9.1

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、53 ページの『特記事項』に記載されている情報をお読みください。

本書は、 IBM® MQ バージョン 9 リリース 1、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

目次

Linux コンテナー内の IBM MQ

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。

IBM MQ は、IBM MQ Advanced および IBM MQ Advanced for Developers で提供されているプリパッケージ・コンテナーで実行することができます。 この IBM MQ Advanced certified container はサポートされる画像および Helm チャートを提供し、実働対応 IBM MQ イメージを Red Hat® OpenShift®、 IBM Cloud® Private、または IBM Cloud Kubernetes Service にデプロイするのに使用できます。

また、IBM MQ を IBM Cloud Pak® for Integration コンテナーや、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 Advanced certified container について詳しくは、以下のリンクを参照してください。

Linux MQ Adv. CD 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

コンテナー内の IBM MQ のアーキテクチャーについて計画する前に、IBM MQ の基本概念 (IBM MQ の技術概要を参照) と Kubernetes/OpenShift の基本概念 (OpenShift Container Platform architecture を参照) の両方についてよく理解しておく必要があります。

手順

- 5 ページの『コンテナー内の IBM MQ の使用方法の選択』.
- 10 ページの『IBM MQ Advanced certified container の高可用性』.
- 12 ページの『IBM MQ Advanced certified container のユーザー認証と許可』.

Linux MQ Adv. CD 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認定コンテナーを使用することも、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 Advanced 認定コンテナーを使用する

Red Hat OpenShift Container Platform にデプロイする場合は、おそらく、認定コンテナーを使用したいはずです。 認定コンテナーには、以下の 3 種類があります。

- IBM Cloud Pak for Integration の場合は IBM MQ Advanced certified container 。 これは、認定コンテナー・バージョンが含まれている、別個の IBM 製品です。
- IBM MQ Advanced certified container
- IBM MQ Advanced for Developers 認定コンテナー (保証なし)

IBM MQ 9.1.4 および CD の旧リリースは、IBM Cloud Private および IBM Cloud Kubernetes Service でもサポートされていました。

認定コンテナーは速いペースで展開されているので、Continuous Delivery リリースでしかサポートされないことに注意してください。

認定コンテナーには、事前作成されたコンテナー・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 9.1.5 以降では、キュー・マネージャーは IBM MQ オペレーターを使用して管理されます。 以前のバージョンの IBM MQ(バージョン 9.1.5 まで) は、 Helm チャートを使用して管理されます。

IBM MQ の機能の中には、認定コンテナー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、認証済みコンテナーを使用して、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・マネージャーを提供することができます。
  - 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)
- Helm チャートを使用してデプロイする場合は Web サーバーを使用します (IBM Cloud Pak for Integration を除く)。
- **crtmqm**、**strmqm**、**endmqm** で使用するオプションをカスタマイズする (リカバリー・ログの構成など)

独自のイメージとチャートのビルド

これは最も柔軟なコンテナソリューションですが、コンテナの設定に強いスキルが必要であり、結果としてのコンテナを"所有する"必要があります。 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。 34 ページの『独自の IBM MQ コンテナーのビルド』を参照してください。 認定コンテナーの一部として提供される Helm チャートが GitHub に公開されており、独自イメージを作成する際に参考にするサンプルとして使用できます。

- IBM MQ Advanced certified container のための Helm チャート
- IBM MQ Advanced for Developers 認定コンテナーのための Helm チャート

関連概念

6 ページの『IBM MQ 認定コンテナーのサポート』
IBM MQ 認定コンテナーは、Kubernetes 環境でのみサポートされます。

8 ページの『独自の IBM MQ コンテナー・イメージとチャートのビルドのサポート』
Linux システムでコンテナーを使用する場合の考慮点。

Linux IBM MQ 認定コンテナーのサポート

IBM MQ 認定コンテナーは、Kubernetes 環境でのみサポートされます。

V 9.1.4 MQ Adv. CD CD リリース V9.1.4 以降では、IBM MQ Advanced certified container を Red Hat OpenShift と一緒に使用できます。 25 ページの『Helm CLI 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。

V9.1.4 より前の CD リリースは、以下の Kubernetes 環境でサポートされていました。

- IBM Cloud Kubernetes Service
- IBM Cloud Private
- IBM Cloud Private で Red Hat OpenShift

Kubernetes の特定のサポート対象バージョンについては、ダウンロードした IBM MQ Advanced Helm チャート内のファイル `qualification.yaml` および `Chart.yaml` を参照してください。 対象のバージョンはリリースごとに異なります。

IBM MQ Advanced certified container は、 IBM MQ オペレーターを使用してデプロイした場合、または以下のいずれかの Helm チャートを使用する場合にのみサポートされます。

- `ibm-mqadvanced-server-prod`
- `ibm-mqadvanced-server-integration-prod` in the IBM Cloud Pak for Integration

注 : Helm チャートの使用は、 IBM MQ Operator のリリースに従って非推奨になりました。

コンテナー技術は急速な進歩を遂げているため、IBM MQ Advanced certified container がサポートされるのは、リリース時点でこのチャートがサポートしている最新バージョンのプラットフォーム上のみです。古いプラットフォーム・バージョンを使用する場合は、古いバージョンの IBM MQ Advanced certified container を使用しなければならないことがあります。

IBM MQ Advanced certified container イメージは、 IBM MQ Continuous Delivery (CD) リリースに基づいています。 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。 IBM MQ の Long Term Support リリースは、認定コンテナーとして使用できません。

IBM MQ Advanced certified container V4.0 以降、このイメージは Red Hat Universal Base Image (UBI) への IBM MQ のインストールを提供します。これには、 IBM MQ によって使用される主要な Linux ライブラリーおよびユーティリティーが含まれています。 UBI は、Red Hat Enterprise Linux ホストで実行される場合、 Red Hat によってサポートされます。 以前のバージョンの IBM MQ Advanced certified container では、サポートされていない Ubuntu ベース・イメージが使用されていました。

関連概念
8 ページの『独自の IBM MQ コンテナー・イメージとチャートのビルドのサポート』
Linux システムでコンテナーを使用する場合の考慮点。

Linux MQ Adv. CD IBM MQ Advanced certified container のバージョン・サポート

IBM MQ Advanced certified container、IBM MQ、 IBM Cloud Kubernetes Service、IBM Cloud Pak for Integration、および IBM Cloud Private のサポート対象バージョン間の対応関係を示す表。

オペレーター IBM MQ

V 9.1.5

IBM MQ Operator は、 IBM Cloud Pak for Integration バージョン 2020.2 の一部として使用することも、IBM MQ バージョン 9.1.5 以上で単独で使用することもできます。

IBM MQ Operator は、 Red Hat OpenShift Container Platform バージョン 4.4 以上でサポートされます。

IBM MQ Advanced certified container V 9.1.5 (Helm チャート)-非推奨

Helm チャート `ibm-mqadvanced-server-prod` が含まれています。

V 9.1.5 IBM MQ Advanced certified container V5.0.x 以降、Helm のチャート、イメージ、および修正は、IBM Entitled Catalog and Registry を通じて出荷されます。 旧バージョンは、Passport Advantage® によって出荷されていました。 IBM Fix Central から修正リリースを入手可能です。

表 *1. IBM MQ Advanced certified container* のサポート

バージョン	IBM MQ バージョン	サポート終了	サポートされているオペレーティング・システム
6.0.x	9.1.5 継続的デリバリー・リリース	2021 年 3 月	詳細なシステム要件

表 1. *IBM MQ Advanced certified container* のサポート (続き)			
バージョン	**IBM MQ バージョン**	**サポート終了**	**サポートされているオペレーティング・システム**
5.0.x	9.1.4 継続的デリバリー・リリース	2020 年 12 月	詳細なシステム要件
4.1.x	9.1.3 継続的デリバリー・リリース	2020 年 7 月	詳細なシステム要件

IBM Cloud Pak for Integration V 9.1.5 (Helm チャート)-非推奨の IBM MQ Advanced certified container ソフトウェア

Helm チャート `ibm-mqadvanced-server-integration-prod` が含まれています。

表 2. *IBM Cloud Pak for Integration* の *IBM MQ Advanced certified container* ソフトウェアのバージョン・サポート		
バージョン	**IBM MQ バージョン**	**IBM Cloud Pak for Integration バージョン**
6.0.x	9.1.4 継続的デリバリー・リリース	2020.1.1 (システム要件)
5.0.x	9.1.3 継続的デリバリー・リリース	2019.4.1 (システム要件)
4.1.x	9.1.3 継続的デリバリー・リリース	2019.3.2.2 (システム要件)
4.0.x	9.1.3 継続的デリバリー・リリース	2019.3.2 (システム要件)
3.0.x	9.1.3 継続的デリバリー・リリース	2019.3.1 (システム要件)

サポートされるバージョンの情報については、IBM Cloud Pak for Integration リリース・ノートを参照してください。

Linux 独自の IBM MQ コンテナー・イメージとチャートのビルドのサポート

Linux システムでコンテナーを使用する場合の考慮点。

- コンテナー・イメージによって使用されるベース・イメージは、サポートされている Linux オペレーティング・システムを使用する必要があります。
- IBM MQ インストーラーを使用して、コンテナー・イメージ内の製品をインストールする必要があります。
- サポートされるパッケージのリストについては、Linux システム用の IBM MQ rpm コンポーネントを参照してください。
- V 9.1.0 以下のパッケージはサポートされません。
  - `MQSeriesBCBridge`
  - `MQSeriesRDQM`
- キュー・マネージャーのデータ・ディレクトリー (デフォルトで `/var/mqm`) は、永続的な状態を保持するコンテナー・ボリューム上に保管する必要があります。

  重要 : union ファイル・システムは使用できません。

ホスト・ディレクトリーをデータ・ボリュームとしてマウントするか、またはデータ・ボリューム・コンテナーを使用する必要があります。 詳しくは、Manage data in containers を参照してください。

- コンテナー内で IBM MQ 制御コマンド (**endmqm** など) を実行できる必要があります。
- 診断で使用するために、コンテナー内からファイルやディレクトリーを取得できる必要があります。
- V 9.1.0 別のコンテナーで実行されているキュー・マネージャーにアプリケーションをローカル・バインドするために、名前空間操作を使用して、キュー・マネージャーのコンテナーの名前空間を他のコンテナーと共有することができます。 詳しくは、37 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。

関連概念

6 ページの『IBM MQ 認定コンテナーのサポート』
IBM MQ 認定コンテナーは、Kubernetes 環境でのみサポートされます。

V 9.1.5 Linux MQ Adv. CD IBM MQ Advanced certified container のストレージに関する考慮事項

IBM MQ Advanced certified container 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すべてのコンテナー状態を破棄できる場合に使用します。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する場合は、再始動時に、すべてのキュー・マネージャーの状態が失われます。 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破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、21 ページの『OpenShift CLI を使用したカスタムの MQSC および INI ファイルによるイメージのビルド』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、10 ページの『IBM MQ Advanced certified container 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:
        enabled: true
        class: ibmc-file-gold-gid
          persistedData:
            enabled: true
            class: ibmc-file-gold-gid
          recoveryLogs:
            enabled: true
            class: ibmc-file-gold-gid
     securityContext:
       supplementalGroups: [99]
```

Linux MQ Adv. CD IBM MQ Advanced certified container の高可用性

IBM MQ Advanced certified container で高可用性を使用する場合、**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用ファイル・システムを使用するアクティブとスタンバイのペア) と**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) という 2 つの主な選択肢があります。

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。

複数インスタンス・キュー・マネージャー

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

複数インスタンス・キュー・マネージャーには、永続ボリュームへの同時アクセスを可能にするために、**アクティブな**ポッドと**スタンバイ状態の**ポッドの両方が必要です。 これを構成するには、 **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. 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 Helm チャートで構成可能です。
4. IBM MQ クライアントが再接続するまでにかかる時間。

シングル・レジリエント・キュー・マネージャー

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

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

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

1. Liveness プローブの実行にかかる時間、および Liveness プローブが許容する失敗の数。 これは Helm チャートで構成可能です。
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 Helm チャートで構成可能です。
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。 Kubernetes マスター・ノードがワーカー・ノードと通信できなくなった場合、マスター・ノードはワーカー・ノードの障害が発生したのか、単にネットワーク接続が失われたのか判断できません。 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。

関連概念
高可用性の構成

Linux MQ Adv. CD IBM MQ Advanced certified container のユーザー認証と許可

LDAP ユーザーとグループを許可に使用するよう IBM MQ を構成できます。 IBM MQ Advanced certified container では、この方法が推奨されています。

Red Hat OpenShift Container Platform のようなマルチテナントのコンテナー環境では、セキュリティー問題を防ぐためにセキュリティー制約が適用されます。 例えば、Red Hat OpenShift Container Platform のデフォルトの `SecurityContextConstraints` (`restricted`) では、ランダム化されたユーザー ID が使用され、コンテナー自体のローカルのユーザーは使用できません。 IBM MQ は、通常、ユーザーのパスワードを検査するために特権のエスカレーションを使用しますが、これもマルチテナントのコンテナー環境では推奨されません。 このような理由から、実行中のコンテナーの内部のオペレーティング・システム・ライブラリーに定義されているユーザーを使用することは、IBM MQ 認定コンテナーではサポートされません。

ユーザーの認証と許可に LDAP を使用するようにキュー・マネージャーを構成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。

V 9.1.5 Linux MQ Adv. CD OpenShift での IBM MQ Operator のインストールおよびアンインストール

Operator Hub を使用して、OpenShift に IBM MQ Operator をインストールできます。

始める前に

手順

- 14 ページの『OpenShift CLI を使用した IBM MQ Operator のインストール』.
- 12 ページの『OpenShift Web コンソールを使用した IBM MQ Operator のインストール』.

V 9.1.5 Linux MQ Adv. CD OpenShift Web コンソールを使用した IBM MQ Operator のインストール

Operator Hub を使用して、OpenShift に IBM MQ Operator をインストールできます。

始める前に
OpenShift クラスターの Web コンソールにログインします。

手順

1. 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。

a) 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: docker.io/ibmcom/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) 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
  publisher: IBM Content
  sourceType: grpc
  image: docker.io/ibmcom/ibm-operator-catalog
  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 オペレーター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)「**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)「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。

f)「**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。

g)「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

16 ページの『IBM MQ 認定コンテナーのデプロイ』

V 9.1.5 Linux MQ Adv. CD OpenShift CLI を使用した IBM MQ Operator のインストール

Operator Hub を使用して、OpenShift に IBM MQ Operator をインストールできます。

始める前に

oc login を使用して、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. IBM Common Services オペレーター用の OperatorSource を作成します。
   a) OperatorSource 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: docker.io/ibmcom/ibm-common-service-catalog:latest
     updateStrategy:
       registryPoll:
         interval: 45m
   ```

   b) OperatorSource をサーバーに適用します。

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

2. IBM オペレーター用の OperatorSource を作成します。
   a) OperatorSource 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
     publisher: IBM Content
     sourceType: grpc
     image: docker.io/ibmcom/ibm-operator-catalog
     updateStrategy:
       registryPoll:
         interval: 45m
   ```

   b) OperatorSource 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。

   オペレーターをサブスクライブする名前空間に、オペレーターの InstallMode (AllNamespaces モードまたは SingleNamespace モードのどちらか) と一致する OperatorGroup がなければなりません。 インストールするオペレーターが AllNamespaces を使用する場合は、openshift-operators 名前空間に、対応する OperatorGroup が既に配置されています。

   一方、オペレーターが SingleNamespace モードを使用し、対応する OperatorGroup がまだ配置されていない場合は、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>
spec:
  targetNamespaces:
  - <namespace>
```

   b) OperatorGroup オブジェクトを作成します

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

7. 名前空間を MQ Operator にサブスクライブするための Subscription オブジェクトの YAML ファイルを作成します

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

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

   AllNamespaces **InstallMode** を使用する場合は、openshift-operators 名前空間を指定します。 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。

   b) 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。

次のタスク

16 ページの『IBM MQ 認定コンテナーのデプロイ』

Linux MQ Adv. CD IBM MQ 認定コンテナーのデプロイ

IBM MQ バージョン 9.1.5 以上は、 IBM MQ Operator を使用して Red Hat OpenShift にデプロイできます。 IBM MQ バージョン 9.1.5 および 9.1.4 は、 Helm を使用して Red Hat OpenShift にデプロイできます。 以前の CD バージョンは、 Helm を使用して IBM Cloud Private クラスターまたは IBM Cloud Kubernetes Service クラスターにデプロイできます。

このタスクについて

手順

- 25 ページの『Helm CLI を使用したキュー・マネージャーのデプロイ』.
- 27 ページの『以前の CD リリースの IBM MQ の IBM Cloud Private クラスターへのデプロイ』.
- 29 ページの『以前の CD リリースの IBM MQ イメージの IBM Cloud Private クラスターへの追加』.
- 30 ページの『以前の CD リリースの IBM MQ イメージの IBM Cloud Kubernetes Service クラスターへの追加』.

Linux MQ Adv. CD OpenShift CLI を使用した IBM MQ 用の OpenShift プロジェクトの準備

IBM MQ オペレーターを使用してキュー・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 Advanced certified container・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

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

V 9.1.5 Linux MQ Adv. CD 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、16 ページの『OpenShift CLI を使用した IBM MQ 用の 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、「**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)「**キュー・マネージャー**」を選択して、「**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・マネージャー構成 (Queue Manager Config)**」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。

   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。特定の名前を使用することを強制するには、ここで編集することができます。

   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

30 ページの『OpenShift クラスターでデプロイされているキュー・マネージャーへの接続』
Red Hat OpenShift クラスターでデプロイされているキュー・マネージャーに接続するための構成例をいくつか取り上げます。

32 ページの『OpenShift クラスターでデプロイされている IBM MQ Console 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。

V 9.1.5 Linux MQ Adv. CD

OpenShift Web 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

OpenShift クラスターの Web 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、16 ページの『OpenShift CLI を使用した IBM MQ 用の 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) OpenShift Web コンソールで、ナビゲーション・ペインの「**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。
   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、40 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

30 ページの『OpenShift クラスターでデプロイされているキュー・マネージャーへの接続』
Red Hat OpenShift クラスターでデプロイされているキュー・マネージャーに接続するための構成例をいくつか取り上げます。

32 ページの『OpenShift クラスターでデプロイされている IBM MQ Console 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。

V 9.1.5 Linux MQ Adv. CD

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、16 ページの『OpenShift CLI を使用した IBM MQ 用の 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: 9.1.5.0-r2
  license:
    accept: false
    license: L-RJON-BN7PN3
    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 の使用条件に同意する場合は、`accept: false` を `accept: true` に変更してください。 ライセンスについて詳しくは、40 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、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: 9.1.5.0-r2
  license:
    accept: false
    license: L-APIG-BM7GDH
    use: Development
  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、40 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

30 ページの『OpenShift クラスターでデプロイされているキュー・マネージャーへの接続』
Red Hat OpenShift クラスターでデプロイされているキュー・マネージャーに接続するための構成例をいくつか取り上げます。

32 ページの『OpenShift クラスターでデプロイされている IBM MQ Console 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。

V 9.1.4 Linux MQ Adv. CD 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。

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、17 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。

   OpenShift CLI または 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有効にすることができます。

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

   Helm を使用してデプロイする場合は、 `odTracingConfig.enabled=true` および `odTracingConfig.odTracingNamespace=<Operations_Dashboard_Namespace` を設定してトレースを有効にすることができます。 既存のキュー・マネージャー上で Operations Dashboard 統合を有効にする場合は、Helm リリースのアップグレード時にこの設定を適用することができます。

   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. Operations Dashboard 統合を使用するキュー・マネージャーをこの名前空間に初めてデプロイする場合は、Operations Dashboard に 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。

V 9.1.5 Linux MQ Adv. CD OpenShift CLI を使用したカスタムの MQSC および INI ファイルによるイメージのビルド

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。

IBM Entitled Registry に対する OpenShift シークレットが Red Hat OpenShift プロジェクトにない場合は、16 ページの『OpenShift CLI を使用した IBM MQ 用の OpenShift プロジェクトの準備』 の手順に従ってください。

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:9.1.5.0-r2-amd64
         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:9.1.5.0-r2-amd64"
         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。 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。

19 ページの『OpenShift CLI を使用したキュー・マネージャーのデプロイ』で説明している手順に従って、新規カスタム・イメージを YAML に追加します。

YAML の以下のスニペットを通常の QueueManager YAML に追加することができます。ここで、*my-namespace* は、使用する OpenShift プロジェクト/名前空間です。*image* は、先ほど作成したイメージの名前です (例えば「mymq:latest-amd64」など) 。

```
spec:
  queueManager:
    image: image-registry.openshift-image-registry.svc:5000/my-namespace/my-image
```

関連タスク

19 ページの『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。

Linux MQ Adv. CD Helm を使用した IBM MQ 認定コンテナーのデプロイ

IBM MQ 9.1.5.0 以降、キュー・マネージャーをデプロイするための推奨される方法は、 IBM MQ オペレーターを使用することです。 IBM MQ 9.1.5.0 および以前の CD リリースは、以下の手順で Helm を使用してデプロイできます。

このタスクについて

手順

- 23 ページの『Helm を使用した OpenShift 上の IBM MQ 用の OpenShift クラスターの準備』.
- 25 ページの『Helm CLI を使用したキュー・マネージャーのデプロイ』.

Linux MQ Adv. CD Helm を使用した OpenShift 上の IBM MQ 用の OpenShift クラスターの準備

Helm を使用してキュー・マネージャーをデプロイする準備ができるように、Red Hat OpenShift Container Platform クラスターを準備します。 このタスクは、クラスター管理者が実行する必要があります。

始める前に

注 : IBM Cloud Pak for Integration を使用している場合、 IBM MQ で使用するためにインストーラーによって OpenShift プロジェクト (名前空間) が準備されている必要があるため、これら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。

手順

1. IBM Helm リポジトリーを Helm のローカル・コピーに追加してあることを確認します。
   例えば、次のコマンドを実行することができます。

   ```
   helm repo add ibm-entitled-charts https://raw.githubusercontent.com/IBM/charts/master/repo/entitled
   ```

2. Helm サーバー ("Tiller") が自分のクラスター上にインストールされていることを確認します。
   Getting started with Helm on OpenShift に記載されている手順に従って、クラスターに Helm をインストールします。
3. OpenShift プロジェクト (名前空間) のサービス・アカウントに対して、適切なセキュリティー・コンテキスト制約 (SCC) を使用するための権限が付与されていることを確認します。

   V 9.1.5 IBM MQ は、デフォルトの SCC "restricted" の下で動作するため、通常、この手順はスキップすることができます。

   SCC に変更内容を適用する作業は、OpenShift クラスター管理者が実行する必要があります。 SCC の要件は Helm チャートのバージョンごとに異なっています。それについては、その Helm チャートに該当する個々の README ファイルに記載されています。

   ```
   helm inspect readme ibm-entitled-charts/ibm-mqadvanced-server-prod
   ```

   SCC の権限の設定については、各 README の中に指示があります。 IBM MQ Helm チャートは、それぞれが独自に使用するサービス・アカウントを作成します。 したがって、SCC 権限は「グループ」のレベルで (名前空間内のすべてのサービス・アカウントについて) 適用する必要があります。
4. 選択したコンテナー・レジストリーからイメージをプルするための有効な「イメージ・プル・シークレット」があることを確認します

   IBM MQ Advanced certified container・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。

   a) 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。

      i) 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。

      ii)「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。

   b) キュー・マネージャーをデプロイする名前空間に秘密情報を作成します。

   - *<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

25 ページの『Helm CLI を使用したキュー・マネージャーのデプロイ』

V 9.1.4 Linux MQ Adv. CD

Helm CLI を使用したキュー・マネージャーのデプロイ

Helm 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

Helm V2 および Red Hat OpenShift Container Platform 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。 IBM Cloud Pak for Integration を使用していない場合、23 ページの『Helm を使用した OpenShift 上の IBM MQ 用の OpenShift クラスターの準備』の手順を実行してください。

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

1. IBM Helm リポジトリーを Helm のローカル・コピーに追加してあることを確認します。
   例えば、次のコマンドを実行することができます。

   ```
   helm repo add ibm-entitled-charts https://raw.githubusercontent.com/IBM/charts/master/repo/entitled
   ```

2. キュー・マネージャーの構成オプションを検討します

   デプロイメント手順には、インストールと構成の両方の手順が含まれています。 キュー・マネージャーの設定のいくつかは、デプロイメント時に設定する必要があり、それらを変更する場合は再デプロイメントが必要になります。

   デプロイメントで使用可能なすべてのオプションの詳細については、以下のうちのいずれかのコマンドを実行して、Helm チャートの README を参照してください。

   - IBM Cloud Pak for Integration の IBM MQ Advanced certified container の場合、

     ```
     helm inspect readme ibm-entitled-charts/ibm-mqadvanced-server-integration-prod
     ```

   - IBM MQ Advanced certified container の場合、

     ```
     helm inspect readme ibm-entitled-charts/ibm-mqadvanced-server-prod
     ```

   多くの場合、少なくとも以下のパラメーターが必要です。

   a. リリース名。 例: `my-release`
   b. リモート Helm リポジトリー。 例: `ibm-entitled-charts`
   c. Helm チャート: 例えば、`ibm-mqadvanced-server-prod` または `ibm-mqadvanced-server-integration-prod`
   d. イメージ・プル・シークレット名。 例えば、`entitled-registry`。 IBM Cloud Pak for Integration において MQ の事前定義プロジェクトの中にデプロイしている場合、これは不要であることに注意してください。

3. キュー・マネージャーをデプロイします。

   デフォルトでは、Red Hat OpenShift Container Platform クラスター内にデフォルトの記憶域クラス・セットがあると Helm チャートでは想定されていることに注意してください。

   例えば、IBM Cloud Pak for Integration で基本的なキュー・マネージャーをインストールするには、以下のコマンドを実行します。

   ```
   helm install \
   --tls \
   --name my-release \
   ibm-entitled-charts/ibm-mqadvanced-server-integration-prod \
   --set license=accept \
   ```

```
--set tls.hostname=my.cluster \
--set tls.generate=true
```

`tls.hostname` フィールドには任意のホスト名を入力できます (これは必須フィールドですが、この例では新しい自己署名証明書を生成しているため、これは使用されません)

IBM Cloud Pak for Integration とは独立して基本的なキュー・マネージャーをインストールするには、以下のコマンドを実行します。

```
helm install \
--name my-release \
ibm-entitled-charts/ibm-mqadvanced-server-prod \
--set license=accept \
--set image.pullSecret=ibm-entitlement-key
```

関連タスク

30 ページの『OpenShift クラスターでデプロイされているキュー・マネージャーへの接続』
Red Hat OpenShift クラスターでデプロイされているキュー・マネージャーに接続するための構成例をいくつか取り上げます。

32 ページの『OpenShift クラスターでデプロイされている IBM MQ Console 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。

V 9.1.5 Linux MQ Adv. CD

Helm CLI を使用した *IBM Cloud File Storage* でのキュー・マネージャーのデプロイ

Helm を使用して、 IBM Cloud File Storage を使用して IBM Cloud クラスター上の Red Hat OpenShift にキュー・マネージャーをデプロイするシナリオ例。 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

Helm V2 および Red Hat OpenShift Container Platform 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。 IBM Cloud Pak for Integration を使用していない場合、23 ページの『Helm を使用した OpenShift 上の IBM MQ 用の OpenShift クラスターの準備』の手順を実行してください。

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

1. IBM Helm リポジトリーを Helm のローカル・コピーに追加したことを確認します。
   例えば、次のコマンドを実行することができます。

   ```
   helm repo add ibm-entitled-charts https://raw.githubusercontent.com/IBM/charts/master/repo/entitled
   ```

2. キュー・マネージャーをデプロイします。

   IBM Cloud File Storage を使用する場合、通常、 `ibmc-file-gold-gid` ストレージ・クラスを使用すると最良の結果が得られます。 このストレージ・クラスは、正しいファイル・システム・グループ内のユーザーが書き込むことができるストレージを使用可能にします。

   例えば、IBM Cloud Pak for Integration で基本的なキュー・マネージャーをインストールするには、以下のコマンドを実行します。

   ```
   helm install \
   --tls \
   --name my-release \
   ibm-entitled-charts/ibm-mqadvanced-server-integration-prod \
   --set license=accept \
   --set tls.hostname=my.cluster \
   --set tls.generate=true \
   --set dataPVC.storageClassName=ibmc-file-gold-gid \
   --set security.context.supplementalGroups={99}
   ```

`tls.hostname` フィールドには任意のホスト名を入力できます (これは必須フィールドですが、この例では新しい自己署名証明書を生成するため、ここでは使用されません)。

IBM Cloud Pak for Integration とは独立して基本的なキュー・マネージャーをインストールするには、以下のコマンドを実行します。

```
helm install \
--name my-release \
ibm-entitled-charts/ibm-mqadvanced-server-prod \
--set license=accept \
--set image.pullSecret=ibm-entitlement-key \
--set dataPVC.storageClassName=ibmc-file-gold-gid \
--set security.context.supplementalGroups={99}
```

関連タスク

30 ページの『OpenShift クラスターでデプロイされているキュー・マネージャーへの接続』
Red Hat OpenShift クラスターでデプロイされているキュー・マネージャーに接続するための構成例をいくつか取り上げます。

32 ページの『OpenShift クラスターでデプロイされている IBM MQ Console 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。

Linux MQ Adv. CD 以前の CD リリースの IBM MQ の IBM Cloud Private クラスターへのデプロイ

9.1.4 より前の CD バージョンの IBM MQ の場合、IBM Cloud Private 管理コンソールを使用して、キュー・マネージャーを IBM Cloud Private にデプロイします。

始める前に

重要 : V 9.1.4 このデプロイメントは、IBM MQ 9.1.4 以降のバージョンではサポートされません。

このタスクは、IBM MQ イメージが IBM Cloud Private クラスターに既に追加されていることを前提としています。

Helm チャート README.md ファイルは IBM Cloud Private カタログ・エントリーから入手できます。このカタログ・エントリーは、このサブステップを完了するか、またはコマンド行から IBM Cloud Private の **local-charts** リポジトリーをリモート Helm リポジトリーとして追加して次のコマンドを実行すると表示されます。

```
helm inspect readme remote_repo_name/ibm-mqadvanced-server-prod
```

PodSecurity ポリシー、または必要なセキュリティー・コンテキストをサポートする SecurityContextConstraint (IBM Cloud Private on Red Hat OpenShift の場合) が必要です。 例を含む詳細については、Helm チャート README.md ファイルを参照してください。

ご使用の Helm リリースを構成する方法の詳細についても、Helm チャート README.md ファイルを参照してください。

注 :

- デフォルトで必要なセキュリティー設定をサポートしていない IBM Cloud Private 環境にデプロイする場合は、 IBM Cloud Private 製品資料の「非デフォルト名前空間での昇格された特権を必要とする Helm チャートのデプロイ 」の手順に従って、デプロイメントを有効にします。
- SELinux を使用している場合は、「IBM MQ support for SELinux on Red Hat Enterprise Linux」に記載されている IBM MQ 要件を満たす必要があります。

このタスクについて

IBM Cloud Private は、オンプレミスのコンテナー化されたアプリケーションを管理するためのプラットフォームを提供します。 IBM MQ イメージを IBM Cloud Private クラスターに追加した後に、IBM Cloud Private 管理コンソールまたはコマンド行を使用して、キュー・マネージャーをデプロイすることができます。

手順

- IBM Cloud Private 管理コンソールの使用
  a) Web ブラウザーで IBM Cloud Private 管理コンソールを開き、「**カタログ**」をクリックします。

  IBM Cloud Private 製品資料の「 管理コンソールを使用した IBM Cloud Private クラスターへのアクセス 」を参照してください。

  b) リストから `ibm-mqadvanced-server-prod` チャートを選択します。

  c)「**構成**」を選択して、以下の構成手順を実行します。

  a. リリース名を入力します。

  b. ご使用条件を読んで同意します。

  c.「**dataPVC**」セクションで、「**storageclass**」を希望する記憶域クラスに設定します。 デフォルトのストレージ・クラスを選択するにはブランクのままにします。

  d.「**image**」セクションで、リポジトリーを完全イメージ・パスに設定します。 以下に例を示します。

  ```
  mycluster.icp:8500/namespace_name/ibm-mqadvanced-server-prod
  ```

  e.「**image**」セクションで、タグをイメージ・タグに設定します。 以下に例を示します。

  ```
  9.1.3.0-r1
  ```

  f. イメージ・レジストリーにアクセスするために Kubernetes プル・シークレットが必要な場合は、**pullSecret** として追加します。

  g.「**queueManager**」セクションで、キュー・マネージャーの名前を設定します。

  d)「**インストール**」をクリックして、キュー・マネージャーを *Helm* リリース としてデプロイします。
- コマンド行の使用
  a) IBM Cloud Private クラスターにアクセスするように **cloudctl** を構成します。

  IBM Cloud Private 製品資料の「 IBM Cloud Private CLI のインストール 」を参照してください。

  b) 必ず IBM Cloud Private の **local-charts** リポジトリーをリモート Helm リポジトリーとして追加しておきます。

  c) チャートをインストールします。

  次のパラメーターを指定して、下記のコマンドを実行します。

  a. リリース名 (例 `my-release`)

  b. `ibm-mqadvanced-server-prod` チャートを含むリモート helm リポジトリーの名前 (例 `my-repo`)

  c. イメージ・リポジトリー (例 `mycluster.icp:8500/`*`namespace_name`*`/ibm-mqadvanced-server-prod`)

  d. イメージ・タグ (例 `9.1.3.0-r1`)

  ```
  helm install --name my-release --repo my-repo ibm-mqadvanced-server-prod --set
  license=accept --set image.repository=mycluster.icp:8500/namespace_name/ibm-mqadvanced-
  server-prod --set image.tag=9.1.3.0-r1 --tls
  ```

関連タスク

25 ページの『Helm CLI を使用したキュー・マネージャーのデプロイ』
Helm 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。

29 ページの『以前の CD リリースの IBM MQ イメージの IBM Cloud Private クラスターへの追加』
9.1.4 より前の IBM MQ の CD バージョンの場合、 IBM MQ の実動用イメージをデプロイするように IBM Cloud Private クラスターを準備します。

30 ページの『以前の CD リリースの IBM MQ イメージの IBM Cloud Kubernetes Service クラスターへの追加』
9.1.4 より前の IBM MQ の CD バージョンの場合は、 IBM MQ の実動用イメージを IBM Cloud Kubernetes Service にインポートします。

Linux MQ Adv. CD 以前の CD リリースの IBM MQ イメージの IBM Cloud Private クラスターへの追加

9.1.4 より前の IBM MQ の CD バージョンの場合、 IBM MQ の実動用イメージをデプロイするように IBM Cloud Private クラスターを準備します。

このタスクについて

重要 : V 9.1.4 このインポートは、IBM MQ 9.1.4 以降のバージョンではサポートされません。

IBM MQ イメージは Passport Advantage からダウンロードして、IBM Cloud Private コンテナーにインポートできます。

手順

1. 最新の IBM MQ イメージを Passport Advantage および Passport Advantage Express® Web サイト からダウンロードします。

   使用可能なダウンロードの詳細については、IBM MQ 9.1 のダウンロード にアクセスし、ダウンロードするリリースのタブをクリックします。 ダウンロードするパーツの名前と番号が表に一覧表示されます。
2. ダウンロードしたアーカイブ・ファイルを IBM Cloud Private にインポートします。

   IBM Cloud Private 製品資料の「IBM Cloud Private カタログへの IBM ソフトウェアの追加」を参照してください。

次のタスク

これで、キュー・マネージャーを IBM Cloud Private にデプロイする準備ができました。

関連タスク

25 ページの『Helm CLI を使用したキュー・マネージャーのデプロイ』
Helm 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。

27 ページの『以前の CD リリースの IBM MQ の IBM Cloud Private クラスターへのデプロイ』
9.1.4 より前の CD バージョンの IBM MQ の場合、IBM Cloud Private 管理コンソールを使用して、キュー・マネージャーを IBM Cloud Private にデプロイします。

30 ページの『以前の CD リリースの IBM MQ イメージの IBM Cloud Kubernetes Service クラスターへの追加』
9.1.4 より前の IBM MQ の CD バージョンの場合は、 IBM MQ の実動用イメージを IBM Cloud Kubernetes Service にインポートします。

Linux MQ Adv. CD 以前の CD リリースの IBM MQ イメージの IBM Cloud Kubernetes Service クラスターへの追加

9.1.4 より前の IBM MQ の CD バージョンの場合は、 IBM MQ の実動用イメージを IBM Cloud Kubernetes Service にインポートします。

このタスクについて

重要 : V 9.1.4 このインポートは、IBM MQ 9.1.4 以降のバージョンではサポートされません。

IBM MQ イメージは Passport Advantage からダウンロードして、IBM Cloud Kubernetes Service クラスターにインポートできます。

手順

1. 最新の IBM MQ イメージを Passport Advantage および Passport Advantage Express Web サイト からダウンロードします。

   使用可能なダウンロードの詳細については、IBM MQ 9.1 のダウンロード にアクセスし、ダウンロードするリリースのタブをクリックします。 ダウンロードするパーツの名前と番号が表に一覧表示されます。
2. ダウンロードしたアーカイブ・ファイルを IBM Cloud Kubernetes Service にインポートします。

   パブリック Kubernetes コンテナーでの IBM Cloud Private イメージの実行を参照してください。

関連タスク

25 ページの『Helm CLI を使用したキュー・マネージャーのデプロイ』
Helm 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。

27 ページの『以前の CD リリースの IBM MQ の IBM Cloud Private クラスターへのデプロイ』
9.1.4 より前の CD バージョンの IBM MQ の場合、IBM Cloud Private 管理コンソールを使用して、キュー・マネージャーを IBM Cloud Private にデプロイします。

29 ページの『以前の CD リリースの IBM MQ イメージの IBM Cloud Private クラスターへの追加』
9.1.4 より前の IBM MQ の CD バージョンの場合、 IBM MQ の実動用イメージをデプロイするように IBM Cloud Private クラスターを準備します。

V 9.1.4 Linux MQ Adv. CD OpenShift クラスターでデプロイされているキュー・マネージャーへの接続

Red Hat OpenShift クラスターでデプロイされているキュー・マネージャーに接続するための構成例をいくつか取り上げます。

このタスクについて

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、 OpenShift ルート が必要です。

IBM MQ キュー・マネージャーとクライアント・アプリケーションで TLS を有効にする必要があります。 Server Name Indication (SNI) は TLS プロトコルでのみ使用できるためです。 Red Hat OpenShift Container Platform ルーターでは、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 ルートで必要な構成は、クライアント・アプリケーションの SNI 動作によって異なります。

SNI ヘッダーを TLS 1.2 以上として設定するには、TLS 通信に CipherSpec または CipherSuite を使用する必要があります。

以下の条件が満たされる場合、SNI は MQ チャネルに設定されます。

- IBM MQ C クライアントは V8 以降です。
- Java/JMS クライアントは V9.1.1 以降であり、Java インストール済み環境では `javax.net.ssl.SNIHostName` クラスがサポートされます。
- .NET クライアントは非管理対象モードです。

接続名としてホスト名が指定され、以下の条件が満たされる場合、SNI はホスト名に設定されます。

- .NET クライアントは管理対象モードです。
- AMQP または XR クライアントが使用されます。
- Java/JMS クライアントは、 **AllowOutboundSNI** を NO に設定して使用されます。

以下の条件下では、SNI は設定されず、ブランクになります。

- IBM MQ C クライアントが V7.5 以前である。
- IBM MQ C クライアントは、 **AllowOutboundSNI** が NO に設定された状態で使用されます。
- Java/JMS クライアントは、 `javax.net.ssl.SNIHostName` クラスをサポートしない Java インストール済み環境で使用されます。

例

OpenShift ルートに基づくホスト名 : SNI がホスト名に設定されるクライアント・アプリケーションの場合

以下の Helm チャートでは、IBM MQ キュー・マネージャーへのアプリケーションの接続に関して、OpenShift ルートに基づくホスト名が自動的に作成されます。 SNI がホスト名に設定されるクライアント・アプリケーションでは、この OpenShift ルートを使用できます。

- ibm-mqadvanced-server-dev
- ibm-mqadvanced-server-prod
- IBM Cloud Pak for Integration 内の ibm-mqadvanced-server-integration-prod

これらのチャートを使用していない場合で、OpenShift ルートに基づくホスト名を独自に作成する必要があるときには、 クラスターで以下の `yaml` を適用できます。

```
apiVersion: route.openshift.io/v1
  kind: Route
  metadata:
    name: <provide a unique name for the Route>
    namespace: <namespace of your MQ deployment>
  spec:
    to:
      kind: Service
      name: <name of the Kubernetes Service for your MQ deployment (for example "<Helm Release>-
ibm-mq")>
    port:
      targetPort: 1414
    tls:
      termination: passthrough
```

OpenShift ルートに基づく MQ チャネル : SNI が MQ チャネルに設定されるクライアント・アプリケーションの場合

SNI が MQ チャネルに設定されるクライアント・アプリケーションでは、接続先のチャネルごとに、新しい OpenShift ルートを作成する必要があります。 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 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。

新しい各 OpenShift ルートで必要なホスト名を判別するには、https://www.ibm.com/support/pages/ibm-websphere-mq-how-does-mq-provide-multiple-certificates-certlabl-capability に記載されているとおり、それぞれのチャネル名を SNI アドレスにマップする必要があります。

その後、クラスターで以下の `yaml` を適用して新しい OpenShift ルート (チャネルごと) 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 "<Helm
Release>-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 "<Helm Release>-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、OpenShift Container Platform (OCP) 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

25 ページの『Helm CLI を使用したキュー・マネージャーのデプロイ』
Helm 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。

32 ページの『OpenShift クラスターでデプロイされている IBM MQ Console 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。

V 9.1.4 Linux MQ Adv. CD OpenShift クラスターでデプロイされている IBM MQ Console 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 オペレーターを使用している場合、 IBM MQ Console URL は、 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。 また、OpenShift CLI から次のコマンドを実行して確認することもできます。

```
oc get queuemanager <QueueManager Name> -n <namespace of your MQ deployment> --output
jsonpath='{.status.adminUiUrl}'
```

例

以下の Helm チャートによって、IBM MQ Console にアクセスするための OpenShift ルートが自動的に作成されます。

- ibm-mqadvanced-server-dev
- IBM Cloud Pak for Integration 内の ibm-mqadvanced-server-integration-prod

以下のコマンドを実行して、OpenShift ルートのホスト名を取得できます。

```
oc get route <Route Name (for example "<Helm Release>-ibm-mq-web")>
-n <namespace of your MQ deployment> --output jsonpath='{.spec.host}'
```

以下の URL を使用すると、IBM MQ Console にアクセスできます。

```
https://<Route Hostname>/ibmmq/console
```

関連タスク

25 ページの『Helm CLI を使用したキュー・マネージャーのデプロイ』
Helm 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。

30 ページの『OpenShift クラスターでデプロイされているキュー・マネージャーへの接続』
Red Hat OpenShift クラスターでデプロイされているキュー・マネージャーに接続するための構成例をいくつか取り上げます。

Linux MQ Adv. CD 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と復元

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、Operator を使用する場合は、以下のコマンドを実行できます。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
  ```

  例えば、 Helm を使用している場合は、以下のコマンドを実行できます。ここで、 *release_name* は Helm リリースの名前です。

  ```
  oc get pods --selector release=release_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、Operator を使用する場合は、以下のコマンドを実行できます。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
  ```

例えば、 Helm を使用している場合は、以下のコマンドを実行できます。ここで、 *release_name* は Helm リリースの名前です。

```
oc get pods --selector release=release_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
```

独自の IBM MQ コンテナーのビルド

以前は "Docker コンテナー・イメージ"と呼ばれていた、自己作成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

- Docker を使用して IBM MQ 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。
  - Linux 8 ページの『独自の IBM MQ コンテナー・イメージとチャートのビルドのサポート』
  - 34 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』
  - 35 ページの『Docker を使用したサンプル IBM MQ キュー・マネージャー・イメージの作成』
  - 37 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

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。

V 9.1.0 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。

Docker を使用したサンプル IBM MQ キュー・マネージャー・イメージの作成

コンテナーで IBM MQ キュー・マネージャーを実行するためにサンプルのコンテナー・イメージをビルドするには、この情報を活用してください。

このタスクについて

まず、Red Hat Universal Base Image ファイル・システムと IBM MQ のクリーン・インストールが含まれているベース・イメージをビルドします。

次に、そのベースの上に、ユーザー ID とパスワードによる基本的なセキュリティーを可能にする IBM MQ 構成を追加するための別のコンテナー・イメージ層をビルドします。

最後に、このイメージをファイル・システムとして使用し、ホスト・ファイル・システム上のコンテナー固有のボリュームによって提供される /var/mqm の内容を使用してコンテナーを実行します。

手順

- コンテナーで IBM MQ キュー・マネージャーを実行するためにサンプルのコンテナー・イメージをビルドする方法については、以下のサブトピックを参照してください。
  - 35 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』
  - 35 ページの『サンプルの構成済みの IBM MQ キュー・マネージャーのイメージのビルド』

サンプルの IBM MQ キュー・マネージャーのベース・イメージのビルド

独自のコンテナー・イメージの IBM MQ を使用するためには、まず、IBM MQ クリーン・インストールを含むベース・イメージをビルドする必要があります。 以下の手順は、GitHub でホストされているサンプル・コードを使用してサンプルのベース・イメージをビルドする方法を示しています。

手順

- mq-container GitHub リポジトリーで提供されている Make ファイルを使用して実稼働用のコンテナー・イメージをビルドします。

  GitHub の コンテナー・イメージの作成の指示に従います。

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。

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

IBM MQ 9.1 イメージの場合、 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ことはできません。 "制限付き" SCC はランダム・ユーザー ID を使用し、別のユーザーに変更することによって特権のエスカレーションを防止します。 IBM MQ 9.1 RPM ベースのインストーラーは、 mqm ユーザーおよびグループに依存し、実行可能プログラムの setuid ビットも使用します。

この制限は、 IBM MQ 9.2 で削除されました。

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 CHANNEL(PASSWORD.SVRCONN) CHLTYPE(SVRCONN)
SET CHLAUTH(PASSWORD.SVRCONN) TYPE(BLOCKUSER) USERLIST('nobody') +
DESCR('Allow privileged users on this channel')
SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS) DESCR('BackStop rule')
SET CHLAUTH(PASSWORD.SVRCONN) TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(CHANNEL) CHCKCLNT(REQUIRED)
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) ADOPTCTX(YES)
REFRESH SECURITY TYPE(CONNAUTH)
```

   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
RUN useradd johndoe -G mqm && \
    echo johndoe:passw0rd | chpasswd
COPY config.mqsc /etc/mqm/
```

   ここで、

   - johndoe は、追加するユーザー ID です。
   - passw0rd は、元のパスワードです。

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。

```
sudo 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。

   RUN コマンドで、johndoe というユーザーを追加してパスワード passw0rd を指定し、 **COPY** コマンドで、親イメージにとって既知の特定の場所に config.mqsc ファイルを追加します。

   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。 **podman** の場合は、コマンドの前の **sudo** は不要です。

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。

```
sudo 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

構成済みのコンテナー・イメージを作成できました。 Docker の **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。 コンテナーで実行されている IBM MQ プロセスは、docker **top** コマンドを使用して表示できます。

重要 :

コンテナーのログは、docker **logs ${CONTAINER_ID}** コマンドを使用して表示できます。

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、コンテナーは失敗した可能性があります。 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** コマンドを実行したときにも出力されています。
- コンテナーのログは、docker **logs ${CONTAINER_ID}** コマンドを使用して表示できます。
- オープン・ファイルの最大数は、コマンド **sysctl fs.file-max=524288** を使用して設定できます。

V 9.1.0 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、38 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。

V 9.1.0 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、37 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
```

V 9.1.5 Linux MQ Adv. CD IBM MQ Operator の API リファレンス

IBM MQ には、OpenShift Container Platform とのネイティブな統合を可能にする Kubernetes オペレーターが用意されています。

V 9.1.5 Linux MQ Adv. CD 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。

V 9.1.5 Linux MQ Adv. CD mq.ibm.com/v1beta1 のライセンスのリファレンス

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
L-RJON-BN7PN3	Production または NonProduction	IBM Cloud Pak for Integration 2020.2
L-RJON-BPHL2Y		IBM Cloud Pak for Integration 限定版 2020.2
L-APIG-BJAKBF	Production または Development	IBM MQ Advanced 9.1.5
L-APIG-BM7GDH	Development	IBM MQ Advanced for Developers 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。

V 9.1.5 Linux MQ Adv. CD 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。

フィールド	説明
metadata	
spec 44 ページの『QueueManager の仕様』	QueueManager について必要とする状態。
status 45 ページの『キュー・マネージャーの状況』	QueueManager について観測された状態。

可用性

キュー・マネージャーの可用性の設定 (アクティブ/スタンバイのペアを使用するかどうかなど)。

以下の中に含まれます:

- 42 ページの『QueueManager 構成』

フィールド	説明
type 文字列	使用する可用性のタイプ。 単一のポッドには「SingleInstance」を使用します。これは、 Kubernetes によって自動的に (場合によっては) 再始動されます。 ポッドのペアに「MultiInstance」を使用します。一方のポッドは「アクティブな」キュー・マネージャーであり、もう一方のポッドはスタンバイです。 最新バージョンの IBM MQ の コンテナー内の IBM MQ の高可用性 を参照してください。

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:

- 44 ページの『QueueManager の仕様』

フィールド	説明
use 文字列	複数の使用方法をサポートするライセンスの場合に、ソフトウェアの使用方法を制御する設定。 有効な値については、「 https://ibm.biz/BdqvCF 」を参照してください。
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、「 https://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」などです。

制限

QueueManagerResourceList は、CPU & メモリー設定を定義します。

以下の中に含まれます:

- 48 ページの『リソース』

フィールド	説明
cpu	
memory	

LocalObject リファレンス

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:

- 44 ページの『QueueManager の仕様』

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/#names TODO: その他の便利なフィールドを追加してください。 apiVersion、kind、uid など。

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:

- 44 ページの『QueueManager の仕様』

フィールド	説明
keys 42 ページの『PKISource』 配列	キュー・マネージャーの鍵リポジトリーに追加する秘密鍵です。
trust 42 ページの『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。

PKISource

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:

- 42 ページの『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 48 ページの『秘密』	Kubernetes シークレットを使用して鍵を渡します。

QueueManager 構成

QueueManagerConfig は、キュー・マネージャー・コンテナーおよび基礎となるキュー・マネージャーの設定を定義します。

以下の中に含まれます:

- 44 ページの『QueueManager の仕様』

フィールド	説明
logFormat 文字列	このコンテナーで使用するログの形式。 コンテナーからの JSON 形式のログには「JSON」を使用します。 テキスト形式のメッセージには「基本」を使用してください。
metrics 43 ページの『QueueManager メトリック』	Prometheus スタイルのメトリックの設定。

フィールド	説明
readinessProbe 44 ページの『QueueManagerReadinessProbe』	Readiness プローブを制御する設定。
resources 48 ページの『リソース』	リソース要件を制御する設定。
storage 47 ページの『QueueManager ストレージ』	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。
availability 41 ページの『可用性』	キュー・マネージャーの可用性の設定 (アクティブ/スタンバイのペアを使用するかどうかなど)。
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。
livenessProbe 43 ページの『QueueManagerLivenessProbe』	Liveness プローブを制御する設定。
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。
image 文字列	使用するコンテナー・イメージ。
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。

QueueManagerLivenessProbe

Liveness プローブを制御する設定。

以下の中に含まれます:

- 42 ページの『QueueManager 構成』

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。
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。 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)。
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。
timeoutSeconds 整数	プローブがタイムアウトになるまでの秒数。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。

QueueManager メトリック

Prometheus スタイルのメトリックの設定。

以下の中に含まれます:

- 42 ページの『QueueManager 構成』

フィールド	説明
enabled ブール値	Prometheus 対応のメトリック・エンドポイントを有効にするかどうか。

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。

以下の中に含まれます:

- 47 ページの『QueueManager ストレージ』

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" の場合にのみ有効です。
enabled ブール値	このボリュームを別のボリュームとして使用可能にするかどうか、またはデフォルトの「queueManager」ボリュームに配置するかどうか。
size 文字列	Kubernetes に渡される PersistentVolume のサイズ。 "type" が "persistent-claim" の場合にのみ有効です。
sizeLimit 文字列	「一時」ボリュームを使用する場合のサイズ制限。 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「一時」の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。 ephemeral を選択して非永続「emptyDir」ボリュームを作成するか、 persistent-claim を選択して永続ボリュームを使用します。

QueueManagerReadinessProbe

Readiness プローブを制御する設定。

以下の中に含まれます:

- 42 ページの『QueueManager 構成』

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。
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。 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)。
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。
timeoutSeconds 整数	プローブがタイムアウトになるまでの秒数。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。

QueueManager の仕様

QueueManager について必要とする状態。

以下の中に含まれます:

- 40 ページの『QueueManager』

フィールド	説明
license 41 ページの『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 42 ページの『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 42 ページの『QueueManager 構成』	QueueManagerConfig は、キュー・マネージャー・コンテナーおよび基礎となるキュー・マネージャーの設定を定義します。
securityContext 48 ページの『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。
tracing 50 ページの『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して、 MQ バージョン 9.1.5.0 を指定します。 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。
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。
imagePullSecrets 42 ページの『LocalObject リファレンス』 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。
template 49 ページの『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。
web 51 ページの『WebServer 構成』	MQ Web サーバーの設定。

キュー・マネージャーの状況

QueueManager について観測された状態。

以下の中に含まれます:

- 40 ページの『QueueManager』

フィールド	説明
endpoints 46 ページの『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)。

フィールド	説明
name 文字列	キュー・マネージャーの名前。
versions 46 ページの『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。
adminUiUrl 文字列	管理 UI の URL。
conditions 46 ページの『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。

QueueManagerStatusCondition

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:

- 45 ページの『キュー・マネージャーの状況』

フィールド	説明
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。
type 文字列	状況のタイプ。
lastTransitionTime 文字列	状況のステータスが最後に遷移した時刻。

QueueManagerStatusEndpoint

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:

- 45 ページの『キュー・マネージャーの状況』

フィールド	説明
name 文字列	エンドポイントの名前。
type 文字列	エンドポイントのタイプ。 UI エンドポイントの場合は「UI」、 API エンドポイントの場合は「API」、 API 資料の場合は「OpenAPI」です。
uri 文字列	エンドポイントの URI。

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。

以下の中に含まれます:

- 45 ページの『キュー・マネージャーの状況』

フィールド	説明
available 47 ページの『QueueManagerStatusVersion 使用可能』	IBM Entitled Registry から取得できるその他のバージョンの MQ。

フィールド	説明
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。

QueueManagerStatusVersion 使用可能

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:

- 46 ページの『QueueManagerStatusVersion』

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 51 ページの『バージョン』 配列	使用可能な MQ の具体的なバージョン。

QueueManager ストレージ

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:

- 42 ページの『QueueManager 構成』

フィールド	説明
persistedData 44 ページの『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 47 ページの『QueueManager ボリューム』	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 44 ページの『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。

QueueManager ボリューム

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:

- 47 ページの『QueueManager ストレージ』

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" の場合にのみ有効です。
size 文字列	Kubernetes に渡される PersistentVolume のサイズ。 "type" が "persistent-claim" の場合にのみ有効です。

フィールド	説明
sizeLimit 文字列	「一時」ボリュームを使用する場合のサイズ制限。 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「一時」の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。 ephemeral を選択して非永続「emptyDir」ボリュームを作成するか、 persistent-claim を選択して永続ボリュームを使用します。

要求

QueueManagerResourceList は、CPU & メモリー設定を定義します。

以下の中に含まれます:

- 48 ページの『リソース』

フィールド	説明
memory	
cpu	

リソース

リソース要件を制御する設定。

以下の中に含まれます:

- 42 ページの『QueueManager 構成』

フィールド	説明
limits 41 ページの『制限』	QueueManagerResourceList は、CPU & メモリー設定を定義します。
requests 48 ページの『要求』	QueueManagerResourceList は、CPU & メモリー設定を定義します。

秘密

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:

- 42 ページの『PKISource』

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。

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:

- 44 ページの『QueueManager の仕様』

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。
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されたボリュームにアクセスするために root ユーザーでなければならないストレージ・プロバイダーを使用している場合は、これを「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) オブジェクトに影響します。root ユーザーを許可する SCC の使用を許可されていない場合、キュー・マネージャーは開始に失敗する可能性があります。 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。

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。

以下の中に含まれます:

- 44 ページの『QueueManager の仕様』

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。

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。

以下の中に含まれます:

- 50 ページの『TracingConfig』

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。
livenessProbe 50 ページの『TracingProbe』	Liveness プローブを制御する設定。
readinessProbe 50 ページの『TracingProbe』	Readiness プローブを制御する設定。

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。

以下の中に含まれます:

- 50 ページの『TracingConfig』

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。
livenessProbe 50 ページの『TracingProbe』	Liveness プローブを制御する設定。
readinessProbe 50 ページの『TracingProbe』	Readiness プローブを制御する設定。

TracingConfig

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:

- 44 ページの『QueueManager の仕様』

フィールド	説明
agent 49 ページの『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 50 ページの『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。
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。

TracingProbe

Readiness プローブを制御する設定。

以下の中に含まれます:

- 50 ページの『TracingCollector』

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。
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。 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)。
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。

フィールド	説明
timeoutSeconds 整数	プローブがタイムアウトになるまでの秒数。 詳細情報: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes。

バージョン

QueueManagerStatusVersion は MQ のバージョンを示します。

以下の中に含まれます:

- 47 ページの『QueueManagerStatusVersion 使用可能』

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

WebServer 構成

MQ Web サーバーの設定。

以下の中に含まれます:

- 44 ページの『QueueManager の仕様』

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。

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 (http://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: