keys/qmlabel/example-qm.crt,uid=1001 \
--secret ca.crt,target=/etc/mqm/pki/keys/qmlabel/ca.crt,uid=1001 --env
LICENSE=accept --env MQ_QMGR_NAME=QM1 \
--publish 1414:1414 --publish 9443:9443 --detach --name QM1 cp.icr.io/cp/ibm-
mqadvanced-server:9.4.1.0-r1
```

注 : コンテナが実行されているユーザが別のユーザにオーバーライドされている場合は、UID を変更して一致させる。

キュー・マネージャーは「Podman」で実行されている。

```
podman ps
```

妥当性検査

- **TLS によるメッセージングの検証**

  a) 上記で作成したクライアントディレクトリに、例えば'mq-client ように、'mqclient.ini という名前のクライアント INI ファイルを作成する。 クライアントとキュー・マネージャ間のメッセージング用 TLS の構成」の手順 1 で p12 キーストアに使用した *PASSWORD* を指定します：

```
Channels:
  ChannelDefinitionDirectory=.
  ChannelDefinitionFile=CCDT.json
SSL:
  OutboundSNI=HOSTNAME
  SSLKeyRepository=example-app1.p12
  SSLKeyRepositoryPassword=PASSWORD
```

  b) 同じディレクトリに、先に定義した'mqclient.ini ファイルの参照と同じ名前の CCDT ファイル、例えば'CCDT.json を作成する：

```
{
    "channel":
    [
        {
            "name": "MTLS.SVRCONN",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "localhost",
                        "port": 1414
                    }
                ],
                "queueManager": "QM1"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ANY_TLS13",
              "certificateLabel": "example-app1"
            },
            "type": "clientConnection"
        }
   ]
}
```

  c) Mac では、 以下のコマンドを実行します。

```
export MQSSLTRUSTSTORE=example-app1-chain.crt
```

  d) 同じディレクトリ、例えば'mq-client から、サンプル・アプリケーションを使ってメッセージを入れる：

```
amqsputc EXAMPLE.QUEUE QM1
```

  e) サンプル・アプリケーションを使ってメッセージを取得してください：

```
amqsgetc EXAMPLE.QUEUE QM1
```

- **IBM MQ Console を検証する**

  ブラウザでコンソールに接続する：

  a) オープン https://localhost:9443/ibmmq/console

  b) 自己署名証明書を受け入れます。

  c) ログインします。

  – ユーザー: admin

  – パスワード：以前に *ADMIN_PASSWORD* に指定した値。

- **REST API を検証する**

  以下の例では、*ADMIN_PASSWORD* と *APP_PASSWORD* に以前に設定した値を指定する。

  キュー・マネージャーのステータスを取得するために'administrative REST API 呼び出す例：

  ```
  curl -k -X GET -u admin:ADMIN_PASSWORD https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1
  ```

  メッセージング 'REST API を呼び出してキューにメッセージを入れ、それからキューからメッセージを取得する例。

  メッセージを書き込みます。

  ```
  curl -k -X POST -u app1:APP_PASSWORD \
  -H 'Content-Type: application/json' -H "ibm-mq-rest-csrf-token: value" \
  -d 'msg from REST' https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/
  EXAMPLE.QUEUE/message
  ```

  メッセージを閲覧する

  ```
  curl -k -X GET -u app1:APP_PASSWORD \
  -H "ibm-mq-rest-csrf-token: value" https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/
  queue/EXAMPLE.QUEUE/message
  ```

  注 : REST API'**GET** 操作は、実際には'IBM MQ ブラウズ操作である。 IBM MQ'**GET** を行うには、'REST API'**DELETE** 動詞を使う。

トラブルシューティング

- **amqputc** または'**amqgetc** サンプルアプリケーションが失敗した場合、クライアントマシン（例えばラップトップ）の'~/IBM/MQ/data/errors/AMQERR01.LOG でエラーログを見る。
- Podman のコンテナ上で exec して、キューマネージャのエラーログを通常の場所に表示する。

  以下に例を示します。

  ```
  podman exec -it QM1 /bin/bash
  ```

OpenShift CP4I-SC2 CD IBM MQ Operator へのマイグレーション

この一連のトピックでは、Red Hat OpenShift Container Platform の IBM MQ Operator を使用して、既存の IBM MQ キュー・マネージャーをコンテナー環境にマイグレーションするための主要なステップについて説明します。

このタスクについて

IBM MQ on Red Hat OpenShift で展開するクライアントは、以下のシナリオに分けることができる。

1. 新規アプリケーションのための新しい IBM MQ デプロイメントを Red Hat OpenShift に作成する。
2. Red Hat OpenShift 内の新規アプリケーションのために IBM MQ ネットワークを Red Hat OpenShift に拡張します。
3. 既存のアプリケーションを引き続きサポートするには、 IBM MQ デプロイメントを Red Hat OpenShift に移動します。

ご使用の IBM MQ 構成をマイグレーションする必要があるのは、シナリオ 3 の場合のみです。 その他のシナリオは、新規デプロイメントと見なされます。

この一連のトピックでは、シナリオ 3 に焦点を当て、 IBM MQ Operator を使用して既存の IBM MQ キュー・マネージャーをコンテナー環境にマイグレーションするための重要なステップについて説明します。 IBM MQ は柔軟性と拡張性に優れているため、オプションで行える手順がいくつかあります。 これらの各手順に「これを行う必要がありますか?」セクションがあります。 社内でどのようなニーズがあるかを確認しておくなら、マイグレーション時に時間を節約することができるはずです。

どのデータをマイグレーションするか検討することも必要です。

1. 同じ構成で IBM MQ をマイグレーションするが、キューに入っている既存のメッセージは含めない。

2. 同じ構成で IBM MQ をマイグレーションし、既存のメッセージも含める。

標準的なバージョン間マイグレーションでは、どちらのアプローチでも使用できます。 IBM MQ キュー・マネージャーの標準的なマイグレーションでは、マイグレーション時にキューにメッセージが保管されていたとしてもわずかなので、多くの場合はオプション 1 が適切です。 コンテナー・プラットフォームへのマイグレーションでは、それにも増してオプション 1 を使用することが一般的になっています。これは、マイグレーションの複雑な手順をなくして、ブルー・グリーン・デプロイメントを行えるようにするためです。 そのため、このシナリオに焦点を当てて説明します。

このシナリオの目的は、既存のキュー・マネージャーの定義と一致するキュー・マネージャーをコンテナー環境内に作成することです。 この方法では、ネットワークに接続された既存のアプリケーションは、新しいキュー・マネージャーを指すように再構成するだけで済みますので、他の構成やアプリケーション・ロジックを変更する必要はありません。

このマイグレーション全体を通して、新しいキュー・マネージャーに適用される複数の構成ファイルを生成します。 これらのファイルの管理を簡素化するために、ディレクトリーを 1 つ作成し、生成したファイルはそのディレクトリーに入れるようにしてください。

手順

1. 194 ページの『必要な機能を利用できることの確認』
2. 195 ページの『キュー・マネージャー構成の抽出』
3. オプション: 195 ページの『オプション: キュー・マネージャーの鍵と証明書の抽出および取得』
4. オプション: 197 ページの『オプション: LDAP の構成』
5. オプション: 205 ページの『オプション: IBM MQ 構成内の IP アドレスとホスト名の変更』
6. 206 ページの『コンテナー環境用のキュー・マネージャー構成の更新』
7. 209 ページの『コンテナーで実行されている IBM MQ のためのターゲット HA アーキテクチャーの選択』
8. 210 ページの『キュー・マネージャー用のリソースの作成』
9. 211 ページの『Red Hat OpenShift での新しいキュー・マネージャーの作成』
10. 215 ページの『新規コンテナー・デプロイメントの検証』

OpenShift CP4I-SC2 CD 必要な機能を利用できることの確認

IBM MQ Operator には、IBM MQ Advanced 内で使用可能な機能がすべて含まれているわけではないので、除外されている機能が必要ないことを確認する必要があります。 その他の機能は部分的にサポートされており、コンテナー内で使用可能なものと一致するように再構成することもできます。

始める前に

これは、193 ページの『IBM MQ Operator へのマイグレーション』の最初のステップです。

手順

1. 必要なすべての機能がターゲット・コンテナー・イメージに含まれていることを確認します。

   最新情報については、28 ページの『コンテナー内の IBM MQ の使用方法の選択』を参照してください。
2. IBM MQ Operator には、リスナーと呼ばれる IBM MQ トラフィック・ポートが 1 つあります。 複数のリスナーがある場合は、これを単純化して、コンテナーで 1 つのリスナーを使用するようにします。 これは一般的なシナリオではないため、この変更についての詳細な説明は行いません。
3. IBM MQ 出口が使用されている場合は、IBM MQ 出口バイナリー内で階層化することにより、それらをコンテナーにマイグレーションします。 これは上級のマイグレーション・シナリオであるため、ここには記載しません。 手順の概要については、159 ページの『Red Hat OpenShift CLI を使用した、カスタム MQSC および INI ファイルを使用したイメージの作成』を参照してください。
4. IBM MQ システムに高可用性が設定されている場合は、使用可能なオプションを確認します。

   40 ページの『コンテナー内の IBM MQ の高可用性の計画』 を参照してください。

次のタスク
これで、キュー・マネージャー構成を抽出する準備ができました。

OpenShift CP4I-SC2 CD キュー・マネージャー構成の抽出

構成の大部分は、キュー・マネージャー間で移植可能です。 例えば、アプリケーションが対話する内容 (キュー、トピック、およびチャネルの定義など) です。 既存の IBM MQ キュー・マネージャーから構成を抽出するには、このタスクを使用します。

始める前に
このタスクでは、必要な機能が使用可能であることを確認済みであることを前提としています。

手順
1. 既存の IBM MQ インストール済み環境があるマシンにログインします。
2. 構成のバックアップをとります。

   以下のコマンドを実行します。

   ```
   dmpmqcfg -m QMGR_NAME > /tmp/backup.mqsc
   ```

   このコマンドの使用上の注意:

   - このコマンドは、バックアップを tmp ディレクトリーに保管します。 別の場所にバックアップを保管することもできますが、このシナリオの以下のコマンドでは、tmp ディレクトリーを使用することを想定しています。
   - *QMGR_NAME* は、ご使用の環境のキュー・マネージャー名に置き換えてください。 値が分からない場合は、**dspmq** コマンドを実行して、このマシンで使用可能なキュー・マネージャーを表示します。 ここでは、qm1 という名前のキュー・マネージャーの **dspmq** コマンド出力例を示します。

     ```
     QMNAME(qm1)                   STATUS(Running)
     ```

     dspmq コマンドを実行するには、IBM MQ キュー・マネージャーが開始している必要があります。開始していない場合は、次のエラーを受け取ります。

     ```
     AMQ8146E: IBM MQ queue manager not available.
     ```

     必要に応じて、次のコマンドを実行してキュー・マネージャーを開始します。

     ```
     strmqm QMGR_NAME
     ```

次のタスク
これで、キュー・マネージャーの鍵と証明書を抽出して取得する準備ができました。

OpenShift CP4I-SC2 CD オプション: キュー・マネージャーの鍵と証明書の抽出および取得

IBM MQ は、TLS を使用してキュー・マネージャーへのネットワーク・トラフィックを暗号化するように構成できます。 このタスクを使用して、キュー・マネージャーが TLS を使用していることを確認し、鍵と証明書を抽出し、マイグレーションされたキュー・マネージャーで TLS を構成します。

始める前に
このタスクでは、キュー・マネージャー構成を抽出済みであることを前提としています。

このタスクについて
これを行う必要がありますか?

キュー・マネージャーへのトラフィックを暗号化するように IBM MQ を構成することができます。 この暗号化は、キュー・マネージャーで構成されている鍵リポジトリーを使用して実行されます。 IBM MQ チャネルはそれを使って TLS 通信を有効にします。 ご使用の環境で TLS 通信が構成されているかどうかが不明な場合は、以下のコマンドを実行して確認してください。

```
grep 'SECCOMM(ALL\|SECCOMM(ANON\|SSLCIPH' backup.mqsc
```

結果が表示されない場合、TLS は使用されていません。 しかし、このことはマイグレーション済みのキュー・マネージャーで TLS を構成できないことを意味するものではありません。 以下の状況では、この状態を変更することが必要となる場合があります。

- Red Hat OpenShift 環境に対するセキュリティー・アプローチは、前の環境と比較して拡張する必要があります。
- Red Hat OpenShift 環境の外部からマイグレーション済みキュー・マネージャーにアクセスする必要がある場合は、Red Hat OpenShift ルートを通過するために TLS が必要です。

注 : 発行者 (CA) 証明書と同じサブジェクト識別名 (DN) を持つキュー・マネージャー証明書はサポートされません。 証明書には固有のサブジェクト識別名が必要です。 製品は、DN が同じでないことを検査します。

手順

1. 既存のストアから信頼できる証明書をすべて抽出します。

   現在、キュー・マネージャーで TLS を使用している場合、キュー・マネージャーにいくつかのトラステッド証明書が保管されている可能性があります。 これらを抽出して、新しいキュー・マネージャーにコピーする必要があります。 以下のいずれかのオプションの手順を実行します。

   - 証明書の抽出を簡素化するには、ローカル・システム上で以下のスクリプトを実行します。

     ```
     #!/bin/bash

     keyr=$(grep SSLKEYR $1)
     if [ -n "${keyr}" ]; then
        keyrlocation=$(sed -n "s/^.*'\(.*\)'.*$/\1/ p" <<< ${keyr})
        mapfile -t runmqakmResult < <(runmqakm -cert -list -db ${keyrlocation}.kdb -stashed)
        cert=1
        for i in "${runmqakmResult[@]:2}"
        do
           certlabel=$(echo ${i:2} | xargs)
           echo Extracting certificate $certlabel to $cert.cert
           runmqakm -cert -extract -db ${keyrlocation}.kdb -label "$certlabel" -target $
     {cert}.cert -stashed
           cert=$[$cert+1]
        done
     fi
     ```

     このスクリプトを実行するときに、IBM MQ バックアップの場所を引数として指定すると、証明書が抽出されます。 例えば、スクリプトが `extractCert.sh` という名前で、IBM MQ バックアップが `/tmp/backup.mqsc` にある場合は、以下のコマンドを実行します。

     ```
     extractCert.sh /tmp/backup.mqsc
     ```

   - または、以下のコマンドを上から順に実行します。

     a. キュー・マネージャーの TLS キー・リポジトリーの場所を識別します。

     ```
     grep SSLKEYR /tmp/backup.mqsc
     ```

     出力例は次のとおりです。

     ```
     SSLKEYR('/run/runmqserver/tls/key') +
     ```

     ここで、鍵ストアは `/run/runmqserver/tls/key.kdb` にあります

     b. このロケーション情報に基づいて鍵ストアを照会し、保管されている証明書を判別します。

     ```
     runmqakm -cert -list -db /run/runmqserver/tls/key.kdb -stashed
     ```

出力例は次のとおりです。

```
Certificates in database /run/runmqserver/tls/key.kdb:
           default
           CN=cs-ca-certificate,O=cert-manager
```

c. リストされた各証明書を抽出します。 これを行うには、以下のコマンドを実行します。

```
runmqakm -cert -extract -db KEYSTORE_LOCATION -label "LABEL_NAME" -target OUTPUT_FILE
-stashed
```

上記のサンプルでは、これは以下のコマンドと同等です。

```
runmqakm -cert -extract -db /run/runmqserver/tls/key.kdb -label "CN=cs-ca-
certificate,O=cert-manager" -target /tmp/cert-manager.crt -stashed
runmqakm -cert -extract -db /run/runmqserver/tls/key.kdb -label "default" -target /tmp/
default.crt -stashed
```

2. キュー・マネージャーの新しい鍵と証明書を取得します。

マイグレーション済みのキュー・マネージャー上で TLS を構成するには、新しい鍵と証明書を生成します。 これが後でデプロイメント時に使用されます。 多くの組織では、このためにセキュリティー・チームに連絡して鍵と証明書を提供してもらうことが必要になります。 組織によっては、このオプションが使えないため、自己署名証明書を使用します。

以下の例では、有効期限を 10 年に設定して自己署名証明書を生成します。

```
openssl req \
       -newkey rsa:2048 -nodes -keyout qmgr.key \
       -subj "/CN=mq queuemanager/OU=ibm mq" \
       -x509 -days 3650 -out qmgr.crt
```

次の 2 つの新規ファイルが作成されます。

- `qmgr.key` は、キュー・マネージャーの秘密鍵です
- `qmgr.crt` はパブリック証明書です

次のタスク

これで、LDAP を構成する準備ができました。

OpenShift CP4I-SC2 CD オプション: LDAP の構成

IBM MQ Operator は、複数の異なるセキュリティー・アプローチを使用するように構成できます。 通常、LDAP はエンタープライズ・デプロイメントに最も効果的なので、このマイグレーション・シナリオでは LDAP を使用します。

始める前に

このタスクでは、キュー・マネージャーの鍵と証明書を抽出して取得済みであることを前提としています。

このタスクについて

これを行う必要がありますか?

認証と許可に LDAP を既に使用している場合は、変更する必要はありません。

LDAP を使用しているかどうかが分からない場合は、次のコマンドを実行します。

```
connauthname="$(grep CONNAUTH backup.mqsc | cut -d "(" -f2 | cut -d ")" -f1)"; grep -A 20
AUTHINFO\($connauthname\) backup.mqsc
```

出力例は次のとおりです。

```
DEFINE AUTHINFO('USE.LDAP') +
   AUTHTYPE(IDPWLDAP) +
```

```
    ADOPTCTX(YES) +
    CONNAME('ldap-service.ldap(389)') +
    CHCKCLNT(REQUIRED) +
    CLASSGRP('groupOfUniqueNames') +
    FINDGRP('uniqueMember') +
    BASEDNG('ou=groups,dc=ibm,dc=com') +
    BASEDNU('ou=people,dc=ibm,dc=com') +
    LDAPUSER('cn=admin,dc=ibm,dc=com') +
*   LDAPPWD('******************************') +
    SHORTUSR('uid') +
    GRPFIELD('cn') +
    USRFIELD('uid') +
    AUTHORMD(SEARCHGRP) +
*   ALTDATE(2020-11-26) +
*   ALTTIME(15.44.38) +
    REPLACE
```

出力の中に、特に注目できる 2 つの属性があります。

AUTHTYPE
ここに値 IDPWLDAP が設定されている場合は、認証に LDAP を使用しています。

この値がブランクであるか、または別の値である場合は、LDAP が構成されていません。 この場合、許可に LDAP ユーザーが使用されているかどうかを確認するために、AUTHORMD 属性を確認してください。

AUTHORMD
ここに値 OS が設定されている場合は、許可に LDAP を使用していません。

LDAP を使用するように許可と認証を変更するには、以下の作業を行います。

手順

1. LDAP サーバーの IBM MQ バックアップを更新します。
2. LDAP 許可情報の IBM MQ バックアップを更新します。

OpenShift CP4I-SC2 CD LDAP パート 1: LDAP サーバー用の IBM MQ バックアップの更新

LDAP をセットアップする方法についての包括的な説明は、このシナリオでは扱われません。 このトピックでは、プロセスの要約、サンプル、および詳細情報の参照先を示します。

始める前に

このタスクでは、キュー・マネージャーの鍵と証明書を抽出して取得済みであることを前提としています。

このタスクについて

これを行う必要がありますか?

認証と許可に LDAP を既に使用している場合は、変更する必要はありません。 LDAP を使用しているかどうかが分からない場合は、197 ページの『オプション: LDAP の構成』を参照してください。

LDAP サーバーのセットアップには 2 つの段階があります。

1. LDAP 構成を定義します。
2. LDAP 構成をキュー・マネージャー定義に関連付けます。

この構成について詳しくは、以下を参照してください。

- ユーザー・リポジトリーの概要
- AUTHINFO コマンドの参照ガイド

手順

1. LDAP 構成を定義します。

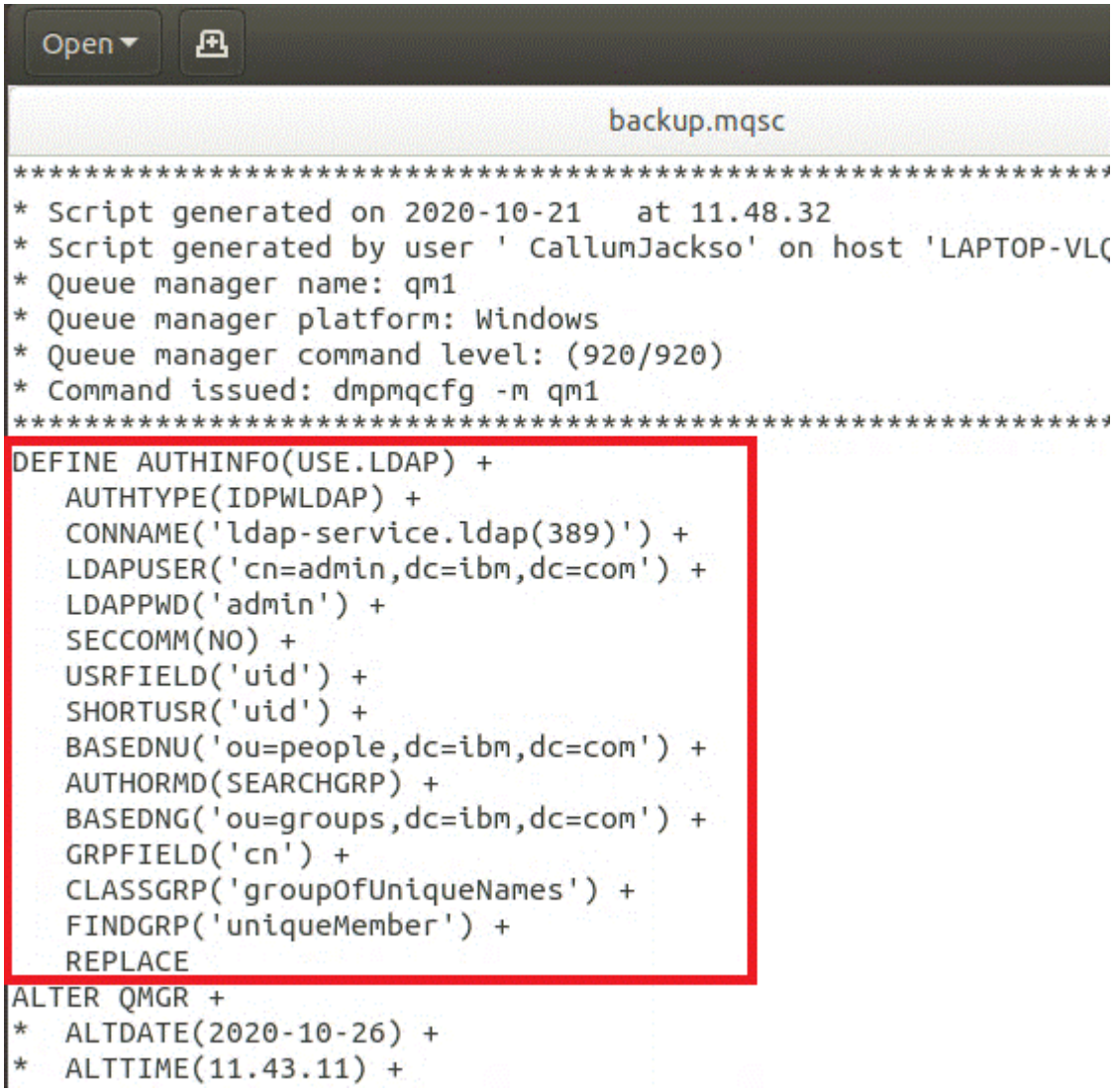
backup.mqsc ファイルを編集して、LDAP システム用の新しい **AUTHINFO** オブジェクトを定義します。以下に例を示します。

```
DEFINE AUTHINFO(USE.LDAP) +
   AUTHTYPE(IDPWLDAP) +
   CONNAME('ldap-service.ldap(389)') +
   LDAPUSER('cn=admin,dc=ibm,dc=com') +
   LDAPPWD('admin') +
   SECCOMM(NO) +
   USRFIELD('uid') +
   SHORTUSR('uid') +
   BASEDNU('ou=people,dc=ibm,dc=com') +
   AUTHORMD(SEARCHGRP) +
   BASEDNG('ou=groups,dc=ibm,dc=com') +
   GRPFIELD('cn') +
   CLASSGRP('groupOfUniqueNames') +
   FINDGRP('uniqueMember')
   REPLACE
```

説明:

- **CONNAME** は、LDAP サーバーに対応するホスト名とポートです。 回復力を高めるために複数のアドレスが用意されている場合は、これらをコンマ区切りリストにして構成することができます。
- **LDAPUSER** は、IBM MQ が LDAP に接続してユーザー・レコードを照会するときに使用するユーザーに対応する識別名です。
- **LDAPPWD** は、**LDAPUSER** ユーザーに対応するパスワードです。
- **SECCOM** は、LDAP サーバーへの通信に TLS を使用する必要があるかどうかを指定します。 指定可能な値:
  - YES: TLS を使用し、証明書は IBM MQ サーバーによって提供されます。
  - ANON: TLS を使用しますが、証明書は IBM MQ サーバーによって提供されません。
  - NO: 接続中に TLS を使用しません。
- **USRFIELD** は、提示されたユーザー名の突き合わせに使用する LDAP レコード内のフィールドを指定します。
- **SHORTUSR** は、LDAP レコード内の長さが 12 文字を超えないフィールドです。 認証に成功すると、このフィールド内の値は表明された ID になります。
- **BASEDNU** は、LDAP の検索に使用する必要がある基本 DN です。
- **BASEDNG** は、LDAP 内のグループの基本 DN です。
- **AUTHORMD** は、ユーザーのグループ・メンバーシップを解決するために使用するメカニズムを定義します。 次の 4 つのオプションがあります。
  - OS: 短い名前に関連付けられているグループのオペレーティング・システムを照会します。
  - SEARCHGRP: LDAP 内のグループ・エントリーから認証済みユーザーを検索します。
  - SEARCHUSR: 認証済みユーザー・レコードからグループ・メンバーシップ情報を検索します。
  - SRCHGRPSN: LDAP 内のグループ・エントリーから、認証済みユーザーの短いユーザー名 (SHORTUSR フィールドで定義する) を検索します。
- **GRPFIELD** は、単純名に対応する LDAP グループ・レコード内の属性です。 これを指定すると、許可レコードの定義に使用できます。
- **CLASSUSR** は、ユーザーに対応する LDAP オブジェクト・クラスです。
- **CLASSGRP** は、グループに対応する LDAP オブジェクト・クラスです。
- **FINDGRP** は、グループ・メンバーシップに対応する LDAP レコード内の属性です。

新しい項目はファイル内の任意の場所に置くことができますが、新しい項目をファイルの先頭に置くと便利です。



2. LDAP 構成をキュー・マネージャー定義に関連付けます。

   LDAP 構成をキュー・マネージャー定義に関連付ける必要があります。 DEFINE AUTHINFO 項目のすぐ下に ALTER QMGR 項目があります。 新しく作成された AUTHINFO 名に対応するように CONNAUTH 項目を変更します。 例えば直前の例では、AUTHINFO(USE.LDAP) が定義されています。これは、名前が USE.LDAP であることを示しています。 それで、CONNAUTH('SYSTEM.DEFAULT.AUTHINFO.IDPWOS') を CONNAUTH('USE.LDAP') に変更します。

LDAP への切り替えがすぐに行われるように、ALTER QMGR コマンドの直後に行を追加して、REFRESH SECURITY コマンドを呼び出します。

*backup.mqsc

```
********************************************************************
* Script generated on 2020-10-21   at 11.48.32
* Script generated by user ' CallumJackso' on host 'LAPTOP-VLQKJ5UH'
* Queue manager name: qm1
* Queue manager platform: Windows
* Queue manager command level: (920/920)
* Command issued: dmpmqcfg -m qm1
********************************************************************
DEFINE AUTHINFO(USE.LDAP) +
   AUTHTYPE(IDPWLDAP) +
   CONNAME('ldap-service.ldap(389)') +
   LDAPUSER('cn=admin,dc=ibm,dc=com') +
   LDAPPWD('admin') +
   SECCOMM(NO) +
   USRFIELD('uid') +
   SHORTUSR('uid') +
   BASEDNU('ou=people,dc=ibm,dc=com') +
   AUTHORMD(SEARCHGRP) +
   BASEDNG('ou=groups,dc=ibm,dc=com') +
   GRPFIELD('cn') +
   CLASSGRP('groupOfUniqueNames') +
   FINDGRP('uniqueMember') +
   REPLACE
ALTER QMGR +
*  ALTDATE(2020-10-26) +
*  ALTTIME(11.43.11) +
   CCSID(850) +
   CERTLABL('default') +
   CLWLUSEQ(LOCAL) +
*  COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE) +
   CONNAUTH('USE.LDAP') +
*  CRDATE(2020-10-26) +
*  CRTIME(11.43.11) +
*  QMID(qm1_2020-10-26_11.43.11) +
   SSLCRYP(' ') +
   SSLKEYR('/run/runmqserver/tls/key') +
   SUITEB(NONE) +
*  VERSION(09020000) +
   FORCE
REFRESH SECURITY
```

次のタスク

これで、LDAP 許可情報の IBM MQ バックアップを更新する準備ができました。

OpenShift CP4I-SC2 CD

LDAP パート 2: LDAP 許可情報の IBM MQ バックアップの更新

IBM MQ には、IBM MQ オブジェクトへのアクセスを制御する、細分化された許可規則が用意されています。 この認証および許可を LDAP に変更すると、許可規則が無効になって更新が必要になる場合があります。

始める前に

このタスクでは、LDAP サーバーのバックアップを更新済みであることを前提としています。

このタスクについて

これを行う必要がありますか?

認証と許可に LDAP を既に使用している場合は、変更する必要はありません。 LDAP を使用しているかどうかが分からない場合は、197 ページの『オプション: LDAP の構成』を参照してください。

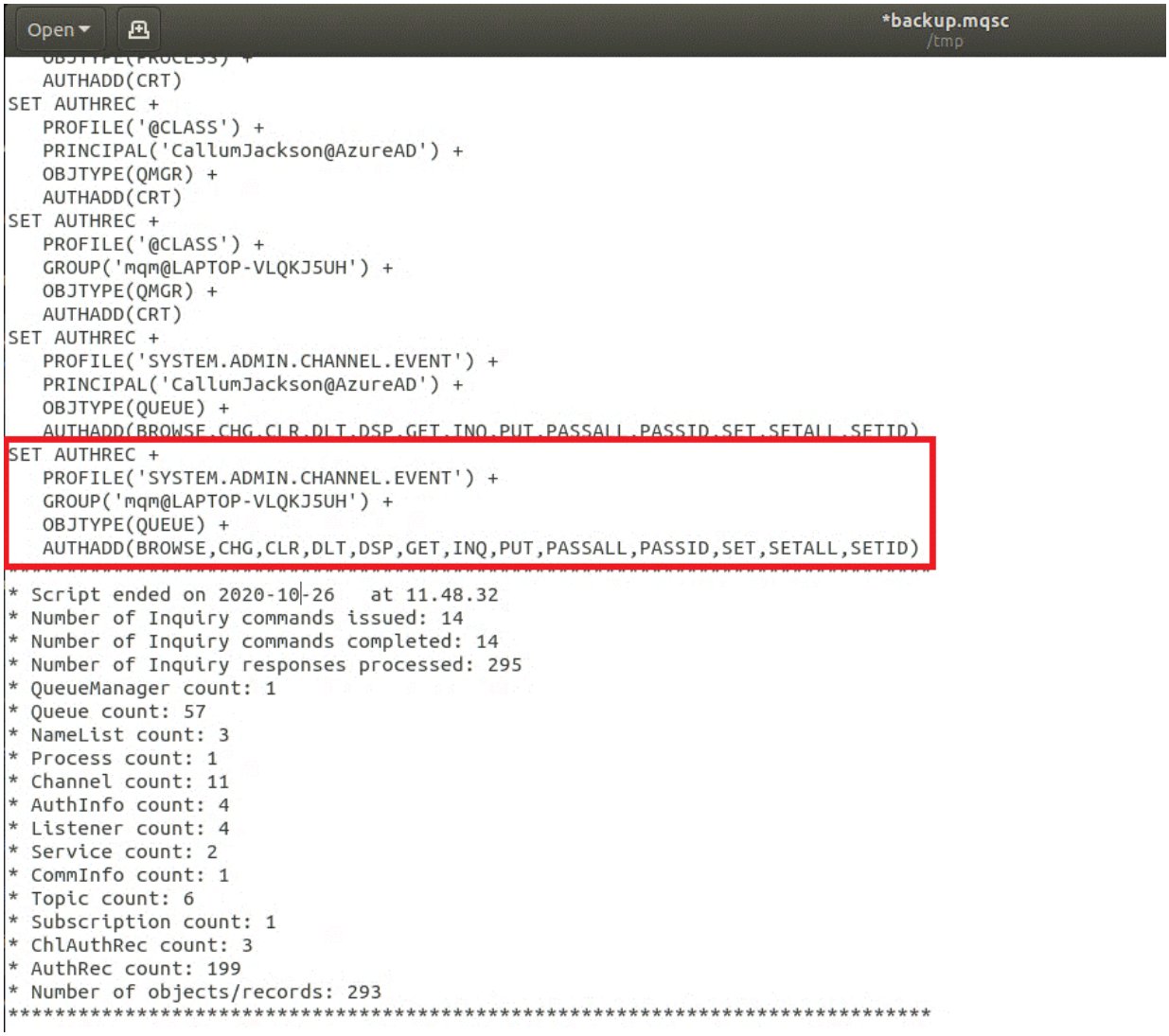
LDAP 許可情報の更新には 2 つの部分があります。

1. 既存のすべての許可をファイルから削除します。
2. LDAP 用の新しい許可情報を定義します。

手順

1. 既存のすべての許可をファイルから削除します。

   バックアップ・ファイルのファイルの終わり近くに、SET AUTHREC で始まるいくつかの項目があるはずです。



既存の項目を見つけて削除します。 一番手っ取り早い方法は、既存の SET AUTHREC ルールをすべて削除してから、LDAP 項目に基づいて新しい項目を作成する方法です。

2. LDAP 用の新しい許可情報を定義します。

   キュー・マネージャーの構成、およびリソースとグループの数によっては、この作業に時間がかかる場合もありますし、簡単にできる場合もあります。 以下の例では、キュー・マネージャーには Q1 という名前のキューが 1 つしかなく、LDAP グループ apps にアクセス権限を許可しようとしている状況を想定しています。

   ```
   SET AUTHREC GROUP('apps') OBJTYPE(QMGR) AUTHADD(ALL)
   SET AUTHREC PROFILE('Q1') GROUP('apps') OBJTYPE(QUEUE) AUTHADD(ALL)
   ```

   最初の AUTHREC コマンドは、キュー・マネージャーにアクセスするための権限を追加し、2 番目のコマンドはキューへのアクセス権限を設定します。 2 番目のキューにアクセスする必要がある場合は、3 番目の AUTHREC コマンドが必要になります。あるいは、ワイルドカードを使用して一般化したアクセス権限を設定することもできます。

   ここで別の例を見てみましょう。 管理者グループ (名前は admins) がキュー・マネージャーに対する全アクセス権限を必要とする場合は、以下のコマンドを追加します。

   ```
   SET AUTHREC PROFILE('*') OBJTYPE(QUEUE) GROUP('admins') AUTHADD(ALL)
   SET AUTHREC PROFILE('*') OBJTYPE(TOPIC) GROUP('admins') AUTHADD(ALL)
   SET AUTHREC PROFILE('*') OBJTYPE(CHANNEL) GROUP('admins') AUTHADD(ALL)
   SET AUTHREC PROFILE('*') OBJTYPE(CLNTCONN) GROUP('admins') AUTHADD(ALL)
   SET AUTHREC PROFILE('*') OBJTYPE(AUTHINFO) GROUP('admins') AUTHADD(ALL)
   ```

```
SET AUTHREC PROFILE('*') OBJTYPE(LISTENER) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(NAMELIST) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(PROCESS) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(SERVICE) GROUP('admins') AUTHADD(ALL)
SET AUTHREC PROFILE('*') OBJTYPE(QMGR) GROUP('admins') AUTHADD(ALL)
```

次のタスク

これで、IBM MQ 構成の IP アドレスとホスト名を変更する準備ができました。

OpenShift CP4I-SC2 CD オプション: IBM MQ 構成内の IP アドレスとホスト名の変更

IBM MQ 構成には、IP アドレスとホスト名が指定されている場合があります。 状況によっては、これらをそのまま使用できる場合もありますが、更新する必要がある場合もあります。

始める前に

このタスクでは、LDAP を構成済みであることを前提としています。

このタスクについて

これを行う必要がありますか?

まず、直前のセクションで定義した LDAP 構成とは別に、IP アドレスまたはホスト名を指定したかどうかを判別します。 そのためには、以下のコマンドを実行します。

```
grep 'CONNAME\|LOCLADDR\|IPADDRV' -B 3  backup.mqsc
```

出力例は次のとおりです。

```
*****************************************************
DEFINE AUTHINFO(USE.LDAP) +
   AUTHTYPE(IDPWLDAP) +
   CONNAME('ldap-service.ldap(389)') +
--
DEFINE AUTHINFO('SYSTEM.DEFAULT.AUTHINFO.IDPWLDAP') +
   AUTHTYPE(IDPWLDAP) +
   ADOPTCTX(YES) +
   CONNAME(' ') +
--
   REPLACE
DEFINE AUTHINFO('SYSTEM.DEFAULT.AUTHINFO.CRLLDAP') +
   AUTHTYPE(CRLLDAP) +
   CONNAME(' ') +
```

この例では、検索によって 3 つの結果が返されています。 1 つの結果は、前に定義した LDAP 構成に対応しています。 LDAP サーバーのホスト名は同じままであるため、これは無視できます。 他の 2 つの結果は空の接続項目であるため、これらも無視できます。 さらに他の項目がなければ、このトピックの残りの部分をスキップできます。

手順

1. 返された項目の意味を考えます。

   IBM MQ では、構成の多くの側面に IP アドレス、ホスト名、およびポートを含めることができます。 これらは、次の 2 つのカテゴリーに分類できます。

   a. **このキュー・マネージャーのロケーション**: このキュー・マネージャーが使用またはパブリッシュするロケーション情報。IBM MQ ネットワーク内の他のキュー・マネージャーやアプリケーションはこの情報を使用して接続することができます。

   b. **キュー・マネージャーの依存関係のロケーション**: このキュー・マネージャーが認識している必要がある他のキュー・マネージャーまたはシステムのロケーション。

このシナリオは、このキュー・マネージャー構成への変更のみに焦点を当てているため、カテゴリー (a) の構成の更新のみについて説明します。 ただし、このキュー・マネージャーのロケーションが他のキュー・マネージャーまたはアプリケーションによって参照されている場合は、このキュー・マネージャーの新しいロケーションと一致するように、それらの構成の更新が必要になることがあります。

更新する必要がある情報が含まれている可能性がある主なオブジェクトは、次の 2 つです。

- リスナー: これは、IBM MQ が listen するネットワーク・アドレスを表します。
- CLUSTER RECEIVER チャネル: キュー・マネージャーが IBM MQ クラスターの一部である場合、このオブジェクトが存在します。 これは、他のキュー・マネージャーが接続できるネットワーク・アドレスを指定します。

2. `grep 'CONNAME\|LOCLADDR\|IPADDRV' -B 3 backup.mqsc` コマンドからの元の出力で、CLUSTER RECEIVER チャネルが定義されているかどうかを確認します。 定義されている場合は、その IP アドレスを更新します。

   CLUSTER RECEIVER チャネルが定義されているかどうかを確認するには、オリジナルの出力の中に、次のように CHLTYPE(CLUSRCVR) を持つ項目を見つけます。

   ```
   DEFINE CHANNEL(ANY_NAME) +
      CHLTYPE(CLUSRCVR) +
   ```

   項目が存在する場合は、 IBM MQ Red Hat OpenShift 経路を使用して CONNAME を更新します。 この値は、 Red Hat OpenShift 環境に基づいており、予測可能な構文を使用します。

   ```
   queue_manager_resource_name-ibm-mq-qm-openshift_project_name.openshift_app_route_hostname
   ```

   例えば、キュー・マネージャーのデプロイメントが `cp4i` 名前空間内で `qm1` に という名前で、*openshift_app_route_hostname* が `apps.callumj.icp4i.com` の場合、経路 URL は次のようになります。

   ```
   qm1-ibm-mq-qm-cp4i.apps.callumj.icp4i.com
   ```

   このルートのポート番号は、通常 443 です。 Red Hat OpenShift 管理者が異なる方法で指示しない限り、これは通常、正しい値になります。 この情報を使用して、CONNAME フィールドを更新します。 以下に例を示します。

   ```
   CONNAME('qm1-ibm-mq-qm-cp4i.apps.callumj.icp4i.com(443)')
   ```

   `grep 'CONNAME\|LOCLADDR\|IPADDRV' -B 3 backup.mqsc` コマンドの元の出力で、LOCLADDR または IPADDRV の項目が存在するかどうかを確認します。 それらが存在する場合は、削除します。 それらはコンテナー環境では関係ありません。

次のタスク

これで、コンテナー環境用にキュー・マネージャー構成を更新する準備ができました。

OpenShift CP4I-SC2 CD コンテナー環境用のキュー・マネージャー構成の更新

コンテナーで実行する場合、構成の特定の側面はコンテナーによって定義され、エクスポートされた構成と対立する可能性があります。

始める前に

このタスクでは、IBM MQ 構成の IP アドレスとホスト名を変更済みであることを前提としています。

このタスクについて

以下の構成の側面は、コンテナーによって定義されます。

- リスナー定義 (公開されたポートに対応)。

- TLS ストアの候補となる場所。

そのような理由で、エクスポートした構成を更新する必要があります。

1. リスナー定義をすべて削除します。
2. TLS 鍵リポジトリーの場所を定義します。

手順

1. リスナー定義をすべて削除します。

   バックアップ構成内で DEFINE LISTENER を検索します。 これは、AUTHINFO 定義と SERVICE 定義の間にあるはずです。 強調表示されている領域を削除します。

```
*backup.mqsc
*  ALTDATE(2020-11-26) +
*  ALTTIME(11.43.28) +
   REPLACE
DEFINE AUTHINFO('SYSTEM.DEFAULT.AUTHINFO.CRLLDAP') +
   AUTHTYPE(CRLLDAP) +
   CONNAME(' ') +
*  ALTDATE(2020-10-26) +
*  ALTTIME(11.43.28) +
   REPLACE
DEFINE LISTENER('SYSTEM.DEFAULT.LISTENER.LU62') +
   TRPTYPE(LU62) +
   CONTROL(MANUAL) +
*  ALTDATE(2020-10-26) +
*  ALTTIME(11.43.28) +
   REPLACE
DEFINE LISTENER('SYSTEM.DEFAULT.LISTENER.NETBIOS') +
   TRPTYPE(NETBIOS) +
   CONTROL(MANUAL) +
   LOCLNAME(' ') +
*  ALTDATE(2020-10-26) +
*  ALTTIME(11.43.28) +
   REPLACE
DEFINE LISTENER('SYSTEM.DEFAULT.LISTENER.SPX') +
   TRPTYPE(SPX) +
   CONTROL(MANUAL) +
*  ALTDATE(2020-10-26) +
*  ALTTIME(11.43.28) +
   REPLACE
DEFINE LISTENER('SYSTEM.DEFAULT.LISTENER.TCP') +
   TRPTYPE(TCP) +
   CONTROL(MANUAL) +
*  ALTDATE(2020-10-26) +
*  ALTTIME(11.43.28) +
   REPLACE
DEFINE SERVICE('SYSTEM.AMQP.SERVICE') +
   CONTROL(QMGR) +
   SERVTYPE(SERVER) +
   STARTCMD('+MQ_INSTALL_PATH+\bin\amqp.bat') +
   STARTARG('start -m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+\."
   STOPCMD('+MQ INSTALL PATH+\bin64\endmqsde.exe') +
```

2. TLS 鍵リポジトリーの場所を定義します。

   キュー・マネージャーのバックアップには、元の環境の TLS 構成が含まれています。 これはコンテナー環境とは異なるため、いくつかの更新が必要です。

   - **CERTLABL** 項目を `default` に変更します
   - TLS キー・リポジトリーの場所 (**SSLKEYR**) を `/run/runmqserver/tls/key` に変更します

   ファイル内の **SSLKEYR** 属性の位置を検索するには、**SSLKEYR** を検索します。 通常は 1 つの項目のみが検出されます。 複数の項目が検出された場合は、以下の図に示すように、**QMGR** オブジェクトを編集中であるか確認します。

*backup.mqsc

```
************************************************************************
* Script generated on 2020-10-21   at 11.48.32
* Script generated by user ' CallumJackso' on host 'LAPTOP-VLQKJ5UH'
* Queue manager name: qm1
* Queue manager platform: Windows
* Queue manager command level: (920/920)
* Command issued: dmpmqcfg -m qm1
************************************************************************
DEFINE AUTHINFO(USE.LDAP) +
   AUTHTYPE(IDPWLDAP) +
   CONNAME('ldap-service.ldap(389)') +
   LDAPUSER('cn=admin,dc=ibm,dc=com') +
   LDAPPWD('admin') +
   SECCOMM(NO) +
   USRFIELD('uid') +
   SHORTUSR('uid') +
   BASEDNU('ou=people,dc=ibm,dc=com') +
   AUTHORMD(SEARCHGRP) +
   BASEDNG('ou=groups,dc=ibm,dc=com') +
   GRPFIELD('cn') +
   CLASSGRP('groupOfUniqueNames') +
   FINDGRP('uniqueMember') +
   REPLACE
ALTER QMGR +
*  ALTDATE(2020-10-26) +
*  ALTTIME(11.43.11) +
   CCSID(850) +
   CERTLABL('default') +
   CLWLUSEQ(LOCAL) +
*  COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE) +
   CONNAUTH('USE.LDAP') +
*  CRDATE(2020-10-26) +
*  CRTIME(11.43.11) +
*  QMID(qm1_2020-10-26_11.43.11) +
   SSLCRYP(' ') +
   SSLKEYR('/run/runmqserver/tls/key') +
   SUITEB(NONE) +
*  VERSION(09020000) +
   FORCE
REFRESH SECURITY
```

次のタスク

これで、コンテナーで実行されている IBM MQ のためのターゲット・アーキテクチャーを選択する準備が整いました。

OpenShift CP4I-SC2 CD コンテナーで実行されている IBM MQ のためのターゲット HA アーキテクチャーの選択

高可用性の要件を満たすために、単一インスタンス (単一の Kubernetes ポッド)、複数インスタンス (2 つのポッド)、およびネイティブ HA (1 つのアクティブ・レプリカ・ポッドと 2 つのスタンバイ・レプリカ・ポッド) のいずれかを選択します。

始める前に

このタスクでは、コンテナー環境用のキュー・マネージャー構成を更新済みであることを前提としています。

このタスクについて

IBM MQ Operator には、以下の 3 つの高可用性オプションがあります。

- **単一インスタンス**: 単一のコンテナー (ポッド) が開始され、障害が発生した場合に Red Hat OpenShift が再始動するのはその責任になります。 Kubernetes 内のステートフル・セットの特性が原因で、このフェイルオーバーの際に、長い時間がかかったり管理アクションの実行が求められたりする状態がいくつかあります。
- **複数インスタンス**: 2 つのコンテナー (それぞれを別個のポッドに配置) が、1 つはアクティブ・モード、もう 1 つはスタンバイで開始します。 このトポロジーでは、フェイルオーバーの大幅な高速化が可能です。 これには、IBM MQ の要件を満たす Read Write Many ファイル・システムが必要です。
- **ネイティブ HA**: 3 つのコンテナー (それぞれが別個のポッド内にある) があり、それぞれにキュー・マネージャーのインスタンスがあります。 1 つのインスタンスがアクティブ・キュー・マネージャーとして機能し、メッセージはそこで処理されてリカバリー・ログに書き込まれます。 そのリカバリー・ログへの書き込みが行われると、アクティブ・キュー・マネージャーはレプリカと呼ばれる他の 2 つのインスタンスにデータを送信します。 アクティブ・キュー・マネージャーを実行しているポッドに障害が発生すると、キュー・マネージャーのレプリカ・インスタンスの 1 つがアクティブの役割を引き継ぎ、このインスタンスにある最新データで運用が行われます。

このタスクでは、ターゲット HA アーキテクチャーのみを選択します。 選択したアーキテクチャーを構成する手順については、このシナリオの後続のタスク (211 ページの『Red Hat OpenShift での新しいキュー・マネージャーの作成』) で説明します。

手順

1. 3 つのオプションを確認します。

   これらのオプションの包括的な説明については、 40 ページの『コンテナー内の IBM MQ の高可用性の計画』 を参照してください。
2. ターゲット HA アーキテクチャーを選択します。

   どちらのオプションを選択すべきか分からない場合、まず**単一インスタンス**のオプションで開始し、これがご使用の環境の高可用性要件を満たしているかどうかを確認します。

次のタスク

これで、キュー・マネージャー・リソースを作成する準備ができました。

OpenShift CP4I-SC2 CD キュー・マネージャー用のリソースの作成

IBM MQ 構成、TLS 証明書、および鍵を Red Hat OpenShift 環境にインポートします。

始める前に

このタスクでは、コンテナーで実行される IBM MQ 用のターゲット・アーキテクチャーを選択済みであることを前提としています。

このタスクについて

前の各セクションで、次の 2 つのリソースの抽出、更新、および定義を完了しました。

- IBM MQ 構成
- TLS 証明書および鍵

キュー・マネージャーがデプロイされる前に、これらのリソースを Red Hat OpenShift 環境にインポートする必要があります。

手順

1. IBM MQ 構成を Red Hat OpenShift にインポートします。

   以下の説明では、現行ディレクトリーの backup.mqsc というファイルに IBM MQ 構成があることを前提としています。 そうでない場合は、ご使用の環境に基づいてファイル名をカスタマイズする必要があります。

   a) oc login を使用してクラスターにログインします。

   b) IBM MQ 構成を configmap にロードします。

   以下のコマンドを実行します。

   ```
   oc create configmap my-mqsc-migrated --from-file=backup.mqsc
   ```

   c) ファイルが正常にロードされたことを確認します。

   以下のコマンドを実行します。

   ```
   oc describe configmap my-mqsc-migrated
   ```

2. IBM MQ TLS リソースをインポートします。

   195 ページの『オプション: キュー・マネージャーの鍵と証明書の抽出および取得』 で説明しているように、キュー・マネージャーのデプロイメントに TLS が必要な場合があります。 その場合は、 .crt および .key で終わるファイルの数が既に存在するはずです。 キュー・マネージャーがこれらをデプロイメント時に参照するために、これらのファイルを Kubernetes シークレットに追加する必要があります。

   キュー・マネージャー用の鍵と証明書が存在する場合、それらは一例として次のような名前になります。

   - qmgr.crt
   - qmgr.key

   これらのファイルをインポートするには、以下のコマンドを実行します。

   ```
   oc create secret tls my-tls-migration --cert=qmgr.crt --key=qmgr.key
   ```

   Kubernetes では、一致する公開鍵と秘密鍵のインポート時に、この役立つユーティリティーが提供されます。 例えば、キュー・マネージャーのトラストストアに追加する証明書がさらにある場合、次のコマンドを実行します。

   ```
   oc create secret generic my-extra-tls-migration --from-file=comma_separated_list_of_files
   ```

   例えば、インポートするファイルが trust1.crt、trust2.crt と trust3.crt の場合、コマンドは次のようになります：

   ```
   oc create secret generic my-extra-tls-migration --from-file=trust1.crt,trust2.crt,trust3.crt
   ```

次のタスク

これで、Red Hat OpenShift で新しいキュー・マネージャーを作成する準備ができました。

OpenShift CP4I-SC2 CD Red Hat OpenShift での新しいキュー・マネージャーの作成

Red Hat OpenShift に単一インスタンス・キュー・マネージャーまたは複数インスタンス・キュー・マネージャーのいずれかをデプロイします。

始める前に

このタスクは、 キュー・マネージャー・リソースの作成および IBM MQ Operator を Red Hat OpenShift にインストールするがあることを前提としています。

このタスクについて

209 ページの『コンテナーで実行されている IBM MQ のためのターゲット HA アーキテクチャーの選択』で概説されているように、3 つの可能なデプロイメント・トポロジーがあります。 したがって、このトピックでは以下の 3 つの異なるテンプレートを提供します

- テンプレート 1: 単一インスタンス・キュー・マネージャーをデプロイします。
- テンプレート 2: 複数インスタンス・キュー・マネージャーをデプロイします。
- テンプレート 3: ネイティブ HA キュー・マネージャーをデプロイします。

重要 : 優先トポロジーに基づいて、3 つのテンプレートのいずれか 1 つのみを実行します。

手順

- **テンプレート 1: 単一インスタンス・キュー・マネージャーをデプロイします。**

  マイグレーションされたキュー・マネージャーは、 YAML ファイルを使用して Red Hat OpenShift にデプロイされます。 前の各トピックで使用された名前に基づいたサンプルを以下に示します。

  注 : この例では、 IBM MQ Advanced ライセンスを使用していますが、 IBM MQ ライセンスを使用することもできます。 詳しくは、161 ページの『を使用して、IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm1
spec:
  version: 9.4.3.0-r1
  license:
    accept: true
    license: L-CYPF-CRPF3H
    use: "Production"
  pki:
    keys:
    - name: default
      secret:
        secretName: my-tls-migration
        items:
          - tls.key
          - tls.crt
  web:
    enabled: true
  queueManager:
    name: QM1
    mqsc:
      - configMap:
          name: my-mqsc-migrated
          items:
            - backup.mqsc
```

  実行した手順によっては、直前の YAML をカスタマイズする必要があります。 これを行うための手引きとして、この YAML の説明を次に示します。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm1
```

  これは、Kubernetes のオブジェクト、タイプ、および名前を定義します。 カスタマイズが必要となるフィールドは name フィールドのみです。

```
 spec:
  version: 9.4.3.0-r1
  license:
    accept: true
    license: L-CYPF-CRPF3H
    use: "Production"
```

これは、デプロイメントのバージョン情報とライセンス情報に対応しています。 これをカスタマイズする必要がある場合は、249 ページの『mq.ibm.com/v1beta1: 現在のライセンスバージョン』に記載されている情報を使用してください。

```
  pki:
    keys:
    - name: default
      secret:
        secretName: my-tls-migration
        items:
          - tls.key
          - tls.crt
```

TLS を使用するように構成されているキュー・マネージャーの場合は、関連する証明書と鍵を参照する必要があります。 `secretName` フィールドは、Kubernetes シークレット (IBM MQ TLS リソースのインポートのセクション内で作成されたもの) を参照します。項目 (`tls.key` および `tls.crt`) のリストは、Kubernetes が `oc create secret tls` 構文の使用時に割り当てる標準名です。 トラストストアに追加する証明書がさらにある場合は、それらは同様の方法で追加できますが、各項目はインポート時に使用された対応する各ファイル名です。 例えば、トラストストア証明書を作成するために次のコードを使用できます。

```
oc create secret generic my-extra-tls-migration --from-file=trust1.crt,trust2.crt,trust3.crt
```

```
  pki:
    trust:
    - name: default
      secret:
        secretName: my-extra-tls-migration
        items:
          - trust1.crt
          - trust2.crt
          - trust3.crt
```

重要 : TLS が必要ない場合は、YAML の TLS セクションを削除します。

```
  web:
    enabled: true
```

これは、デプロイメントの Web コンソールを有効にします。

```
  queueManager:
    name: QM1
```

これは、キュー・マネージャーの名前を QM1 と定義します。 キュー・マネージャーは、お客様の要件 (例えば、元のキュー・マネージャー名) に基づいてカスタマイズします。

```
    mqsc:
      - configMap:
          name: my-mqsc-migrated
          items:
            - backup.mqsc
```

直前のコードは、IBM MQ 構成をインポートするセクションでインポートされたキュー・マネージャー構成をプルします。 別の名前を使用した場合は、`my-mqsc-migrated` と `backup.mqsc` を変更する必要があります。

サンプルの YAML では、Red Hat OpenShift 環境のデフォルトのストレージ・クラスが RWX または RWO ストレージ・クラスのいずれかとして定義されていることが前提となっていることに注意してください。 ご使用の環境内でデフォルトが定義されていない場合は、使用するストレージ・クラスを指定する必要があります。 これは、YAML を次のように拡張することで行えます。

```
  queueManager:
    name: QM1
    storage:
      defaultClass: my_storage_class
```

```
      queueManager:
        type: persistent-claim
```

強調表示されたテキストを、ご使用の環境に合わせてカスタマイズしたクラス属性を設定して追加します。 ご使用の環境内のストレージ・クラス名を検出するには、次のコマンドを実行します。

```
oc get storageclass
```

このコマンドによって返される出力例を次に示します。

```
NAME          PROVISIONER             RECLAIMPOLICY
aws-efs       openshift.org/aws-efs   Delete
gp2 (default) kubernetes.io/aws-ebs   Delete
```

以下のコードは、IBM MQ 構成 (IBM MQ 構成のインポートのセクションでインポートされたもの) を参照する方法を示しています。 別の名前を使用した場合は、`my-mqsc-migrated` と `backup.mqsc` を変更する必要があります。

```
    mqsc:
      - configMap:
          name: my-mqsc-migrated
          items:
            - backup.mqsc
```

単一インスタンス・キュー・マネージャーのデプロイが完了しました。 これにより、テンプレートが完了します。 これで、新しいコンテナー・デプロイメントを検証する準備ができました。

- **テンプレート 2: 複数インスタンス・キュー・マネージャーをデプロイします。**

  マイグレーションされたキュー・マネージャーは、 YAML ファイルを使用して Red Hat OpenShift にデプロイされます。 前の各セクションで使用された名前に基づいたサンプルを以下に示します。

  注 : この例では、 IBM MQ Advanced ライセンスを使用していますが、 IBM MQ ライセンスを使用することもできます。 詳しくは、161 ページの『を使用して、IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm1mi
spec:
  version: 9.4.3.0-r1
  license:
    accept: true
    license: L-CYPF-CRPF3H
    use: "Production"
  pki:
    keys:
    - name: default
      secret:
        secretName: my-tls-migration
        items:
          - tls.key
          - tls.crt
  web:
    enabled: true
  queueManager:
    name: QM1
    availability: MultiInstance
    storage:
      defaultClass: aws-efs
      persistedData:
        enabled: true
      queueManager:
        enabled: true
      recoveryLogs:
        enabled: true
    mqsc:
      - configMap:
          name: my-mqsc-migrated
          items:
            - backup.mqsc
```

ここでは、この YAML について説明します。 構成の大部分は、単一インスタンス・キュー・マネージャーをデプロイする方法と同じであるため、ここでは、キュー・マネージャーの可用性とストレージの側面についてのみ説明します。

```
queueManager:
    name: QM1
    availability: MultiInstance
```

これは、キュー・マネージャー名を QM1 として指定し、デプロイメントがデフォルトの単一インスタンスではなく MultiInstance になるように設定します。

```
    storage:
      defaultClass: aws-efs
      persistedData:
        enabled: true
      queueManager:
        enabled: true
      recoveryLogs:
        enabled: true
```

IBM MQ 複数インスタンス・キュー・マネージャーは、RWX ストレージに依存します。 デフォルトでは、キュー・マネージャーは単一インスタンス・モードでデプロイされるため、複数インスタンス・モードに変更する場合は追加のストレージ・オプションが必要になります。 直前の YAML サンプルでは、3 つのストレージ永続ボリュームと 1 つの永続ボリューム・クラスが定義されています。 この永続ボリューム・クラスは、RWX ストレージ・クラスでなければなりません。 ご使用の環境内の各ストレージ・クラス名が分からない場合は、次のコマンドを実行してそれらを検出することができます。

```
oc get storageclass
```

このコマンドによって返される出力例を次に示します。

```
NAME          PROVISIONER             RECLAIMPOLICY
aws-efs       openshift.org/aws-efs   Delete
gp2 (default) kubernetes.io/aws-ebs   Delete
```

以下のコードは、IBM MQ 構成 (IBM MQ 構成のインポートのセクションでインポートされたもの) を参照する方法を示しています。 別の名前を使用した場合は、my-mqsc-migrated と backup.mqsc を変更する必要があります。

```
    mqsc:
      - configMap:
          name: my-mqsc-migrated
          items:
            - backup.mqsc
```

複数インスタンス・キュー・マネージャーのデプロイが完了しました。 これにより、テンプレートが完了します。 これで、新しいコンテナー・デプロイメントを検証する準備ができました。

- **テンプレート 3: ネイティブ HA キュー・マネージャーをデプロイします。**

  ネイティブ HA キュー・マネージャーの作成の作業例については、112 ページの『例: IBM MQ Operator を使用したネイティブ HA の構成』を参照してください。

OpenShift CP4I-SC2 CD 新規コンテナー・デプロイメントの検証

IBM MQ が Red Hat OpenShift にデプロイされたので、IBM MQ サンプルを使用して環境を検査できます。

始める前に

このタスクでは、Red Hat OpenShift に新しいキュー・マネージャーを作成済みであることを前提としています。

重要 : このタスクでは、キュー・マネージャーで TLS が有効になっていないことを前提としています。

このタスクについて

このタスクでは、マイグレーション済みキュー・マネージャーのコンテナーの内部から IBM MQ サンプルを実行します。 ただし、別の環境から実行される独自のアプリケーションを使用することもできます。

以下の情報が必要になります。

- LDAP ユーザー名
- LDAP パスワード
- IBM MQ チャネル名
- キュー名

このサンプル・コードは、以下の設定を使用します。 この設定はご使用の環境によって異なることに注意してください。

- LDAP ユーザー名: mqapp
- LDAP パスワード: mqapp
- IBM MQ チャネル名: DEV.APP.SVRCONN
- キュー名: Q1

手順

1. 実行中の IBM MQ コンテナーに exec を実行します。

   以下のコマンドを使用します。

   ```
   oc exec -it qm1-ibm-mq-0 /bin/bash
   ```

   ここで qm1-ibm-mq-0 は、211 ページの『Red Hat OpenShift での新しいキュー・マネージャーの作成』でデプロイしたポッドです。 別のデプロイメントを呼び出した場合は、この値をカスタマイズします。
2. メッセージを送信します。

   以下のコマンドを実行します。

   ```
   cd /opt/mqm/samp/bin
   export IBM MQSAMP_USER_ID=mqapp
   export IBM MQSERVER=DEV.APP.SVRCONN/TCP/'localhost(1414)'
   ./amqsputc Q1 QM1
   ```

   パスワードの入力を求めるプロンプトが表示されたら、メッセージを送信できます。
3. メッセージが正常に受信されたことを確認します。

   GET サンプルを実行します。

   ```
   ./amqsgetc Q1 QM1
   ```

タスクの結果

193 ページの『IBM MQ Operator へのマイグレーション』が完了しました。

次のタスク

より複雑なマイグレーション・シナリオについては、以下の情報を参考にしてください。

キューに入ったメッセージのマイグレーション
キューに入っている既存のメッセージをマイグレーションするには、新しいキュー・マネージャーの配置後にメッセージをエクスポートおよびインポートするために、2 つのシステム間での dmpmqmsg ユーティリティーの使用のトピックのガイドに従ってください。

Red Hat OpenShift 環境の外部からの IBM MQ への接続

デプロイされたキュー・マネージャーは、 Red Hat OpenShift 環境外の IBM MQ クライアントおよびキュー・マネージャーに公開することができます。 このプロセスは、 Red Hat OpenShift 環境に接続する IBM MQ のバージョンによって異なります。 148 ページの『Red Hat OpenShift クラスターの外部からキュー・マネージャーに接続するためのルートの構成』を参照してください。

コンテナーでの IBM MQ の操作

コンテナーで実行されている IBM MQ キュー・マネージャーを操作または操作する必要がある場合、詳しくは以下のトピックを参照してください。

手順

- 217 ページの『IBM MQ Operator を使用した IBM MQ』.
- 230 ページの『ネイティブ HA キュー・マネージャーの状況の表示』.
- 240 ページの『ネイティブ HA キュー・マネージャーの終了』.

OpenShift CP4I IBM MQ Operator を使用した IBM MQ

手順

- 218 ページの『IBM MQ Console クラスターにデプロイされた Red Hat OpenShift への接続』.
- 218 ページの『IBM MQ Operator 使用時のモニター』.
- 229 ページの『Red Hat OpenShift CLI を使用したキュー・マネージャー構成のバックアップおよびリストア』.

OpenShift CP4I IBM MQ Console に対する権限の付与

IBM MQ Console の権限は、ライセンス使用状況に応じて異なる方法で管理されます。

このタスクについて

キュー・マネージャに使用されているライセンスに基づいて、関連するオプションを選択してください。

手順

- **オプション 1:** Keycloak てユーザーとロールを設定する。

  IBM Cloud Pak for Integration ライセンスを使用している場合、 IBM MQ Console は、ID およびアクセス管理に Keycloak を使用します。

  - Keycloak 構成については、IBM Cloud Pak for Integration ドキュメントの Identity and Access management を参照のこと。
  - *MQ* ロールとそれに対応する Keycloak ロールについては、ドキュメントの Cloud Pak for Integrationroles and permissions を参照してください IBM Cloud Pak for Integration ロールは、インストール全体、または個々のインスタンス（キューマネージャ）ごとに設定できます。 *MQ* ロールに相当するものは、**キューマネージャー**の下に記載されています。
  - 旧バージョンの IBM MQ Operator で IAM を使用してユーザーを構成していた場合は、IBM Cloud Pak for Integration ドキュメントの「Migrating users from IAM toKeycloak を参照してください。
- **オプション 2：**ユーザーとロールを手動で設定する。

  If you are using an IBM MQ license, the IBM MQ Console is not preconfigured and you need to configure it yourself.

  - ユーザーとロールについては、ユーザーとロールの設定を参照してください。
  - わかりやすい例を 166 ページの『IBM MQ Operator を使用した基本レジストリーでの IBM MQ Console の構成』で参照することができます。

– あるいは、キュー・マネージャを IBM Cloud Pak for Integration ライセンスを使用するように変更した場合、前述のように Keycloak 構成することができます。

OpenShift CP4I IBM MQ Console クラスターにデプロイされた Red Hat OpenShift への接続

Red Hat OpenShift Container Platform クラスターにデプロイされているキュー・マネージャーの IBM MQ Console に接続する方法について説明します。

始める前に

IBM MQ Console アクセス許可が必要です。 217 ページの『IBM MQ Console に対する権限の付与』を参照してください。

このタスクについて

IBM MQ Console URL は、 Red Hat OpenShift Web コンソールの `QueueManager` 詳細ページまたは IBM Cloud Pak for Integration Platform UI で見つけることができます。 あるいは、以下のコマンドを実行することにより、Red Hat OpenShiftCLI からそれを検出することもできます。

```
oc get queuemanager QueueManager Name -n namespace_of_your_MQ_deployment --output
jsonpath='{.status.adminUiUrl}'
```

関連タスク

148 ページの『Red Hat OpenShift クラスターの外部からキュー・マネージャーに接続するためのルートの構成』
Red Hat OpenShift クラスターの外部から IBM MQ キュー・マネージャーにアプリケーションを接続するには、 Red Hat OpenShift 経路が必要です。 IBM MQ キュー・マネージャーおよびクライアント・アプリケーションで TLS を有効にする必要があります。SNI は、TLS 1.2 以上のプロトコルが使用されている場合にのみ TLS プロトコルで使用できるためです。 Red Hat OpenShift Container Platform Router では、IBM MQ キュー・マネージャーへの要求のルーティングに SNI が使用されます。

OpenShift CP4I IBM MQ Operator 使用時のモニター

IBM MQ Operator が管理するキュー・マネージャーは、Prometheus と互換性のあるメトリックを生成できます。 キューマネージャーのメトリクスは、デフォルトで HTTP で有効化され、提供されています。

始める前に

キューマネージャーのメトリクスに HTTPS を使用する

9.4.2.0-r1 から、キューマネージャーは、 OpenShift で作成した証明書を使用して、 HTTPS 経由でメトリクスを提供することができます。 9.4.2.0-r1 キューマネージャーでは、 HTTP 経由でのみメトリクスを提供できます。

注 : すべてのキューマネージャーは、デフォルトで HTTP を通じてメトリクスを提供します。

このタスクについて

Prometheus オープンソースの時系列データベースであり、メトリクス用のルール評価エンジンです。 IBM MQ コンテナは、 Prometheus によって問い合わせ可能なメトリクスエンドポイントを公開しています。 メトリクスは、監視とアクティビティのトレース用に IBM MQ Operator システムトピックから生成されます。

OpenShift Container Platform には、Prometheus サーバーを使用する自己更新型のモニタリング・スタックが、事前インストールおよび事前構成されています。 ユーザー定義のプロジェクトをモニターするには、OpenShift Container Platform モニタリング・スタックを構成する必要があります。 ユーザー定義プロジェクトのモニタリングを有効にするを参照してください。

`QueueManager` をメトリクスを有効にして作成すると、 IBM MQ Operator が `ServiceMonitor` を作成し、 Prometheus オペレーターがそれを検出できるようになります。

手順

- Red Hat OpenShift Container Platform (OCP) モニタリングスタックを使用してキューマネージャーのメトリクスを表示します。

  a) OCP で 「**Metrics**」**タブ**を開きます。

  b)「**メトリック > の観察**」**を**クリックします。

- キューマネージャーのメトリクスを配信する HTTPS

  .spec.queueManager.metrics.tls.provider を openshift に設定します。

```
spec:
  queueManager:
    metrics:
      enabled: true
      tls:
        provider: openshift
```

  注 :

  この設定を使用すると、キューマネージャの HTTPS メトリックサーバーで使用される OpenShift 証明書が作成されます。 IBM MQ Operator によって使用されるように設定できるプロバイダーは他にありません。

- キューマネージャーのメトリクスを無効にします。

  .spec.queueManager.metrics.enabled を false に設定します。

関連タスク

246 ページの『Troubleshooting container clusters with the OpenShift Monitoring CLI』
As a troubleshooting aid, you can query the available container cluster metrics using the OpenShift Monitoring CLI.

OpenShift CP4I *Monitoring container clusters with the OpenShift Monitoring console*

You can query the available container cluster metrics using the OpenShift Monitoring web console.

Before you begin

OpenShift Monitoring provides a web console and a CLI. For day-to-day operations you might prefer to use the console, as described in this task. For gathering troubleshooting information you might prefer to use the CLI, as described in “Troubleshooting container clusters with the OpenShift Monitoring CLI” on page 246.

About this task

OpenShift Container Platform includes a pre-configured, pre-installed and self-updating monitoring stack that uses an open source Prometheus server. Queue managers running under an IBM MQ Operator can produce metrics compatible with Prometheus. For a complete list of the metrics available for IBM MQ, see “IBM MQ コンテナが公開するメトリクス” on page 221.

OpenShift Monitoring is the default cluster monitoring method. It is installed with the cluster. It provides Prometheus, Thanos Querier, and a Grafana dashboard that shows queue manager metrics.

To query the available container cluster metrics using the OpenShift Monitoring console, complete the following steps.

Procedure

1. Enable OpenShift Monitoring for the operands deployed in an IBM MQ container cluster.

   a) Log in to the cluster as a cluster administrator.

b) Add "`enableUserWorkload: true`" to the configMap `cluster-monitoring-config` in the `openshift-monitoring` namespace. To do this, follow the steps in Enabling monitoring for user-defined projects in the Red Hat OpenShift documentation. If the `cluster-monitoring-config` ConfigMap does not exist in the `openshift-monitoring` namespace, follow the instructions to create it.

2. Query the IBM MQ metrics using the OpenShift Monitoring console.

   Use the static Grafana dashboard, which presents cluster metrics to cluster administrators.

   Query IBM MQ metrics, and examine the data plotted on a chart, as described in Querying metrics in the Red Hat OpenShift documentation.

   For example, if you query the metric **`ibmmq_qmgr_cpu_load_fifteen_minute_average_percentage`** you get a chart similar to the following screen capture:

   This screen capture also shows the web console prompt. You can run `PromQL` queries from this prompt. For example:

   Get a list of queue managers that have a positive FDC count:

```
ibmmq_qmgr_fdc_files > 0
```

   Pass the `namespace` filter in the query to get the results for a specific namespace:

```
ibmmq_qmgr_fdc_files{namespace='your-namespace'} > 0
```

   Return the count of all such queue managers:

```
count(ibmmq_qmgr_fdc_files{namespace='your-namespace'} > 0)
```

   Get the Write latency of queue manager pods over the last 5 hours:

```
ibmmq_qmgr_log_write_latency_seconds{namespace='your-namespace'}[5h]
```

   Get the RAM usage estimate in bytes for a queue manager for a particular namespace summed by "Job":

```
sum by (job)(ibmmq_qmgr_ram_usage_estimate_for_queue_manager_bytes{qmgr='your-queue-manager', namespace='your-namespace'})
```

The previous queries illustrate use of filters "namespace", "job" and "qmgr". For this metric you can also filter on "container", "endpoint", "instance", "pod", and "prometheus". Note that the available filters are the column headings shown in the screen capture, except for “Name” which is the name of the metric itself.

Related tasks

“Troubleshooting container clusters with the OpenShift Monitoring CLI” on page 246
As a troubleshooting aid, you can query the available container cluster metrics using the OpenShift Monitoring CLI.

Thanos Querier Versus Thanos Querier

Related reference

“IBM MQ コンテナが公開するメトリクス” on page 221
キューマネージャーコンテナは、Red Hat OpenShift モニタリングと互換性のあるメトリクスを公開することができます。

IBM MQ コンテナが公開するメトリクス

キューマネージャーコンテナは、Red Hat OpenShift モニタリングと互換性のあるメトリクスを公開することができます。

メトリック	タイプ	説明
ibmmq_nha_recovery_average_network_round_trip_time_seconds	gauge	平均ネットワーク往復時間
ibmmq_nha_recovery_backlog_average_bytes	gauge	バックログ平均バイト数
ibmmq_nha_recovery_backlog_bytes	gauge	バックログ・バイト
ibmmq_nha_recovery_compressed_log_sent_bytes	gauge	圧縮されたログ・バイトが送信されました
ibmmq_nha_recovery_log_data_average_compression_time_seconds	gauge	ログ・データの平均圧縮時間
ibmmq_nha_recovery_log_data_average_decompression_time_seconds	gauge	ログ・データの平均解凍時間
ibmmq_nha_recovery_log_decompressed_bytes	gauge	ログ・バイトが解凍されました
ibmmq_nha_recovery_log_sent_bytes	counter	送信されたログ・バイト
ibmmq_nha_recovery_rebase_count	gauge	リベース数
ibmmq_nha_recovery_recovery_log_sequence_number	gauge	リカバリ・ログ・シーケンス番号

メトリック	タイプ	説明
ibmmq_nha_replication_acknowledged_log_sequence_number_total	gauge	確認済みログ・シーケンス番号
ibmmq_nha_replication_average_network_round_trip_time_seconds	gauge	平均ネットワーク往復時間
ibmmq_nha_replication_backlog_average_bytes	gauge	バックログ平均バイト数
ibmmq_nha_replication_backlog_bytes	gauge	バックログ・バイト
ibmmq_nha_replication_catch_up_compressed_log_sent_bytes	counter	送信されたキャッチアップ圧縮ログ・バイト数
ibmmq_nha_replication_catch_up_log_data_average_compression_time_seconds	gauge	キャッチアップ・ログ・データの平均圧縮時間
ibmmq_nha_replication_catch_up_log_data_average_decompression_time_seconds	gauge	キャッチアップ・ログ・データの平均解凍時間
ibmmq_nha_replication_catch_up_log_decompressed_bytes	gauge	キャッチアップ・ログの解凍済みバイト数
ibmmq_nha_replication_catch_up_uncompressed_log_sent_bytes	counter	送信されたキャッチアップ非圧縮ログ・バイト数
ibmmq_nha_replication_catchup_log_sent_bytes	counter	送信されたキャッチアップ・ログのバイト数
ibmmq_nha_replication_log_file_system_free_space_percent	gauge	ログ・ファイル・システム - 空き領域
ibmmq_nha_replication_log_file_system_in_use_bytes	gauge	ログ・ファイル・システム - 使用中バイト数
ibmmq_nha_replication_log_write_average_acknowledgement_latency_seconds	gauge	ログ書き込み確認応答の平均待ち時間

メトリック	タイプ	説明
ibmmq_nha_replication_log_write_average_acknowledgement_size_bytes	gauge	ログ書き込み確認応答の平均サイズ
ibmmq_nha_replication_mq_fdc_file_count	gauge	MQ FDC ファイル数
ibmmq_nha_replication_queue_manager_file_system_free_space_percent	gauge	キュー・マネージャー・ファイル・システム - 空き領域
ibmmq_nha_replication_queue_manager_file_system_in_use_bytes	gauge	キュー・マネージャー・ファイル・システム - 使用中バイト数
ibmmq_nha_replication_synchronous_compressed_log_sent_bytes	counter	送信された同期圧縮ログ・バイト数
ibmmq_nha_replication_synchronous_log_data_average_compression_time_seconds	gauge	同期ログ・データの平均圧縮時間
ibmmq_nha_replication_synchronous_log_data_average_decompression_time_seconds	gauge	同期ログ・データの平均減圧時間
ibmmq_nha_replication_synchronous_log_decompressed_bytes	gauge	同期ログ・バイトが解凍されました
ibmmq_nha_replication_synchronous_log_sent_bytes	counter	送信された同期ログのバイト数
ibmmq_nha_replication_synchronous_uncompressed_log_sent_bytes	counter	送信された同期非圧縮ログ・バイト数
ibmmq_qmgr_commit_total	counter	コミット・カウント
ibmmq_qmgr_cpu_load_fifteen_minute_average_percentage	gauge	CPU 負荷 - 15 分間の平均
ibmmq_qmgr_cpu_load_five_minute_average_percentage	gauge	CPU 負荷 - 5 分間の平均

メトリック	タイプ	説明
ibmmq_qmgr_cpu_load_one_minute_average_percentage	gauge	CPU 負荷 - 1 分間の平均
ibmmq_qmgr_destructive_get_bytes_total	counter	破壊的 GET のインターバルにおける合計 - バイト数
ibmmq_qmgr_destructive_get_total	counter	破壊的 GET のインターバルにおける合計 - 数
ibmmq_qmgr_durable_subscription_alter_total	counter	永続サブスクリプション変更回数
ibmmq_qmgr_durable_subscription_create_total	counter	永続サブスクリプション作成回数
ibmmq_qmgr_durable_subscription_delete_total	counter	永続サブスクリプション削除回数
ibmmq_qmgr_durable_subscription_resume_total	counter	永続サブスクリプション再開回数
ibmmq_qmgr_errors_file_system_free_space_percentage	gauge	MQ エラー・ファイル・システム - 空き領域
ibmmq_qmgr_errors_file_system_in_use_bytes	gauge	MQ エラー・ファイル・システム - 使用中バイト数
ibmmq_qmgr_expired_message_total	counter	期限切れメッセージ数
ibmmq_qmgr_failed_browse_total	counter	失敗したブラウズの数
ibmmq_qmgr_failed_mqcb_total	counter	失敗した MQCB の数
ibmmq_qmgr_failed_mqclose_total	counter	失敗した MQCLOSE の数
ibmmq_qmgr_failed_mqconn_mqconnx_total	counter	失敗した MQCONN/MQCONNX の数
ibmmq_qmgr_failed_mqget_total	counter	失敗した MQGET - 数
ibmmq_qmgr_failed_mqinq_total	counter	失敗した MQINQ の数
ibmmq_qmgr_failed_mqopen_total	counter	失敗した MQOPEN の数
ibmmq_qmgr_failed_mqput1_total	counter	失敗した MQPUT1 の数

メトリック	タイプ	説明
ibmmq_qmgr_failed_mqput_total	counter	失敗した MQPUT の数
ibmmq_qmgr_failed_mqset_total	counter	失敗した MQSET の数
ibmmq_qmgr_failed_mqsubrq_total	counter	失敗した MQSUBRQ の数
ibmmq_qmgr_failed_subscription_create_alter_resume_total	counter	サブスクリプションの作成/変更/再開に失敗した回数
ibmmq_qmgr_failed_subscription_delete_total	counter	サブスクリプション削除に失敗した回数
ibmmq_qmgr_failed_topic_mqput_mqput1_total	counter	失敗したトピック MQPUT/MQPUT1 の数
ibmmq_qmgr_fdc_files	gauge	MQ FDC ファイル数
ibmmq_qmgr_log_file_system_free_space_percentage	gauge	ログ・ファイル・システム - 空き領域
ibmmq_qmgr_log_file_system_in_use_bytes	gauge	ログ・ファイル・システム - 使用中バイト数
ibmmq_qmgr_log_file_system_max_bytes	gauge	ログ・ファイル・システム - 最大バイト数
ibmmq_qmgr_log_in_use_bytes	gauge	ログ - 使用中バイト数
ibmmq_qmgr_log_logical_written_bytes_total	counter	ログ - 書き込まれた論理バイト数
ibmmq_qmgr_log_max_bytes	gauge	ログ - 最大バイト数
ibmmq_qmgr_log_occupied_by_extents_waiting_to_be_archived_bytes	gauge	ログ - アーカイブ待ちのエクステントが占有している
ibmmq_qmgr_log_occupied_by_reusable_extents_bytes	gauge	ログ - 再使用可能エクステントが占有するバイト数
ibmmq_qmgr_log_physical_written_bytes_total	counter	ログ - 書き込まれた物理バイト数
ibmmq_qmgr_log_primary_space_in_use_percentage	gauge	ログ - 使用中の現行 1 次スペース

メトリック	タイプ	説明
ibmmq_qmgr_log_required_for_media_recovery_bytes	gauge	ログ - メディア・リカバリーに必要なバイト数
ibmmq_qmgr_log_sequence_number_disk_total	gauge	ログ - ディスクに書き込まれたログのシーケンス番号
ibmmq_qmgr_log_sequence_number_quorum_total	gauge	ログ - クォーラム・ログ・シーケンス番号
ibmmq_qmgr_log_slowest_write_since_restart_seconds	gauge	ログ - 再起動後の最も遅い書き込み
ibmmq_qmgr_log_workload_primary_space_utilization_percentage	gauge	ログ - ワークロード 1 次スペースの使用状況
ibmmq_qmgr_log_write_latency_seconds	gauge	ログ - 書き込み待ち時間
ibmmq_qmgr_log_write_size_bytes	gauge	ログ - 書き込みサイズ
ibmmq_qmgr_mqcb_total	counter	MQCB の数
ibmmq_qmgr_mqclose_total	counter	MQCLOSE の数
ibmmq_qmgr_mqconn_mqconnx_total	counter	MQCONN/MQCONNX の数
ibmmq_qmgr_mqctl_total	counter	MQCTL の数
ibmmq_qmgr_mqdisc_total	counter	MQDISC の数
ibmmq_qmgr_mqinq_total	counter	MQINQ の数
ibmmq_qmgr_mqopen_total	counter	MQOPEN の数
ibmmq_qmgr_mqput_mqput1_bytes_total	counter	MQPUT/MQPUT1 のバイト数のインターバルにおける合計
ibmmq_qmgr_mqput_mqput1_total	counter	MQPUT/MQPUT1 の数のインターバルにおける合計
ibmmq_qmgr_mqset_total	counter	MQSET の数
ibmmq_qmgr_mqstat_total	counter	MQSTAT の数
ibmmq_qmgr_mqsubrq_total	counter	MQSUBRQ の数

メトリック	タイプ	説明
ibmmq_qmgr_non_durable_subscription_create_total	counter	非永続サブスクリプション作成回数
ibmmq_qmgr_non_durable_subscription_delete_total	counter	非永続サブスクリプション削除回数
ibmmq_qmgr_non_persistent_message_browse_bytes_total	counter	非持続メッセージのブラウズ - バイト数
ibmmq_qmgr_non_persistent_message_browse_total	counter	非持続メッセージのブラウズ - 数
ibmmq_qmgr_non_persistent_message_destructive_get_total	counter	非持続メッセージの破壊的 GET - 数
ibmmq_qmgr_non_persistent_message_get_bytes_total	counter	取得された非持続メッセージ - バイト数
ibmmq_qmgr_non_persistent_message_mqput1_total	counter	非持続メッセージ MQPUT1 の数
ibmmq_qmgr_non_persistent_message_mqput_total	counter	非持続メッセージ MQPUT の数
ibmmq_qmgr_non_persistent_message_put_bytes_total	counter	書き込まれた非持続メッセージ - バイト数
ibmmq_qmgr_non_persistent_topic_mqput_mqput1_total	counter	非持続 - トピック MQPUT/MQPUT1 の数
ibmmq_qmgr_persistent_message_browse_bytes_total	counter	持続メッセージのブラウズ - バイト数
ibmmq_qmgr_persistent_message_browse_total	counter	持続メッセージのブラウズ - 数
ibmmq_qmgr_persistent_message_destructive_get_total	counter	持続メッセージの破壊的 GET - 数
ibmmq_qmgr_persistent_message_get_bytes_total	counter	取得された持続メッセージ - バイト数
ibmmq_qmgr_persistent_message_mqput1_total	counter	持続メッセージ MQPUT1 の数

メトリック	タイプ	説明
ibmmq_qmgr_persistent_message_mqput_total	counter	持続メッセージ MQPUT の数
ibmmq_qmgr_persistent_message_put_bytes_total	counter	書き込まれた持続メッセージ - バイト数
ibmmq_qmgr_persistent_topic_mqput_mqput1_total	counter	持続 - トピック MQPUT/MQPUT1 の数
ibmmq_qmgr_published_to_subscribers_bytes_total	counter	サブスクライバーへのパブリッシュ - バイト数
ibmmq_qmgr_published_to_subscribers_message_total	counter	サブスクライバーへのパブリッシュ - メッセージ数
ibmmq_qmgr_purged_queue_total	counter	パージされたキュー数
ibmmq_qmgr_queue_manager_file_system_free_space_percentage	gauge	キュー・マネージャー・ファイル・システム - 空き領域
ibmmq_qmgr_queue_manager_file_system_in_use_bytes	gauge	キュー・マネージャー・ファイル・システム - 使用中バイト数
ibmmq_qmgr_ram_free_percentage	gauge	RAM 空き領域パーセンテージ
ibmmq_qmgr_ram_usage_estimate_for_queue_manager_bytes	gauge	RAM 合計バイト数 - キュー・マネージャーの見積もり
ibmmq_qmgr_rollback_total	counter	ロールバック数
ibmmq_qmgr_system_cpu_time_estimate_for_queue_manager_percentage	gauge	システム CPU 時間 - キュー・マネージャーのパーセンテージ見積もり
ibmmq_qmgr_system_cpu_time_percentage	gauge	システム CPU 時間パーセンテージ
ibmmq_qmgr_topic_mqput_mqput1_total	counter	トピック MQPUT/MQPUT1 のインターバルにおける合計
ibmmq_qmgr_topic_put_bytes_total	counter	書き込まれたトピック・バイト数のインターバルにおける合計
ibmmq_qmgr_trace_file_system_free_space_percentage	gauge	MQ トレース・ファイル・システム - 空き領域

メトリック	タイプ	説明
ibmmq_qmgr_trace_file_system_in_use_bytes	gauge	MQ トレース・ファイル・システム - 使用中バイト数
ibmmq_qmgr_user_cpu_time_estimate_for_queue_manager_percentage	gauge	ユーザー CPU 時間 - キュー・マネージャーのパーセンテージ見積もり
ibmmq_qmgr_user_cpu_time_percentage	gauge	ユーザー CPU 時間パーセンテージ

関連情報

システム・トピックにパブリッシュされるメトリック

OpenShift V 9.4.0 V 9.4.0 キュー・マネージャーの停止 (mq.ibm.com/stop)

キュー・マネージャー定義に注釈を追加して、キュー・マネージャーを停止します。

このタスクについて

IBM MQ Operator によって作成されたキュー・マネージャーには、 StatefulSet が関連付けられています。 この StatefulSet は、「.replicas」フィールドを使用して、特定のキュー・マネージャー可用性タイプに対してデプロイされる Pods の数を宣言します。 これは、1 (単一インスタンス)、2 (複数インスタンス)、または 3 (NativeHA) の値を取ります。

注: 「.replicas」フィールドの値を手動で変更すると、キュー・マネージャーが正しく機能しなくなります。

場合によっては、キュー・マネージャーを停止して、 StatefulSet のレプリカ・カウントが 0 になり、Pods がデプロイされないようにすることができます。 これを行う必要がある場合の例としては、保守中やバックアップ手順中などがあります。

注: キュー・マネージャーの停止時にはキュー・マネージャー Pods がデプロイされていないため、キュー・マネージャーが再始動されるまで、ユーザーとアプリケーションはキュー・マネージャーにアクセスできません。

手順

- キュー・マネージャーを停止するには、「.metadata.annotations」セクションの下のキュー・マネージャー定義に以下のアノテーションを追加します。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: my-qm
  annotations:
    "mq.ibm.com/stop" : "true"
```

- キュー・マネージャーを再始動して正しいレプリカ数に戻すには、キュー・マネージャーからアノテーションを削除するか、その値を「false'」に設定します。

OpenShift CP4I Red Hat OpenShift CLI を使用したキュー・マネージャー構成のバックアップおよびリストア

キュー・マネージャー構成をバックアップすると、キュー・マネージャー構成が失われた場合に、キュー・マネージャーをその定義から再構築することができます。 この手順を実行しても、キュー・マネージャー

のログ・データはバックアップされません。 メッセージは特定の状況で出される一時的なものなので、履歴ログ・データは復元時の対象となりません。

始める前に

oc login を使用してクラスターにログインします。

手順

- キュー・マネージャー構成をバックアップします。

  dmpmqcfg コマンドを使用して、IBM MQ キュー・マネージャーの構成をダンプすることができます。

  a) キュー・マネージャーのポッドの名前を取得します。
  例えば、次のコマンドを実行します。ここで、*queue_manager_name* は QueueManager リソースの名前です。

  ```
  oc get pods --selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/
  instance=queue_manager_name
  ```

  b) 出力をローカル・マシン上のファイルに指定して、ポッド上で **dmpmqcfg** コマンドを実行します。

  dmpmqcfg がキュー・マネージャーの MQSC 構成を出力します。

  ```
  oc exec -it pod_name -- dmpmqcfg > backup.mqsc
  ```

- キュー・マネージャー構成を復元します。

  前のステップで概説したバックアップ手順に従っている場合は、キュー・マネージャー構成を含む backup.mqsc ファイルが必要になります。 このファイルを新しいキュー・マネージャーに適用することで、構成を復元できます。

  a) キュー・マネージャーのポッドの名前を取得します。
  例えば、次のコマンドを実行します。ここで、*queue_manager_name* は QueueManager リソースの名前です。

  ```
  oc get pods --selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/
  instance=queue_manager_name
  ```

  b) ポッド上で **runmqsc** コマンドを実行し、backup.mqsc ファイルの内容を読み込みます。

  ```
  oc exec -i pod_name -- runmqsc < backup.mqsc
  ```

MQ Adv. ネイティブ HA キュー・マネージャーの状況の表示

カスタムビルト・コンテナーの場合、 **dspmq** コマンドを使用してネイティブ HA インスタンスの状況を表示できます。

このタスクについて

実行中のポッドのうち 1 つで **dspmq** コマンドを使用すると、キュー・マネージャー・インスタンスの運用状況を表示できます。 返される情報は、インスタンスがアクティブとレプリカのどちらであるかに応じて異なります。 アクティブ・インスタンスから提供される情報は確定的であり、レプリカ・インスタンスからの情報は古い可能性がある。

V 9.4.2 ネイティブ HA クロスリージョンレプリケーション（CRR）構成の場合、提供される情報も、インスタンスがライブグループまたはリカバリグループのいずれに属しているかによって異なります。 ライブグループからの情報はより確実ですが、リカバリーグループが定期的にしか情報を送り返さないため、まだ古い情報である可能性があります。

以下のアクションを実行できます。

- 現行ノード上のキュー・マネージャー・インスタンスがアクティブかレプリカを表示します。
- 現行ノード上のインスタンスのネイティブ HA の運用状況を表示します。
- ネイティブ HA 構成に属する 3 つのインスタンスすべての運用状況を表示します。
- V 9.4.2 そのインスタンスが所属するグループ、または接続されているグループのステータスを表示します。

以下のステータスフィールドは、キューマネージャインスタンスのネイティブ HA およびネイティブ HA クロスリージョンレプリケーション（CRR）の構成ステータスを報告するために使用されます。 これらのフィールドは常に表示される：

ROLE
インスタンスの現在の役割を指定します。 `Active`、 `Replica`、 `Unknown`、(リカバリグループ用) `Leader`、または `Not Configured` のいずれかです。

INSTANCE
このキュー・マネージャー・インスタンスの作成時に **crtmqm** コマンドの **-lr** オプションを使用してこのキュー・マネージャー・インスタンスに対して指定された名前。

INSYNC
必要な場合にインスタンスがアクティブ・インスタンスとしてテークオーバーできるかどうかを示します。

QUORUM
クォーラムの状況を *number_of_instances_in-sync/number_of_instances_configured* という形式でレポートします。

V 9.4.2 **GRPLSN**
HA グループ内のインスタンスの過半数間で複製されたログシーケンス番号（LSN）を *nnnnn:nnnnn:nnnnn:nnnnn* 形式で指定します。 LSN が不明な場合は空白。

V 9.4.2 **GRPNAME**
HA グループの名前を指定します。 名前が設定されていない場合は空白。

V 9.4.2 **GRPROLE**
HA グループの現在の役割を指定する：

- `Live` - ライブグループ
- `Recovery` - 回復グループ
- `Pending live` - グループになるのを待っている人々 `Live`
- `Pending recovery` - グループになるのを待っている人々 `Recovery`
- `Not configured` - ネイティブ HA は構成されていません
- `Unknown` - 現在の役割を特定できません

これらのフィールドは、`-x` パラメータを使ってすべてのインスタンスに対して拡張情報を要求したときに表示されます：

REPLADDR
キュー・マネージャー・インスタンスの複製アドレス。

CONNACTV
インスタンスがアクティブ・インスタンスに接続されているかどうかを示します。

BACKLOG
アクティブなインスタンスに対して、インスタンスが遅れている KB 数を示します。

CONNINST
指定されたインスタンスがこのインスタンスに接続されているかどうかを示します。

ALTDATE
この情報が最後に更新された日付を示します。 更新されたことがない場合は空白にしてください。

ALTTIME
この情報が最後に更新された時刻を示す。 更新されたことがない場合は空白にしてください。

V 9.4.2 **ACKLSN**
インスタンスが回復ログに書き込まれたことを認めた LSN を *nnnnn:nnnnn:nnnnn:nnnnn* の形式で示します。 LSN 値が不明な場合は空白。

V 9.4.2 **HAStatus**

このインスタンスの動作ステータスを示し、以下の値のいずれかである：

- `Normal` - インスタンスは正常に動作しています。
- `Checking` - インスタンスは、リカバリログの一貫性を確保するためにチェックされています。
- `Synchronizing` - インスタンスにデータを送信して同期させる。
- `Rebasing` - 新しいリカバリログがインスタンスに送信されています。
- `Disk full` - インスタンスのディスクがいっぱいになっている。
- `Disconnected` - インスタンスと通信できません。
- `Unknown` - インスタンスの状態が不明です。

V 9.4.2 **SYNCTIME**
このインスタンスが最後にアクティブなキューマネージャと同期した時刻を ISO 8601 形式で表示します。 時間が不明の場合は空白にしてください。

V 9.4.2 これらのフィールドは、`-g` パラメータを使ってグループ情報を要求したときに表示されます。 これらのフィールドは、グループが Native HA CRR 構成の一部である場合のステータスを表示するのに便利です

V 9.4.2 **GRPNAME**
HA グループの名前を指定します。 名前が設定されていない場合は空白。

V 9.4.2 **GRPROLE**
HA グループの現在の役割を指定する：

- `Live` - ライブグループ
- `Recovery` - 回復グループ
- `Pending live` - グループになるのを待っている人々 `Live`
- `Pending recovery` - グループになるのを待っている人々 `Recovery`
- `Unknown` - 現在の役割を特定できません
- `Not configured` - ネイティブ HA は構成されていません

V 9.4.2 **GRPADDR**
グループに接続するために使用される IP アドレス。 グループへの接続が発生していないため、アドレスが不明な場合は、 `Unknown` に設定されます。

V 9.4.2 **グループバー**
`V.R.M.F` 形式で現在のグループリーダーのバージョンを示します。 `V.R.M.F` は、バージョン、リリース、修正、および修正パックの番号です。 値が不明な場合は、これが設定されます。 `?.?.?.?`

V 9.4.2 **CONNGRP**
グループがローカルグループとつながっているかどうかを示す：

- `yes` - グループはつながっている。
- `no` - グループは切断されています。
- `unknown` - 接続状態が不明です。
- `suspended` - グループは接続されていますが、グループ間の構成上の不整合が解決されるまでは、グループはデータを複製できません。

このフィールドは、ローカルグループ以外のグループに対してのみ表示されます。

V 9.4.2 **INSYNC**

これは、ロールが Recovery または Pending live のグループに対してのみ表示されます。 フェイルオーバーが発生した場合に、グループがデータ損失なしに Live になる可能性があるかどうかを示します。 （Recovery グループと Live グループは、ログが非同期で複製されるため、常に同期しているとは限らないことに注意してください。） RCOVLSN と RCOVTIME の値を使用して、予期せぬフェールオーバーでデータ損失が発生する可能性があるかどうかを示します。

V 9.4.2 **SYNCTIME**

これは、ロールが Recovery または Pending live のグループに対してのみ表示されます。 これは、このグループが Live グループと最後に同期した日時を ISO 8601 形式で示しています（不明の場合は空白）。 （Recovery グループと Live グループは、ログが非同期で複製されるため、常に同期しているとは限らないことに注意してください。） RCOVLSN と RCOVTIME の値を使用して、予期せぬフェールオーバーでデータ損失が発生する可能性があるかどうかを示します。

V 9.4.2 **BACKLOG**

これは、ロールが Recovery または Pending live のグループに対してのみ表示されます。 これは、そのグループがライブグループより遅れている KB 数を示しています。

V 9.4.2 **GRSTATUS**

このグループの動作状態を示し、以下の値のいずれかである：

- Normal - グループは通常通り運営されています。
- Checking - グループは、リカバリログの一貫性を確保するために確認されています。
- Synchronizing - グループに同期させるためのデータが送信されています。
- Rebasing - 新しいリカバリログがインスタンスに送信されています。
- Waiting for connection - グループは、Live または Pending Recovery の役割を持つグループからの接続を待っています。
- Partitioned - 2 つのグループが、自分たちはライブグループであると信じている（つまり、「スプリットブレイン」の状態）。
- Unknown - グループの状態が不明です。

V 9.4.2 **RCOVLSN**

nnnnn:nnnnn:nnnnn:nnnnn フォーマットで、グループが回復できるログシーケンス番号(LSN)。 この時点以降にリカバリログに書き込まれたデータは、予期せぬフェイルオーバーが発生した場合に失われる可能性があります。 この値は、**dmpmqlog** コマンドの -s オプションに渡すことができる。

V 9.4.2 **時間**

ISO 8601 形式で、グループが回復できる時間。 この時間以降にリカバリログに書き込まれたデータは、予期せぬフェールオーバーが発生した場合に失われる可能性があります。 この値は、 **dmpmqlog** コマンドの -t または -u オプションに渡すことができます。

V 9.4.2 **イニットエルエスエヌ**

これは、ロールが Live または Pending recovery のグループに対してのみ表示されます。 これは、ネイティブ HA グループが最初にライブになった際に回復された最後のログレコードのログシーケンス番号（LSN）を示します。 *nnnnn:nnnnn:nnnnn:nnnnn* という形式で表示され、グループがライブになっていない場合は空白になります。

V 9.4.2 **initTime**

これは、ロールが Live または Pending recovery のグループに対してのみ表示されます。 ネイティブ HA グループが最初に稼働した際に、最後に回復されたログレコードの時刻を ISO 8601 形式で表示します。 グループがライブでない場合は空白。

V 9.4.2 **ライブタイム**

これは、ロールが Live または Pending recovery のグループに対してのみ表示されます。 ISO 8601 形式で、グループがライブグループになった最初の日時を示します。 グループがライブでない場合は空白。

V 9.4.2 **ALTDATE**
この情報が最後に更新された日付を示します。

V 9.4.2 **ALTTIME**
この情報が最後に更新された時刻を示す。

手順

- キュー・マネージャーの一部であるポッドを見つけます。

```
oc get pod --selector app.kubernetes.io/instance=nativeha-qm
```

- いずれかのポッドで dspmq を実行します。

```
oc exec -t Pod dspmq
```

```
oc rsh Pod
```

対話式シェルの場合は、dspmq を直接実行できる場所です。

- キュー・マネージャー・インスタンスがアクティブ・インスタンスとして実行されているか、それともレプリカとして実行されているか判別するには、次のようにします

```
oc exec -t Pod dspmq -o status -m QMgrName
```

BOB という名前のキュー・マネージャーのアクティブ・インスタンスからは、次の状況が報告されます

```
QMNAME(BOB)                STATUS(Running)
```

V 9.4.2 ネイティブ HA CRR 構成のリカバリグループリーダーである BOB のインスタンスは、以下のステータスを報告する：

```
QMNAME(BOB)                STATUS(Recovery group leader)
```

BOB という名前のキュー・マネージャーのレプリカ・インスタンスからは、次の状況が報告されます

```
QMNAME(BOB)                STATUS(Replica)
```

非アクティブ・インスタンスからは、次の状況が報告されます

```
QMNAME(BOB)                STATUS(Ended Immediately)
```

- 指定されたポッド内のインスタンスのネイティブ HA 運用状況を判別するには、次のようにします

```
oc exec -t Pod dspmq -o nativeha -m QMgrName
```

BOB という名前のキュー・マネージャーのアクティブ・インスタンスからは、次のような状況が報告されます

```
QMNAME(BOB)         ROLE(Active) INSTANCE(inst1) INSYNC(Yes) QUORUM(3/3) V 9.4.2
GRPLSN(<0:25:365:1000>) GRPNAME() GRPROLE(Live)
```

BOB という名前のキュー・マネージャーのレプリカ・インスタンスからは、次のような状況が報告されます

```
QMNAME(BOB)         ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3) V 9.4.2
GRPLSN(<0:25:365:1000>) GRPNAME() GRPROLE(Live)
```

BOB という名前のキュー・マネージャーの非アクティブ・インスタンスからは、次のような状況が報告されます

```
QMNAME(BOB)         ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3) V 9.4.2
GRPLSN() GRPNAME() GRPROLE(Live)
```

V 9.4.2 Native HA CRR 構成の Live グループに属する BOB という名前のキューマネージャのアクティブなインスタンスは、以下のステータスを報告するかもしれません：

```
QMNAME(BOB)         ROLE(Active) INSTANCE(inst1) INSYNC(yes) QUORUM(3/3)
GRPLSN(<0:25:365:1000>) GRPNAME(alpha) GRPROLE(Live)
```

V 9.4.2 グループが回復グループになるのを待っているとき、同じインスタンスが次のようなステータスを報告するかもしれない：

```
QMNAME(BOB)         ROLE(Active) INSTANCE(inst1) INSYNC(yes) QUORUM(3/3)
GRPLSN(<0:25:365:1000>) GRPNAME(alpha) GRPROLE(Pending recovery)
```

V 9.4.2 Native HA または Native HA CRR 構成のインスタンスではない CAROL という名前のキューマネージャは、以下のステータスを報告します：

```
QMNAME(CAROL)       ROLE(Not configured) INSTANCE() INSYNC() QUORUM() GRPLSN() GRPNAME()
GRPROLE(Not configured)
```

- ネイティブ HA 構成内のすべてのインスタンスのネイティブ HA 運用状況を判別するには、次のようにします

```
oc exec -t Pod dspmq -o nativeha -x -m QMgrName
```

例えば、キューマネージャー BOB のアクティブなインスタンスが実行されているノードでこのコマンドを発行した場合、すべてのインスタンスが同期されているため、以下のステータスが返される可能性があります

```
QMNAME(BOB)         ROLE(Active) INSTANCE(inst1) INSYNC(yes) QUORUM(2/3)
GRPLSN(<0:25:365:1000>) GRPROLE(Live) GRPNAME()
 INSTANCE(inst1) ROLE(Active)  REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN(<0:26:365:1000>)  HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN (<0:24:365:1000>) HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN (<0:25:365:1000>) HASTATUS(Synchronizing)
SYNCTIME(2023-12-20T17:13:22.123321Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
```

別の例として、遅れをとっているキューマネージャー BOB のレプリカインスタンスを実行しているノードでこのコマンドを発行すると、以下のステータスが返される場合があります

```
QMNAME(BOB)         ROLE(Replica) INSTANCE(inst2) INSYNC(no) QUORUM(2/3)
GRPLSN(<0:25:365:1000>) GRPROLE(Live) GRPNAME()
 INSTANCE(inst1) ROLE(Active)  REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN(<0:26:365:1000>)  HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst3) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
CONNINST(yes) ACKLSN (<0:24:365:1000>) HASTATUS(Normal)
SYNCTIME(2023-12-20T17:13:27.853412Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
 INSTANCE(inst2) ROLE(Replica) REPLADDR(localhost) CONNACTV(yes) INSYNC(no) BACKLOG(435)
CONNINST(yes) ACKLSN (<0:25:365:1000>) HASTATUS(Synchronizing)
SYNCTIME(2023-12-20T17:13:22.123321Z) ALTDATE(2023-12-20) ALTTIME(17.13.27)
```

キュー・マネージャー BOB の非アクティブ・インスタンスを実行しているノード上でこのコマンドを発行すると、以下のような状況が表示されます

```
QMNAME(BOB)           ROLE(Unknown) INSTANCE(inst3) INSYNC(no) QUORUM(0/3) GRPLSN()
GRPROLE() GRPNAME()
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ACKLSN()  HASTATUS(Unknown) SYNCTIME() ALTDATE() ALTTIME()
ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ACKLSN()  HASTATUS(Unknown) SYNCTIME() ALTDATE() ALTTIME()
ALTDATE() ALTTIME()
```

```
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No)      INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ACKLSN()  HASTATUS(Unknown) SYNCTIME() ALTDATE() ALTTIME()
ALTDATE() ALTTIME()
```

どのインスタンスがアクティブでどれがレプリカになるかをまだネゴシエーションしている間にコマンドを発行すると、次の状況が表示されます

```
QMNAME(BOB)               STATUS(Negotiating)
```

- V 9.4.2
  ネイティブ HA CRR 構成内のすべてのネイティブ HA グループの稼動状況を確認する：

  ```
  oc exec -t Pod dspmq -o nativeha -g -m QMgrName
  ```

  Live グループのメンバーで alpha という名前のユーザーにこのコマンドを発行すると、alpha と Recovery グループの beta のステータスが次のようになっている可能性があります

  ```
  GRPNAME(alpha) GRPROLE(Live) GRPADDR(9.20.135.129) GRPVER(09040200)
  GRSTATUS(Normal) RCOVLSN(<0:0:104:65163>) RCOVTIME(2023-12-20T17:37:27.114787Z)
  INITLSN(<0:0:101:352>) INITTIME(2023-12-19T09:13:27.512294)
  LIVETIME(2023-12-19T06:14:27.512294) ALTDATE(2023-12-20) ALTTIME(17.13.27)
  ```

  ```
  GRPNAME(beta) GRPROLE(Recovery) GRPADDR(9.20.135.130) GRPVER(09040300)
  CONNGRP(yes) GRSTATUS(Synchronizing) RCOVLSN(<0:0:19:16723>)
  RCOVTIME(2023-12-20T17:36:23.463209Z) BACKLOG(5619) INSYNC(no) SYNCTIME()
  ALTDATE(2023-12-20) ALTTIME(17.13.27)
  ```

関連資料

dspmq (キュー・マネージャーの表示) コマンド

V 9.4.2 ネイティブ HA CRR の切り替えとフェイルオーバー

ネイティブ HA クロスリージョンレプリケーション（CRR）構成により、キューマネージャーの実行を別のリージョンに切り替えることができます。

ネイティブ HA CRR 構成を使用して、災害復旧ソリューションを提供することができます。 ライブサイトで何らかの災害が発生した場合、フェイルオーバーを実行して、リカバリサイトでキューマネージャーを継続して実行することができます。 このような状況では、フェイルオーバーが発生した際にリカバリログが同期されていることを保証できないため、データの損失が発生する可能性があります。 同様に、障害が発生したサイトが復旧した際には、キューマネージャーの 2 つの稼働インスタンスが存在することになり、パーティショニング（スプリットブレイン）されたデータが発生する可能性があります。

Native HA CRR 構成を使用すると、メンテナンスやテストを実行しながら継続性を確保することができます。 このような状況下では、データの損失が発生しないように切り替えが制御されます。 同様に、切り替える際には解決すべき分割されたデータはありません。

V 9.4.2 ネイティブ HA CRR 構成における計画された切り替えを完了する

ネイティブ HA クロスリージョンレプリケーション（CRR）構成では、ライブグループとリカバリグループ間の計画的な切り替えを実行できます。

このタスクについて

設定をテストしたい場合やメンテナンスを適用する必要がある場合は、ライブグループからリカバリグループに切り替える必要があるかもしれません。

計画的な切り替えを行う際には、データが失われたり、パーティショニング（スプリットブレイン）の問題が発生しないように、システムが適切な措置を講じます。 グループごとにグループの役割を変更することで、切り替えをリクエストします。 しばらくの間、両方のグループは保留状態となり、復旧ログが同期化されます。 これが達成されると、グループは新しい役割を担い、新しいライブグループで新しいリーダーが選出され、アクティブな役割を担いキューマネージャーを実行します。

手順

1. qm.ini ファイルの NativeHALocalInstance スタンザで、 GroupRole を Recovery に設定することにより、ライブグループをリカバリグループにするよう指定します。

   MQ オペレータを使用している場合、 GroupRole は QueueManager YAML の .spec.queueManager.availability.nativeHAGroups.local.role フィールドの設定により変更されます。 例えば、ライブグループの役割をリカバリに変更するには、現在ライブ中のグループの .spec.queueManager.availability.nativeHAGroups.local.role フィールドの値を Recovery に設定します。

   切り替えが成功したことを確認するには、 **oc get pods** コマンドを使ってポッドの状態を取得します。1 つのポッドはレディ状態になっているはずです：

   ```
   NAME                  READY   STATUS    RESTARTS   AGE
   exampleqm-ibm-mq-0   1/1     Running   0          2m45s
   exampleqm-ibm-mq-1   0/1     Running   0          2m45s
   exampleqm-ibm-mq-2   0/1     Running   0          2m45s
   ```

   以下のコマンドを実行して、回復グループのステータスを確認する：

   ```
   oc exec -t <leader-pod> -- dspmq -o nativeha -g
   ```

   ここで<leader-pod>はレディ 1/1 ポッドの名前である（例では exampleqm-ibm-mq-0 ）：

   GRPROLE が Pending recovery であることを確認する。
2. qm.ini ファイルの NativeHALocalInstance スタンザで、 GroupRole を Live に設定することにより、リカバリグループをライブグループにするよう指定します。

   MQ オペレータを使用している場合、 GroupRole は QueueManager YAML の .spec.queueManager.availability.nativeHAGroups.local.role フィールドの設定により変更されます。 例えば、Live グループの役割を Live に変更するには、現在リカバリ中のグループの .spec.queueManager.availability.nativeHAGroups.local.role フィールドの値を Live に設定します。

   グループ内の各インスタンスが再起動した後で、このキュー・マネージャーに対して **dspmq -o nativeha -g** コマンドを発行すると、GRPROLE が Pending live と表示されます。

   グループ内の各インスタンスが再起動したら、以下のコマンドを実行する：

   ```
   oc exec -t <active-pod> -- dspmq -o nativeha -g
   ```

   ここで<active-pod>は、 oc get pods コマンドで表示されるレディ 1/1 ポッドの名前である。 GRPROLE が Pending live 、グループが同期した後に live 。
3. 本番サイトとリカバリサイトのリカバリログが同期されると、リカバリグループが本番サイトとなり、アクティブなキューマネージャーインスタンスとなるリーダーが選出されます。 ここから、アプリケーションは自動的にこのインスタンスに再接続し、すべての作業負荷は通常通り再開されます。

次のタスク

メンテナンスが完了したら、元のライブグループに切り替えるために、同じ手順を繰り返します。

V 9.4.2 ネイティブ HA CRR 構成で予定外のフェイルオーバーを完了する

ライブサイトで災害が発生し、運用できなくなった場合、ネイティブ HA クロスリージョンレプリケーション（CRR）構成で、ライブグループとリカバリグループ間で計画外のフェイルオーバーを実行します。

このタスクについて

ライブグループがダウンした場合、レプリケーションログデータの受信が停止するだけで、リカバリグループはこれを検知できないため、これは進行中のワークロードがないことを意味する可能性がある。 したがって、計画外のフェールオーバーは、リカバリグループの役割をライブに変更し、以前ライブだったグ

ループが使用不可になったことを通知することで、データがそのグループにレプリケートされないようにする必要があります。

ログデータが非同期で複製されるため、リカバリグループとライブグループが同期していない可能性があります。 リカバリグループがライブグループと同期していない場合、予期せぬフェールオーバーが発生するとデータが失われます。 予定外のフェールオーバーを開始する前に、リカバリグループ上で **dspmq -o nativeha -g** コマンドを発行し、RCOVLSN と RCOVTIME の値を使用して、そのグループが回復可能なログシーケンス番号（LSN）と時刻を表示します。

手順

1. アプリケーションがライブグループから切断され、現在リカバリグループとなっているグループに再接続するように設定されていることを確認します。
2. 次の手順を完了することで、リカバリグループがライブグループになるように指定します

   a. `qm.ini` ファイルの `NativeHALocalInstance` の節で、`GroupRole` を `Live` に設定します。 例えば、 MQ オペレータを使用している場合は、 `.spec.queueManager.availability.nativeHAGroups.local.role` の設定を `Live` に変更します。

   b. `qm.ini` ファイルの `NativeHARecoveryGroup` スタンザで、`Enabled` フィールドを `False` に設定します。 例えば、 MQ オペレータを使用している場合は、 `.spec.queueManager.availability.nativeHAGroups.remotes.enabled` の設定を `false` に変更します。

   グループ内の各インスタンスが再起動した後で、このキューマネージャに対して **dspmq -o nativeha -g** コマンドを発行すると、GRPROLE が `Live` と表示されます。 グループメンバーの間で選挙が行われ、そのうちの 1 つがアクティブインスタンスとなり、キューマネージャの作業を処理します。
3. フェイルオーバー後にレプリケーションを再開するには、元のライブグループをリカバリグループとして使用するか、新しいサイトに新しいリカバリグループを作成します。

   - 元のライブサイトが復元された場合、元のライブグループを再起動する前に、グループの役割をリカバリに変更します。 これにより、グループがライブ状態に復帰しないようにします。
   - 元のライブサイトを復元できない場合は、新しいサイトで新しいネイティブ HA リカバリグループを作成します。

   リカバリグループを設定するには、238 ページの『予期せぬフェールオーバー発生後のレプリケーションの復元』 に記載されている手順に従ってください。
4. オリジナルのライブサイトの構成を変更せず、リストア後にグループがライブの役割を再開した場合、キューマネージャーのアクティブなインスタンスが 2 つ存在することになります。 この場合、分割された状態が検出され、2 つのグループ間でのレプリケーションは行われなくなります。 元のライブロールとリカバリロールを復元する前に、保持するデータを決定する必要があります。 ガイダンスについては、239 ページの『パーティショニング（スプリットブレイン）問題の解決』を参照してください。

関連情報

dspmq (キュー・マネージャーの表示)

V 9.4.2 予期せぬフェールオーバー発生後のレプリケーションの復元

予期せぬフェールオーバーが発生した後、ライブグループからリカバリグループへのレプリケーションを復元する。

このタスクについて

予期せぬフェールオーバーが発生した後、元のライブサイトが復元された場合、そのサイトをリカバリグループとして再構成することができます。 キュー・マネージャーが分割された（「スプリットブレイン」）状態になるのを防ぐため、元のライブサイトでネイティブ HA グループを再起動する前に、このタスクを完了させてください。

また、予期せぬフェールオーバー後に新しいリカバリグループを作成する際にも、オリジナルのライブサイトを復元できない場合は、これらの手順を完了することができます。

手順

1. 以下の手順を完了することで、グループをリカバリグループにすることを指定します
   a) qm.ini ファイルの NativeHALocalInstance スタンザで、GroupRole を Recovery に設定します。 例えば、 MQ オペレータを使用している場合は、 QueueManager YAML の .spec.queueManager.availability.nativeHAGroups.local.role フィールドの設定を Recovery に変更します。
   b) qm.ini ファイルの NativeHARecoveryGroup スタンザの Enabled フィールドを False に設定します。 例えば、 MQ オペレータを使用している場合は、 QueueManager YAML の .spec.queueManager.availability.nativeHAGroups.remotes.enabled フィールドの設定を false に変更します。

      各インスタンスが再起動した後に、グループのインスタンスに対して **dspmq -o nativeha -g** コマンドを発行すると、GRPROLE が Recovery と表示されます。 グループメンバーの中から選挙を行い、そのうちの 1 人が回復グループのリーダーとなります。
2. 本番用グループの qm.ini ファイルの NativeHARecoveryGroup スタンザで Enabled フィールドを True に設定し、本番用グループからリカバリ・グループへのレプリケーションを再開します。 例えば、MQ オペレータを使用している場合は、 QueueManager YAML の .spec.queueManager.availability.nativeHAGroups.remotes.enabled フィールドの設定を true に変更します。

V 9.4.2 パーティショニング（スプリットブレイン）問題の解決

ネイティブ HA クロスリージョンレプリケーション（CRR）構成のキューマネージャーが同時に両方の HA グループで実行されると、分割された、または「スプリットブレイン」状態が発生することがあります。

この状態は、予期せぬフェールオーバーが発生した場合に発生することがあります。 ライブグループとの通信が失われたため、キューマネージャをリカバリグループのプライマリにし、実行を開始します。 一方、キューマネージャーは依然として元の Live グループ上で実行されています。 古い Live グループがまだログを更新できる可能性があり、これにより、両方のグループが管理するデータに解決できない分岐が生じます。 グループのデータを破棄しなければ、再結合できません。

パーティションされた状態を解決するには、どのグループのデータを保持し、どのデータを削除するかを決定します。 予期せぬフェールオーバーがトリガーされた際にグループ間で同期が取れていなかった可能性もありますし、アプリケーションの負荷が両方のグループで処理可能であった可能性もあります。「最善」のデータの定義は主観的な要素が大きいものです。 例えば、一方のグループがログに多くのデータを保存している場合、その多くは自動的に記録された IBM MQ オブジェクトのメディア画像である可能性が高いでしょう。一方、もう一方のグループは、より価値の高いビジネス取引を処理しているかもしれません。 いずれのグループが選択されたとしても、パーティショニングされた状態からの復旧においては、削除されるグループ上の複製されていないメッセージとトランザクションの結果は、データ損失の可能性を調整する際に役立つ可能性が高いでしょう。

IBM MQ ライブロールを持つグループが、ライブロールで実行中の別のグループに接続すると、分割された状態を検出します。 両方のグループで dspmq -o nativeha -g コマンドを発行すると、GRSTATUS(Partitioned) というステータスが返されます。 両グループ間のログデータの複製は中断されています。 Partitioned の状態は、グループがリカバリの役割を持つ別のグループに十分なログデータを複製できるまで報告され続けます。

どちらのグループを保持するか明白な選択肢ではない場合、また、予定外のフェイルオーバー時に両方のグループからログエクステントがまだ利用可能な場合は、ログの内容を比較することができます。

dspmq -o nativeha -g コマンドを使用して、グループがライブになったログ内の正確なポイントの LSN（ログシーケンス番号）を特定します。 LSN は INITLSN（*nnnnn*:*nnnnn*:*nnnnn*:*nnnnn*）の形式で返されます。 このイベントの時刻も、INITTIME 属性で報告されます。 次に、**dmpmqlog -s** *startLSN* コマンドを使用して、その時点からのログの内容を調べます。 **dspmq** と **dmpmqlog** を組み合わせて使用し、一方のグループによって作成されたログ記録がもう一方のグループに複製されていないものを特定し

ます。 **dmpmqlog** コマンドは、メッセージングとトランザクションのアクティビティの概要も報告します。これは選択を行う際に役立つ場合があります。

例

グループ「アルファ」と「ベータ」は、ライブとリカバリ構成で最初に設定されました。 「ベータ」グループをリカバリからライブに変更するという決定が下され、予定外のフェールオーバーが発生しました。しかし、「アルファ」グループは、「ベータ」に複製されていない一部のアプリケーションの作業負荷を進めることができました。 その間、一部のアプリケーションは「ベータ版」に接続し、一部のメッセージング処理を実行しました。 「ベータ」グループの INITLSN は、「ベータ」グループが「アルファ」グループの内容から分岐したログ上のポイントを示しています。

キューマネージャーのデータのコピーは比較プロセスに必要であり、キューマネージャーは非アクティブにする必要があります。 これは、Kubernetes 環境でキュー・マネージャーのステートフルセット・サイズをゼロに縮小するか、あるいは永続ボリュームのスナップショットなどを使ってデータのコピーを作成することで実現できます。

ログデータの分析が完了し、破棄するグループデータの選択が完了したら、グループの役割を「Recovery」に変更し、キューマネージャーの永続ボリュームをすべて削除するか、または「Recovery」グループの情報を削除することで破棄できます。

関連情報

dspmq

dmpmqlog

OpenShift MQ Adv. ネイティブ HA キュー・マネージャーの終了

コンテナ内の IBM MQ では、**endmqm** コマンドを使用して、Native HA グループの一部であるアクティブまたはレプリカのキュー・マネージャを終了できます。

始める前に

注 : この情報はコンテナ環境のみに適用される。

このタスクについて

Native HA グループの一部であるキュー・マネージャを停止する手順は、アクティブ・インスタンスかレプリカ・インスタンスかによって異なります。 どちらのタイプのインスタンスを終了する場合も、インスタンスを終了してもネイティブ HA グループのクォーラムが壊れないようにチェックが行われます。 もしクォーラムが壊れてしまうなら、**endmqm** コマンドは失敗する。

endmqm コマンドを発行すると、グループ内の他のインスタンスは、接続が切れたときにエラーを報告しないように、これが起こっていることを警告されます。

あまりにも多くのレプリカ・インスタンスが終了したり切断されたりしてアクティブ・インスタンスがクォーラムを失った場合、アクティブ・インスタンスは完全に終了する前に設定可能な時間だけ待機します。これにより、アプリケーションの接続が切断されるだけでなく、処理を優雅にシャットダウンする期間ができる。 このタイムアウト値は、`qm.ini` ファイルの `NativeHALocalInstance` スタンザ内の `QuorumConnectivityTimeout` 属性で指定できます。 デフォルト値は 0 秒です。

手順

- キュー・マネージャのアクティブ・インスタンスを終了するには、アクティブ・インスタンスが実行されているノードで以下のコマンドを実行します：

  ```
  endmqm -s QMgrName
  ```

  - `-r` オプションを指定して、クライアント・アプリケーションが別のインスタンスに再接続できるようにします。

- このインスタンスが Native HA グループのアクティブ・インスタンスでない場合、コマンドは失敗します。
- このアクティブ・インスタンスを終了してグループ・クォーラムが失敗した場合、コマンドは失敗します。(このコマンドの実行と同時に他のインスタンスが終了するか、または使用できなくなった場合、クォーラム・チェックでこれが検出されない可能性があります。その場合、Native HA グループは終了し、十分な数のインスタンスが使用可能な場合にのみ再起動できます。)

アクティブなキュー・マネージャが終了すると、レプリカ・インスタンスの 1 つがアクティブな役割を引き継ぎます。 どのレプリカが引き継ぐかはグループ内の交渉によって決定され、どちらが最新のトランザクションログを持っているかに依存します。

- キュー・マネージャのレプリカ・インスタンスを終了するには、以下のコマンドを実行する：

```
endmqm -x QMgrName
```

- このインスタンスがアクティブ・インスタンスである場合、コマンドは失敗する。
- このレプリカ・インスタンスを終了してグループ・クォーラムが失敗した場合、コマンドは失敗します。(このコマンドの実行と同時に他のインスタンスが終了するか、または使用できなくなった場合、クォーラム・チェックでこれが検出されない可能性があります。その場合、Native HA グループは終了し、十分な数のインスタンスが使用可能な場合にのみ再起動できます。)

注：また、-c、-i、-p、または-w スイッチは、Native HA インスタンスがどのロールに属しているかに関係なく、**endmqm** コマンドで使用できます。 キューマネージャインスタンスは，グループクォーラムへの影響を無視して終了します。 しかし，情報はグループ内の他のインスタンスと共有される。 これらのスイッチは、アクティブ・インスタンスの-s と一緒に使うことができる。 これらのスイッチを-x スイッチと一緒にレプリカ・インスタンスに使用することはできません。

コンテナー内の IBM MQ のトラブルシューティング

コンテナーでの IBM MQ の実行に問題がある場合は、ここで説明する手法を使用して、問題の診断と解決に役立てることができます。

手順

- 241 ページの『コンテナー内の IBM MQ の計画外の再始動のトラブルシューティング』.
- 243 ページの『IBM MQ Operator に関する問題のトラブルシューティング』.

OpenShift CP4I Kubernetes コンテナー内の IBM MQ の計画外の再始動のトラブルシューティング

Red Hat OpenShift Container Platform や Kubernetes などのほとんどのコンテナー管理システムでは、通常、コンテナーが再始動されます。 コンテナーが長期間存続することは通常ではありません。 このトピックでは、コンテナーのライフサイクル、再始動の調査方法、および計画外のコンテナー再始動の背後にある理由について説明します。

IBM MQ デプロイメントに問題がなく、引き続き予期したとおりに実行される場合は、ソリューションが意図したとおりに実行されている可能性があります。 コンテナー・ログに次のようなログ・メッセージが表示されることがあります。

```
Signal received: terminated
```

これは、 MQ コンテナに SIGTERM シグナルが送信され、終了が要求されたことを意味します。 Linux コンテナは、 シグナルに応答する責任があります。これは、プログラムに送信される標準化されたメッセージで、動作をトリガーします。 POSIX SIGTERM シグナルは、 Kubernetes （したがって Red Hat OpenShift Container Platform ）によってコンテナを正常に終了するために使用されます。

IBM MQ コンテナーは、SIGTERM シグナルを受信すると、 `endmqm -w -r -tp` コマンドを発行してキュー・マネージャーを停止します。 キュー・マネージャーが停止すると、コンテナーは停止します。 キュー・マネージャーの停止に長時間かかる場合は、SIGKILL シグナルが送信される可能性があります。これ

により、 Linux プロセスが即時に終了します。 SIGTERM と SIGKILL の間の時間の長さは、 Kubernetes では「終了猶予期間」と呼ばれ、 `QueueManager` リソース (IBM MQ Operator を使用している場合) またはポッド・リソースで直接構成可能です。 デフォルトは 30 秒で、そのうちの 1 秒はコンテナーをシャットダウンするために予約されており、残りは IBM MQ に指定されています。 例えば、デフォルトの場合、 `endmqm -w -tp 29` が発行されます。これにより、シャットダウンに 29 秒かかったことがキュー・マネージャーに通知されます。

コンテナ終了の理由

Kubernetes 既存の Pod 内のコンテナを再起動したり、Pod を削除したりすることができます。 Pod の「status」セクションには、これがコンテナの再起動であるかどうかを示すフィールドがあります。例えば:

```
status:
  containerStatuses:
  - restartCount: 2
    started: true
    state:
      running:
        startedAt: "2025-01-20T12:03:15Z"
  startTime: "2024-12-28T13:46:12Z"
```

この例では、コンテナが再起動され、コンテナが Pod よりもはるかに新しいことがわかります。 これは、1 つのコンテナが再起動したことを示しており、おそらくコンテナがライブネストプローブに失敗したことを意味します。 Kubernetes でライブネステストを設定し、コンテナが依然として健全であることを確認することができます。 IBM MQ Operator は、有効な実行状態を確認するために **dspmq** コマンドを呼び出すキュー・マネージャーの活性プローブをセットアップします。 キュー・マネージャーが正常な状態でない場合、またはプローブ自体の実行に時間がかかりすぎる場合、kubelet は失敗と見なします。 許容される障害数のしきい値は、 `QueueManager` リソース (IBM MQ Operator を使用している場合) またはポッド・リソースで直接構成できます。

コンテナの再起動が見られない場合は、おそらく Pod 全体が置き換えられていると思われます。 Kubernetes のドキュメントの「Pod の終了」 を参照してください。 Kubernetes 「ノード上のポッドが自主的または非自主的に終了するプロセス」を「ポッドの混乱」 および「立ち退き」という用語で表現しています。 ポッドが追い出される理由には、以下のような多くの理由があります。

- **ローリング再起動**。 IBM MQ Operator またはステートフルセットコントローラは、ロールアウトが必要な構成変更が行われた場合、キューマネージャーの Pod のローリングアップデートをトリガーすることができます。 例えば、イメージのバージョン、マウントされたファイル、環境変数、注釈、ラベルなどを変更します。
- **kubelet による終了**。 これには、以下のようないくつかの理由が考えられます。
  - ノードが (おそらくローリング・クラスター更新の一部として) シャットダウンされているため、ポッドを強制終了できます。
  - ポッドは、ノードの「圧力」(kubelet がノード上のリソースを再利用するためにポッドをプロアクティブに強制終了する) が原因で終了する可能性があります。 Kubernetes クラスター管理者は、クラスター間で異なる可能性がある排除しきい値を構成できます。
- **Kubernetes スケジューラーによる優先使用**。 これは、 Kubernetes スケジューラーが優先度の高いポッドを実行する必要がある場合に発生する可能性があります
- **テイント適用ノード**。 ノードは「テイント」にすることができ、テイントを許容しないポッドは排除されます。 テイントは、 Kubernetes 管理者が特定のノードからポッドを「反復」するために使用します。 例えば、 IBM MQ ポッドは、現在他のワークロード用に予約されている特別なハードウェアを持つノード上では実行されなくなります。
- **Eviction API を介した要求**。 これは、管理者がポッドを除去するために呼び出すことができます。
- **ポッド・ガーベッジ・コレクション**。 これは、ノードがサービス休止になった場合、または Kubernetes API を使用してノードが削除された場合に発生する可能性があります。

キュー・マネージャー・ポッドが除去された理由の判別

ポッドが強制排除された理由を理解するために役立つ可能性のある情報源には、以下のものがあります。

- **クラスター・イベント**。 例えば、 OpenShift Container Platform クラスター内のシステム・イベント情報の表示。
- **クラスター監査イベント**。 Red Hat OpenShift Container Platform での監査ログの表示を参照してください。
- **圧力がかかっているノード**。 CPU、ネットワーク、またはメモリーの圧力がかかっているノードを探します。 これはノード状況で確認できます。 表示するまでには、ノードに圧力がかかっていない可能性があることに注意してください。
- **Red Hat OpenShift Container Platform Monitoring** またはその他のモニター・メトリックは、ディスク待ち時間の問題などを示すことができる場合があります。 有用な Prometheus メトリックは、 `ibmmq_qmgr_log_write_latency_seconds` です。 この情報は、 MQ 統計のトピックから得られます。

関連情報

スケジューリング、優先使用、および回避に関する Kubernetes 資料

OpenShift CP4I IBM MQ Operator に関する問題のトラブルシューティング

IBM MQ Operator で問題が発生する場合、説明されている手法を使用して問題の診断と解決を実施します。

手順

- 243 ページの『IBM MQ Operator を使用してデプロイされたキュー・マネージャーのトラブルシューティング情報の収集』
- 245 ページの『トラブルシューティング: キュー・マネージャー・データへのアクセスの取得』

OpenShift CP4I IBM MQ Operator を使用してデプロイされたキュー・マネージャーのトラブルシューティング情報の収集

新しいサポート Case を提出する際に IBM サポートに提供する必要があるトラブルシューティング情報を収集する。

手順

1. クラウド・プロバイダー情報を収集します。

   これは、 Red Hat OpenShift クラスターをホストするクラウド・プロバイダーです (例えば、 IBM Cloud)。
2. アーキテクチャー情報を収集します。

   Red Hat OpenShift クラスターのアーキテクチャーは、以下のいずれかです。

   - Linux for x86-64
   - Linux on Power Systems (ppc64le)
   - Linux on IBM Z
3. IBM MQ デプロイメント情報を収集します。

   a) `bash/zsh` シェルを使用して、 Red Hat OpenShift クラスターにログオンします。

   b) 以下の環境変数を設定します。

   ```
   export QM=QueueManager_name
   export QM_NAMESPACE=QueueManager_namespace
   export MQ_OPERATOR_NAMESPACE=mq_operator_namespace
   ```

   ここで *QueueManager_name* は QueueManager リソースの名前です (キューマネージャ YAML では `metadata.name`)、 *QueueManager_namespace* はそれが配置される名前空間 (キューマネージャ YAML では `metadata.namespace`) で、 *mq_operator_namespace* は IBM MQ Operator が配置される名前空間です。 これは QueueManager ネームスペースと同じかもしれない。

   c) 以下のコマンドを実行し、結果の出力ファイルをすべて IBM サポートに提供します。

```
# OCP / Kubernetes: Version
oc version -o yaml > ocversion.yaml

# QueueManager: YAML
oc get qmgr $QM -n $QM_NAMESPACE -o yaml > "queue-manager-$QM.yaml"

# MQ Queue Manager: Pods
oc get pods -n $QM_NAMESPACE -o wide --selector "app.kubernetes.io/instance=$QM" > "qm-
pods-$QM.txt"

# MQ Queue Manager: Pod YAML
oc get pods -n $QM_NAMESPACE -o yaml --selector "app.kubernetes.io/instance=$QM" > "qm-
pods-$QM.yaml"

# MQ Queue Manager: Pod Logs
for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/
instance=$QM" | cut -d ' ' -f 1); do oc logs -n $QM_NAMESPACE --previous "$p" > "qm-logs-
previous-$p.txt"; oc logs -n $QM_NAMESPACE $p > "qm-logs-$p.txt";done

# MQ Queue Manager: Describe Pods
for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/
instance=$QM" | cut -d ' ' -f 1); do oc describe pod $p -n $QM_NAMESPACE > "qm-pod-
describe-$p.txt"; done

# MQ Web UI: Console Log
for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/
instance=$QM" | cut -d ' ' -f 1); do oc cp -n $QM_NAMESPACE --retries=10 "$p:var/mqm/web/
installations/Installation1/servers/mqweb/logs/console.log" "web-$p-console.log"; done

# MQ Web UI: Messages Log
for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/
instance=$QM" | cut -d ' ' -f 1); do oc cp -n $QM_NAMESPACE --retries=10 "$p:var/mqm/web/
installations/Installation1/servers/mqweb/logs/messages.log" "web-$p-messages.log"; done

# MQ Queue Manager: routes defined by operator
oc get routes -n $QM_NAMESPACE -o yaml --selector "app.kubernetes.io/instance=$QM" > "qm-
routes-$QM.yaml"

# MQ Queue Manager: routes to QM
oc get routes -n $QM_NAMESPACE -o yaml --field-selector "spec.to.name=$QM-ibm-mq" > "qm-
routes2-$QM.yaml"

# MQ Queue Manager: stateful set
oc get statefulset -n $QM_NAMESPACE -o yaml ${QM}-ibm-mq > "qm-statefulset-$QM.yaml"

# MQ Queue Manager: revisions of the stateful set
oc get controllerrevisions.apps -o yaml -n $QM_NAMESPACE --selector "app.kubernetes.io/
instance=$QM" > "qm-statefulset-revisions-$QM.yaml"

# MQ Queue Manager: Pod events
for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/
instance=$QM" | cut -d ' ' -f 1); do oc get -o custom-columns="LAST
SEEN:.lastTimestamp,TYPE:.type,REASON:.reason,KIND:.involvedObject.kind,NAME:.involvedObje
ct.name,MESSAGE:.message" event -n $QM_NAMESPACE --field-selector
involvedObject.name="$p" > "qm-pod-events-$p.txt"; done

# MQ Queue Manager: StatefulSet events
oc get events -n $QM_NAMESPACE -o custom-columns="LAST
SEEN:.lastTimestamp,TYPE:.type,REASON:.reason,KIND:.involvedObject.kind,NAME:.involvedObje
ct.name,MESSAGE:.message" --field-selector involvedObject.name="${QM}-ibm-mq" > "qm-
statefulset-events-$QM.txt"

# MQ Queue Manager: services
oc get services -n $QM_NAMESPACE -o yaml --selector "app.kubernetes.io/instance=$QM" >
"qm-services-$QM.yaml"

# MQ Queue Manager: PVCs
oc get pvc -n $QM_NAMESPACE -o yaml --selector "app.kubernetes.io/instance=$QM" > "qm-
pvcs-$QM.yaml"

# MQ Operator: Version
oc get csv -n $QM_NAMESPACE | grep "^ibm-mq\|NAME" > mq-operator-csv.txt

# Cloud Pak Foundational Services: Version
oc get csv -n $QM_NAMESPACE | grep "^ibm-common-service-operator\|NAME" > common-services-
csv.txt

# Cloud Pak for Integration: Version (if applicable)
oc get csv -n $QM_NAMESPACE | grep "^ibm-integration-platform-navigator\|NAME" > cp4i-
csv.txt
```

```
# Output from runmqras (this may take a while to execute)
for p in $(oc get pods -n $QM_NAMESPACE --no-headers --selector "app.kubernetes.io/
instance=$QM" | cut -d ' ' -f 1); do timestamp=$(TZ=UTC date +"%Y%m%d_%H%M%S"); oc exec
-n $QM_NAMESPACE $p -- runmqras -workdirectory "/tmp/runmqras_$timestamp" -section
logger,mqweb,nativeha,trace; oc cp -n $QM_NAMESPACE --retries=10 "$p:tmp/
runmqras_$timestamp/" .; done

# MQ Operator: Pod Log
oc logs -n $MQ_OPERATOR_NAMESPACE $(oc get pods -n $MQ_OPERATOR_NAMESPACE --no-headers --
selector app.kubernetes.io/name=ibm-mq,app.kubernetes.io/managed-by=olm | cut -d ' ' -f
1) > mq-operator-log.txt
```

注:

- これらのコマンドの大部分は、キュー・マネージャーがデプロイされている名前空間へのアクセス権限を必要とします。 ただし、IBM MQ Operator が **cluster-scoped** にインストールされている場合は、 IBM MQ Operator ログを収集するために、 **クラスター管理者** アクセス権限が追加で必要になることがあります。
- 実行中のキュー・マネージャがある場合は、これらのコマンドによって生成される出力ファイルが空でないことを確認してください（ `qm-logs-previous-`*`pod_name`*`.txt` のポッドログを除き、コンテナが再起動しなかった場合は空であることが期待されます)。

関連タスク

トラブルシューティング情報の収集

OpenShift CP4I トラブルシューティング: キュー・マネージャー・データへのアクセスの取得

PVC インスペクター・ツールを使用して、キュー・マネージャー・ポッドに対してリモート・シェルを確立できないキュー・マネージャー PVC 上のファイルにアクセスできるようにします。 これは、ポッドが **Error** 状態または **CrashLoopBackOff** 状態であることが原因である可能性があります。 このツールは、IBM MQ Operator によってデプロイされたキュー・マネージャーで使用するために設計されています。

始める前に

PVC Inspector ツールを使用する場合。 キュー・マネージャーの名前空間へのアクセス権限が必要です。

このタスクについて

トラブルシューティングを支援するために、特定のキュー・マネージャーに関連付けられた永続ボリューム要求 (PVC) に保管されているデータにアクセスできます。 これを行うには、ツールを使用して、一連のインスペクター・ポッドに PVC をマウントします。 その後、リモート・シェルをインスペクター・ポッドのいずれかに取得して、ファイルを読み取ることができます。

デプロイメントのタイプに応じて、1 つから 3 つのインスペクター・ポッドが作成されます。 Native-HA キュー・マネージャーまたは複数インスタンス・キュー・マネージャーの特定のポッドに固有のボリュームは、関連付けられた PVC インスペクター・ポッドで使用可能です。 共有ボリュームはすべてのインスペクターで使用可能です。 インスペクター・ポッドの名前には、関連付けられたキュー・マネージャー・ポッドの名前が含まれます。

手順

1. MQ PVC インスペクター・ツールをダウンロードします。

   このツールは、 https://github.com/ibm-messaging/mq-pvc-tool から入手できます。
2. クラスターにログインしていることを確認します。
3. キュー・マネージャーの名前と、キュー・マネージャーが実行されている名前空間を調べます。
4. キュー・マネージャーに対してインスペクター・ツールを実行します。

   a) キュー・マネージャー名とその名前空間名を指定して、以下のコマンドを実行します。

```
./pvc-tool.sh queue_manager_name queue_manager_namespace_name
```

b) ツールが完了したら、以下のコマンドを実行して、作成されるインスペクター・ポッドを表示します。

```
oc get pods
```

5. インスペクター・ポッドにマウントされたファイルを表示します。
   a) 各 PVC インスペクター・ポッドはキュー・マネージャー・ポッドに関連付けられるため、複数のインスペクター・ポッドが存在する可能性があります。 以下のコマンドを実行して、これらのポッドのいずれかにアクセスします。

   ```
   oc rsh pvc-inspector-pod-name
   ```

   マウントされた PVC ディレクトリーが入っているディレクトリーに置かれます。
   b) 以下のコマンドを実行して、PVC ディレクトリーをリストします。

   ```
   ls
   ```

   c) リモート・シェル・セッションの外部で以下のコマンドを実行して、PVC のリストを表示します。

   ```
   oc get pvc
   ```

   d) 以下のコマンドを実行して、ツールによって作成されたポッドをクリーンアップします。

   ```
   oc delete pods -l tool=mq-pvc-inspector
   ```

OpenShift CP4I Troubleshooting container clusters with the OpenShift Monitoring CLI

As a troubleshooting aid, you can query the available container cluster metrics using the OpenShift Monitoring CLI.

Before you begin

OpenShift Monitoring provides a web console and a CLI. For day-to-day operations you might prefer to use the console, as described in “Monitoring container clusters with the OpenShift Monitoring console” on page 219. For gathering troubleshooting information you might prefer to use the CLI, as described in the rest of this task.

About this task

OpenShift Container Platform includes a pre-configured, pre-installed and self-updating monitoring stack that uses an open source Prometheus server. Queue managers running under an IBM MQ Operator can produce metrics compatible with Prometheus. For a complete list of the metrics available for IBM MQ, see “IBM MQ コンテナが公開するメトリクス” on page 221.

OpenShift Monitoring is the default cluster monitoring method. It is installed with the cluster. To query the available container cluster metrics using the OpenShift Monitoring CLI, complete the following steps.

Procedure

1. Enable OpenShift Monitoring for the operands deployed in an IBM MQ container cluster.
   a) Log in to the cluster as a cluster administrator.
   b) Add "`enableUserWorkload: true`" to the configMap `cluster-monitoring-config` in the `openshift-monitoring` namespace. To do this, follow the steps in Enabling monitoring for user-defined projects in the Red Hat OpenShift documentation. If the `cluster-monitoring-config` ConfigMap does not exist in the `openshift-monitoring` namespace, follow the instructions to create it.
2. Install the OpenShift command line tool (oc).

See Getting started with the OpenShift CLI.

3. Query the IBM MQ metrics using the OpenShift Monitoring CLI.

   When using the CLI, first declare the variables by using the **export** command. For example:

```
export OCP_LOGIN_TOKEN=$(oc whoami -t)
export THANOS_QUERIER_HOST=$(oc -n openshift-monitoring get route thanos-querier -o
jsonpath={.spec.host})
export QM_NAMESPACE=namespace-in-which-the-target-queue-manager-is-installed
export QM_METRIC_NAME=IBM-MQ-metric
```

   To run PromQL queries from the CLI, use the **PROMQL_QUERY** command. For example:

   - Get a list of queue managers that have a positive FDC count:

```
export QM_METRIC_NAME=ibmmq_qmgr_fdc_files
export PROMQL_QUERY="$QM_METRIC_NAME{namespace=\"$QM_NAMESPACE\"} > 0"
```

   - Get data for a specific time period:

```
export START=start_time
export END=end_time
export STEP=step_interval
curl -G -s -k -H "Authorization: Bearer $OCP_LOGIN_TOKEN" --data-urlencode
"query=$PROMQL_QUERY" \
--data-urlencode "start=$START" --data-urlencode "end=$END" \
--data-urlencode "step=$STEP" "https://$THANOS_QUERIER_HOST/api/v1/query_range"
```

     where *start_time* and *end_time* specify the time in ISO 8601 format, and *step_interval* is (for example) 15m, or 30s, or 1h.

   You can also export the data to a CSV file:

   a. The following command uses the same headers that display in the web console output shown in the screen capture in "Monitoring container clusters with the OpenShift Monitoring console" on page 219. To export to a CSV file, and pass the same headers that display in the web console output, first download and install jq.

   b. Run the following command:

```
(
echo
"metric_name,container,endpoint,instance,job,namespace,pod,prometheus,qmgr,service,timesta
mp,value"
curl -G -s -k -H "Authorization: Bearer $OCP_LOGIN_TOKEN" --data-urlencode
"query=$PROMQL_QUERY" \
--data-urlencode "start=$START" --data-urlencode "end=$END" --data-urlencode "step=$STEP"
\
"https://$THANOS_QUERIER_HOST/api/v1/query_range"| jq -r '.data.result[] | .metric as
$metric | .values[] | \
[$metric.__name__,
$metric.container,
$metric.endpoint,
$metric.instance,
$metric.job,
$metric.namespace,
$metric.pod,
$metric.prometheus,
$metric.qmgr,
$metric.service,
(.[0] | strftime("%Y-%m-%dT%H:%M:%SZ")), /* converts the timestamp from milliseconds to
ISO 8601 format */
.[1] ] | @csv'
) > your_filename.csv
```

CLI example: Write latency of Queue manager pods over the last 5 hours with a resolution of 10 minutes

```
export QM_METRIC_NAME=ibmmq_qmgr_log_write_latency_seconds
export PROMQL_QUERY="$QM_METRIC_NAME{namespace=\"$QM_NAMESPACE\"} [5h:10m]"
```

Export the data to a CSV file:

```
(
echo
"metric_name,container,endpoint,instance,job,namespace,pod,prometheus,qmgr,service,timestamp,val
ue"
curl -G -s -k -H "Authorization: Bearer $OCP_LOGIN_TOKEN" \
--data-urlencode "query=$PROMQL_QUERY" \
"https://$THANOS_QUERIER_HOST/api/v1/query" | jq -r '.data.result[] | .metric as $metric
| .values[] | \
[$metric.__name__,
$metric.container,
$metric.endpoint,
$metric.instance,
$metric.job,
$metric.namespace,
$metric.pod,
$metric.prometheus,
$metric.qmgr,
$metric.service,
(.[0] | strftime("%Y-%m-%dT%H:%M:%SZ")), /* converts the timestamp from milliseconds to ISO
8601 format */
.[1] ] | @csv'
) > your_filename.csv
```

Related tasks

“IBM MQ Operator 使用時のモニター” on page 218
IBM MQ Operator が管理するキュー・マネージャーは、Prometheus と互換性のあるメトリックを生成できます。 キューマネージャーのメトリクスは、デフォルトで HTTP で有効化され、提供されています。

“Monitoring container clusters with the OpenShift Monitoring console” on page 219
You can query the available container cluster metrics using the OpenShift Monitoring web console.

Related reference

“IBM MQ コンテナが公開するメトリクス” on page 221
キューマネージャーコンテナは、Red Hat OpenShift モニタリングと互換性のあるメトリクスを公開することができます。

Kubernetes V 9.4.1 に配置されたキュー・マネージャのトラブルシューティング情報を収集する Kubernetes

新しいサポートケースを提起する際に IBM サポートに提供すべきトラブルシューティング情報を収集する。

手順

1. クラウドプロバイダーの情報を収集する。

   これは、Kubernetes クラスタをホストするクラウド・プロバイダーである（例えば、IBM Cloud
2. Kubernetes のバージョン情報を収集する
   例えば、次を実行します。 `kubectl version -o yaml > kubeversion.yaml`
3. IBM MQ どのように配置されるかの主要な要素を記述したリソース YAML を収集する。
   以下を含む：

   a. StatefulSet

   b. ポッド（各キューマネージャインスタンス）

   c. サービス

   d. PersistentVolumeClaim

   e. ConfigMaps

   例えば、各キュー・マネージャー・ポッドに対して `kubectl get pod PodName -o yaml` を実行する。
4. それぞれのポッドについて、出来事も含めて説明文を集める。
   例えば、各キュー・マネージャー・ポッドに対して `kubectl describe pod` を実行する。

5. すべてのポッドログを収集する
   例えば、各キュー・マネージャー・ポッドに対して kubectl logs *PodName* を実行する。
6. **runmqras** の出力を収集する
   各キュー・マネージャ・コンテナ内で **runmqras** コマンドを実行するには kubectl exec を使用する。

関連タスク
トラブルシューティング情報の収集

OpenShift CP4I コンテナー内の IBM MQ の参照情報

IBM MQ は、Red Hat OpenShift コンテナー・プラットフォームとのネイティブ統合を提供する、Kubernetes オペレーターを提供します。

OpenShift CP4I IBM MQ Operator のライセンスと API リファレンス

IBM MQ は Kubernetes オペレータを提供し、Red Hat OpenShift コンテナプラットフォームとのネイティブな統合を提供します。 v1beta1 API は、キューマネージャリソースの作成と管理に使うことができます。

mq.ibm.com/v1beta1: 現在のライセンスバージョン

spec.license.license フィールドには、同意しようとしているライセンスのライセンス ID が含まれていなければなりません。 有効な値は以下のとおりです。

spec.license.license の値	**spec.license.use の値**	**ライセンス情報**	**利用可能な IBM MQ のバージョン**
L-CYPF-CRPF3H	Production または NonProduction	IBM Cloud Pak for Integration 16.1.2	9.4.3
L-MQQP-KBWMYE	Production または NonProduction	IBM Cloud Pak for Integration 限定版 16.1.2	9.4.3
L-NUUP-23NH8Y	Production	IBM MQ Advanced 9.4 - 02/2025	9.4.3 または 9.4.2
L-QWBN-T3PKLE	NonProduction	IBM MQ Advanced for Non-Production Environment 9.4 - 02/2025	9.4.3 または 9.4.2
L-HYGL-6STWD6	Development	IBM MQ Advanced for Developers （無保証） - 10/2024 9.4	9.4.3 9.4.2 または 9.4.1
L-QYVA-B365MB	Production または NonProduction	IBM Cloud Pak for Integration 16.1.1	9.4.2 または 9.4.1
L-JVML-UFQVM4	Production または NonProduction	IBM Cloud Pak for Integration リミテッド・エディション 16.1.1	9.4.2 または 9.4.1
L-JTPV-KYG8TF	Production または NonProduction	IBM Cloud Pak for Integration 16.1.0	9.9.4.1 または 9.4.0
L-BMSF-5YDSLR	Production または NonProduction	IBM Cloud Pak for Integration リミテッド・エディション 16.1.0	9.9.4.1 または 9.4.0

spec.license.license の値	spec.license.use の値	ライセンス情報	利用可能な IBM MQ のバージョン
L-EHXT-MQCRN9	Production	IBM MQ Advanced 9.4	9.9.4.1 または 9.4.0
L-DRNQ-NGCA76	NonProduction	IBM MQ Advanced for Non-Production Environment 9.4	9.4.1
L-CLXQ-ADXTK3	Development	IBM MQ Advanced for Developers（警告なし） 9.4	9.9.4.1 または 9.4.0

ライセンスのバージョン を指定しますが、これは必ずしも IBM MQ のバージョンとは同じでないことに注意してください。

mq.ibm.com/v1beta1: 古いライセンスバージョン

IBM MQ 9.3 ドキュメントの Older license versions を参照してください。

OpenShift CP4I QueueManager (mq.ibm.com/v1beta1) の API リファレンス

QueueManager

QueueManager は、アプリケーションにキューイングとパブリッシュ/サブスクライブのサービスを提供する IBM MQ サーバーです。 IBM MQ ドキュメンテーション https://ibm.biz/BdPZqj. ライセンスの参照：https://ibm.biz/Bdm9be.

フィールド	説明
apiVersion 文字列	APIVersion は、このオブジェクトの表記のスキーマのバージョンを定義します。サーバーは、認識されたスキーマを最新の内部値に変換する必要があります。認識されない値は拒否されることがあります。 詳細情報: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources。
kind 文字列	kind は、このオブジェクトが表している REST リソースを表す文字列の値です。サーバーは、クライアントが要求を送信するエンドポイントからこれを推測することがあります。 更新することはできません。 キャメル・ケースの値です。詳細情報: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds。
metadata	
spec QueueManagerSpec	QueueManager について必要とする状態。
status QueueManagerStatus	QueueManager について観測された状態。

.spec

QueueManager について必要とする状態。

以下の中に含まれます:

- 250 ページの『QueueManager』

フィールド	説明
affinity	標準的な Kubernete アフィニティー・ルール。 詳しくは、https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.17/#affinity-v1-core を参照してください。

フィールド	説明
annotations Annotations	annotations フィールドは、ポッド・アノテーションのパススルーの役目を果たします。 ユーザーはこのフィールドにアノテーションを追加することによって、ポッドにアノテーションを適用できます。 ここでアノテーションを指定すると、デフォルトのアノテーションが上書きされます。 MQ Operator 1.3.0 以上が必要です。
imagePullSecrets LocalObjectReference 配列	(オプション) この QueueManager で使用するイメージをプルするために使用する、同じ名前空間にあるシークレットへの参照のリスト。 指定すると、それらのシークレットがプル実行プログラムに渡されて使用されます。 例えば、Docker の場合は、DockerConfig タイプのシークレットだけが適用されます。 詳しくは、https://kubernetes.io/docs/concepts/containers/images#specifying-imagepullsecrets-on-a-pod を参照してください。
labels Labels	labels フィールドは、ポッド・ラベルのパススルーの役目を果たします。 ユーザーはこのフィールドにラベルを追加することによって、ポッドにラベルを適用できます。 ここでラベルを指定すると、デフォルトのラベルが上書きされます。 MQ Operator 1.3.0 以上が必要です。
license License	ライセンスの受け入れを制御する設定、および使用するライセンス・メトリック。
pki PKI	Transport Layer Security (TLS) または MQ Advanced Message Security (AMS) で使用する鍵と証明書を定義するための Public Key Infrastructure の設定。
queueManager QueueManagerConfig	キュー・マネージャーのコンテナーおよび基礎キュー・マネージャーの設定。
securityContext SecurityContext	キュー・マネージャー・ポッドの securityContext に追加するセキュリティー設定です。
service サービス	すべての QueueManager サービスに適用される設定。 このセクションまたはその中のフィールドを変更または削除すると、すべての QueueManager サービスが再生成される場合があります。 サービス再生成により通信が中断され、NativeHA を含むすべての QueueManager タイプでダウンタイムが発生する。 再生成されたサービスには、 Kubernetes によって新しい ClusterIP アドレスが割り当てられる。 MQ Operator 3.6.0 以上が必要。
telemetry テレメトリー	Open Telemetry 構成の設定。 MQ Operator 2.2.0 以上が必要です。
template テンプレート	Kubernetes リソースの拡張テンプレート。 このテンプレートは、基礎の Kubernetes リソース (StatefulSet、Pod、Service など) を IBM MQ で生成する方法をユーザーがオーバーライドすることを可能にします。 これは上級者専用です。誤って使用すると MQ の正常な動作が阻害される可能性があります。 QueueManager リソースの他の場所で指定された値は、このテンプレートの設定によってオーバーライドされます。
terminationGracePeriodSeconds 整数	(オプション) ポットの正常な終了にかかる時間 (秒単位)。 値は負以外の整数でなければなりません。 ゼロの値は、ただちに削除することを意味します。 このキュー・マネージャーの終了の目標時間を達成するために、アプリケーションの切断のフェーズが押し進められます。 必要な場合は、キュー・マネージャーの重要なメンテナンス・タスクが中断されます。 デフォルトは 30 秒です。
tracing TracingConfig	Cloud Pak for Integration Operations Dashboard とのトレースの統合のための設定。
version 文字列	使用する MQ のバージョンを制御する設定 (必須)。 例えば、9.1.5.0-r2 は、コンテナー・イメージの 2 番目のリビジョンを使用する MQ バージョン 9.1.5.0 を指します。 ベース・イメージのフィックスなどのように、コンテナー固有のフィックスが、リビジョンの中で適用されることがよくあります。
web WebServerConfig	MQ Web サーバーの設定。

.spec.annotations

annotations フィールドは、ポッド・アノテーションのパススルーの役目を果たします。 ユーザーはこのフィールドにアノテーションを追加することによって、ポッドにアノテーションを適用できます。 ここでアノテーションを指定すると、デフォルトのアノテーションが上書きされます。 MQ Operator 1.3.0 以上が必要です。

以下の中に含まれます:

- 250 ページの『.spec』

.spec.imagePullSecrets

LocalObjectReference に、同じ名前空間の内部で参照されているオブジェクトを見つけることができる十分な情報が含まれています。

以下の中に含まれます:

- 250 ページの『.spec』

フィールド	説明
name 文字列	参照の名前。 このフィールドは事実上必須であるが、後方互換性のため、空でも許される。 ここに空の値を持つこの型のインスタンスは、ほぼ間違いない。 詳細はこちら： https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names. デフォルト：""。

.spec.labels

labels フィールドは、ポッド・ラベルのパススルーの役目を果たします。 ユーザーはこのフィールドにラベルを追加することによって、ポッドにラベルを適用できます。 ここでラベルを指定すると、デフォルトのラベルが上書きされます。 MQ Operator 1.3.0 以上が必要です。

以下の中に含まれます:

- 250 ページの『.spec』

.spec.license

ライセンスの受け入れを制御する設定、および使用するライセンス・メトリック。

以下の中に含まれます:

- 250 ページの『.spec』

フィールド	説明
accept ブール値	このソフトウェアに関連付けられているライセンスを受け入れるかどうか (必須)。
license 文字列	受け入れるライセンスの ID。 これは、使用する MQ のバージョンの正確なライセンス ID である必要があります。 有効な値は https://ibm.biz/Bdm9be を参照のこと。
metric 文字列	使用するライセンス・メトリックを指定する設定。 例えば、ProcessorValueUnit、 VirtualProcessorCore、ManagedVirtualServer、 FREE。 デフォルトは、MQ Advanced ライセンスまたは MQ Advancedfor Non-Production ライセンスを使用している場合は ProcessorValueUnit Cloud Pak for Integration ライセンスを使用している場合は VirtualProcessorCore MQ Advanced for Developers ライセンスを使用している場合は FREE です。

フィールド	説明
use 文字列	複数の使用方法をサポートするライセンスの場合に、ソフトウェアの使用方法を制御する設定。 有効な値は https://ibm.biz/Bdm9be を参照のこと。

.spec.pki

Transport Layer Security (TLS) または MQ Advanced Message Security (AMS) で使用する鍵と証明書を定義するための Public Key Infrastructure の設定。

以下の中に含まれます:

- 250 ページの『.spec』

フィールド	説明
keys PKISource 配列	キュー・マネージャーの鍵リポジトリーに追加する秘密鍵です。
trust PKISource 配列	キュー・マネージャーの鍵リポジトリーに追加する証明書です。

.spec.pki.keys

PKISource は、鍵や証明書などの Public Key Infrastructure 情報のソースを定義します。

以下の中に含まれます:

- 253 ページの『.spec.pki』

フィールド	説明
name 文字列	name は鍵や証明書のラベルとして使用されます。 小文字の英数字の文字列である必要があります。
secret Secret	Kubernetes シークレットを使用して鍵を渡します。

.spec.pki.keys.secret

Kubernetes シークレットを使用して鍵を渡します。

以下の中に含まれます:

- 253 ページの『.spec.pki.keys』

フィールド	説明
items 配列	キュー・マネージャーのコンテナーに追加する必要がある Kubernetes シークレットの内部にある鍵。
secretName 文字列	Kubernetes シークレットの名前。

.spec.pki.trust

PKISource は、鍵や証明書などの Public Key Infrastructure 情報のソースを定義します。

以下の中に含まれます:

- 253 ページの『.spec.pki』

フィールド	説明
name 文字列	name は鍵や証明書のラベルとして使用されます。 小文字の英数字の文字列である必要があります。
secret Secret	Kubernetes シークレットを使用して鍵を渡します。

.spec.pki.trust.secret

Kubernetes シークレットを使用して鍵を渡します。

以下の中に含まれます:

- 253 ページの『.spec.pki.trust』

フィールド	説明
items 配列	キュー・マネージャーのコンテナーに追加する必要がある Kubernetes シークレットの内部にある鍵。
secretName 文字列	Kubernetes シークレットの名前。

.spec.queueManager

キュー・マネージャーのコンテナーおよび基礎キュー・マネージャーの設定。

以下の中に含まれます:

- 250 ページの『.spec』

フィールド	説明
availability Availability	キュー・マネージャーの可用性の設定 (アクティブとスタンバイのペアやネイティブの高可用性を使用するかどうかなど)。
debug ブール値	コンテナー固有のコードからのデバッグ・メッセージをコンテナー・ログに記録するかどうか。 デフォルトは false です。
env Env 配列	キューマネージャーに環境変数を供給するための設定。 MQ オペレータ 3.5.0 以上が必要です。
files ファイル配列	キューマネージャーにファイルを供給するための設定。 MQ オペレータ 3.5.0 以上が必要です。
image 文字列	使用するコンテナー・イメージ。
imagePullPolicy 文字列	指定されたイメージのプルを Kubelet が試行するタイミングを制御する設定。 デフォルトは IfNotPresent です。
ini INISource 配列	キュー・マネージャーに INI を提供するための設定。 MQ Operator 1.1.0 以上が必要です。
livenessProbe QueueManagerLivenessProbe	Liveness プローブを制御する設定。
logFormat 文字列	このコンテナーで使用するログの形式。 JSON 形式のコンテナー・ログには、JSON を使用します。 テキスト形式のメッセージには、Basic を使用します。 デフォルトは Basic です。
metrics QueueManagerMetrics	Prometheus スタイルのメトリックの設定。
mqsc MQSCSource 配列	キュー・マネージャーに MQSC を提供するための設定。 MQ Operator 1.1.0 以上が必要です。
name 文字列	基礎 MQ キュー・マネージャーの名前 (metadata.name と異なる場合)。 名前の Kubernetes 規則に準拠しないキュー・マネージャー名 (例えば、大文字を含む名前) が必要な場合は、このフィールドを使用します。
readinessProbe QueueManagerReadinessProbe	Readiness プローブを制御する設定。

フィールド	説明
recoveryLogs 回復ログ	MQ リカバリー・ログの設定。 MQ Operator 2.4.0 以上が必要です。
resources Resources	リソース要件を制御する設定。
route Route	キュー・マネージャー・ルートの設定。 MQ Operator 1.4.0 以上が必要です。
startupProbe StartupProbe	始動プローブを制御する設定。 MultiInstance デプロイメントと NativeHA デプロイメントにのみ適用されます。 MQ Operator 1.5.0 以上が必要です。
storage QueueManagerStorage	キュー・マネージャーが永続ボリュームおよびストレージ・クラスを使用することを制御するストレージ設定です。

.spec.queueManager.availability

キュー・マネージャーの可用性の設定 (アクティブとスタンバイのペアやネイティブの高可用性を使用するかどうかなど)。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
nativeHAGroups ネイティブハグループ	ネイティブ HA グループの設定。 type of availability を NativeHA に変更する必要があります。 MQ オペレータ 3.5.0 以上が必要です。
tls Tls	NativeHA レプリカ間のセキュア通信を構成するためのオプションの TLS 設定。 MQ Operator 1.5.0 以上が必要です。
type 文字列	使用する可用性のタイプ。 Kubernetes が (一部の状況で) 自動的に再始動する単一ポッドには、SingleInstance を使用します。 MultiInstance は、ポッドのペア (1 つは active キュー・マネージャー、もう 1 つはスタンバイ) に使用します。 ネイティブの高可用性の複製には NativeHA を使用します (MQ Operator 1.5.0 以上が必要です)。 デフォルトは SingleInstance です。 参照 https://ibm.biz/Bdm9TW を参照のこと。
updateStrategy 文字列	MultiInstance および NativeHA キュー・マネージャーに使用する更新方針。 RollingUpdate を使用して、キュー・マネージャーの構成が変更されるたびに自動ローリング更新を有効にします。 自動ローリング更新を無効にするには、OnDelete を使用します。キュー・マネージャーの変更は、ポッドが削除された場合にのみ適用されます (外部要因によってトリガーされたポッド削除も含む)。 デフォルトは RollingUpdate です。 MQ Operator 1.6.0 以上が必要です。

.spec.queueManager.availability.nativeHAGroups

ネイティブ HA グループの設定。 type of availability を NativeHA に変更する必要があります。 MQ オペレータ 3.5.0 以上が必要です。

以下の中に含まれます:

- 255 ページの『.spec.queueManager.availability』

フィールド	説明
local ローカル	この NativeHA グループの設定。
remotes リモコン配列	リモート NativeHA グループへの接続設定。

.spec.queueManager.availability.nativeHAGroups ローカル

この NativeHA グループの設定。

以下の中に含まれます:

- 255 ページの『.spec.queueManager.availability.nativeHAGroups』

フィールド	説明
name 文字列	NativeHA グループの名称。
role 文字列	この NativeHA グループが取るべき役割。 ライブまたはリカバリのどちらか。 有効な値は "Live"; "Recovery" です。
route Route	NativeHA グループの経路の設定。
tls Tls	NativeHA グループ間の安全な通信を設定するための TLS 設定。 親の可用性セクションで TLS が設定されていない場合に必要です。

.spec.queueManager.availability.nativeHAGroups.local.route

NativeHA グループの経路の設定。

以下の中に含まれます:

- 256 ページの『.spec.queueManager.availability.nativeHAGroups ローカル』

フィールド	説明
enabled ブール値	ルートを有効にするかどうか。 デフォルトは true です。

.spec.queueManager.availability.nativeHAGroups.local.tls

NativeHA グループ間の安全な通信を設定するための TLS 設定。 親の可用性セクションで TLS が設定されていない場合に必要です。

以下の中に含まれます:

- 256 ページの『.spec.queueManager.availability.nativeHAGroups ローカル』

フィールド	説明
cipherSpec 文字列	NativeHA グループ TLS の CipherSpec 名。
key キー	Native HA グループ間のセキュアな通信で使用される TLS キーの設定。

.spec.queueManager.availability.nativeHAGroups.local.tls.key

Native HA グループ間のセキュアな通信で使用される TLS キーの設定。

以下の中に含まれます:

- 256 ページの『.spec.queueManager.availability.nativeHAGroups.local.tls』

フィールド	説明
secretName 文字列	Kubernetes シークレットの名前。 秘密鍵は、単一の秘密鍵、公開証明書、信頼の連鎖を含む ' kubernetes.io/tls ' 型でなければならない。

.spec.queueManager.availability.nativeHAGroups.remotes

以下の中に含まれます:

- 255 ページの『.spec.queueManager.availability.nativeHAGroups』

フィールド	説明
addresses 配列	リモート NativeHA グループと通信するためのホスト名とポートアドレスの一覧。 ローカルグループロールが Live の場合のみ有効です。
enabled ブール値	リモートとの通信を開始するかどうか。 ローカルの NativeHA グループで、ロールが「Live」のもののみと互換性があります。 設定されていない場合、 MQ コンテナは内部デフォルトを使用します。
name 文字列	リモート NativeHA グループの名前。 これは、リモートクラスタで構成された NativeHA グループ名と一致する必要があります。
retries 再試行	リモートに対する再試行動作の設定。
statusInterval 整数	リカバリログのステータスを送信する前に、リカバリグループが待機する最長時間（ミリ秒単位）。
trust 信頼配列	リモート NativeHA グループとの通信のためにキューマネージャーのキーリポジトリに追加する証明書。

.spec.queueManager.availability.nativeHAGroups.remotes.retries

リモートに対する再試行動作の設定。

以下の中に含まれます:

- 256 ページの『.spec.queueManager.availability.nativeHAGroups.remotes』

フィールド	説明
long 長い	長い再試行フェーズ用の構成を再試行します。
reporting 文字列	再試行の試行をすべて報告するか、再試行フェーズのみ報告するか。
short 短い	短い再試行フェーズ用の構成を再試行します。

.spec.queueManager.availability.nativeHAGroups.remotes.retries.long

長い再試行フェーズ用の構成を再試行します。

以下の中に含まれます:

- 257 ページの『.spec.queueManager.availability.nativeHAGroups.remotes.retries』

フィールド	説明
interval 整数	リトライの長いフェーズ中にリモートへの接続を試みる間の待機時間（ミリ秒単位）。

.spec.queueManager.availability.nativeHAGroups.remotes.retries.short

短い再試行フェーズ用の構成を再試行します。

以下の中に含まれます:

- 257 ページの『.spec.queueManager.availability.nativeHAGroups.remotes.retries』

フィールド	説明
count 整数	短い再試行フェーズ中に接続を試みる回数。
interval 整数	リトライの短いフェーズ中にリモートへの接続を試みる間の待機時間（ミリ秒単位）。

.spec.queueManager.availability.nativeHAGroups.remotes.trust

NativeHAGroupTrust 証明機関の公開鍵のソースを定義します。

以下の中に含まれます:

- 256 ページの『.spec.queueManager.availability.nativeHAGroups.remotes』

フィールド	説明
configMap コンフィグマップ	Kubenetes Config Map を使用して CA 証明書を供給します。
secret Secret	Kubernetes Secret を使用して CA 証明書を供給する。

.spec.queueManager.availability.nativeHAGroups.remotes.trust.configMap

Kubenetes Config Map を使用して CA 証明書を供給します。

以下の中に含まれます:

- 258 ページの『.spec.queueManager.availability.nativeHAGroups.remotes.trust』

フィールド	説明
items 配列	適用すべき Kubernetes 構成マップソース内のキー。 指定されていない場合は、すべてのキーを使用します。
name 文字列	Kubernetes configmap ソースの名前。

.spec.queueManager.availability.nativeHAGroups.remotes.trust.secret

Kubernetes Secret を使用して CA 証明書を供給する。

以下の中に含まれます:

- 258 ページの『.spec.queueManager.availability.nativeHAGroups.remotes.trust』

フィールド	説明
items 配列	Kubernetes 秘密鍵に適用すべきキー。 指定されていない場合は、すべてのキーを使用します。
secretName 文字列	Kubernetes シークレットの名前。

.spec.queueManager.availability.tls

NativeHA レプリカ間のセキュア通信を構成するためのオプションの TLS 設定。 MQ Operator 1.5.0 以上が必要です。

以下の中に含まれます:

- 255 ページの『.spec.queueManager.availability』

フィールド	説明
cipherSpec 文字列	NativeHA TLS 用の CipherSpec の名前。
secretName 文字列	Kubernetes シークレットの名前。

.spec.queueManager 環境

EnvVar コンテナに存在する環境変数を表します。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
name 文字列	環境変数の名前。 C_IDENTIFIER でなければなりません。
value 文字列	変数参照 $(VAR_NAME) は、コンテナ内で事前に定義された環境変数と、すべてのサービス環境変数を使用して展開されます。 変数が解決できない場合、入力文字列の参照は変更されません。 2 つの$$は 1 つの$に縮小され、これにより、$(VAR_NAME)という構文をエスケープすることが可能になります。すなわち、 $$(VAR_NAME) 文字列リテラル を生成します。 $(VAR_NAME) エスケープされた参照は、変数が存在するかどうかに関わらず、決して展開されることはありません。 にデフォルト設定されています。
valueFrom 価値	環境変数の値のソース。 値が空でない場合は使用できません。

.spec.queueManager.env.valueFrom

環境変数の値のソース。 値が空でない場合は使用できません。

以下の中に含まれます:

- 258 ページの『.spec.queueManager 環境』

フィールド	説明
configMapKeyRef コンフィグマップキーレフ	ConfigMap のキーを選択します。
fieldRef 分野参照	ポッドのフィールドを選択： metadata.name、 metadata.namespace、 metadata.labels['<KEY>']、 metadata.annotations['<KEY>']、 spec.nodeName,、 spec.serviceAccountName,、 status.hostIP,、 status.podIP,、 status.podIPs をサポート。
resourceFieldRef リソースフィールド参照	コンテナのリソースを選択：現在サポートされているのは、リソースの制限とリクエスト（ limits.cpu、 limits.memory、 limits.ephemeral-storage、 requests.cpu、 requests.memory、 requests.ephemeral-storage ）のみです。
secretKeyRef 秘密鍵参照	ポッドの名前空間における秘密のキーを選択します。

.spec.queueManager.env.valueFrom.configMapKeyRef

ConfigMap のキーを選択します。

以下の中に含まれます:

- 259 ページの『.spec.queueManager.env.valueFrom』

フィールド	説明
key 文字列	選択のポイント。
name 文字列	参照の名前。 このフィールドは事実上必須であるが、後方互換性のため、空でも許される。 ここに空の値を持つこの型のインスタンスは、ほぼ間違いない。 詳細はこちら： https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names. デフォルト：""。
optional ブール値	ConfigMap またはそのキーを定義する必要があるかどうかを指定します。

.spec.queueManager.env.valueFrom.fieldRef

ポッドのフィールドを選択： metadata.name、 metadata.namespace、 metadata.labels['<KEY>']、 metadata.annotations['<KEY>']、 spec.nodeName,、 spec.serviceAccountName,、 status.hostIP,、 status.podIP,、 status.podIPs をサポート。

以下の中に含まれます:

- 259 ページの『.spec.queueManager.env.valueFrom』

フィールド	説明
apiVersion 文字列	FieldPath が記述されているスキーマのバージョンは、 v1 をデフォルトとします。
fieldPath 文字列	指定の API バージョンで選択するフィールドのパス。

.spec.queueManager.env.valueFrom.resourceFieldRef

コンテナのリソースを選択：現在サポートされているのは、リソースの制限とリクエスト（ limits.cpu、 limits.memory、 limits.ephemeral-storage、 requests.cpu、 requests.memory、 requests.ephemeral-storage ）のみです。

以下の中に含まれます:

- 259 ページの『.spec.queueManager.env.valueFrom』

フィールド	説明
containerName 文字列	コンテナ名：ボリュームには必須、環境変数には任意。
divisor	公開されたリソースの出力フォーマットを指定します。デフォルトは 1 です。
resource 文字列	必須：選択するリソース。

.spec.queueManager.env.valueFrom.secretKeyRef

ポッドの名前空間における秘密のキーを選択します。

以下の中に含まれます:

- 259 ページの『.spec.queueManager.env.valueFrom』

フィールド	説明
key 文字列	選ぶための秘訣の鍵。 有効な秘密鍵でなければなりません。
name 文字列	参照の名前。 このフィールドは事実上必須であるが、後方互換性のため、空でも許される。 ここに空の値を持つこの型のインスタンスは、ほぼ間違いない。 詳細はこちら： https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names. デフォルト：""。
optional ブール値	秘密またはそのキーを定義する必要があるかどうかを指定します。

.spec.queueManager ファイル

ファイルのソース。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
configMap コンフィグマップ	ConfigMap Kubernetes というファイルを含むフォルダを表します。 ConfigMap
secret Secret	Secret は、ファイルを含む Kubernetes Secret を表します。

.spec.queueManager.files.configMap

ConfigMap Kubernetes というファイルを含むフォルダを表します。 ConfigMap

以下の中に含まれます:

- 260 ページの『.spec.queueManager ファイル』

フィールド	説明
defaultMountPath 文字列	Kubernetes ソース内のキーに使用するデフォルトのマウントパス。 これは、キーごとに個別のマウントパスを指定することで上書きできます。 指定がない場合は、 /etc/mqm/customfiles が使用されます。
items アイテム配列	Kubernetes ソース内の適用すべきキー。
name 文字列	Kubernetes ソースの名前。

.spec.queueManager.files.configMap アイテム

以下の中に含まれます:

- 261 ページの『.spec.queueManager.files.configMap』

フィールド	説明
item 文字列	キーの名前です。
mountPath 文字列	キーに使用するマウントパスは、デフォルトのマウントパスを上書きします。

.spec.queueManager.files.secret

Secret は、ファイルを含む Kubernetes Secret を表します。

以下の中に含まれます:

- 260 ページの『.spec.queueManager ファイル』

フィールド	説明
defaultMountPath 文字列	Kubernetes ソース内のキーに使用するデフォルトのマウントパス。 これは、キーごとに個別のマウントパスを指定することで上書きできます。 指定がない場合は、 /etc/mqm/customfiles が使用されます。
items アイテム配列	Kubernetes ソース内の適用すべきキー。
name 文字列	Kubernetes ソースの名前。

.spec.queueManager.files.secret.items

以下の中に含まれます:

- 261 ページの『.spec.queueManager.files.secret』

フィールド	説明
item 文字列	キーの名前です。
mountPath 文字列	キーに使用するマウントパスは、デフォルトのマウントパスを上書きします。

.spec.queueManager.ini

INI 構成ファイルのソース。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
configMap ConfigMapINISource	configMap は、INI 情報が入っている Kubernetes ConfigMap を表します。
secret SecretINISource	secret は、INI 情報が入っている Kubernetes シークレットを表します。

.spec.queueManager.ini.configMap

configMap は、INI 情報が入っている Kubernetes ConfigMap を表します。

以下の中に含まれます:

- 261 ページの『.spec.queueManager.ini』

フィールド	説明
items 配列	Kubernetes ソース内の適用すべきキー。
name 文字列	Kubernetes ソースの名前。

.spec.queueManager.ini.secret

secret は、INI 情報が入っている Kubernetes シークレットを表します。

以下の中に含まれます:

- 261 ページの『.spec.queueManager.ini』

フィールド	説明
items 配列	Kubernetes ソース内の適用すべきキー。
name 文字列	Kubernetes ソースの名前。

.spec.queueManager.livenessProbe

Liveness プローブを制御する設定。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
failureThreshold 整数	プローブが成功した後に、この回数以上連続して失敗すると、失敗したプローブと見なされます。 デフォルトは 1 です。
initialDelaySeconds 整数	コンテナーが始動してからプローブが開始されるまでの秒数。 SingleInstance の場合、デフォルトは 90 秒です。 MultiInstance デプロイメントと NativeHA デプロイメントの場合、デフォルトは 0 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。
periodSeconds 整数	プローブを実行する間隔 (秒単位)。 デフォルトは 10 秒です。
successThreshold 整数	プローブが成功した後に、この回数以上連続して成功すると、成功したプローブと見なされます。 デフォルトは 1 です。

フィールド	説明
timeoutSeconds 整数	プローブがタイムアウトになるまでの秒数。 デフォルトは 5 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。

.spec.queueManager.metrics

Prometheus スタイルのメトリックの設定。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
enabled ブール値	Prometheus 互換のメトリックのエンドポイントを有効にするかどうか。 デフォルトは true です。
serviceMonitor サービス監視員	Prometheus -互換メトリクスのサービス・モニターの設定。 MQ Operator 3.6.0 以上が必要。
tls Tls	Prometheus 互換のメトリクスで安全な通信を設定するための TLS 設定。 MQ オペレータ 3.5.0 以上が必要です。

.spec.queueManager.metrics.serviceMonitor

Prometheus -互換メトリクスのサービス・モニターの設定。 MQ Operator 3.6.0 以上が必要。

以下の中に含まれます:

- 263 ページの『.spec.queueManager.metrics』

フィールド	説明
enabled ブール値	サービスモニターを有効にするかどうか。 デフォルトは true です。

.spec.queueManager.metrics.tls

Prometheus 互換のメトリクスで安全な通信を設定するための TLS 設定。 MQ オペレータ 3.5.0 以上が必要です。

以下の中に含まれます:

- 263 ページの『.spec.queueManager.metrics』

フィールド	説明
provider 文字列	Prometheus 互換のメトリクスで安全な通信を設定するために使用するプロバイダー。 デフォルトは none です。 none を、安全でない HTTP のメトリクスに使用してください。 openshift を使用して、 OpenShift-provided の証明書を ServiceMonitor で使用します。

.spec.queueManager.mqsc

MQSC 構成ファイルのソース。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
configMap ConfigMapMQSCSource	configMap は、MQSC 情報が入っている Kubernetes ConfigMap を表します。
secret SecretMQSCSource	secret は、MQSC 情報が入っている Kubernetes シークレットを表します。

.spec.queueManager.mqsc.configMap

configMap は、MQSC 情報が入っている Kubernetes ConfigMap を表します。

以下の中に含まれます:

- 263 ページの『.spec.queueManager.mqsc』

フィールド	説明
items 配列	Kubernetes ソース内の適用すべきキー。
name 文字列	Kubernetes ソースの名前。

.spec.queueManager.mqsc.secret

secret は、MQSC 情報が入っている Kubernetes シークレットを表します。

以下の中に含まれます:

- 263 ページの『.spec.queueManager.mqsc』

フィールド	説明
items 配列	Kubernetes ソース内の適用すべきキー。
name 文字列	Kubernetes ソースの名前。

.spec.queueManager.readinessProbe

Readiness プローブを制御する設定。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
failureThreshold 整数	プローブが成功した後に、この回数以上連続して失敗すると、失敗したプローブと見なされます。 デフォルトは 1 です。
initialDelaySeconds 整数	コンテナーが始動してからプローブが開始されるまでの秒数。 SingleInstance の場合、デフォルトは 10 秒です。 MultiInstance デプロイメントと NativeHA デプロイメントの場合、デフォルトは 0 です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。
periodSeconds 整数	プローブを実行する間隔 (秒単位)。 デフォルトは 5 秒です。
successThreshold 整数	プローブが成功した後に、この回数以上連続して成功すると、成功したプローブと見なされます。 デフォルトは 1 です。
timeoutSeconds 整数	プローブがタイムアウトになるまでの秒数。 デフォルトは 3 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。

.spec.queueManager.recoveryLogs

MQ リカバリー・ログの設定。 MQ Operator 2.4.0 以上が必要です。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
logFilePages 整数	回復ログ・データは一連のファイルに保持されます。 ログ・ファイル・サイズは、4 KB ページ単位で指定します。

.spec.queueManager.resources

リソース要件を制御する設定。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
limits Limits	CPU とメモリーの設定。
requests Requests	CPU とメモリーの設定。

.spec.queueManager.resources.limits

CPU とメモリーの設定。

以下の中に含まれます:

- 265 ページの『.spec.queueManager.resources』

フィールド	説明
cpu	
memory	

.spec.queueManager.resources.requests

CPU とメモリーの設定。

以下の中に含まれます:

- 265 ページの『.spec.queueManager.resources』

フィールド	説明
cpu	
memory	

.spec.queueManager.route

キュー・マネージャー・ルートの設定。 MQ Operator 1.4.0 以上が必要です。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
enabled ブール値	ルートを有効にするかどうか。 デフォルトは true です。

.spec.queueManager.startupProbe

始動プローブを制御する設定。 MultiInstance デプロイメントと NativeHA デプロイメントにのみ適用されます。 MQ Operator 1.5.0 以上が必要です。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
failureThreshold 整数	この回数以上連続して失敗すると、失敗したプローブと見なされます。 デフォルトは 24 です。
initialDelaySeconds 整数	コンテナーが始動してからプローブが開始されるまでの秒数。 デフォルトは 0 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。
periodSeconds 整数	プローブを実行する間隔 (秒単位)。 デフォルトは 5 秒です。
successThreshold 整数	この回数以上連続して成功すると、成功したプローブと見なされます。 デフォルトは 1 です。
timeoutSeconds 整数	プローブがタイムアウトになるまでの秒数。 デフォルトは 5 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。

.spec.queueManager.storage

キュー・マネージャーが永続ボリュームおよびストレージ・クラスを使用することを制御するストレージ設定です。

以下の中に含まれます:

- 254 ページの『.spec.queueManager』

フィールド	説明
allowVolumeExpansion ブール値	ボリュームの拡張を許可するかどうか。
defaultClass 文字列	このキュー・マネージャーのすべての永続ボリュームにデフォルトで適用するストレージ・クラス。 個々の永続ボリュームで独自のストレージ・クラスを定義すると、このデフォルトのストレージ・クラス設定がオーバーライドされます。 type of availability が SingleInstance または NativeHA の場合は、ストレージ・クラスのタイプを ReadWriteOnce か ReadWriteMany にできます。 type of availability が MultiInstance の場合は、ストレージ・クラスのタイプを ReadWriteMany にしなければなりません。
defaultDeleteClaim ブール値	キュー・マネージャーの削除時にすべてのボリュームを削除するかどうか。 個々の永続ボリュームで独自の deleteClaim 値を定義すると、この defaultDeleteClaim 設定がオーバーライドされます。 デフォルトは false です。
persistedData QueueManagerOptionalVolume	構成、キュー、メッセージなどの MQ の永続データに使用する永続ボリュームの詳細。 マルチインスタンスのキュー・マネージャーを使用する場合は必須です。
queueManager QueueManagerVolume	あらゆるデータに使用するデフォルトの永続ボリューム (通常は、/var/mqm 下に置かれます)。 他のボリュームを指定しない場合は、すべての永続データとリカバリー・ログがここに格納されます。
recoveryLogs QueueManagerOptionalVolume	MQ リカバリー・ログ用の永続ボリュームの詳細。 マルチインスタンスのキュー・マネージャーを使用する場合は必須です。

フィールド	説明
scratch スクラッチ	キュー・マネージャーの Scratch 一時ボリュームの設定。 このボリュームは、「/run」フォルダーとしてコンテナーにマウントされます。 ルート・ファイル・システムが読み取り専用に設定されている場合にのみ適用されます。 MQ Operator 3.0.0 以上が必要です。
tmp 一時	キュー・マネージャーの一時一時ボリュームの設定。 このボリュームは、「/tmp」フォルダーとしてコンテナーにマウントされます。 runmqras コマンドによって作成された zip ファイルなどの診断データ・ファイルが、このボリュームに作成されます。 ルート・ファイル・システムが読み取り専用に設定されている場合にのみ適用されます。 MQ Operator 3.0.0 以上が必要です。

.spec.queueManager.storage.persistedData

構成、キュー、メッセージなどの MQ の永続データに使用する永続ボリュームの詳細。 マルチインスタンスのキュー・マネージャーを使用する場合は必須です。

以下の中に含まれます:

- 266 ページの『.spec.queueManager.storage』

フィールド	説明
class 文字列	このボリュームに使用するストレージ・クラス。 type が persistent-claim である場合にのみ有効です。 type of availability が SingleInstance または NativeHA の場合は、ストレージ・クラスのタイプを ReadWriteOnce か ReadWriteMany にできます。 type of availability が MultiInstance の場合は、ストレージ・クラスのタイプを ReadWriteMany にしなければなりません。
deleteClaim ブール値	キュー・マネージャーの削除時にこのボリュームを削除するかどうか。
enabled ブール値	このボリュームを別個のボリュームとして使用可能にするかどうか、あるいは、デフォルトの queueManager ボリュームに置くかどうか。 デフォルトは false です。
size 文字列	Kubernetes に渡す PersistentVolume のサイズ (SI 単位を含む)。 type が persistent-claim である場合にのみ有効です。 例: 2Gi。 デフォルトは 2Gi です。
sizeLimit 文字列	ephemeral ボリュームを使用する場合のサイズの制限。 ファイルは引き続き一時ディレクトリーに書き込まれるので、このオプションを使用してサイズを制限できます。 type が ephemeral で、ルート・ファイル・システムが読み取り専用に設定されている場合にのみ有効です。 MQ Operator 3.0.0 以上が必要です。
type 文字列	使用するボリュームのタイプ。 非永続ストレージを使用する場合は ephemeral を選択し、永続ボリュームを使用する場合は persistent-claim を選択します。 デフォルトは persistent-claim です。

.spec.queueManager.storage.queueManager

あらゆるデータに使用するデフォルトの永続ボリューム (通常は、/var/mqm 下に置かれます)。 他のボリュームを指定しない場合は、すべての永続データとリカバリー・ログがここに格納されます。

以下の中に含まれます:

- 266 ページの『.spec.queueManager.storage』

フィールド	説明
class 文字列	このボリュームに使用するストレージ・クラス。 type が persistent-claim である場合にのみ有効です。 type of availability が SingleInstance または NativeHA の場合は、ストレージ・クラスのタイプを ReadWriteOnce か ReadWriteMany にできます。 type of availability が MultiInstance の場合は、ストレージ・クラスのタイプを ReadWriteMany にしなければなりません。
deleteClaim ブール値	キュー・マネージャーの削除時にこのボリュームを削除するかどうか。
size 文字列	Kubernetes に渡す PersistentVolume のサイズ (SI 単位を含む)。 type が persistent-claim である場合にのみ有効です。 例: 2Gi。 デフォルトは 2Gi です。
sizeLimit 文字列	ephemeral ボリュームを使用する場合のサイズの制限。 ファイルは引き続き一時ディレクトリーに書き込まれるので、このオプションを使用してサイズを制限できます。 type が ephemeral で、ルート・ファイル・システムが読み取り専用に設定されている場合にのみ有効です。 MQ Operator 3.0.0 以上が必要です。
type 文字列	使用するボリュームのタイプ。 非永続ストレージを使用する場合は ephemeral を選択し、永続ボリュームを使用する場合は persistent-claim を選択します。 デフォルトは persistent-claim です。

.spec.queueManager.storage.recoveryLogs

MQ リカバリー・ログ用の永続ボリュームの詳細。 マルチインスタンスのキュー・マネージャーを使用する場合は必須です。

以下の中に含まれます:

- 266 ページの『.spec.queueManager.storage』

フィールド	説明
class 文字列	このボリュームに使用するストレージ・クラス。 type が persistent-claim である場合にのみ有効です。 type of availability が SingleInstance または NativeHA の場合は、ストレージ・クラスのタイプを ReadWriteOnce か ReadWriteMany にできます。 type of availability が MultiInstance の場合は、ストレージ・クラスのタイプを ReadWriteMany にしなければなりません。
deleteClaim ブール値	キュー・マネージャーの削除時にこのボリュームを削除するかどうか。
enabled ブール値	このボリュームを別個のボリュームとして使用可能にするかどうか、あるいは、デフォルトの queueManager ボリュームに置くかどうか。 デフォルトは false です。
size 文字列	Kubernetes に渡す PersistentVolume のサイズ (SI 単位を含む)。 type が persistent-claim である場合にのみ有効です。 例: 2Gi。 デフォルトは 2Gi です。
sizeLimit 文字列	ephemeral ボリュームを使用する場合のサイズの制限。 ファイルは引き続き一時ディレクトリーに書き込まれるので、このオプションを使用してサイズを制限できます。 type が ephemeral で、ルート・ファイル・システムが読み取り専用に設定されている場合にのみ有効です。 MQ Operator 3.0.0 以上が必要です。

フィールド	説明
type 文字列	使用するボリュームのタイプ。 非永続ストレージを使用する場合は ephemeral を選択し、永続ボリュームを使用する場合は persistent-claim を選択します。 デフォルトは persistent-claim です。

.spec.queueManager.storage.scratch

キュー・マネージャーの Scratch 一時ボリュームの設定。 このボリュームは、「/run」フォルダーとしてコンテナーにマウントされます。 ルート・ファイル・システムが読み取り専用に設定されている場合にのみ適用されます。 MQ Operator 3.0.0 以上が必要です。

以下の中に含まれます:

- 266 ページの『.spec.queueManager.storage』

フィールド	説明
sizeLimit 文字列	一時ボリュームのサイズ制限 (SI 単位を含む)。 例えば、 2Gi などです。 ルート・ファイル・システムが読み取り専用に設定されている場合にのみ有効です。 MQ Operator 3.0.0 以上が必要です。

.spec.queueManager.storage.tmp

キュー・マネージャーの一時一時ボリュームの設定。 このボリュームは、「/tmp」フォルダーとしてコンテナーにマウントされます。 runmqras コマンドによって作成された zip ファイルなどの診断データ・ファイルが、このボリュームに作成されます。 ルート・ファイル・システムが読み取り専用に設定されている場合にのみ適用されます。 MQ Operator 3.0.0 以上が必要です。

以下の中に含まれます:

- 266 ページの『.spec.queueManager.storage』

フィールド	説明
sizeLimit 文字列	一時ボリュームのサイズ制限 (SI 単位を含む)。 例えば、 2Gi などです。 ルート・ファイル・システムが読み取り専用に設定されている場合にのみ有効です。 MQ Operator 3.0.0 以上が必要です。

.spec.securityContext

キュー・マネージャー・ポッドの securityContext に追加するセキュリティー設定です。

以下の中に含まれます:

- 250 ページの『.spec』

フィールド	説明
fsGroup 整数	ポッド内のすべてのコンテナーに適用される特殊な補助グループ。 いくつかのボリューム・タイプでは、 Kubelet がそのボリュームの所有権をポッドによって所有されるように変更することができます : 1。 所有 GID は、 FSGroup 2 になります。 setgid ビットが設定されます (ボリューム内に作成された新規ファイルは、 FSGroup が所有します) 3 です。 許可ビットは 0660 と OR される。 未設定の場合、Kubelet はボリュームの所有権とパーミッションを変更しません。

フィールド	説明
initVolumeAsRoot ブール値	これは、PersistentVolume を初期化するコンテナーで使用されるセキュリティー・コンテキストに影響を与えます。 新しくプロビジョンしたボリュームには root ユーザーを使用してアクセスしなければならないストレージ・プロバイダーを使用する場合は、true に設定します。 これを true に設定すると、使用できるセキュリティー・コンテキスト制約 (SCC) オブジェクトが影響を受けます。 root ユーザーを許可する SCC を使用する権限を持っていないと、キュー・マネージャーの始動に失敗する可能性があります。 デフォルトは false です。 詳しくは、https://docs.openshift.com/container-platform/latest/authentication/managing-security-context-constraints.html を参照してください。
readOnlyRootFilesystem ブール値	キュー・マネージャーの読み取り専用ルート・ファイル・システム設定を有効にするかどうか。 デフォルトは false です。 MQ Operator 3.0.0 以上が必要です。
supplementalGroups 配列	コンテナーの 1 次 GID に加えて、最初のプロセスに適用されるグループのリストは、各コンテナーで実行されます。 指定しない場合は、コンテナーにグループは追加されません。

.spec.service

すべての QueueManager サービスに適用される設定。 このセクションまたはその中のフィールドを変更または削除すると、すべての QueueManager サービスが再生成される場合があります。 サービス再生成により通信が中断され、 NativeHA を含むすべての QueueManager タイプでダウンタイムが発生する。 再生成されたサービスには、 Kubernetes によって新しい ClusterIP アドレスが割り当てられる。 MQ Operator 3.6.0 以上が必要。

以下の中に含まれます:

- 250 ページの『.spec』

フィールド	説明
ipFamilies 配列	すべての QueueManager サービスで使用する IP ファミリを指定します。 有効なオプションは IPv4 と IPv6。 未指定の場合、 Kubernetes は、 ipFamilyPolicy の設定に基づいてデフォルト値を設定する。 手動で設定した場合 IP ファミリがクラスタ上で利用できない場合、または値が ipFamilyPolicy の設定と互換性がない場合、サービスの作成は失敗します。 配列の最初の値(プライマリ IP ファミリ)を変更するか、このフィールドを削除すると、すべての QueueManager サービスが再生成される。 サービス再生成により通信が中断され、 NativeHA を含むすべての QueueManager タイプでダウンタイムが発生する。 再生成されたサービスには、 Kubernetes によって新しい ClusterIP アドレスが割り当てられる。
ipFamilyPolicy 文字列	すべての QueueManager サービスに使用する IP ファミリーポリシーを指定します。 有効なオプションは SingleStack, PreferDualStack と RequireDualStack。 SingleStack は単一の IP ファミリーのものである。 PreferDualStack は、デュアルスタック構成のクラスタでは 2 つの IP ファミリー、シングルスタック構成のクラスタでは 1 つの IP ファミリーの場合です。 RequireDualStack はデュアルスタック構成のクラスタで 2 つの IP ファミリーの場合です。 このフィールドを削除すると、すべての QueueManager サービスが再生成される。 サービス再生成により通信が中断され、 NativeHA を含むすべての QueueManager タイプでダウンタイムが発生する。 再生成されたサービスには、 Kubernetes によって新しい ClusterIP アドレスが割り当てられる。

.spec.telemetry

Open Telemetry 構成の設定。 MQ Operator 2.2.0 以上が必要です。

以下の中に含まれます:

- 250 ページの『.spec』

フィールド	説明
tracing トレース	Open Telemetry トレースの設定。

.spec.telemetry.tracing

Open Telemetry トレースの設定。

以下の中に含まれます:

- 270 ページの『.spec.telemetry』

フィールド	説明
instana インスタナ	Instana トレースの設定。

.spec.telemetry.tracing.instana

Instana トレースの設定。

以下の中に含まれます:

- 271 ページの『.spec.telemetry.tracing』

フィールド	説明
agentHost 文字列	トレース・データの送信先の Instana エージェントのホスト名。 これにプロトコルを含めることはできません。
enabled ブール値	Instana トレースを有効にするかどうか。 デフォルトは false です。
protocol 文字列	Instana エージェントとの通信で使用されるプロトコル。 http および https がサポートされています。

.spec.template

Kubernetes リソースの拡張テンプレート。 このテンプレートは、基礎の Kubernetes リソース (StatefulSet、Pod、Service など) を IBM MQ で生成する方法をユーザーがオーバーライドすることを可能にします。 これは上級者専用です。誤って使用すると MQ の正常な動作が阻害される可能性があります。 QueueManager リソースの他の場所で指定された値は、このテンプレートの設定によってオーバーライドされます。

以下の中に含まれます:

- 250 ページの『.spec』

フィールド	説明
pod	ポッドに使用するテンプレートのオーバーライド。 https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.17/#podspec-v1-core を参照してください。

.spec.tracing

Cloud Pak for Integration Operations Dashboard とのトレースの統合のための設定。

以下の中に含まれます:

- 250 ページの『.spec』

フィールド	説明
agent TracingAgent	Cloud Pak for Integration でのみ、オプションの Tracing Agent の設定を構成できます。
collector TracingCollector	Cloud Pak for Integration でのみ、オプションの Tracing Collector の設定を構成できます。
enabled ブール値	Cloud Pak for Integration Operations Dashboard とのトレースの統合を有効にするかどうか。 デフォルトは false です。
namespace 文字列	Cloud Pak for Integration Operations Dashboard がインストールされている名前空間。

.spec.tracing.agent

Cloud Pak for Integration でのみ、オプションの Tracing Agent の設定を構成できます。

以下の中に含まれます:

- 271 ページの『.spec.tracing』

フィールド	説明
image 文字列	使用するコンテナー・イメージ。
imagePullPolicy 文字列	指定されたイメージのプルを Kubelet が試行するタイミングを制御する設定。 デフォルトは IfNotPresent です。
livenessProbe TracingProbe	Liveness プローブを制御する設定。
readinessProbe TracingProbe	Readiness プローブを制御する設定。

.spec.tracing.agent.livenessProbe

Liveness プローブを制御する設定。

以下の中に含まれます:

- 272 ページの『.spec.tracing.agent』

フィールド	説明
failureThreshold 整数	プローブが成功した後に、この回数以上連続して失敗すると、失敗したプローブと見なされます。 デフォルトは 1 です。
initialDelaySeconds 整数	コンテナーが始動してから Liveness プローブが開始されるまでの秒数。 デフォルトは 10 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。
periodSeconds 整数	プローブを実行する間隔 (秒単位)。 デフォルトは 10 秒です。
successThreshold 整数	プローブが成功した後に、この回数以上連続して成功すると、成功したプローブと見なされます。 デフォルトは 1 です。
timeoutSeconds 整数	プローブがタイムアウトになるまでの秒数。 デフォルトは 2 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。

.spec.tracing.agent.readinessProbe

Readiness プローブを制御する設定。

以下の中に含まれます:

- 272 ページの『.spec.tracing.agent』

フィールド	説明
failureThreshold 整数	プローブが成功した後に、この回数以上連続して失敗すると、失敗したプローブと見なされます。 デフォルトは 1 です。
initialDelaySeconds 整数	コンテナーが始動してから Liveness プローブが開始されるまでの秒数。 デフォルトは 10 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。
periodSeconds 整数	プローブを実行する間隔 (秒単位)。 デフォルトは 10 秒です。
successThreshold 整数	プローブが成功した後に、この回数以上連続して成功すると、成功したプローブと見なされます。 デフォルトは 1 です。
timeoutSeconds 整数	プローブがタイムアウトになるまでの秒数。 デフォルトは 2 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。

.spec.tracing.collector

Cloud Pak for Integration でのみ、オプションの Tracing Collector の設定を構成できます。

以下の中に含まれます:

- 271 ページの『.spec.tracing』

フィールド	説明
image 文字列	使用するコンテナー・イメージ。
imagePullPolicy 文字列	指定されたイメージのプルを Kubelet が試行するタイミングを制御する設定。 デフォルトは IfNotPresent です。
livenessProbe TracingProbe	Liveness プローブを制御する設定。
readinessProbe TracingProbe	Readiness プローブを制御する設定。

.spec.tracing.collector.livenessProbe

Liveness プローブを制御する設定。

以下の中に含まれます:

- 273 ページの『.spec.tracing.collector』

フィールド	説明
failureThreshold 整数	プローブが成功した後に、この回数以上連続して失敗すると、失敗したプローブと見なされます。 デフォルトは 1 です。
initialDelaySeconds 整数	コンテナーが始動してから Liveness プローブが開始されるまでの秒数。 デフォルトは 10 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。
periodSeconds 整数	プローブを実行する間隔 (秒単位)。 デフォルトは 10 秒です。
successThreshold 整数	プローブが成功した後に、この回数以上連続して成功すると、成功したプローブと見なされます。 デフォルトは 1 です。

フィールド	説明
timeoutSeconds 整数	プローブがタイムアウトになるまでの秒数。 デフォルトは 2 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。

.spec.tracing.collector.readinessProbe

Readiness プローブを制御する設定。

以下の中に含まれます:

- 273 ページの『.spec.tracing.collector』

フィールド	説明
failureThreshold 整数	プローブが成功した後に、この回数以上連続して失敗すると、失敗したプローブと見なされます。 デフォルトは 1 です。
initialDelaySeconds 整数	コンテナーが始動してから Liveness プローブが開始されるまでの秒数。 デフォルトは 10 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。
periodSeconds 整数	プローブを実行する間隔 (秒単位)。 デフォルトは 10 秒です。
successThreshold 整数	プローブが成功した後に、この回数以上連続して成功すると、成功したプローブと見なされます。 デフォルトは 1 です。
timeoutSeconds 整数	プローブがタイムアウトになるまでの秒数。 デフォルトは 2 秒です。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。

.spec.web

MQ Web サーバーの設定。

以下の中に含まれます:

- 250 ページの『.spec』

フィールド	説明
console コンソール	MQ Web コンソールの設定。 MQ Operator 3.0.0 以上が必要です。
enabled ブール値	Web サーバーを有効にするかどうか。 デフォルトは false です。
manualConfig マニュアル構成	Web サーバー XML 構成を指定するための設定。 MQ Operator 3.0.0 以上が必要です。
route Route	ウェブサーバールートの設定。 MQ Operator 3.6.0 以上が必要。

.spec.web.console

MQ Web コンソールの設定。 MQ Operator 3.0.0 以上が必要です。

以下の中に含まれます:

- 274 ページの『.spec.web』

フィールド	説明
authentication 認証	MQ Web コンソールの認証設定。 MQ Operator 3.0.0 以上が必要です。
authorization 許可	MQ Web コンソールの許可設定。 MQ Operator 3.0.0 以上が必要です。

.spec.web.console.authentication

MQ Web コンソールの認証設定。 MQ Operator 3.0.0 以上が必要です。

以下の中に含まれます:

- 274 ページの『.spec.web.console』

フィールド	説明
`provider` 文字列	MQ Web コンソール用に使用する認証プロバイダー。 `integration-keycloak` を使用して、 Cloud Pak for Integration Platform UI (Keycloak) でシングル・サインオンを使用します。 デフォルトは、 Cloud Pak for Integration ライセンスを使用する場合は `integration-keycloak` 、 MQ ライセンスを使用する場合は `manual` です。 独自の構成を指定する場合は、 `manual` を使用します。

.spec.web.console.authorization

MQ Web コンソールの許可設定。 MQ Operator 3.0.0 以上が必要です。

以下の中に含まれます:

- 274 ページの『.spec.web.console』

フィールド	説明
`provider` 文字列	MQ Web コンソール用に使用する許可プロバイダー。 `integration-keycloak` を使用して、 Cloud Pak for Integration Keycloak によって提供される役割を使用します。 独自の構成を指定する場合は、 `manual` を使用します。 デフォルトは、 Cloud Pak for Integration ライセンスを使用する場合は `integration-keycloak` 、 MQ ライセンスを使用する場合は `manual` です。

.spec.web.manualConfig

Web サーバー XML 構成を指定するための設定。 MQ Operator 3.0.0 以上が必要です。

以下の中に含まれます:

- 274 ページの『.spec.web』

フィールド	説明
`configMap` コンフィグマップ	ConfigMap は、Web サーバー XML 構成を含む Kubernetes ConfigMap を表します。
`secret` Secret	シークレットは、Web サーバー XML 構成を含む Kubernetes シークレットを表します。 シークレットを使用すると、 Kubernetes レイヤー内のすべての資格情報が保護されますが、モニター・ツールやトラブルシューティング・ツールによって、基礎となるファイルが非セキュアに公開される可能性があります。 セキュリティを向上させるには、“securityUtility”を使用してクレデンシャルをエンコードする。

.spec.web.manualConfig.configMap

ConfigMap は、Web サーバー XML 構成を含む Kubernetes ConfigMap を表します。

以下の中に含まれます:

- 275 ページの『.spec.web.manualConfig』

フィールド	説明
name 文字列	Kubernetes ソースの名前。

.spec.web.manualConfig 秘密

シークレットは、Web サーバー XML 構成を含む Kubernetes シークレットを表します。 シークレットを使用すると、 Kubernetes レイヤー内のすべての資格情報が保護されますが、モニター・ツールやトラブルシューティング・ツールによって、基礎となるファイルが非セキュアに公開される可能性があります。 セキュリティを向上させるには、“securityUtility”を使用してクレデンシャルをエンコードする。

以下の中に含まれます:

- 275 ページの『.spec.web.manualConfig』

フィールド	説明
name 文字列	Kubernetes ソースの名前。

.spec.web.route

ウェブサーバールートの設定。 MQ Operator 3.6.0 以上が必要。

以下の中に含まれます:

- 274 ページの『.spec.web』

フィールド	説明
enabled ブール値	ルートを有効にするかどうか。 デフォルトは true です。

. 状況

QueueManager について観測された状態。

以下の中に含まれます:

- 250 ページの『QueueManager』

フィールド	説明
adminUiUrl 文字列	管理 UI の URL。
availability Availability	キュー・マネージャーの可用性状況。
conditions QueueManagerStatusCondition 配列	条件は、キュー・マネージャーの状態に関する最新の使用可能な監視を表します。
customImages ブール値	オペレーターが選択したキューマネージャーの画像が、カスタム画像で上書きされているかどうか。
endpoints QueueManagerStatusEndpoint 配列	このキュー・マネージャーが公開しているエンドポイントに関する情報 (API エンドポイントや UI エンドポイントなど)。
metadata メタデータ	メタデータは、統合-Keycloak 状況など、キュー・マネージャーの追加情報を表します。
name 文字列	キュー・マネージャーの名前。
nativeHACRR ネイティブ・ハッカー	構成されたネイティブ HA クロスリージョンレプリケーションに関する情報。

フィールド	説明
phase 文字列	キュー・マネージャーの状態のフェーズです。
versions QueueManagerStatusVersion	使用されている MQ のバージョン、および IBM Entitled Registry から取得できるその他のバージョン。

.status.availability

キュー・マネージャーの可用性状況。

以下の中に含まれます:

- 276 ページの『. 状況』

フィールド	説明
initialQuorumEstablished ブール値	NativeHA に初期クォーラムが確立されているかどうか。

.status.conditions

QueueManagerStatusCondition は、キュー・マネージャーの状況を示します。

以下の中に含まれます:

- 276 ページの『. 状況』

フィールド	説明
lastTransitionTime 文字列	状況のステータスが最後に遷移した時刻。
message 文字列	最後の遷移についての詳細を示す、人間が読める形式のメッセージ。
reason 文字列	最後にこのステータスに遷移した理由。
status 文字列	状況のステータス。 有効な値は "True"; "False"; "Unknown" です。
type 文字列	状況のタイプ。

.status.endpoints

QueueManagerStatusEndpoint は、QueueManager のエンドポイントを示します。

以下の中に含まれます:

- 276 ページの『. 状況』

フィールド	説明
name 文字列	エンドポイントの名前。
type 文字列	エンドポイントのタイプ。 UI エンドポイントの場合は「UI」、 API エンドポイントの場合は「API」、 API 資料の場合は「OpenAPI」です。
uri 文字列	エンドポイントの URI。

.status.metadata

メタデータは、統合-Keycloak 状況など、キュー・マネージャーの追加情報を表します。

以下の中に含まれます:

- 276 ページの『. 状況』

フィールド	説明
integrationKeycloak 統合 keycloak	QueueManagerStatusIntegrationKeycloak は、QueueManager の統合キー Keycloak のステータスを定義します。

.status.metadata.integrationKeycloak

QueueManagerStatusIntegrationKeycloak は、QueueManager の統合キー Keycloak のステータスを定義します。

以下の中に含まれます:

- 277 ページの『.status.metadata』

フィールド	説明
clientName 文字列	

.status.nativeHACRR

構成されたネイティブ HA クロスリージョンレプリケーションに関する情報。

以下の中に含まれます:

- 276 ページの『. 状況』

フィールド	説明
local ローカル	ローカルクラスタの状態。

.status.nativeHACRR ローカル

ローカルクラスタの状態。

以下の中に含まれます:

- 278 ページの『.status.nativeHACRR』

フィールド	説明
address 文字列	ローカルクラスタのアドレス。
groupName 文字列	ローカルクラスタのグループ名。

.status.versions

使用されている MQ のバージョン、および IBM Entitled Registry から取得できるその他のバージョン。

以下の中に含まれます:

- 276 ページの『. 状況』

フィールド	説明
available QueueManagerStatusVersionAvailable	IBM Entitled Registry から取得できるその他のバージョンの MQ。
reconciled 文字列	使用されている IBM MQ の具体的なバージョン。 カスタム・イメージが示されている場合は、実際に使用されている MQ のバージョンと一致していない可能性があります。

.status.versions.available

IBM Entitled Registry から取得できるその他のバージョンの MQ。

以下の中に含まれます:

- 278 ページの『.status.versions』

フィールド	説明
channels 配列	MQ のバージョンを自動的に更新するために使用できるチャネル。
versions Versions 配列	使用可能な MQ の具体的なバージョン。

.status.versions.available.versions

QueueManagerStatusVersion は MQ のバージョンを示します。

以下の中に含まれます:

- 279 ページの『.status.versions.available』

フィールド	説明
licenses Licenses 配列	このバージョンの QueueManager に適用可能なライセンス。
name 文字列	このバージョンの QueueManager のバージョン name 。 これは、spec.version フィールドで有効な値です。

.status.versions.available.versions.licenses

QueueManagerStatusLicense は、ライセンスを定義します。

以下の中に含まれます:

- 279 ページの『.status.versions.available.versions』

フィールド	説明
displayName 文字列	ライセンスの表示名。
link 文字列	ライセンス・コンテンツへのリンク。
matchesCurrentType ブール値	ライセンスが現在使用されているライセンスのタイプと一致するかどうか。
name 文字列	ライセンスの名前。

OpenShift CP4I QueueManager (mq.ibm.com/v1beta1) の状況状態

status.conditions フィールドは、QueueManager リソースの状態に合わせて更新されます。 状態は通常、異常な状況を説明するものです。 正常な作動可能状態のキュー・マネージャーは、**Error** や **Pending** の状態になりません。 通知のための **Warning** 状態になることはあります。

QueueManager リソースでは以下の状態が定義されています。

表 2. キュー・マネージャー状況状態

コンポーネント	状態タイプ	理由コード	メッセージ警告
QueueManager[2]	ブロック	OperatorDependency	このインスタンスをインストールするには、[IBM Cloud Pak for Integration] で Keycloak を構成する必要があります。 このインスタンスは、この QueueManager の Cp4iServicesBinding リソースで Keycloak が [KeycloakReady] として報告されるまで、[Pending] 状況のままになります。
			このインスタンスをインストールするには、オペレーター [IBM IAM] が必要です。 このインスタンスは、オペレーターが [IBM Cloud Pak 基本サービス] によってインストールされるまで「ブロック」状況のままになります。
	保留中	Creating	MQ キュー・マネージャーのデプロイ中です
	保留中	OidcPending	MQ キュー・マネージャーが OIDC クライアント登録を待っています。
	保留中	停止	「mq.ibm.com/stop」アノテーションが存在し、 QueueManager 定義で「true」に設定されているため、 MQ キュー・マネージャーが停止されました。 停止すると、 QueueManager StatefulSet レプリカ数がゼロに設定され、すべての MQ キュー・マネージャー・ポッドが削除されます。
	エラー	失敗	MQ キュー・マネージャーのデプロイが失敗しました
	警告	UnsupportedVersion	オペランドが、 OCP バージョン *<ocp_version>*でサポートされていないオペレーターによってインストールされました。 このオペランドはサポートされません。
	警告	CP4I-LTS のサポート	CP4I-LTS オペランド *< mq_version>* がインストールされましたが、拡張サポート期間に適格でないオペレーターによって管理されています。 このオペランドは拡張サポート期間に対応していません。
	警告	CP4I-LTS のサポート	CP4I-LTS オペランド *< mq_version>* がインストールされていますが、 OCP バージョン *< ocp_version>* が拡張サポート期間に適格ではありません。 このオペランドは拡張サポート期間に対応もしていません。

[2] 条件 Creating および Failed は、キュー・マネージャーのデプロイメントの全体的な進行状況をモニターします。 IBM Cloud Pak for Integration ライセンスを使用していて、Web コンソールが有効になっている場合、 OidcPending 条件は、OIDC クライアント登録が IAM で完了するのを待機している間のキュー・マネージャーの状況をログに記録します。

表 2. キュー・マネージャー状況状態 (続き)			
コンポーネント	**状態タイプ**	**理由コード**	**メッセージ警告**
ポッド [3]	保留中	PodPending	MQ キュー・マネージャーのポッドのデプロイ中です
	エラー	PodFailed	MQ キュー・マネージャーのポッドのデプロイ中です
記憶域 [4]	保留中	StoragePending	MQ キュー・マネージャーのストレージのプロビジョン中です
	警告	StorageEphemeral	実動 MQ キュー・マネージャーで一時ストレージを使用しています
	警告	StorageExpansion 保留中	次の PVC のボリューム拡張は保留中です [< list of pvcs>]
	警告	StorageMismatch	QueueManager リソースで定義されたストレージ・サイズが、1 つ以上のプロビジョンされた PVC [< list of pvcs>] の容量と一致しません。QueueManager リソースで AllowVolumeExpansion が false に設定されているため、 MQ オペレーターはこれらの差分の調整を試行しません。
	エラー	StorageFailed	MQ キュー・マネージャーのストレージのプロビジョンが失敗しました

Linux IBM MQ コンテナライセンス注釈

Kubernetes および Red Hat OpenShift Container Platform 環境では、IBM License Service は Pod アノテーションを使用して、基礎となるマシンではなくコンテナで定義された制限に基づいて使用量を追跡する。IBM MQ Operator 、これらの Pod アノテーションを設定してくれますが、他の方法でデプロイする場合は、自分で設定する必要があります。

キューマネージャの YAML の値に基づいて IBM MQ Operator がどのようにポッドにアノテーションをつけるかについては、 286 ページの『IBM MQ によって適用されるコンテナライセンス注釈。 IBM MQ Operator』 を参照してください。

IBM MQ コンテナをデプロイする場合、ライセンスには 2 つの一般的なアプローチがある：

- コンテナーを実行するマシン全体のライセンスを取得します。
- 関連する制限に基づくコンテナのライセンス (Kubernetes と Red Hat OpenShift Container Platform のみ)

どちらのオプションもお客様にご利用いただけます。詳細は Passport Advantage® の IBM Container Licenses ページをご覧ください。

[3] POD 条件は、キュー・マネージャーのデプロイメント中にポッドの状況をモニターします。 `PodFailed` 条件が表示された場合は、キュー・マネージャー全体の条件も `Failed` に設定されます。

[4] ストレージ条件は、永続ストレージのボリュームを作成する要求の進行状況 (`StoragePending` 条件) をモニターし、バインディング・エラーおよびその他の障害を報告します。 また、ストレージ条件は、ボリューム拡張の進行状況をモニターし、キュー・マネージャー定義で定義されたストレージ・サイズとデプロイされた PVC のサイズの間の不一致のアラートもモニターします。 ストレージ・プロビジョニング中にエラーが発生した場合、 `StorageFailed` 条件が条件リストに追加され、キュー・マネージャー全体の条件が `Failed` に設定されます。

コンテナーの制限に基づいて IBM MQ コンテナーのライセンスを取得する場合は、使用量を追跡管理するために IBM License Service をインストールする必要があります。 サポートされている環境やインストール手順の詳細については、GitHub の ibm-licensing-operator のページを参照してください。

IBM MQ コンテナーがデプロイされている Kubernetes クラスターに IBM License Service がインストールされ、ポッドのアノテーションを使用して使用量が追跡されます。 そのためクライアントで、IBM License Service がその後に使用する特定のアノテーションを付けてポッドをデプロイする必要があります。 コンテナー内にデプロイされた資格および機能に基づいて、以下の 1 つ以上のアノテーションを使用します。

注 : アノテーションの多くには、以下の行のいずれかまたは両方が含まれています。

```
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
```

アノテーションを使用する前に、以下の行を編集する必要があります。

- `productChargedContainers` の場合は、`"All"`を選択するか、コンテナーの実際の名前を置き換える必要があります。
- `productMetric` の場合は、提供されている値のいずれかを選択する必要があります。

IBM MQ 製品ライセンスで使用するアノテーション

IBM MQ 製品ライセンスがある場合は、購入した使用する資格に一致する以下のアノテーションを選択します。

- 284 ページの『IBM MQ』
- 284 ページの『IBM MQ 拡張』
- 284 ページの『IBM MQ (非実稼働環境)』
- 284 ページの『IBM MQ Advanced for Non-Production Environment (非実稼働環境向け拡張)』
- 284 ページの『IBM MQ 開発者向けアドバンス』

IBM MQ マルチインスタンス高可用性構成で使用する IBM MQ アノテーションは、以下のとおりです。 283 ページの『高可用性構成の正しいアノテーションの選択』も参照してください。

- 285 ページの『IBM MQ コンテナー・マルチインスタンス』
- 285 ページの『IBM MQ 拡張コンテナー・マルチインスタンス』
- 285 ページの『IBM MQ 非生産環境のコンテナ・マルチインスタンス』
- 285 ページの『IBM MQ 非生産環境のための高度なコンテナ・マルチインスタンス』

CP4I 製品ライセンスで使用するアノテーション

IBM Cloud Pak for Integration (CP4I) 資格がある場合は、購入して使用する資格に一致する以下の注釈を選択します。

- 285 ページの『IBM MQ with CP4I ライセンス』
- 285 ページの『IBM MQ Advanced with CP4I ライセンス』
- 285 ページの『IBM MQ for Non-Production Environment with CP4I の使用許諾』
- 285 ページの『IBM MQ Advanced for Non-Production Environment with CP4I の使用許諾』

IBM MQ マルチインスタンス高可用性構成で使用する CP4I アノテーションは、以下のとおりです。 283 ページの『高可用性構成の正しいアノテーションの選択』も参照してください。

- 286 ページの『IBM MQ Container Multi Instance with CP4I のライセンス』
- 286 ページの『IBM MQ Advanced Container Multi Instance with CP4I のライセンス』
- 286 ページの『IBM MQ Container Multi Instance for Non-Production Environment with CP4I の使用許諾』
- 286 ページの『IBM MQ Advanced Container Multi Instance for Non-Production Environment with CP4I のライセンス』

高可用性構成の正しいアノテーションの選択

IBM MQ マルチインスタンス

IBM MQ 複数インスタンスの高可用性構成でキュー・マネージャーのペアをデプロイする場合は、両方のインスタンスで同じアノテーションを使用する必要があります。 購入したライセンスに応じて、以下のいずれかのアノテーションを選択する必要があります。

- IBM MQ または IBM MQ Advanced スタンドアロン・ライセンス
  - 285 ページの『IBM MQ コンテナー・マルチインスタンス』
  - 285 ページの『IBM MQ 拡張コンテナー・マルチインスタンス』
  - 285 ページの『IBM MQ 非生産環境のコンテナ・マルチインスタンス』
  - 285 ページの『IBM MQ 非生産環境のための高度なコンテナ・マルチインスタンス』
- IBM Cloud Pak for Integration 資格
  - 286 ページの『IBM MQ Container Multi Instance with CP4I のライセンス』
  - 286 ページの『IBM MQ Advanced Container Multi Instance with CP4I のライセンス』
  - 286 ページの『IBM MQ Container Multi Instance for Non-Production Environment with CP4I の使用許諾』
  - 286 ページの『IBM MQ Advanced Container Multi Instance for Non-Production Environment with CP4I のライセンス』

IBM Cloud Pak for Integration 資格で使用する場合、アノテーション内の資格比率により、正しい資格使用量が記録されます。 スタンドアロンの IBM MQ または IBM MQ Advanced の資格で使用する場合は、以下のように、インスタンスごとに License Service で報告されるアノテーションを IBM MQ 資格パーツにマップする必要があります。

- IBM MQ Advanced Container マルチインスタンス
  - 1 x IBM MQ Advanced **および** 1 x IBM MQ Advanced 高可用性レプリカ **または**
  - 2 つの IBM MQ Advanced[5]
- 非実稼働環境用の IBM MQ Advanced Container マルチインスタンス
  - 1 x IBM MQ Advanced **および** 1 x IBM MQ Advanced 高可用性レプリカ **または**
  - 2 x IBM MQ Advanced (非実稼働環境の場合)[5]
- IBM MQ コンテナー・マルチインスタンス
  - 1 x IBM MQ **および** 1 x IBM MQ 高可用性レプリカ **または**
  - 2 つの IBM MQ[5]
- IBM MQ 非生産環境のコンテナ・マルチインスタンス
  - 1 x IBM MQ **および** 1 x IBM MQ 高可用性レプリカ **または**
  - 2 x IBM MQ (非実稼働環境の場合) [5]

IBM MQ ネイティブ HA

ネイティブ HA クォーラムに 3 つのキュー・マネージャーをデプロイする場合、アクティブ・インスタンスのみがライセンスを消費します。 すべてのインスタンスに同じアノテーションを付ける必要があります。 購入したライセンスに応じて、以下のいずれかを選択する必要があります。

- IBM MQ または IBM MQ Advanced スタンドアロン・ライセンス
  - 284 ページの『IBM MQ 拡張』
  - 284 ページの『IBM MQ Advanced for Non-Production Environment (非実稼働環境向け拡張)』
- IBM Cloud Pak for Integration 資格

[5] このライセンス・オプションは最適ではありません。関連する高可用性レプリカ・パーツのライセンスがない場合にのみ使用してください。

– 285 ページの『IBM MQ Advanced with CP4I ライセンス』
– 285 ページの『IBM MQ Advanced for Non-Production Environment with CP4I の使用許諾』

アノテーション

このトピックの残りの部分では、各アノテーションの内容について詳しく説明します。

IBM MQ

```
productID: "c661609261d5471fb4ff8970a36bccea"
productName: "IBM MQ"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

これらのライセンスアノテーションを使用するキューマネージャでネイティブ HA およびクロスリージョンレプリケーション機能を使用するには、ご使用の環境に合ったトピックを参照してください：

- 116 ページの『IBM MQ 、キューマネージャのネイティブ HA とクロスリージョンレプリケーションを設定するには、次のようにします。 IBM MQ Operator』
- 186 ページの『IBM MQ のキューマネージャでネイティブ HA とクロスリージョンレプリケーションを設定する。 Kubernetes』

IBM MQ 拡張

```
productID: "208423bb063c43288328b1d788745b0c"
productName: "IBM MQ Advanced"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ (非実稼働環境)

```
productID: "151bec68564a4a47a14e6fa99266deff"
productName: "IBM MQ for Non-Production Environment"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

これらのライセンスアノテーションを使用するキューマネージャでネイティブ HA およびクロスリージョンレプリケーション機能を使用するには、ご使用の環境に合ったトピックを参照してください：

- 116 ページの『IBM MQ 、キューマネージャのネイティブ HA とクロスリージョンレプリケーションを設定するには、次のようにします。 IBM MQ Operator』
- 186 ページの『IBM MQ のキューマネージャでネイティブ HA とクロスリージョンレプリケーションを設定する。 Kubernetes』

IBM MQ Advanced for Non-Production Environment (非実稼働環境向け拡張)

```
productID: "21dfe9a0f00f444f888756d835334909"
productName: "IBM MQ Advanced for Non-Production Environment"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ 開発者向けアドバンス

```
productID: "2f886a3eefbe4ccb89b2adb97c78b9cb"
productName: "IBM MQ Advanced for Developers (Non-Warranted)"
productMetric: "FREE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ コンテナー・マルチインスタンス

```
productID: "2dea73b866b648b6b4abe2a85eb76964"
productName: "IBM MQ Container Multi Instance"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ 拡張コンテナー・マルチインスタンス

```
productID: "bd35bff411bb47c2a3f3a4590f33a8ef"
productName: "IBM MQ Advanced Container Multi Instance"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ 非生産環境のコンテナ・マルチインスタンス

```
productID: "af11b093f16a4a26806013712b860b60"
productName: "IBM MQ Container Multi Instance for Non-Production Environment"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ 非生産環境のための高度なコンテナ・マルチインスタンス

```
productID: "31f844f7a96b49749130cd0708fdbb17"
productName: "IBM MQ Advanced Container Multi Instance for Non-Production Environment"
productMetric: "PROCESSOR_VALUE_UNIT" | "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
```

IBM MQ with CP4I ライセンス

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "c661609261d5471fb4ff8970a36bccea"
productName: "IBM MQ"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "4:1"
```

IBM MQ Advanced with CP4I ライセンス

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "208423bb063c43288328b1d788745b0c"
productName: "IBM MQ Advanced"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "2:1"
```

IBM MQ for Non-Production Environment with CP4I の使用許諾

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "151bec68564a4a47a14e6fa99266deff"
productName: "IBM MQ for Non-Production Environment"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers:  "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "8:1"
```

IBM MQ Advanced for Non-Production Environment with CP4I の使用許諾

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "21dfe9a0f00f444f888756d835334909"
```

```
productName: "IBM MQ Advanced for Non-Production Environment"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "4:1"
```

IBM MQ Container Multi Instance with CP4I のライセンス

```
productName: "IBM MQ Container Multi Instance"
productID: "2dea73b866b648b6b4abe2a85eb76964"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "10:3"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ Advanced Container Multi Instance with CP4I のライセンス

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "bd35bff411bb47c2a3f3a4590f33a8ef"
productName: "IBM MQ Advanced Container Multi Instance"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "5:3"
```

IBM MQ Container Multi Instance for Non-Production Environment with CP4I の使用許諾

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "af11b093f16a4a26806013712b860b60"
productName: "IBM MQ Container Multi Instance for Non-Production Environment"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "20:3"
```

IBM MQ Advanced Container Multi Instance for Non-Production Environment with CP4I のライセンス

```
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
cloudpakName: "IBM Cloud Pak for Integration"
productID: "31f844f7a96b49749130cd0708fdbb17"
productName: "IBM MQ Advanced Container Multi Instance for Non-Production Environment"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productCloudpakRatio: "10:3"
```

Linux IBM MQ によって適用されるコンテナライセンス注釈。 IBM MQ Operator

Kubernetes および Red Hat OpenShift Container Platform 環境では、IBM License Service は Pod アノテーションを使用して、基礎となるマシンではなくコンテナで定義された制限に基づいて使用量を追跡する。 IBM MQ Operator 、これらの Pod アノテーションを設定してくれますが、他の方法でデプロイする場合は、自分で設定する必要があります。

利用可能な注釈については、281 ページの『IBM MQ コンテナライセンス注釈』を参照。 次の表は、キューマネージャの YAML にあるこれらの値に基づいて、 IBM MQ Operator によってこれらのアノテーションのどれがポッドに適用されるかを説明します：

- の値である。 `spec.license.license`
- `spec.license.use` の値
- `spec.queueManager.availability.type` の値

注 : この表は、 IBM MQ 9.4.3.0 の最新の CD ライセンスを参照しています。 その他の IBM MQ バージョンのライセンスについては、 249 ページの『IBM MQ Operator のライセンスと API リファレンス』 を参照。

アノテーション (annotation)	Value of spec.license.license	spec.license.use の値	spec.queueManager.availability.type の値
IBM MQ Advanced	L-NUUP-23NH8Y	Production	SingleInstance または NativeHA
IBM MQ Advanced for Non-Production Environment	L-QWBN-T3PKLE	NonProduction	SingleInstance または NativeHA
IBM MQ Advanced for Developers	L-HYGL-6STWD6	Development	SingleInstance または NativeHA
IBM MQ Advanced Container Multi Instance	L-NUUP-23NH8Y	Production	MultiInstance
IBM MQ Advanced Container Multi Instance for Non-Production Environment	L-QWBN-T3PKLE	NonProduction	MultiInstance
IBM MQ Advanced CP4I 権利付き	L-CYPF-CRPF3H	Production	SingleInstance または NativeHA
IBM MQ Advanced for Non-Production Environment CP4I 権利付き	L-CYPF-CRPF3H	NonProduction	SingleInstance または NativeHA
IBM MQ Advanced Container Multi Instance CP4I 権利付き	L-CYPF-CRPF3H	Production	MultiInstance
IBM MQ Advanced Container Multi Instance for Non-Production Environment CP4I 権利付き	L-CYPF-CRPF3H	NonProduction	MultiInstance

Linux V 9.4.1 IBM MQ Advanced コンテナー・イメージ

IBM MQ Advanced にはビルド済みのコンテナイメージがあります。 この画像は IBM Container Registry から入手できます。

使用法

画像を使うには、環境変数 **LICENSE** を設定して、IBM MQ Advanced ライセンスの条件を受け入れる必要があります。

サポートされる環境変数

LANG
ライセンスを印刷する言語を設定します。

LICENSE
IBM MQ Advanced for Developers のライセンス条件に同意するには、`accept` を設定します。

ライセンス条件を表示するには、`view` を設定します。

MQ_CMDLEVEL
使用する MQ コマンドレベルを設定する。

mq_enable_embedded_web_server
`true` に設定すると、コンテナ起動時に Web サーバーを起動する。

MQ_ENABLE_FIPS
FIPS モードを有効にするには `true` を設定する。 FIPS モードを無効にするには `false` に設定する。 デフォルトでは、Linux ホストで FIPS が有効になっている場合、FIPS モードが自動的に有効になりますが、このオプションはテストに便利です。

mq_enable_metrics
`true` を設定すると、Prometheus メトリクスがキュー・マネージャ用に生成されます。

mq_enable_trace_crtmqdir
true" に設定すると、**crtmqdir** コマンドの実行中に MQ 診断トレースを作成する。

mq_enable_trace_crtmqm
true" に設定すると、**crtmqm** コマンドの実行中に MQ 診断トレースを作成する。

mq_enable_trace_strmqm
true" に設定すると、**strmqm** コマンドの実行中に MQ 診断トレースを作成する。

mq_grace_period
アプリケーションの切断のフェーズをエスカレートさせ、キューマネージャの終了を試みる目標時間。 コンテナが終了するときに使われる **endmqm -tp** オプションを設定します。 詳細は endmqm (end queue manager) を参照のこと。

mq_logging_console_source
コンテナの **stdout** ロケーションにミラーされるログのソースをカンマ区切りで指定します。

有効な値は、`qmgr`、`web`、`mqsc` です。

デフォルト値は `qmgr,web`。

オプション値は `mqsc`。 このオプションを使うと、`autocfgmqsc.LOG` の内容をコンテナログに反映させることができる。

mq_logging_console_format
コンテナの **stdout** の場所に出力されるログのフォーマットを変更する。

`basic` を設定すると、人間が読めるシンプルなフォーマットになります。 これがデフォルト値です。

`json` を設定して、JSON フォーマット（各行に 1 つの JSON オブジェクト）を使用する。

mq_logging_console_exclude_id
除外するログメッセージのメッセージ ID をカンマ区切りで指定します。

ログメッセージはディスク上のログファイルに表示されますが、コンテナの **stdout** の場所には出力されません。

デフォルト値は `AMQ5041I,AMQ5052I,AMQ5051I,AMQ5037I,AMQ5975I` です。

V 9.4.3 **mq_logging_metrics_audit_enabled**
Prometheus メトリクス・エンドポイントへのアクセスの監査ロギングを有効にするには、"true" に設定します。 ログ出力は `/var/mqm/errors/` の JSON ファイルに送られる。 また、 Prometheus メトリクスを生成するには、 `MQ_ENABLE_METRICS=true` を設定する必要があります。

mq_multi_instance
マルチインスタンス・キュー・マネージャとして動作させるためには "true "を設定する。 これは **endmqm** で使われるオプションを変更します。

MQ_NATIVE_HA
ネイティブ HA を有効にするには "true "を設定する。 これは **crtmqm -lr** オプションを設定します。 詳細は crtmqm (create queue manager)を参照のこと。 INI ファイルをマウントすることで、その他の Native HA 設定を行うことができる。 qm.ini ファイルの `NativeHAInstance` スタンザと'qm.ini ファイルの `NativeHALocalInstance` スタンザを参照。

Deprecated **MQ_NATIVE_HA_INSTANCE_0_NAME**
3 つの Native HA インスタンスのうちの 1 つに対して、「NativeHAInstanceINI スタンザ内の「Name 属性を設定する。 qm.ini ファイルの NativeHAInstance スタンザ を参照。 IBM MQ 9.4.1 から、この変数は非推奨となり、代わりに INI ファイルの断片を '/etc/mqm に与える必要がある。

Deprecated **MQ_NATIVE_HA_INSTANCE_1_NAME**
3 つの Native HA インスタンスのうちの 1 つに対して、「NativeHAInstanceINI スタンザ内の「Name 属性を設定する。 qm.ini ファイルの NativeHAInstance スタンザ を参照。 IBM MQ 9.4.1 から、この変数は非推奨となり、代わりに INI ファイルの断片を '/etc/mqm に与える必要がある。

Deprecated **MQ_NATIVE_HA_INSTANCE_2_NAME**
3 つの Native HA インスタンスのうちの 1 つに対して、「NativeHAInstanceINI スタンザ内の「Name 属性を設定する。 qm.ini ファイルの NativeHAInstance スタンザ を参照。 IBM MQ 9.4.1 から、この変数は非推奨となり、代わりに INI ファイルの断片を '/etc/mqm に与える必要がある。

Deprecated **MQ_NATIVE_HA_INSTANCE_0_REPLICATION_ADDRESS**
3 つの Native HA インスタンスのうちの 1 つに対して、「NativeHAInstanceINI スタンザ内の「ReplicationAddress 属性を設定する。 qm.ini ファイルの NativeHAInstance スタンザ を参照。 IBM MQ 9.4.1 から、この変数は非推奨となり、代わりに INI ファイルの断片を '/etc/mqm に与える必要がある。

Deprecated **MQ_NATIVE_HA_INSTANCE_1_REPLICATION_ADDRESS**
3 つの Native HA インスタンスのうちの 1 つに対して、「NativeHAInstanceINI スタンザ内の「ReplicationAddress 属性を設定する。 qm.ini ファイルの NativeHAInstance スタンザ を参照。 IBM MQ 9.4.1 から、この変数は非推奨となり、代わりに INI ファイルの断片を '/etc/mqm に与える必要がある。

Deprecated **MQ_NATIVE_HA_INSTANCE_2_REPLICATION_ADDRESS**
3 つの Native HA インスタンスのうちの 1 つに対して、「NativeHAInstanceINI スタンザ内の「ReplicationAddress 属性を設定する。 qm.ini ファイルの NativeHAInstance スタンザ を参照。 IBM MQ 9.4.1 から、この変数は非推奨となり、代わりに INI ファイルの断片を '/etc/mqm に与える必要がある。

Deprecated **mq_native_ha_cipherspec**
3 つの Native HA インスタンスのうちの 1 つに対して、「NativeHALocalInstanceINI スタンザ内の「CipherSpec 属性を設定する。 qm.ini ファイルの NativeHALocalInstance スタンザ を参照。 IBM MQ 9.4.1 から、この変数は非推奨となり、代わりに INI ファイルの断片を '/etc/mqm に与える必要がある。

Deprecated **mq_native_ha_key_repository**
Native HA 用に自動生成された鍵リポジトリを無視し、指定された鍵リポジトリを使用する。 IBM MQ 9.4.1 から、この変数は非推奨となり、代わりに'/etc/mqm の INI ファイルの断片に、キーリポジトリの場所を指定する必要がある。

mq_qmgr_log_file_pages
MQ の回復ログデータは、ログファイルと呼ばれる一連のファイルに保持される。 ログ・ファイル・サイズは、4 KB ページ単位で指定します。 この環境変数は **crtmqm -lf** オプションを設定する。 詳細は crtmqm (create queue manager)を参照のこと。

mq_qmgr_name
キュー・マネージャーを作成する際の名前を設定します。

ファイルシステムとマウントポイント

/etc/mqm
このディレクトリにある MQSC ファイルや INI ファイルは、キューマネージャの起動時に処理されます。 これは自動コンフィギュレーション機能で使用されるディレクトリである。 詳しくは、起動時の MQSC スクリプトからの自動設定と起動時の qm.ini の自動設定を参照してください。

/etc/mqm/pki/keys
PKCS #1、ASN.1DER 形式、または暗号化されていない PKCS #8、ASN.1DER 形式の秘密鍵が格納されたサブディレクトリ。 /etc/mqm/pki/keys 以下の各サブディレクトリは、.key の拡張子を持つファイルがないかスキャンされます。 公開 X.509 証明書チェーンは、ASN.1DER フォーマットで、「.crt 拡張子を持つ 1 つ以上のファイルとして、同じディレクトリにあるべきである。 このようにして鍵がマウントされると、「/run/runmqserver/tls」に鍵リポジトリが自動的に生成される。 キーリポジトリはキューマネージャの'SSLKEYR 設定に設定され、デフォルトのキー（CERTLABL）は、キー名のリストを辞書順にソートした後の最初の名前になります。

/etc/mqm/pki/trust
X.509 DER 形式の ASN.1 公開証明書を含むサブディレクトリ。 各サブディレクトリは、.crt ファイルがないかスキャンされ、信頼できる証明書として追加されます。 秘密鍵の証明書は自動的に信頼されるので、この方法で追加する必要はない。

/etc/mqm/ha/pki/keys
PKCS #1、ASN.1DER 形式、または暗号化されていない PKCS #8、ASN.1DER 形式の秘密鍵が格納されたサブディレクトリ。 /etc/mqm/ha/pki/keys 以下の各サブディレクトリは、.key の拡張子を持つファイルがないかスキャンされます。 公開 X.509 証明書チェーンは、ASN.1DER フォーマットで、「.crt 拡張子を持つ 1 つ以上のファイルとして、同じディレクトリにあるべきである。 この方法で鍵がマウントされると、鍵リポジトリが自動的に生成され、その鍵リポジトリを使用するための INI ファイルの断片も生成される。 使用されるキーは、キー名のリストを辞書順にソートした後の最初の名前となる。 生成されたキー・リポジトリーは'/run/runmqserver/ha/tls に書き込まれる。

/etc/mqm/metrics/pki/keys
9.4.2.0-r1 から、このディレクトリに TLS キーが提供され、キューマネージャーでメトリクスが有効になっている場合、HTTPS サーバーが /metrics ポート 9157 で起動します。 このディレクトリ内のファイルは、PEM エンコードされた証明書であるべきです

- tls.crt サーバ公開証明書
- tls.key サーバー秘密鍵
- ca.crt CA 公開証明書（オプション）

TLS キーが提供されない場合、または以前のバージョンの IBM MQ が使用される場合、 HTTP サーバーが起動します。

/etc/mqm/web
/etc/mqm/web の下のディレクトリツリーは、コンテナ起動時に/var/mqm/web の上にコピーされる。 例えば、/etc/mqm/web/installations/Installation1/servers/mqweb/mqwebuser.xml を作成してウェブサーバーを設定することができます。 このファイルは、通常マウントされたボリュームである「/var/mqm/web 下の正しい場所にコピーされる。 IBM MQ Console と 'REST API 設定を参照のこと。

/mnt/mqm
/var/mqm ディレクトリはこの場所にシンボリックリンクされている。 この場所に永続ボリュームをマウントすることで、MQ データをコンテナの実行にまたがって永続化できる。

/mnt/mqm-log
キューマネージャのリカバリログファイルを保持するためのディレクトリ。 ボリュームがこの場所にマウントされている場合、コンテナコードは **crtmqm -ld** オプションを設定する。 詳細は crtmqm (create queue manager)を参照のこと。

/mnt/mqm-data
キューマネージャのデータファイルを保持するためのディレクトリ。 ボリュームがこの場所にマウントされている場合、コンテナコードは **crtmqm -md** オプションを設定する。 詳細は crtmqm (create queue manager)を参照のこと。

/run
コンテナは、このディレクトリの下に一時ファイルを書き込む。 コンテナに読み取り専用のファイルシステムを使いたい場合は、この場所にボリュームをマウントし、/tmp にマウントする必要がある

/run/termination-log
コンテナ・コードがキュー・マネージャを終了させるようなエラーに遭遇した場合、その理由をこのファイルに書き込む。 このファイルの場所は、Kubernetes で終了情報を取得するためにデフォルトで使われます。

/tmp
コンテナは、このディレクトリの下に一時ファイルを書き込む。 コンテナに読み取り専用のファイルシステムを使いたい場合は、この場所にボリュームをマウントし、/run にマウントする必要がある

Linux IBM MQ ライセンスコンテナイメージ

IBM MQ には、ビルド済みのコンテナ・イメージが用意されている。 この画像は IBM Container Registry。

使用法

イメージを使用するには、**LICENSE** 環境変数を設定し、IBM MQ ライセンスの条項に同意する必要があります。 詳しくは、161 ページの『を使用して、 IBM MQ ライセンスアノテーションでキューマネージャーを設定します。 IBM MQ Operator』を参照してください。

サポートされる環境変数

IBM MQ ライセンスのコンテナ・イメージでサポートされる環境変数は、 IBM MQ Advanced コンテナ・イメージでサポートされる環境変数と同じである。 完全なリストについては、IBM MQ Advanced コンテナ・イメージがサポートする環境変数を参照のこと。

注 : IBM MQ 9.4.2 から、 IBM MQ ライセンスを取得したキューマネージャは、キューマネージャにライセンスを追加することで、ネイティブ HA およびクロスリージョンレプリケーション機能を使用するように構成できます。 詳しくは、116 ページの『IBM MQ 、キューマネージャのネイティブ HA とクロスリージョンレプリケーションを設定するには、次のようにします。 IBM MQ Operator』を参照してください。 IBM MQ ライセンスを使用する IBM MQ 9.4.2 より前のキューマネージャでは、ネイティブ HA 機能は使用できません。

ファイルシステムとマウントポイント

IBM MQ ライセンス・コンテナ・イメージでサポートされるファイル・システムとマウント・ポイントは、IBM MQ Advanced コンテナ・イメージでサポートされるものと同じです。 完全なリストについては、IBM MQ Advanced コンテナ・イメージのファイル・システムとマウント・ ポイントを参照。

OpenShift CP4I Kubernetes IBM MQ Advanced for Developers コンテナー・イメージ

IBM MQ Advanced for Developers には、事前ビルドされたコンテナー・イメージが用意されています。 このイメージは、 IBM Container Registry から入手できます。 このイメージは、 Docker、 Podman、 Kubernetes、その他のコンテナ環境での使用に適しています。

使用可能なイメージ

IBM MQ イメージは、 IBM Container Registry に保管されます。

- IBM MQ Advanced for Developers 9.4.0.11: `icr.io/ibm-messaging/mq:9.4.0.11-r3`
- IBM MQ Advanced for Developers 9.4.3.0: `icr.io/ibm-messaging/mq:9.4.3.0-r1`

クイック・リファレンス

- ライセンス:
  - 249 ページの『mq.ibm.com/v1beta1: 現在のライセンスバージョン』と Apache ライセンス 2.0. IBM MQ Advanced for Developers ライセンスではこれ以上の配布は許可されておらず、条件によって開発者マシンへの使用が制限されていることに注意してください。

- 問題のファイリング先:
  - GitHub
- 以下の CPU アーキテクチャーで使用可能です。
  - amd64
  - s390x
  - ppc64le

使用法

コンテナーで IBM MQ Advanced for Developers を実行します。

コンテナーの実行方法について詳しくは、 使用法の資料 を参照してください。

イメージを使用できるようにするには、**LICENSE** 環境変数を設定して IBM MQ ライセンスの条項に同意する必要があります。

環境変数

287 ページの『IBM MQ Advanced コンテナー・イメージ』の環境変数がすべて利用でき、さらに以下の追加変数が利用できます：

Deprecated **MQ_ADMIN_PASSWORD**
admin ユーザーのパスワードを指定します。

長さは 8 文字以上でなければなりません。

admin ユーザーのデフォルト・パスワードはありません。

V 9.4.0 V 9.4.0 IBM MQ 9.4.0 以降、この変数は提供されなくなりました。 このトピックの YAML の例では、シークレットをマウントすることで管理者パスワードを提供する方法を示しています。

Deprecated **MQ_APP_PASSWORD**
アプリケーション・ユーザーのパスワードを指定します。

これを設定すると、**DEV.APP.SVRCONN** チャネルが保護され、有効なユーザー ID とパスワードを提供する接続のみが許可されます。

長さは 8 文字以上でなければなりません。

アプリケーション・ユーザーのデフォルト・パスワードはありません。

V 9.4.0 V 9.4.0 IBM MQ 9.4.0 以降、この変数は提供されなくなりました。 このトピックの YAML 例では、シークレットをマウントしてアプリのパスワードを提供する方法を示している。

MQ_DEV
作成中のデフォルト・オブジェクトを停止するには、 `false` に設定します。

IBM MQ Advanced for Developers イメージでサポートされるデフォルトの開発者構成について詳しくは、デフォルトの開発者構成の資料を参照してください。

admin および app ユーザーのパスワードを指定する方法を説明するキュー・マネージャー YAML の例

admin ユーザー ID および **app** ユーザー ID のユーザーの場合、`Development` ライセンスを使用してキュー・マネージャーをデプロイするときにパスワードを指定する必要があります。 以下に、IBM MQ Operator でこれを行う方法を示すキュー・マネージャー YAML の例を示します。

以下のコマンドは、 **admin** および **app** ユーザーのパスワードを含むシークレットを作成します。

```
oc create secret generic my-mq-dev-passwords --from-literal=mqAdminPassword=passw0rd --from-
literal=mqAppPassword=passw0rd
```

以下の YAML は、キュー・マネージャーのデプロイ時にこれらのパスワードを使用します。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm-dev
  annotations: com.ibm.mq/write-defaults
  spec: 'false'
spec:
  license:
    accept: true
    license: L-HYGL-6STWD6
    use: Development
  web:
    enabled: true
  queueManager:
    env:
      - name: MQ_DEV
        value: "true"
      - name: MQ_CONNAUTH_USE_HTP
        value: "true"
    files:
      - secret:
          items:
            - item: mqAdminPassword
            - item: mqAppPassword
      name: my-mq-dev-passwords
      defaultMountPath: /run/secrets
    storage:
      queueManager:
        type: persistent-claim
    name: QUICKSTART
  version: 9.4.3.0-r1
```

IBM MQ MFT Agent for Developers コンテナー・イメージ

IBM MQ MFT Agent for Developers には、ビルド済みのコンテナ・イメージが用意されている。 この画像は IBM Container Registry。 このイメージは、 Docker、 Podman、 Kubernetes、その他のコンテナ環境での使用に適しています。

重要： Deprecated IBM MQ MFT Agent for Developers イメージは以前は Hub から入手できましたが、これは非推奨となっており、 Hub ではこれ以上のアップデートはできません。 Docker Docker

使用可能なイメージ

IBM MQ MFT Agent for Developers 画像は に保存される： IBM Container Registry

- IBM MQ 9.3.0.0: `icr.io/ibm-messaging/mqmft:latest`

クイック・リファレンス

- ライセンス:
  - IBM MQ Advanced for Developers および Apache ライセンス 2.0. IBM MQ Advanced for Developers ライセンスは、それ以上の配布を許可しておらず、使用は開発者用マシンに限定されている。
- 問題の提出先
  - GitHub
- 以下の CPU アーキテクチャで利用可能：
  - amd64

コンテナー・イメージのビルド

amd64 アーキテクチャの MQ Managed File Transfer 再頒布可能パッケージを使用して、独自のエージェントコンテナイメージを構築する方法については、 使用説明書を参照してください。

使用法

Podman ランタイムを使用してイメージ・コンテナを実行する方法の詳細については、 使用法を参照してください。

OpenShift Container Platform にイメージを配置する方法の詳細については、使用説明書を参照してください。

イメージを使用するには、**LICENSE** 環境変数を設定し、IBM MQ ライセンスの条項に同意する必要があります。

環境変数

LICENSE
IBM MQ Advanced for Developers のライセンス条件に同意する場合は `accept` を設定してください。

ライセンス条件を表示するには、 ビューを設定します。

mft_agent_config_file
必須。

エージェントの設定に必要な情報を含む json ファイルのパス。 パスはマウントポイント上になければならない。 例えば `configMap` を OpenShift。 属性の詳細については、 追加 MFT エージェント設定ファイルを参照してください。

MFT_AGENT_NAME
admin ユーザーのパスワードを指定する。

設定するエージェントの名前。

BFG_JVM_PROPERTIES
オプション。

エージェント JVM の実行時に設定する必要がある JVM プロパティ。

MFT_LOG_LEVEL
オプション。

表示される情報のレベル。 `info` と `verbose` がサポートされており、 `info` が デフォルトである。 MFT_LOG_LEVEL が `verbose` に設定されている場合、エージェントの `output0.log` の内容が表示されます。

mft_agent_start_wait_time
オプション。

fteStartAgent コマンドが発行された後、エージェントの起動に時間がかかる場合があります。 これは、コンテナがエージェントの開始を待つ時間（秒）である。 エージェントが指定された待ち時間内に待機しない場合、コンテナは終了する。

MFT_MOUNT_PATH
オプション。

エージェントがファイルを読み込んだり、書き込んだりするパスを指す環境変数。

mft_coord_qmgr_cipher
コーディネーション・キュー・マネージャへのセキュアな接続に使用される CipherSpec の名前。

mft_cmd_qmgr_cipher
コマンド・キュー・マネージャに安全に接続するために使用する CipherSpec の名前。

mft_agent_qmgr_cipher
エージェント・キュー・マネージャに安全に接続するために使用する CipherSpec の名前。

IBM MQ Advanced for Developers イメージでサポートされているデフォルトの開発者設定についての詳細は、 デフォルトの開発者設定のドキュメントを参照してください。

エージェント設定ファイルの場所

コンテナ内のエージェントは、固定ディレクトリ /mnt/mftdata の下にエージェント設定ファイルとログファイルを作成する。 このフォルダは、永続ボリューム上にも置くことができる。その場合、そのボリュームを /mnt/mftdata マウント・ポイントとしてコンテナにマウントする必要がある。

OpenShift CP4I IBM の MQ CASE は、「Red Hat OpenShift」にエアギャップのインストールとアップグレードを行います。

IBM MQ Operator と'IBM MQ キューマネージャをエアギャップされた'Red Hat OpenShift クラスタにインストールする場合、IBM MQ CASE を使用します。 IBM MQ CASE には、IBM MQ CASE の特定のリリースを構成するすべてのイメージとメタデータの詳細が含まれています。

エアギャップと IBM MQ CASE 3.5.0 の変更

IBM MQ CASE 3.5.0 以降、IBM MQ CASE にはキューマネージャーの単一バージョンの画像のみが含まれています。 これは、対応する IBM MQ Operator が展開できるキューマネージャーの最新バージョンです。 IBM MQ CASE 3.5.0 以前は、 CASE には、対応するオペレータが展開可能なすべてのキューマネージャーのバージョンが含まれていました。 各 IBM MQ CASE で使用可能な正確なバージョンについては、IBM MQ CASE ルックアップ テーブルを参照してください。

IBM MQ CASE を使用して、オペレータが展開可能な最新のキューマネージャーをインストールまたはアップグレードすることができます。つまり、 CD または SC2 キューマネージャーは、 CASE および対応するオペレータと同時にリリースされました。 この方法では、インストールまたはアップグレードのプロセスは同じですが、ミラーリングするイメージが少ないため、より短時間で完了します。

ただし、オペレータが展開できる最新バージョン以外の 1 つまたは複数のキューマネージャーをインストールする場合は、必要なキューマネージャーのバージョンごとに個別の IBM MQ CASE を使用して、エアギャップイメージのミラーリングプロセスを繰り返す必要があります。 各 IBM MQ CASE で使用可能な正確なバージョンについては、 IBM MQ CASE ルックアップ テーブルを参照してください。

注 : IBM MQ Operator は依然として古いキューマネージャーのバージョンを認識しているため、古いキューマネージャーのバージョンの展開を要求することができます。 このため、どのキューマネージャーのバージョンをミラーリングする必要があるかを考慮することが重要です。

キューマネージャー関連画像

IBM MQ CASE 3.5.0 以降、IBM MQ CASE には、対応するオペレータが展開できる最新のキューマネージャーバージョンの画像のみが含まれています。 これ以前は、 CASE には、対応するオペレータが展開可能なすべてのキューマネージャーのバージョンが含まれています。

注 : 3.5.0、 IBM、 MQ、 CASE から、最新キューマネージャーのバージョンではオペレーションダッシュボードを展開できないため、オペレーションダッシュボードの画像は削除されました。

IBM MQ CASE の参照テーブルを使用して、必要な IBM MQ CASE がどの IBM MQ キューマネージャーのバージョンに該当するかを特定します。

オペレーター関連画像

IBM MQ CASE の参照テーブルを使用して、必要な IBM MQ Operator のバージョンがどの IBM MQ CASE に含まれているかを確認します。

重要 :

- IBM MQ Operator をエアギャップ環境にインストールする際には、インストールする予定の IBM MQ Operator のバージョンに対応する IBM MQ CASE のエアギャップミラーリングプロセスを必ず完了させてください。 異なるオペレータバージョンをインストールしたり、アップグレードしたりしないでください。 IBM MQ CASE。
- IBM MQ CASE には、追加の IBM MQ Operator バージョンが含まれていますが、ミラーリングした IBM MQ CASE に対応するオペレータのみをインストールしてください。

- IBM MQ Operator をアップグレードする際、オペレータチャンネルを変更する場合は、現在のチャンネルで利用可能な最新バージョンの IBM MQ Operator にアップグレードするために、エアギャップアップグレードのベストプラクティスに従ってください。 新しい IBM MQ Operator チャンネルに購読を変更する前に、この作業を行うことが重要です。

IBM MQ CASE ルックアップテーブル

表 *3.* 最近の *CD* インスタンスの内容 *IBM MQ CASE*

IBM MQ CASE	**IBM MQ オペレーター**	**IBM MQ キュー・マネージャー**
3.6.0	3.6.0	9.4.3.0-r1
3.5.3	3.5.3	9.4.2.1-r2
3.5.2	3.5.2	9.4.2.1-r1
3.5.1	3.5.1	9.4.2.0-r2
3.5.0	3.5.0	9.4.2.0-r1
3.4.1	3.4.1	9.4.1.1-r1, 9.4.1.0-r2, 9.4.1.0-r1, 9.4.0.7-r1, 9.4.0.6-r2, 9.4.0.6-r1, 9.4.0.5-r2, 9.4.0.5-r1, 9.4.0.0-r3, 9.4.0.0-r2, 9.4.0.0-r1, 9.3.5.1-r2, 9.3.5.1-r1, 9.3.5.0-r2, 9.3.5.0-r1, 9.3.4.1-r1, 9.3.4.0-r1, 9.3.3.3-r2, 9.3.3.3-r1, 9.3.3.2-r3, 9.3.3.2-r2, 9.3.3.2-r1, 9.3.3.1-r2, 9.3.3.1-r1, 9.3.3.0-r2, 9.3.3.0-r1, 9.3.2.1-r2, 9.3.2.1-r1, 9.3.2.0-r2, 9.3.2.0-r1, 9.3.1.1-r1, 9.3.1.0-r3, 9.3.1.0-r2, 9.3.1.0-r1, 9.3.0.25-r1, 9.3.0.21-r3, 9.3.0.21-r2, 9.3.0.21-r1, 9.3.0.20-r2, 9.3.0.20-r1, 9.3.0.17-r3, 9.3.0.17-r2, 9.3.0.17-r1, 9.3.0.16-r2, 9.3.0.16-r1, 9.3.0.15-r1, 9.3.0.11-r2, 9.3.0.11-r1, 9.3.0.10-r2, 9.3.0.10-r1, 9.3.0.6-r1, 9.3.0.5-r3, 9.3.0.5-r2, 9.3.0.5-r1, 9.3.0.4-r2, 9.3.0.4-r1, 9.3.0.3-r1, 9.3.0.1-r4, 9.3.0.1-r3, 9.3.0.1-r2, 9.3.0.1-r1, 9.3.0.0-r3, 9.3.0.0-r2, 9.3.0.0-r1

表 4. IBM、 MQ の最近の SC2 インスタンスの内容 CASE		
IBM MQ CASE	**IBM MQ オペレーター**	**IBM MQ キュー・マネージャー**
3.2.13	3.2.13	9.4.0.11-r3, 9.4.0.11-r2, 9.4.0.11-r1, 9.4.0.10-r2, 9.4.0.10-r1, 9.4.0.7-r1, 9.4.0.6-r2, 9.4.0.6-r1, 9.4.0.5-r2, 9.4.0.5-r1, 9.4.0.0-r3, 9.4.0.0-r2, 9.4.0.0-r1, 9.3.5.1-r2, 9.3.5.1-r1, 9.3.5.0-r2, 9.3.5.0-r1, 9.3.4.1-r1, 9.3.4.0-r1, 9.3.3.3-r2, 9.3.3.3-r1, 9.3.3.2-r3, 9.3.3.2-r2, 9.3.3.2-r1, 9.3.3.1-r2, 9.3.3.1-r1, 9.3.3.0-r2, 9.3.3.0-r1, 9.3.2.1-r2, 9.3.2.1-r1, 9.3.2.0-r2, 9.3.2.0-r1, 9.3.1.1-r1, 9.3.1.0-r3, 9.3.1.0-r2, 9.3.1.0-r1, 9.3.0.25-r1, 9.3.0.21-r3, 9.3.0.21-r2, 9.3.0.21-r1, 9.3.0.20-r2, 9.3.0.20-r1, 9.3.0.17-r3, 9.3.0.17-r2, 9.3.0.17-r1, 9.3.0.16-r2, 9.3.0.16-r1, 9.3.0.15-r1, 9.3.0.11-r2, 9.3.0.11-r1, 9.3.0.10-r2, 9.3.0.10-r1, 9.3.0.6-r1, 9.3.0.5-r3, 9.3.0.5-r2, 9.3.0.5-r1, 9.3.0.4-r2, 9.3.0.4-r1, 9.3.0.3-r1, 9.3.0.1-r4, 9.3.0.1-r3, 9.3.0.1-r2, 9.3.0.1-r1, 9.3.0.0-r3, 9.3.0.0-r2, 9.3.0.0-r1

前の演算子の削除

参考までに、 IBM MQ CASE 3.4.0 から、以下のチャンネルの IBM MQ Operator 画像が IBM MQ CASE から削除されました

- v1.1
- v1.2
- v1.4
- v1.5
- v1.6
- v1.7
- v2.1

クラスタとイメージレジストリの再作成

Red Hat OpenShift クラスタにデプロイされているすべてのキューマネージャーと、それらのバージョンをリストアップするには、次のコマンドを使用します。

```
oc get queuemanager --all-namespaces -o=jsonpath='{range .items[*]}{.metadata.name}{":\t"}
{.spec.version}{"\n"}{end}'
```

このコマンドを定期的に実行し、出力を保存してください。 このリストは、 Red Hat OpenShift クラスタ、または air-gap 用にミラーリングされたイメージを保存するイメージレジストリを再作成したい場合に役立ちます。

関連タスク

76 ページの『CD「IBM MQ Operator」を現在の「CD チャンネルにアップグレードする』
IBM MQ **ライセンス**のみが使用されている「IBM MQ Operator「CD デプロイメントをアップグレードします。

75 ページの『3.2.x の'SC2'IBM MQ Operator を現在の'CD チャンネルにアップグレードする』
IBM MQ **ライセンス**のみが使用されている 'IBM MQ Operator 3.2.x'SC2 デプロイメントをアップグレードします。

57 ページの『IBM MQ Operator のインストール』
IBM MQ Operator は、 OpenShift コンソールまたはコマンド・ライン・インターフェース (CLI) を使用して Red Hat OpenShift にインストールできます。

65 ページの『CP4I で使用するための IBM MQ Operator のインストール』
IBM Cloud Pak for Integration (CP4I) で使用する場合、 OpenShift コンソールまたはコマンド・ライン・インターフェース (CLI) を使用して、 IBM MQ Operator を Red Hat OpenShift にインストールできます。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

IBM 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。 日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。 これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。 ただし、IBM 以外の製品とプログラムの操作またはサービスの 評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。 本書の提供は、お客様にこれらの特許権について 実施権を許諾することを意味するものではありません。 実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 103-8510
東京都中央区日本橋箱崎町 19 番 21 号
日本アイ・ビー・エム株式会社
日本アイ・ビー・エム株式会社
法務・知的財産
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
〒 103-8510
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 INTERNATIONAL BUSINESS MACHINES CORPORATION は、法律上の瑕疵担保責任、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。 "" 国または地域によっては、法律の強行規定により、保証責任の制限が 禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。 本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。 それらの Web サイトにある資料は、この IBM 製品の資料の一部では ありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

東京都中央区日本橋箱崎町 19 番 21 号
日本アイ・ビー・エム株式会社
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

© Copyright IBM Corp. 2007 年, 2025

本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他の ライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で 決定されたものです。 そのため、他の操作環境で得られた結果は、異なる可能性があります。 一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。 さらに、一部の測定値が、推定値である可能性があります。 実際の結果は、異なる可能性があります。 お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、 もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。 したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。 より具体性を与えるために、それらの例には、個人、企業、ブランド、 あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、 名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。 お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。 このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。 従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性が あることをほのめかしたり、保証することはできません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミング・インターフェース情報 (提供されている場合) は、このプログラムで使用するアプリケーション・ソフトウェアの作成を支援することを目的としています。

本書には、プログラムを作成するユーザーが IBM MQ のサービスを使用できるようにするためのプログラミング・インターフェースに関する情報が記載されています。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。 診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

重要 : この診断、修正、およびチューニング情報は、変更される可能性があるため、プログラミング・インターフェースとして使用しないでください。

商標

IBM、 IBM ロゴ、 ibm.com®は、世界の多くの国で登録された IBM Corporation の商標です。 現時点での IBM の商標リストについては、 "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml をご覧ください。 他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Linux は、Linus Torvalds 氏の米国およびその他の国における登録商標です。

この製品には、 Eclipse Project (https://www.eclipse.org/) により開発されたソフトウェアが含まれています。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

部品番号:

(1P) P/N: