9.2

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、163 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。

目次

Multi 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。

MQ Adv. CD 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。

Multi 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 (Red Hat OpenShift Container Platform のアーキテクチャーを参照してください) 。

手順

- 5 ページの『コンテナー内の IBM MQ の使用方法の選択』.
- 6 ページの『IBM MQ Operator のサポート』.
- 13 ページの『独自の IBM MQ キュー・マネージャー・コンテナー・イメージを作成するためのサポート』.
- 11 ページの『IBM MQ Operator のストレージに関する考慮事項』.
- 16 ページの『コンテナー内の IBM MQ の高可用性』.
- 18 ページの『コンテナー内の IBM MQ の災害復旧』.
- 18 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

OpenShift CP4I

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、新しい `QueueManager` カスタム・リソースを Red Hat OpenShift Container Platform に追加します。 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。 157 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

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。 以下のいずれかを行う場合は、独自のイメージとチャートをビルドする必要があります。

- 管理またはメッセージングに REST API を使用する
- 以下のいずれかの MQ コンポーネントを使用する
  - Managed File Transfer Agent とそのリソース。 ただし、 IBM MQ Operator を使用して、1 つ以上の調整キュー・マネージャー、コマンド・キュー・マネージャー、またはエージェント・キュー・マネージャーを指定できます。
  - AMQP
  - IBM MQ Bridge to Salesforce
  - IBM MQ Bridge to blockchain (コンテナー内ではサポートされません)
  - IBM MQ Telemetry Transport (MQTT).
- **crtmqm**、**strmqm**、**endmqm** で使用するオプションをカスタマイズします (ログ・ファイル・ページの構成など)。 ほとんどのオプションは、INI ファイルを使用して構成できます。

IBM MQ Operator コンテナーは速いペースで発展しているので、Long Term Support リリースではサポートされないことに注意してください。

IBM MQ Operator には、事前にビルドされたコンテナー・イメージと、Red Hat OpenShift Container Platform 上で実行するためのデプロイメント・コードの両方が含まれています。 IBM MQ Operator を使用すると、提供された IBM MQ コンテナー・イメージまたはその上に階層化されたコンテナー・イメージをデプロイすることはできますが、カスタム・ビルドされた MQ コンテナー・イメージをデプロイすることはできません。

独自のイメージおよびデプロイメント・コードのビルド

Multi

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。 Red Hat OpenShift Container Platform を使用しない場合は、独自のイメージとデプロイメント・コードをビルド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。 149 ページの『独自の IBM MQ コンテナーおよびデプロイメント・コードのビルド』を参照してください。

関連概念

6 ページの『IBM MQ Operator のサポート』
IBM MQ Operator は、Red Hat OpenShift Container Platform 上でデプロイされる場合に限りサポートされます。

13 ページの『独自の IBM MQ キュー・マネージャー・コンテナー・イメージを作成するためのサポート』
IBM MQ は、 GitHub 上に IBM MQ キュー・マネージャー・コンテナーを作成するためのコードを提供します。 これは、 IBM が独自のサポート・コンテナーを構築するために使用するプロセスに基づいており、この GitHub リポジトリーを使用して、独自のコンテナー・イメージの作成を単純化および加速することができます。

OpenShift CP4I CD EUS IBM MQ Operator のサポート

IBM MQ Operator は、Red Hat OpenShift Container Platform 上でデプロイされる場合に限りサポートされます。

IBM MQ Operator は、 IBM MQ Continuous Delivery (CD) リリースに基づくイメージを使用しますが、IBM Cloud Pak for Integration では拡張更新サポート (EUS) リリースを使用できます。 CD リリースのサポート期間は、1 年または 2 回の CD リリースのどちらか長い方です。 Long Term Support リリースの IBM MQ は IBM MQ Operator では使用できません。 IBM Cloud Pak for Integration 2020.4.1 は、使用する IBM MQ のバージョンが -eus とマークされている場合は拡張更新サポート (EUS) リリースです。このリリースのサポート期間は 18 カ月です。 それ以外の場合、IBM MQ 9.2 は Continuous Delivery リリース (IBM MQ Operator を持つ) とみなされます。

IBM MQ Operator は、IBM MQ で使用される主要な Linux®のライブラリやユーティリティを含む Red Hat Universal Base Image（UBI）上の IBM MQ のインストールを提供するコンテナイメージを使用しています。 UBI は、 Red Hat OpenShift での実行時に Red Hat によってサポートされます。

IBM MQ Operator は、amd64 アーキテクチャーおよび s390x (z/Linux) アーキテクチャーでサポートされます。

関連概念

13 ページの『独自の IBM MQ キュー・マネージャー・コンテナー・イメージを作成するためのサポート』
IBM MQ は、 GitHub 上に IBM MQ キュー・マネージャー・コンテナーを作成するためのコードを提供します。 これは、 IBM が独自のサポート・コンテナーを構築するために使用するプロセスに基づいており、この GitHub リポジトリーを使用して、独自のコンテナー・イメージの作成を単純化および加速することができます。

IBM MQ Operator のバージョン・サポート

IBM MQ、Red Hat 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。

- 7 ページの『使用可能な IBM MQ のバージョン』
- 8 ページの『互換性のある Red Hat OpenShift Container Platform のバージョン』
- 8 ページの『IBM Cloud Pak for Integration のバージョン』
- 8 ページの『以前のので使用可能な IBM MQ バージョン』
- 9 ページの『以前のオペレーターの互換性のある Red Hat 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							
		9.1.5	9.2.0 CD	9.2.0 EUS	9.2.1	9.2.2	9.2.3	9.2.4	9.2.5
v1.6	1.6	⚠	⚠	→	⚠	●	●		
v1.7	1.7	⚠	⚠	→	⚠	●	●	●	
v1.8	1.8	⚠	⚠	→	⚠	⚠	●	●	●

キー

●
使用可能な Continuous Delivery サポート

▯
使用可能な Extended Update Support

→
Extended Update 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。

⚠
非推奨。 IBM MQ リリースはサポートから解放されるため、オペレーターではまだ構成可能ですが、サポートの対象として適格ではなくなり、将来のリリースでは削除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、20 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	Red Hat OpenShift Container Platform のバージョン [1]				
		4.6	4.7[2]	4.8	4.9	4.10
v1.6	1.6	●	●	●	●	●
v1.7	1.7	●	●	●	●	●
v1.8	1.8	●	●	●	●	●

キー

●
使用可能な Continuous Delivery サポート

▯
使用可能な Extended Update Support

IBM Cloud Pak for Integration のバージョン

IBM MQ Operator 1.8.x は、IBM Cloud Pak for Integration バージョン 2021.4.1 の一部として使用するためにサポートされているとともに、単独でもサポートされています。

IBM MQ Operator 1.7.x は、 IBM Cloud Pak for Integration バージョン 2021.4.1 の一部として、または独立して使用するためにサポートされています。

IBM MQ Operator 1.6.x は、 IBM Cloud Pak for Integration バージョン 2021.2.1、 2021.3.1、または独立の一部として使用することがサポートされています。

IBM MQ Operator 1.5.x はサポートされなくなりました。

IBM MQ Operator 1.4.x はサポートされなくなりました。

IBM MQ Operator 1.3.x はサポートされなくなりました。

IBM MQ Operator 1.2.x はサポートされなくなりました。

IBM MQ Operators 1.1.x および 1.0.x はサポートされなくなりました。

以前ので使用可能な IBM MQ バージョン

以下の表は、「生命の終わり」に到達した IBM MQ Operator のバージョンに適用されます。

オペレーター・チャネル	Operator バージョン	IBM MQ のバージョン							
		9.1.5	9.2.0 CD	9.2.0 EUS	9.2.1	9.2.2	9.2.3	9.2.4	9.2.5
v1.0	1.0	⚠							
v1.1	1.1	⚠	⚠						
v1.2	1.2	⚠	⚠						
v1.3-eus	1.3	⚠	⚠	⚠					
v1.4	1.4	⚠	⚠	→	⚠				

[1] Red Hat 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、 Red Hat OpenShift Container Platform ライフ・サイクル・ポリシー を参照してください。

[2] IBM MQ Operator は、 IBM Cloud Pak foundational services によって異なります。 Red Hat OpenShift Container Platform 4.7 を使用したい場合は、まず IBM Cloud Pak foundational services のバージョンをアップグレードする必要があります。

オペレーター・チャネル	Operator バージョン	IBM MQ のバージョン							
		9.1.5	9.2.0 CD	9.2.0 EUS	9.2.1	9.2.2	9.2.3	9.2.4	9.2.5
v1.5	1.5	⚠	⚠	→	⚠	⚠			

キー

→
Extended Update 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。

⚠
非推奨。 IBM MQ リリースはサポートから解放されるため、 IBM MQ Operator ではまだ構成可能ですが、サポートの対象として適格ではなくなりました。

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、20 ページの『IBM MQ Operator のリリース履歴』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 Red Hat OpenShift Container Platform バージョン

以下の表は、「生命の終わり」に到達した IBM MQ Operator のバージョンに適用されます。

オペレーター・チャネル	Operator バージョン	Red Hat OpenShift Container Platform のバージョン [3]						
		4.4[4]	4.5[5]	4.6	4.7[6]	4.8	4.9	4.10
v1.0	1.0	⚠	⚠	⚠	⚠			
v1.1	1.1	⚠	⚠	⚠	⚠	⚠		
v1.2	1.2	⚠	⚠	⚠	⚠	⚠		
v1.3-eus	1.3			⚠	→	→	→	→
v1.4	1.4			⚠	⚠	⚠	⚠	
v1.5	1.5			⚠	⚠	⚠	⚠	⚠

キー

→
Extended Update 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。

⚠
IBM MQ Operator のバージョンは「生命の終わり」に達しましたが、このバージョンの Red Hat OpenShift Container Platform では以前に使用可能になっていました。

---

[3] Red Hat 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、 Red Hat OpenShift Container Platform ライフ・サイクル・ポリシー を参照してください。

[4] Red Hat OpenShift Container Platform 4.4 は「生命の終わり」に到達しました。 詳しくは、 Red Hat OpenShift Container Platform ライフ・サイクル・ポリシー を参照してください。

[5] Red Hat OpenShift Container Platform 4.5 は「生命の終わり」に到達しました。 詳しくは、 Red Hat OpenShift Container Platform ライフ・サイクル・ポリシー を参照してください。

[6] IBM MQ Operator は、 IBM Cloud Pak foundational services によって異なります。 Red Hat OpenShift Container Platform 4.7 を使用したい場合は、まず IBM Cloud Pak foundational services のバージョンをアップグレードする必要があります。

OpenShift CP4I IBM MQ Operator の依存関係

IBM MQ Operator は IBM Cloud Pak foundational services オペレーターに依存し、IBMODLM（Operand Deployment Lifecycle Manager）オペレーターもインストールされます。 これらの演算子は、IBM MQ Operator のインストール時に自動的にインストールされます。 これらの依存オペレーターは、CPU とメモリのフットプリントが小さく、状況によっては追加リソースを配置するために使用されます。

QueueManager を作成すると、IBM MQ Operator は、必要な追加サービス用に OperandRequest を作成します。 OperandRequest は ODLM オペレーターによって実行され、必要に応じて必要なサービスをインストールしてインスタンス化します。 どのサービスが必要かは、キュー・マネージャーの導入時に受け入れたライセンス契約と、どのキュー・マネージャー・コンポーネントが要求されるかに基づいて決定されます。

- 「IBM MQ Advanced」または「IBM MQ Advanced for Developers」ライセンスを選択した場合、追加のサービスは要求されません。 例えば、以下の場合、IBM Cloud Pak foundational services は使用されません。

  ```
  spec:
    license:
      accept: true
      license: L-APIG-BZDDDY
      use: "Production"
  ```

- IBM Cloud Pak for Integration ライセンスを選択し、 Web サーバーを使用可能にすることを選択した場合、IBM MQ Operator は、IBMID およびアクセス管理 (IAM) オペレーターのインスタンスを生成して、シングル・サインオンを可能にします。 IBM Cloud Pak for Integration Operator がインストール済みの場合、IAM Operator はすでに使用可能です。 以下に例を示します。

  ```
  spec:
    license:
      accept: true
      license: L-RJON-BUVMQX
      use: "Production"
  ```

  ただし、ウェブサーバーを使用不可に設定すると、IBM Cloud Pak foundational services は要求されません。 以下に例を示します。

  ```
  spec:
    license:
      accept: true
      license: L-RJON-BUVMQX
      use: "Production"
    web:
      enabled: false
  ```

古いバージョンの IBM MQ Operator では、ライセンスの使用を追跡するために、 IBM のライセンス・オペレーター (およびその依存関係) のインストールを常に要求しています。 IBM MQ Operator1.5 以降では、ライセンス・サービスは要求されず、別々に要求する必要があります。

IBM MQ Operator には、1 つの CPU コアと 1 GB のメモリーが必要です。 従属オペレーターのハードウェア要件およびソフトウェア要件の詳細については、 ファウンデーションサービスのためのハードウェア要件と推奨事項を参照してください。

キュー・マネージャーが使用する CPU とメモリーの量を選択できます。 詳しくは、 138 ページの『.spec.queueManager.resources』 を参照してください。

関連資料

128 ページの『mq.ibm.com/v1beta1 のライセンスのリファレンス』

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

IBM MQ Operator には、アドミッション Webhook やサンプルを管理したり、ストレージ・クラスやクラスター・バージョンの情報を読み取ったりするためのクラスター・スコープの権限が必要です。

IBM MQ Operator は、以下のクラスター・スコープ許可を必要とします。

- アドミッション Webhook を管理する権限。 オペレーターが提供するコンテナーの作成と管理のプロセスで使用される特定の Webhook の作成、取得、更新が可能になります。
  - API グループ: **admissionregistration.k8s.io**
  - リソース: **validatingwebhookconfigurations**
  - verbs: **create, get, update**
- カスタム・リソースの作成時にサンプルおよびスニペットを提供するために、Red Hat OpenShift コンソールで使用されるリソースを作成および管理する権限です。
  - API グループ: **console.openshift.io**
  - リソース: **consoleyamlsamples**
  - verbs: **create, get, update, delete**
- 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。 オペレーターがコンテナー内の選択したストレージ・クラスに関する問題をフィードバックすることが可能になります。
  - API グループ: **storage.k8s.io**
  - リソース: **storageclasses**
  - verbs: **get, list**

IBM MQ Operator のストレージに関する考慮事項

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、次の事柄が含まれます。

- 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、114 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、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、通常は永続ストレージを使用して実行されます。 したがって、これがデフォルトの動作になります。 さまざまなストレージ・プロバイダーがあり、プロバイダーごとに異なる機能がサポートされるので、多くの場合は、構成のカスタマイズが必要であるということになります。 v1beta1 API で MQ のストレージ構成をカスタマイズするためによく使用されるフィールドについて、以下の例にまとめます。

- spec.queueManager.availability は、可用性モードを制御します。 SingleInstance を使用する場合、必要なストレージは ReadWriteOnce ストレージだけです。一方、multiInstance の場合は、ファイル・ロックのための適切な特性を備えた、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、16 ページの『コンテナー内の IBM MQ の高可用性』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、永続ボリュームを 1 つから 4 つまで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: [99]
```

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。

注 : ネイティブ HA を使用する場合は、共有ファイル・システムは必要ありません (16 ページの『コンテナー内の IBM MQ の高可用性』を参照)。 特に、 NFSv3 は使用しないでください。

Linux 独自の IBM MQ キュー・マネージャー・コンテナー・イメージを作成するためのサポート

IBM MQ は、 GitHub 上に IBM MQ キュー・マネージャー・コンテナーを作成するためのコードを提供します。 これは、 IBM が独自のサポート・コンテナーを構築するために使用するプロセスに基づいており、この GitHub リポジトリーを使用して、独自のコンテナー・イメージの作成を単純化および加速することができます。

このコードは、mq-container GitHub リポジトリー内に、https://github.com/ibm-messaging/mq-container というコードで提供されています。 これは、Apache2.0 ライセンスの下で提供されます。サポートはコミュニティーによって提供されます。

リポジトリーは、標準の Linuxrpm パッケージを使用しません。コンテナー・デプロイメントには圧縮パッケージが使用されます。 この方法の利点は、強化されたアクセス権を必要とせずに、よりセキュアなコンテナー環境で実行できることです。 ただし、これは、使用可能なセキュリティー・オプションに影響を与えます。なぜなら、IBM MQ はオペレーティング・システム・ベースの認証にエスカレートされた許可 コンテナー・デプロイメントの場合、オペレーティング・システム・ベースの認証を使用するのは通常は適していません。代わりに、相互 TLS 認証または LDAP 認証を使用できます。 IBM MQ Advanced for Developers では、ファイル・ベースの認証を使用することもできます。これにより、ユーザーはすぐに開始できます。

複製データ・キュー・マネージャー (RDQM) は、コンテナー環境内ではサポートされません。 93 ページの『ネイティブ HA』を使用して RDQM に類似の機能を入手することができます。

関連概念

6 ページの『IBM MQ Operator のサポート』
IBM MQ Operator は、Red Hat OpenShift Container Platform 上でデプロイされる場合に限りサポートされます。

IBM MQ 非インストール・イメージ

Linux 独自の IBM MQ コンテナー・イメージを作成する時のライセンス・アノテーション

ライセンス・アノテーションを使用すると、基礎になっているマシンではなくコンテナーで定義した制限に基づいて使用量を追跡管理できます。 クライアントで特定のアノテーションを付けてコンテナーをデプロイするための構成を行うと、IBM License Service はそのアノテーションに基づいて使用量を追跡管理します。

独自に作成した IBM MQ コンテナー・イメージをデプロイする場合は、ライセンス交付に関して以下の 2 つの一般的なアプローチがあります。

- 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いてコンテナーのライセンスを取得します。

どちらのオプションもお客様が使用できます。詳細については、 Passport Advantage®の「 IBM Container Licenses 」ページを参照してください。

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。 お持ちの使用権と、コンテナー内にデプロイされている機能に基づいて、以下のアノテーションを 1 つ以上使用してください。

- 14 ページの『IBM MQ Advanced コンテナー』
- 14 ページの『IBM MQ Advanced High Availability Replica コンテナー』
- 14 ページの『IBM MQ Base コンテナー』

IBM MQ Advanced コンテナー

```
productName: "IBM MQ Advanced"
productID: "208423bb063c43288328b1d788745b0c"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "PROCESSOR_VALUE_UNIT" |  "VIRTUAL_PROCESSOR_CORE"
```

IBM MQ Advanced High Availability Replica コンテナー

```
productName: "IBM MQ Advanced High Availability Replica"
productID: "546cb719714942c18748137ddd8d5659"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "PROCESSOR_VALUE_UNIT" |  "VIRTUAL_PROCESSOR_CORE"
```

IBM MQ Base コンテナー

```
productName: "IBM MQ"
productID: "c661609261d5471fb4ff8970a36bccea"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "PROCESSOR_VALUE_UNIT" |  "VIRTUAL_PROCESSOR_CORE"
```

IBM MQ Base High Availability Replica コンテナー

```
productName: "IBM MQ High Availability Replica"
productID: "2a2a8e0511c849969d2f286670ea125e"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "PROCESSOR_VALUE_UNIT" |  "VIRTUAL_PROCESSOR_CORE"
```

IBM MQ Advanced for Developers コンテナー

```
productName: "IBM MQ Advanced for Developers"
productID: "2f886a3eefbe4ccb89b2adb97c78b9cb"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "FREE"
```

IBM MQ Advanced コンテナー (CP4I 使用権付き) (実動)

```
productName: "IBM MQ Advanced with CP4I License"
productID: "208423bb063c43288328b1d788745b0c"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "2:1"
```

```
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ Advanced High Availability Replica コンテナー (CP4I 使用権付き) (実動)

```
productName: "IBM MQ Advanced High Availability Replica with CP4I License"
productID: "546cb719714942c18748137ddd8d5659"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "10:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ Advanced コンテナー (CP4I 使用権付き) (非実動)

```
productName: "IBM MQ Advanced for Non-Production with CP4I License"
productID: "21dfe9a0f00f444f888756d835334909"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "4:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ Advanced High Availability Replica コンテナー (CP4I 使用権付き) (非実動)

```
productName: "IBM MQ Advanced High Availability Replica for Non-Production with CP4I License"
productID: "b3f8f984007d47fb981221589cc50081"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "20:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ Base (CP4I 使用権付き) (実動)

```
productName: "IBM MQ with CP4I License"
productID: "c661609261d5471fb4ff8970a36bccea"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "4:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ Base High Availability Replica (CP4I 使用権付き) (実動)

```
productName: "IBM MQ High Availability Replica with CP4I License"
productID: "2a2a8e0511c849969d2f286670ea125e"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "20:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ Base (CP4I 使用権付き) (非実動)

```
productName: "IBM MQ with CP4I License Non-Production"
productID: "151bec68564a4a47a14e6fa99266deff"
productChargedContainers:  "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "8:1"
```

```
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

IBM MQ Base High Availability Replica (CP4I 使用権付き) (非実動)

```
productName: "IBM MQ High Availability Replica with CP4I License Non-Production"
productID: "f5d0e21c013c4d4b8b9b2ce701f31928"
productChargedContainers: "All" | "NAME_OF_CONTAINER"
productMetric: "VIRTUAL_PROCESSOR_CORE"
productCloudpakRatio: "40:1"
cloudpakName: "IBM Cloud Pak for Integration"
cloudpakId: "c8b82d189e7545f0892db9ef2731b90d"
```

OpenShift CP4I Kubernetes コンテナー内の IBM MQ の高可用性

IBM MQ Operator で高可用性を使用する場合、次の 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** (1 つのアクティブ・レプリカと 2 つのスタンバイ・レプリカを含む)、**複数インスタンス・キュー・マネージャー** (ネットワーク接続した共有ファイル・システムを使用するアクティブとスタンバイのペア)、または**シングル・レジリエント・キュー・マネージャー** (ネットワーク接続したストレージを使用して HA のシンプルなアプローチを提供する)。 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

CP4I V 9.2.3

ネイティブ HA キュー・マネージャーは、IBM Cloud Pak for Integration 2021.2.1 以降、IBM MQ Operator 1.6 以上、IBM MQ 9.2.3 以上を使用することで使用可能になります。

ネイティブ HA キュー・マネージャーでは、1 つの**アクティブ**と 2 つの**レプリカ**の Kubernetes ポッドを使用します。それらのポッドを Kubernetes StatefulSet の一部として実行し、その 3 つのレプリカがそれぞれ独自のセットの Kubernetes Persistent Volume を使用します。 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。

詳しくは、93 ページの『ネイティブ HA』 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

Multi

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、 **access mode** を `ReadWriteMany` に設定した Kubernetes 永続ボリュームを使用します。 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

Multi

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 のようなステートフル・ワークロードを Kubernetes StatefulSet で使用する場合、 Kubernetes マスター・ノードは、ワーカー・ノードとの接続を失うと、ノードに障害が発生したかどうか、または単にネットワーク接続が失われたかどうかを判別できません。 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。

関連タスク

93 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』

関連資料

高可用性の構成

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

コンテナー内の IBM MQ でのユーザーの認証と許可

LDAP ユーザーとグループを使用するよう IBM MQ を構成できます。 あるいは、コンテナー・イメージ内のローカル・オペレーティング・システムのユーザーおよびグループを使用することもできます。 IBM MQ Operator では、セキュリティー上の問題により、オペレーティング・システムのユーザーとグループをユーザーとして使用することは許可されていません。

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。これを行うために、「setuid」プログラムを使用して「root」ユーザーになります。

OpenShift CP4I これらのセキュリティー対策を確実に遵守するために、IBM MQ Operator では、コンテナー内のオペレーティング・システムのライブラリーで定義されている ID の使用は許可されていません。 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。 IBM Cloud Pak for Integration と Red Hat OpenShift で IBM MQ を使用する場合、ユーザー認証と認可に LDAP を使用するようキュー・マネージャーを設定する必要があります。 そのための IBM MQ の構成方法については、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。

Multi 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 では、コンテナー当たり 4096 プロセス (OpenShift 4.11 までは 1024 プロセス) というデフォルトの制限があります。 これは大部分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る場合があります。

  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。

IBM Cloud Pak for Integration と Red Hat OpenShift で IBM MQ を使用する

IBM MQ Operator は、IBM MQ を IBM Cloud Pak for Integration の一部としてデプロイして管理するか、Red Hat OpenShift Container Platform で単体としてデプロイして管理します。

手順

- 20 ページの『IBM MQ Operator のリリース履歴』.
- 36 ページの『IBM Cloud Pak for Integration への IBM MQ のマイグレーション』.
- 60 ページの『Red Hat OpenShift 上での IBM MQ Operator のインストールおよびアンインストール』.
- 72 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』.
- 80 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』.
- 117 ページの『IBM MQ Operator を使用した IBM MQ』.
- 128 ページの『IBM MQ Operator の API リファレンス』.

IBM MQ Operator のリリース履歴

IBM MQ Operator

IBM MQ Operator 1.8.2

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2021.4.1

オペレーター・チャネル
v1.8

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1, 9.2.4.0-r1, 9.2.5.0-r1, 9.2.5.0-r2, 9.2.5.0-r3

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 以上

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.8 以上 (v3 チャネル)

新機能

- IBM MQ Operator 1.8.0 に基づいて作成されたセキュリティーのみの更新。
- 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 1.8.1

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2021.4.1

オペレーター・チャネル
v1.8

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1, 9.2.4.0-r1, 9.2.5.0-r1, 9.2.5.0-r2

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 以上

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.8 以上 (v3 チャネル)

新機能

- IBM MQ Operator 1.8.0 に基づいて作成されたセキュリティーのみの更新。
- 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 1.8.0

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2021.4.1

オペレーター・チャネル
v1.8

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1, 9.2.4.0-r1, 9.2.5.0-r1

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 以上

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.8 以上 (v3 チャネル)

新機能

- 非推奨の IBM MQ バージョンについて状況条件が追加されます。

変更点

- イメージが Docker Hub から IBM Container Registry に移動されました。
  - ファイアウォール・ルールを使用しているお客様は、IBM Container Registry 上のイメージにアクセスするためにファイアウォール・ルールを調整しなければならない場合があります。
  - エアギャップのお客様に関しては、IBM MQ Operator 1.8.0 へのアップグレード時にノードが再始動します。
- 非推奨バージョン: IBM MQ 9.1.5、9.2.0 CD、9.2.1、9.2.2。 これらのバージョンは、IBM MQ Operator の将来のバージョンでは調整されない可能性があります。
- ライセンス・ロジックの変更: IBM MQ 9.2.5 にアップグレードするお客様は、IBM MQ 9.2.5 を操作するために指定されたライセンスのみを使用できます。 128 ページの『mq.ibm.com/v1beta1 のライセンスのリファレンス』を参照してください。
- 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 1.7.0

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2021.4.1

オペレーター・チャネル
v1.7

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1, 9.2.4.0-r1

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 以上

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.8 以上 (v3 チャネル)

新機能

- 継続的デリバリー・リリースとして IBM MQ 9.2.4 を追加する

IBM MQ Operator 1.6.0

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2021.2.1

オペレーター・チャネル
v1.6

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.2-r2-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1, 9.2.3.0-r1

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 以上

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.7 以上 (v3 チャネル)

新機能

- IBM MQ 9.2.3 が継続的デリバリー・リリースとして追加されます (amd64 (IBM Cloud Pak for Integration 2021.2.1 の場合のみ)、amd64 または s390x (IBM MQ ライセンスを使用する場合))
- キュー・マネージャーの新しい可用性のタイプ: ネイティブ HA。 IBM Cloud Pak for Integration 2021.2.1 の一部として、実動で使用できます。

変更点

- IBM MQ Operator 1.6 以降は、Docker Hub の代わりに IBM Container Registry を使用します。 つまり、CatalogSource を icr.io から使用する必要があります。 60 ページの『Red Hat OpenShift 上での IBM MQ Operator のインストールおよびアンインストール』を参照してください。
- ネイティブ HA のローリング更新では、レプリカが同期するのを待たずに、次のレプリカに移動するようになりました。
- OCP 4.7 以上でのネイティブ HA アフィニティーに関する問題を修正しました。
- CA 署名証明書をネイティブ HA で使用する場合の問題を修正しました。

IBM MQ Operator 1.5.0

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2021.1.1

オペレーター・チャネル
v1.5

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.0-r3, 9.2.0.1-r1-eus, 9.2.1.0-r1, 9.2.1.0-r2, 9.2.2.0-r1

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 以上

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.7 以上 (v3 チャネル)

新機能

- IBM MQ 9.2.2 が継続的デリバリー・リリースとして追加されます (amd64 (IBM Cloud Pak for Integration 2021.1.1 の場合のみ)、amd64 または s390x (IBM MQ ライセンスを使用する場合))
- キュー・マネージャーの新しい可用性のタイプ: ネイティブ HA。 IBM Cloud Pak for Integration 2021.1.1 の一部として、評価目的に限り使用できます。
- Red Hat OpenShift Container Platform Cluster Monitoring for Prometheus メトリックとの統合 (`ServiceMonitor` リソースを提供することによる)

変更点

- キュー・マネージャーの作成時に、デフォルトでは IBM Licensing Operator が作成されなくなりました。
- 複数インスタンス・キュー・マネージャーに対する更新が、ローリング順序で処理されるようになりました。 この変更の一部として、Liveness Probe の構成時に使用される値に影響する Kubernetes の Startup Probe が導入されています。 Startup Probe はすぐに開始され、キュー・マネージャーが正常に開始するまで待機します。 この待機期間内に Startup Probe が合格すると、Liveness Probe と Readiness Probe が開始します。 以前は、キュー・マネージャーの開始に時間がかかっていた場合に、活性プローブ上の `initialDelaySeconds` 設定を増やしていた可能性があります。 これを行った場合は、この時点で `initialDelaySeconds` を前の設定に戻す必要があります。
- `CustomResourceDefinition` が `apiextensions.k8s.io/v1beta1` から `apiextensions.k8s.io/v1` にアップグレードされました

既知の問題と制限事項

- IBM Cloud Pak foundational services 3.7 が必要です。これには、ID およびアクセス管理 (IAM) コンポーネントにおける互換性のない変更が含まれています。 IBM Cloud Pak for Integration ライセンスを使用するキュー・マネージャーがある場合は、このアップグレードの後、Web コンソールにアクセスするためにキュー・マネージャーの再始動が必要になります。また、Web コンソールにログインするときにその他のエラーが表示されます。 オペレーター・アップグレードが完了した後で、`.spec.version` の最新値 (選択した IBM MQ バージョン) にアップグレードすることで、このようなエラーを修正できます。
- MQ のバージョンをアップグレードしている場合、ローリング更新は自動的には開始しません。 ポッドを手動で削除する必要があります。

IBM MQ Operator 1.4.0

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.4.1 (IBM MQ Operator 1.4.0 は CD リリースで、拡張更新サポートの対象ではありません)

オペレーター・チャネル
v1.4

`.spec.version` に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.1.0-r1

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 以上

新機能

- IBM MQ 9.2.1 が継続的デリバリー・リリースとして追加されました。
- `.spec.queueManager.route.enabled` を `false` に設定することにより、デフォルト・キュー・マネージャー経路が作成されないようにできるようになりました

既知の問題と制限事項

- QueueManager を可用性タイプ MultiInstance で更新すると、両方のポッドが即時に削除されます。 それら両方とも Red Hat OpenShift Container Platform ですぐに再始動する必要があります。

IBM MQ Operator 1.3.8 (EUS)

EUS

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.4.1

オペレーター・チャネル
v1.3-eus

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus, 9.2.0.6-r1-eus, 9.2.0.6-r2-eus, 9.2.0.6-r3-eus

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 のみ

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.6 (stable-v1 チャネル)

新機能

- 新しいオペランド・バージョン 9.2.0.6-r3-eus が追加されました。
- 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 1.3.7 (EUS)

EUS

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.4.1

オペレーター・チャネル
v1.3-eus

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus, 9.2.0.6-r1-eus, 9.2.0.6-r2-eus

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 のみ

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.6 (stable-v1 チャネル)

新機能

- 新規オペランド・バージョン 9.2.0.6-r2-eus が追加されました。
- 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 1.3.6 (EUS)

EUS

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.4.1

オペレーター・チャネル
v1.3-eus

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus, 9.2.0.6-r1-eus

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 のみ

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.6 (`stable-v1` チャネル)

新機能

- 新しいオペランド・バージョン 9.2.0.6-r1-eus が追加されました。
- 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 1.3.5 (EUS)

EUS

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.4.1

オペレーター・チャネル
`v1.3-eus`

`.spec.version` に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus, 9.2.0.5-r3-eus

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 のみ

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.6 (`stable-v1` チャネル)

新機能

- 新しいオペランド・バージョン 9.2.0.5-r3-eus が追加されました。
- 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 1.3.4 (EUS)

EUS

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.4.1

オペレーター・チャネル
`v1.3-eus`

`.spec.version` に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus, 9.2.0.5-r2-eus

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 のみ

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.6 (`stable-v1` チャネル)

新機能

- 新しいオペランド・バージョン 9.2.0.5-r2-eus が追加されました。
- 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 1.3.3 (EUS)

EUS

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.4.1

オペレーター・チャネル
v1.3-eus

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus, 9.2.0.5-r1-eus

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 のみ

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.6 (stable-v1 チャネル)

新機能

- 新規オペランド・バージョン 9.2.0.5-r1-eus が追加されました。
- 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 1.3.2 (EUS)

EUS

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.4.1

オペレーター・チャネル
v1.3-eus

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus, 9.2.0.4-r1-eus

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 のみ

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.6 (stable-v1 チャネル)

新機能

- 新規オペランド・バージョン 9.2.0.4-r1-eus

IBM MQ Operator 1.3.1 (EUS)

EUS

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.4.1

オペレーター・チャネル
v1.3-eus

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus, 9.2.0.2-r1-eus

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 のみ

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.6 (stable-v1 チャネル)

新機能

- 新しいオペランド・バージョン 9.2.0.2-r1-eus を追加しました。

IBM MQ Operator 1.3.0 (EUS)

EUS

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.4.1

オペレーター・チャネル
v1.3-eus

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2, 9.2.0.1-r1-eus

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.6 のみ

IBM Cloud Pak foundational services のバージョン
IBM Cloud Pak foundational services 3.6 (stable-v1 チャネル)

新機能

- .spec.version フィールド (末尾 -eus のもの) について、IBM Cloud Pak for Integration ライセンスの使用時に、拡張更新サポート (EUS) が提供されます
- QueueManager リソース上で .spec.labels および .spec.annotations を使用してラベルとアノテーションを設定する新しい方法が追加されます

変更点

- 単一インスタンスから複数インスタンスに変更しようとする時のエラー処理が改善されました。
- QueueManager プロパティーが IBM Cloud Pak for Integration Platform Navigator においてレンダリングされる仕組みの改善、および"フォーム・ビュー" (Red Hat OpenShift Container Platform Web コンソール)
- IBM Cloud Pak for Integration ライセンスを使用する場合のデフォルト・ライセンス・メトリックが VirtualProcessorCore になるように修正されます
- 「**リソース**」タブが QueueManager について Red Hat OpenShift Container Platform Web コンソールで修正されます。このタブで、当該キュー・マネージャーに関して IBM MQ Operator によって管理されてるリソースが正しく表示されるようになりました

既知の問題と制限事項

- QueueManager を可用性タイプ MultiInstance で更新すると、両方のポッドが即時に削除されます。 それら両方とも Red Hat OpenShift Container Platform ですぐに再始動する必要があります。

IBM MQ Operator 1.2.0

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.3.1

オペレーター・チャネル
v1.2

.spec.version に許可される値
9.1.5.0-r2, 9.2.0.0-r1, 9.2.0.0-r2

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.4 以上

新機能

- z/Linux のサポートが追加されました。
- QueueManager リソースに詳細状況条件が追加されます。 詳しくは、147 ページの『QueueManager (mq.ibm.com/v1beta1) の状況状態』 を参照してください
- 無効なストレージ・クラスの使用を防止するための実行時チェックが追加されました。 詳しくは、116 ページの『実行時 Webhook チェックの無効化』 を参照してください。
- マルチインスタンス・キュー・マネージャーのエクスペリエンスが単純化される: これは、1 つのプロパティー (.spec.queueManager.availability.type) のみを使用して QueueManager リソースで選択できるようになりました

- `.spec.queueManager.storage.defaultClass` プロパティーを `QueueManager` リソースに導入することにより、非デフォルト・ストレージ・クラスの選択が単純化されます

変更点

- `QueueManager` プロパティーが IBM Cloud Pak for Integration Platform Navigator においてレンダリングされる仕組みの改善、および"フォーム・ビュー" (Red Hat OpenShift Container Platform Web コンソール)
- キュー・マネージャーのアップグレードされたバージョンを利用できる場合は、IBM Cloud Pak for Integration Platform Navigator でフラグが立てられます。

IBM MQ Operator 1.1.0

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.2.1

オペレーター・チャネル
`v1.1`

`.spec.version` に許可される値
9.1.5.0-r2, 9.2.0.0-r1

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.4 以上

新機能

- IBM MQ Advanced 9.2.0 が継続的デリバリー・リリースとして追加されました。
- ConfigMap またはシークレットで INI と MQSC の情報を指定するための機能が追加されました。
- Red Hat OpenShift Container Platform Web コンソールを使用する時にスキーマ・ナビゲーターが有効になります。

変更点

- ネットワーク・ポリシーに関する問題が修正され、 Red Hat OpenShift の IBM Cloud® に影響があります。
- `QueueManager` リソースにおける設定が無効に組み合わされないようにするための検証 Web フックの改善

IBM MQ Operator 1.0.0

CD

IBM Cloud Pak for Integration バージョン
IBM Cloud Pak for Integration 2020.2.1

オペレーター・チャネル
`v1.0`

`.spec.version` に許可される値
9.1.5.0-r2

Red Hat OpenShift Container Platform のバージョン
Red Hat OpenShift Container Platform 4.4 以上

新機能

- オペレーターの初期バージョン、 `mq.ibm.com/v1beta1` API の導入

IBM MQ Operator で使用するためのキュー・マネージャー・コンテナー・イメージ

9.2.5.0-r3

CD

必要なオペレーター・バージョン
1.8.2 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.5.0-r3
- cp.icr.io/cp/ibm-mqadvanced-server:9.2.5.0-r3
- icr.io/ibm-messaging/mq:9.2.5.0-r3

新機能

- IBM MQ 9.2.5 の新機能

変更点

- IBM MQ 9.2.5 の変更内容
- Red Hat Universal Base Image 8.6-751 をベースにしています。

9.2.5.0-r2

CD

必要なオペレーター・バージョン
1.8.1 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.5.0-r2
- cp.icr.io/cp/ibm-mqadvanced-server:9.2.5.0-r2
- icr.io/ibm-messaging/mq:9.2.5.0-r2

新機能

- IBM MQ 9.2.5 の新機能

変更点

- IBM MQ 9.2.5 の変更内容
- Red Hat Universal Base Image 8.5-240.1648458092 に基づいています。

9.2.5.0-r1

CD

必要なオペレーター・バージョン
1.8.0 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.5.0-r1
- cp.icr.io/cp/ibm-mqadvanced-server:9.2.5.0-r1
- icr.io/ibm-messaging/mq:9.2.5.0-r1

新機能

- IBM MQ 9.2.5 の新機能

変更点

- IBM MQ 9.2.5 の変更内容
- 無効なリモート・キュー・マネージャー・オプションが IBM MQ Console から削除されました
- Red Hat Universal Base Image 8.5-240 に基づく

9.2.4.0-r1

CD

必要なオペレーター・バージョン
1.7.0 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.4.0-r1`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.4.0-r1`
- `docker.io/ibmcom/mq:9.2.4.0-r1`

新機能

- IBM MQ 9.2.4 の新機能

変更点

- IBM MQ 9.2.4 の変更内容
- Red Hat ユニバーサル・ベース・イメージ 8.5-204 に基づいています

9.2.3.0-r1

CD

必要なオペレーター・バージョン
1.6.0 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.3.0-r1` (amd64 のみ)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.3.0-r1`
- `docker.io/ibmcom/mq:9.2.3.0-r1`

新機能

- IBM MQ 9.2.3 の新機能
- IBM Cloud Pak for Integration ライセンスで使用する場合の実動用の MQ ネイティブ HA のサポート。 IBM MQ 9.2.2 の評価ライセンスの下でネイティブ HA を使用するキュー・マネージャーは、9.2.3 にアップグレードすることはできません。 評価期間が終了しました。

変更点

- IBM MQ 9.2.3 の変更内容
- Red Hat Universal Base Image 8.4-205 がベースになりました。

9.2.2.0-r1

CD

必要なオペレーター・バージョン
1.5.0 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.2.0-r1` (amd64 のみ)
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.2.0-r1`
- `docker.io/ibmcom/mq:9.2.2.0-r1`

新機能

- IBM MQ 9.2.2 の新機能
- IBM Cloud Pak for Integration ライセンスで使用する場合の、評価目的の MQ ネイティブ HA のサポート

変更点

- IBM MQ 9.2.2 の変更内容
- IBM MQ Advanced for Developers キュー・マネージャーのシャットダウン時に FDC が発生する問題が修正されました
- Red Hat Universal Base Image 8.3-291 がベースになりました。

9.2.1.0-r2

CD

必要なオペレーター・バージョン
1.5.0 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.1.0-r2`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.1.0-r2`
- `docker.io/ibmcom/mq:9.2.1.0-r2`

変更点

- IBM Cloud Pak foundational services 3.7 以上のシングル・サインオンに関する問題を修正しました。
- Red Hat Universal Base Image 8.3-291 がベースになりました。

9.2.1.0-r1

CD

必要なオペレーター・バージョン
1.4.0 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.1.0-r1`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.1.0-r1`

- docker.io/ibmcom/mq:9.2.1.0-r1

新機能

- IBM MQ 9.2.1 の新機能
- MQ Web コンソールでデフォルト・ルートの接続情報を利用できるようになりました。

変更点

- IBM MQ 9.2.1 の変更内容
- Red Hat Universal Base Image 8.3-230 がベースになりました。

9.2.0.6-r3-eus

EUS

必要なオペレーター・バージョン
1.3.8 およびそれ以降のフィックスパック

サポートされているアーキテクチャー
amd64, s390x

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.6-r3-eus

変更点

- IBM MQ 9.2.0 Fix Pack 6 を組み込みます。 詳しくは、 Fix list for IBM MQ バージョン 9.2 LTS を参照してください。
- Red Hat Universal Base Image 8.6-941 に基づいています。

9.2.0.6-r2-eus

EUS

必要なオペレーター・バージョン
1.3.7 およびそれ以降のフィックスパック

サポートされているアーキテクチャー
amd64, s390x

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.6-r2-eus

変更点

- IBM MQ 9.2.0 Fix Pack 6 を組み込みます。 詳しくは、 Fix list for IBM MQ バージョン 9.2 LTS を参照してください。
- Red Hat Universal Base Image 8.6-902 に基づいています。

9.2.0.6-r1-eus

EUS

必要なオペレーター・バージョン
1.3.6 およびそれ以降のフィックスパック

サポートされているアーキテクチャー
amd64, s390x

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.6-r1-eus

変更点

- IBM MQ 9.2.0 Fix Pack 6 を組み込みます。 詳しくは、 Fix list for IBM MQ バージョン 9.2 LTS を参照してください。
- Red Hat Universal Base Image 8.6-854 に基づいています。

9.2.0.5-r3-eus

EUS

必要なオペレーター・バージョン
1.3.5 およびそれ以降のフィックスパック

サポートされているアーキテクチャー
amd64, s390x

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.5-r3-eus`

変更点

- IBM MQ 9.2.0 Fix Pack 5 を組み込みます。 詳しくは、 What 's changed in IBM MQ 9.2.0 Fix Pack 5 and Fix list for IBM MQ バージョン 9.2 LTS を参照してください。
- Red Hat Universal Base Image 8.6-751.1655117800 に基づいています。

9.2.0.5-r2-eus

EUS

必要なオペレーター・バージョン
1.3.4 およびそれ以降のフィックスパック

サポートされているアーキテクチャー
amd64, s390x

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.5-r2-eus`

変更点

- IBM MQ 9.2.0 Fix Pack 5 を組み込みます。 詳しくは、 IBM MQ 9.2.0 Fix Pack 5 の変更内容と IBM MQ バージョン 9.2 LTS の修正リスト を参照してください。
- Red Hat Universal Base Image 8.6-751 をベースにしています。

9.2.0.5-r1-eus

EUS

必要なオペレーター・バージョン
1.3.3 およびそれ以降のフィックスパック

サポートされているアーキテクチャー
amd64, s390x

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.5-r1-eus`

変更点

- IBM MQ 9.2.0 Fix Pack 5 を組み込みます。 詳しくは、 IBM MQ 9.2.0 Fix Pack 5 の変更内容と IBM MQ バージョン 9.2 LTS の修正リスト を参照してください。
- Red Hat Universal Base Image 8.5-240.1648458092 に基づいています。

