Red Hat AMQ 7.4

OpenShift Container Platform での AMQ Broker のデプロイ

AMQ Broker 7.4 向け

Last Updated: 2023-03-01

Red Hat AMQ 7.4 OpenShift Container Platform での AMQ Broker のデプロイ

AMQ Broker 7.4 向け

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

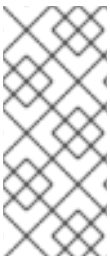
概要

OpenShift Container Platform に AMQ Broker をインストールし、デプロイする方法を説明します。

目次

第1章 はじめに

Red Hat AMQ Broker 7.4 は、OpenShift Container Platform(OCP)3.11 以降で使用するために提供されるコンテナー化イメージとして利用できます。



注記

AMQ Broker 7.4 は、Long Term Support (LTS) リリースバージョンとして指定されています。バグ修正およびセキュリティーアドバイザリーは、少なくとも 12 カ月間、一連のマイクロリリース（7.4.1、7.42、7.3 など）で AMQ Broker 7.4 で利用可能になります。つまり、新しいマイナーリリースにアップグレード **することなく**、AMQ Broker の最新のバグ修正およびセキュリティーアドバイザリーを取得できます。

AMQ Broker は Apache ActiveMQ Artemis をベースにしています。JMS に準拠するメッセージブローカーを提供します。初期ブローカー Pod を設定した後に、OpenShift Container Platform 機能を使用して重複を迅速にデプロイできます。

OCP 上の AMQ Broker は、Red Hat AMQ Broker と同様の機能を提供しますが、機能の一部は OpenShift Container Platform で使用するために特別に設定する必要があります。

1.1. バージョンの互換性とサポート

OpenShift Container Platform 4.1 イメージバージョンの互換性についての詳細は、「OpenShift and Atomic Platform Tested Integrations」のページ を参照してください。

1.2. サポートされない特性

- マスタースレーブベースの高可用性
  マスターとスレーブのペアを設定して実現する高可用性 (HA) はサポートされません。その代わりに、Pod がスケールダウンされると、スケールダウンコントローラーを使用して HA が OpenShift で提供され、メッセージの移行が可能になります。

  OpenShift プロキシーまたはバインドポートを使用して、ブローカーのクラスターに接続する外部クライアントを HA に適切に設定しなければならない場合があります。クラスター化されたシナリオでは、ブローカーによって、ブローカーのホストとポート情報のすべてのアドレスについて特定のクライアントに通知します。これらは内部でのみアクセスできるため、一部のクライアント機能は機能しないか、または無効にする必要があります。

クライアント	設定
Core JMS クライアント	外部 Core Protocol JMS クライアントは HA またはいずれのフェイルオーバーもサポートしないため、接続ファクトリーは **useTopologyForLoadBalancing=false** で設定する必要があります。
AMQP クライアント	AMQP クライアントがフェイルオーバーリストをサポートしません。

- クラスター内の永続サブスクリプション
  永続サブスクリプションが作成されると、これはクライアントが接続したブローカーの永続キューとして表されます。クラスターが OpenShift 内で実行されている場合、クライアントが

永続サブスクリプションキューが作成されたブローカーを認識しません。サブスクリプションが永続的であり、クライアントが再接続する方法は現在、ロードバランサーが同じノードに再接続する方法はありません。このような場合には、クライアントが別のブローカーに接続し、重複したサブスクリプションキューを作成できます。このため、ブローカーのクラスターで永続サブスクリプションを使用することは推奨されていません。

第2章 OPERATOR を使用した OPENSHIFT CONTAINER PLATFORM での AMQ BROKER のデプロイ

2.1. AMQ BROKER OPERATOR の概要

Kubernetes、および OpenShift Container Platform では、シークレットの処理、負荷分散、サービス検出、自動スケーリングなどの機能が含まれており、複雑な分散システムをビルドできます。Operator は、Kubernetes アプリケーションをパッケージ化、デプロイ、および管理できるようにするプログラムです。多くの場合、Operator は共通タスクまたは複雑なタスクを自動化します。

通常、Operator は以下を提供することを目的としています。

- 一貫性のある繰り返し可能なインストール
- システムコンポーネントのヘルスチェック
- OTA (Over-the-air) 更新
- 管理アップグレード

Operator は **Custom Resource Definitions**および対応するカスタムリソースと呼ばれる Kubernetes 拡張メカニズムを使用して、カスタムオブジェクトがネイティブでビルトインされた Kubernetes オブジェクトのように表示され、機能することを確認します。*カスタムリソース定義およびカスタムリソースは、デプロイする予定の OpenShift オブジェクトの設定を指定する方法です。*

以前のバージョンでは、アプリケーションテンプレートのみを使用して OpenShift Container Platform に AMQ Broker をデプロイできました。テンプレートは初期デプロイメントの作成に有効ですが、デプロイメントを更新するメカニズムを提供しません。Operator は、設定を指定するカスタムリソースへの変更を常にリッスンするため、ブローカーインスタンスの実行中に変更を加えることができます。カスタムリソースに変更を加えると、Operator は変更をプロジェクトの既存のブローカーインストールと調整し、加えた変更を反映させます。

2.2. カスタムリソース定義の概要

*通常、カスタムリソース定義 (CRD) は、Operator でデプロイされたカスタム OpenShift オブジェクトのスキーマです。付随するカスタムリソース(CR)ファイルを使用すると、CRD の設定アイテムの値を指定できます。Operator 開発者の場合、CRD を使用して公開する内容は基本的に、デプロイされたオブジェクトの設定および使用方法のために API になります。CRD は Kubernetes 経由で自動的に公開されるため、通常の HTTP **curl** コマンドを使用して CRD に直接アクセスできます。Operator は、HTTP 要求を使用して **kubectl** コマンド経由で Kubernetes と対話します。*

*メインブローカー CRD は、Operator のインストール時にダウンロードして抽出するアーカイブの **deploy/crds** ディレクトリーにある **broker_v2alpha1_activemqartemis** ファイルです。この CRD を使用すると、指定の OpenShift プロジェクトでブローカーデプロイメントを設定できます。**deploy/crds** ディレクトリーの他の CRD はアドレスを設定し、Operator がスケールダウンコントローラーをインスタンス化する際に使用するために使用されます。*

デプロイされる場合、各 CRD は Operator 内で独立して実行される個別のコントローラーになります。

各 CRD の完全な設定リファレンスについては、以下を参照してください。

- *ブローカー CRD 設定リファレンス*
- *アドレス指定 CRD 設定リファレンス*

2.2.1. ブローカーカスタムリソースのサンプル

インストール時にダウンロードおよび抽出する AMQ Broker Operator アーカイブには、**deploy/crs** ディレクトリーにサンプル CR ファイルが含まれています。以下のサンプル CR ファイルでは、以下が可能になります。

- SSL またはクラスタリングなしで最小ブローカーをデプロイします。
- アドレスを定義します。

ダウンロードするブローカー Operator アーカイブには、以下に一覧表示されているように **deploy/examples** ディレクトリーにデプロイメントの CR が含まれます。

artemis-basic-deployment.yaml

基本ブローカーデプロイメント。

artemis-persistence-deployment.yaml

永続ストレージのあるブローカーデプロイメント。

artemis-cluster-deployment.yaml

クラスター化したブローカーのデプロイメント。

artemis-persistence-cluster-deployment.yaml

永続ストレージのあるクラスターブローカーのデプロイメント。

artemis-ssl-deployment.yaml

SSL セキュリティーを使用したブローカーデプロイメント。

artemis-ssl-persistence-deployment.yaml

SSL セキュリティーおよび永続ストレージを使用したブローカーデプロイメント。

artemis-aio-journal.yaml

ブローカージャーナルで非同期 I/O (AIO) の使用。

address-queue-create.yaml

アドレスおよびキューの作成。

以下のセクションでは、Operator、CRD、および一部の CR を使用して OpenShift Container Platform でコンテナーベースのブローカーデプロイメントを作成する方法を説明します。この手順を正常に完了したら、Operator が個別の Pod で実行されます。作成する各ブローカーインスタンスは、プロジェクトの Pod を含む個別の StatefulSet で実行されます。専用の CR を使用してブローカーデプロイメントでアドレスを定義します。

注記

複数のブローカー CR インスタンスをデプロイすることで、指定の OpenShift プロジェクトに複数のブローカーデプロイメントを作成することはできません。ただし、プロジェクトにブローカーデプロイメントを作成した場合は、アドレスに複数の CR インスタンスをデプロイ**できます**。

2.3. AMQ BROKER OPERATOR のインストール

本セクションの手順では以下の方法を説明します。

- Operator for AMQ Broker 7.4 の最新バージョンのインストール
- ブローカーデプロイメント用のAMQ Broker 7.4の最新ブローカーコンテナーイメージの指定

以下では、Operator のインストールについて注意すべき重要な事項について説明します。

重要

- *AMQ Broker 7.4 は、Long Term Support (LTS) リリースバージョンとして指定されています。バグ修正およびセキュリティーアドバイザリーは、少なくとも 12 カ月間、一連のマイクロリリース（7.4.1、7.42、7.3 など）で AMQ Broker 7.4 で利用可能になります。つまり、新しいマイナーリリースにアップグレード**することなく**、AMQ Broker の最新のバグ修正およびセキュリティーアドバイザリーを取得できます。*

- *サポート対象の設定を維持するには、LTS リリースストリームの最新マイクロリリースにアップグレードする必要があります。つまり、ブローカーデプロイメントの LTS ストリームから最新のブローカーコンテナーイメージを使用する必要があります。ここでは、AMQ Broker Operator をベースとしたブローカーデプロイメントの最新ブローカーコンテナーイメージを指定する方法を説明します。*

- *AMQ Broker Operator for AMQ Broker 7.4 のバージョン 0.6 は、テクノロジープレビュー機能としてのみご利用いただけます。Operator のバージョン 0.6 が OpenShift プロジェクトにインストールされている場合、Operator を最新の LTS(Long Term Support)バージョンに更新することが推奨されます。Operator の LTS バージョンには、バグおよびセキュリティーアドバイザリーの修正が含まれます。Red Hat は、実稼働環境での使用のために Operator の LTS バージョンをサポートします。*

- *AMQ Broker Operator のバージョン 0.6 によって使用されるカスタムリソース定義(CRD)は、Long Term Support(LTS)バージョンと互換性がありません。Operator の最新の LTS バージョンをインストールするには、最新の CRD をデプロイする前に、OpenShift クラスターにデプロイされた CRD を削除する必要があります。また、Operator の LTS バージョンをインストールするプロジェクトから既存の Operator およびブローカーデプロイメントを削除する必要もあります。以下の手順は、本項の手順で説明します。*

- *最新の CRD をデプロイするには、OpenShift クラスターのクラスター管理者権限が必要です。管理者以外のユーザーは、カスタムリソース(CR)をデプロイして OpenShift プロジェクトにブローカーインスタンスを作成できます。*

- *最新の CRD で OpenShift クラスターを更新すると、この更新はクラスター内の**すべてのプロジェクト** に影響します。バージョン 0.6 から以前にデプロイされたブローカー Pod は機能しなくなりました。OpenShift クラスターの影響を受ける各プロジェクトを、Operator の LTS バージョンを使用するように更新する必要があります。その後、Operator の LTS バージョンに含まれるカスタムリソース(CR)をデプロイして、以前のブローカーデプロイメントを再作成できます。*

2.3.1. Operator コードの取得

この手順では、AMQ Broker 7.4 用の Operator の最新の LTS バージョンをインストールするために必要なコードにアクセスして準備する方法を説明します。

手順

1. *Web ブラウザーで、AMQ Broker **Software Downloads** ページ に移動します。*
2. ***Version** ドロップダウンメニューで **7.4.6** を選択します。*

3. *AMQ Broker 7.4.6 Operator Installation and Example Files の横にある Download をクリックします。*
   ***amq-broker-operator-7.4.6-ocp-install-examples.zip** 圧縮アーカイブのダウンロードが自動的に開始されます。*

4. *ダウンロードが完了したら、アーカイブを選択したインストールディレクトリーに移動します。以下の例では、アーカイブを **/broker/operator** というディレクトリーに移動します。*

```
sudo mv amq-broker-operator-7.4.6-ocp-install-examples.zip /broker/operator
```

5. *選択したインストールディレクトリーで、アーカイブの内容を展開します。以下に例を示します。*

```
cd /broker/operator
sudo unzip amq-broker-operator-7.4.6-ocp-install-examples.zip
```

6. *クラスター管理者として OpenShift Container Platform にログインします。以下に例を示します。*

```
$ oc login -u system:admin
```

7. *Operator をインストールするプロジェクトを指定します。新規プロジェクトを作成するか、または既存プロジェクトに切り替えることができます。*

   a. *新しいプロジェクトを作成します。*

```
$ oc new-project <project_name>
```

   b. *または、既存のプロジェクトに切り替えます。*

```
$ oc project <project_name>
```

8. *Operator で使用するサービスアカウントを指定します。*

   a. *展開した Operator アーカイブの **deploy** ディレクトリーで、**service_account.yaml** ファイルを開きます。*

   b. ***kind** 要素が **ServiceAccount** に設定されていることを確認します。*

   c. ***metadata** セクションで、カスタム名をサービスアカウントに割り当てるか、デフォルト名を使用します。デフォルトの名前は **amq-broker-operator** です。*

   d. *プロジェクトにサービスアカウントを作成します。*

```
$ oc create -f deploy/service_account.yaml
```

9. *Operator のロール名を指定します。*

   a. ***role.yaml** ファイルを開きます。このファイルは、Operator が使用できるリソースを指定し、変更します。*

   b. ***kind** 要素が **Role** に設定されていることを確認します。*

   c. ***metadata** セクションで、カスタム名をロールに割り当てるか、デフォルト名を使用します。デフォルトの名前は **amq-broker-operator** です。*

d. プロジェクトにロールを作成します。

```
$ oc create -f deploy/role.yaml
```

10. Operator のロールバインディングを指定します。ロールバインディングは、指定した名前に基づいて、事前に作成されたサービスアカウントを Operator ロールにバインドします。

a. ***role_binding.yaml*** ファイルを開きます。***ServiceAccount*** と ***Role*** の ***name*** の値が ***service_account.yaml*** および ***role.yaml*** ファイルで指定された値と一致していることを確認します。以下に例を示します。

```
metadata:
    name: amq-broker-operator
subjects:
    kind: ServiceAccount
    name: amq-broker-operator
roleRef:
    kind: Role
    name: amq-broker-operator
```

b. プロジェクトでロールバインディングを作成します。

```
$ oc create -f deploy/role_binding.yaml
```

2.3.2. Operator のデプロイ

本セクションの手順では、OpenShift プロジェクトに Operator for AMQ Broker 7.4 の最新の LTS バージョンをデプロイする方法を説明します。

前提条件

- Operator デプロイメント用に OpenShift プロジェクトを準備している。「Operator コードの取得」を参照してください。
- AMQ Broker 7.3 以降では、新しいバージョンの Red Hat コンテナーレジストリーを使用してコンテナーイメージにアクセスします。この新しいバージョンのレジストリーでは、イメージにアクセスする前に認証されたユーザーである必要がある。本セクションの手順を実行する前に、「Red Hat Container Registry Authentication」で説明されている手順を完了する必要がある。
- 永続ストレージを使用してブローカーをデプロイする予定で、OpenShift クラスターにコンテナーネイティブストレージがない場合は、永続ボリュームを手動でプロビジョニングし、それらを Operator で要求できるようにする必要があります。たとえば、永続ストレージを持つ 2 つのブローカーのクラスターを作成する場合（カスタムリソースに ***persistenceEnabled=true*** を設定して）、永続ボリュームを 2 つ利用可能にする必要があります。デフォルトでは、各ブローカーインスタンスには 2 GiB のストレージが必要である。
  カスタムリソースで ***persistenceEnabled=false*** を指定した場合、デプロイされたブローカーは**一時**ストレージを使用する。一時ストレージは、ブローカー Pod を再起動するたびに、既存のデータが失われることを意味する。

  OpenShift Container Platform での永続ストレージのプロビジョニングに関する詳細は、OpenShift Container Platform ドキュメントの「永続ストレージについて」を参照してください。

手順

1. *OpenShift Container Platform Web コンソールで、ブローカーデプロイメントするプロジェクトを開きます。*
   新規プロジェクトを作成した場合、これは現時点で空になります。デプロイメント、StatefulSets、Pods、Services、または Routes がないことを確認します。

2. *以前のバージョンの AMQ Broker Operator をプロジェクトにデプロイした場合は、メインブローカーカスタムリソース(CR)をプロジェクトから削除します。メイン CR を削除すると、プロジェクトの既存のブローカーデプロイメントが削除されます。以下に例を示します。*

```
oc delete -f deploy/crs/broker_v1alpha1_activemqartemis_cr.yaml.
```

3. *以前のバージョンの AMQ Broker Operator をプロジェクトにデプロイした場合は、この Operator インスタンスを削除します。以下に例を示します。*

```
$ oc delete -f deploy/operator.yaml
```

4. *以前のバージョンの AMQ Broker Operator について OpenShift クラスターにカスタムリソース定義(CRD)をデプロイした場合は、これらの CRD をクラスターから削除します。以下に例を示します。*

```
oc delete -f deploy/crds/broker_v1alpha1_activemqartemis_crd.yaml
oc delete -f deploy/crds/broker_v1alpha1_activemqartemisaddress_crd.yaml
oc delete -f deploy/crds/broker_v1alpha1_activemqartemisscaledown_crd.yaml
```

5. *ダウンロードして抽出した Operator アーカイブの **deploy/crds** ディレクトリに含まれる CRD をデプロイします。Operator をデプロイして起動する前に、OpenShift クラスターに最新の CRD をインストールする必要があります。*

   a. *メインブローカー CRD をデプロイします。*

```
$ oc create -f deploy/crds/broker_v2alpha1_activemqartemis_crd.yaml
```

   b. *アドレス指定 CRD をデプロイします。*

```
$ oc create -f deploy/crds/broker_v2alpha1_activemqartemisaddress_crd.yaml
```

   c. *スケールダウンコントローラー CRD をデプロイします。*

```
$ oc create -f deploy/crds/broker_v2alpha1_activemqartemisscaledown_crd.yaml
```

6. *Red Hat Container Registry の認証に使用するアカウントに関連付けられたプルシークレットを、OpenShift プロジェクトの **デフォルト** の、**デプロイヤー**、および **builder** サービスアカウントにリンクします。*

```
$ oc secrets link --for=pull default <secret-name>
$ oc secrets link --for=pull deployer <secret-name>
$ oc secrets link --for=pull builder <secret-name>
```

注記

OpenShift Container Platform 4.1 以降では、Web コンソールを使用して、プルシークレットを AMQ Broker Operator などのコンテナーイメージをデプロイするプロジェクトに関連付けることもできます。そのためには、**Administration → Service Accounts** をクリックします。Red Hat コンテナーレジストリーでの認証に使用するアカウントに関連付けられたプルシークレットを指定します。

7. ダウンロードした Operator アーカイブの **deploy** ディレクトリーで、**operator.yaml** ファイルを開きます。**spec.containers.image** を、Red Hat Container Registry の Operator for AMQ Broker 7.4 の最新 LTS バージョンのイメージへの完全パスで更新します。

```
spec:
  template:
    spec:
      containers:
        image: registry.redhat.io/amq7/amq-broker-lts-rhel7-operator:0.9
```

8. Operator をデプロイします。

```
$ oc create -f deploy/operator.yaml
```

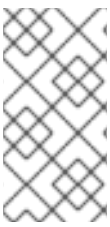
OpenShift プロジェクトの、デプロイした **amq-broker-operator** イメージは新しい Pod で起動します。

新規 Pod の **Events** タブにある情報は、OpenShift が指定した Operator イメージをデプロイし、新規コンテナーを OpenShift クラスターのノードに割り当て、新規コンテナーを起動していることを確認します。

さらに、Pod 内の **Logs** タブをクリックしても、出力には、以下のような行が含まれるはずです。

```
...
{"level":"info","ts":1553619035.8302743,"logger":"kubebuilder.controller","msg":"Starting
Controller","controller":"activemqartemisaddress-controller"}
{"level":"info","ts":1553619035.830541,"logger":"kubebuilder.controller","msg":"Starting
Controller","controller":"activemqartemis-controller"}
{"level":"info","ts":1553619035.9306898,"logger":"kubebuilder.controller","msg":"Starting
workers","controller":"activemqartemisaddress-controller","worker count":1}
{"level":"info","ts":1553619035.9311671,"logger":"kubebuilder.controller","msg":"Starting
workers","controller":"activemqartemis-controller","worker count":1}
```

上記の出力では、新たにデプロイされた Operator が Kubernetes と通信していること、ブローカーおよびアドレス指定のコントローラーが実行されていることと、これらのコントローラーが一部のワーカーを起動していることを確認します。



注記

所定の OpenShift プロジェクトに AMQ Interconnect Operator の **単一のインスタンス** のみをデプロイすることが推奨されます。具体的には、Operator デプロイメントの **replicas** 要素を **1** を超える値に設定することや、同じプロジェクトに Operator を複数回デプロイすることは推奨されていません。

2.4. 基本ブローカーのデプロイ

以下の手順では、AMQ Broker Operator のインストール時に基本的なブローカーインスタンスを OpenShift プロジェクトにデプロイする方法を説明します。

注記

*複数のブローカー CR インスタンスをデプロイすることで、指定の OpenShift プロジェクトに複数のブローカーデプロイメントを作成することはできません。ただし、プロジェクトにブローカーデプロイメントを作成した場合は、アドレスに複数の CR インスタンスをデプロイ**できます**。*

前提条件

- *AMQ Broker 7.3 以降では、新しいバージョンの Red Hat コンテナーレジストリーを使用してコンテナーイメージにアクセスします。この新しいバージョンのレジストリーでは、イメージにアクセスする前に認証されたユーザーである必要がある。本セクションの手順を実行する前に、「Red Hat Container Registry Authentication」で説明されている手順を完了する必要がある。*
- *AMQ Broker Operator がすでにインストールされている。「Installing the AMQ Broker Operator」を参照してください。*

手順

Operator が正常にインストールされると、Operator は実行され、カスタムリソース(CR)に関連する変更をリッスンします。この手順例では、CR を使用して基本的なブローカーをプロジェクトにデプロイする方法を説明します。

1. *ダウンロードして抽出したOperatorアーカイブの **deploy/crs** ディレクトリーで、**broker_v2alpha1_activemqartemis_cr.yaml** ファイルを開きます。このファイルは、基本的なブローカー CR のインスタンスです。*
   ファイルのデフォルトの内容は、以下のようになります。

```
apiVersion: broker.amq.io/v2alpha1
kind: ActiveMQArtemis
metadata:
  name: ex-aao
  application: ex-aao-app
...
spec:
    deploymentPlan:
        size: 2
        image: registry.redhat.io/amq7/amq-broker-lts-rhel7:7.4
```

   size

   *デプロイするブローカーの数を指定します。クラスター化されたデプロイメントの場合、この値は **2** 以上になります。ただし、基本的なブローカーインスタンスの場合は、値を **1** に変更します。*

   image

   *ブローカーの起動に使用するコンテナーイメージを指定します。デフォルトで、CR は **7.4 のフローティングタグを使用します**。floating タグは、CR が Red Hat Container Registry の LTS イメージストリームで利用可能な最新のブローカーコンテナーイメージを使用することを意味します。*

2. ***broker_v2alpha1_activemqartemis** CR に基づいて基本的なブローカーをデプロイします。*

```
$ oc create -f deploy/crs/broker_v2alpha1_activemqartemis_cr.yaml
```

OpenShift Container Platform Web コンソールで **Workloads → Stateful Sets** (OpenShift Container Platform 4.1) または **Applications → Stateful Sets** (OpenShift Container Platform 3.11) をクリックします。**ex-aao-ss** という新しい Stateful Set が表示されます。

ex-aao-ss Stateful Set セクションを展開します。CR の **size** 属性に設定した値に対応する 1 つ以上の Pod があることを確認できます。

各ブローカー Pod の **Events** タブで、ブローカーが起動したことを確認できます。

注記

ブローカーデプロイメントを削除するには、デプロイメント用に作成したカスタムリソースインスタンスを削除します。Stateful Set のみを削除するだけでは不十分です。

関連情報

- 稼働中のブローカーを AMQ Broker 管理コンソールに接続する方法は、「AMQ Broker 管理コンソール へのブローカーの接続」を 参照してください。

2.5. ブローカーデプロイメントの実行へのカスタムリソース変更の適用

以下は、ブローカーデプロイメントの実行にカスタムリソース(CR)の変更の適用について留意すべき点になります。

- CR の **persistenceEnabled** 属性を動的に更新することはできません。この属性を変更するには、クラスターをゼロにスケールダウンします。既存の CR を削除します。次に、変更で CR を再作成し、再デプロイします。また、デプロイメントサイズも指定します。

- CR の **image** 属性で **7.4** などのフローティングタグを使用する場合、デプロイメントは、デプロイメント設定の **imagePullPolicy** 属性または Stateful Set が **Always** に設定されている場合に、Red Hat Container Registry で利用可能な新規イメージバージョンを自動的にプルします。たとえば、デプロイメントで現在ブローカーイメージバージョン **7.4-6 を使用し、新しいブローカーイメージバージョン 7.** 4-7 が利用できる場合には、デプロイメントは、新しいイメージバージョンを自動的にプルし、使用します。新しいイメージを使用するには、デプロイメントの各ブローカーを再起動します。デプロイメントに複数のブローカーがある場合は、各ブローカーを順番に再起動します。

- CR の **deploymentPlan.size** 属性の値は、**oc scale** コマンドによるブローカーデプロイメントのサイズの変更を上書きします。たとえば、**oc scale** を使用してデプロイメントのサイズを **3** つのブローカーから 2 つ変更する場合、CR の **deploymentPlan.size** の値は 3 つになります。この場合、OpenShift はまずデプロイメントを 2 つのブローカーにスケールダウンします。ただし、縮小操作が完了すると、Operator は CR で指定される 3 つのブローカーにデプロイメントを復元します。

- アクティブなスケーリングイベント時に、さらに適用する変更は Operator によってキューに入れられ、スケーリングが完了した場合にのみ実行されます。たとえば、デプロイメントのサイズを 4 つのブローカーから 1 つにスケールダウンする場合などです。次に、縮小が行われる間、ブローカー管理者のユーザー名およびパスワードの値も変更します。この場合、Operator は 1 つのアクティブなブローカーでデプロイメントが実行されるまで、ユーザー名とパスワードの変更をキューに入れます。

- すべてのカスタムリソースの変更 - デプロイメントのサイズを変更したり、アクセプター、コネクター、またはコンソールの **expose** 属性の値を変更することとは別に、既存のブローカー

がスケールダウンしてから元に戻されます。デプロイメントに複数のブローカーがある場合は、1 度に 1 つのブローカーのみを縮小します。

2.6. クライアント接続用の OPERATOR ベースのブローカーデプロイメントの設定

2.6.1. アクセプターの設定

OpenShift デプロイメントでブローカー Pod へのクライアント接続を有効にするには、Pod で**アクセプター** を定義します。アクセプターは、ブローカーが接続を受け入れる方法を定義します。ブローカーのデプロイメントに使用されるカスタムリソース(CR)でアクセプターを定義します。アクセプターを作成する場合は、アクセプターを有効にするメッセージングプロトコルや、これらのプロトコルに使用するブローカー Pod のポートなどの情報を指定します。

以下の手順は、ブローカーデプロイメントの CR で新規アクセプターを定義する方法を示しています。

前提条件

- アクセプターを設定するには、ブローカーデプロイメントは AMQ Broker Operator の LTS バージョンに基づいている必要があります。Operator の LTS バージョンのインストールに関する詳細は、「Installing the AMQ Broker Operator 」を参照してください。
- このセクションの情報は、AMQ Broker Operator をベースとしたブローカーデプロイメントにのみ適用されます。アプリケーションテンプレートを使用してブローカーデプロイメントを作成する場合は、プロトコル固有のアクセプターは定義できません。詳細は、外部クライアントのテンプレートベースのブローカーデプロイメントへの接続に ついて参照してください。

手順

1. 初期インストール時にダウンロードおよび展開した Operator アーカイブの **deploy/crs** ディレクトリーで、**broker_v2alpha1_activemqartemis_cr.yaml** カスタムリソース(CR)を開きます。
2. **acceptors** 要素に名前付きアクセプターを追加します。通常、これらのプロトコルを公開するためにブローカー Pod のアクセプターおよびポートで使用されるプロトコルなどの最小限の属性セットを指定します。以下に例を示します。

```
spec:
...
  acceptors:
  - name: amqp_acceptor
    protocols: amqp
    port: 5672
    sslEnabled: false
...
```

上記の例は、単純な AMQP アクセプターの設定を示しています。アクセプターはポート 5672 を AMQP クライアントに公開します。

2.6.1.1. アクセプター設定の追加の注意点

本セクションでは、アクセプター設定に関する追加の方法を説明します。

- 内部クライアント（ブローカー Pod と同じ OpenShift クラスターのクライアントアプリケー

ション）、または内部および外部クライアント（OpenShift 外部のアプリケーション）のいずれかに対してアクセプターを定義できます。外部クライアントにアクセプターを公開するには、アクセプター設定の **expose** パラメーターを **true** に設定します。このパラメーターのデフォルト値は **false** です。

- 単一のアクセプターは複数のクライアント接続を受け入れ、アクセプター設定の connection **Allowed** パラメーターで指定される最大制限まで使用できます。
- CR でアクセプターを定義しない場合、デプロイメントのブローカー Pod はデフォルトでポート 61616 で作成される単一のアクセプターを使用します。このデフォルトのアクセプターには、Core プロトコルのみが指定されています。
- ポート 8161 は、AMQ Broker 管理コンソールで使用するためにブローカー Pod で自動的に公開されます。OpenShift ネットワーク内では、このポートはブローカーデプロイメントで実行される **ヘッドレス** サービスを介してアクセスできます。詳細は、「ブローカー管理コンソールへのアクセス」を参照してください。
- **sslEnabled** を **true** に設定すると、アクセプターで SSL を有効にできます。以下のような追加情報を指定できます。
  - SSL 認証情報の保存に使用されるシークレット名（必須）。
  - SSL 通信に使用する暗号スイートおよびプロトコル。
  - アクセプターが双方向 SSL を使用するかどうか、つまりブローカーとクライアント間の相互認証。

  定義するアクセプターが SSL を使用する場合は、アクセプターによって使用される SSL 認証情報をシークレットに保存する必要があります。独自のシークレットを作成し、このシークレット名をアクセプター設定の **sslSecret** パラメーターで指定する必要があります。**sslSecret** パラメーターでシークレット名を明示的に指定しない場合、アクセプターはデフォルトのシークレット名を推測します。デフォルトのシークレット名は **<CustomResourceName>-<AcceptorName>-secret 形式を** 使用します。例： **ex-aao-amqp-secret**

  シークレットに必要な SSL 認証情報は **broker.ks** です。base64 でエンコードされたキーストア、**client.ts** でなければなりません。base64 でエンコードされたトラストストア、**keyStorePassword**、および **trustStorePassword**（raw テキストに指定されるパスワード）である必要があります。この要件は、設定するコネクターでも同じです。SSL 接続の認証情報の生成に関する詳細は、「SSL 接続の認証情報の生成」を参照してください。

関連情報

- アクセプターの設定を含むメインのブローカーカスタムリソース定義(CRD)の完全な設定リファレンスは、「カスタムリソース定義」を参照してください。

2.6.2. 内部および外部クライアントからのブローカーへの接続

- 内部クライアントは **<Pod Name>:<AcceptorPortNumber> 形式でアドレスを指定してブローカー Pod に接続できます**。OpenShift DNS は、Operator ベースのブローカーデプロイメントによって作成されたステートフルセットが安定した Pod 名を提供するため、この形式でアドレスが正常に解決されます。
- 外部クライアントにアクセプターを公開すると、専用のサービスとルートが自動的に作成されます。指定のブローカー Pod で設定された Routes を表示するには、OpenShift Container Platform Web コンソールの Pod を選択し、**Routes** タブをクリックします。外部クライアント

は、アクセプター用に作成された Route の完全なホスト名を指定して、ブローカーに接続できます。**curl** コマンドを使用して、この完全なホスト名への外部アクセスをテストできます。以下に例を示します。

```
$ curl https://ex-aao-0-svc-my_project.my_openshift_domain
```

Route の完全なホスト名は、OpenShift ルーターをホストするノードに解決する必要があります。OpenShift ルーターは、ホスト名を使用して、OpenShift 内部ネットワーク内のトラフィックを送信する場所を判別します。

デフォルトでは、OpenShift ルーターは、セキュアでないトラフィック (SSL 以外) トラフィックとポート 443 (SSL で暗号化した) トラフィックに対してポート 80 をリッスンします。HTTP 接続の場合、ルーターはセキュアな接続 URL (**https**) を指定する場合 (**https**) またはポート 80 を指定する場合は、トラフィックをポート 443 に自動的に転送します。

一方、TCP を使用するメッセージングクライアントは、接続 URL の一部としてポート番号を明示的に指定する必要があります。以下に例を示します。

```
tcp://ex-aao-0-svc-my_project.my_openshift_domain:443
```

- ルートを使用する代わりに、OpenShift 管理者は NodePort を OpenShift 外部のクライアントからブローカー Pod に接続するように設定できます。NodePort は、ブローカーに設定されたアクセプターによって指定されるプロトコル固有のポートのいずれかにマップする必要があります。デフォルトで、NodePort は 30000 から 32767 の範囲に置かれます。つまり、NodePort はブローカー Pod の意図されるポートとは一致しません。OpenShift の外部のクライアントから NodePort を介してブローカーに接続するには、URI を < **Protocol>://<OCPNodeIP>:<NodePortNumber>** の形式で指定します。

関連情報

- クラスターで実行されているサービスを使って OpenShift クラスター外からの通信を行うために Routes および NodePort などの方法についての詳細は、以下を参照してください。
  - ingress クラスタートラフィックの設定の概要 (OpenShift Container Platform 4.1 以降)
  - クラスターへのトラフィックの送信 (OpenShift Container Platform 3.11)

2.6.3. SSL 接続の認証情報の生成

SSL 接続では、AMQ Broker にはブローカーキーストア、クライアントキーストア、およびブローカーキーストアが含まれるクライアントトラストストアが必要です。この手順では、認証情報を生成する方法を説明します。この手順では、Java Development Kit に含まれるパッケージである Java Keytool を使用します。

手順

1. ブローカーキーストアの自己署名証明書を生成します。

```
$ keytool -genkey -alias broker -keyalg RSA -keystore broker.ks
```

2. 証明書をエクスポートして、クライアントと共有できるようにします。

```
$ keytool -export -alias broker -keystore broker.ks -file broker_cert
```

3. クライアントキーストアの自己署名証明書を生成します。

```
$ keytool -genkey -alias client -keyalg RSA -keystore client.ks
```

4. ブローカー証明書をインポートするクライアントトラストストアを作成します。

```
$ keytool -import -alias broker -keystore client.ts -file broker_cert
```

5. 以下の例のように、ブローカーキーストアファイルを使用してシークレットを作成し、SSL 認証情報を保存します。

```
$ oc secrets new ex-aao-amqp-secret broker.ks client.ts
```

6. 以下の例のように、シークレットを Operator のインストール時に作成したサービスアカウントに追加します。

```
$ oc secrets add sa/amq-broker-operator secret/ex-aao-amqp-secret
```

2.6.4. ブローカーデプロイメントのネットワークサービス

ブローカーデプロイメントの OpenShift Container Platform Web コンソールの **Networking** ペインで、2 つの実行中のサービスがあり、**ヘッドレス** サービスと ping サービスが 2 つあります。ヘッドレスサービスのデフォルト名は **<Custom Resource name>-hdls-svc** 形式を使用します（例： **ex-aao-hdls-svc** ）。ping サービスのデフォルト名は **<Custom Resource name>-ping-svc** 形式を使用します（例： **ex-aao-ping-svc** ）。

ヘッドレスサービスは、各ブローカー Pod のポート 8161 および 61616 へのアクセスを提供します。ポート 8161 はブローカー管理コンソールに使用され、ポート 61616 はブローカーのクラスタリングに使用されます。

ping サービスは検出のためにブローカーによって使用されるサービスで、ブローカーは OpenShift 環境内でクラスターを形成できるようにします。内部的には、このサービスは 8888 ポートを公開します。

2.6.5. ブローカーの AMQ Broker 管理コンソールへの接続

ブローカーは、ポート 8161 で独自の管理コンソールをホストします。デプロイメントの各ブローカー Pod には、コンソールへのアクセスを提供するサービスとルートがあります。

以下の手順では、稼働中のブローカーインスタンスの AMQ Broker 管理コンソールに接続する方法を説明します。

前提条件

- AMQ Broker Operator を使用して基本的なブローカーをデプロイしている。詳細は、「基本的なブローカーのデプロイ」を参照してください。

2.6.5.1. ブローカー管理コンソールへのアクセス

デプロイメントの各ブローカー Pod には、コンソールへのアクセスを提供するサービスがあります。このサービスのデフォルト名は **<Custom Resource name>-wconsj-<broker Pod ordinal>-svc 形式を使用します。たとえば、デプロイメントのブローカー Pod 0 の場合、サービス名は ex-aao-wconsj-0-svc になり** ます。各サービスには、**'<Custom Resource name>-wconsj-<broker Pod ordinal>-svc-rte** 形式を使用する対応するルートがあります。たとえば、**ex-aao-wconsj-0-svc-rte** です。

この手順では、稼働中のブローカーインスタンスの AMQ Broker 管理コンソールにアクセスする方法を説明します。

手順

1. *OpenShift Container Platform Web コンソールで、**Networking → Routes** (OpenShift Container Platform 4.1) または **Applications → Routes** (OpenShift Container Platform 3.11) をクリックします。*
   ***Routes** ペインで、**wconsj** サービスに対応する Route が表示されます。*
2. ***Hostname** の下の、完全な URL をメモします。コンソールにアクセスするには、この URL を指定する必要があります。*
3. *Web ブラウザーで、ホスト名の URL を入力します。*
   a. *コンソール設定で SSL を使用しない場合は、URL に **http** を指定してください。この場合、ホスト名の DNS が解決されて、トラフィックは OpenShift ルーターのポート 80 に転送されます。*
   b. *コンソール設定で SSL を使用する場合は、URL に **https** を指定します。この場合、ブラウザーはデフォルトで OpenShift ルーターのポート 443 になります。この設定により、OpenShift ルーターが SSL トラフィックにポート 443 も使用する場合には、コンソールに正常に接続できます (これは、ルーターのデフォルト設定)。*
4. *管理コンソールにログインするには、ブローカーデプロイメントカスタムリソースの **adminUser** パラメーターおよび **adminPassword** パラメーターに指定されたユーザー名とパスワードを入力します。**adminUser** および **adminPassword** に値が指定されていない場合は、「管理コンソールのログイン認証情報へのアクセス」の手順に従い、コンソール へのログインに必要な認証情報を取得します。*

2.6.5.2. 管理コンソールのログイン認証情報へのアクセス

*ブローカーのカスタムリソース(CR)に **adminUser** および **adminPassword** の値を指定しない場合、Operator はブローカーユーザー名およびパスワードを自動的に生成し（AMQ Broker 管理コンソールにログインするために必要）、これらの認証情報をシークレットに保存します。デフォルトのシークレット名の形式は &lt **;Custom Resource name>-credentials-secret** です（例： **ex-aao-credentials-secret** ）。*

この手順では、管理コンソールへのログインに必要なログイン認証情報にアクセスする方法を説明します。

手順

1. *OpenShift プロジェクトのシークレットの詳細な一覧を参照してください。*
   a. *OpenShift Container Platform Web コンソールから、**Workload → Secrets** (OpenShift Container Platform 4.1) または **Resources → Secrets** (OpenShift Container Platform 3.11) をクリックします。*
   b. *コマンドラインで以下を行います。*

```
$ oc get secrets
```

2. *適切なシークレットを開き、コンソールログインの認証情報を表示します。*
   a. *OpenShift Container Platform Web コンソールから、名前にブローカーカスタムリソースイ*

インスタンスが含まれるシークレットをクリックします。暗号化されたユーザー名とパスワードの値を表示するには、**YAML** タブ(OpenShift Container Platform 4.1)または **Actions** → **Edit YAML** (OpenShift Container Platform 3.11)をクリックします。

b. コマンドラインで以下を行います。

```
$ oc edit secret <my_custom_resource_name-credentials-secret>
```

2.7. ブローカーのデプロイメント例

2.7.1. クラスター化されたブローカーのデプロイ

2 つ以上のブローカー Pod がプロジェクトで実行されている場合、Pod はブローカークラスターを自動的に形成します。クラスター化の設定により、ブローカーは相互に接続でき、必要に応じてメッセージを再配布できます。

以下の手順では、クラスター化されたブローカーをデプロイする方法を説明します。デフォルトでは、このデプロイメントのブローカーは**オンデマンド**負荷分散を使用します。つまり、ブローカーは一致するコンシューマーを持つ他のブローカーにのみメッセージを転送します。

前提条件

- 基本的なブローカーがすでにデプロイされています。「基本的なブローカーのデプロイ」を参照してください。

手順

1. ダウンロードして抽出したOperatorアーカイブの **deploy/crs** ディレクトリーで、**broker_v2alpha1_activemqartemis_cr.yaml** カスタムリソースファイルを開きます。
2. 最小限のクラスターデプロイメントの場合は、**deploymentPlan.size** の値が **2** であることを確認します。
3. コマンドラインで変更を適用します。

```
$ oc apply -f deploy/crs/broker_v2alpha1_activemqartemis_cr.yaml
```

OpenShift Container Platform Web コンソールでは、CR で指定した追加のブローカーについて 2 つ目の Pod がプロジェクトで起動します。デフォルトでは、プロジェクトで実行している 2 つのブローカーがクラスター化されます。

4. 各 Pod の **Logs** タブを開きます。ログには、OpenShift が各ブローカーでクラスター接続ブリッジが確立されていることが示されています。具体的には、ログ出力には以下のような行が含まれます。

```
targetConnector=ServerLocatorImpl (identity=(Cluster-connection-
bridge::ClusterConnectionBridge@6f13fb88
```

2.7.2. ブローカークラスターでのキューの作成

以下の手順では、カスタムリソース定義(CRD)およびサンプルカスタムリソース(CR)を使用して、Operator を使用してデプロイされたブローカークラスターからキューを追加し、削除する方法を説明します。

前提条件

- ブローカークラスターがすでにデプロイされている。「クラスター化されたブローカーのデプロイ」を参照してください。

手順

1. アドレス指定 CRD をデプロイします。

```
$ oc create -f deploy/crds/broker_v2alpha1_activemqartemisaddress_crd.yaml
```

2. CR ファイルの例 **broker_v2alpha1_activemqartemisaddress_cr.yaml** は、ダウンロードして抽出したOperatorアーカイブに含まれています。カスタムリソースの例には、以下が含まれます。

```
spec:
  # Add fields here
  spec:
    addressName: myAddress0
    queueName: myQueue0
    routingType: anycast
```

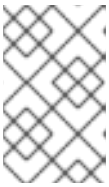
ブローカークラスターが Operator 経由ですでにデプロイおよび実行されている場合、カスタムリソースのサンプルを使用して、クラスター内の稼働しているすべてのブローカーにアドレスを作成します。

```
$ oc create -f deploy/crs/broker_v2alpha1_activemqartemisaddress_cr.yaml
```

CR の例をデプロイすると、任意の **キャスト** ルーティングタイプを持つ **myQueue0** という名前のキューでアドレス **myAddress0** が作成されます。このアドレスは実行中のブローカーすべてに作成されます。

注記

ブローカークラスターに複数のアドレスやキューを作成するには、個別の CR ファイルを作成し、それらを個別にデプロイし、各ケースに新しいアドレスやキュー名を指定する必要があります。



注記

アドレス指定 CR のデプロイ後にブローカーをクラスターに追加する場合、新規ブローカーには以前に作成したアドレスがありません。この場合、アドレスを削除し、アドレス指定 CR を再デプロイする必要があります。

3. サンプル CR から作成されたキューを削除するには、以下のコマンドを使用します。

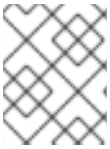
```
$ oc delete -f deploy/crs/broker_v2alpha1_activemqartemisaddress_cr.yaml
```

2.8. スケールダウン時のメッセージの移行

ブローカーデプロイメントの縮小時にメッセージを移行するには、メインのブローカーカスタムリソース定義(CRD)を使用してメッセージの移行を有効にします。AMQ Broker Operator は、クラスター化されたブローカーデプロイメントをスケールダウンする際に、専用のスケールダウンコントローラーを実

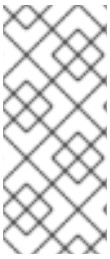
行し、メッセージの移行を実行します。

メッセージ移行を有効にすると、Operator 内のスケールダウンコントローラーがブローカー Pod のシャットダウンを検出し、ドレイン Pod を開始し、メッセージ移行を実行します。ドレイナー Pod はクラスター内の他のライブブローカー Pod のいずれかに接続し、メッセージをそのライブブローカー Pod に移行します。移行が完了すると、スケールダウンコントローラーがシャットダウンします。



注記

縮小コントローラーは、単一の OpenShift プロジェクト内でのみ機能します。コントローラーは、別のプロジェクトのブローカー間でメッセージを移行できません。



注記

ブローカーデプロイメントを 0 (ゼロ) にスケールダウンする場合、メッセージングデータを移行できる稼働中のブローカー Pod がないため、メッセージ移行は行われません。ただし、デプロイメントをゼロブローカーにスケールダウンし、元のデプロイメントに含まれていた一部のブローカーのみに戻っても、シャットダウンされたブローカーのドレイン Pod が起動します。

以下の手順の例は、スケールダウンコントローラーの動作を示しています。

前提条件

- 基本的なブローカーデプロイメントがすでにある。「基本的なブローカーのデプロイ」を参照してください。
- メッセージの移行の仕組みを理解している。詳細は、「メッセージの移行」を参照してください。

手順

1. 最初にダウンロードして展開した Operator リポジトリーの **deploy/crs** ディレクトリーで、メインブローカー CR **broker_v2alpha1_activemqartemis_cr.yaml** を開きます。

メインブローカー CR では、messageMigration および persistenceEnabled を true に設定します。

+ これらの設定は、クラスターブローカーデプロイメントのサイズを後でスケールダウンすると、Operator はスケールダウンコントローラーを自動的に起動し、メッセージを実行中のブローカー Pod に移行することを意味します。

1. 既存のブローカーデプロイメントで、実行中の Pod を確認します。

```
$ oc get pods
```

以下のような出力が表示されます。

```
activemq-artemis-operator-8566d9bf58-9g25l   1/1   Running   0   3m38s
ex-aao-ss-0                                  1/1   Running   0   112s
ex-aao-ss-1                                  1/1   Running   0   8s
```

上記の出力では、3 つの Pod が実行されていることが示されています。1 つはブローカー Operator 自体用で、デプロイメントの各ブローカーに個別の Pod が実行されていることを示しています。

2. 各 Pod にログインし、各ブローカーにメッセージを送信します。

   a. Pod **ex-aao-ss-0** にクラスター IP アドレスが **172.17.0.6** である場合は、以下のコマンドを実行します。

   ```
   $ /opt/amq-broker/bin/artemis producer --url tcp://172.17.0.6:61616 --user admin --password admin
   ```

   b. Pod **ex-aao-ss-1** にクラスター IP アドレスが **172.17.0.7** である場合は、以下のコマンドを実行します。

   ```
   $ /opt/amq-broker/bin/artemis producer --url tcp://172.17.0.7:61616 --user admin --password admin
   ```

   前述のコマンドは、各ブローカーに **TEST** というキューを作成し、各キューに 1000 個のメッセージを追加します。

3. クラスターを 2 つのブローカーにスケールダウンします。

   a. メインブローカー CR **broker_v2alpha1_activemqartemis_cr.yaml** を開きます。

   b. CR で、**deploymentPlan.size** を **1** に設定します。

   c. コマンドラインで変更を適用します。

   ```
   $ oc apply -f deploy/crs/broker_v2alpha1_activemqartemis_cr.yaml
   ```

   Pod **ex-aao-ss-1** がシャットダウンを開始したことを確認します。縮小コントローラーは、同じ名前の新しいドレイン Pod を起動します。このドレイン Pod は、ブローカー Pod **ex-aao-ss-1** からクラスター内の他のブローカー Pod にすべてのメッセージを移行した後にシャットダウンします (**ex-aao-ss-0**)。

4. ドレイン Pod がシャットダウンされたら、ブローカー Pod **ex-aao-ss-0** の **TEST** キューのメッセージ数を確認します。キューのメッセージ数が 2000 であることを確認できます。これは、ドレイン Pod がシャットダウンするブローカー Pod から 1000 個のメッセージを正常に移行しました。

2.9. OPERATOR LIFECYCLE MANAGER を使用したブローカー OPERATOR の管理

2.9.1. Operator Lifecycle Manager の概要

OpenShift Container Platform 4.0 以降では、Operator Lifecycle Manager(OLM) を使用することにより、ユーザーはすべての Operator およびクラスター全体で実行される関連サービスをインストールし、更新し、一般的に管理することができます。これは、Kubernetes のネイティブアプリケーション (Operator) を効率的に自動化された拡張可能な方法で管理するために設計されたオープンソースツールキットの Operator Framework の一部です。

OLM は OpenShift Container Platform 4.0 でデフォルトで実行されます。これは、クラスター管理者がクラスターで実行されている Operator をインストールし、アップグレードし、そのアクセス権限を付与するのに役立ちます。OpenShift Container Platform Web コンソールでは、クラスター管理者が

Operator をインストールし、特定のプロジェクトアクセスを付与して、クラスターで利用可能な Operator のカタログを使用するための管理画面を利用できます。

OperatorHub は、OpenShift クラスター管理者が Operator を検出、インストール、およびアップグレードするために使用するグラフィカルインターフェースです。1 回のクリックで、これらの Operator を OperatorHub からプルし、クラスターにインストールし、OLM で管理して、エンジニアリングチームが開発環境、テスト環境、および本番環境でソフトウェアをセルフサービスで管理できるようにします。

OperatorHub に AMQ Broker Operator をインストールする場合、グラフィカルインターフェースを使用して、スタンドアロンブローカー、ブローカークラスター、スケールダウンコントローラーを含むクラスターなどの各種のブローカーデプロイメントを作成できます。

2.9.2. OperatorHub での AMQ Broker Operator のインストール

OperatorHub で自動的に利用可能な Operator for AMQ Broker 7.4 の最新の LTS バージョンが表示されない場合は、以下の手順を実行して OperatorHub で Operator を手動でインストールします。

手順

1. Web ブラウザーで、AMQ Broker **Software Downloads** ページ に移動します。
2. **Version** ドロップダウンメニューで **7.4.6** を選択します。
3. **AMQ Broker 7.4.6 Operator Installation and Example Files**の横にある **Download** をクリックします。
   amq-broker-operator-7.4.6-ocp-install-examples.zip 圧縮アーカイブのダウンロードが自動的に開始されます。
4. ダウンロードが完了したら、アーカイブを選択したインストールディレクトリーに移動します。以下の例では、アーカイブを **/broker/operator** というディレクトリーに移動します。

```
sudo mv amq-broker-operator-7.4.6-ocp-install-examples.zip /broker/operator
```

5. 選択したインストールディレクトリーで、アーカイブの内容を展開します。以下に例を示します。

```
cd /broker/operator
unzip amq-broker-operator-7.4.6-ocp-install-examples.zip
```

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

```
$ oc login -u system:admin
```

7. ダウンロードして抽出したOperatorアーカイブの **deploy** ディレクトリーから、AMQ Broker Operator ソースバンドルをデプロイします。

```
$ oc create -f deploy/catalog_resources/courier/amq-broker-operatorsource.yaml
```

数分後に、OpenShift Container Platform Web コンソールの **OperatorHub** セクションで AMQ Broker Operator を利用できます。

第3章 アプリケーションテンプレートを使用した AMQ BROKER の OPENSHIFT CONTAINER PLATFORM へのデプロイ

本セクションの手順では、ブローカーデプロイメントを準備し、OpenShift Container Platform Web コンソールを使用して基本的なブローカーインスタンスをデプロイする方法を説明します。他のブローカー設定のデプロイ例については、チュートリアル を参照してください。

注記

以下の手順では、「OpenShift Container Platform での AMQ Broker のインストール」でインストール したブローカーイメージおよびアプリケーションテンプレートがグローバル **openshift** プロジェクトにあることを前提としています。イメージおよびアプリケーションテンプレートを特定のプロジェクトの名前空間にインストールしている場合、**amq-demo** などの新規プロジェクトを作成するのではなく、このプロジェクトを引き続き使用します。

3.1. AMQ BROKER イメージストリームおよびアプリケーションテンプレートのインストール

AMQ Broker on OpenShift Container Platform イメージストリームおよびアプリケーションテンプレートはデフォルトで OpenShift Container Platform では利用できません。このセクションの手順に従って手動でインストールする必要があります。手動インストールが完了したら、選択したブローカー設定を OpenShift クラスターにデプロイできるようにするテンプレートをインスタンス化できます。このように各種のブローカー設定を作成する例については、「Deploying AMQ Broker on OpenShift Container Platform and Tutorials 」を参照してください。

手順

1. コマンドラインで、クラスター管理者 (またはグローバル **openshift** プロジェクト名前空間に対する名前空間固有の管理者アクセスを持つユーザー) として OpenShift にログインします。

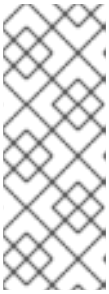
```
$ oc login -u system:admin
$ oc project openshift
```

openshift プロジェクトを使用すると、この手順の後半でインストールするイメージストリームおよびアプリケーションテンプレートは、OpenShift クラスターのすべてのプロジェクトでグローバルに利用できるようになります。イメージストリームとアプリケーションテンプレートが **openshift** プロジェクトにインポートされるように明示的に指定する場合は、オプションのパラメーターとして、この手順で後で使用する **oc replace** コマンドに **-n openshift** を追加することもできます。

openshift プロジェクトを使用する代わりに (クラスター管理者が利用できない場合など)、以下のようにブローカーデプロイメントを作成する特定の OpenShift プロジェクトにログインします。

```
$ oc login -u <USERNAME>
$ oc project <PROJECT_NAME>
```

特定のプロジェクトにログインすると、この手順の後半でインストールするイメージストリームとテンプレートが、そのプロジェクトの名前空間でのみ利用可能になります。



注記

OpenShift Container Platform 上の AMQ Broker は、***-persistence*.yaml** テンプレートすべてで StatefulSet リソースを使用します。***-persistence*.yaml** ではないテンプレートでは、AMQ Broker は Deployment リソースを使用します。どちらのタイプのリソースは、テンプレートがインスタンス化される同じプロジェクト名前空間から**のみ**イメージストリームを使用できる Kubernetes ネイティブリソースです。

2. コマンドラインで以下のコマンドを実行し、ブローカーイメージストリームをプロジェクト名前空間にインポートします。**--force** オプションを **oc replace** コマンドに使用してリソースを更新するか、存在しない場合は作成します。

```
$ oc replace --force  -f \
https://raw.githubusercontent.com/jboss-container-images/jboss-amq-7-broker-openshift-image/74-7.4.6.GA/amq-broker-7-image-streams.yaml
```

3. 以下のコマンドを実行して AMQ Broker アプリケーションテンプレートを更新します。

```
$ for template in amq-broker-74-basic.yaml \
amq-broker-74-ssl.yaml \
amq-broker-74-custom.yaml \
amq-broker-74-persistence.yaml \
amq-broker-74-persistence-ssl.yaml \
amq-broker-74-persistence-clustered.yaml \
amq-broker-74-persistence-clustered-ssl.yaml;
 do
 oc replace --force -f \
https://raw.githubusercontent.com/jboss-container-images/jboss-amq-7-broker-openshift-image/74-7.4.6.GA/templates/${template}
 done
```

3.2. テンプレートベースのブローカーデプロイメントの準備

前提条件

- Broker インスタンスを OpenShift Container Platform にデプロイする前に、AMQ Broker イメージストリームおよびアプリケーションテンプレートをインストールする必要があります。詳細は、「Installing the AMQ Broker image streams and application templates 」を参照してください。

手順

1. コマンドプロンプトを使用して新しいプロジェクトを作成します。

```
$ oc new-project amq-demo
```

2. AMQ Broker デプロイメントに使用するサービスアカウントを作成します。

```
$ echo '{"kind": "ServiceAccount", "apiVersion": "v1", "metadata": {"name": "amq-service-account"}}' | oc create -f -
```

3. view ロールをサービスアカウントに追加します。view ロールにより、サービスアカウントは

amq-demo namespace のすべてのリソースを表示できます。これは、ブローカークラスターエンドポイントの検出に OpenShift dns-ping プロトコルを使用する場合に、クラスターを管理するために必要です。

```
$ oc policy add-role-to-user view system:serviceaccount:amq-demo:amq-service-account
```

4. *AMQ Broker には、ブローカーキーストア、クライアントキーストア、およびブローカーキーストアが含まれるクライアントトラストストアが必要です。この例では、Java Development Kit に含まれるパッケージである Java Keytool を使用して、AMQ Broker インストールで使用するダミーの認証情報を生成します。*

   a. *ブローカーキーストアの自己署名証明書を生成します。*

   ```
   $ keytool -genkey -alias broker -keyalg RSA -keystore broker.ks
   ```

   b. *証明書をクライアントと共有できるようにエクスポートします。*

   ```
   $ keytool -export -alias broker -keystore broker.ks -file broker_cert
   ```

   c. *クライアントキーストア用に自己署名証明書を生成します。*

   ```
   $ keytool -genkey -alias client -keyalg RSA -keystore client.ks
   ```

   d. *ブローカー証明書をインポートするクライアントトラストストアを作成します。*

   ```
   $ keytool -import -alias broker -keystore client.ts -file broker_cert
   ```

   e. *ブローカーのキーストアファイルを使用して、AMQ Broker シークレットを作成します。*

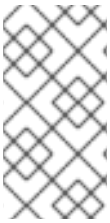
   ```
   $ oc create secret generic amq-app-secret --from-file=broker.ks
   ```

   f. *シークレットを先に作成したサービスアカウントに追加します。*

   ```
   $ oc secrets add sa/amq-service-account secret/amq-app-secret
   ```

3.3. 基本ブローカーのデプロイ

本セクションの手順では、一時である基本ブローカーをデプロイし、SSL をサポートしない基本的なブローカーをデプロイする方法を説明します。



注記

このブローカーは SSL に対応せず、外部クライアントにアクセスできません。OpenShift クラスターの内部で実行しているクライアントのみがブローカーに接続できます。SSL をサポートするブローカー設定の作成例は、「チュートリアル」を参照してください。

前提条件

- *ブローカーデプロイメントがすでに準備済みである。「AMQ Broker デプロイメントの準備」を参照してください。*

- *AMQ Broker 7.3 以降では、新しいバージョンの Red Hat コンテナーレジストリーを使用してコンテナーイメージにアクセスします。この新しいバージョンのレジストリーでは、イメージにアクセスして OpenShift プロジェクトにプルする前に、認証されたユーザーである必要があります。このセクションの手順を実施する前に、「Red Hat コンテナーレジストリーの認証」で説明されている手順を完了する必要があります。*

- *AMQ Broker 7.4 は、Long Term Support (LTS) リリースバージョンとして指定されています。バグ修正およびセキュリティーアドバイザリーは、少なくとも 12 カ月間、一連のマイクロリリース（7.4.1、7.42、7.3 など）で AMQ Broker 7.4 で利用可能になります。つまり、新しいマイナーリリースにアップグレード**することなく**、AMQ Broker の最新のバグ修正およびセキュリティーアドバイザリーを取得できます。*

- *サポート対象の設定を維持するには、LTS リリースストリームに最新のマイクロリリースをインストールする必要があります。つまり、ブローカーデプロイメントの LTS ストリームから最新のブローカーコンテナーイメージを使用する必要があります。このセクションでは、アプリケーションテンプレートに基づいてブローカーデプロイメントの最新ブローカーコンテナーイメージを指定する方法を説明します。*

3.3.1. ブローカーアプリケーションの作成

手順

1. ***amq-demo** プロジェクトスペースまたはブローカーをデプロイする既存のプロジェクトにログインします。*

```
$ oc login -u <USER_NAME>
$ oc project <PROJECT_NAME>
```

2. *基本的なブローカーのテンプレートに基づいて、新しいブローカーアプリケーションを作成します。このテンプレートによって作成されるブローカーは一時的なもので、SSL はサポートしません。*

```
$ oc new-app --template=amq-broker-74-basic \
  -p AMQ_PROTOCOL=openwire,amqp,stomp,mqtt,hornetq \
  -p AMQ_QUEUES=demoQueue \
  -p AMQ_ADDRESSES=demoTopic \
  -p AMQ_USER=amq-demo-user \
  -p AMQ_PASSWORD=password \
```

基本的なブローカーアプリケーションテンプレートは、以下の表に記載されている環境変数を設定します。

表3.1 基本ブローカーアプリケーションテンプレート

環境変数	表示名	値	説明
AMQ_PROTOCOL	AMQ プロトコル	openwire,amqp,stomp,mqtt,hornetq	ブローカーによって使用されるプロトコル
AMQ_QUEUES	キュー	demoQueue	demoQueue という名前のキャストキューを作成します。

環境変数	表示名	値	説明
AMQ_ADDRESSES	アドレス	demoTopic	demoTopic という名前のアドレス (またはトピック) を作成します。デフォルトでは、このアドレスには割り当てられたルーティングタイプが割り当てられていません。
AMQ_USER	AMQ ユーザー名	amq-demo-user	クライアントが使用するユーザー名
AMQ_PASSWORD	AMQ パスワード	パスワード	クライアントがユーザー名で使用するパスワード

3.3.2. ブローカーアプリケーションのデプロイおよび起動

ブローカーアプリケーションの作成後に、これをデプロイする必要があります。アプリケーションのデプロイにより、ブローカーが実行される Pod が作成されます。

手順

1. *OpenShift Container Platform Web コンソールで **Deployments** をクリックします。*
2. ***broker-amq** アプリケーションをクリックします。*
3. ***デプロイ** をクリックします。*

注記

*アプリケーションがデプロイされない場合、**Events** タブをクリックして設定を確認できます。正しくない場合には、**Actions** ボタンをクリックしてデプロイメント設定を編集します。*

4. *ブローカーアプリケーションのデプロイ後に、ブローカー Pod の現在の状態を確認します。*
   a. ***Deployment Configs** をクリックします。*
   b. ***broker-amq** Pod をクリックした後、**Logs** タブをクリックしてブローカーの状態を確認します。アプリケーションテンプレートで以前に作成されたキューが表示されるはずです。ログに以下が表示される場合:*
      - *ブローカーが稼働しているため、この手順のステップ 9 に進みます。*
      - *ブローカーログが読み込まれておらず、Pod のステータスは **ErrImagePull** または **ImagePullBackOff** を表示し、デプロイメント設定は Red Hat Container Registry から指定されたブローカーイメージを直接プルできませんでした。この手順では、ステップ 5 に進みます。*
5. *ブローカーコンテナーイメージのインストール用に Pod を準備するには、実行されているブローカーの数を **0** にスケーリングします。*
   a. ***Deployment Configs** → **broker-amq** をクリックします。*

b. **Actions → Edit Deployment Configs** をクリックします。

c. デプロイメントの **.yaml** ファイルで、**replicas** 属性の値を **0** に設定します。

d. **Save** をクリックします。

e. Pod は、稼働しているブローカーインスタンスがゼロで再起動します。

6. 最新のブローカーコンテナーイメージをインストールします。

a. Web ブラウザーで Red Hat Container Catalog に移動します。

b. 検索ボックスに **AMQ Broker LTS** を入力します。**Search** をクリックします。

c. 検索結果で **AMQ Broker LTS** をクリックします。amq7 /**amq-broker-lts-rhel7** リポジトリーが開き、**最新** のイメージタグが自動的に選択されます。

d. **Get this image** タブをクリックします。

e. **レジストリートークンの認証** の下で、「**OpenShift シークレットの使用**」セクションの「ページ」の手順を確認してください。この手順では、ブローカーコンテナーイメージと、Red Hat Container Registry の Pod デプロイメント設定ファイルへの認証に使用されるアカウントに関連付けられたイメージプルシークレット名を指定する方法について説明します。
たとえば、**amq-demo** プロジェクト namespace の **broker-amq** デプロイメント設定でブローカーコンテナーイメージおよびプルシークレットを参照するには、以下のような行が含まれます。

```
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
..
metadata:
  name: broker-amq
  namespace: amq-demo
..
  spec:
    containers:
        name: broker-amq
        image: 'registry.redhat.io/amq7/amq-broker-lts-rhel7:7.4'
    ..
    imagePullSecrets:
      - name: {PULL-SECRET-NAME}
```

f. **Save** をクリックします。

7. 最新のブローカーイメージバージョンをプロジェクトの名前空間にインポートします。以下に例を示します。

```
$ oc import-image amq7/amq-broker-lts-rhel7:7.4 --from=registry.redhat.io/amq7/amq-broker-lts-rhel7 --confirm
```

8. 前述のように **broker-amq** デプロイメント設定を再度編集します。**replicas** 属性の値を元の値に設定します。
ブローカー Pod は、新規ブローカーイメージを参照する実行中のブローカーすべてで再起動します。

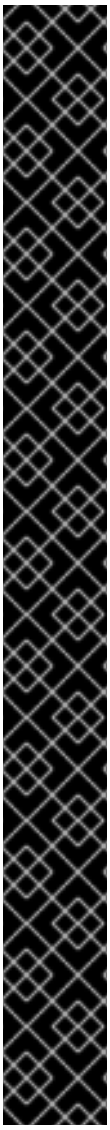
9. *Terminal タブをクリックして、ブローカーを起動し、CLI を使用してメッセージの送受信をテストすることができるシェルにアクセスします。*

```
sh-4.2$ ./broker/bin/artemis run
sh-4.2$ ./broker/bin/artemis producer --destination queue://demoQueue
Producer ActiveMQQueue[demoQueue], thread=0 Started to calculate elapsed time ...

Producer ActiveMQQueue[demoQueue], thread=0 Produced: 1000 messages
Producer ActiveMQQueue[demoQueue], thread=0 Elapsed time in second : 4 s
Producer ActiveMQQueue[demoQueue], thread=0 Elapsed time in milli second : 4584 milli
seconds
sh-4.2$ ./broker/bin/artemis consumer --destination queue://demoQueue
Consumer:: filter = null
Consumer ActiveMQQueue[demoQueue], thread=0 wait until 1000 messages are consumed
Received 1000
Consumer ActiveMQQueue[demoQueue], thread=0 Consumed: 1000 messages
Consumer ActiveMQQueue[demoQueue], thread=0 Consumer thread finished
```

または、以下の例のように、OpenShift クライアントを使用して Pod 名を使用してシェルにアクセスします。

```
// Get the Pod names and internal IP Addresses
$ oc get pods -o wide

// Access a broker Pod by name
$ oc rsh <broker-pod-name>
```

第4章 OPENSHIFT CONTAINER PLATFORM での AMQ BROKER のアップグレード



重要

- AMQ Broker 7.4 は、Long Term Support (LTS) リリースバージョンとして指定されています。バグ修正およびセキュリティーアドバイザリーは、少なくとも 12 カ月間、一連のマイクロリリース（7.4.1、7.42、7.3 など）で AMQ Broker 7.4 で利用可能になります。つまり、新しいマイナーリリースにアップグレード**することなく**、AMQ Broker の最新のバグ修正およびセキュリティーアドバイザリーを取得できます。
- サポート対象の設定を維持するには、LTS リリースストリームの最新マイクロリリースにアップグレードする必要があります。
  - Operator を最新の LTS バージョンに更新する方法については、「Operator のアップグレード」を参照し てください。
  - テンプレートベースのデプロイメントを更新して、LTS ストリームで最新のブローカーコンテナーイメージを指定するには、「テンプレートベースのデプロイメントのブローカーコンテナーイメージのアップグレード」を参照してください。
- OpenShift Container Platform 3.11 の既存の AMQ Broker デプロイメントを OpenShift Container Platform 4.1 で実行するには、まず OpenShift Container Platform インストールをアップグレードしてから、既存のデプロイメントに一致する AMQ Broker のクリーンインストールを実行する必要があります。AMQ Broker をクリーンインストールするには、以下のいずれかの方法を使用します。
  - Operator を使用した OpenShift Container Platform への AMQ Broker のデプロイ （推奨）。
  - アプリケーションテンプレートを使用した AMQ Broker の OpenShift Container Platform へのデプロイ

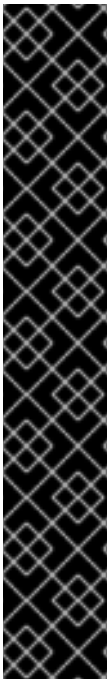
4.1. OPERATOR ベースのブローカーデプロイメントのアップグレード

このセクションでは、OpenShift プロジェクトで使用される Operator バージョンを更新する方法について説明します。

4.1.1. Operator のアップグレード

AMQ Broker Operator のバージョン 0.6 はテクノロジープレビュー機能としてのみご利用いただけます。Operator のバージョン 0.6 が OpenShift プロジェクトにインストールされている場合、Operator を最新の LTS(Long Term Support)バージョンに更新することが推奨されます。Operator の最新の LTS バージョンには、バグおよびセキュリティーアドバイザリーの修正が含まれます。Red Hat は、実稼働環境での使用のために Operator の LTS バージョンをサポートします。

以下では、Operator のテクニカルプレビューバージョンを最新の LTS バージョンにアップグレードする際に注意すべき重要な事項を説明します。



重要

- *Operator のバージョン 0.6 によって使用されるカスタムリソース定義(CRD)は、Long Term Support(LTS)バージョンと互換性がありません。このため、Operator のシームレスなアップグレードを実行できません。Operator をバージョン 0.6 からアップグレードするには、OpenShift クラスターに以前にデプロイされた CRD を削除する必要があります。また、Operator の LTS バージョンをインストールするプロジェクトから既存の Operator およびブローカーデプロイメントを削除する必要もあります。*

- *最新の CRD で OpenShift クラスターを更新すると、この更新はクラスター内の **すべてのプロジェクト** に影響します。バージョン 0.6 から以前にデプロイされたブローカー Pod は機能しなくなりました。OpenShift クラスターの影響を受ける各プロジェクトを、Operator の LTS バージョンを使用するように更新する必要があります。その後、Operator の LTS バージョンに含まれるカスタムリソース(CR)をデプロイして、以前のブローカーデプロイメントを再作成できます。*

- *新規 CR をデプロイして以前のブローカーデプロイメントを再作成する場合、CR の LTS ストリームに最新のブローカーコンテナーイメージを指定できます。*

Operator の LTS バージョンをインストールし、新しいブローカーデプロイメントを作成する方法は、「Operator を使用した OpenShift Container Platform での AMQ Broker のデプロイ」を参照してください。

4.2. テンプレートベースのブローカーデプロイメントのアップグレード

以下の手順では、アプリケーションテンプレートに基づいてブローカーデプロイメントのブローカーコンテナーイメージをアップグレードする方法を説明します。

4.2.1. 永続的でないブローカーデプロイメントのアップグレード

この手順では、永続的ではないブローカーデプロイメントをアップグレードする方法を説明します。OpenShift Container Platform サービスカタログの永続ではないブローカーテンプレートには、以下のようなラベルがあります。

- *Red Hat AMQ Broker 7.x(Ephemeral, no SSL)*
- *Red Hat AMQ Broker 7.x(Ephemeral, with SSL)*
- *Red Hat AMQ Broker 7.x（Custom Config、Ephemeral、no SSL）*

前提条件

- *AMQ Broker 7.3 以降では、新しいバージョンの Red Hat コンテナーレジストリーを使用してコンテナーイメージにアクセスします。この新しいバージョンのレジストリーでは、イメージにアクセスして OpenShift プロジェクトにプルする前に、認証されたユーザーである必要があります。このセクションの手順を実施する前に、「Red Hat コンテナーレジストリーの認証」で説明されている手順を完了する必要があります。*

手順

1. *OpenShift Container Platform Web コンソールに移動し、ログインします。*
2. *永続的ではないブローカーデプロイメントをアップグレードするプロジェクトをクリックします。*

3. ブローカーデプロイメントに対応する Deployment Config(DC) を選択します。

   a. OpenShift Container Platform 4.1 で、**Workloads → Deployment Configs** をクリックします。

   b. OpenShift Container Platform 3.11 で、**Applications → Deployments** をクリックします。ブローカーのデプロイメント内で、**Configuration** タブをクリックします。

4. **Actions** メニューから **Edit Deployment Config** (OpenShift Container Platform 4.1) または **Edit YAML** (OpenShift Container Platform 3.11) をクリックします。
   Deployment Config の **YAML** タブが開き、**.yaml** ファイルが編集可能なモードで開きます。

5. **image** 属性を編集して、Long Term Support(LTS) イメージストリームに最新のブローカーコンテナーイメージを指定します。

```
...
spec:
   containers:
        image: 'registry.redhat.io/amq7/amq-broker-lts-rhel7:7.4'
```

注記

イメージ 属性（**7**.4 **-6** などの特定のバージョン **ID** を持つタグではなく）に **7.4**などのタグ形式を指定する場合、このタグ形式は **Floating タグ** として知られています。Floating タグを指定する場合、OpenShift Container Platform は指定されたリポジトリーで利用可能な最新のイメージを自動的に識別し、このイメージを使用してブローカーデプロイメントをアップグレードします。

6. **imagePullSecrets** 属性を追加し、Red Hat コンテナーレジストリーで認証に使用するアカウントに関連付けられたイメージプルシークレットを指定します。

```
...
spec:
   containers:
        image: 'registry.redhat.io/amq7/amq-broker-lts-rhel7:7.4'
..
imagePullSecrets:
  - name: {PULL-SECRET-NAME}
```

7. **Save** をクリックします。
   現在インストールされているものよりも新しいブローカーイメージが Red Hat Container レジストリーで利用可能な場合、OpenShift Container Platform はブローカーデプロイメントをアップグレードします。これを実行するには、OpenShift Container Platform は既存ブローカー Pod を停止し、新規イメージを使用する新規 Pod を起動します。

4.2.2. 永続的なブローカーデプロイメントのアップグレード

この手順では、永続的なブローカーデプロイメントをアップグレードする方法を説明します。
OpenShift Container Platform サービスカタログの永続ブローカーテンプレートには、以下のようなラベルがあります。

- Red Hat AMQ Broker 7.x (Persistence, clustered, no SSL)
- Red Hat AMQ Broker 7.x (Persistence, clustered, with SSL)

- *Red Hat AMQ Broker 7.x (Persistence, with SSL)*

前提条件

- *AMQ Broker 7.3 以降では、新しいバージョンの Red Hat コンテナーレジストリーを使用してコンテナーイメージにアクセスします。この新しいバージョンのレジストリーでは、イメージにアクセスして OpenShift プロジェクトにプルする前に、認証されたユーザーである必要があります。このセクションの手順を実施する前に、「Red Hat コンテナーレジストリーの認証」で説明されている手順を完了する必要があります。*

手順

1. *OpenShift Container Platform Web コンソールに移動し、ログインします。*
2. *永続的なブローカーデプロイメントをアップグレードするプロジェクトをクリックします。*
3. *ブローカーのデプロイメントに対応する StatefulSet(SS) を選択します。*
   a. *OpenShift Container Platform 4.1 で、**Workloads → Stateful Sets** をクリックします。*
   b. *OpenShift Container Platform 3.11 で、**Applications → Stateful Sets** をクリックします。*
4. ***Actions** メニューから **Edit Stateful Set** (OpenShift Container Platform 4.1) または **Edit YAML** (OpenShift Container Platform 3.11) をクリックします。*
   *StatefulSet の **YAML** タブが開き、編集可能なモードの **.yaml** ファイルが表示されます。*
5. *ブローカーのデプロイメントをアップグレードする準備をするには、デプロイメントをゼロのブローカーにスケールダウンします。*
   a. ***replicas** 属性が現在 **1** 以上に設定されている場合は、**0** に設定します。*
   b. ***Save** をクリックします。*
6. *すべてのブローカー Pod がシャットダウンしたら、Stateful Set **.yaml** ファイルを再度編集します。**image** 属性を編集して、LTS イメージストリームに最新のブローカーコンテナーイメージを指定します。*

```
...
spec:
   containers:
        image: 'registry.redhat.io/amq7/amq-broker-lts-rhel7:7.4'
```

注記

***イメージ** 属性（**7**.4 **-6 などの特定のバージョン ID を持つタグではなく）に **7.4**などのタグ形式を指定する場合、このタグ形式は* **Floating タグ** *として知られています。Floating タグを指定する場合、OpenShift Container Platform は指定されたリポジトリーで利用可能な最新のイメージを自動的に識別し、このイメージを使用してブローカーデプロイメントをアップグレードします。*

7. ***imagePullSecrets** 属性を追加し、Red Hat コンテナーレジストリーで認証に使用するアカウントに関連付けられたイメージプルシークレットを指定します。*

```
...
spec:
```

```
    containers:
        image: 'registry.redhat.io/amq7/amq-broker-lts-rhel7:7.4'
..
imagePullSecrets:
  - name: {PULL-SECRET-NAME}
```

8. ***replicas*** *属性を元の値に設定します。*

9. ***Save*** *をクリックします。*
   現在インストールされているものよりも新しいブローカーイメージが Red Hat Container レジストリーで利用可能な場合、OpenShift Container Platform はブローカーデプロイメントをアップグレードします。これを実行するには、OpenShift Container Platform はブローカー Pod を再起動します。

第5章 外部クライアントのテンプレートベースのブローカーデプロイメントへの接続

5.1. SSL の設定

最小の SSL 設定で OpenShift Container Platform 外の接続を許可するには、AMQ Broker にはブローカーキーストア、クライアントキーストア、およびブローカーキーストアが含まれるクライアントトラストストアが必要です。ブローカーキーストアは、サービスアカウントに追加される OpenShift Container Platform イメージにある AMQ Broker のシークレット作成にも使用されます。

以下のコマンド例では、Java Development Kit に含まれるパッケージ Java KeyTool を使用して、必要な証明書とストアを生成します。

SSL をサポートするブローカーインスタンスをデプロイする詳細な例は、SSL を使用した基本的なブローカーのデプロイ を参照してください。

手順

1. ブローカーキーストアの自己署名証明書を生成します。

```
$ keytool -genkey -alias broker -keyalg RSA -keystore broker.ks
```

2. 証明書をクライアントと共有できるようにエクスポートします。

```
$ keytool -export -alias broker -keystore broker.ks -file broker_cert
```

3. クライアントキーストア用に自己署名証明書を生成します。

```
$ keytool -genkey -alias client -keyalg RSA -keystore client.ks
```

4. ブローカー証明書をインポートするクライアントトラストストアを作成します。

```
$ keytool -import -alias broker -keystore client.ts -file broker_cert
```

5. キーストアからクライアントの証明書をエクスポートします。

```
$ keytool -export -alias client -keystore client.ks -file client_cert
```

6. クライアントのエクスポートされた証明書をブローカーの SERVER トラストストアにインポートします。

```
$ keytool -import -alias client -keystore broker.ts -file client_cert
```

5.2. AMQ BROKER シークレットの生成

ブローカーキーストアを使用して namespace のシークレットを生成します。このシークレットは、アプリケーションの認証を可能にするためにサービスアカウントに追加されます。

手順

- コマンドラインで、次のコマンドを実行します。

```
$ oc create secret generic <secret-name> --from-file=<broker-keystore> --from-file=<broker-truststore>
$ oc secrets add sa/<service-account-name> secret/<secret-name>
```

5.3. SSL ルートの作成

OpenShift Container Platform イメージの AMQ Broker のデプロイ後に、AMQ Broker トランスポートプロトコルポートの SSL Route を作成し、OpenShift 外部の AMQ Broker への接続を許可する必要があります。OpenShift ルーターではトラフィックを正しいサービスに送信するのに SNI を必要とするため、SSL ルートのみを公開することができます。

TLS Termination の **Passthrough** を選択すると、OpenShift ルーターが復号化して再送することなく AMQ Broker へのすべての通信がリレーされます。

注記

OpenShift ルーターは HTTP プロキシーである **HAProxy** を使用するため、通常の HTTP トラフィックには TLS パススルールートは必要ありません。

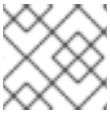
OpenShift Container Platform 上の AMQ Broker の外部クライアントは、SSL 接続のブローカー URL の設定時に OpenShift ルーターポート（デフォルトで443）を指定する必要があります。それ以外の場合は、AMQ Broker はデフォルトの SSL ポート(61617)の使用を試行します。

注記

デフォルトで、OpenShift ルーターは 443 ポートを使用します。ただし、ルーターは **ROUTER_SERVICE_HTTPS_PORT** 環境変数に指定された値に基づいて異なるポート番号を使用するように設定できます。詳細は、「OpenShift Container Platform 4.1 ルート」を参照してください。

また、URL にフェイルオーバープロトコルを含めると、Pod が再起動またはアップグレードされる場合や、ルーターで中断が発生した場合にクライアント接続が保持されます。これら両方の設定を以下に示します。

```
...
factory.setBrokerURL("failover://ssl://<route-to-broker-pod>:443");
...
```



注記

外部クライアントは HA をサポートしません。

各種の AMQ Broker トランスポートプロトコルのデフォルトポートが表に表示されます。

表5.1 AMQ Broker トランスポートプロトコルのデフォルトポート

AMQ Broker トランスポートプロトコル	デフォルトのポート
すべてのプロトコル（OpenWire、AMQP、STOMP、MQTT、および HornetQ）	61616

AMQ Broker トランスポートプロトコル	デフォルトのポート
すべてのプロトコル - SSL（OpenWire AMQP、STOMP、MQTT、および HornetQ）	61617
AMQP	5672
AMQP -SSL	5671
MQTT	1883
MQTT -SSL	8883
STOMP	61613
STOMP -SSL	61612

関連情報

- *クラスターネットワークの詳細は、「セキュリティー保護されたルート」を参照してください。*

第6章 デプロイメント用の AMQ BROKER 設定ファイルのカスタマイズ

代替リポジトリーからテンプレートを使用している場合は、**artemis-users.properties** などの AMQ Broker 設定ファイルを含めることができます。デプロイメント用にイメージがダウンロードされると、これらのファイルは< **amq-home>/conf/** から AMQ Broker の< **broker-instance-dir>/etc/** ディレクトリーにコピーされます。これはコンテナーにコミットされ、OpenShift レジストリーにプッシュされます。

注記

この方法を使用する場合には、設定ファイル（**AUTHENTICATION**など）のプレースホルダーが削除されていないことを確認します。プレースホルダーは、OpenShift Container Platform イメージで AMQ Broker をビルドするために必要です。

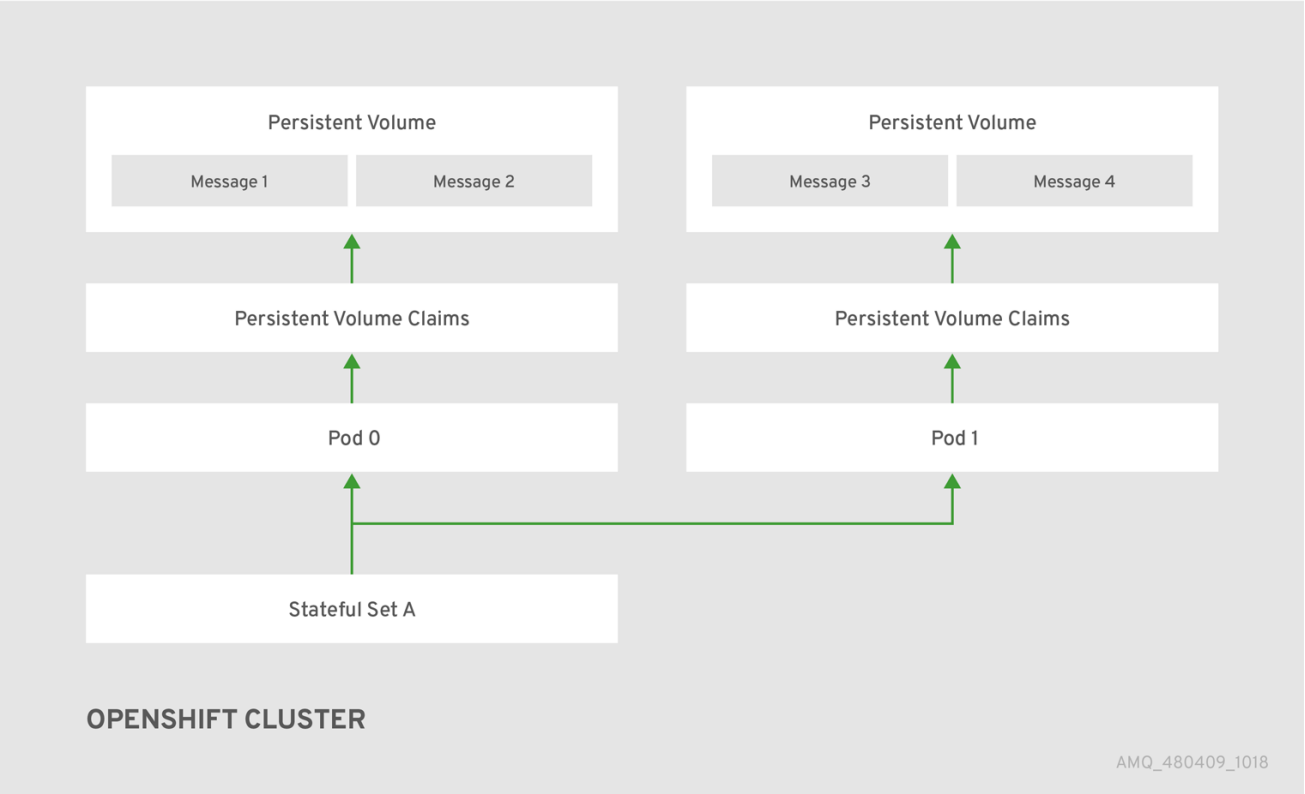
第7章 高可用性

7.1. 高可用性の概要

高可用性 という用語は、そのシステムの一部に障害が発生したりオフラインになった場合でも、残りの稼働状態を実現するシステムを指します。OCP のブローカーでは、HA はブローカーの可用性と、ブローカーに障害が発生した場合のメッセージングデータの整合性の両方を維持することを指します。

OpenShift Container Platform の AMQ Broker の HA 設定で、ブローカー Pod の複数のインスタンスを同時に実行します。各ブローカー Pod は、システムでストレージ**ボリュームを論理的に定義する永続ボリューム** (PV) にメッセージデータを書き込みます。ブローカー Pod が失敗するか、またはオフラインになると、その PV に保存されているメッセージデータは代替の利用可能なブローカーに再分散され、これを独自の PV に保存します。

図7.1 通常、StatefulSet の稼働



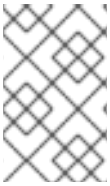
ブローカー Pod をオフラインにすると、StatefulSet がスケールダウンされ、接続されていない PV のメッセージデータに対して何が起こるかを管理する必要があります。now-offline Pod に関連付けられた PV に保持されるメッセージを移行するには、スケールダウンコントローラーを使用します。このようにメッセージデータを移行するプロセスは、**Pod のドレイン（解放）** と呼ばれる場合があります。

7.2. メッセージの移行

7.2.1. メッセージの移行の概要

デプロイメントの失敗や意図的なスケールダウンによってクラスターデプロイメントのブローカーがシャットダウンする場合に、メッセージの移行はメッセージングデータの整合性を確保することです。**Pod のドレイン（解放）** というメソッドを使用するメッセージの移行は、メッセージングデータを保存するためにブローカーによって使用される永続ボリュームから「孤立した」メッセージの削除および再分配を指します。メッセージ移行を有効にすると、AMQ Broker Operator の一部であるスケール

ダウンコントローラーは、デプロイメント内のブローカー Pod のシャットダウンを検出します。スケールダウンコントローラーは、シャットダウンされた各ブローカー Pod の専用のドレイナー Pod を開始し、メッセージ移行の準備を行います。それぞれのドレイナー Pod はクラスター内の残りのライブブローカー Pod のいずれかに接続し、メッセージをそのライブブローカー Pod に移行します。移行が完了すると、それぞれのドレイナー Pod がシャットダウンします。稼働中のブローカーで使用されていた永続ボリュームは、「Released」の状態に戻ります。



注記

AMQ Broker Operator 内のスケールダウンコントローラーは、単一の OpenShift プロジェクト内でのみ動作します。コントローラーは、別のプロジェクトのブローカー間でメッセージを移行できません。

注記

ブローカーデプロイメントを 0 (ゼロ) にスケールダウンする場合、メッセージングデータを移行できる稼働中のブローカー Pod がないため、メッセージ移行は行われません。ただし、デプロイメントをゼロブローカーにスケールダウンし、元のデプロイメントに含まれていた一部のブローカーのみに戻っても、シャットダウンされたブローカーのドレイン Pod が起動します。

7.2.1.1. メッセージの移行の仕組み

AMQ Broker Operator を使用して作成されたブローカーデプロイメントでメッセージ移行を有効にすると、スケールダウンコントローラーはブローカー Pod と同じプロジェクト namespace 内の Operator によって開始されます。

スケールダウンコントローラーは、それ自体を登録し、プロジェクト namespace の永続ボリューム要求(PVC)に関連する Kubernetes イベントをリッスンします。

スケールダウンコントローラーは、ボリューム要求の ordinal を確認して、孤立した PVC を確認します。ボリューム要求の ordinal は、プロジェクト namespace の StatefulSet の一部である既存のブローカー Pod の ordinal と比較されます。

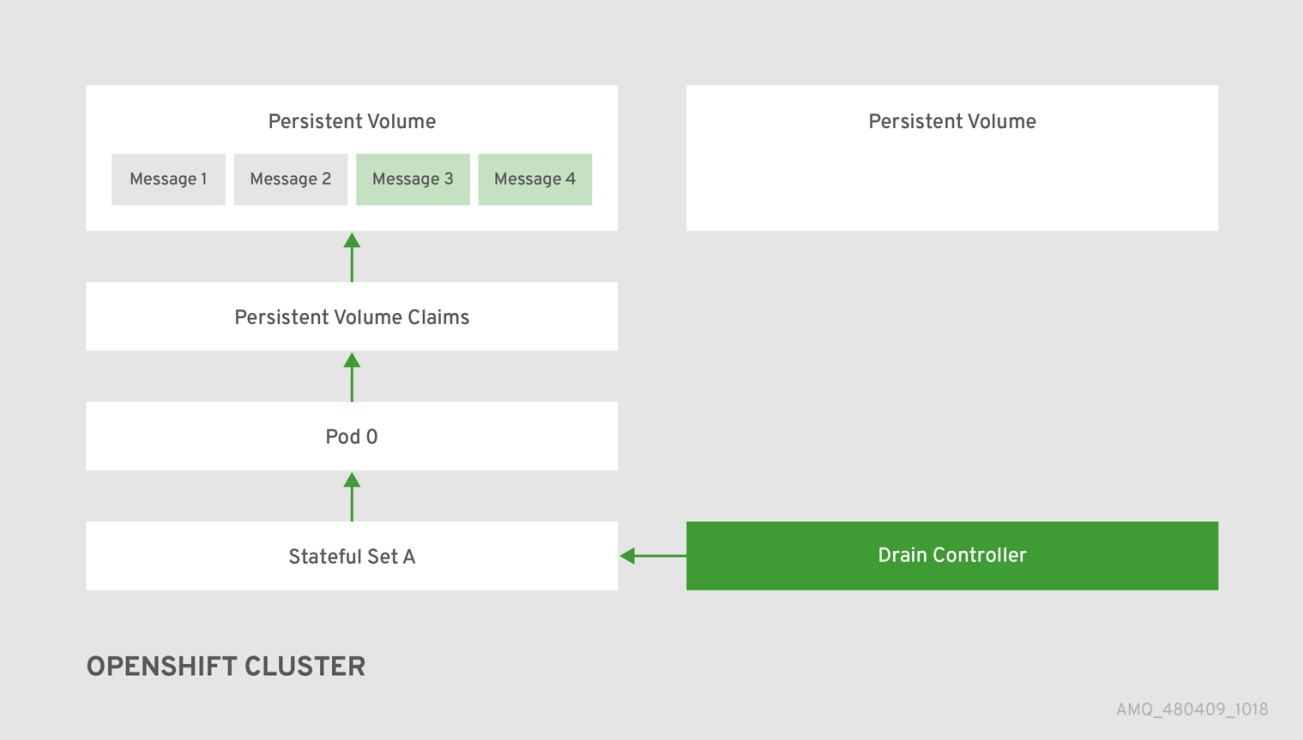
ボリューム要求の ordinal が既存のブローカー Pod の ordinal よりも大きい場合、そのordinal の Pod は終了され、データは別のブローカーに移行する必要があります。

これらの条件が満たされると、ドレイナー Pod が起動します。ドレイン Pod はブローカーを実行し、メッセージ移行を実行します。次に、ドレイナー Pod は、孤立した PVC メッセージの移行先となる代替ブローカー Pod を特定します。

注記

メッセージの移行を可能にするには、少なくとも 1 つ以上のブローカー Pod がデプロイメントで実行されている必要があります。

図7.2 スケールダウンコントローラーは、それ自体を登録し、PVC を削除して、永続ボリュームでメッセージを再配布します。



メッセージを運用ブローカー Pod に正常に移行した後、ドレイナー Pod はシャットダウンし、スケールダウンコントローラーは孤立した PVC を削除します。永続ボリュームが「Released」の状態に戻ります。

関連情報

- *ブローカーデプロイメントをスケールダウンする際のメッセージ移行の例については、「Migrating Messages Upon Scaledown」を参照してください。*

第8章 チュートリアル

前提条件

- これらの手順では、「OpenShift Container Platform 4.1 Getting Started」で作成したものと同様の OpenShift Container Platform 4.1 インスタンスを前提とし ています。

以下の手順では、アプリケーションテンプレートを使用してブローカーのさまざまなデプロイメントを作成する方法を説明します。

8.1. SSL を使用した基本的なブローカーのデプロイ

一時的で、かつ SSL をサポートする基本ブローカーをデプロイします。

8.1.1. イメージおよびテンプレートのデプロイ

前提条件

- このチュートリアルでは、ブローカー の準備 に基づいて実行され ます。
- 「基本的なブローカーのデプロイ」のチュートリアルを完了する ことをお勧めします。

手順

1. OpenShift Web コンソールに移動し、ログインします。
2. **amq-demo** プロジェクトスペースを選択します。
3. **Add to Project** をクリックした後に、**Browse Catalog** をクリックしてデフォルトのイメージストリームおよびテンプレートをすべて一覧表示します。
4. **Filter** 検索バーを使用して、一覧を **amq** に一致するものに限定します。**See all** をクリックして、必要なアプリケーションテンプレートを表示する必要がある場合があります。
5. **Red Hat AMQ Broker 7.4(Ephemeral, with SSL)** というラベルが付けられた **amq-broker-74-ssl** テンプレートを選択します。
6. 設定で以下の値を設定し、**Create** をクリックします。

表8.1 テンプレートの例

環境変数	表示名	値	説明
AMQ_PROTOCOL	AMQ プロトコル	openwire,amqp,stomp,mqtt,hornetq	ブローカーによって使用されるプロトコル
AMQ_QUEUES	キュー	demoQueue	demoQueue という名前のキャストキューを作成します。

環境変数	表示名	値	説明
AMQ_ADDRESSES	アドレス	demoTopic	demoTopic という名前のアドレス (またはトピック) を作成します。デフォルトでは、このアドレスには割り当てられたルーティングタイプが割り当てられていません。
AMQ_USER	AMQ ユーザー名	amq-demo-user	クライアントが使用するユーザー名
AMQ_PASSWORD	AMQ パスワード	パスワード	クライアントがユーザー名で使用するパスワード
AMQ_TRUSTSTORE	トラストストアファイル名	broker.ts	SSL トラストストアファイル名
AMQ_TRUSTSTORE_PASSWORD	トラストストアのパスワード	パスワード	トラストストアの作成時に使用されるパスワード
AMQ_KEYSTORE	AMQ キーストアファイル名	broker.ks	SSL キーストアのファイル名
AMQ_KEYSTORE_PASSWORD	AMQ キーストアのパスワード	パスワード	キーストアの作成時に使用されるパスワード

8.1.2. アプリケーションのデプロイ

アプリケーションの作成後に、これをデプロイして Pod を作成し、ブローカーを起動します。

手順

1. *OpenShift Container Platform Web コンソールで **Deployments** をクリックします。*
2. ***broker-amq** デプロイメントをクリックします。*
3. ***Deploy** をクリックしてアプリケーションをデプロイします。*
4. *ブローカーの Pod をクリックした後、**Logs** タブをクリックしてブローカーの状態を確認します。*
   *ブローカーログが読み込まれておらず、Pod のステータスは **ErrImagePull** または **ImagePullBackOff** を表示し、デプロイメント設定は Red Hat Container Registry から指定されたブローカーイメージを直接プルできませんでした。この場合、デプロイメント設定を編集して、正しいブローカーイメージ名と、Red Hat Container Registry の認証に使用するアカウントに関連付けられたイメージプルシークレット名を参照します。これで、ブローカーイメージをインポートし、ブローカーを起動できます。これを実行するには、ブローカーアプリケーションのデプロイおよび起動 にある手順と同様の手順を実行します。*

8.1.3. ルートの作成

OpenShift Container Platform の外部のクライアントが SSL を使用して接続できるようにブローカーのルートを作成します。デフォルトでは、すべてのブローカープロトコルは 61617/TCP ポートで利用できます。

注記

デプロイメントを、クラスター内にある複数のブローカーにスケールアップする場合、各ブローカーのサービスとルートを手動で作成してから、各サービスおよびルートの組み合わせを使用して特定のクライアントを特定のブローカー、またはブローカーリストにダイレクトする必要があります。クラスター化されたブローカーを AMQ Broker 管理コンソール独自のインスタンスに接続するためのサービスおよびルートを複数設定する例は、Creating Routes for the AMQ Broker management console を参照してください。

手順

1. **Services** → **broker-amq-tcp-ssl** をクリックします。
2. **Actions** → **Create route** をクリックします。
3. TLS パラメーターを表示するには、**Secure route** チェックボックスを選択します。
4. **TLS Termination** ドロップダウンメニューから、**Passthrough** を選択します。ここで選択する内容では、OpenShift ルーターが復号化および再送信せずに AMQ Broker との通信をすべてリレーします。
5. ルートを表示するには、**Routes** をクリックします。以下に例を示します。

```
https://broker-amq-tcp-amq-demo.router.default.svc.cluster.local
```

このホスト名は、SNI を使用する SSL でブローカーに接続するのに外部クライアントが使用します。

関連情報

- OpenShift Container Platform のルートの詳細は、ルートを参照してください。

8.2. 永続性および SSL を使用した基本的なブローカーのデプロイ

SSL をサポートする永続ブローカーをデプロイします。ブローカーが永続性を必要とする場合、ブローカーは StatefulSet としてデプロイされ、ジャーナルに使用するストレージデバイスが割り当てられます。ブローカー Pod が作成されると、Pod をシャットダウンする場合や、Pod が予期せずにシャットダウンする場合に残るストレージが割り当てられます。この設定は、標準のデプロイメントであるため、メッセージが失われないことを意味します。

前提条件

- このチュートリアルでは、ブローカーの準備 に基づいて構築 されます。
- 基本的なブローカーのデプロイ のチュートリアルを完了することを推奨します。

8.2.1. イメージおよびテンプレートのデプロイ

手順

1. OpenShift Web コンソールに移動し、ログインします。

2. *amq-demo プロジェクトスペースを選択します。*

3. *Add to Project → Browse catalog をクリックしてデフォルトのイメージストリームおよびテンプレートを一覧表示します。*

4. *Filter 検索バーを使用して、一覧を amq に一致するものに限定します。See all をクリックして、必要なアプリケーションテンプレートを表示する必要がある場合があります。*

5. *amq-broker-74-persistence-ssl テンプレートを選択します。これは、Red Hat AMQ Broker 7.4(Persistence)にラベルが付けられて います。*

6. *設定で以下の値を設定し、Create をクリックします。*

表8.2 テンプレートの例

環境変数	表示名	値	説明
AMQ_PROTOCOL	AMQ プロトコル	openwire,amqp,stomp,mqtt,hornetq	ブローカーによって使用されるプロトコル
AMQ_QUEUES	キュー	demoQueue	demoQueue という名前のキャストキューを作成します。
AMQ_ADDRESSES	アドレス	demoTopic	demoTopic という名前のアドレス (またはトピック) を作成します。デフォルトでは、このアドレスには割り当てられたルーティングタイプが割り当てられていません。
VOLUME_CAPACITY	AMQ ボリュームのサイズ	1Gi	ジャーナル用に作成される永続ボリュームサイズ
AMQ_USER	AMQ ユーザー名	amq-demo-user	クライアントが使用するユーザー名
AMQ_PASSWORD	AMQ パスワード	パスワード	クライアントがユーザー名で使用するパスワード
AMQ_TRUSTSTORE	トラストストアファイル名	broker.ts	SSL トラストストアファイル名
AMQ_TRUSTSTORE_PASSWORD	トラストストアのパスワード	パスワード	トラストストアの作成時に使用されるパスワード
AMQ_KEYSTORE	AMQ キーストアファイル名	broker.ks	SSL キーストアのファイル名
AMQ_KEYSTORE_PASSWORD	AMQ キーストアのパスワード	パスワード	キーストアの作成時に使用されるパスワード

8.2.2. アプリケーションのデプロイ

アプリケーションを作成したら、デプロイする必要があります。アプリケーションのデプロイにより Pod が作成され、ブローカーが起動します。

手順

1. OpenShift Container Platform Web コンソールで **Stateful Sets** をクリックします。
2. **broker-amq** デプロイメントをクリックします。
3. **Deploy** をクリックしてアプリケーションをデプロイします。
4. ブローカーの Pod をクリックした後、**Logs** タブをクリックしてブローカーの状態を確認します。テンプレートから作成されたキューが表示されます。
   ブローカーログが読み込まれておらず、Pod のステータスは **ErrImagePull** または **ImagePullBackOff** を表示し、設定は Red Hat Container Registry から指定されたブローカーイメージを直接プルできませんでした。この場合、デプロイメント設定を編集して、正しいブローカーイメージ名と、Red Hat Container Registry の認証に使用するアカウントに関連付けられたイメージプルシークレット名を参照します。これで、ブローカーイメージをインポートし、ブローカーを起動できます。これを実行するには、ブローカーアプリケーションのデプロイおよび起動 にある手順と同様の手順を実行します。
5. **Terminal** タブをクリックして、CLI で一部のメッセージを送信できるシェルにアクセスします。

```
sh-4.2$ ./broker/bin/artemis producer --destination queue://demoQueue
Producer ActiveMQQueue[demoQueue], thread=0 Started to calculate elapsed time ...

Producer ActiveMQQueue[demoQueue], thread=0 Produced: 1000 messages
Producer ActiveMQQueue[demoQueue], thread=0 Elapsed time in second : 4 s
Producer ActiveMQQueue[demoQueue], thread=0 Elapsed time in milli second : 4584 milli seconds

sh-4.2$ ./broker/bin/artemis consumer  --destination queue://demoQueue
Consumer:: filter = null
Consumer ActiveMQQueue[demoQueue], thread=0 wait until 1000 messages are consumed
Received 1000
Consumer ActiveMQQueue[demoQueue], thread=0 Consumed: 1000 messages
Consumer ActiveMQQueue[demoQueue], thread=0 Consumer thread finished
```

または、以下の例のように、OpenShift クライアントを使用して Pod 名を使用してシェルにアクセスします。

```
// Get the Pod names and internal IP Addresses
oc get pods -o wide

// Access a broker Pod by name
oc rsh <broker-pod-name>
```

6. oc コマンドを使用してブローカーをスケールダウンします。

```
$ oc scale statefulset broker-amq --replicas=0
statefulset "broker-amq" scaled
```

コンソールを使用して、Pod 数が 0 であることを確認します。

7. 次に、ブローカーを **1** に再びスケールアップします。

```
$ oc scale statefulset broker-amq --replicas=1
statefulset "broker-amq" scaled
```

8. ターミナルを使用してメッセージを再度消費します。以下に例を示します。

```
sh-4.2$ broker/bin/artemis consumer --destination queue://demoQueue
Consumer:: filter = null
Consumer ActiveMQQueue[demoQueue], thread=0 wait until 1000 messages are consumed
Received 1000
Consumer ActiveMQQueue[demoQueue], thread=0 Consumed: 1000 messages
Consumer ActiveMQQueue[demoQueue], thread=0 Consumer thread finished
```

関連情報

- ステートフルなアプリケーションの管理の詳細は、StatefulSets (external) を参照してください。

8.2.3. ルートの作成

OpenShift Container Platform の外部のクライアントが SSL を使用して接続できるようにブローカーのルートを作成します。デフォルトでは、ブローカープロトコルは 61617/TCP ポートから入手できます。

注記

デプロイメントを、クラスター内にある複数のブローカーにスケールアップする場合、各ブローカーのサービスとルートを手動で作成してから、各サービスおよびルートの組み合わせを使用して特定のクライアントを特定のブローカー、またはブローカーリストにダイレクトする必要があります。複数のサービスとルートを、クラスターブローカーを AMQ Broker 管理コンソールの独自のインスタンスに接続するように設定する例については、「AMQ Broker 管理コンソールのルートの作成」を参照してください。

手順

1. **Services → broker-amq-tcp-ssl** をクリックします。
2. **Actions → Create a route** をクリックします。
3. TLS パラメーターを表示するには、**Secure route** チェックボックスを選択します。
4. **TLS Termination** ドロップダウンメニューから、**Passthrough** を選択します。ここで選択する内容では、OpenShift ルーターが復号化および再送信せずに AMQ Broker との通信をすべてリレーします。
5. ルートを表示するには、**Routes** をクリックします。以下に例を示します。

```
https://broker-amq-tcp-amq-demo.router.default.svc.cluster.local
```

このホスト名は、SNI を使用する SSL でブローカーに接続するのに外部クライアントが使用します。

関連情報

- OpenShift Container Platform のルートについての詳細は、「ルート」を参照して ください。

8.3. クラスター化されたブローカーのセットのデプロイ

ブローカーのクラスターセットをデプロイし、ここではブローカーごとに独自の Pod で実行されます。

8.3.1. メッセージの分散

メッセージ分散は ON_DEMAND を使用するように設定されます。つまり、メッセージがクラスター化されたブローカーに到達すると、コンシューマーがあるブローカーにラウンドロビン方式で配布されます。

このメッセージ分散ポリシーは、直接または Open Shift ルーターを介して接続されているコンシューマーが別のブローカーに接続されているときに、メッセージが特定のブローカーでスタックするのを防ぎます。

再分散の遅延は、デフォルトではゼロです。メッセージがコンシューマーのないキューにある場合は、別のブローカーに再分散されます。

注記

再分散を有効にすると、メッセージが順番に関係なく分散される可能性があります。

8.3.2. イメージおよびテンプレートのデプロイ

前提条件

- この手順では、ブローカーの準備 に基づいて構築 されます。
- 基本的なブローカーのデプロイ のチュートリアルを完了することを推奨します。

手順

1. OpenShift Web コンソールに移動し、ログインします。
2. **amq-demo** プロジェクトスペースを選択します。
3. **Add to Project** > **Browse catalog** の順にクリックしてデフォルトのイメージストリームおよびテンプレートを一覧表示します。
4. Filter 検索バーを使用して、一覧を **amq** に一致するものに限定します。**See all** をクリックして、必要なアプリケーションテンプレートを表示します。
5. **Red Hat AMQ Broker 7.4（SSL、clustered）というラベルが付けられた amq-broker-74-persistence- clustered** テンプレートを選択します。
6. 設定で以下の値を設定し、**Create** をクリックします。

   表8.3 テンプレートの例

環境変数	表示名	値	説明
AMQ_PROTOCOL	AMQ プロトコル	openwire,amqp,stomp,mqtt,hornetq	ブローカーによって使用されるプロトコル
AMQ_QUEUES	キュー	demoQueue	demoQueue という名前のキャストキューを作成します。
AMQ_ADDRESSES	アドレス	demoTopic	demoTopic という名前のアドレス (またはトピック) を作成します。デフォルトでは、このアドレスには割り当てられたルーティングタイプが割り当てられていません。
VOLUME_CAPACITY	AMQ ボリュームのサイズ	1Gi	ジャーナル用に作成される永続ボリュームサイズ
AMQ_CLUSTERED	クラスター化	true	ブローカーがクラスター化されるようにするには、true に指定する必要があります。
AMQ_CLUSTER_USER	クラスターユーザー	generated	ブローカーが相互の接続に使用するユーザー名
AMQ_CLUSTER_PASSWORD	クラスターパスワード	generated	ブローカーが相互接続に使用するパスワード
AMQ_USER	AMQ ユーザー名	amq-demo-user	クライアントが使用するユーザー名
AMQ_PASSWORD	AMQ パスワード	パスワード	クライアントがユーザー名で使用するパスワード

8.3.3. アプリケーションのデプロイ

アプリケーションを作成したら、デプロイする必要があります。アプリケーションのデプロイにより Pod が作成され、ブローカーが起動します。

手順

1. *OpenShift Container Platform Web コンソールで **Stateful Sets** をクリックします。*
2. ***broker-amq** デプロイメントをクリックします。*
3. ***Deploy** をクリックしてアプリケーションをデプロイします。*

注記

クラスター化されたテンプレートのデフォルトのレプリカ数は 0 です。Pod は表示されないはずです。

4. *Pod を 3 つにスケールアップし、ブローカーのクラスターを作成します。*

```
$ oc scale statefulset broker-amq --replicas=3
statefulset "broker-amq" scaled
```

5. *3 つの Pod が実行されていることを確認します。*

```
$ oc get pods
NAME          READY     STATUS    RESTARTS   AGE
broker-amq-0  1/1       Running   0          33m
broker-amq-1  1/1       Running   0          33m
broker-amq-2  1/1       Running   0          29m
```

6. *表示される Pod のステータスが **ErrImagePull** または **ImagePullBackOff** の場合には、デプロイメントは Red Hat Container Registry から指定されたブローカーイメージを直接プルできていません。この場合、Stateful Set を編集して、正しいブローカーイメージ名と、Red Hat コンテナーレジストリーの認証に使用されるアカウントに関連付けられたイメージプルシークレット名を参照します。次に、ブローカーイメージをインポートし、ブローカーを起動できます。これを実行するには、ブローカーアプリケーションのデプロイおよび起動 にある手順と同様の手順を実行します。*

7. *ログをチェックして、ブローカーが新しい Pod でクラスター化されていることを確認します。*

```
$ oc logs broker-amq-2
```

これにより、新しいブローカーのログと、ブローカー間で作成されるクラスター化されたブリッジのエントリーが表示されます。

```
2018-08-29 07:43:55,779 INFO  [org.apache.activemq.artemis.core.server] AMQ221027:
Bridge ClusterConnectionBridge@1b0e9e9d [name=$.artemis.internal.sf.my-
cluster.4333c830-ab5f-11e8-afb8-0a580a82006e,
queue=QueueImpl[name=$.artemis.internal.sf.my-cluster.4333c830-ab5f-11e8-afb8-
0a580a82006e, postOffice=PostOfficeImpl
[server=ActiveMQServerImpl::serverUUID=9cedb69d-ab5e-11e8-87a4-0a580a82006c],
temp=false]@5e0c0398 targetConnector=ServerLocatorImpl (identity=(Cluster-connection-
bridge::ClusterConnectionBridge@1b0e9e9d [name=$.artemis.internal.sf.my-
cluster.4333c830-ab5f-11e8-afb8-0a580a82006e,
queue=QueueImpl[name=$.artemis.internal.sf.my-cluster.4333c830-ab5f-11e8-afb8-
0a580a82006e, postOffice=PostOfficeImpl
[server=ActiveMQServerImpl::serverUUID=9cedb69d-ab5e-11e8-87a4-0a580a82006c],
temp=false]@5e0c0398 targetConnector=ServerLocatorImpl [initialConnectors=
[TransportConfiguration(name=artemis, factory=org-apache-activemq-artemis-core-remoting-
impl-netty-NettyConnectorFactory) ?port=61616&host=10-130-0-110],
discoveryGroupConfiguration=null]]::ClusterConnectionImpl@806813022[nodeUUID=9cedb69d
-ab5e-11e8-87a4-0a580a82006c, connector=TransportConfiguration(name=artemis,
factory=org-apache-activemq-artemis-core-remoting-impl-netty-NettyConnectorFactory) ?
port=61616&host=10-130-0-108, address=,
server=ActiveMQServerImpl::serverUUID=9cedb69d-ab5e-11e8-87a4-0a580a82006c]))
[initialConnectors=[TransportConfiguration(name=artemis, factory=org-apache-activemq-
artemis-core-remoting-impl-netty-NettyConnectorFactory) ?port=61616&host=10-130-0-110],
discoveryGroupConfiguration=null]] is connected
```

8.3.4. AMQ Broker 管理コンソールのルートの作成

クラスタリングテンプレートはデフォルトでコンソールを公開しません。これは、OpenShift プロキシーがクラスター内の各ブローカーに対して負荷分散を行うため、どのブローカーコンソールを接続するかを制御することができないためです。

以下の手順では、AMQ Broker 管理コンソールのルートを作成し、クラスターのブローカーに接続する方法を説明します。

注記

クラスターがSSL を使用する場合は、各ブローカーにサービスとルートを手動で作成し、それぞれのService-and-Route の組み合わせを使用して指定のクライアントを特定のブローカーまたはブローカーリストに転送する必要があります。詳細は、「AMQ Broker 管理コンソールのルートの作成」を参照してください。

手順

1. **Add to Project** ドロップダウンから **import YAML/JSON** を選択します。
2. 以下を入力し、create をクリックします。

```
apiVersion: v1
kind: Route
metadata:
  labels:
    app: broker-amq
    application: broker-amq
  name: console-jolokia
spec:
  port:
    targetPort: console-jolokia
  to:
    kind: Service
    name: broker-amq-headless
    weight: 100
  wildcardPolicy: Subdomain
  host: star.broker-amq-headless.amq-demo.svc
```

注記

host: star.broker-amq-headless.amq-demo.svc 設定は、ブローカーの各 Pod に使用されるホスト名です。スターは Pod 名に置き換えられたため、Pod 名が **broker-amq-0** の場合、ホスト名は broker **-amq-0.broker-amq-headless.amq-demo.svc になります。**

3. /etc/hosts ファイルにエントリーを追加して、ルート名を OpenShift クラスターの IP アドレスにマッピングします。たとえば、3 つの Pod がある場合は、以下のようにエントリーを追加します。

```
10.0.0.1 broker-amq-0.broker-amq-headless.amq-demo.svc
10.0.0.1 broker-amq-1.broker-amq-headless.amq-demo.svc
10.0.0.1 broker-amq-2.broker-amq-headless.amq-demo.svc
```

4. ブラウザーで、**http://broker-amq-0.broker-amq-headless.amq-demo.svc** アドレスを使用してコンソールに移動します。

関連情報

- ブローカーのクラスタリングの詳細は、「メッセージ再分配の有効化」を参照してください。

8.4. クラスター化された SSL ブローカーセットのデプロイ

ブローカーのクラスター化されたセットをデプロイします。各ブローカーは独自の Pod で実行され、ブローカーは SSL を使用した接続を受け入れるように設定されています。

8.4.1. メッセージの分散

メッセージ分散は ON_DEMAND を使用するように設定されます。つまり、メッセージがクラスター化されたブローカーに到達すると、コンシューマーがあるブローカーにラウンドロビン方式で配布されます。

このメッセージ分散ポリシーは、直接または Open Shift ルーターを介して接続されているコンシューマーが別のブローカーに接続されているときに、メッセージが特定のブローカーでスタックするのを防ぎます。

再分散の遅延は、デフォルトではゼロではありません。メッセージがコンシューマーのないキューにある場合は、別のブローカーに再分散されます。

注記

再分散を有効にすると、メッセージが順番に関係なく分散される可能性があります。

8.4.2. イメージおよびテンプレートのデプロイ

前提条件

- この手順では、ブローカーの準備 に基づいて構築 されます。
- 基本的なブローカーのデプロイ の例を完了しておくことをお勧めします。

手順

1. OpenShift Web コンソールに移動し、ログインします。
2. **amq-demo** プロジェクトスペースを選択します。
3. **Add to Project** > **Browse catalog** の順にクリックしてデフォルトのイメージストリームおよびテンプレートを一覧表示します。
4. **Filter** 検索バーを使用して、一覧を **amq** に一致するものに限定します。**See all** をクリックして、必要なアプリケーションテンプレートを表示します。
5. **Red Hat AMQ Broker 7.4（SSL、clustered）というラベルが付けられた amq-broker-74-persistence-clustered -ssl** テンプレートを選択します。
6. 設定で以下の値を設定し、**Create** をクリックします。

   表8.4 テンプレートの例

環境変数	表示名	値	説明
AMQ_PROTOCOL	AMQ プロトコル	openwire,amqp,stomp,mqtt,hornetq	ブローカーによって使用されるプロトコル
AMQ_QUEUES	キュー	demoQueue	demoQueue という名前のキャストキューを作成します。
AMQ_ADDRESSES	アドレス	demoTopic	demoTopic という名前のアドレス (またはトピック) を作成します。デフォルトでは、このアドレスには割り当てられたルーティングタイプが割り当てられていません。
VOLUME_CAPACITY	AMQ ボリュームのサイズ	1Gi	ジャーナル用に作成される永続ボリュームサイズ
AMQ_CLUSTERED	クラスター化	true	ブローカーがクラスター化されるようにするには、true に指定する必要があります。
AMQ_CLUSTER_USER	クラスターユーザー	generated	ブローカーが相互の接続に使用するユーザー名
AMQ_CLUSTER_PASSWORD	クラスターパスワード	generated	ブローカーが相互接続に使用するパスワード
AMQ_USER	AMQ ユーザー名	amq-demo-user	クライアントが使用するユーザー名
AMQ_PASSWORD	AMQ パスワード	パスワード	クライアントがユーザー名で使用するパスワード
AMQ_TRUSTSTORE	トラストストアファイル名	broker.ts	SSL トラストストアファイル名
AMQ_TRUSTSTORE_PASSWORD	トラストストアのパスワード	パスワード	トラストストアの作成時に使用されるパスワード
AMQ_KEYSTORE	AMQ キーストアファイル名	broker.ks	SSL キーストアのファイル名
AMQ_KEYSTORE_PASSWORD	AMQ キーストアのパスワード	パスワード	キーストアの作成時に使用されるパスワード

8.4.3. アプリケーションのデプロイ

アプリケーションの作成後にデプロイします。アプリケーションのデプロイにより Pod が作成され、ブローカーが起動します。

手順

1. OpenShift Container Platform Web コンソールで **Stateful Sets** をクリックします。
2. **broker-amq** デプロイメントをクリックします。
3. **Deploy** をクリックしてアプリケーションをデプロイします。

注記

クラスター化されたテンプレートのデフォルトのレプリカ数は **0** であるため、Pod は表示されません。

4. Pod を 3 つにスケールアップし、ブローカーのクラスターを作成します。

```
$ oc scale statefulset broker-amq --replicas=3
statefulset "broker-amq" scaled
```

5. 3 つの Pod が実行されていることを確認します。

```
$ oc get pods
NAME          READY     STATUS    RESTARTS   AGE
broker-amq-0  1/1       Running   0          33m
broker-amq-1  1/1       Running   0          33m
broker-amq-2  1/1       Running   0          29m
```

6. 表示される Pod のステータスが **ErrImagePull** または **ImagePullBackOff** の場合には、デプロイメントは Red Hat Container Registry から指定されたブローカーイメージを直接プルできていません。この場合、Stateful Set を編集して、正しいブローカーイメージ名と、Red Hat コンテナーレジストリーの認証に使用されるアカウントに関連付けられたイメージプルシークレット名を参照します。次に、ブローカーイメージをインポートし、ブローカーを起動できます。これを実行するには、ブローカーアプリケーションのデプロイおよび起動 にある手順と同様の手順を実行します。

7. ログをチェックして、ブローカーが新しい Pod でクラスター化されたことを確認します。

```
$ oc logs broker-amq-2
```

これにより、以下のように、新しいブローカーのログと、ブローカー間で作成されたクラスター化されたブリッジのエントリーがすべて表示されます。

```
2018-08-29 07:43:55,779 INFO  [org.apache.activemq.artemis.core.server] AMQ221027:
Bridge ClusterConnectionBridge@1b0e9e9d [name=$.artemis.internal.sf.my-
cluster.4333c830-ab5f-11e8-afb8-0a580a82006e,
queue=QueueImpl[name=$.artemis.internal.sf.my-cluster.4333c830-ab5f-11e8-afb8-
0a580a82006e, postOffice=PostOfficeImpl
[server=ActiveMQServerImpl::serverUUID=9cedb69d-ab5e-11e8-87a4-0a580a82006c],
temp=false]@5e0c0398 targetConnector=ServerLocatorImpl (identity=(Cluster-connection-
bridge::ClusterConnectionBridge@1b0e9e9d [name=$.artemis.internal.sf.my-
cluster.4333c830-ab5f-11e8-afb8-0a580a82006e,
queue=QueueImpl[name=$.artemis.internal.sf.my-cluster.4333c830-ab5f-11e8-afb8-
```

```
0a580a82006e, postOffice=PostOfficeImpl
[server=ActiveMQServerImpl::serverUUID=9cedb69d-ab5e-11e8-87a4-0a580a82006c],
temp=false]@5e0c0398 targetConnector=ServerLocatorImpl [initialConnectors=
[TransportConfiguration(name=artemis, factory=org-apache-activemq-artemis-core-remoting-
impl-netty-NettyConnectorFactory) ?port=61616&host=10-130-0-110],
discoveryGroupConfiguration=null]]::ClusterConnectionImpl@806813022[nodeUUID=9cedb69d
-ab5e-11e8-87a4-0a580a82006c, connector=TransportConfiguration(name=artemis,
factory=org-apache-activemq-artemis-core-remoting-impl-netty-NettyConnectorFactory) ?
port=61616&host=10-130-0-108, address=,
server=ActiveMQServerImpl::serverUUID=9cedb69d-ab5e-11e8-87a4-0a580a82006c]))
[initialConnectors=[TransportConfiguration(name=artemis, factory=org-apache-activemq-
artemis-core-remoting-impl-netty-NettyConnectorFactory) ?port=61616&host=10-130-0-110],
discoveryGroupConfiguration=null]] is connected
```

8.4.4. AMQ Broker 管理コンソールのルートの作成

クラスタリングテンプレートでは、デフォルトで AMQ Broker 管理コンソールは公開されません。これは、OpenShift プロキシーがクラスター内の各ブローカー間で負荷分散を実行し、特定のタイミングで接続されるブローカーコンソールを制御できないためです。

以下の手順は、独自の管理コンソールインスタンスに接続するようにクラスター内の各ブローカーを設定する方法を示しています。これを行うには、クラスター内の各ブローカー Pod に専用のサービスとルートの組み合わせを作成します。

前提条件

- クラスター化されたブローカーのセットをすでにデプロイし、各ブローカーは独自の Pod で実行され、ブローカーは SSL を使用して接続を受け入れるように設定されます。「クラスター化された SSL ブローカーのセットのデプロイ」を参照してください。

手順

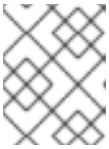
1. StatefulSet セレクターを使用して Pod 間を選択し、クラスターの各 Pod に通常のサービスを作成します。これを実行するには、以下のようなサービステンプレートを **.yaml** 形式でデプロイします。

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    description: 'Service for the management console of broker pod XXXX'
  labels:
    app: application2
    application: application2
    template: amq-broker-74-persistence-clustered-ssl
  name: amq2-amq-console-XXXX
  namespace: amq74-p-c-ssl-2
spec:
  ports:
    - name: console-jolokia
      port: 8161
      protocol: TCP
      targetPort: 8161
  selector:
```

```
    deploymentConfig: application2-amq
    statefulset.kubernetes.io/pod-name: application2-amq-XXXX
  type: ClusterIP
```

上記のテンプレートの **XXXX** は、サービスに関連付けるブローカー Pod の ordinal 値に置き換えます。たとえば、サービス をクラスターの最初の Pod に関連付けるには、**XXXX** を **0** に設定します。サービスと 2 つ目の Pod を関連付けるには、**XXXX** を **1** に設定するなどです。

クラスターにある各ブローカー Pod のテンプレートのインスタンスを保存してデプロイします。



注記

上記のテンプレートのサンプルでは、セレクターは Kubernetes で定義される Pod 名を使用します。

2. 各ブローカー Pod にルートを作成し、AMQ Broker 管理コンソールが SSL を使用して Pod に接続できるようにします。
Routes → **Create Route** の順にクリックします。

   Edit Route ページが表示されます。

   a. **Services** ドロップダウンメニューで、以前に作成したブローカーサービスで、ルートの関連付け先を選択します（例： **amq2-amq-console-0** ）。

   b. **Target Port** を **8161** に設定し、AMQ Broker 管理コンソールへのアクセスを有効にします。

   c. TLS パラメーターを表示するには、**Secure route** チェックボックスを選択します。

      i. **TLS Termination** ドロップダウンメニューから、**Passthrough** を選択します。
      ここで選択する内容では、OpenShift ルーターが復号化および再送信せずに AMQ Broker との通信をすべてリレーします。

   d. **Create** をクリックします。
   ブローカー Pod の 1 つに関連付けられたルートを作成する場合には、生成される **.yaml** ファイルには以下のような行が含まれます。

```
spec:
  host: amq2-amq-console-0-amq74-p-c-ssl-2.apps-ocp311.example.com
  port:
    targetPort: console-jolokia
  tls:
    termination: passthrough
  to:
    kind: Service
    name: amq2-amq-console-0
    weight: 100
  wildcardPolicy: None
```

3. 特定のブローカーインスタンスの管理コンソールにアクセスするには、上記の **ホスト** URL を Web ブラウザーにコピーします。

関連情報

- ブローカークラスターのメッセージングの詳細は、メッセージ再分配の有効化 を参照してください。

8.5. カスタム設定を使用したブローカーのデプロイ

カスタム設定でブローカーをデプロイします。テンプレートを使用すると機能を取得できますが、必要に応じてブローカー設定をカスタマイズできます。

前提条件

- このチュートリアルでは、ブローカーの準備 に基づいて構築 されます。
- 基本的なブローカーのデプロイ のチュートリアルを完了することを推奨します。

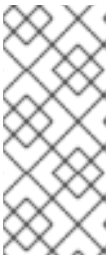
8.5.1. イメージおよびテンプレートのデプロイ

手順

1. OpenShift Web コンソールに移動し、ログインします。
2. **amq-demo** プロジェクトスペースを選択します。
3. **Add to Project** > **Browse catalog** の順にクリックしてデフォルトのイメージストリームおよびテンプレートを一覧表示します。
4. **Filter** 検索バーを使用して、**amq** に一致するものだけに結果を限定します。**See all** をクリックして、必要なアプリケーションテンプレートを表示します。
5. **Red Hat AMQ Broker 7.4(Ephemeral, no SSL)** というラベルが付けられた **amq-broker-74-custom** テンプレートを選択します。
6. 設定で **broker.xml** を、使用するカスタム設定に更新します。**Create** をクリックします。

注記

テキストエディターを使用してブローカーの XML 設定を作成します。次に、変更詳細を **broker.xml** フィールドに切り取って貼り付けます。



注記

OpenShift Container Platform では、このプラットフォームにデプロイされる多くのアプリケーションに共通するので、**ConfigMap** オブジェクトを使用して **broker.xml** フィールドに指定するカスタム設定は保存されません。代わりに、ブローカーコンテナーの起動時に設定をスタンドアロンファイルに転送する前に、OpenShift は指定された設定を環境変数に一時的に保存します。

8.5.2. アプリケーションのデプロイ

アプリケーションを作成したら、デプロイする必要があります。アプリケーションのデプロイにより Pod が作成され、ブローカーが起動します。

手順

1. OpenShift Container Platform Web コンソールで **Deployments** をクリックします。

2. **broker-amq** デプロイメントをクリックします。
3. **Deploy** をクリックしてアプリケーションをデプロイします。

8.6. 基本的な SSL クライアントの例

Qpid JMS クライアントを使用して、SSL を使用するように設定されたブローカーからメッセージを送受信するクライアントを実装します。

前提条件

- このチュートリアルでは、ブローカー の準備 に基づいて実行され ます。
- SSL チュートリアルを使用した基本的なブローカーのデプロイ の完了が推奨されます。
- AMQ JMS の例

8.6.1. クライアントの設定

SSL ブローカーに接続するために更新可能なサンプルクライアントを作成します。以下の手順では、AMQ JMS の例 を基にしています。

手順

1. /etc/hosts ファイルにエントリーを追加して、ルート名を OpenShift クラスターの IP アドレスにマッピングします。

```
10.0.0.1 broker-amq-tcp-amq-demo.router.default.svc.cluster.local
```

2. 以下のように、jndi.properties 設定ファイルは、前のステップで作成したルート、トラストストア、およびキーストアを使用するように更新します。

```
connectionfactory.myFactoryLookup = amqps://broker-amq-tcp-amq-
demo.router.default.svc.cluster.local:8443?transport.keyStoreLocation=<keystore-
path>client.ks&transport.keyStorePassword=password&transport.trustStoreLocation=
<truststore-
path>/client.ts&transport.trustStorePassword=password&transport.verifyHost=false
```

3. jndi.properties 設定ファイルは、先に作成したキューを使用するように更新します。

```
queue.myDestinationLookup = demoQueue
```

4. 送信側のクライアントを実行してテキストメッセージを送信します。
5. 受信側クライアントを実行して、テキストメッセージを受信します。以下が表示されるはずです。

```
Received message: Message Text!
```

8.7. サブドメインを使用した外部クライアントの例

*ノードポートを介してクラスター化されたブローカーセットを公開し、コア JMS クライアントを使用して接続します。これにより、クライアントは **amq-broker-74-persistence-clustered-ssl** テンプレートを使用して設定されるブローカーセットに接続できます。*

8.7.1. ブローカーの公開

ブローカーのクラスターが外部で利用可能になり、OpenShift ルーターをバイパスして直接接続できるようにブローカーを設定します。これは、各 Pod を独自のホスト名で公開するルートを作成して実行されます。

前提条件

- *クラスター化されたブローカーのセットのデプロイ*

手順

1. ***Add to Project** ドロップダウンから **import YAML/JSON** を選択します。*
2. *以下を入力し、Create をクリックします。*

```
apiVersion: v1
kind: Route
metadata:
  labels:
    app: broker-amq
    application: broker-amq
  name: tcp-ssl
spec:
  port:
    targetPort: ow-multi-ssl
  tls:
    termination: passthrough
  to:
    kind: Service
    name: broker-amq-headless
    weight: 100
  wildcardPolicy: Subdomain
  host: star.broker-ssl-amq-headless.amq-demo.svc
```

注記

*ここで重要な設定は、**Subdomain** のワイルドカードポリシーです。これにより、各ブローカーは独自のホスト名からアクセスできます。*

8.7.2. クライアントの接続

SSL ブローカーに接続するために更新可能なサンプルクライアントを作成します。この手順は、AMQ JMS の例 を基にしています。

手順

1. */etc/hosts ファイルにエントリーを追加して、ルート名をブローカーの実際の IP アドレスにマッピングします。*

```
10.0.0.1 broker-amq-0.broker-ssl-amq-headless.amq-demo.svc broker-amq-1.broker-ssl-amq-headless.amq-demo.svc broker-amq-2.broker-ssl-amq-headless.amq-demo.svc
```

2. 以下のように、jndi.properties 設定ファイルを、前のステップで作成したルート、トラストストア、およびキーストアを使用するように更新します。

```
connectionfactory.myFactoryLookup = amqps://broker-amq-0.broker-ssl-amq-headless.amq-demo.svc:443?transport.keyStoreLocation=/home/ataylor/projects/jboss-amq-7-broker-openshift-image/client.ks&transport.keyStorePassword=password&transport.trustStoreLocation=/home/ataylor/projects/jboss-amq-7-broker-openshift-image/client.ts&transport.trustStorePassword=password&transport.verifyHost=false
```

3. jndi.properties 設定ファイルは、先に作成したキューを使用するように更新します。

```
queue.myDestinationLookup = demoQueue
```

4. 送信側のクライアントコードを実行してテキストメッセージを送信します。

5. 受信側クライアントコードを実行して、テキストメッセージを受信します。以下が表示されるはずです。

```
Received message: Message Text!
```

関連情報

- AMQ JMS クライアントの使用の詳細は、AMQ JMS Examples を参照してください。

8.8. ポートバインディングを使用した外部クライアントの例

NodePort を介してブローカーのクラスターセットを公開し、コア JMS クライアントを使用して接続します。これにより、SNI または SSL をサポートしないクライアントが有効になります。これは、**amq-broker-74-persistence-clustered** テンプレートを使用して設定されるクラスターで使用されます。

8.8.1. ブローカーの公開

ブローカーのクラスターが外部で利用可能になり、OpenShift ルーターをバイパスして直接接続できるようにブローカーを設定します。これは、NodePort を使用してクラスター周辺の負荷分散を行うサービスを作成して行われます。

前提条件

- クラスター化されたブローカーのセットのデプロイ

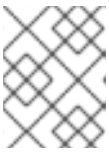
手順

1. **Add to Project** ドロップダウンから **import YAML/JSON** を選択します。

2. 以下を入力し、Create をクリックします。

```
apiVersion: v1
kind: Service
metadata:
```

```
  annotations:
    description: The broker's OpenWire port.
    service.alpha.openshift.io/dependencies: >-
      [{"name": "broker-amq-amqp", "kind": "Service"},{"name":
       "broker-amq-mqtt", "kind": "Service"},{"name": "broker-amq-stomp", "kind":
       "Service"}]
  creationTimestamp: '2018-08-29T14:46:33Z'
  labels:
    application: broker
    template: amq-broker-74-statefulset-clustered
  name: broker-external-tcp
  namespace: amq-demo
  resourceVersion: '2450312'
  selfLink: /api/v1/namespaces/amq-demo/services/broker-amq-tcp
  uid: 52631fa0-ab9a-11e8-9380-c280f77be0d0
spec:
  externalTrafficPolicy: Cluster
  ports:
   - nodePort: 30001
     port: 61616
     protocol: TCP
     targetPort: 61616
  selector:
    deploymentConfig: broker-amq
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```



注記

NodePort 設定は重要です。NodePort はクライアントがブローカーにアクセスするポートで、タイプは **NodePort** *です。*

8.8.2. クライアントの接続

AMQ ブローカー CLI を使用して、クラスターのブローカーにラウンドロビンされるコンシューマーを作成します。

手順

1. *端末でコンシューマーを作成し、OpenShift が実行している IP アドレスにアタッチします。*

```
artemis consumer --url tcp://<IP_ADDRESS>:30001 --message-count 100 --destination
queue://demoQueue
```

2. *ステップ1 を 2 回繰り返して、2 つのコンシューマーを起動します。*

注記

これで、3 つのブローカー全体で 3 つのコンシューマー負荷が分散されます。

3. *メッセージを送信するプロデューサーを作成します。*

```
artemis producer --url tcp://<IP_ADDRESS>:30001 --message-count 300 --destination
queue://demoQueue
```

4. 各コンシューマーがメッセージを受信することを確認します。

```
Consumer:: filter = null
Consumer ActiveMQQueue[demoQueue], thread=0 wait until 100 messages are consumed
Consumer ActiveMQQueue[demoQueue], thread=0 Consumed: 100 messages
Consumer ActiveMQQueue[demoQueue], thread=0 Consumer thread finished
```

8.9. AMQ BROKER の監視

このチュートリアルでは、AMQ Broker のモニタリング方法について説明します。

前提条件

- このチュートリアルでは、 ブローカーの準備 に基づいて構築 されます。
- 基本的なブローカーのデプロイ のチュートリアルを完了することを推奨します。

手順

1. 実行中の Pod のリストを取得します。

```
$ oc get pods

NAME                READY     STATUS    RESTARTS   AGE
broker-amq-1-ftqmk   1/1       Running   0          14d
```

2. **oclogs**コマンドを実行します。

```
$ oc logs -f broker-amq-1-ftqmk

Running /amq-broker-71-openshift image, version 1.3-5
INFO: Loading '/opt/amq/bin/env'
INFO: Using java '/usr/lib/jvm/java-1.8.0/bin/java'
INFO: Starting in foreground, this is just for debugging purposes (stop process by pressing
CTRL+C)
...
INFO | Listening for connections at: tcp://broker-amq-1-ftqmk:61616?
maximumConnections=1000&wireFormat.maxFrameSize=104857600
INFO | Connector openwire started
INFO | Starting OpenShift discovery agent for service broker-amq-tcp transport type tcp
INFO | Network Connector DiscoveryNetworkConnector:NC:BrokerService[broker-amq-1-
ftqmk] started
INFO | Apache ActiveMQ 5.11.0.redhat-621084 (broker-amq-1-ftqmk, ID:broker-amq-1-
ftqmk-41433-1491445582960-0:1) started
INFO | For help or more information please see: http://activemq.apache.org
WARN | Store limit is 102400 mb (current store usage is 0 mb). The data directory:
/opt/amq/data/kahadb only has 9684 mb of usable space - resetting to maximum available
disk space: 9684 mb
WARN | Temporary Store limit is 51200 mb, whilst the temporary data directory:
/opt/amq/data/broker-amq-1-ftqmk/tmp_storage only has 9684 mb of usable space - resetting
to maximum available 9684 mb.
```

3. *クエリーを実行して、ブローカーの**Max Consumers**を監視します。*

```
$ curl -k -u admin:admin http://console-broker.amq-
demo.apps.example.com/console/jolokia/read/org.apache.activemq.artemis:broker=%22broker
%22,component=addresses,address=%22TESTQUEUE%22,subcomponent=queues,routing-
type=%22anycast%22,queue=%22TESTQUEUE%22/MaxConsumers

{"request":
{"mbean":"org.apache.activemq.artemis:address=\"TESTQUEUE\",broker=\"broker\",compone
nt=addresses,queue=\"TESTQUEUE\",routing-
type=\"anycast\",subcomponent=queues","attribute":"MaxConsumers","type":"read"},"value":-
1,"timestamp":1528297825,"status":200}
```

第9章 リファレンス

9.1. カスタムリソース定義設定リファレンス

カスタムリソース定義(CRD)は、Operator でデプロイされたカスタム OpenShift オブジェクトの変更可能な設定項目のスキーマです。付随するカスタムリソース(CR)ファイルを使用すると、CRD の設定アイテムの値を指定できます。

以下のサブセクションでは、ブローカーおよびアドレス指定CRD で利用可能な設定項目について詳しく説明します。

9.1.1. ブローカー CRD 設定リファレンス

ブローカーカスタムリソース定義(CRD)を使用すると、OpenShift プロジェクトでデプロイメントのブローカーを設定できます。次の表で、設定できる項目の詳細を示します。

重要

*アスタリスク(*)でマークされた設定アイテムは、該当するカスタムリソース(CR)でデプロイに必要です。不要なアイテムの値を明示的に指定しない場合には、設定にデフォルト値が使用されます。*

エントリー	サブエントリー	タイプ	例	デフォルト値	説明
adminUser*		string	my_user	無作為に自動生成された値	ブローカーおよび管理コンソールに接続するために必要なパスワード。 値を指定しない場合、値は自動的に生成され、シークレットに保存されます。デフォルトのシークレット名の形式は <**Custom Resource name>-credentials-secret** です。例：**ex-aao-credentials-secret**

エントリー	サブエントリー	タイプ	例	デフォルト値	説明
adminPassword*		string	my_password	無作為に自動生成された値	ブローカーおよび管理コンソールに接続するために必要なパスワード。 値を指定しない場合、値は自動的に生成され、シークレットに保存されます。デフォルトのシークレット名の形式は <**Custom Resource name>-credentials-secret** です。例：**ex-aao-credentials-secret**
deploymentPlan*					ブローカーのデプロイメント設定
	image*	string	registry.redhat.io/amq7/amq-broker:latest	registry.redhat.io/amq7/amq-broker:7.5	Red Hat Container Registry からプルするブローカーコンテナーイメージの URL。default タグはブローカー Operator のバージョンと一致します。

エントリー	サブエントリー	タイプ	例	デフォルト値	説明
	size*	int	2	2	デプロイメントで作成するブローカー Pod の数。 **2** 以上の値を指定すると、ブローカーのデプロイメントはデフォルトでクラスター化されます。クラスターのユーザー名とパスワードは自動的に生成され、同じシークレットに保存されます (デフォルトで**admin User**および**admin Password**)。
	requireLogin	Boolean	true	true	ブローカーに接続するためにログイン認証情報が必要であるかどうかを指定します。
	persistenceEnabled	Boolean	false	true	永続ボリューム要求(PVC)で作成された永続ボリューム(PV)経由でジャーナルストレージを使用するかどうかを指定します。
	journalType	string	AIO	AIO	非同期 I/O (AIO)と非ブロッキング I/O(NIO)のどちらを使用するかを指定します。
	messageMigration	Boolean	true	true	ブローカーのスケールダウン時にメッセージを移行するかどうかを指定します。

エントリー	サブエントリー	タイプ	例	デフォルト値	説明
acceptors.acceptor		object			単一のアクセプターの設定インスタンス。
	name*	string	my_acceptor	未指定	アクセプターの名前。
	port	int	5672	定義する最初のアクセプターは 61626 です。デフォルト値は、定義する後続のアクセプターごとに 10 ずつ増加します。	アクセプターインスタンスに使用されるポート番号。
	protocols	string	amqp,core	all	アクセプターインスタンスで有効にするメッセージングプロトコル。
	sslEnabled	Boolean	false	false	アクセプターポートで SSL が有効であるかどうかを指定します。**true** に設定した場合は、アクセプターで SSL を有効にするために必要なデータがないか、シークレットを探します。

エントリー	サブエントリー	タイプ	例	デフォルト値	説明
	sslSecret	string	ex-aao-my_acceptor-secret	未指定	クライアントトラストストアおよびブローカーキーストア（base64 でエンコードされた）および **keyStorePassword** および **trustStorePassword**（エンコードされていない）が保存されるシークレット。**sslSecret** の値を指定しない場合、アクセプターはデフォルトのシークレットを使用します。デフォルトのシークレット名の形式は **<Custom Resource name>-<acceptor name>-secret** です。
	enabledCipherSuites	string	SSL_RSA_WITH_RC4_128_SHA,SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	未指定	SSL 通信に使用する暗号スイートのコンマ区切りリスト。
	enabledProtocols	string	TLSv1,TLSv1.1,TLSv1.2	未指定	SSL 通信に使用するプロトコルのコンマ区切りリスト。

エントリー	サブエントリー	タイプ	例	デフォルト値	説明
	needClientAuth	Boolean	true	未指定	アクセプターで双方向 SSL が必要であることをブローカーがクライアントに通知するかどうかを指定します。このプロパティーは、**wantClientAuth**をオーバーライドします。
	wantClientAuth	Boolean	true	未指定	アクセプターで双方向 SSL が要求されているが、必須ではないことをブローカーがクライアントに伝えるかどうかを指定します。**needClientAuth** で上書きされます。
	verifyHost	Boolean	true	未指定	クライアントの SSL 証明書の Common Name(CN)をホスト名と比較して、それらが一致することを確認するかどうかを指定します。このオプションは、双方向 SSL が使用される場合にのみ適用されます。
	sslProvider	string	JDK	JDK	SSL プロバイダーが JDK か OPENSSL のいずれかを指定します。

エントリー	サブエントリー	タイプ	例	デフォルト値	説明
	sniHost	string	some_regular_expression	未指定	受信 SSL 接続の **server_name** 拡張と照合する正規表現。名前が一致しない場合には、アクセプターへの接続は拒否されます。
	expose	Boolean	true	false	OpenShift Container Platform の外部でアクセプターを公開するかどうかを指定します。
	anycastPrefix	string	jms.topic.	未指定	**エニーキャスト**のルーティングタイプを使用するようクライアントによって使用される接頭辞。
	multicastPrefix	string	/queue/	未指定	**マルチキャスト**ルーティングタイプを使用する必要があることを指定するためにクライアントによって使用されるプレフィックス。
	connectionsAllowed	integer	2	0	アクセプターで許可される接続の数。この制限に達すると、**DEBUG** メッセージがログに出力され、接続は拒否されます。使用中のクライアントのタイプによって、接続が拒否されたときに何が起こるかが決まります。
connectors.connector		object			単一のコネクター構成インスタンス。

エントリー	サブエントリー	タイプ	例	デフォルト値	説明
	name*	string	my_connector	該当なし	コネクターの名前
	type	string	tcp	tcp	作成するコネクターのタイプ、**tcp**、または**vm**。
	host*	string	localhost	未指定	接続するホスト名または IP アドレス。
	port*	int	22222	未指定	コネクターインスタンスに使用されるポート番号。
	sslEnabled	Boolean	false	false	コネクターポートで SSL が有効であるかどうかを指定します。**true** に設定した場合は、コネクターで SSL を有効にするために必要なデータのシークレットを探します。

エントリー	サブエントリー	タイプ	例	デフォルト値	説明
	sslSecret	string	ex-aao-my_connector-secret	未指定	クライアントトラストストアおよびブローカーキーストア（base64 でエンコードされた）および **keyStorePassword** および **trustStorePassword**（エンコードされていない）が保存されるシークレット。**sslSecret** の値を指定しない場合、コネクターはデフォルトのシークレットを使用します。デフォルトのシークレット名の形式は **<Custom Resource name>-<connector name>-secret** です。
	enabledCipherSuites	string	SSL_RSA_WITH_RC4_128_SHA,SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	未指定	SSL 通信に使用する暗号スイートのコンマ区切りリスト。
	enabledProtocols	string	TLSv1,TLSv1.1,TLSv1.2	未指定	SSL 通信に使用するプロトコルのコンマ区切りリスト。

エントリー	サブエントリー	タイプ	例	デフォルト値	説明
	needClientAuth	Boolean	true	未指定	コネクターで双方向 SSL が必要であることをブローカーがクライアントに通知するかどうかを指定します。このプロパティーは、**wantClientAuth**をオーバーライドします。
	wantClientAuth	Boolean	true	未指定	コネクターで双方向 SSL が要求されているが、必須ではないことをブローカーがクライアントに通知するかどうかを指定します。**needClientAuth** で上書きされます。
	verifyHost	Boolean	true	未指定	クライアントの SSL 証明書の Common Name(CN)をホスト名と比較して、それらが一致することを確認するかどうかを指定します。このオプションは、双方向 SSL が使用される場合にのみ適用されます。
	sslProvider	string	JDK	JDK	SSL プロバイダーが **JDK** か **OPENSSL** のいずれかを指定します。

エントリー	サブエントリー	タイプ	例	デフォルト値	説明
	sniHost	string	some_regular_expression	未指定	SSL 接続の **server_name** 拡張と照合する正規表現。名前が一致しない場合には、コネクター接続は拒否されます。
	expose	Boolean	true	false	OpenShift Container Platform 外でコネクターを公開するかどうかを指定します。
console					ブローカー管理コンソールの設定。
	expose	Boolean	true	false	管理コンソールポートを公開するかどうかを指定します。
	sslEnabled	Boolean	true	false	管理コンソールポートで SSL を使用するかどうかを指定します。

エントリー	サブエントリー	タイプ	例	デフォルト値	説明
	sslSecret	string	ex-aao-my_console-secret	未指定	クライアントトラストストアおよびブローカーキーストア（base64 でエンコードされた）および **keyStorePassword** および **trustStorePassword**（エンコードされていない）が保存されるシークレット。**sslSecret** の値を指定しない場合、コンソールはデフォルトのシークレットを使用します。デフォルトのシークレット名の形式は <**Custom Resource name>-console-secret** です。
	useClientAuth	Boolean	true	false	管理コンソールにクライアント認証が必要かどうかを指定します。

9.1.2. アドレス指定 CRD 設定リファレンス

アドレス指定カスタムリソース定義(CRD)により、ブローカーで作成されるアドレスおよびキューおよび関連するルーティングタイプを定義できます。次の表で、設定できる項目の詳細を示します。

重要

*アスタリスク(*)でマークされた設定アイテムは、該当するカスタムリソース(CR)でデプロイに必要です。不要なアイテムの値を明示的に指定しない場合には、設定にデフォルト値が使用されます。*

エントリー	タイプ	例	デフォルト値	説明
addressName*	string	address0	未指定	ブローカーで作成されるアドレス名。

エントリー	タイプ	例	デフォルト値	説明
queueName*	string	queue0	未指定	ブローカーで作成されるキュー名。
routingType*	string	anycast	未指定	使用するルーティングタイプ（エニーキャストまたはマルチキャスト）。

9.2. アプリケーションテンプレートパラメーター

OpenShift Container Platform イメージでの AMQ Broker の設定は、アプリケーションテンプレートパラメーターの値を指定して実行されます。次のパラメーターを設定できます。

表9.1 アプリケーションテンプレートパラメーター

パラメーター	説明
AMQ_ADDRESSES	起動時にブローカーでデフォルトで使用可能なアドレスを、コンマ区切りのリストで指定します。
AMQ_ANYCAST_PREFIX	多重化プロトコルポート 61616 および 61617 に適用される anycast プレフィックスを指定します。
AMQ_CLUSTERED	クラスタリングを有効にします。
AMQ_CLUSTER_PASSWORD	クラスタリングに使用するパスワードを指定します。値の指定がない場合は、無作為にパスワードが生成されます。
AMQ_CLUSTER_USER	クラスタリングに使用するクラスターユーザーを指定します。値の指定がない場合は、ランダムなユーザー名が生成されます。
AMQ_DATA_DIR	データのディレクトリーを指定します。ステートフルセットで使用されます。
AMQ_DATA_DIR_LOGGING	データディレクトリーロギング用のディレクトリーを指定します。
AMQ_EXTRA_ARGS	**artemiscreate** に渡す追加の引数を指定します。
AMQ_GLOBAL_MAX_SIZE	メッセージデータが使用可能な最大メモリー量を指定します。値が指定されていない場合は、システムのメモリーの半分が割り当てられます。
AMQ_KEYSTORE	SSL キーストアファイル名を指定します。値が指定されていない場合に、パスワードが無作為に生成されますが、SSL は構成されません。

パラメーター	説明
AMQ_KEYSTORE_PASSWORD	(オプション) SSL キーストアの復号化に使用するパスワードを指定します。
AMQ_KEYSTORE_TRUSTSTORE_DIR	シークレットがマウントされるディレクトリーを指定します。デフォルト値は **/etc/amq-secret-volume** です。
AMQ_MAX_CONNECTIONS	SSL の場合のみ、アクセプターが受け入れる接続の最大数を指定します。
AMQ_MULTICAST_PREFIX	多重化プロトコルポート 61616 および 61617 に適用される multicast プレフィックスを指定します。
AMQ_NAME	ブローカーインスタンスの名前を指定します。
AMQ_PASSWORD	ブローカーへの認証に使用するパスワードを指定します。値の指定がない場合は、無作為にパスワードが生成されます。
AMQ_PROTOCOL	ブローカーが使用するメッセージングプロトコルをコンマ区切りのリストで指定します。使用可能なオプションは、**amqp**、**mqtt**、**openwire**、**stomp**、および**hornetq**です。何も指定されていない場合は、全プロトコルを使用できます。イメージを Red Hat JBoss Enterprise Application Platform と統合するには、OpenWire プロトコルを指定する必要がありますが、他のプロトコルもオプションで指定できます。
AMQ_QUEUES	起動時にブローカーでデフォルトで使用可能なキューを、コンマ区切りのリストで指定します。
AMQ_REQUIRE_LOGIN	**true**に設定すると、ログインが必要になります。指定しない場合、または**false**に設定した場合、匿名アクセスを使用できます。デフォルトでは、このパラメーターの値は指定されていません。
AMQ_ROLE	作成されたロールの名前を指定します。デフォルト値は **amq** です。
AMQ_TRUSTSTORE	SSL トラストストアのファイル名を指定します。値が指定されていない場合に、パスワードが無作為に生成されますが、SSL は構成されません。
AMQ_TRUSTSTORE_PASSWORD	(オプション) SSL トラストストアの復号化に使用されるパスワードを指定します。

パラメーター	説明
AMQ_USER	ブローカーへの認証に使用されるユーザー名を指定します。値の指定がない場合は、ランダムなユーザー名が生成されます。
APPLICATION_NAME	OpenShift 内で使用されるアプリケーションの名前を指定します。これは、アプリケーション内のサービス、Pod、およびその他のオブジェクトの名前で使用されます。
IMAGE	イメージを指定します。**永続性**、**persistent-ssl**、および**statefulset-clustered**テンプレートで使用されます。
IMAGE_STREAM_NAMESPACE	イメージストリームの名前空間を指定します。**ssl** および **basic**テンプレートで使用されます。
OPENSHIFT_DNS_PING_SERVICE_PORT	OpenShift DNS ping のポート番号を指定します。
VOLUME_CAPACITY	データベースボリュームの永続ストレージのサイズを指定します。

注記

*カスタム設定に**broker.xml**を使用する場合に、そのファイルで次のパラメーターに指定された値は、アプリケーションテンプレートで同じパラメーターに指定された値をオーバーライドします。*

- *AMQ_NAME*
- *AMQ_ROLE*
- *AMQ_CLUSTER_USER*
- *AMQ_CLUSTER_PASSWORD*

9.3. ロギング

OpenShift ログの表示に加えて、コンテナーのコンソールに出力される AMQ ログを表示することにより、OpenShift Container Platform イメージで実行中の AMQ Broker のトラブルシューティングを行うことができます。

手順

- *コマンドラインで、次のコマンドを実行します。*

```
$ oc logs -f <pass:quotes[<pod-name>]> <pass:quotes[<container-name>]>
```

改訂日時： 2022-09-08 15:59:03 +1000