9.2.0.4-r1-eus

EUS

必要なオペレーター・バージョン
1.3.2 および将来のフィックスパック

サポートされているアーキテクチャー
amd64, s390x

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.4-r1-eus

変更点

- IBM MQ 9.2.0 Fix Pack 4 を組み込みます。 詳しくは、 IBM MQ 9.2.0 Fix Pack 4 の変更内容と IBM MQ バージョン 9.2 LTS の修正リスト を参照してください。
- Red Hat ユニバーサル・ベース・イメージ 8.5-204 に基づいています

9.2.0.2-r2-eus

EUS

必要なオペレーター・バージョン
1.6.0 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.2-r2-eus

変更点

- IBM Cloud Pak foundational services 3.7 以上のシングル・サインオンに関する問題を修正しました。これは、EUS リリースから CD リリースにマイグレーションする場合にのみ必要です。
- Red Hat ユニバーサル・ベース・イメージ 8.4-200.1622548483 に基づいています

9.2.0.2-r1-eus

EUS

必要なオペレーター・バージョン
1.3.1 および将来のフィックスパック ; 1.6.0 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.2-r1-eus

変更点

- Operations Dashboard 統合では、トレース・エージェントおよびコレクター・バージョン 1.0.8 を使用します。
- IBM MQ 9.2.0 Fix Pack 2 を組み込みます。 詳しくは、 IBM MQ 9.2.0 Fix Pack 2 の変更内容と IBM MQ バージョン 9.2 LTS の修正リスト を参照してください。
- Red Hat ユニバーサル・ベース・イメージ 8.4-200.1622548483 に基づいています

9.2.0.1-r1-eus

EUS

必要なオペレーター・バージョン
1.3.0 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.1-r1-eus

新機能

- IBM Cloud Pak for Integration ライセンスを使用する場合にのみ利用できます。
- 拡張更新サポート (EUS) は、 Red Hat OpenShift Container Platform 4.6 で IBM MQ Operator 1.3.x および IBM Common Services 3.6 を使用している場合に使用できます。

変更点

- IBM MQ 9.2.0 Fix Pack 1 を組み込みます。 詳しくは、 IBM MQ 9.2.0 Fix Pack 1 の変更内容と IBM MQ バージョン 9.2 LTS の修正リスト を参照してください。
- Red Hat Universal Base Image 8.3-201 がベースになりました。
- 活性プローブ (chkmqhealthy) と作動可能プローブ (chkmqready) に関する問題 (特権のエスカレーションを許可する SecurityContextContraints の下での実行時) が修正されました。

9.2.0.0-r3

CD

必要なオペレーター・バージョン
1.5.0 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.0-r3
- cp.icr.io/cp/ibm-mqadvanced-server:9.2.0.0-r3
- docker.io/ibmcom/mq:9.2.0.0-r3

変更点

- Red Hat Universal Base Image 8.3-291 がベースになりました。

9.2.0.0-r2

CD

必要なオペレーター・バージョン
1.2.0 以上

サポートされているアーキテクチャー
amd64, s390x

イメージ

- cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.0-r2
- cp.icr.io/cp/ibm-mqadvanced-server:9.2.0.0-r2
- docker.io/ibmcom/mq:9.2.0.0-r2

新機能

- z/Linux で利用できるようになりました。

変更点

- Red Hat Universal Base Image 8.2-349 がベースになりました。

9.2.0.0-r1

CD

必要なオペレーター・バージョン
1.1.0 以上

サポートされているアーキテクチャー
amd64

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.2.0.0-r1-amd64`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.2.0.0-r1-amd64`
- `docker.io/ibmcom/mq:9.2.0.0-r1`

新機能

- IBM MQ 9.2.0 の新機能

変更点

- IBM MQ 9.2.0 の変更内容
- MQSC ファイルを自動的に適用するために `-ic` 引数 (`crtmqm` に対するもの) が使用されます。 以前の `runmqsc` コマンドの使用が置き換えられます。
- Red Hat Universal Base Image 8.2-301.1593113563 がベースになりました。

9.1.5.0-r2

CD

必要なオペレーター・バージョン
1.0.0 以上

サポートされているアーキテクチャー
amd64

イメージ

- `cp.icr.io/cp/ibm-mqadvanced-server-integration:9.1.5.0-r2-amd64`
- `cp.icr.io/cp/ibm-mqadvanced-server:9.1.5.0-r2-amd64`
- `docker.io/ibmcom/mq:9.1.5.0-r2`

変更点

- Red Hat Universal Base Image 8.2-267 がベースになりました。

OpenShift V 9.2.1 CD EUS

IBM Cloud Pak for Integration への IBM MQ のマイグレーション

この一連のトピックでは、IBM Cloud Pak for Integration 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. 37 ページの『必要な機能を利用できることの確認』
2. 38 ページの『キュー・マネージャー構成の抽出』
3. オプション: 39 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: 41 ページの『オプション: LDAP の構成』
5. オプション: 48 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. 50 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. 52 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. 53 ページの『キュー・マネージャー用のリソースの作成』
9. 54 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. 58 ページの『新規コンテナー・デプロイメントの検証』

OpenShift V 9.2.1 CD EUS 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、36 ページの『IBM Cloud Pak for Integration への IBM MQ 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、5 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、114 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。

   16 ページの『コンテナー内の IBM MQ の高可用性』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 V 9.2.1 CD EUS 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 V 9.2.1 CD EUS

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

TLS を使用してキュー・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 を使用しているかどうかの検証、鍵と証明書の抽出、マイグレーション済みのキュー・マネージャーでの TLS の構成を行います。

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。 この暗号化には、キュー・マネージャー上で構成された鍵リポジトリーの使用が欠かせません。 IBM MQ チャネルはそれを使って TLS 通信を有効にします。 鍵リポジトリーがご使用の環境で構成済みであるか分からない場合は、次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。

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 runmqckmResult < <(runmqckm -cert -list -db ${keyrlocation}.kdb -stashed)
             cert=1
         for i in "${runmqckmResult[@]:1}"
             do
                certlabel=$(echo ${i} | xargs)
                echo Extracting certificate $certlabel to $cert.cert
            runmqckm -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. TLS ストアの場所を特定します。

  ```
  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。

  ```
  runmqckm -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。

  ```
  runmqckm -cert -extract -db KEYSTORE_LOCATION -label "LABEL_NAME" -target OUTPUT_FILE
  -stashed
  ```

  直前に示した例は、次のコマンドに相当します。

  ```
  runmqckm -cert -extract -db /run/runmqserver/tls/key.kdb -label "CN=cs-ca-
  certificate,O=cert-manager" -target /tmp/cert-manager.crt -stashed
  runmqckm -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。

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 V 9.2.1 CD EUS

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、41 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。

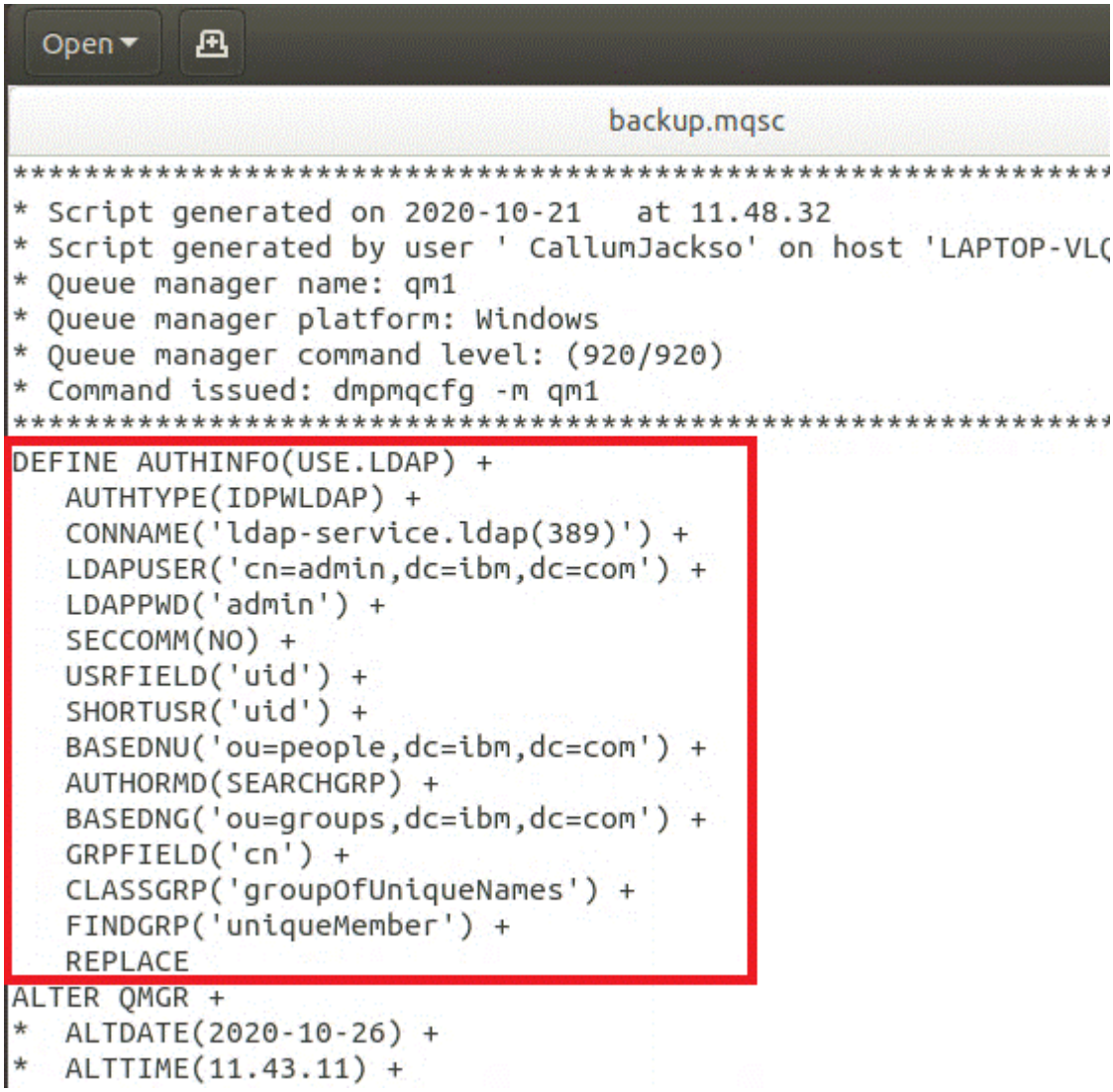
```
DEFINE AUTHINFO(USE.LDAP) +
   AUTHTYPE(IDPWLDAP) +
   CONNAME('ldap-service.ldap(389)') +
   LDAPUSER('cn=admin,dc=ibm,dc=com') +
   LDAPPWD('admin') +
   SECCOMM(NO) +
   USRFIELD('uid') +
   SHORTUSR('uid') +
   BASEDNU('ou=people,dc=ibm,dc=com') +
   AUTHORMD(SEARCHGRP) +
   BASEDNG('ou=groups,dc=ibm,dc=com') +
   GRPFIELD('cn') +
   CLASSGRP('groupOfUniqueNames') +
   FINDGRP('uniqueMember')
   REPLACE
```

   この

   - **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。 考えられる値：
     - 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 V 9.2.1 CD EUS

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、41 ページの『オプション: LDAP の構成』を参照してください。

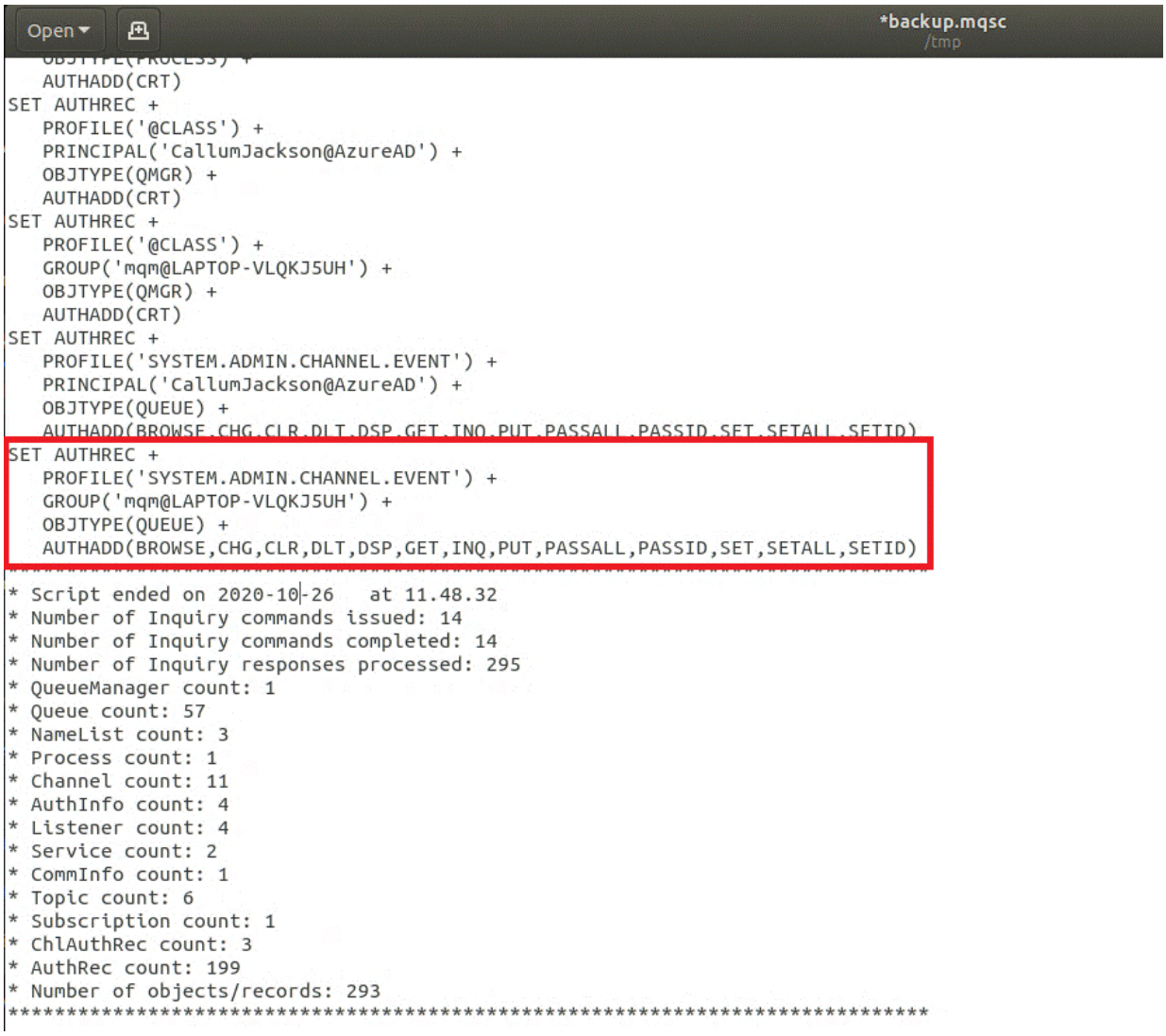
LDAP 許可情報の更新には 2 つの部分があります。

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 V 9.2.1 CD EUS 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。

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 V 9.2.1 CD EUS 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) と複数インスタンス (2 つのポッド) 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 2 つの高可用性オプションが用意されています。

- **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。

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 (54 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. 2 つのオプションを検討します。

   これらの 2 つのオプションの総合的な説明については、16 ページの『コンテナー内の IBM MQ の高可用性』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 V 9.2.1 CD EUS 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。

39 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。

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

52 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』 で概略を説明しているように、候補となるデプロイメント・トポロジーは 2 つあります。 このため、このトピックでは、次の 2 つの異なるテンプレートが用意されています。

- 単一インスタンス・キュー・マネージャーをデプロイする。
- 複数インスタンス・キュー・マネージャーをデプロイする。

重要 : ご使用のトポロジーに基づいて、2 つのテンプレートのうちの 1 つのみを実行してください。

手順

- 単一インスタンス・キュー・マネージャーをデプロイする。

  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。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm1
spec:
  version: 9.2.5.0-r3
  license:
    accept: true
    license: L-RJON-C7QG3S
    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: 9.2.5.0-r3
  license:
    accept: true
    license: L-RJON-C7QG3S
    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、128 ページの『mq.ibm.com/v1beta1 のライセンスのリファレンス』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。

- 複数インスタンス・キュー・マネージャーをデプロイする。

  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。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: qm1mi
spec:
  version: 9.2.5.0-r3
  license:
    accept: true
    license: L-RJON-C7QG3S
    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。

OpenShift V 9.2.1 CD EUS 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、54 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

36 ページの『IBM Cloud Pak for Integration への IBM MQ 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。 110 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。

OpenShift CP4I Red Hat OpenShift 上での IBM MQ Operator のインストールおよびアンインストール

IBM MQ Operator は、オペレーター・ハブを使用して Red Hat OpenShift にインストールすることができます。

手順

- 10 ページの『IBM MQ Operator の依存関係』.
- 10 ページの『IBM MQ Operator に必要なクラスター・スコープ許可』.
- 60 ページの『Red Hat OpenShift ウェブコンソールを使用して IBM MQ Operator をインストールする』.
- 62 ページの『Red Hat OpenShift CLI を使用した IBM MQ Operator のインストール』.
- 66 ページの『エアギャップ環境への IBM MQ Operator のインストール』.

関連タスク

62 ページの『Red Hat OpenShift ウェブコンソールを使用して IBM MQ Operator をアンインストールする』
Red Hat OpenShift から IBM MQ Operator をアンインストールするには、Red Hat OpenShift の Web コンソールを使用します。

65 ページの『Red Hat OpenShift CLI を使用した IBM MQ Operator のアンインストール』
Red Hat OpenShift から IBM MQ Operator をアンインストールするには、Red Hat OpenShiftCLI を使用します。 IBM MQ Operator が 1 つの名前空間にインストールされているか、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かによって、アンインストール・プロセスは異なります。

OpenShift CP4I Red Hat OpenShift ウェブコンソールを使用して IBM MQ Operator をインストールする

IBM MQ Operator は、オペレーター・ハブを使用して Red Hat OpenShift にインストールすることができます。

始める前に

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。

手順

1. EUS
   オプション: インストール可能なオペレーターのリストに IBM Common Services オペレーターを追加します。

   注:

   このステップは、IBM MQ Operator 1.5 以前のリリースに適用されます。 このステップでは、別個の共通サービス・カタログを追加します。 Operator のより新しいリリースでは、共通サービスは IBM カタログに含まれています。

   a) 画面の右上にあるプラス・アイコンをクリックします。 「**Import YAML**」ダイアログ・ボックスが表示されます。

   b) 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: opencloud-operators
  namespace: openshift-marketplace
spec:
  displayName: IBMCS Operators
  publisher: IBM
```

```
  sourceType: grpc
  image: icr.io/cpopen/ibm-common-service-catalog:latest
  updateStrategy:
    registryPoll:
      interval: 45m
```

c)「**作成**」 をクリックします。

2. インストール可能なオペレーターのリストに IBM オペレーターを追加します。
   a) 画面の右上にあるプラス・アイコンをクリックします。 「**Import YAML**」ダイアログ・ボックスが表示されます。
   b) 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
```

   c)「**作成**」 をクリックします。
3. 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。 この手順は、まだ存在してい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。
   a) ナビゲーション・ペインで、「**ホーム**」 > 「**プロジェクト (Projects)**」をクリックします。

   「Projects」ページが表示されます。
   b)「**Create Project**」をクリックします。 「Create Project」エリアが表示されます。
   c) 作成する名前空間の詳細を入力します。 例えば、「ibm-mq」などの名前を指定できます。
   d)「**作成**」 をクリックします。 IBM MQ Operator 用の名前空間が作成されます。
4. IBM MQ Operator のインストール
   a) 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。
   b)「**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。
   c)「**IBM MQ**」を選択します。

   「IBM MQ」ウィンドウが表示されます。
   d)「**インストール**」をクリックします。

   「Create Operator Subscription」ページが表示されます。
   e) 7 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。
   f) インストール・モードを、作成した特定の名前空間、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。
   g)「**サブスクライブ**」をクリックします。

   「インストール済みのオペレーター (Installed Operators)」ページに IBM MQ が表示されます。
   h)「インストール済みのオペレーター (Installed Operators)」ページでオペレーターの状況を確認します。インストールが完了すると状況が「Succeeded」に変わります。

次のタスク

81 ページの『Red Hat OpenShift ウェブコンソールを使用した IBM MQ 用の Red Hat OpenShift プロジェクトの準備』

OpenShift CP4I *Red Hat OpenShift ウェブコンソールを使用して IBM MQ Operator をアンインストールする*

Red Hat OpenShift から IBM MQ Operator をアンインストールするには、Red Hat OpenShift の Web コンソールを使用します。

始める前に

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。

IBM MQ Operator がクラスターのすべてのプロジェクト/名前空間に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、以下のステップ 1 から 5 の手順を繰り返します。

手順

1. 「**オペレーター (Operators)**」 > 「**インストール済みのオペレーター (Installed Operators)**」を選択します。
2. 「**プロジェクト**」ドロップダウン・リストから、プロジェクトを選択します。
3. 「**IBM MQ**」オペレーターをクリックします。
4. 「**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。
5. 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。
6. 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 1 から 5 を繰り返します。
7. 「**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。
8. 「**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。
9. Red Hat OpenShift Container Platform4.7 を使用している場合は、コマンド行から検証ウェブフックを手動で削除する必要がある場合があります。

   ```
   oc delete validatingwebhookconfiguration namespace.validator.queuemanagers.mq.ibm.com
   ```

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、オペレーター・ハブを使用して Red Hat OpenShift にインストールすることができます。

始める前に

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。 この手順は、クラスター管理者が行う必要があります。

手順

1. EUS

   オプション: **CatalogSource** を IBM Common Services オペレーター用に作成します。

   注:

このステップは、IBM MQ Operator 1.5 以前のリリースに適用されます。 このステップでは、別個の共通サービス・カタログを追加します。 Operator のより新しいリリースでは、共通サービスは IBM カタログに含まれています。

a) **CatalogSource** リソースを定義する YAML ファイルを作成します。

以下を内容とする「operator-source-cs.yaml」というファイルを作成します。

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

b) **CatalogSource** をサーバーに適用します。

```
oc apply -f operator-source-cs.yaml -n openshift-marketplace
```

2. **CatalogSource** を IBM オペレーター用に作成します

a) **CatalogSource** リソースを定義する YAML ファイルを作成します

以下を内容とする「operator-source-ibm.yaml」というファイルを作成します。

```
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
```

b) **CatalogSource** をサーバーに適用します。

```
oc apply -f operator-source-ibm.yaml -n openshift-marketplace
```

3. 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。 この手順は、まだ存在してい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。

```
oc new-project ibm-mq
```

4. OperatorHub で、クラスターに使用できるオペレーターのリストを表示します。

```
oc get packagemanifests -n openshift-marketplace
```

5. IBM MQ Operator でサポートされるインストール・モード (InstallMode) と使用可能なチャネル (Channel) を調べます。

```
oc describe packagemanifests ibm-mq -n openshift-marketplace
```

6. **OperatorGroup** オブジェクト YAML ファイルを作成します

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。 インストール対象となる予定のオペレーターが AllNamespaces を使用する場合、openshift-operators 名前空間には適切な **OperatorGroup** があらかじめ配置されています。

ただし、オペレーターが SingleNamespace モードを使用する場合に、適切な **OperatorGroup** がまだ準備されていないのであれば、それを作成する必要があります。

a) 以下を内容とする「mq-operator-group.yaml」というファイルを作成します。

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: <operatorgroup_name>
  namespace: <namespace_name>
spec:
  targetNamespaces:
  - <namespace_name>
```

b) **OperatorGroup** オブジェクトを作成します

```
oc apply -f mq-operator-group.yaml
```

7. **Subscription** オブジェクト YAML ファイルを、IBM MQ Operator に名前空間をサブスクライブするために作成します

a) 7 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) 以下の内容を含むファイル「mq-sub. yaml」を作成します。ただし、**channel** は、インストールする IBM MQ Operator のバージョンのチャネルに適合するように変更します。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ibm-mq
  namespace: openshift-operators
spec:
  channel: <ibm-mq-operator-channel>
  name: ibm-mq
  source: ibm-operator-catalog
  sourceNamespace: openshift-marketplace
```

AllNamespaces **InstallMode** を使用する場合は、名前空間に **openshift-operators** を指定します。 それ以外の場合は、SingleNamespace **InstallMode** を使用するための関連単一名前空間を指定します。 **namespace** フィールドのみを変更し、 **sourceNamespace** フィールドはそのままにしておくことに注意してください。

c) **Subscription** オブジェクトを作成します

```
oc apply -f mq-sub.yaml
```

8. オペレーターの状況を確認します

オペレーターが正常にインストールされると、ポッド状況は「実行中」と表示されます。
AllNamespaces **InstallMode** を使用する場合は、名前空間として **openshift-operators** を指定します。 それ以外の場合は、SingleNamespace **InstallMode** を使用するための関連単一名前空間を指定します。

```
oc get pods -n <namespace_name>
```

次のタスク

82 ページの『Red Hat OpenShift CLI を使用した IBM MQ のための Red Hat OpenShift プロジェクトの準備』

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

Red Hat OpenShift から IBM MQ Operator をアンインストールするには、Red Hat OpenShiftCLI を使用します。 IBM MQ Operator が 1 つの名前空間にインストールされているか、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かによって、アンインストール・プロセスは異なります。

始める前に

`oc login` を使用して Red Hat OpenShift クラスターにログインします。

手順

- 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) 正しいプロジェクトに入って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) オプション: 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。

  a. IBM Cloud Pak foundational services 製品資料の「共通サービスのアンインストール」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。

  b. オペレーター・グループを表示します。

  ```
  oc get operatorgroup
  ```

  c. オペレーター・グループを削除します。

  ```
  oc delete OperatorGroup <operator_group_name>
  ```

- 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) Red Hat OpenShift Container Platform 4.7 を使用している場合は、検証ウェブフックを手動で削除する必要がある場合があります。

```
oc delete validatingwebhookconfiguration namespace.validator.queuemanagers.mq.ibm.com
```

h) オプション: 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できます。

IBM Cloud Pak foundational services 製品資料の「共通サービスのアンインストール」の説明に従ってください。

OpenShift CP4I Linux エアギャップ環境への IBM MQ Operator のインストール

このチュートリアルは、インターネットに接続されていない Red Hat OpenShift クラスタに IBM MQ Operator をインストールする方法を説明します。 ポータブル・ストレージ・デバイスを使用するか、踏み台マシンを使用して、エアギャップ環境内で IBM MQ Operator をインストールすることができます。

ポータブル・ストレージ・デバイスを使用したエアギャップ環境での IBM MQ Operator のインストール

インストールを完了する手順については、IBM Cloud Pak for Integration 資料の ポータブル・ストレージ・デバイスを使用したミラーリング・イメージ を参照してください。 IBM MQ のみをインストールする場合は、以下の環境変数のすべてのオカレンスを、以下に示す値に置き換えます。

```
export CASE_NAME=ibm-mq
export CASE_ARCHIVE_VERSION=version_number
export CASE_INVENTORY_SETUP=ibmMQOperator
```

ここで、*version_number* は、エアギャップ・インストールのために使用するケースのバージョンです。 使用可能なケースのバージョンのリストについては、https://github.com/IBM/cloud-pak/tree/master/repo/case/ibm-mq を参照してください。 7 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。

踏み台マシンを使用したエアギャップ環境での IBM MQ Operator のインストール

1. 67 ページの『前提条件』

2. 67 ページの『Docker レジストリーの準備』
3. 68 ページの『踏み台ホストの準備』
4. 69 ページの『インストーラーおよびイメージ・インベントリー用の環境変数の作成』
5. 69 ページの『IBM MQ インストーラーおよびイメージ・インベントリーのダウンロード』
6. 69 ページの『クラスター管理者として Red Hat OpenShift Container Platform クラスターにログインします。』
7. 69 ページの『IBM MQ Operator 用の Kubernetes 名前空間の作成』
8. 69 ページの『イメージのミラーリングおよびクラスターの構成』
9. 71 ページの『IBM MQ Operator のインストール』
10. 72 ページの『IBM MQ キュー・マネージャーのデプロイ』

前提条件

1. Red Hat OpenShift Container Platform クラスター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 Red Hat OpenShift Container Platform のバージョンについては、7 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。
2. Docker レジストリーが使用可能である必要があります。 詳しくは、67 ページの『Docker レジストリーの準備』を参照してください。
3. 踏み台サーバーを構成する必要があります。 詳しくは、68 ページの『踏み台ホストの準備』を参照してください。

Docker レジストリーの準備

ローカルの Docker レジストリーを使用して、ローカル環境内のすべてのイメージを保管します。 そのようなレジストリーを作成し、それが以下の要件を満たすようにする必要があります。

- Docker Manifest V2, Schema 2 をサポートしている。
- マルチ・アーキテクチャー・イメージをサポートしている。
- 踏み台サーバーと Red Hat OpenShift Container Platform クラスター・ノードの両方からアクセス可能である。
- 踏み台ホストからターゲット・レジストリーに書き込むことができるユーザーのユーザー名とパスワードがある。
- Red Hat OpenShift クラスター・ノード上にあるターゲット・レジストリーから読み取ることができるユーザーのユーザー名とパスワードを持っています。
- イメージ名でパス分離文字が許可される。

Docker レジストリーを作成した後に、以下のようにレジストリーを構成する必要があります。

1. レジストリー名前空間を作成します。
   - `ibmcom` - `dockerhub.io/ibmcom` 名前空間からのすべてのイメージを保管するための名前空間。

     `ibmcom` 名前空間は、資格情報がなくてもプルできる IBM のすべての公開イメージに使用します。
   - `cp` - `cp.icr.io/cp` リポジトリーからの IBM イメージを保管するための名前空間。

     `cp` 名前空間は、プルするために製品ライセンス・キーおよび資格情報を必要とする、IBM Entitled Registry 内のイメージ用です。 使用権キーを取得するには、使用権付き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。 「**ライセンス・キー**」セクションで、「**キーのコピー**」を選択してライセンス・キーをクリップボードにコピーしてから、後の手順で使用するために保存します。
   - `opencloudio` - `quay.io/opencloudio` からのイメージを保管するための名前空間。

     `opencloudio` 名前空間は、 quay.io で使用可能な一部の IBM オープン・ソース・コンポーネント・イメージ用です。 IBM Cloud Pak foundational services イメージは、`opencloudio` 上にホストされます。

2. 各名前空間が以下の要件を満たしていることを確認します。
   - 自動リポジトリー作成をサポートしている。
   - リポジトリーの書き込みと作成ができるユーザーの資格情報がある。 踏み台ホストが該当する資格情報を使用します。
   - すべてのリポジトリーを読み取ることができるユーザーの資格情報がある。 Red Hat OpenShift Container Platform クラスターがこれらの資格情報を使用している。

踏み台ホストの準備

Red Hat OpenShift Container Platform クラスター、ローカル Docker レジストリー、およびインターネットにアクセスできる踏み台ホストを準備します。 要塞ホストは、 IBM Cloud Pak CLI および Red Hat OpenShift Container Platform CLI がサポートするオペレーティング・システムを備えた Linux for x86-64 プラットフォーム上になければなりません。

踏み台ノードで以下のステップを実行します。

1. OpenSSL バージョン 1.11.1 以上をインストールします。
2. 踏み台ノードに Docker または Podman をインストールします。
   - Docker をインストールするには、以下のコマンドを実行します。

     ```
     yum check-update
     yum install docker
     ```

   - Podman をインストールするには、Podman Installation Instructions を参照してください。
3. 踏み台ノードに skopeo バージョン 1.x.x をインストールします。 skopeo をインストールするには、以下のコマンドを実行します。

   ```
   yum check-update
   yum install skopeo
   ```

4. IBM Cloud PakCLI をインストールします。 ご使用のプラットフォーム用のバイナリー・ファイルの最新バージョンをインストールします。 詳しくは、cloud-pak-cli を参照してください。
   a. バイナリー・ファイルをダウンロードします。

      ```
      wget https://github.com/IBM/cloud-pak-cli/releases/download/vversion-number/binary-file-name
      ```

      以下に例を示します。

      ```
      wget https://github.com/IBM/cloud-pak-cli/releases/latest/download/cloudctl-linux-amd64.tar.gz
      ```

   b. バイナリー・ファイルを解凍します。

      ```
      tar -xf binary-file-name
      ```

   c. 以下のコマンドを実行して、ファイルを変更および移動します。

      ```
      chmod 755 file-name
      mv file-name /usr/local/bin/cloudctl
      ```

   d. `cloudctl` がインストールされていることを確認します。

      ```
      cloudctl --help
      ```

5. oc Red Hat OpenShift Container Platform CLI ツールをインストールします。

   詳しくは、Red Hat OpenShift Container Platform CLI ツールを参照してください。
6. オフライン・ストアとして機能するディレクトリーを作成します。

以下に、ディレクトリーの例を示します。 後続のステップでこの例を使用します。

```
mkdir $HOME/offline
```

注: データを複数回転送しなくても済むように、このオフライン・ストアはパーシスタントでなければなりません。 また、パーシスタンスは、複数回、またはスケジュールに従って、ミラーリング・プロセスを実行するのにも役立ちます。

インストーラーおよびイメージ・インベントリー用の環境変数の作成

インストーラー・イメージ名とイメージ・インベントリーを使用して、以下の環境変数を作成します。

```
export CASE_ARCHIVE_VERSION=version_number
export CASE_ARCHIVE=ibm-mq-$CASE_ARCHIVE_VERSION.tgz
export CASE_INVENTORY=ibmMQOperator
```

ここで、*version_number* は、エアギャップ・インストールのために使用するケースのバージョンです。 使用可能なケースのバージョンのリストについては、https://github.com/IBM/cloud-pak/tree/master/repo/case/ibm-mq を参照してください。 選択するオペレーター・チャネルを決定するには、IBM MQ Operator のバージョン・サポート を参照してください。

IBM MQ インストーラーおよびイメージ・インベントリーのダウンロード

ibm-mq インストーラーおよびイメージ・インベントリーを要塞ホストにダウンロードします。

```
cloudctl case save \
    --case https://github.com/IBM/cloud-pak/raw/master/repo/case/ibm-mq/$CASE_ARCHIVE_VERSION/
$CASE_ARCHIVE \
    --outputdir $HOME/offline/
```

クラスター管理者として Red Hat OpenShift Container Platform クラスターにログインします。

以下に、Red Hat OpenShift Container Platform クラスターにログインするためのコマンド例を示します。

```
oc login cluster_host:port --username=cluster_admin_user --password=cluster_admin_password
```

IBM MQ Operator 用の Kubernetes 名前空間の作成

IBM MQ Operator をインストールするための名前空間を持つ環境変数を作成してから、名前空間を作成します。

```
export NAMESPACE=ibm-mq-test
oc create namespace ${NAMESPACE}
```

イメージのミラーリングおよびクラスターの構成

以下のステップを実行して、イメージをミラーリングし、クラスターを構成します。

注: いずれのコマンドでも二重引用符の間にティルドは使用しないでください。 例えば、`args "--registry registry --user registry_userid --pass registry_password --inputDir ~/offline"` は使用しないでください。 波形記号が展開されず、コマンドが失敗する可能性があります。

1. すべてのソース Docker レジストリーの認証資格情報を保管します。

   IBM Cloud Platform Common Services、IBM MQ Operator イメージ、IBM MQ Advanced Developer イメージはすべて、認証を必要としないパブリック・レジストリーに保管されています。 ただし、IBM MQ Advanced Server (Developer 以外)、他の製品、サード・パーティーのコンポーネントには、認証ありのレジストリーが 1 つ以上必要です。 以下のレジストリーには認証が必要です。

   - cp.icr.io

- registry.redhat.io
- registry.access.redhat.com

これらのレジストリーについて詳しくは、レジストリー名前空間の作成を参照してください。

以下のコマンドを実行して、認証が必要なすべてのレジストリーの資格情報を構成する必要があります。 該当するレジストリーごとに別個に以下のコマンドを実行します。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-creds-airgap \
--namespace ${NAMESPACE} \
--args "--registry registry --user registry_userid --pass registry_password --inputDir $HOME/
offline"
```

コマンドは、$HOME/.airgap/secrets ロケーションにあるファイル・システム上のファイルにレジストリー資格情報を保管およびキャッシュします。

2. ローカル Docker レジストリー接続情報を指定した環境変数を作成します。

```
export LOCAL_DOCKER_REGISTRY=IP_or_FQDN_of_local_docker_registry
export LOCAL_DOCKER_USER=username
export LOCAL_DOCKER_PASSWORD=password
```

注: Docker レジストリーは、80 や 443 などの標準ポートを使用します。 Docker レジストリーで標準以外のポートを使用する場合は、構文 *host:port* を使用してポートを指定します。 以下に例を示します。

```
export LOCAL_DOCKER_REGISTRY=myregistry.local:5000
```

3. ローカル Docker レジストリーの認証秘密を構成します。

注: このステップは、1 回のみ実行する必要があります。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-creds-airgap \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD}"
```

コマンドは、$HOME/.airgap/secrets ロケーションにあるファイル・システム上のファイルにレジストリー資格情報を保管およびキャッシュします。

4. グローバル・イメージ・プル秘密および **ImageContentSourcePolicy** を構成します。

   a. ノードの再始動が必要かどうかを確認してください。

   - Red Hat OpenShift Container Platform バージョン 4.4 以降、およびエアギャップを使用する IBM MQ Operator の新規インストールでは、このステップによりすべてのクラスター・ノードが再始動されます。 新しいプル秘密が適用されるまで、クラスター・リソースは使用不可になる可能性があります。
   - IBM MQ Operator 1.8 では、 CASE が更新され、イメージの追加ミラーリング・ソースが組み込まれました。 そのため、以前のバージョンの IBM MQ Operator からバージョン 1.8 以上にアップグレードすると、ノードの再始動がトリガーされます。
   - このステップでノードの再始動が必要かどうかを確認するには、このステップのコードに --dry-run オプションを追加します。 これにより、最新の **ImageContentSourcePolicy** が生成され、コンソール・ウィンドウ (**stdout**) に表示されます。 この **ImageContentSourcePolicy** 構成されているクラスターと異なる **ImageContentSourcePolicy** 場合は、再始動が行われます。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-cluster-airgap \
--namespace ${NAMESPACE} \
```

```
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline --dryRun"
```

b. グローバル・イメージ・プル・シークレットと **ImageContentSourcePolicy** を構成するには、--dry-run オプションを指定せずにこのステップのコードを実行します。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-cluster-airgap \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline"
```

5. **ImageContentSourcePolicy** リソースが作成されていることを確認します。

```
oc get imageContentSourcePolicy
```

6. オプション: 非セキュアなレジストリーを使用している場合は、ローカル・レジストリーをクラスターの **insecureRegistries** リストに追加する必要があります。

```
oc patch image.config.openshift.io/cluster --type=merge -p '{"spec":{"registrySources":
{"insecureRegistries":["'${LOCAL_DOCKER_REGISTRY}'"]}}}'
```

7. クラスター・ノード状況を確認します。

```
oc get nodes
```

imageContentsourcePolicy およびグローバル・イメージ・プル秘密が適用されると、ノード状況が **Ready**、**Scheduling**、または **Disabled** として表示されることがあります。 すべてのノードで **Ready** 状況が表示されるまで待機します。

8. イメージをローカル・レジストリーにミラーリングします。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action mirror-images \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline"
```

IBM MQ Operator のインストール

1. 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。
2. カタログ・ソースを作成します。 前の手順を実行したのと同じ端末を使用してください。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action install-catalog \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --recursive"
```

3. 共通サービス・インストーラー・オペレーターの **CatalogSource** が作成されていることを確認します。

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

4. OLM を使用して IBM MQ Operator をインストールします。
   a. 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。

   b.「**すべてのアイテム (All Items)**」 フィールドに、「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。

c.「**IBM MQ**」を選択します。

「**IBM MQ**」ウィンドウが表示されます。

d.「**インストール**」をクリックします。

「**オペレーター・サブスクリプションの作成 (Create Operator Subscription)**」ページが表示されます。

e. 7 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。

f.「**インストール・モード (Installation Mode)**」を、作成した特定の名前空間、または、クラスター全体の範囲のどちらかに設定します。

g.「**サブスクライブ**」をクリックします。

「**インストール済みのオペレーター (Installed Operators)**」ページに **IBM MQ** が追加されます。

h.「**インストール済みのオペレーター (Installed Operators)**」ページで、オペレーターの状況を確認します。 インストールが完了すると、状況が **Succeeded** に変わります。

IBM MQ キュー・マネージャーのデプロイ

インストールしたオペレーターを使用して新規キュー・マネージャーを作成するには、80 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』を参照してください。

関連タスク

72 ページの『エアギャップ環境での IBM MQ Operator またはキュー・マネージャーのアップグレードの準備』

インターネット接続がない Red Hat OpenShift クラスターでは、 IBM MQ Operator をアップグレードする前に実行する必要がある準備ステップがあります。

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 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るようになります。

手順

- 75 ページの『Red Hat OpenShift ウェブコンソールを使用して IBM MQ Operator をアップグレードする』.
- 77 ページの『Red Hat OpenShiftCLI を使用して IBM MQ Operator をアップグレードする』.
- 78 ページの『Red Hat OpenShift ウェブコンソールを使用した IBM MQ キュー・マネージャーのアップグレード』.
- 79 ページの『Red Hat OpenShift CLI を使用した IBM MQ キュー・マネージャーのアップグレード』.

OpenShift CP4I Linux エアギャップ環境での IBM MQ Operator またはキュー・マネージャーのアップグレードの準備

インターネット接続がない Red Hat OpenShift クラスターでは、 IBM MQ Operator をアップグレードする前に実行する必要がある準備ステップがあります。

始める前に

このトピックでは、以前にリリースされた IBM Cloud Pak for Integration イメージがミラーリングされるローカル・イメージ・レジストリーが既に構成されていることを前提としています。

このタスクについて

エアギャップ環境で IBM MQ Operator またはキュー・マネージャーをアップグレードする前に、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。

このタスクの最初の 4 つのステップは、 66 ページの『エアギャップ環境への IBM MQ Operator のインストール』時に実行するステップと同じであることに注意してください。

手順

1. インストーラーおよびイメージ・インベントリー用の環境変数を作成します。

   インストーラー・イメージ名とイメージ・インベントリーを使用して、以下の環境変数を作成します。

   ```
   export CASE_ARCHIVE_VERSION=version_number
   export CASE_ARCHIVE=ibm-mq-$CASE_ARCHIVE_VERSION.tgz
   export CASE_INVENTORY=ibmMQOperator
   ```

   ここで、*version_number* は、エアギャップ・インストールのために使用するケースのバージョンです。使用可能なケースのバージョンのリストについては、https://github.com/IBM/cloud-pak/tree/master/repo/case/ibm-mq を参照してください。 選択するオペレーター・チャネルを決定するには、 IBM MQ Operator のバージョン・サポート を参照してください。

2. IBM MQ インストーラーおよびイメージ・インベントリーをダウンロードします。

   ibm-mq インストーラーおよびイメージ・インベントリーを要塞ホストにダウンロードします。

   ```
   cloudctl case save \
       --case https://github.com/IBM/cloud-pak/raw/master/repo/case/ibm-mq/
   $CASE_ARCHIVE_VERSION/$CASE_ARCHIVE \
       --outputdir $HOME/offline/
   ```

3. クラスター管理者として Red Hat OpenShift Container Platform クラスターにログインします。

   以下に、Red Hat OpenShift Container Platform クラスターにログインするためのコマンド例を示します。

   ```
   oc login cluster_host:port --username=cluster_admin_user --password=cluster_admin_password
   ```

4. イメージをミラーリングして、クラスターを構成します。

   以下のステップを実行して、イメージをミラーリングし、クラスターを構成します。

   注: いずれのコマンドでも二重引用符の間にティルドは使用しないでください。 例えば、`args "--registry registry --user registry_userid --pass registry_password --inputDir ~/offline"` は使用しないでください。 波形記号が展開されず、コマンドが失敗する可能性があります。

   a. すべてのソース Docker レジストリーの認証資格情報を保管します。

      IBM Cloud Platform Common Services、IBM MQ Operator イメージ、IBM MQ Advanced Developer イメージはすべて、認証を必要としないパブリック・レジストリーに保管されています。 ただし、IBM MQ Advanced Server (Developer 以外)、他の製品、サード・パーティーのコンポーネントには、認証ありのレジストリーが 1 つ以上必要です。 以下のレジストリーには認証が必要です。

      - cp.icr.io
      - registry.redhat.io
      - registry.access.redhat.com

      これらのレジストリーについて詳しくは、レジストリー名前空間の作成を参照してください。

      以下のコマンドを実行して、認証が必要なすべてのレジストリーの資格情報を構成する必要があります。 該当するレジストリーごとに別個に以下のコマンドを実行します。

      ```
      cloudctl case launch \
      --case $HOME/offline/${CASE_ARCHIVE} \
      --inventory ${CASE_INVENTORY} \
      --action configure-creds-airgap \
      --namespace ${NAMESPACE} \
      --args "--registry registry --user registry_userid --pass registry_password --inputDir
      $HOME/offline"
      ```

コマンドは、$HOME/.airgap/secrets ロケーションにあるファイル・システム上のファイルにレジストリー資格情報を保管およびキャッシュします。

b. ローカル Docker レジストリー接続情報を指定した環境変数を作成します。

```
export LOCAL_DOCKER_REGISTRY=IP_or_FQDN_of_local_docker_registry
export LOCAL_DOCKER_USER=username
export LOCAL_DOCKER_PASSWORD=password
```

注: Docker レジストリーは、80 や 443 などの標準ポートを使用します。 Docker レジストリーで標準以外のポートを使用する場合は、構文 *host:port* を使用してポートを指定します。 以下に例を示します。

```
export LOCAL_DOCKER_REGISTRY=myregistry.local:5000
```

c. ローカル Docker レジストリーの認証秘密を構成します。

注: このステップは、1 回のみ実行する必要があります。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-creds-airgap \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD}"
```

コマンドは、$HOME/.airgap/secrets ロケーションにあるファイル・システム上のファイルにレジストリー資格情報を保管およびキャッシュします。

d. グローバル・イメージ・プル秘密および **ImageContentSourcePolicy** を構成します。

i) ノードの再始動が必要かどうかを確認してください。

- Red Hat OpenShift Container Platform バージョン 4.4 以降、およびエアギャップを使用する IBM MQ Operator の新規インストールでは、このステップによりすべてのクラスター・ノードが再始動されます。 新しいプル秘密が適用されるまで、クラスター・リソースは使用不可になる可能性があります。
- IBM MQ Operator 1.8 では、CASE が更新され、イメージの追加ミラーリング・ソースが組み込まれました。 そのため、以前のバージョンの IBM MQ Operator からバージョン 1.8 以上にアップグレードすると、ノードの再始動がトリガーされます。
- このステップでノードの再始動が必要かどうかを確認するには、このステップのコードに --dry-run オプションを追加します。 これにより、最新の **ImageContentSourcePolicy** が生成され、コンソール・ウィンドウ (**stdout**) に表示されます。 この **ImageContentSourcePolicy** 構成されているクラスターと異なる **ImageContentSourcePolicy** 場合は、再始動が行われます。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-cluster-airgap \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline --dryRun"
```

ii) グローバル・イメージ・プル・シークレットと **ImageContentSourcePolicy** を構成するには、--dry-run オプションを指定せずにこのステップのコードを実行します。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action configure-cluster-airgap \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline"
```

e. **ImageContentSourcePolicy** リソースが作成されていることを確認します。

```
oc get imageContentSourcePolicy
```

f. オプション: 非セキュアなレジストリーを使用している場合は、ローカル・レジストリーをクラスターの **insecureRegistries** リストに追加する必要があります。

```
oc patch image.config.openshift.io/cluster --type=merge -p '{"spec":{"registrySources":
{"insecureRegistries":["'${LOCAL_DOCKER_REGISTRY}'"]}}}'
```

g. クラスター・ノード状況を確認します。

```
oc get nodes
```

imageContentsourcePolicy およびグローバル・イメージ・プル秘密が適用されると、ノード状況が **Ready**、**Scheduling**、または **Disabled** として表示されることがあります。 すべてのノードで **Ready** 状況が表示されるまで待機します。

h. イメージをローカル・レジストリーにミラーリングします。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action mirror-images \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --user ${LOCAL_DOCKER_USER} --pass $
{LOCAL_DOCKER_PASSWORD} --inputDir $HOME/offline"
```

5. カタログ・ソースをアップグレードします。

前の手順を実行したのと同じ端末を使用してください。

```
cloudctl case launch \
--case $HOME/offline/${CASE_ARCHIVE} \
--inventory ${CASE_INVENTORY} \
--action install-catalog \
--namespace ${NAMESPACE} \
--args "--registry ${LOCAL_DOCKER_REGISTRY} --recursive"
```

次のタスク

これで、以下のいずれかのタスクを実行して、 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準備ができました。

- 75 ページの『Red Hat OpenShift ウェブコンソールを使用して IBM MQ Operator をアップグレードする』
- 77 ページの『Red Hat OpenShiftCLI を使用して IBM MQ Operator をアップグレードする』
- 78 ページの『Red Hat OpenShift ウェブコンソールを使用した IBM MQ キュー・マネージャーのアップグレード』
- 79 ページの『Red Hat OpenShift CLI を使用した IBM MQ キュー・マネージャーのアップグレード』
- 80 ページの『プラットフォーム Navigator を使用した Red Hat OpenShift での IBM MQ キュー・マネージャーのアップグレード』

OpenShift CP4I Red Hat OpenShift ウェブコンソールを使用して IBM MQ Operator をアップグレードする

Operator Hub を使用して、IBM MQ Operator をアップグレードできます。

始める前に

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。

エアギャップ環境で IBM MQ Operator をアップグレードする前に、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。 エアギャップ環境での IBM MQ Operator またはキュー・マネージャーのアップグレードの準備を参照してください。

手順

1. 7 ページの『IBM MQ Operator のバージョン・サポート』を確認して、どのオペレーター・チャネルにアップグレードすべきかを判別します。
2. オプション: 1.5 より前のバージョンの IBM MQ Operator から IBM MQ Operator 1.5 以降にアップグレードする場合は、まず IBM Cloud Pak foundational services のバージョンをアップグレードする必要があります。

   詳しくは、76 ページの『IBM Cloud Pak foundational services を、Red Hat OpenShift Web コンソールを使用してアップグレード』を参照してください。
3. 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。 オペレーターを新規メジャー/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)「**サブスクリプション**」タブにナビゲートします。
   d)「**チャネル**」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) 必要なチャネルを選択して、「**保存**」をクリックします。
   新しいチャネルで使用できる最新バージョンに Operator がアップグレードされます。 7 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。

次のタスク

IBM Cloud Pak foundational services 3.7 にアップグレードした場合は、IBM Cloud Pak for Integration ライセンスを使用するすべてのキュー・マネージャーをアップグレードまたは再始動する必要があります。これを行う方法について詳しくは、78 ページの『Red Hat OpenShift ウェブコンソールを使用した 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 *IBM Cloud Pak foundational services を、Red Hat OpenShift Web コンソールを使用してアップグレード*

1.5 より前のバージョンの IBM MQ Operator から IBM MQ Operator 1.5 以降にアップグレードする場合は、まず IBM Cloud Pak foundational services のバージョンをアップグレードする必要があります。

始める前に

注 : このタスクを実行する必要があるのは、1.5 より前のバージョンの IBM MQ Operator から IBM MQ Operator 1.5 以降にアップグレードする場合に限られます。

CP4I IBM Cloud Pak for Integration ライセンスを使用するキュー・マネージャーがある場合は、このアップグレードの後、Web コンソールにアクセスするためにキュー・マネージャーの再始動が必要になります。また、Web コンソールにログインするときにその他のエラーが表示されます。 オペレーター・アップグレードが完了した後で、`.spec.version` の最新値 (選択した IBM MQ バージョン) にアップグレードすることにより、このようなエラーを修正できます。

CP4I 既存のキュー・マネージャーがあり、IBM Cloud Pak for Integration オペレーション・ダッシュボードを使用する場合は、アップグレードの前に 113 ページの『IBM MQ 9.2.2 または 9.2.3 を IBM

Cloud Pak for Integration2021.4 のオペレーション・ダッシュボード統合してデプロイまたはアップグレードします』を参照してください。

手順

1. 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。
2. 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。
3. 「**IBM Cloud Pak foundational services Operator**」を選択します。 これは、バージョン 3.7 より前のバージョンでは、「**IBM Common Services Operator**」と呼ばれていました。
4. 「**Subscription**」タブにナビゲートします。
5. 「**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。
6. **v3** チャネルを選択し、「**保存**」をクリックします。
   IBM Cloud Pak foundational services オペレーターは、新規チャネルで使用できる最新バージョンにアップグレードします。 7 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。

次のタスク

IBM MQ Operator をアップグレードする準備ができました。

OpenShift CP4I Red Hat OpenShiftCLI を使用して IBM MQ Operator をアップグレードする

IBM MQ Operator はコマンド・ラインからアップグレードできます。

始める前に

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。

エアギャップ環境で IBM MQ Operator をアップグレードする前に、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。 エアギャップ環境での IBM MQ Operator またはキュー・マネージャーのアップグレードの準備を参照してください。

手順

1. 7 ページの『IBM MQ Operator のバージョン・サポート』を確認して、どのオペレーター・チャネルにアップグレードすべきかを判別します。
2. オプション: 1.5 より前のバージョンの IBM MQ Operator から IBM MQ Operator 1.5 以降にアップグレードする場合は、まず IBM Cloud Pak foundational services のバージョンをアップグレードする必要があります。

   詳しくは、78 ページの『IBM Cloud Pak foundational services を Red Hat OpenShift CLI でアップグレード』を参照してください。
3. 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。 Operator を新規のメジャー/マイナーのバージョンにアップグレードするには、IBM MQ Operator の「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) 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
```

次のタスク

IBM Cloud Pak foundational services 3.7 にアップグレードした場合は、IBM Cloud Pak for Integration ライセンスを使用するすべてのキュー・マネージャーをアップグレードまたは再始動する必要があります。これを行う方法について詳しくは、79 ページの『Red Hat OpenShift CLI 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 *IBM Cloud Pak foundational services を Red Hat OpenShift CLI でアップグレード*

1.5 より前のバージョンの IBM MQ Operator から IBM MQ Operator 1.5 以降にアップグレードする場合は、まず IBM Cloud Pak foundational services のバージョンをアップグレードする必要があります。

始める前に

注 : このタスクを実行する必要があるのは、1.5 より前のバージョンの IBM MQ Operator から IBM MQ Operator 1.5 以降にアップグレードする場合に限られます。

CP4I IBM Cloud Pak for Integration ライセンスを使用するキュー・マネージャーがある場合は、このアップグレードの後、Web コンソールにアクセスするためにキュー・マネージャーの再始動が必要になります。また、Web コンソールにログインするときにその他のエラーが表示されます。 オペレーター・アップグレードが完了した後で、`.spec.version` の最新値 (選択した IBM MQ バージョン) にアップグレードすることにより、このようなエラーを修正できます。

CP4I 既存のキュー・マネージャーがあり、IBM Cloud Pak for Integration オペレーション・ダッシュボードを使用する場合は、アップグレードの前に 113 ページの『IBM MQ 9.2.2 または 9.2.3 を IBM Cloud Pak for Integration2021.4 のオペレーション・ダッシュボード統合してデプロイまたはアップグレードします』 を参照してください。

手順

1. **`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。
2. v3 IBM Cloud Pak foundational services 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 -n ibm-common-services ibm-common-service-operator
   -o=jsonpath='{.status.channels[*].name}'
   ```

3. `Subscription` にパッチを適用して希望の更新チャネルに移動します: v3

   ```
   oc patch subscription ibm-common-service-operator --patch '{"spec":{"channel":"v3"}}' --
   type=merge
   ```

次のタスク

IBM MQ Operator をアップグレードする準備ができました。

OpenShift CP4I Red Hat OpenShift ウェブコンソールを使用した IBM MQ キュー・マネージャーのアップグレード

IBM MQ Operator を使用してデプロイされた IBM MQ キュー・マネージャーは、オペレーター・ハブを使用して Red Hat OpenShift でアップグレードすることができます。

始める前に

- 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。

- IBM MQ Operator で対象の更新チャネルを使用していることを確認してください。 72 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。

エアギャップ環境でキュー・マネージャーをアップグレードする前に、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。 エアギャップ環境での IBM MQ Operator またはキュー・マネージャーのアップグレードの準備を参照してください。

手順

1. 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。
2. 「**IBM MQ Operator**」を選択します。
   「**IBM MQ Operator**」ウィンドウが表示されます。
3. 「**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。
4. アップグレードするキュー・マネージャーを選択します。
5. YAML タブに移動します。
6. 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

   チャネルから IBM MQ Operator バージョンおよび IBM MQ キュー・マネージャー・バージョンへのマッピングについては、 7 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。
7. 更新したキュー・マネージャー YAML を保存します。

OpenShift CP4I Red Hat OpenShift CLI 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 を使用してデプロイされた IBM MQ キュー・マネージャーは、コマンド行を使用して Red Hat OpenShift にアップグレードすることができます。

始める前に

以下の手順は、クラスター管理者でないと実行できません。

- `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。
- IBM MQ Operator で対象の更新チャネルを使用していることを確認してください。 72 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。

エアギャップ環境でキュー・マネージャーをアップグレードする前に、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。 エアギャップ環境での IBM MQ Operator またはキュー・マネージャーのアップグレードの準備を参照してください。

手順

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

チャネルから IBM MQ Operator バージョンおよび IBM MQ キュー・マネージャー・バージョンへのマッピングについては、 7 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。

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。

CP4I プラットフォーム Navigator を使用した Red Hat OpenShift での IBM MQ キュー・マネージャーのアップグレード

IBM MQ Operator を使用してデプロイされた IBM MQ キュー・マネージャーは、 IBM Cloud Pak for Integration Platform Navigator を使用して Red Hat OpenShift でアップグレードすることができます。

始める前に

- アップグレードするキュー・マネージャーが含まれている名前空間内の IBM Cloud Pak for Integration Platform Navigator にログインします。
- IBM MQ Operator で対象の更新チャネルを使用していることを確認してください。 72 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。

エアギャップ環境でキュー・マネージャーをアップグレードする前に、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。 エアギャップ環境での IBM MQ Operator またはキュー・マネージャーのアップグレードの準備を参照してください。

手順

1. IBM Cloud Pak for Integration Platform Navigator ホーム・ページで「**ランタイム (Runtimes)**」タブをクリックします。
2. アップグレードが可能なキュー・マネージャーには、「**バージョン**」の隣に青色の **i** マークが付いています。 **i** マークをクリックすると、「**利用できる新しいバージョン (New version available)**」が表示されます。
3. アップグレードするキュー・マネージャーの右端にある 3 点メニューをクリックして、「**バージョンの変更 (Change version)**」をクリックします。
4. 「**新しいチャネルまたはバージョンの選択 (Select a new channel or version)**」の下で、必要なアップグレード・バージョンを選択します。
5. 「**バージョンの変更 (Change version)**」をクリックします。

タスクの結果

キュー・マネージャーがアップグレードされます。

OpenShift CP4I 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

IBM MQ 9.1.5 以降は、 IBM MQ Operator を使用して Red Hat OpenShift にデプロイされます。

このタスクについて

手順

- 80 ページの『IBM MQ 用の Red Hat OpenShift プロジェクトの準備』.
- 82 ページの『Red Hat OpenShift Container Platform クラスターにキュー・マネージャーをデプロイする方法』.

OpenShift CP4I IBM MQ 用の Red Hat OpenShift プロジェクトの準備

Red Hat OpenShift Container Platform クラスターを準備して、キュー・マネージャーのデプロイができるようにします。

手順

- 81 ページの『Red Hat OpenShift ウェブコンソールを使用した IBM MQ 用の Red Hat OpenShift プロジェクトの準備』.
- 82 ページの『Red Hat OpenShift CLI を使用した IBM MQ のための Red Hat OpenShift プロジェクトの準備』.

関連タスク

82 ページの『Red Hat OpenShift Container Platform クラスターにキュー・マネージャーをデプロイする方法』
QueueManager カスタム・リソースを使用して Red Hat OpenShift Container Platform クラスターにキュー・マネージャーをデプロイできます。

OpenShift CP4I *Red Hat OpenShift ウェブコンソールを使用した IBM MQ 用の Red Hat OpenShift プロジェクトの準備*

IBM MQ Operator を使用してキュー・マネージャーをデプロイできるように、Red Hat OpenShift Container Platform クラスターを準備します。 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

注 : 他の IBM Cloud Pak for Integration コンポーネントが既にインストールされているプロジェクトで IBM MQ を使用する予定の場合は、以下の手順に従う必要はありません。

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。

このタスクについて

IBM MQ Operator・イメージは、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。 使用権キーがまだない場合には、以下の説明に従って、使用権キーを取得し、プル・シークレットを作成してください。

手順

1. ご自身の ID に割り当てられている使用権キーを取得します。
   a) 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。
2. キュー・マネージャーをデプロイするプロジェクトに、使用権キーを含むシークレットを作成します。
   a) ナビゲーション・ペインで、「**ワークロード (Workloads)**」 > 「**シークレット (Secret)**」をクリックします。
   「シークレット (Secrets)」ページが表示されます。
   b)「**プロジェクト (Project)**」ドロップダウンで、IBM MQ をインストールするプロジェクトを選択します。
   c)「**作成**」ボタンをクリックし、「**イメージ・プル・シークレット (Image Pull Secret)**」を選択します。
   d)「**名前**」フィールドに、`ibm-entitlement-key` を入力します。
   e)「**レジストリー・サーバー・アドレス (Registry Server Address)**」フィールドに、`cp.icr.io` と入力します。
   f)「**ユーザー名 (Username)**」フィールドに、`cp` と入力します。
   g)「**パスワード**」 フィールドに、前の手順でコピーした使用権キーを入力します。
   h)「**E メール (Email)**」フィールドに、使用権を持っているソフトウェアに関連付けられた IBM ID を入力します。

次のタスク

84 ページの『Red Hat OpenShift ウェブコンソールを使用したキュー・マネージャーのデプロイ』

OpenShift CP4I *Red Hat OpenShift CLI を使用した IBM MQ のための Red Hat OpenShift プロジェクトの準備*

IBM MQ Operator を使用してキュー・マネージャーをデプロイできるように、Red Hat OpenShift Container Platform クラスターを準備します。 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

注 : 他の IBM Cloud Pak for Integration コンポーネントが既にインストールされているプロジェクトで IBM MQ を使用する予定の場合は、以下の手順に従う必要はありません。

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。

このタスクについて

IBM MQ Operator・イメージは、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。 使用権キーがまだない場合には、以下の説明に従って、使用権キーを取得し、プル・シークレットを作成してください。

手順

1. ご自身の ID に割り当てられている使用権キーを取得します。
   a) 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。
2. キュー・マネージャーをデプロイするプロジェクトに、使用権キーを含むシークレットを作成します。
   *<entitlement-key>*は手順 1 で取得した鍵、*<user-email>*は権利付きソフトウェアに関連する IBM の ID を指定して、以下のコマンドを実行してください。

```
oc create secret docker-registry ibm-entitlement-key \
--docker-server=cp.icr.io \
--docker-username=cp \
--docker-password=<entitlement-key> \
--docker-email=<user-email>
```

次のタスク

85 ページの『Red Hat OpenShift CLI を使用したキュー・マネージャーのデプロイ』

OpenShift CP4I Red Hat OpenShift Container Platform クラスターにキュー・マネージャーをデプロイする方法

QueueManager カスタム・リソースを使用して Red Hat OpenShift Container Platform クラスターにキュー・マネージャーをデプロイできます。

手順

- CP4I
  83 ページの『IBM Cloud Pak for Integration Platform Navigator を使用したキュー・マネージャーのデプロイ』.
- OpenShift
  84 ページの『Red Hat OpenShift ウェブコンソールを使用したキュー・マネージャーのデプロイ』.

- OpenShift
  85 ページの『Red Hat OpenShift CLI を使用したキュー・マネージャーのデプロイ』.

関連タスク

87 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。

CP4I *IBM Cloud Pak for Integration Platform Navigator を使用したキュー・マネージャーのデプロイ*

IBM Cloud Pak for Integration Platform Navigator を使用して Red Hat OpenShift Container Platform クラスターにキュー・マネージャーをデプロイするには、 QueueManager カスタム・リソースを使用します。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

ブラウザーで、IBM Cloud Pak for Integration Platform Navigator を起動します。

今回初めてこの Red Hat OpenShift プロジェクトにキュー・マネージャーをデプロイするという場合は、80 ページの『IBM MQ 用の Red Hat OpenShift プロジェクトの準備』の手順に従ってください。

手順

1. キュー・マネージャーをデプロイします。

   以下の例では、「クイック・スタート」のキュー・マネージャーをデプロイします。このキュー・マネージャーでは、一時 (非永続) ストレージを使用し、MQ セキュリティーはオフにします。 キュー・マネージャーを再始動するとメッセージは失われます。 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。

   a) IBM Cloud Pak for Integration Platform Navigator で、「**管理**」 をクリックしてから 「**統合ランタイム (Integration Runtimes)**」をクリックします。 古いバージョンの IBM Cloud Pak for Integration Platform Navigator で、「**ランタイムとインスタンス (Runtime and instances)**」をクリックします。

   b)「**インスタンスの作成 (Create instance)**」をクリックします。

   c)「**メッセージング (Messaging)**」を選択して、「**次へ**」をクリックします。 古いバージョンの IBM Cloud Pak for Integration Platform Navigator で、「**キュー・マネージャー**」をクリックし、「**次へ**」をクリックします。

   QueueManager のインスタンスを作成するためのフォームが表示されます。

   注 :「**コード (Code)**」をクリックして、QueueManager 構成 YAML を表示したり変更したりすることもできます。

   d)「**詳細**」 セクションで、「**名前**」フィールドを確認または更新し、キュー・マネージャー・インスタンスを作成する**名前空間**を指定します。

   e) IBM Cloud Pak for Integration の使用条件に同意する場合は、「**ライセンスへの同意 (License acceptance)**」を「**オン (On)**」に変更します。

   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

   f)「**キュー・マネージャー (Queue Manager)**」セクションで、基礎キュー・マネージャーの**名前**を確認または更新します。 古いバージョンの IBM Cloud Pak for Integration Platform Navigator で、「**キュー・マネージャーの構成 (Queue Manager Config)**」セクションを使用します。

   デフォルトでは、IBM MQ のクライアント・アプリケーションで使用するキュー・マネージャーの名前は QueueManager と同じ名前になります。ただし、無効な文字 (ハイフンなど) は除去されます。

   g) **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 の状況は Pending になっているはずです。

2. キュー・マネージャーが実行されていることを確認します。

   QueueManager の状況が Running になったら、作成は完了です。

関連タスク

110 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。

117 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 への接続方法。

OpenShift CP4I *Red Hat OpenShift* ウェブコンソールを使用したキュー・マネージャーのデプロイ

Red Hat OpenShift Web コンソールで、QueueManager カスタム・リソースを使用して Red Hat OpenShift Container Platform クラスターにキュー・マネージャーをデプロイできます。 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 クラスターのウェブコンソールにログインします。 使用する既存のプロジェクト (名前空間) を選択するか、新規プロジェクトを作成する必要があります。

今回初めてこの Red Hat OpenShift プロジェクトにキュー・マネージャーをデプロイするという場合は、80 ページの『IBM MQ 用の Red Hat OpenShift プロジェクトの準備』の手順に従ってください。

手順

1. キュー・マネージャーをデプロイします。

   以下の例では、「クイック・スタート」のキュー・マネージャーをデプロイします。このキュー・マネージャーでは、一時 (非永続) ストレージを使用し、MQ セキュリティーはオフにします。 キュー・マネージャーを再始動するとメッセージは失われます。 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。

   a) Red Hat OpenShift ウェブコンソールで、ナビゲーション・ペインから **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。

   b)「**IBM MQ**」をクリックします。

   c)「**キュー・マネージャー (Queue Manager)**」タブをクリックします。

   d)「**QueueManager の作成 (Create QueueManager)**」ボタンをクリックします。

   YAML エディターが表示され、QueueManager リソースのサンプル YAML が示されます。

   注 :「**フォームの編集 (Edit Form)**」をクリックして、QueueManager 構成を表示したり変更したりすることもできます。

   e) 使用条件に同意する場合は、「**ライセンスへの同意 (License acceptance)**」を「**オン (On)**」に変更します。

   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、128 ページの『mq.ibm.com/v1beta1 のライセンスのリファレンス』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

   f) **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。

2. キュー・マネージャーが実行されていることを確認します。

   QueueManager の状況が Running になったら、作成は完了です。

関連タスク

110 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。

117 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 への接続方法。

OpenShift CP4I *Red Hat OpenShift CLI* を使用したキュー・マネージャーのデプロイ

コマンド・ライン・インターフェース (CLI) で、QueueManager カスタム・リソースを使用して Red Hat OpenShift Container Platform クラスターにキュー・マネージャーをデプロイできます。 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。

今回初めてこの Red Hat OpenShift プロジェクトにキュー・マネージャーをデプロイするという場合は、80 ページの『IBM MQ 用の Red Hat OpenShift プロジェクトの準備』の手順に従ってください。

手順

1. キュー・マネージャーをデプロイします。

   以下の例では、「クイック・スタート」のキュー・マネージャーをデプロイします。このキュー・マネージャーでは、一時 (非永続) ストレージを使用し、MQ セキュリティーはオフにします。 キュー・マネージャーを再始動するとメッセージは失われます。 YAML の内容を調整することで、キュー・マネージャーのさまざまな設定を変更できます。

   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: 9.2.5.0-r3
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: NonProduction
  web:
    enabled: true
  queueManager:
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
  template:
    pod:
      containers:
        - name: qmgr
          env:
```

```
          - name: MQSNOAUT
            value: "yes"
```

重要 : IBM Cloud Pak for Integration 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、128 ページの『mq.ibm.com/v1beta1 のライセンスのリファレンス』を参照してください。

この例では、キュー・マネージャーとともに Web サーバーもデプロイし、IBM Cloud Pak Identity and Access Manager を使用したシングル・サインオンを Web コンソールで有効にしています。

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。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: quickstart
spec:
  version: 9.2.5.0-r3
  license:
    accept: false
    license: L-APIG-BZDDDY
  web:
    enabled: true
  queueManager:
    name: "QUICKSTART"
    storage:
      queueManager:
        type: ephemeral
  template:
    pod:
      containers:
       - name: qmgr
         env:
         - name: MQSNOAUT
           value: "yes"
```

重要: MQ のご使用条件に同意する場合は、`accept: false` を `accept: true` に変更します。 ライセンスについて詳しくは、128 ページの『mq.ibm.com/v1beta1 のライセンスのリファレンス』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
```

2. キュー・マネージャーが実行されていることを確認します。
   デプロイメントを検証するには、次のコマンドを実行します。

```
oc describe queuemanager <QueueManagerResourceName>
```

その後、状況を確認します。
例えば、次を実行します。

```
oc describe queuemanager quickstart
```

さらに、`status.Phase` フィールドが `Running` を示していることを確認します

関連タスク

110 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。

117 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 への接続方法。

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

- 87 ページの『例: MQSC ファイルと INI ファイルの提供』
- 88 ページの『例: TLS 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。 キュー・マネージャーをデプロイする名前空間 (プロジェクト) 内にそれらを作成してください。

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。

この方法で MQSC と INI を提供するには、IBM MQ Operator 1.1 以上が必要です。

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。

ConfigMap の例 - 以下の YAML をクラスターに適用します。

```
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
```

QueueManager の例 - コマンド・ラインか IBM Cloud Pak for Integration Platform Navigator を使用して、以下の構成でキュー・マネージャーをデプロイします。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: mqsc-ini-cp4i
spec:
  version: 9.2.5.0-r3
  license:
    accept: false
```

```
    license: L-RJON-C7QG3S
    use: NonProduction
  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 Cloud Pak for Integration 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。 ライセンスの詳細については、mq.ibm.com/v1beta1 のライセンス交付に関する参照資料を参照してください。

追加情報:

- キュー・マネージャーでは、この例のように 1 つの Kubernetes ConfigMap またはシークレットを使用するように構成することも、複数の Kubernetes ConfigMap とシークレットを使用するように構成することも可能です。
- 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、`example1.mqsc` は、キュー・マネージャー構成での表示順序に関係なく、常に `example2.mqsc` の前に処理されます。
- 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。

OpenShift CP4I Linux 例: *TLS* 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 Red Hat OpenShift Container Platform にデプロイします。 サンプル・クライアントとキュー・マネージャーの間に単方向の TLS 通信を構成します。 この例では、メッセージの書き込みと読み取りによって構成が成功したことを確認します。

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 をインストールし、 `samp/bin` と `bin` を パスに追加します。 **runmqakm**、**amqsputc**、**amqsgetc** の各アプリケーションが必要です。これらのアプリケーションは、以下のとおり、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: 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 https://ibm.biz/mq92redistclients から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/
- オペレーティング・システムに対応した OpenSSL ツールをインストールします。
- この例のための Red Hat 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、Red Hat 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。 完了したら、メッセージの書き込みと読み取りによって、TLS を使用してキュー・マネージャーが構成されていることを検証します。

IBM MQ サーバーの TLS 秘密鍵と証明書の作成

以下のコード例は、キュー・マネージャーの自己署名証明書を作成する方法と、クライアントのトラストストアとして機能する鍵データベースに証明書を追加する方法を示しています。 秘密鍵と証明書がすでに存在する場合は、代わりにそれらを使用できます。

自己署名証明書は、開発目的のみで使用すべきであることに注意してください。

現行ディレクトリーへの自己署名の秘密鍵と公開証明書の作成

以下のコマンドを実行します。

```
openssl req -newkey rsa:2048 -nodes -keyout tls.key -subj "/CN=localhost" -x509 -days 3650
-out tls.crt
```

クライアントの鍵データベースへのサーバーの公開鍵の追加

クライアント・アプリケーションのトラストストアとして鍵データベースを使用します。

クライアントの鍵データベースを作成します。

```
runmqakm -keydb -create -db clientkey.kdb -pw password -type cms -stash
```

前に生成した公開鍵をクライアントの鍵データベースに追加します。

```
runmqakm -cert -add -db clientkey.kdb -label mqservercert -file tls.crt -format ascii
-stashed
```

キュー・マネージャー・デプロイメントの TLS 証明書の構成

キュー・マネージャーが鍵と証明書を参照して適用できるようにするために、上で作成したファイルを参照する Kubernetes TLS シークレットを作成します。 まず、このタスクの開始前に作成した名前空間で作業していることを確認してください。

```
oc create secret tls example-tls-secret --key="tls.key" --cert="tls.crt"
```

MQSC コマンドを組み込んだ構成マップの作成

MQSC コマンドを組み込んだ Kubernetes 構成マップを作成します。そのコマンドによって、新しいキューと SVRCONN チャネルを作成し、*nobody* というユーザーだけをブロックすることによって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。

この方法を使用するのは、開発のためだけに限定してください。

前に作成した名前空間で作業していることを確認してから (始める前にを参照)、OCP UI またはコマンド・ラインで以下の YAML を入力します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-tls-configmap
data:
  tls.mqsc: |
    DEFINE QLOCAL('EXAMPLE.QUEUE') REPLACE
    DEFINE CHANNEL(SECUREQMCHL) CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCAUTH(OPTIONAL)
SSLCIPH('ANY_TLS12_OR_HIGHER')
    SET CHLAUTH(SECUREQMCHL) TYPE(BLOCKUSER) USERLIST('nobody') ACTION(ADD)
```

必要な OCP ルートの作成

このタスクを開始する前に作成した名前空間で作業していることを確認してから、OCP UI またはコマンド・ラインで以下の YAML を入力します。

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: example-tls-route
spec:
  host: secureqmchl.chl.mq.ibm.com
  to:
    kind: Service
    name: secureqm-ibm-mq
  port:
    targetPort: 1414
  tls:
    termination: passthrough
```

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。 前に作成した構成マップの MQSC で指定したチャネル名を変更する場合は、ホスト・フィールドも、ここと、後で作成する CCDT ファイルで、変更する必要があります。 詳しくは、110 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。

キュー・マネージャーのデプロイ

重要: この例では、*MQSNOAUT* 変数を使用してキュー・マネージャーでの許可を無効にします。これにより、TLS を使用してクライアントを接続するために必要なステップに集中できるようになります。 ただしこの方法は、IBM MQ の実動デプロイメントではお勧めできません。そのようにすると、接続するすべてのアプリケーションが全管理権限を持つようになり、個々のアプリケーションのアクセス権を低くするメカニズムも存在しないからです。

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。 前に作成した構成マップとシークレットを参照していること、*MQSNOAUT* 変数を使用していることに注意してください。

このタスクを開始する前に作成した名前空間で作業していることを確認してから、OCP UI、コマンド・ライン、IBM Cloud Pak for Integration Platform Navigator のいずれかで以下の 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に同意します。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: secureqm
spec:
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: Production
  queueManager:
    name: SECUREQM
    mqsc:
    - configMap:
        name: example-tls-configmap
        items:
        - tls.mqsc
    storage:
      queueManager:
        type: ephemeral
  template:
    pod:
      containers:
        - env:
            - name: MQSNOAUT
              value: 'yes'
          name: qmgr
  version: 9.2.5.0-r3
  web:
    enabled: true
  pki:
```

```
    keys:
      - name: example
        secret:
          secretName: example-tls-secret
          items:
          - tls.key
          - tls.crt
```

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 secureqm
```

キュー・マネージャーへの接続のテスト

キュー・マネージャーで単方向の TLS 通信が構成されていることを確認するために、サンプル・アプリケーション **amqsputc** と **amqsgetc** を使用します。

キュー・マネージャーのホスト名の検索

次のコマンドを使用して、経路 secureqm-ibm-mq-qm のキュー・マネージャー完全修飾ホスト名を見つけます:

```
oc get routes secureqm-ibm-mq-qm
```

キュー・マネージャーの詳細情報の指定

キュー・マネージャーの詳細を指定するファイル CCDT.JSON を作成します。 ホストの値を、前のステップで確認したホスト名に置き換えてください。

```
{
    "channel":
    [
        {
            "name": "SECUREQMCHL",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "<hostname from previous step>",
                        "port": 443
                    }
                ],
                "queueManager": "SECUREQM"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ECDHE_RSA_AES_128_CBC_SHA256"
            },
            "type": "clientConnection"
        }
   ]
}
```

環境変数のエクスポート

オペレーティング・システムに適した方法で以下の環境変数をエクスポートします。 これらの変数は **amqsputc** と **amqsgetc** によって読み取られます。

システム上のファイルへのパスを更新します。

```
export MQCCDTURL='<full path to file>/CCDT.JSON'
export MQSSLKEYR='<full path to file>/clientkey'
```

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。

```
amqsputc EXAMPLE.QUEUE SECUREQM
```

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。

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。

```
amqsgetc EXAMPLE.QUEUE SECUREQM
```

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。

TLS を有効にしてキュー・マネージャーをデプロイする操作を正常に実行できました。クライアントからキュー・マネージャーに対してメッセージの書き込みと読み取りをセキュアな方法で実行できることも確認できました。

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、115 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 CP4I 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

このタスクについて

手順

- V 9.2.3
  93 ページの『ネイティブ HA』.
- V 9.2.3
  95 ページの『例: ネイティブ HA キュー・マネージャーの構成』.
- 104 ページの『例: マルチ・インスタンス・キュー・マネージャーの構成』.

CP4I CD V 9.2.3 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。

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。

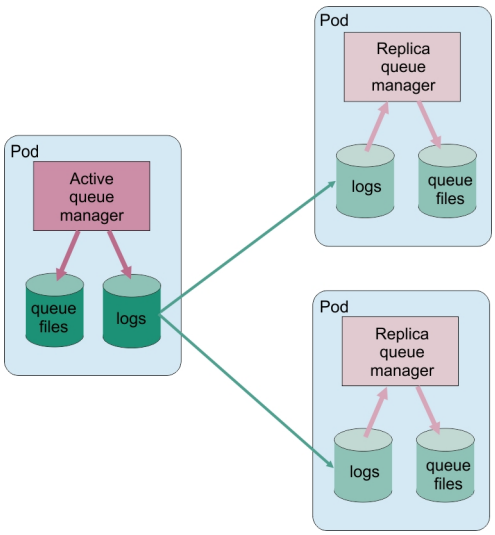


図 1. ネイティブ *HA* 構成の例

CP4I CD V 9.2.3 *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-RJON-C7QG3S
    use: Production
  queueManager:
    availability:
      type: NativeHA
  version: 9.2.5.0-r3
```

`.spec.queueManager.availability.type` フィールドは NativeHA に設定する必要があります。

ネイティブ HA は、IBM MQ 9.2.3 以上で使用できます。

`.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。 95 ページの『例: ネイティブ HA キュー・マネージャーの構成』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。

関連資料

95 ページの『例: ネイティブ HA キュー・マネージャーの構成』
この例は、IBM MQ Operator を使用して、ネイティブ高可用性機能を使用するキュー・マネージャーを Red Hat OpenShift Container Platform (OCP) にデプロイする方法を示しています。

OpenShift CP4I Linux CD V 9.2.3 例: ネイティブ *HA* キュー・マネージャーの構成

この例は、IBM MQ Operator を使用して、ネイティブ高可用性機能を使用するキュー・マネージャーを Red Hat OpenShift Container Platform (OCP) にデプロイする方法を示しています。

開始前に

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 をインストールし、インストール済みの samp/bin ディレクトリーおよび bin ディレクトリーを パスに追加します。 この例で必要な **runmqakm**、**amqsputc**、**amqsgetc** の各アプリケーションはクライアントで提供されています。 以下のように IBM MQ client をインストールします。
  - Windows Linux Windows および Linux の場合: 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 https://ibm.biz/mq92redistclients から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://ibm.biz/mqdevmacclient を参照してください。
- オペレーティング・システムに対応した OpenSSL ツールをインストールします。 この作業は、まだ秘密鍵と証明書がない場合に、キュー・マネージャー用の自己署名証明書を生成するために必要です。
- この例では、Red Hat OpenShift Container Platform(OCP)プロジェクト/名前空間を作成し、タスク 80 ページの『IBM MQ 用の Red Hat OpenShift プロジェクトの準備』のステップに従います
- コマンド・ラインで OCP クラスターにログインし、今作成した名前空間に切り替えます。
- 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。
- キュー・マネージャーで使用されるデフォルトのストレージ・クラスを OCP で構成します。 デフォルトのストレージ・クラスを設定せずにこのチュートリアルを最後まで行う場合は、注 2: デフォルト以外のストレージ・クラスの使用を参照してください。

本タスクについて

ネイティブ HA キュー・マネージャーでは、1 つのアクティブ・ポッドと 2 つのレプリカ・ポッド (いずれも Kubernetes ポッド) を使用します。 これらのポッドは、ちょうど 3 つのレプリカと Kubernetes 永続ボリューム一式で構成される Kubernetes ステートフル・セットの一部として稼働します。 ネイティブ HA キュー・マネージャーの詳細については、16 ページの『コンテナー内の IBM MQ の高可用性』を参照してください。

この例では、永続ストレージを使用し、TLS で構成されている、ネイティブ HA キュー・マネージャーを定義するカスタム・リソース YAML を示します。 キュー・マネージャーを OCP にデプロイした後、アクティブ・キュー・マネージャー・ポッドの障害をシミュレートします。 自動復旧が行われた後、メッセージの書き込みと読み取りを行うことによって、障害発生後の復旧に成功したことが証明されます。

例

MQ サーバーの TLS 秘密鍵と証明書の作成

キュー・マネージャーの自己署名証明書を作成し、クライアントのトラストストアとして機能する鍵データベースに証明書を追加することができます。 秘密鍵と証明書がすでに存在する場合は、代わりにそれらを使用できます。 開発目的の場合、自己署名証明書のみを使用する必要があることに注意してください。

現行ディレクトリーへの自己署名の秘密鍵と公開証明書を作成するには、次のコマンドを実行します。

```
openssl req -newkey rsa:2048 -nodes -keyout tls.key -subj "/CN=localhost" -x509 -days 3650
-out tls.crt
```

ネイティブ HA で内部使用される TLS 秘密鍵と証明書の作成

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。 内部的に複製する際に使用するための自己署名証明書を作成できます。 開発目的の場合、自己署名証明書のみを使用する必要があることに注意してください。

現行ディレクトリーへの自己署名の秘密鍵と公開証明書を作成するには、次のコマンドを実行します。

```
openssl req -newkey rsa:2048 -nodes -keyout nativeha.key -subj "/CN=localhost" -x509 -days
3650 -out nativeha.crt
```

クライアントの鍵データベースへのキュー・マネージャーの公開鍵の追加

クライアント・アプリケーションのトラストストアとしてクライアントの鍵データベースを使用します。

クライアントの鍵データベースを作成します。

```
runmqakm -keydb -create -db clientkey.kdb -pw password -type cms -stash
```

前に生成した公開鍵をクライアントの鍵データベースに追加します。

```
runmqakm -cert -add -db clientkey.kdb -label mqservercert -file tls.crt -format ascii
-stashed
```

キュー・マネージャー・デプロイメントのための TLS 証明書を格納するシークレットの作成

キュー・マネージャーが鍵と証明書を参照して適用できるようにするために、上で作成したファイルを参照する Kubernetes TLS シークレットを作成します。 まず、このタスクの開始前に作成した名前空間で作業していることを確認してください。

```
oc create secret tls example-ha-secret --key="tls.key" --cert="tls.crt"
```

内部で使用するネイティブ HA TLS 証明書と鍵を格納するシークレットの作成

キュー・マネージャーが鍵と証明書を参照して適用できるようにするために、上で作成したファイルを参照する Kubernetes TLS シークレットを作成します。 まず、このタスクの開始前に作成した名前空間で作業していることを確認してください。

```
oc create secret tls example-ha-secret-internal --key="nativeha.key" --cert="nativeha.crt"
```

MQSC コマンドを組み込んだ構成マップの作成

MQSC コマンドを組み込んだ Kubernetes 構成マップを作成します。そのコマンドによって、新しいキューと SVRCONN チャネルを作成し、*nobody* というユーザーだけをブロックすることによって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。

この方法を使用するのは、開発のためだけに限定してください。

前に作成した名前空間で作業していることを確認してから (95 ページの『開始前に』を参照)、OCP UI またはコマンド・ラインで以下の YAML を入力します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-mi-configmap
data:
  tls.mqsc: |
    DEFINE QLOCAL('EXAMPLE.QUEUE') DEFPSIST(YES) REPLACE
    DEFINE CHANNEL(HAQMCHL) CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCAUTH(OPTIONAL)
SSLCIPH('ANY_TLS12_OR_HIGHER')
    SET CHLAUTH(HAQMCHL) TYPE(BLOCKUSER) USERLIST('nobody') ACTION(ADD)
```

ルーティングの構成

IBM MQ 9.2.1 以降で IBM MQ client またはツールキットを使用している場合は、キュー・マネージャー構成ファイル (INI ファイル) を使用して、キュー・マネージャーへのルーティングを構成できます。このファイル内で、チャネル名ではなくホスト名に基づいてルーティングするように *OutboundSNI* 変数を設定します。

コマンドを実行するディレクトリーに、 `mqclient.ini` という名前のファイルを作成します。これには、以下のテキストが正確に含まれます。

```
SSL:
   OutboundSNI=HOSTNAME
```

この INI ファイル内の値を変更しないでください。 例えば、ストリング HOSTNAME は変更してはなりません。

詳細については、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。

IBM MQ 9.2.1 より前の IBM MQ client またはツールキットを使用している場合は、前の構成ファイルの代わりに OCP 経路を作成する必要があります。 注 1: ルートの作成の手順に従ってください。

キュー・マネージャーのデプロイ

重要: この例では、*MQSNOAUT* 変数を使用してキュー・マネージャーでの許可を無効にします。これにより、TLS を使用してクライアントを接続するために必要なステップに集中できるようになります。 ただしこの方法は、IBM MQ の実動デプロイメントではお勧めできません。そのようにすると、接続する

すべてのアプリケーションが全管理権限を持つようになり、個々のアプリケーションのアクセス権を低くするメカニズムも存在しないからです。

以下の YAML をコピーして更新します。

- 正しいライセンスが指定されていることを確認します。 mq.ibm.com/v1beta1 のライセンス交付に関する参照資料を参照してください。 IBM Cloud Pak for Integration 2021.1.1 では、ライセンスは評価ライセンス L-RJON-BYRMYW でなければなりません
- false を true に変更して、ライセンスに同意します。

キュー・マネージャーのカスタム・リソース YAML は、以下のようになります。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: nativeha-example
spec:
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: Production
  queueManager:
    name: HAEXAMPLE
    availability:
      type: NativeHA
      tls:
        secretName: example-ha-secret-internal
    mqsc:
    - configMap:
        name: example-mi-configmap
        items:
        - tls.mqsc
  template:
    pod:
      containers:
        - env:
            - name: MQSNOAUT
              value: 'yes'
          name: qmgr
  version: 9.2.5.0-r3
  pki:
    keys:
      - name: example
        secret:
          secretName: example-ha-secret
          items:
          - tls.key
          - tls.crt
```

上述の手順で作成した名前空間で作業していることを確認し、Red Hat OpenShift Container Platform Web コンソール、コマンド・ライン、または IBM Cloud Pak for Integration Platform Navigator を使用して、更新した YAML をデプロイします。

システムがネイティブ HA キュー・マネージャーを構成している間、少しの遅延があります。その後、キュー・マネージャーを使用できるようになるはずです。

検証

このセクションでは、キュー・マネージャーが想定どおりに動作することを検証します。

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 nativeha-example
```

キュー・マネージャーへの接続のテスト

キュー・マネージャーで単方向の TLS 通信が構成されていることを確認するために、サンプル・アプリケーション **amqsputc** と **amqsgetc** を使用します。

キュー・マネージャーのホスト名の検索

ルート nativeha-example-ibm-mq-qm のキュー・マネージャー・ホスト名を見つけるには、以下のコマンドを実行します。 ホスト名は HOST フィールドに返されます。

```
oc get routes nativeha-example-ibm-mq-qm
```

キュー・マネージャーの詳細情報の指定

キュー・マネージャーの詳細を指定するファイル CCDT.JSON を作成します。 ホストの値は、直前のステップで返されたホスト名に置き換えてください。

```
{
    "channel":
    [
        {
            "name": "HAQMCHL",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "<host from previous step>",
                        "port": 443
                    }
                ],
                "queueManager": "HAEXAMPLE"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ECDHE_RSA_AES_128_CBC_SHA256"
            },
            "type": "clientConnection"
        }
   ]
}
```

環境変数のエクスポート

オペレーティング・システムに適した方法で以下の環境変数をエクスポートします。 これらの変数は **amqsputc** と **amqsgetc** によって読み取られます。

システム上のファイルへのパスを更新します。

```
export MQCCDTURL='<full_path_to_file>/CCDT.JSON'
export MQSSLKEYR='<full_path_to_file>/clientkey'
```

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。

```
amqsputc EXAMPLE.QUEUE HAEXAMPLE
```

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。

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。

```
amqsgetc EXAMPLE.QUEUE HAEXAMPLE
```

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。

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。

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=nativeha-example
```

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。

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 nativeha-example-ibm-mq-<value>
```

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=nativeha-example
```

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。

```
oc exec -t Pod -- dspmq -o nativeha -x -m HAEXAMPLE
```

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(HAEXAMPLE) 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)
```

メッセージの再度の書き込みと読み込み
スタンバイ・ポッドがアクティブ・ポッドになった後 (つまり、READY フィールドの値が 1/1 になった後) で、前述のように以下のコマンドを再び使用して、キュー・マネージャーにメッセージを渡して、その後にキュー・マネージャーからメッセージを取得します。

```
amqsputc EXAMPLE.QUEUE HAEXAMPLE
```

```
amqsgetc EXAMPLE.QUEUE HAEXAMPLE
```

これで成功です。ネイティブ HA キュー・マネージャーが正常にデプロイされ、ポッドの障害から自動的に復旧できることが分かりました。

補足情報

注 1: ルートの作成
IBM MQ 9.2.1 より前の IBM MQ client またはツールキットを使用している場合は、ルートを作成する必要があります。

このルートを作成するには、上述の手順で作成した名前空間で作業していることを確認してから (95 ページの『開始前に』を参照)、Red Hat OpenShift Container Platform Web コンソールまたはコマンド・ラインで以下の YAML を入力します。

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: example-mi-route
spec:
  host: hamqchl.chl.mq.ibm.com
  to:
    kind: Service
    name: nativeha-example-ibm-mq
  port:
    targetPort: 1414
```

```
  tls:
    termination: passthrough
```

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。 MQSC コマンドを含む構成マップで指定されたチャネル名を変更する場合は、ここでホスト・フィールドも変更する必要があります。また、 キュー・マネージャーの詳細を指定する CCDT.JSON ファイルで変更する必要もあります。 詳しくは、110 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。

注 2: デフォルト以外のストレージ・クラスの使用

この例では、Red Hat OpenShift Container Platform でデフォルトのストレージ・クラスが構成されていることを想定しているので、キュー・マネージャーのカスタム・リソース YAML ではストレージ情報は必要ありません。 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。

このストレージ・クラス名は、既存のストレージ・クラスの名前と厳密に一致している必要があります。 つまり、この名前はコマンド `oc get storageclass` によって返される名前と一致しなければなりません。 また、これは `ReadWriteMany` もサポートする必要があります。 詳しくは、11 ページの『IBM MQ Operator のストレージに関する考慮事項』を参照してください。

関連タスク

101 ページの『IBM MQ 認定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 認定コンテナーの場合、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。

CP4I V 9.2.2 CD *IBM MQ* 認定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 認定コンテナーの場合、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

重要 :

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・インスタンスで提供される情報が確定的なもので、レプリカ・ノードからの情報は古くなってい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。

以下の状況フィールドが、ネイティブ HA 構成状況の報告に使用されます。

ROLE
これは、現行インスタンス・ロールを指定します。これは、`Active`、`Replica`、または `Unknown` 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。

REPLADDR
キュー・マネージャー・インスタンスの複製アドレス。

CONNACTV
ノードがアクティブ・インスタンスに接続されているかどうかを示します。

BACKLOG
このインスタンスがどれだけ遅れているかを KB 数で示します。

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にはブランク)。

ALTTIME
この情報が最後に更新された時刻を示します (更新されたことがない場合にはブランク)。

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)
```

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)
```

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)
```

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)
```

- 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
```

キュー・マネージャー BOB のアクティブ・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(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)
```

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。これは、レプリカの 1 つで処理が遅れていることを示しています

```
QMNAME(BOB)            ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
 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(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(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(No)  BACKLOG(435)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

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)
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) 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)
```

関連資料

dspmq (キュー・マネージャーの表示) コマンド

95 ページの『例: ネイティブ HA キュー・マネージャーの構成』

この例は、IBM MQ Operator を使用して、ネイティブ高可用性機能を使用するキュー・マネージャーを Red Hat OpenShift Container Platform (OCP) にデプロイする方法を示しています。

CP4I CD V9.2.3 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、87 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。

CP4I ネイティブ *HA* キュー・マネージャーの終了
endmqm コマンドを使用して、ネイティブ HA グループの一部であるアクティブ・キュー・マネージャーまたはレプリカ・キュー・マネージャーを終了できます。

手順

- キュー・マネージャーのアクティブ・インスタンスを終了するには、この資料の「構成」セクションの「 ネイティブ HA キュー・マネージャーの終了 」を参照してください。

CP4I V9.2.2 CD *IBM Cloud Pak for Integration 2021.1.1 のネイティブ HA 機能の評価*
IBM Cloud Pak for Integration 2021.1.1 ネイティブ HA 評価期間が終了しました。 IBM MQ 9.2.3 以上で IBM MQ Operator 1.6 以上を使用して、 IBM Cloud Pak for Integration 2021.2.1 から入手可能な更新済みのネイティブ HA 機能を使用してください。

関連タスク
101 ページの『IBM MQ 認定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 認定コンテナーの場合、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。

関連資料
95 ページの『例: ネイティブ HA キュー・マネージャーの構成』
この例は、IBM MQ Operator を使用して、ネイティブ高可用性機能を使用するキュー・マネージャーを Red Hat OpenShift Container Platform (OCP) にデプロイする方法を示しています。

OpenShift CP4I 例: マルチ・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 Red Hat OpenShift Container Platform (OCP) にデプロイする方法を示します。 この例では、サンプル・クライアントとキュー・マネージャー間での単方向の TLS 通信も構成します。 この例では、ポッドの障害をシミュレートする前後にメッセージの書き込みと読み取りを行って構成の成功を確認する様子を示します。

開始前に

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 をインストールし、インストール済みの samp/bin ディレクトリーおよび bin ディレクトリーを パスに追加します。 この例で必要な **runmqakm**、**amqsputc**、**amqsgetc** の各アプリケーションはクライアントで提供されています。 以下のように IBM MQ client をインストールします。
  - Windows Linux Windows および Linux の場合: 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 https://ibm.biz/mq92redistclients から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/を参照してください。
- オペレーティング・システムに対応した OpenSSL ツールをインストールします。 この作業は、まだ秘密鍵と証明書がない場合に、キュー・マネージャー用の自己署名証明書を生成するために必要です。
- この例のための Red Hat 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。
- キュー・マネージャーで使用されるデフォルトのストレージ・クラスを OCP で構成します。 デフォルトのストレージ・クラスを設定せずにこのチュートリアルを最後まで行う場合は、注 2: デフォルト以外のストレージ・クラスの使用を参照してください。

本タスクについて

複数インスタンス・キュー・マネージャーでは、アクティブ・ポッドとスタンバイ・ポッド (いずれも Kubernetes ポッド) を使用します。 これらのポッドは、ちょうど 2 つのレプリカと Kubernetes 永続ボリューム一式で構成される Kubernetes ステートフル・セットの一部として稼働します。 複数インスタンス・キュー・マネージャーについて詳しくは、16 ページの『コンテナー内の IBM MQ の高可用性』を参照してください。

この例では、複数インスタンス・キュー・マネージャーを定義するカスタム・リソース YAML を示します。このキュー・マネージャーは、永続ストレージを使用し、TLS が構成されています。 キュー・マネージャーを OCP にデプロイした後、アクティブ・キュー・マネージャー・ポッドの障害をシミュレートします。自動復旧が行われた後、メッセージの書き込みと読み取りを行うことによって、障害発生後の復旧に成功したことが証明されます。

例

MQ サーバーの TLS 秘密鍵と証明書の作成

このセクションでは、キュー・マネージャーの自己署名証明書を作成する方法と、クライアントのトラストストアとして機能する鍵データベースに証明書を追加する方法について説明します。 秘密鍵と証明書がすでに存在する場合は、代わりにそれらを使用できます。 開発目的の場合、自己署名証明書のみを使用する必要があることに注意してください。

現行ディレクトリーへの自己署名の秘密鍵と公開証明書を作成するには、次のコマンドを実行します。

```
openssl req -newkey rsa:2048 -nodes -keyout tls.key -subj "/CN=localhost" -x509 -days 3650
-out tls.crt
```

クライアントの鍵データベースへのキュー・マネージャーの公開鍵の追加

クライアント・アプリケーションのトラストストアとしてクライアントの鍵データベースを使用します。

クライアントの鍵データベースを作成します。

```
runmqakm -keydb -create -db clientkey.kdb -pw password -type cms -stash
```

前に生成した公開鍵をクライアントの鍵データベースに追加します。

```
runmqakm -cert -add -db clientkey.kdb -label mqservercert -file tls.crt -format ascii
-stashed
```

キュー・マネージャー・デプロイメントのための TLS 証明書を格納するシークレットの作成

キュー・マネージャーが鍵と証明書を参照して適用できるようにするために、上で作成したファイルを参照する Kubernetes TLS シークレットを作成します。 まず、このタスクの開始前に作成した名前空間で作業していることを確認してください。

```
oc create secret tls example-mi-secret --key="tls.key" --cert="tls.crt"
```

MQSC コマンドを組み込んだ構成マップの作成

MQSC コマンドを組み込んだ Kubernetes 構成マップを作成します。そのコマンドによって、新しいキューと SVRCONN チャネルを作成し、*nobody* というユーザーだけをブロックすることによって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。

この方法を使用するのは、開発のためだけに限定してください。

前に作成した名前空間で作業していることを確認してから (104 ページの『開始前に』を参照)、OCP UI またはコマンド・ラインで以下の YAML を入力します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: example-mi-configmap
data:
  tls.mqsc: |
    DEFINE QLOCAL('EXAMPLE.QUEUE') DEFPSIST(YES) REPLACE
    DEFINE CHANNEL(MIQMCHL) CHLTYPE(SVRCONN) TRPTYPE(TCP) SSLCAUTH(OPTIONAL)
SSLCIPH('ANY_TLS12_OR_HIGHER')
    SET CHLAUTH(MIQMCHL) TYPE(BLOCKUSER) USERLIST('nobody') ACTION(ADD)
```

ルーティングの構成

IBM MQ 9.2.1 以降で IBM MQ client またはツールキットを使用している場合は、キュー・マネージャー構成ファイル (INI ファイル) を使用して、キュー・マネージャーへのルーティングを構成できます。このファイル内で、チャネル名ではなくホスト名に基づいてルーティングするように *OutboundSNI* 変数を設定します。

コマンドを実行するディレクトリー内に「mqclient.ini」という名前のファイルを作成し、以下のテキストを入力します。

```
#* Module Name: mqclient.ini                                          *#
#* Type       : IBM MQ MQI client configuration file                  *#
#  Function   : Define the configuration of a client                  *#
#*                                                                    *#
#*********************************************************************#
#* Notes      :                                                       *#
#* 1) This file defines the configuration of a client                 *#
#*                                                                    *#
#*********************************************************************#
SSL:
   OutboundSNI=HOSTNAME
```

注: このページ内の値を変更しないでください。 例えば、ストリング HOSTNAME はそのままにしておく必要があります。

詳細については、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。

IBM MQ 9.2.1 より前の IBM MQ client またはツールキットを使用している場合は、前の構成ファイルの代わりに OCP 経路を作成する必要があります。 注 1: ルートの作成の手順に従ってください。

キュー・マネージャーのデプロイ

重要: この例では、*MQSNOAUT* 変数を使用してキュー・マネージャーでの許可を無効にします。これにより、TLS を使用してクライアントを接続するために必要なステップに集中できるようになります。 ただしこの方法は、IBM MQ の実動デプロイメントではお勧めできません。そのようにすると、接続するすべてのアプリケーションが全管理権限を持つようになり、個々のアプリケーションのアクセス権を低くするメカニズムも存在しないからです。

以下の YAML をコピーして更新します。

- 正しいライセンスが指定されていることを確認します。 mq.ibm.com/v1beta1 のライセンス交付に関する参照資料を参照してください。
- false を true に変更して、ライセンスに同意します。
- IBM Cloud File Storage を使用している場合は、注 3: IBM Cloud File Storage の使用を参照してください。

キュー・マネージャーのカスタム・リソース YAML は、以下のようになります。

```
apiVersion: mq.ibm.com/v1beta1
kind: QueueManager
metadata:
  name: miexample
spec:
  license:
    accept: false
    license: L-RJON-C7QG3S
    use: NonProduction
  queueManager:
    name: MIEXAMPLE
    availability:
      type: MultiInstance
    mqsc:
    - configMap:
        name: example-mi-configmap
        items:
        - tls.mqsc
  template:
    pod:
      containers:
        - env:
            - name: MQSNOAUT
              value: 'yes'
          name: qmgr
  version: 9.2.5.0-r3
  web:
    enabled: true
  pki:
    keys:
      - name: example
        secret:
          secretName: example-mi-secret
          items:
          - tls.key
          - tls.crt
```

上述の手順で作成した名前空間で作業していることを確認し、OCP UI、コマンド・ライン、または IBM Cloud Pak for Integration Platform Navigator を使用して、更新した YAML をデプロイします。

検証

少し待つと、複数インスタンス・キュー・マネージャーが構成されて使用可能になるはずです。 このセクションでは、キュー・マネージャーが想定どおりに動作することを検証します。

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 miexample
```

キュー・マネージャーへの接続のテスト

キュー・マネージャーで単方向の TLS 通信が構成されていることを確認するために、サンプル・アプリケーション **amqsputc** と **amqsgetc** を使用します。

キュー・マネージャーのホスト名の検索

ルート miexample-ibm-mq-qm のキュー・マネージャー・ホスト名を見つけるには、以下のコマンドを実行します。 ホスト名は HOST フィールドに返されます。

```
oc get routes miexample-ibm-mq-qm
```

キュー・マネージャーの詳細情報の指定

キュー・マネージャーの詳細を指定するファイル CCDT.JSON を作成します。 ホストの値は、直前のステップで返されたホスト名に置き換えてください。

```
{
    "channel":
    [
        {
            "name": "MIQMCHL",
            "clientConnection":
            {
                "connection":
                [
                    {
                        "host": "<host from previous step>",
                        "port": 443
                    }
                ],
                "queueManager": "MIEXAMPLE"
            },
            "transmissionSecurity":
            {
              "cipherSpecification": "ECDHE_RSA_AES_128_CBC_SHA256"
            },
            "type": "clientConnection"
        }
   ]
}
```

環境変数のエクスポート

オペレーティング・システムに適した方法で以下の環境変数をエクスポートします。 これらの変数は **amqsputc** と **amqsgetc** によって読み取られます。

システム上のファイルへのパスを更新します。

```
export MQCCDTURL='<full_path_to_file>/CCDT.JSON'
export MQSSLKEYR='<full_path_to_file>/clientkey'
```

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。

```
amqsputc EXAMPLE.QUEUE MIEXAMPLE
```

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。

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。

```
amqsgetc EXAMPLE.QUEUE MIEXAMPLE
```

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。

アクティブ・ポッドの障害の強制試行

複数インスタンス・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。

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
```

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。

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 miexample-ibm-mq-<value>
```

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
```

スタンバイ・ポッドのログの表示
スタンバイ・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 logs miexample-ibm-mq-<value>
```

次のようなメッセージが表示されるはずです。

```
IBM MQ queue manager 'MIEXAMPLE' becoming the active instance.
```

メッセージの再度の書き込みと読み込み
スタンバイ・ポッドがアクティブ・ポッドになった後 (つまり、READY フィールドの値が 1/1 になった後) で、前述のように以下のコマンドを再び使用して、キュー・マネージャーにメッセージを渡して、その後にキュー・マネージャーからメッセージを取得します。

```
amqsputc EXAMPLE.QUEUE MIEXAMPLE
```

```
amqsgetc EXAMPLE.QUEUE MIEXAMPLE
```

これで成功です。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、ポッドの障害から自動的に復旧できることが分かりました。

補足情報

注 1: ルートの作成
IBM MQ 9.2.1 より前の IBM MQ client またはツールキットを使用している場合は、OCP 経路を作成する必要があります。

このルートを作成するには、上述の手順で作成した名前空間で作業していることを確認してから (104 ページの『開始前に』を参照)、OCP UI 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: route.openshift.io/v1
kind: Route
metadata:
  name: example-mi-route
spec:
  host: miqmchl.chl.mq.ibm.com
  to:
    kind: Service
    name: miexample-ibm-mq
  port:
    targetPort: 1414
  tls:
    termination: passthrough
```

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。 MQSC コマンドを含む構成マップで指定されたチャネル名を変更する場合は、ここでホスト・フィールドも変更する必要があります。また、 キュー・マネージャーの詳細を指定する CCDT.JSON ファイルで変更する必要もあります。 詳しくは、110 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。

注 2: デフォルト以外のストレージ・クラスの使用

この例では、OCP でデフォルトのストレージ・クラスが構成されていることを想定しているので、キュー・マネージャーのカスタム・リソース YAML ではストレージ情報は必要ありません。 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。

このストレージ・クラス名は、OCP システム上に存在するストレージ・クラスの名前と厳密に一致している必要があります。 つまり、この名前はコマンド `oc get storageclass` によって返される名前と一致しなければなりません。 また、これは `ReadWriteMany` もサポートする必要があります。 詳しくは、11 ページの『IBM MQ Operator のストレージに関する考慮事項』を参照してください。

注 3: IBM Cloud File Storage の使用

状況によっては (例えば、IBM Cloud File Storage を使用している場合などは)、**securityGroups** フィールドをキュー・マネージャー・カスタム・リソース YAML で指定する必要もあります。 例えば、`spec:` の直下に以下の子フィールドを追加します

```
securityContext:
  supplementalGroups: [99]
```

詳しくは、11 ページの『IBM MQ Operator のストレージに関する考慮事項』を参照してください。

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

重要 : この資料は、バージョン 9.2.1 以降の Continuous Delivery の IBM MQ クライアントに適用されます。 お客様がバージョン 9.2.0 Long Term Support のバージョンを使用している場合は、 IBM MQ 9.1 の資料ページ Red Hat OpenShift クラスターにデプロイされたキュー・マネージャーへの接続を参照してください。

V 9.2.1 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。

V 9.2.1 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。

V 9.2.1

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
- 管理対象モードの.NET クライアント （Long Term Support の IBM MQ 9.2.0 Fix Pack 4 の前 および Continuous Delivery の IBM MQ 9.2.3 の前です）

V 9.2.0.4 V 9.2.3

IBM MQ 9.2.0 Fix Pack 4 for Long Term Support および IBM MQ 9.2.3 for Continuous Delivery 以降、IBM MQ 管理対象 .NET クライアントが更新され、 **OutboundSNI** プロパティーが HOSTNAME に設定されている場合に SERVERNAME がそれぞれのホスト名に設定されるようになりました。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。 IBM MQ 9.2.0 Fix Pack 4 では、 **OutboundSNI** プロパティーが追加され、`mqclient.ini` ファイルからのみサポートされることに注意してください。.NET アプリケーションからこのプロパティーを設定することはできません。

V 9.2.5

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 がチャンネル名と SNI ヘッダを対応させるため、MQ チャンネル名は小文字で終わらないようにすることが重要です。

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: <provide a unique name for the Route>
    namespace: <the namespace of your MQ deployment>
  spec:
    host: <SNI address mapping for the channel>
    to:
      kind: Service
      name: <the name of the 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

117 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 への接続方法。

CP4I IBM Cloud Pak for Integration Operations Dashboard との統合

IBM Cloud Pak for Integration によるトランザクションのトレース機能は、Operations Dashboard によって提供されます。

このタスクについて

Operations Dashboard との統合を有効にすると、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。 この API 出口が、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 Operations Dashboard のデータ・ストアに送信します。

MQ クライアント・バインディングを使用して送信されるメッセージのみがトレースされることに注意してください。

IBM MQ Operator 1.5 より前のバージョンでは、トレースを有効にすると、キュー・マネージャーと一緒にデプロイされたトレース・エージェントとコレクターのイメージは常に入手可能な最新バージョンになっていましたので、最新バージョンの IBM Cloud Pak for Integration を使用していない場合には非互換性の問題が生じる可能性があることにも注意してください。

手順

1. 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、トレース機能は無効になっています。

IBM Cloud Pak for Integration Platform Navigator を使用してデプロイする場合は、「**トレースの有効化 (Enable Tracing)**」を「**オン (On)**」に設定し、「**トレース名前空間 (Tracing Namespace)**」を Operations Dashboard がインストールされている名前空間に設定して、デプロイ時にトレースを有効にすることができます。 キュー・マネージャーのデプロイ方法について詳しくは、83 ページの『IBM Cloud Pak for Integration Platform Navigator を使用したキュー・マネージャーのデプロイ』を参照してください。

Red Hat OpenShift CLI または Red Hat OpenShift Web コンソールを使用してデプロイする場合は、以下の YAML スニペットを使用してトレースを使用可能にすることができます。

```
spec:
  tracing:
    enabled: true
    namespace: <Operations_Dashboard_Namespace
```

重要: MQ を Operations Dashboard に登録する (次の手順を参照) まで、キュー・マネージャーは開始されません。

この機能を有効にすると、キュー・マネージャー・コンテナーに加えて、2 つのサイドカー・コンテナー (「エージェント」と「コレクター」) が実行されるようになります。 これらのサイドカー・コンテナーのイメージは、メインの MQ イメージと同じレジストリーにあり、使用するプル・ポリシーとプル・シークレットも同じです。 CPU とメモリーの制限を構成するための追加の設定を使用できます。

2. この名前空間にオペレーション・ダッシュボード統合を持つキュー・マネージャーが初めてデプロイされる場合は、オペレーション・ダッシュボードを使用して レジスター を実行する必要があります。

   登録すると、キュー・マネージャーのポッドの正常な始動に必要な Secret オブジェクトが作成されます。

CP4I CD *IBM MQ 9.2.2 または 9.2.3 を IBM Cloud Pak for Integration2021.4 のオペレーション・ダッシュボード統合してデプロイまたはアップグレードします*

各 IBM MQ バージョンは、キュー・マネージャーとともにデプロイされる特定バージョンの操作ダッシュボード・エージェント/コレクター・コンポーネントに関連付けられます。 IBM Cloud Pak for Integration 2021.4.1 では、古いエージェントとコレクターのコンポーネントが操作ダッシュボードで動作しないようにする変更が導入されました。 これを修正するには、IBM MQ 9.2.2 または 9.2.3 を使用するときに、使用するオペレーション・ダッシュボード・エージェントおよびコレクター・イメージのバージョンをオーバーライドする必要があります。

新規 IBM MQ 9.2.2 または 9.2.3 キュー・マネージャーのデプロイ

IBM Cloud Pak for Integration 2021.4.1 を IBM MQ 9.2.2 または 9.2.3 とともに使用する場合は、操作ダッシュボード・エージェント/コレクター・イメージを QueueManager YAML の 2.4 バージョンにオーバーライドする必要があります。 以下に例を示します。

```
spec:
  tracing:
    agent:
      image: cp.icr.io/cp/icp4i/od/icp4i-od-
agent@sha256:27a211f0f78eff765d1f9520e0f9841f902600bb556827477b206e209cb44d20
    collector:
      image: cp.icr.io/cp/icp4i/od/icp4i-od-
collector@sha256:dc70b1341b23dc72642ce68809811f9db0e8a0c46bda2508e8eb3d4035e04f4b
```

これを行わないと、QueueManager ポッドが Pending 状態のままになります。 IBM MQ 9.2.4 にアップグレードすると、これらのオーバーライドを削除することができます。

IBM Cloud Pak for Integration 2021.4.1 へのアップグレード

注 : IBM MQ 9.2.2 または 9.2.3 のキュー・マネージャーを保持する場合は、ステップ 3 を実行しないでください。

1. 前述のように、QueueManager を更新して、エージェント・イメージとコレクター・イメージをオーバーライドします。

2. 72 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』で説明されているように、オペレーション・ダッシュボードおよび IBM MQ オペレーターを含む IBM Cloud Pak for Integration オペレーターをアップグレードしてください。
3. (オプション) IBM MQ 9.2.4 以降にアップグレードするには、QueueManager を更新して、.spec.version をご使用のバージョンの IBM MQ に使用するようにし、エージェント・イメージおよびコレクター・イメージのオーバーライドを排除します。

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 のタイトル・レジストリー用の Red Hat OpenShift 秘密鍵がない場合は、80 ページの『IBM MQ 用の Red Hat OpenShift プロジェクトの準備』のステップに従ってください。

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:9.2.5.0-r3
         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:9.2.5.0-r3"
         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、20 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。

82 ページの『Red Hat OpenShift Container Platform クラスターにキュー・マネージャーをデプロイする方法』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

82 ページの『Red Hat OpenShift Container Platform クラスターにキュー・マネージャーをデプロイする方法』
QueueManager カスタム・リソースを使用して Red Hat OpenShift Container Platform クラスターにキュー・マネージャーをデプロイできます。

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。

実行時チェックのサポートは IBM MQ Operator 1.2 で導入されました。

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 CP4I IBM MQ Operator を使用した IBM MQ

このタスクについて

手順

- 80 ページの『IBM MQ 用の Red Hat OpenShift プロジェクトの準備』.
- 82 ページの『Red Hat OpenShift Container Platform クラスターにキュー・マネージャーをデプロイする方法』.

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 への接続方法。

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 Navigator 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}'
```

IBM Cloud Pak for Integration ライセンスを使用している場合、IBM MQ ウェブコンソールは、IBM Cloud PakID およびアクセス・マネージャー (IAM) を使用するように構成されています。 IAM コンポーネントは、クラスター管理者によって既にセットアップされている場合があります。 ただし、これが Red Hat OpenShift クラスターで初めて IAM が使用された場合は、初期管理パスワードを取得する必要があります。詳しくは、Getting the initial admin password を参照してください。

IBM MQ ライセンスを使用している場合、MQ Web コンソールは事前構成されておらず、ユーザー自身で構成する必要があります。 詳しくは、ユーザーおよび役割の構成を参照してください。

関連タスク

110 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 Cloud Pak IAM を使用した IBM MQ Console の許可の付与*

IBM MQ Console の許可は、 IBM Cloud Pak for Integration Platform Navigator ではなく、 IBM Cloud Pak 管理ハブを介して管理されます。 IBM MQ は、 IBM Cloud Pak for Integration によって提供される「自動化」権限を使用せず、代わりに IBM Cloud Pak Identity and Access Manager (IAM) によって有効化される基本的な権限を使用します。

手順

1. IBM Cloud Pak 管理コンソールを開きます。

   IBM Cloud Pak for Integration Platform UI で、ツールバーの右上隅にある Cloud Pak スイッチャー (9 ドット・アイコン) をクリックし、 **IBM Cloud** 「**Pak 管理**」 パネルをクリックします。
2. 左上隅のナビゲーション・メニューで、 「**ID およびアクセス (Identity and access)**」を選択し、 「**チームおよびサービス ID (Teams and services IDs)**」を選択します。

3. チームを作成し、それにユーザーを追加します。
   a)「**チームの作成**」を選択します。
   b) チーム名を入力し、管理するユーザーのセキュリティー・ドメインを選択します。
   c) ユーザーを検索します。
      これらのユーザーは、ID プロバイダーに既に存在している必要があります。
   d) 各ユーザーを見つけたら、それらのユーザーに役割を付与します。 IBM MQ Console を使用して IBM MQ を管理するには、「Administrator」または「Cluster Administrator」でなければなりません。
4. 各ユーザーを名前空間に追加します。
   a) 編集するチームを選択します。
   b)「**リソース**」 > 「**リソースの管理**」を選択します。
   c) このチームに管理させる名前空間を選択します。 これらは、キュー・マネージャーを持つ任意の名前空間にすることができます。

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。

これらのメトリックは、 Red Hat OpenShift Container Platform (OCP) モニター・スタックを使用して表示できます。 OCP の 「**メトリック**」 タブを開き、 「**監視**」 > 「**メトリック**」をクリックします。 キュー・マネージャーのメトリックはデフォルトで有効になっていますが、 **.spec.metrics.enabled** を `false` に設定することで無効にすることができます。

Prometheus は、データベースと規則を評価してメトリックを時系列で取得するエンジンです。 Prometheus は、IBM MQ コンテナーで公開されるメトリック・エンドポイントを利用して照会を行うことができます。 MQ システム・トピックから、モニタリングとアクティビティー・トレースを行うためのメトリックが生成されます。

Red Hat 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、Red Hat 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。 詳しくは、Enabling monitoring for user-defined projects を参照してください。 IBM MQ Operator によって `ServiceMonitor` が作成されるのは、メトリックを有効にして `QueueManager` を作成するときです。 Prometheus オペレーターはこれをディスカバーできます。

旧バージョンの IBM Cloud Pak for Integration では、代わりに IBM Cloud Platform Monitoring サービスを使用して Prometheus サーバーを提供することもできます。

OpenShift CP4I *IBM MQ Operator* の使用時にパブリッシュされるメトリック

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_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
`ibmmq_qmgr_cpu_load_one_minute_average_percentage`	`gauge`	CPU 負荷 - 1 分間の平均

メトリック	タイプ	説明
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
ibmmq_qmgr_failed_mqput_total	counter	失敗した MQPUT の数

メトリック	タイプ	説明
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_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_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
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

メトリック	タイプ	説明
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
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

メトリック	タイプ	説明
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
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

メトリック	タイプ	説明
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 - 使用中のバイト数
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
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

CP4I V 9.2.2 CD

IBM MQ 認定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 認定コンテナーの場合、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

重要 :

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・インスタンスで提供される情報が確定的なもので、レプリカ・ノードからの情報は古くなってい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。

以下の状況フィールドが、ネイティブ HA 構成状況の報告に使用されます。

ROLE
これは、現行インスタンス・ロールを指定します。これは、Active、Replica、または Unknown 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。

REPLADDR
キュー・マネージャー・インスタンスの複製アドレス。

CONNACTV
ノードがアクティブ・インスタンスに接続されているかどうかを示します。

BACKLOG
このインスタンスがどれだけ遅れているかを KB 数で示します。

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にはブランク)。

ALTTIME
この情報が最後に更新された時刻を示します (更新されたことがない場合にはブランク)。

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)
```

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)
```

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)
```

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)
```

- 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
```

キュー・マネージャー BOB のアクティブ・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(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)
```

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。これは、レプリカの 1 つで処理が遅れていることを示しています

```
QMNAME(BOB)            ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
 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(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(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(No)  BACKLOG(435)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

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)
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) 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)
```

関連資料

dspmq (キュー・マネージャーの表示) コマンド

95 ページの『例: ネイティブ HA キュー・マネージャーの構成』

この例は、IBM MQ Operator を使用して、ネイティブ高可用性機能を使用するキュー・マネージャーを Red Hat OpenShift Container Platform (OCP) にデプロイする方法を示しています。

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

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。

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
  ```

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

- 127 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 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、ポッド内でリモート・シェルを開きます。

   ```
   ls
   ```

c) マウントされた PVC と同じ名前のディレクトリーが表示されます。 これらのディレクトリーを参照して、キュー・マネージャー PVC 上のファイルにアクセスします。 PVC のリストを表示するには、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 IBM MQ Operator の API リファレンス

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 mq.ibm.com/v1beta1 の API リファレンス

v1beta1 API を使用して、QueueManager リソースを作成および管理できます。

OpenShift CP4I CD EUS *mq.ibm.com/v1beta1* のライセンスのリファレンス

現行バージョンのライセンス

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-RJON-C7QG3S	Production または NonProduction	IBM Cloud Pak for Integration 2021.4.1	9.2.4 または 9.2.5
L-RJON-C7QFZX	Production または NonProduction	IBM Cloud Pak for Integration リミテッド・エディション 2021.4.1	9.2.4 または 9.2.5
L-RJON-C5CSNH	Production または NonProduction	IBM Cloud Pak for Integration 2021.3.1	9.2.3 または 9.2.4
L-RJON-C5CSM2	Production または NonProduction	IBM Cloud Pak for Integration リミテッド・エディション 2021.3.1	9.2.3 または 9.2.4
L-RJON-BZFQU2	Production または NonProduction	IBM Cloud Pak for Integration 2021.2.1	9.2.3
L-RJON-BZFQSB	Production または NonProduction	IBM Cloud Pak for Integration 限定版 2021.2.1	9.2.3
L-RJON-BUVMQX	Production または NonProduction	IBM Cloud Pak for Integration 2020.4.1	9.2.0 EUS または 9.2.1

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-RJON-BUVMYB	Production または NonProduction	IBM Cloud Pak for Integration 限定版 2020.4.1	9.2.0 EUS または 9.2.1
L-APIG-BZDDDY	Production	IBM MQ Advanced および IBM MQ Advanced for Non-Production Environment 9.2 - 2021/07	9.2.3、9.2.4、または 9.2.5
L-APIG-BYHCL7	Development	IBM MQ Advanced for Developers (保証適用外) V9.2 - 2021/07	9.2.3、9.2.4、または 9.2.5
L-APIG-BVJJB3	Production	IBM MQ Advanced および IBM MQ Advanced for Non-Production Environment 9.2 - 2021/03	9.2.2
L-APIG-BMJJBM	Production	IBM MQ Advanced V9.2	9.2.0 CD または 9.2.1
L-APIG-BMKG5H	Development	IBM MQ Advanced for Developers (保証適用外) V9.2	9.2.0 CD、9.2.1 または 9.2.2

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。

古いバージョンのライセンス

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-RJON-BXUPZ2	Production または NonProduction	IBM Cloud Pak for Integration 2021.1.1	9.2.2
L-RJON-BXUQ34	Production または NonProduction	IBM Cloud Pak for Integration 限定版 2021.1.1	9.2.2
L-RJON-BYRMYW	NonProduction	IBM Cloud Pak for Integration Eval-Demo 2021.1.1。 IBM MQ Operator 1.5 のみを使用する ネイティブ HA で使用するための早期リリース。	9.2.2
L-RJON-BQPGWD	Production または NonProduction	IBM Cloud Pak for Integration 2020.3.1	9.2.0 CD
L-RJON-BN7PN3	Production または NonProduction	IBM Cloud Pak for Integration 2020.2.1	9.1.5 または 9.2.0 CD
L-RJON-BPHL2Y	Production または NonProduction	IBM Cloud Pak for Integration 限定版 2020.2.1	9.1.5
L-APIG-BJAKBF	Production	IBM MQ Advanced V9.1 - 2020/04	9.1.5

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-APIG-BM7GDH	Development	IBM MQ Advanced for Developers (保証適用外) V9.1 - 2020/04	9.1.5

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。

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。

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	
specQueueManagerSpec	QueueManager について必要とする状態。
statusQueueManagerStatus	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:

- 130 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。
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。

フィールド	説明
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。
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:

- 130 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:

- 130 ページの『.spec』

フィールド	説明
name 文字列	参照の名前。 詳細情報：https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#namesTODO: 他の有用なフィールドを追加します。apiVersion, kind, uid?

.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:

- 130 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:

- 130 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。 有効な値については、http://ibm.biz/BdqvCF を参照してください。
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 などがあります。 MQ ライセンスを使用する場合のデフォルトは ProcessorValueUnit、Cloud Pak for Integration ライセンスを使用する場合のデフォルトは VirtualProcessorCore です。
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。 有効な値については、http://ibm.biz/BdqvCF を参照してください。

.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:

- 130 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。

フィールド	説明
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:

- 132 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:

- 133 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:

- 132 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:

- 133 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:

- 130 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。
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)。 このフィールドは、Kubernetes の命名規則に準拠していないキュー・マネージャー名 (大文字が含まれる名前など) を使用する場合に使用します。
readinessProbe QueueManagerReadinessProbe	Readiness プローブを制御する設定。
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:

- 134 ページの『.spec.queueManager』

フィールド	説明
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。 一方がアクティブなキュー・マネージャーで、もう一方がスタンバイであるポッドのペアには、MultiInstance を使用します。 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。 詳しくは、http://ibm.biz/BdqAQa を参照してください。
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.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:

- 134 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.ini

INI 構成ファイルのソース。

以下の中に含まれます:

- 134 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:

- 135 ページの『.spec.queueManager.ini』

フィールド	説明
items 配列	適用する必要がある、Kubernetes ソース内部の鍵。

フィールド	説明
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:

- 135 ページの『.spec.queueManager.ini』

フィールド	説明
items 配列	適用する必要がある、Kubernetes ソース内部の鍵。
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:

- 134 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。
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:

- 134 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。

.spec.queueManager.mqsc

MQSC 構成ファイルのソース。

以下の中に含まれます:

- 134 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:

- 136 ページの『.spec.queueManager.mqsc』

フィールド	説明
items 配列	適用する必要がある、Kubernetes ソース内部の鍵。
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:

- 136 ページの『.spec.queueManager.mqsc』

フィールド	説明
items 配列	適用する必要がある、Kubernetes ソース内部の鍵。
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:

- 134 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.resources

リソース要件を制御する設定。

以下の中に含まれます:

- 134 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:

- 138 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:

- 138 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:

- 134 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:

- 134 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 60 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:

- 134 ページの『.spec.queueManager』

フィールド	説明
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。

.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:

- 139 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。 例えば、2Gi などです。 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 である場合にのみ有効です。
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:

- 139 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。 例えば、2Gi などです。 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 である場合にのみ有効です。
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:

- 139 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。 例えば、2Gi などです。 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 である場合にのみ有効です。
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.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:

- 130 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 OR で rw-rw ---- 設定されていない場合、 Kubelet はボリュームの所有権と許可を変更しません。
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。

フィールド	説明
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.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:

- 130 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:

- 130 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:

- 142 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。

フィールド	説明
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:

- 142 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:

- 142 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:

- 142 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:

- 143 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.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:

- 143 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:

- 130 ページの『.spec』

フィールド	説明
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。

. ステータス

QueueManager について観測された状態。

以下の中に含まれます:

- 130 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。
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)。
name 文字列	キュー・マネージャーの名前です。
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:

- 145 ページの『. ステータス』

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:

- 145 ページの『. ステータス』

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。
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:

- 145 ページの『. ステータス』

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.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:

- 145 ページの『. ステータス』

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:

- 146 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:

- 146 ページの『.status.versions.available』

フィールド	説明
name 文字列	このバージョンの QueueManager を表すバージョンの 名前。 これは、spec.version フィールドで有効な値です。

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。

状態のサポートは IBM MQ Operator 1.2 で導入されました。

QueueManager リソースでは以下の状態が定義されています。

表 1. 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[7]	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。
	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	[8] 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	EUSSupport	[9]EUS 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限定されてい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	EUSSupport	[10]EUS オペランド *<mq_version>* がインストールされましたが、 OCP バージョン 4*<ocp_version>* は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していません。
	警告	EUSSupport	[11]EUS オペランド *<mq_version>* がインストールされましたが、 OCP バージョン *<ocp_version>* は拡張サポート期間に適格ではありません。 このオペランドは通常の CD リリースでサポートされています。

[7] 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、 MQ ウェブコンソールが使用可能になっている場合、 `OidcPending` 条件は、 OIDC クライアント登録が IAM で完了するのを待っている間に、キュー・マネージャーの状況をログに記録します。
[8] オペレーター 1.4.0 以降
[9] オペレーター 1.4.0 以降

表 1. 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 [12]	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 [13]	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
	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

Multi 独自の IBM MQ コンテナーおよびデプロイメント・コードのビルド

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、プリパッケージされている IBM 提供のコンテナーのいずれかを代わりに使用できないか検討してください。 コンテナー内の IBM MQ を参照してください。

このタスクについて

IBM MQ をコンテナー・イメージとしてパッケージすると、アプリケーションに対する変更をテスト・システムやステージング・システムに素早く簡単にデプロイできます。 これは、企業の継続的デリバリーに大きな利点をもたらします。

手順

- 150 ページの『コンテナーを使用する独自の IBM MQ キュー・マネージャー・イメージの計画』
- 150 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』
- 153 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』

関連概念

コンテナー内の IBM MQ

---

10 オペレーター 1.4.0 以降

11 オペレーター 1.3.0 のみ

12 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。

13 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。 ストレージの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も Failed に設定されます。

Multi コンテナーを使用する独自の IBM MQ キュー・マネージャー・イメージの計画

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。

これを解決するには、コンテナーの開始時に、 **crtmqdir** コマンドを使用して、初めて実行するときに `/var/mqm` にデータを取り込むことができます。

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。

Multi 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、`/var/mqm` の内容がホスト・ファイル・システム上のコンテナー固有のボリュームから提供されるファイル・システムとして、このイメージ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。

– 151 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』
– 151 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』

Multi 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。 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、「インストールしない」 IBM MQ パッケージを使用する必要があります。

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。

Multi 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

V 9.2.0 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。 IBM MQ 9.2 では、"インストールなし" の IBM MQ パッケージを使用すると、`mqm` ユーザーも `mqm` グループも存在しません。

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。

Multi 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

Docker のコンテナー間でプロセスの名前空間を共有すると、IBM MQ キュー・マネージャーとは別のコンテナー内の IBM MQ にローカル・バインディング接続する必要があ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

この機能は、IBM MQ 9.0.3 以降のキュー・マネージャーでサポートされます。

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。

Docker が既にインストールされている Linux システムで以下のステップを実行することにより、この機能を試すことができます。

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、154 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。

Multi 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、153 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, 2024.
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
```

CP4I 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。

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)]」という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)
```

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 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

5 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。

CP4I カスタムビルト・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

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

dspmq コマンドを使用して、ノード上のキュー・マネージャー・インスタンスの操作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・インスタンスで提供される情報が確定的なもので、レプリカ・ノードからの情報は古くなってい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。

以下の状況フィールドが、ネイティブ HA 構成状況の報告に使用されます。

ROLE
これは、現行インスタンス・ロールを指定します。これは、`Active`、`Replica`、または `Unknown` 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。

REPLADDR
キュー・マネージャー・インスタンスの複製アドレス。

CONNACTV
ノードがアクティブ・インスタンスに接続されているかどうかを示します。

BACKLOG
このインスタンスがどれだけ遅れているかを KB 数で示します。

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にはブランク)。

ALTTIME
この情報が最後に更新された時刻を示します (更新されたことがない場合にはブランク)。

手順

- 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

  ```
  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)
  ```

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)
```

- 現行ノード上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、以下のようにします。

```
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)
```

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)
```

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)
```

- 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

```
dspmq -o nativeha -x -m QMgrName
```

キュー・マネージャー BOB のアクティブ・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(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)
```

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。これは、レプリカの 1 つで処理が遅れていることを示しています

```
QMNAME(BOB)            ROLE(Replica) INSTANCE(inst2) INSYNC(Yes) QUORUM(2/3)
 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(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(inst3) ROLE(Replica) REPLADDR(9.20.123.47) CONNACTV(Yes) INSYNC(No)  BACKLOG(435)
CONNINST(Yes) ALTDATE(2022-01-12) ALTTIME(12.03.44)
```

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)
 INSTANCE(inst1) ROLE(Unknown) REPLADDR(9.20.123.45) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst2) ROLE(Unknown) REPLADDR(9.20.123.46) CONNACTV(Unknown) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) ALTDATE() ALTTIME()
 INSTANCE(inst3) ROLE(Unknown) REPLADDR(9.20.123.47) CONNACTV(No) INSYNC(Unknown)
BACKLOG(Unknown) CONNINST(No) 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)
```

関連資料
dspmq

CP4I ネイティブ HA キュー・マネージャーの終了

endmqm コマンドを使用して、ネイティブ HA グループの一部であるアクティブ・キュー・マネージャーまたはレプリカ・キュー・マネージャーを終了できます。

手順

- キュー・マネージャーのアクティブ・インスタンスを終了するには、この資料の「構成」セクションの「ネイティブ HA キュー・マネージャーの終了」を参照してください。

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。

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
103-8510
東京 103-8510、日本

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.

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。これらの名前はすべて架空のものであり、名前や住所が類似する個人や企業が実在して 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 WebSphere MQ のサービスを使用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 の米国およびその他の国における商標です。

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: